Autonomous Pedestrian Detection in Transit Buses

WPI

A Major Qualifying Project Report Submitted to the Faculty of

Worcester Polytechnic Institute

By:

Thomas Fong
Alima Kargbo
Franc Luga
Dario Martinovic
Michael Milliard
Michael Padberg
Kazim Hyder Nizam Shaikh
Abdelrahman Sirry
Patrick Trant

March 13, 2018

Project Code: MQP-AW1-BUS1

Advisor:
Professor Alexander Wyglinksi

Sponsor:
Transdev North America

Abstract

This project created a proof of concept for an automated pedestrian detection and avoidance system designed for transit buses. The system detects objects up to 12 meters away, calculates the distance from the system using a solid-state LIDAR, and determines if that object is human by passive infrared. This triggers a visual and sound warning. A Xilinx Zynq-SoC utilizing programmable logic and an ARM-based processing system drive data fusion, and an external power unit makes it configurable for transit-buses.

Acknowledgements

This project would not have been capable without the support and guidance received from so many individuals. Transdev, LeddarTech, and Worcester Polytechnic Institute all helped to bring this project to life. Professor Wyglinski, the advisor and the heart and soul of the team, was instrumental in pushing this project forward and driving it to be a success. Specific guidance from Professor Duckworth and Professor Emanuel helped create solutions when the team had reached impasses, and both deserve the utmost of thanks. The team is incredibly grateful for everyone who helped to make this project a possibility.

Table of Contents

Table of Authorship

List of Figures

List of Tables

Executive Summary

Automation within the automobile industry is becoming an increasingly growing area of consideration for manufacturers. Automated functionality within motor vehicles can help to prevent accidents, alert drivers, and regulate the speed at which an individual is traveling. This is something being sought after not just in personal transportation, but in public transportation as well. This project seeks to provide the sensor foundation for automated functionality for transit city buses, specifically within the realm of accident avoidance with pedestrians. The initial proof of concept requires that a system be created that is able to accurately detect an object and its distance from the system, as well as determine if that object is a human being. This prototype seeks to notify the driver so that they can make accurate adjustments with the bus to avoid a potential accident. A final product would have the capability of taking over control of the bus if it reaches a critical point. The project, as proof of concept, focused on the area of greatest accidents which was when a bus would turn left, as this is when a pedestrian is most likely to enter a blind spot.

A block diagram of the overall system can be seen in Figure 1. This system was implemented through the use of an Avnet Zedboard that utilized a Xilinx Zynq 7020 All Programmable System on Chip (SoC). This SoC contains functionality consisting of dual-core ARM Cortex A9 processors as well as a Xilinx Artix 7 Field Programmable Gate Array (FPGA) fabric. This functionality allows for programmable logic, as well as embedded software, which provided flexibility and robustness during the design process. The programmable logic was utilized for interfacing with various sensors, storing received data in memory, and ultimately passing that data to the processing system. The logic is capable of running in parallel, thusly increasing efficiency. Once data is passed to the processing system, the incoming information is then used to interpret the device's surroundings and ultimately drive the output of the alert system, depending on the conditions detected by the combination of the individual sensors. The sensors communicated with the Avnet Zedboard through its digital Peripheral Module (PMOD) I/O Ports.

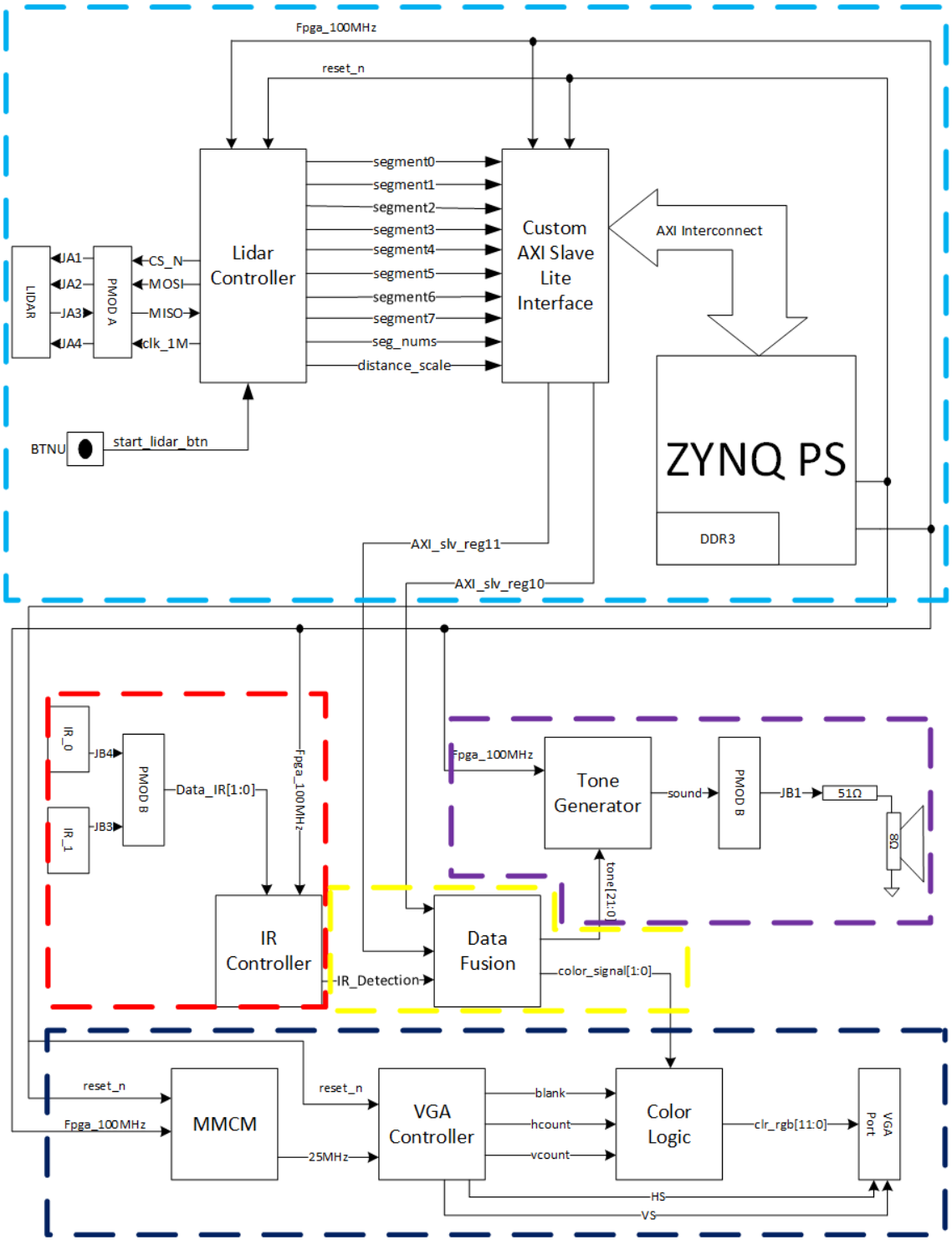

Figure 1: System Hardware Block Diagram. This shows the complete hardware logic block design with its corresponding modules and connections. Each module provides unique functionality that makes the top-level design possible. LIDAR, IR, Data Fusion, Audible Alarm and Visual Alarm logic implementations are outlined in light blue, red, yellow, purple and dark blue, respectively.

The sensors utilized for the final proof of concept consisted of a LeddarVu8 solid state LIDAR, and a Panasonic Electric Works EKMC Series Long Distance Detection Type passive infrared (PIR). These sensors provided object detection and the ability to determine if that object was a human. With the incoming data from the sensors acquired, a warning system was implemented that transitioned between green, yellow, and red displays on a VGA screen along with an audible buzzer. The displayed colors corresponded to the level of threat that an object posed with green indicating no threat, yellow that an object was detected and could potentially be a threat, and red indicating that an object was detected at a distance that requires immediate action to avoid an accident, and that the object was determined to be human. The buzzer provided an additional alert for when the VGA went red to further reiterate the need for immediate action.

The original electrical requirement for all of the devices required a power supply that would output 12 volts and at least 5.17 amps for an output of roughly 56 watts, but this changed as components were removed and added. The new power requirement came out to be 12 volts and at least 6.667 amps for an output of roughly 80 watts. The PMC-12V100W1A from Delta Electronics was used, which converted 120 volts AC to 12 volts DC. It also provided up to 8 amps that satisfied the system and all of the modifications made. The power supply had two output rails, one fed the cooling fan and the other the LIDAR, warning screen and Zedboard. A junction box was used to house the connections of the LIDAR, warning screen and Zedboard to the power supply; it also housed a power switch for the LIDAR. The PIR sensors and buzzer received sufficient power from the Zedboard's 3.3V PMOD connectors.

To effectively test this system, a stationary bus model was used. This model was a wooden structure that held the casing for the system as well as the power supply, which was wired and then fixed to the structure. A testing path was taped on the ground that simulated the relative location of a pedestrian in relation to the "bus" as if it was turning. A picture of the final design mounted to the structure, as well as a picture of the pedestrian model testing session may be seen in Figure 2. Table 1 shows a table of results from this test, where the distances at which the pedestrian was detected in both the danger warning zone (Yellow) and critical action zone (Red) were recorded for each trial.

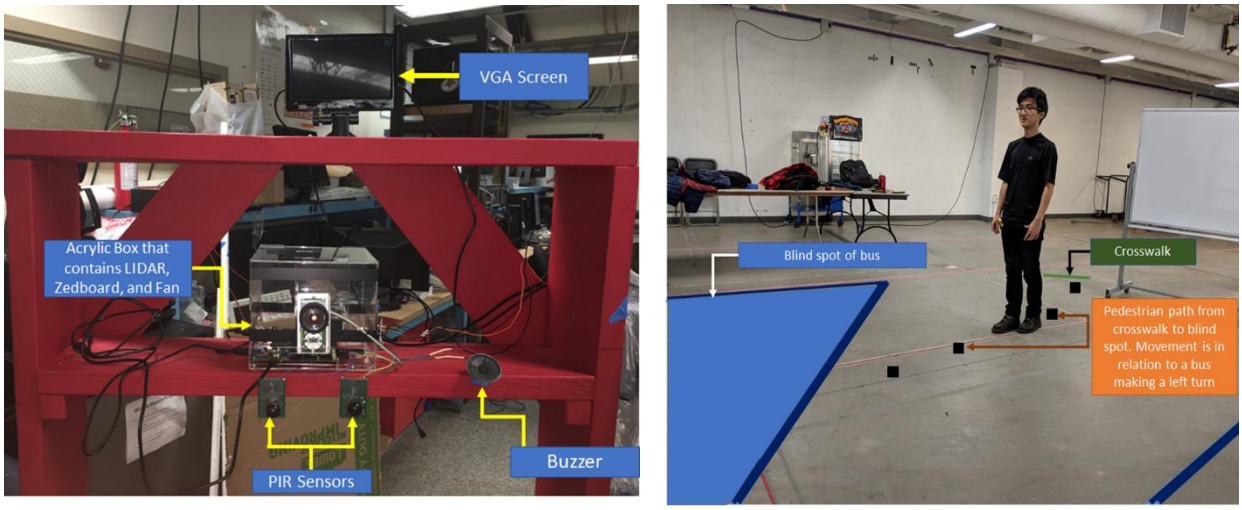

Figure 2: System Mounted to Wooden Structure, Reverse Movement Test. The full system incorporated a LIDAR-based ranging system for object detection and two passive infrared sensors to verify that a detected object was a pedestrian by checking if the subject's temperature was relatively close to human body temperature. A VGA screen and buzzer, which would be mounted to the dashboard of a city bus, are used to provide warning to the driver. The reverse movement test was performed by a test subject moving into the blind spot upon a defined path to mimic the turning of a city bus.

Table 1: Test Results for Pedestrian Following Simulated Bus Turn Path. The data was found to be very consistent across the trials, indicating that the system was able to detect a pedestrian entering each zone with a sufficient reaction time.

Trial	Distance When Yellow	Distance When Red
1	3.435	2.313
2	3.558	2.22
3	3.715	2.388
4	3.695	2.424
5	3.623	2.435
6	3.676	2.352
7	3.443	2.278
8	3.616	2.262
9	3.752	2.29
10	3.673	2.226
TOTAL AVERAGE	3.6186	2.3188
STANDARD DEVIATION	0.10925	0.07787

The consistency of the detection distances recorded from the system testing indicates that this system is able to effectively detect pedestrians in each zone with a sufficient reaction time. For safety and liability purposes, the team was unable to utilize an actual bus for testing. Tests were completed for the PIR sensor and LIDAR sensor individually, for the system as a whole, and a simulation of the bus turning. This ensured everything worked as intended.

The proof of concept showed that the sensor application was viable for detecting an object, determining if that object was human, and providing an effective visual and audible alert in real time. The next steps in completing this system would require full implementation with a bus, as well as adding an accelerometer and temperature sensor to the design. This would make the system more effective and provide more information that can be utilized in the decision making process. These steps would be critical in making the system capable of having full automated control over the bus.

1: Introduction

In 2014, there were 6,064,000 accidents on roadways in the United States. 32,675 fatalities occurred from those accidents, according to the National Highway Traffic Safety Administration [1]. Of those accidents, there were approximately 69,000 involving buses. Approximately 1,088 of the 69,000 accidents involved a collision with a non-occupant of the bus: a pedestrian. 88 of the accidents showed to be fatal for the pedestrians who were hit, while the rest suffered injuries [1]. Bus accidents involving pedestrians are rare compared to other types of automobile traffic accidents, but when they do occur, the aftermath can be quite gruesome.

Blind spots are the primary reason for buses striking pedestrians. One of the most common blind spots on buses is the one on the front left corner of the bus. Blind spots exist in the areas where drivers are unable to see outside the bus, and in total can encompass 15 to 19 pedestrians located within all of them [2] [3]. The mirror placement and the position of the bus frame is often the cause of this particular left-side blind spot [4]. An approximated example of this blind spot, relative to a 40-foot bus, can be found in Figure 3:

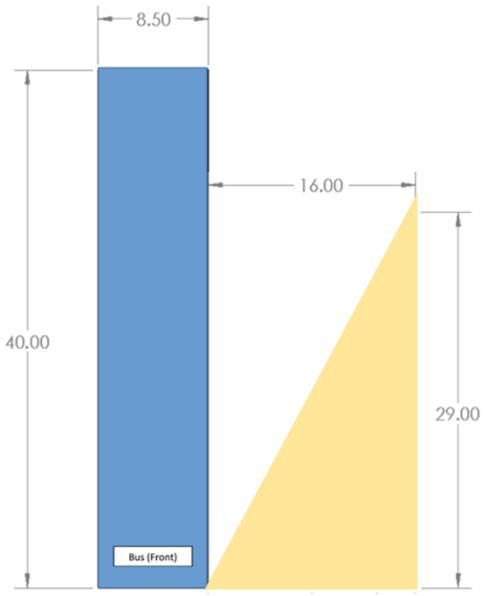

Figure 3: Approximated Left Side Blind Spot of 40-Foot Bus. This estimate ranges from up to 16 feet perpindicularly from the driver's side window and 29 feet back from the front of the bus. The blind spot is shown in yellow to the side of the bus (blue).

Another common blind spot is to the right of the bus driver. Tall fare boxes or other equipment can obstruct the view of the bus driver [4]. Due to these blind spots, many pedestrians were struck while they were in a side street crosswalk. There are some extreme cases where a pedestrian is struck and the bus driver does not realize it. Around two years ago, a bus driver struck a female pedestrian while turning and did not notice until other pedestrians stopped him. The woman was crushed by the back wheels of the vehicle. Unfortunately, she died at the scene [5]. The problem with blind spots on buses is nothing new, it has been known for some time. In 2008 the National Academy of Sciences noticed the problems and concluded that these blind spots were not present with buses built before 1970 [4].

Driving a transit bus is often hard enough without the blind spot. Transit bus drivers have to handle more than just a 30,000 pound vehicle. The driver has to focus on maneuvering around various obstacles, manage riders, keep track of bus fares, stay on schedule, stay aware of other vehicles around them, and most importantly keep passengers safe. The rise of smartphones among other similar products keep some pedestrians, cyclists, and other drivers around them distracted, which makes the bus driver's job more difficult [6]. Occasionally, major traffic engineering jobs take place without the collaboration of transit authorities. A prime example of this would be the addition of bike lanes. Bike lanes may interfere with buses pulling into their stops [6]. Potential issues of this sort can lead to accidents, as they showcase the danger presented by a bus to the surrounding pedestrians. The focus of this project was to develop a system that could aid in reducing bus collisions with pedestrians, such as the terrible accidents described.

1.1: Technical Challenges of Pedestrian Detection

Detecting a human subject is something that has been accomplished through numerous other projects. However, the challenges this project faced in terms of pedestrian detection mostly stem from the necessity of a real-time application and decision making. Seconds can be the difference between an accident and an avoidance, so the need for a pedestrian detection system that can not only detect an individual, but also make split-second decisions was necessary. The rapid movement of a bus performing a left turn, for example, is a frequent cause of pedestrian accidents [7]. This required system that was properly calibrated to avoid false detections, and handle the environmental factors that could affect the effectiveness of the system [8]. It also included distinguishing between an inanimate object, a human, and non-human subjects,

including pets. To accomplish this distinction, a central system that can fuse the incoming data from multiple sensors was needed. The overall design had to be able to acquire distance readings from the object, read in object temperatures to determine if the object is human, measure the traveling speed and motion of the bus, and effectively power all components while maintaining a reasonable operating temperature. This required the integration of multiple sensors and components to effectively bring all factors together into a single solution.

The main focus of this project was reducing the amount of pedestrian fatalities involving transit bus accidents. Annually there are approximately 30 to 60 pedestrian fatalities involving transit buses [6]. A singular pedestrian fatality due to a bus accident is one too many. Studies have shown that the number of pedestrians is increasing, especially in areas of high population density such as New York City, which may increase the chances of more fatalities if corrective action is not taken. This project aimed to create a system that can effectively detect pedestrians surrounding a bus, and take necessary corrective actions that will help prevent an accident from occurring.

1.2: Project Objectives and Contributions

This project sought to accomplish and provide a set of deliverables. This included a proof of concept system that was able to:

1. Detect an object and the distance of that object in relation to the system
2. Determine if that object is a human or not
3. Provide an effective warning system to alert the driver
4. Utilize an effective power system designed to support all components
5. Establish a test environment that is able to simulate the turning of a transit bus

The efforts towards completing this project resulted in a system that could effectively detect an object at a distance, determine if the object is human, and alert the operator in a critical scenario. The contributions that came along with this proof of concept were extensive.

The use of a Zedboard in conjunction with the LeddarVu8 was accomplished for the implementation of object detection and ranging. The team was able to create custom controllers using the Verilog hardware description language to acquire data from a LIDAR sensor using a

Field Programmable Gate Array (FPGA). This functionality provided a means of detecting the distance of an object. This design was further improved using the Zedboard to fuse incoming data from passive infrared sensors to determine if the detected object was a human subject. The Zedboard was also programmed to implement a custom bidirectional Advanced Extensible Interface bus, capable of transferring data between the programmable logic and processing system. This allowed for the creation of a system design utilizing both hardware and software programming to maximize the efficiency and capability of the data fusion and processing. The logic design incorporated a VGA controller and a tone generator that created an effective two tiered warning system.

In order to test the effectiveness of the overall design, the team needed to be able to simulate the turning of a city bus, without access to a physical bus. Research of potential solutions resulted in the creation of a stationary bus model centered around the sensor system. Once assembled, the full system was attached to a portable wooden structure. This created a mobile proof of concept. The structure incorporated a casing to house the components, with ventilation and a cooling fan to ensure proper heat dissipation. An AC-sourced 12 volt DC power system was also mounted to the structure with careful wiring to ensure safe practice and usage. The power system was designed to provide the needed voltage and current specifications to all components, sourced by a single wall outlet connection.

Overall, the contributions towards this project created a proof of concept for pedestrian detection, and established a strong foundation that can be built upon to ultimately provide a fully autonomous pedestrian avoidance solution for transit buses.

1.3: Project Organization and Timeline

In order to effectively manage the various tasks, the team was separated into different functional focuses. The overall design goal was broken down into 5 main concentrations: Zedboard and LIDAR Integration, Bus Structure Design, PIR and Alert System Implementation, Human Factors Research, and Power System Implementation. Michael Milliard and Michael Padberg served as the team leads, organizing the overall project with a functional lead for each of the activities. Dario Martinovic and Michael Padberg led the Zedboard and the LIDAR integration, Kazim Hyder Nizam Shaikh led the IR and Alert System Implementation, Alima Kargbo and Kazim Hyder Nizam Shaikh co-led the bus structure design and implementation,

Patrick Trant led the Human Factors Research, and Thomas Fong and Franc Luga co-led the Power System Implementation. Figure 4 details the breakdown of the project operations by main activities.

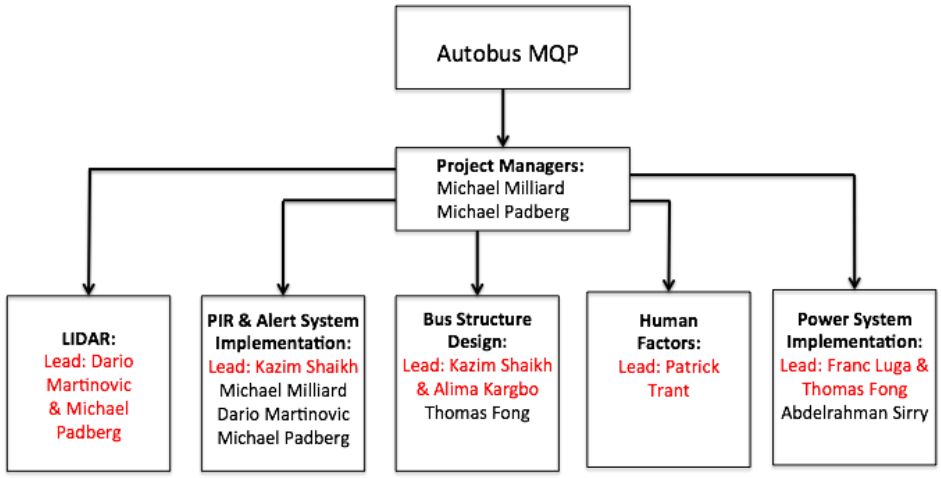

Figure 4: High Level Activity Breakdown Detailing the Contributions and Organization for Tasks of the Project. The individuals leading the completion of each task are shown in red.

The project also utilized a high level Gantt chart to manage the timeline for the project. As issues arose, there were deviations from this schedule to readjust priorities with the critical aspects of the project. This high level aspect of allowed the team to see all the necessary actions that would go towards the final design. The Gantt chart can be seen in Figure 5.

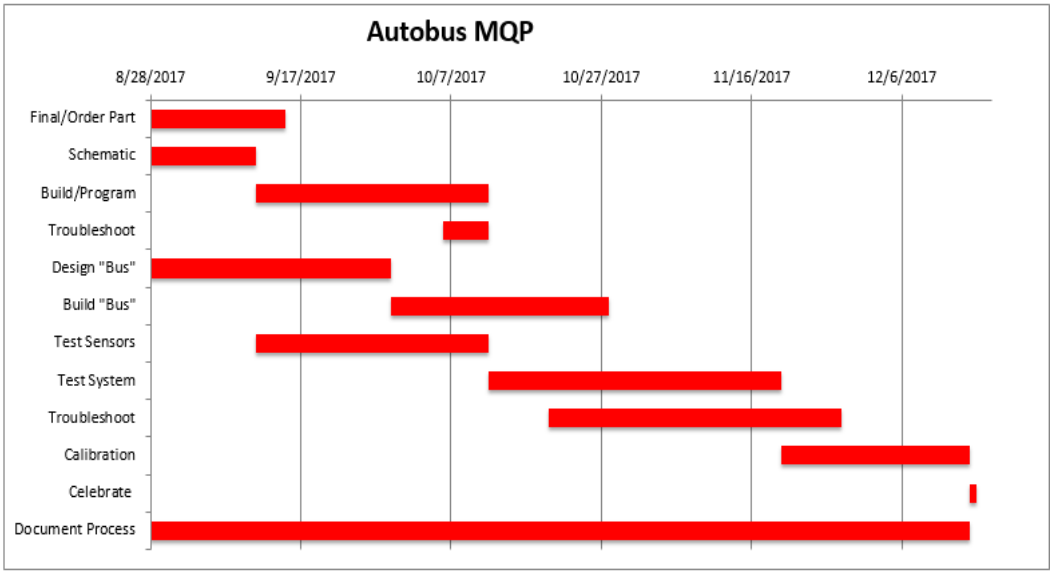

Figure 5: Gantt Chart Detailing the Higher Level Planned Implementation of the Project Over the Course of Academic Terms. Each red bar represents the duration of each task.

1.4: Report Organization

The remainder of this technical report is separated into another 8 chapters, with each chapter pertaining to a necessary step in the project.

Chapter 2 is the background section. This details the research conducted during the proposal phase of the project. With all of the different aspects and considerations that ultimately went into the proof of concept system, it was necessary to have a well-versed foundation and understanding of those considerations.

Chapter 3 covers the considerations made in determining which approach would be the most effective implementation of the project. This chapter covers the necessary pros and cons of each potential component that would be used. Chapter 4 follows a similar structure to Chapter 3, however, it incorporates many human factors on top of the technological aspects in regard to the proactive driver alert and warning capabilities of the system. The information discussed in Chapters 3 and 4 directly tie into Chapter 5, which covers each of the proposed solutions and sensor integration considered in the project's design.

Chapter 6 details the single proposed solution based on the discussion in the three prior chapters, and then discusses what the final choice was for each part of the system design. Chapter 7 shows the methodology and implementation, which details the process the team took in accomplishing each piece of the project. The results of the final testing are shown in Chapter 8, and Chapter 9 discusses the conclusions drawn from them. Chapter 9 also discusses recommendations for the future work, and what could be built upon from the team's proof of concept design. Following this final chapter, there are various appendices, including the code design utilized in the operation of the full system.

2: Background Research and Considerations

This project required background research that explored the many factors that will ultimately tie into the final product. This included research into embedded technology, multiple sensors, the considerations for transit buses and the bus drivers, and previous implementations of pedestrian detection.

2.1: The Bus Factor

The bus factor required that there was a thorough understanding of the effect that the bus and driver would have on different considerations for the project. Understanding this provided a greater insight into the facets of the project that were external to the system.

Transit buses require more maintenance than motor vehicles. Maintenance costs are dependent on multiple variables including vehicle age, duty cycle, topography, fleet maintenance practices and several other factors [9]. Over time, the per mile operating and maintenance costs tend to increase and bus replacement becomes more cost effective. The lifespan of a 40 foot, heavy duty bus can extend as long as 12 years [9]. This indicates the need for reliable systems that can provide the greatest efficiency in tandem with the bus. Operating systems such as the frame, body, electrical and other major components are designed to last in the 12 year time frame as well [9]. As buses decrease in size, the minimum life of the operating bus decreases. This is shown in Table 2.

Table 2: Minimum Life Cycle Cost Replacement Ages & Mileages by Service-life. Listed large bus sizes to smaller bus sizes. Buses smaller in size have a lower minimum life of operation. [10]

Vehicle Type/Category	Annual Vehicle Mileage	Minimum Cost Age	Mileage
Heavy-Duty Large Bus: 12-Years/500,000 Miles	25,000 35,000 45,000	17 14 12	425,000 490,000 540,000
Heavy-Duty Large Bus: 10-Years/350,000 Miles	25,000 35,000 45,000	12 11 11	300,000 385,000 495,000
Medium-Duty Small Bus: 7-Years/200,000 Miles	25,000 35,000 45,000	9 8 7	225,000 280,000 315,000
Light-Duty Midsize Bus/Van: 5-Years/150,000 Miles	20,000 30,000 40,000	7 6 5	140,000 180,000 200,000
Light-Duty Small Bus/Van: 4-Years/100,000 Miles	20,000 30,000 40,000	6 5 4	120,000 150,000 160,000

Regular buses are essential in cities, but have a significant cost to repair and maintain. Companies such as Transdev provide top quality transportation that focuses on safety, efficiency and cost-effective results [11]. Specifically in Nassau County, there has been a 23% reduction in hourly operating cost over previous operators such as New York City's Metropolitan Transportation Authority (MTA) [11]. Future electrical improvements to the Transdev bus models will increase the safety of passengers inside and outside of transit buses, and can help to reduce the frequency of accidents. Reducing accidents will mitigate repair costs due to any damages, and will eliminate the downtime of the bus. This increases the overall cost effectiveness of the bus, and prevents an unnecessary expense to the bus company.

Nearly everyone has had the opportunity to ride in a bus at least once in their lives. Most children will ride a school bus to attend schooling, and adults will take buses for long distances to visit family or to get to work in the morning. In cities and towns, bus routes are defined to transport passengers to major areas of public use. Buses are known to keep a strict run time as well as having defined routes. What is often forgotten is that the individual who is operating the vehicle has upwards of 60 passengers onboard [12]. There are two main types of bus drivers, one being for school buses and the other being for commercial buses. There are certain procedures to follow if one wants to pursue a career as a bus driver. For school bus drivers, there a few minimum requirements from both the state and the school itself. The applicant needs to obtain an "S" type drivers' license to be able to drive a school bus. This entails the applicant going to their local Department for Motor Vehicles (DMV). The applicant will take their initial written test and, upon passing, will return to complete their road test. The certain requirements are, a written test, road skill test, physical test, pass a criminal background check and pay the required fees [13]. Similarly, city bus drivers must fulfill the same requirements, although they do not have to pass a criminal background check. For school bus drivers, the age requirement is that the applicant must be twenty-one years of age or older. A city bus driver can be as young as eighteen years of age, but if the bus is to go on the interstate the applicant must be twenty-one years of age. For a person to become an applicant, they must possess a Commercial Driver's License (CDL) with a passenger (P) endorsement [14].

The city bus driver physical examination and the school bus driver examination are similar. Both need to have 20/40 in each eye to pass the eye test [13]. This means that the driver can have corrective lenses or glasses as long as they wear them while on route. An applicant

needs to possess good eye-hand coordination, good spatial (3D perception) and be able to interact well with both children and adults [13]. A bus driver applicant can go to a commercial driving school but it is not necessary, no previous work experience is needed. As long as the applicant passes the defined tests and pays the $40 fee they can become a bus driver [13].

The requirements for passing the tests are as follows. The written exam is a multiple choice exam and the applicant needs a grade 80 or above to pass. The vision exam, the applicant must pass a standard Snellen exam with 20/40 in each eye as mentioned previously. The skills exam is comprised of on-the-road driving skills test and a pre-trip vehicle inspection [13]. The employer is responsible for obtaining the vehicle to be used in the road test, and is responsible for obtaining and supplying the applicant's driving record [14]. Last is the license, which runs upwards of $164.50 to $180.50 for an 8-year renewal [13]. These are the minimum requirements to become a driver whom is often overlooked by the common bus user. On top of the process of becoming a bus driver, there comes a time when the applicant is actually hired and put on their daily route. Usually, pedestrians will hop on a bus and hop off and not give a second thought about the person responsible for being on time, driving to the correct destination, collecting fares and keeping all users inside and pedestrians outside the bus safe. This is a significant number of tasks to handle. Often times, stress causes the bus drivers to not function at their peak performance, and their driving, awareness, and attitude can be impacted [15].

Bus drivers are not immune to the stresses of their jobs. Drivers usually take on odd hours and have to deal with argumentative passengers. It has been recorded that bus drivers are prone to, heart disease and back pain from sitting down for hours at a time [16]. There have been ties to strains on the minds of bus drivers. Bus drivers have experienced mental strains leading to anxiety and depression. Bus drivers have a higher rate of hypertension than the average working adult. These strains on the bus drivers have led to elevated blood pressure, obesity and course towards coping mechanisms [16].

Bus drivers have been recorded to abuse substances to cope with the stresses of their job. Some bus drivers use alcohol or tobacco to ease the high-stressed life. In 1990 that only 1 in 9 drivers actually reached retirement [16]. On top of the normal stresses of driving, traffic congestion, pedestrians jaywalking and bikers peeking around in the blind spots, drivers also have to communicate with the patrons on their bus, who may be troublesome.

Patrons have been recorded attacking bus drivers. In one case, a driver was stabbed to death in Manitoba by a patron. This act of violence in Manitoba was presumably due to the patron and driver getting into a verbal debate over the patron not paying the bus fare [17]. Other attacks on bus drivers have been recorded over bus fare disputes, but there are some pedestrians that just want to assault bus drivers [18]. There have been other reports of bus drivers being blinded by laser lights and just generally being harassed while they are trying to do their job [16]. A retired bus driver by the name Brian Lennox has had been held at gunpoint, amongst multiple other uncomfortable situations. He tried to be friendly to passengers by saying "Hi" as they arrived on the bus, but some patrons responded to this by throwing feces or urine and even vomit. Lennox, whom is now retired, is living with Post Traumatic Stress Disorder (PTSD) after driving with Winnipeg Transit for 30 years [19]. A reporter had the opportunity to interview a New York City bus driver to gain an insight into what typically occurs on a daily route. An interesting fact is that bus drivers have a button they can press to signify that "X" amount of patrons did not pay the fare when they boarded the bus. This bus driver was told by the company to not worry about the fare because that is where the most confrontations occur that can lead to harm for either the driver or the patron [20]. The biggest truth that came out of the interview was the acts of the pedestrians outside of the bus. Most people do not understand how risky it is to drive a huge vehicle in a city environment. One of the quotes that stood out was the driver's response to the question "Is there anything else that annoys you." The driver's response is shown in the following paragraph, and shows some of the difficulties that bus drivers face when dealing with the actions of surrounding pedestrians and passengers [20]:

"People have to understand, if the light is red, a lot of people don't like to stand on the sidewalk. They like to come off the curb. And I'm sorry to say, but those are the ones that always get their feet run over. People don't understand it. If you see a big vehicle turn, just stay on the sidewalk. Once the vehicle clears, you can go about your business. Another thing that really irks me is people who get off the bus in the front, or even the rear, and they cross right in front of the bus. That is the worst thing you can do".

It is to no surprise that some bus drivers will actually experience PTSD [15]. There was a law passed in 2015 that states, in brief, that a bus driver or taxi driver cannot be held to the same

consequences as other drivers for harming pedestrians or cyclists if there is no evidence of foul play or any evidence that would suggest a criminal act has been committed [21]. This is because of the importance that paid personnel who are practically are the blood vein of the city. A lot of people in the city rely on the bus and taxi drivers to get from destination to destination, if their job is halted or slowed down then there will be consequences. Running along with this bill there have been voices arising about needing more safety measures for the average bus driver.

2.2: Bus Driver Safety

Safety of bus drivers has been a rising concern. It is a job that mixes the stress of being on time, transporting passengers in a safe manner all while overcoming traffic congestion and long hours. Now insert angry bus users that either had a rough day or just cannot pay for their fare. These people may become hostile towards the bus driver. Typically, the abuse faced by the bus driver consists of verbal harassment from patrons but this can quickly escalate to forms of assault if the bus driver becomes confrontational. One driver in 2017 was stabbed to death in Canada, this has led to an up rise in bus driver safety [22]. Bus drivers feel unsafe in their work environment, a feeling that results in an increased level of stress [23].

Bus incidents involving passengers have shown a continued increase over the course of recent years. There were 60 incidents on transit buses in 2015, a 54% increase since 2014. In 2016 there were 46 assaults on buses, 11 involving someone carrying a weapon reported by the ATU (Amalgamated Transit Union) which serves both in the United States and Canada [22]. In total around 2,000 attacks are actually reported each year against bus drivers in Canada according to Canadian Urban Transit Association (CUTA) [19]. CUTA's President Patrick Leclerc summed up what is it like to be a bus driver; "Imagine you're in a driver's seat with a seatbelt, there's no escape possible, you've got a window on your left," said Leclerc. "You're very vulnerable" [19]. Bus drivers are on the front line and are easy targets [22]. Train conductors and airline pilots are all locked away and sit in protected areas, separated from the passengers. Bus drivers have a seat belt. What is being done to protect bus drivers? Currently not a lot which is not good news for the bus drivers. There has been talk about equipping the buses with shields.

Hundreds of Winnipeg transit workers rallied together in protest. They felt the government needed to make buses safer for both the driver and the passengers. One veteran bus driver, Nelson Giesbrecht, who was a colleague of Fraser the man stabbed to death, has 19 years

under his belt at Winnipeg Transit. He was quoted saying "I'm to the point of, 'put me in a cage.' I don't want to see money, I don't want to see nothing," said Nelson Giesbrecht [19]. Bus drivers are concerned with their safety every time they enter their bus in the morning. This adds significant pressure and tension, imagine constantly asking yourself if the next passenger you pick up will attack you. The idea of installing the shields within busses was brought up years ago but did not have enough support. The Canadian Bus Union has advocated for the installation of bulletproof shields, escape doors and windows on the left side as well as adding a panic button that alerts the authority in times of emergency [19].

These plastic shields did little to put a hold on the rising concerns about the violence towards bus drivers. Several bus companies in North America and Europe have actually implemented the shields and there was a negative reaction from the bus drivers themselves. San Francisco transit authority installed plastic shields on 10% of its 800 buses in the 1990s. The main issues were that the shield produced an odd glare, loud rattling, feeling claustrophobic and feeling cut off from the passengers. The number one assault to bus drivers' face is being spat on, and the shield did aid in protecting the drivers from spitting customers [19]. It seems like a shield would be easy to install but in fact, there is a multitude of different bus models out in the world all with their different styles.

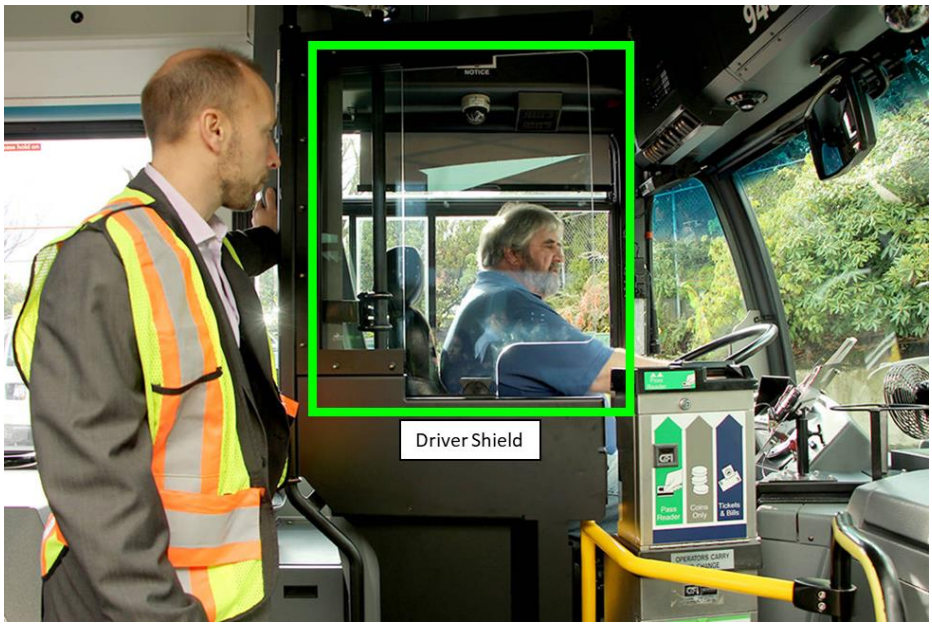

Figure 6: Shield Installed as a Means to Protect Bus Drivers from Pedestrians. The shield is highlighted in green. These shields are usually constructed from strong plastic to withstand hits without worry of shattering. Potential issues with the shields include glare and feelings of claustrophobia, which may turn drivers away from using them. [24]

The shields are also not cheap to install. The shields in San Francisco and Miami were priced at $1,600 to $1,900 [19].

Bus drivers do not have the easiest job in the world, but they do the best that they can. It would be beneficial to alleviate some of their worries by providing an aid that allows them keep their passengers as well as the pedestrians outside the bus safe. This is why the team proposed that an automated system should be implemented on buses. This system will provide the driver with another set of eyes, thus reducing their overall stress and increasing the safety on and around buses within the city.

2.3: Bus Accidents Involving Pedestrians

The following Tables (3-5) are "anonymous data that was obtained for all motor vehicle crashes occurring in New York City over a seven-year period (1991–1997)" [7]. The table shown below depicts three different age ranges and the recorded places that they were hit by a motor vehicle. The data was taken from reports in New York City.

Table 3: New York City Motor Vehicle Accidents – Pedestrian Actions and the Age Range of the Pedestrians Involved in Accidents. It was important to consider the varying actions that humans perform, as well as their age and height when designing a pedestrian detection system. [7]

	Age 5 to 9 Years	Age 10 to 14 Years	Age 15 to 19 Years
Crossing with signal	7.3	13.1	19.1
Crossing against signal	10.6	15	14.2
Crossing, no signal, marked crossway	5.6	5.6	5.5
Crossing, no signal or crosswalk	28	23.7	19.7
Walking along highway with traffic	0.2	0.7	1.4
Walking along highway against traffic	0.2	0.3	0.5
Emerging from in front of/behind parked vehicle	20.6	13.1	6.5
Going to or from stopped school bus	0.6	0.3	0.2
Getting on or off vehicle other than school bus	0.6	0.9	1.7
Pushing/working on car	0	0.1	0.1
Working in roadway	0.2	0.2	0.3
Playing in roadway	7.8	7.1	2.8
Other actions in roadway	5	6.3	7.8
Not in roadway	1.1	1.4	2.6
Unknown	12.2	12.3	17.4
Total	100	100	100

Three different age groups shown above represent the difference in height, knowledge and perhaps how well the bus driver can see them. This is important to note that there are

different sized pedestrians out in the world. With the varying sizes of pedestrians, as well as their knowledge of traffic rules, it is something the team took into consideration. It was desired to detect different pedestrians with similar accuracy for each range. This next table shows how the bus was moving when the different age groups were hit by a motor vehicle.

Table 4: New York City Motor Vehicle Accidents – Motor Vehicle Actions Taken by the Drivers. Pedestrians most often hit by buses driving straight and performing left turns. Pedestrians at very young ages tend to be less attentive to potential danger near buses. [7]

	Age 5 to 9 Years	Age 10 to 14 years	Age 15 to 19 Years
Going straight	82.3	78	63.5
Right turn	1.9	3	5.5
Left turn	2.5	4	7.8
Backing/parking	2.1	4	5.3
Other	3.5	3	6.5
Unknown	7.8	7	11.2
Total	100	100	100

Pedestrians were most often hit while buses were either driving straight or taking a left turn. This supported the decision that the left side was to be most closely investigated in developing a semi-autonomous turning solution. This is important to note to because it shows when and where sensors were required to be the most accurate. This next table shows the days of the week in which each age group of pedestrians were more likely to be struck by a bus.

Table 5: New York City Motor Vehicle Accidents – Age & Day of Week When the Accident Happened. Different age groups tend to show different probabilities of involvement in a bus-pedestrian accident. 10 to 14 year olds tend to be at higher risk than children of other age groups throughout the week. This indicates that children are a critical consideration in pedestrian detection. [7]

	1 to 4 Years Old		5 to 9 Years		10 to 14 Years		15 to 19 Years Old	
Day of Week	Frequency	Percent	Frequency	Percent	Frequency	Percent	Frequency	Percent
Sunday	420	12.7	1,115	10.6	1,015	9.1	686	9.1
Monday	331	10	1,325	12.6	1,524	13.6	1,021	13.5
Tuesday	344	10.4	1,349	12.9	1,718	15.3	1,058	14
Wednesday	287	8.6	1,390	13.3	1,650	14.7	1,011	13.4
Thursday	351	10.6	1,505	14.4	1,563	14	1,035	13.7
Friday	420	12.7	1,700	16.2	1,940	17.3	1,186	15.7
Saturday	464	14	1,444	13.8	1,232	11	913	12.1
Unknown	703	21.2	651	6.2	558	5	628	8.3

Table 5 indicates the highest level of pedestrian traffic based on a daily basis, which should be an important consideration in future testing of pedestrian detection systems to plan for a "worst-case" scenario.

It is not only walking pedestrians that are subject to blind spot collisions. Bicyclists frequenting the New York City roads and are often in just as much, in more danger, than pedestrians. "Between 1996 and 2005, 225 bicyclists died in NYC. Most fatalities resulted from motor vehicle crashes (92%)" [25]. With the large number of pedestrian and cyclist fatalities that involve motor vehicles, especially city buses, cities are determined to find a solution.

2.4: City Solutions to Transit Bus Accidents

Many transit agencies, such as the Metropolitan Transit Authority in New York City, have begun implementing changes within the past few years. One of the biggest changes occurring is the alteration of the design of buses in order maximize the field of view for the bus driver. Altering the design of buses has been a major argument for bus driver unions defending their drivers. The main argument is that if the design of the bus can be made to reduce blind spots, it will provide drivers with an increased level of safety knowing their vision is less obstructed and for protecting pedestrians [2] [4]. The two biggest changes that are necessary to reduce blind spots are redesigning and repositioning the mirror along with making the A-pillar design as slim as possible. The A pillar on a bus consists of the frame that exists on the edge of the front windshield [26]. Along with making physical alterations to buses, transit agencies have explored a large variety of options such as, raising public awareness about the issue, infrastructure and legislative based solutions, and lastly new training methods for bus drivers.

Side view mirrors should be placed in a position in such that bus drivers have to slightly look up or down. In addition to this mirrors should be physically smaller [26]. The American Public Transit Association (APTA) recommended in a report that side mirrors should have height adjustable brackets along with the addition of convex mirrors in order to have a wider mirror view. The APTA also recommends that a mirror design criteria should be set that all transit agencies and bus manufacturers should follow [27]. With this alteration, the only object causing the blind spot is the frame of the bus. According to the MTA, the frame of the bus takes away a 3.16 degree viewing angle, which can mask many pedestrians. The next generation of

buses ordered by the MTA will only have a frame that takes away a 1.65 degree viewing angle [28]. Many transit agencies have already followed suit with this and are looking at ways to make the frame narrower.

Transit agencies have worked and continue to work with government and community associations to reduce pedestrian fatalities. Data analysis has shown that disabled, young, and elderly people are at a higher risk of getting hit by a bus than all other groups [27]. Elderly people make up 38% of all pedestrian fatalities in New York City in 2008 [29]. The APTA recommended numerous community outreach programs to help reduce bus-pedestrian related fatalities. It is recommended that partnerships with local media and schools are a great way to raise awareness of this issue. One of the biggest points that the APTA is trying to spread is that a bus turns very differently from a car. Pedestrians need to be aware that a bus sweeps through the crosswalk, and if they are too close it is much more likely that they will be hit [27]. Research has shown that pedestrians rarely recognize when long rear axle vehicles turn left [30]. Being unaware of a large vehicle that may sweep close to the crosswalk can lead to potential fatalities. Creating an awareness for the danger a bus can pose will help prevent accidents in the future.

Another method of combating this problem is implementing infrastructure and legislative based solutions. Many traffic engineering jobs occur without the input of transit agencies. These agencies have conducted numerous studies, and are experts in their field. Thus, it is essential that transit agencies have a say in traffic engineering jobs so the best possible solutions for drivers, passengers, and pedestrians can be determined. For infrastructure based solutions the APTA recommended many options. A possibility for reducing accidents with pedestrians is the use of "pedestrian scrambles" [27]. A pedestrian scramble works by allowing pedestrians to cross in all directions while all automobile traffic is stopped at the intersection. This can be useful in major street corridors where serious pedestrian crashes are more deadly than smaller local streets [29]. Many major cities around the world use pedestrian scrambles such as San Diego and New York City. A legislative based solution is banning curbside parking spaces at the approach of intersections which has been tested in New York City. The goal of this is to remove obstructions at intersections which helps pedestrians see oncoming vehicles and helps drivers see pedestrians [29].

Lastly, the APTA recommended many ways to improve bus driver training to help combat this problem. The APTA recommends that bus drivers should have refresher training to

help maintain the importance of safe driving. Some safe driving techniques include squaring off turns and waiting a few seconds before performing the turn. It also stresses the constant "Rock and Roll" technique where a bus driver moves around in their seat in order to scan for pedestrians while performing a turn. A training course that combines classroom, behind the wheel practice, and analysis of top pedestrian accident locations is the most ideal training course [27]. Many officials from transit agencies across the US agree that getting pedestrian deaths to zero is the ultimate goal but avoiding these kinds of actions is a joint effort between pedestrians, city officials, transit agencies, and bus drivers [6] [30]. The New York City Transit Authority, the parent agency of the MTA, accounts for 18.7% of the total public transportation market in the United States with the next biggest market share taken up by Los Angeles County Metropolitan Authority with almost 7%. Any recommendations and changes these companies make have the possibility of having a deep impact on other transit agencies [31].

2.5: Pre Existing Technologies in Buses

Currently, many transit agencies and bus manufacturing companies have taken strides in reducing incidents with buses by incorporating various sensors that serve to make bus operations safer for all. Two notable examples are the Future Bus by Mercedes-Benz, and Volvo's upcoming safety system for buses. Mercedes, more specifically the parent company Daimler AG, controls 27% of the truck and bus manufacturing market while Volvo controls roughly 11% [32]. Daimler and Volvo hold a considerable market share and therefore have a wide influence on other bus manufacturers.

In the summer of 2016, Mercedes-Benz launched a semi-autonomous bus called the Future Bus, as seen in Figure 7. It managed to drive 12 miles from Amsterdam's Schiphol airport to Haarlem, a city adjacent to Amsterdam [33]. This bus route is a part of one of the longest Bus Rapid Transit (BRT) system lines in Western Europe. The route serves 125,000 passengers on average every day. Along the route there are tight bends, no barriers to the oncoming bus lane, 22 traffic lights, and three tunnels [34]. The bus had the necessary technology to navigate along a precarious, high traffic route, and can show possible solutions for detecting and decision making based on what is surrounding the bus.

For navigation the bus uses GPS while being supplemented by a lane camera and four other cameras to determine the position of the bus for pinpoint accuracy. The lane camera has a

range of up to 80 meters. For pedestrian and vehicle detection the bus uses a combination of cameras and radar systems. There are four close-range radar sensors in the front end and corners of the bus that detect pedestrians within 10 meters. The radar sensors continuously scan for object detection even if the bus is not moving. This is supplemented by ten cameras that can detect pedestrians within a range of 60 meters. For vehicle detection the bus uses a radar with a range of 200 meters [35]. A vehicle to infrastructure (V2I) communication system is incorporated on the Future Bus which is used to communicate with BRT infrastructure, mainly traffic lights. With the V2I system the bus can send signals to BRT traffic lights 300 meters away and the traffic lights can prioritize the bus instead of other pedestrians or vehicles. This allows for a quicker travel time for the bus. If a traffic light is unable to communicate with the bus, the front cameras can still read the traffic signal [34].

The Future Bus shows that BRT lanes are ideal for autonomous buses. BRT lanes are separate from other lanes, the route will always be the same, and the bus will always encounter similar situations [34]. A driver will still be present in the bus in case the bus requires human control. Since there should be less stress on the drivers they should be fully alert for when they need to take over [35]. Ultimately the bus driver is still responsible for the bus and safety for the passengers.

Figure 7: Mercedes-Benz Future Bus. This model, used to transport passengers in the Netherlands, is a fully-autonomous vehicle designed to avoid potential accidents in real-time. [34]

Research done by Volvo shows that there are 1.25 million annual road traffic deaths across the world. In order to reduce accidents involving people outside the bus, Volvo decided that the buses they produced will have a pedestrian and cyclist detection system that will be launched on its European city bus fleet in 2017. The detection system was field tested during the autumn of 2016. It is important to note that the new buses are not autonomous buses but will have a new safety system, the Pedestrian and Cyclist Detection System. The system works by using cameras to monitor the surroundings of the bus. If a pedestrian, bike rider, or any bystander becomes too close or if the system determines they may become too close to the bus the system transmits a warning. The warning consists of an alert sound that can be heard by pedestrians and the bus drivers is alerted by sound and light signals that come from the dashboard on the bus. Cameras on board the bus along with image processing and tracking software determines if a pedestrian or any other obstacle is getting too close. The system will activate the bus's horn if it senses that anyone is getting too close to the bus [9]. The system in action can be seen in Figure 8.

Figure 8: Volvo Pedestrian & Cyclist Detection System. This system monitors the bus's surroundings using cameras and alerts the driver when action needs to be taken. [9]

Volvo accounted for any noise issues that the system would make when designing it. A synthetic background sound was developed with a frequency range that is not disruptive to daily life. According to Peter Danielsson at Volvo, the sound from the alarm will not penetrate triple

glazed windows due to their sound resistive nature [9]. As previously stated, Volvo's system is a tool to help keep non-bus occupants safe and to help bus drivers stay aware of the surrounding. Again, the bus driver is responsible for driving the bus safely and keeping everyone involved safe.

In addition to a reliable pedestrian detection system that can alert the driver and trigger the bus to stop, a form of proactive technology that can warn unaware walkers and bikers of a turning bus can help to prevent pedestrian fatalities. There have been multiple technologies developed for the purpose of raising pedestrians' awareness of nearby vehicles. These technologies exist for multiple types of vehicles, including factory and construction vehicles, as well as licensed road motor vehicles.

For workplace vehicles, such as warehouse forklifts, Radio Frequency Identification (RFID) systems have been implemented to detect the proximity of nearby workers. RFID devices can be configured to emit electromagnetic fields, which are then read by an RFID receiver to determine the proximity of the transmitter. An example of a workplace RFID system is the Claitec Pedestrian Alert system, which can be mounted to a forklift or similar machine. Claitec's solution uses RFID enabled tags worn by employees to warn the machine operator that other workers are in the area [36]. While appropriate for a workplace application, RFID detection would not likely be appropriate for a city bus, as it would require pedestrians to carry a unique device. The concept of detecting devices nearby may assist in pedestrian detection. However, in a large-scale setting, in this case a city, detection would have to be through use of a common signal, such as Bluetooth from cell phones.

As part of the Federal Transit Administration, the Tri-County Metropolitan Transportation District of Oregon conducted a demonstration test of multiple pedestrian turn warning systems designed for transit systems. The first attempt towards using a pedestrian warning system for turning buses was implemented by TriMet in the city of Portland, Oregon in 2011 [37]. The system measured the position of the driver's steering wheel to detect when a turn was initiated, then provided a verbal warning: "pedestrians, bus is turning" in both English and Spanish to nearby pedestrians. Testing was stopped after only a few months. The system was evaluated by surveying both bus operators and the general public. The system was described as "glitchy", and seen as a nuisance by drivers as well as pedestrians [37]. The system would give very loud alerts as the bus completed turns, which became annoying for some pedestrians and

residents [38]. Reportedly, the alert often began to sound after the bus had already turned into the crosswalk [39]. The project cost $46,000 to test the alerts on 10 buses [37]. This project was revised and attempted by TriMet once again in 2013, which was a more successful experiment. Five new pedestrian warning systems were prepared for testing. The devices differed in their method of delivering a cautionary message to nearby pedestrians. The five systems were configured as follows: One device gave an audible and visual warning, two gave audible warning only, one gave visual only, and one device displayed a warning sign at a fixed location (not attached to a bus).

The audible and visual device, the "Safe Turn Alert" system by Protran technology, detects a bus's turning when the steering wheel passes 45 degrees of turn [37]. A voice recording, as well as Light Emitting Diode (LED) lights warn pedestrians to the sides of the bus. The external speaker volume is adjusted automatically to compensate for the noise level in the area outside the bus. Protran has also developed a "Safe Turn Alert 2.0", which is activated by either proximity sensors on the lower front corners of the bus, or the turn signals [40]. An image of Protran's sensor, LED, and speaker placement can be found in Figure 9:

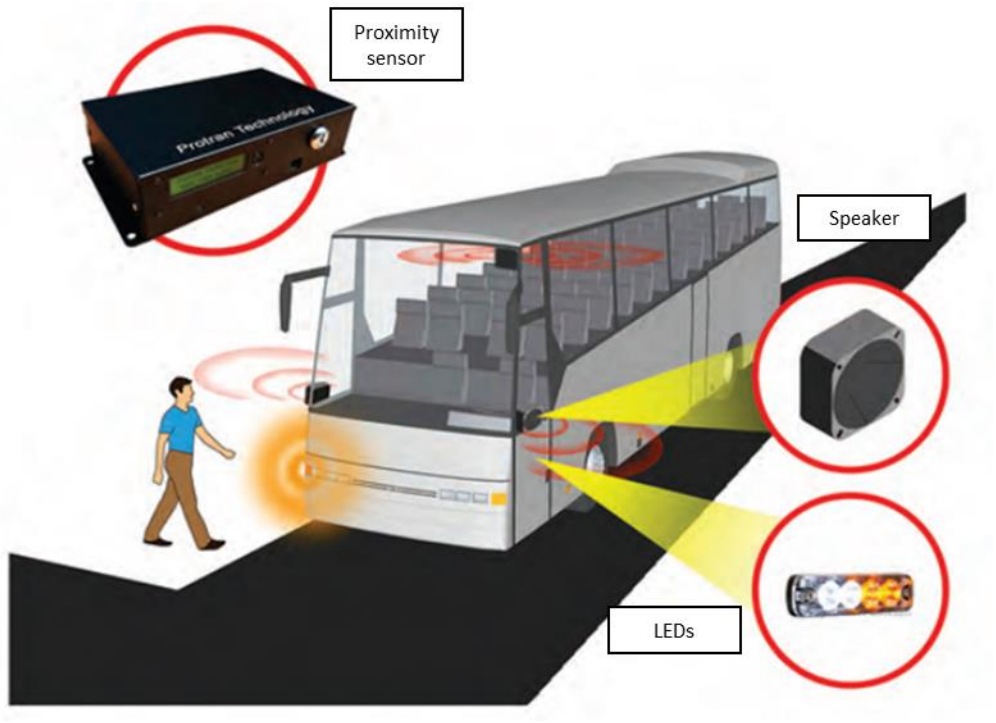

Figure 9: Protran Safe Turn Alert 2.0 Component Placement. Proximity sensors monitor the bus's corner blind spots. Pedestrians are provided warning using LEDs and a speaker mounted to the bus's exterior. [40]

The Clever Devices "Turn Warning System" utilizes a sensor within the housing of the bus's steering column. This system also provides a verbal alert to pedestrians when the wheel is turned 45 degrees or more [37]. The speakers' volume can be adjusted for day driving, night driving, or quieter areas. This system also tracks the bus's location via GPS, which is calibrated when the system starts up with the bus. This location data can be used to adjust exterior volume level on a location-based basis [41].

Transit Tech Solutions develops an audible warning system that is dependent on the activation of the bus's turn signal. When the turn signal is pressed, a verbal warning can be heard on the exterior of the bus. This system is disabled when the bus exceeds 35 miles per hour to avoid turn notifications while the turn signal is used to indicate a lane change on the highway [42].

The Dinex Star system monitors a bus's speed and steering wheel to determine when a turn is initiated. This system integrates pedestrian warning into an upgraded headlight. Designed to provide additional visibility for the driver, as well as warn pedestrians of the direction a bus is turning, one of the two headlights is set brighter than the other while the bus turns. A reported additional 35-degree viewing angle is provided to the driver by this headlight during a turn [10]. This system does not implement any audio-based alert.

The fixed-location device tested was the "BUS Blank-Out Sign" developed by TriMet to be installed at crosswalk corners [10]. The sign, reading "bus", is illuminated when a bus begins to turn in the direction of the crossing. Having to install on every crosswalk would not be a cost efficient however, and it shows that a better implementation would be too have a system incorporated directly with the bus.

The results of daily surveys revealed that bus operators were not exceptionally impressed with the devices' abilities to warn pedestrians. "From the daily surveys, less than half of operators thought the systems were effective at alerting pedestrians at intersections and bus stops, and less than one third thought the systems were effective at reducing close-calls" [10]. During pedestrian surveys, the pedestrians responded more positively than the bus operators in regard to the systems raising their awareness of incoming buses. "About half of the cyclist and bus rider respondents thought the systems were effective at alerting them at intersections and bus stops, respectively. In total, 12% of pedestrians, 17% of cyclists, and 7% of bus riders reported the systems played a role in avoiding a collision with a bus" [10]. During

testing, both operators and pedestrians were asked about potential improvements they would like to see if the systems were deployed full-time. While the system was designed to be fully autonomous, operators suggested having a way to trigger a warning, such as a push button, if they are to notice risky behavior or unawareness in nearby pedestrians [10]. A common suggestion made by both pedestrians and bus drivers was that it is likely unnecessary to have the turn warning sound with every turn the bus makes. Therefore, an integration into the bus's GPS or Computer Aided Dispatch (CAD) system could be used to sound warnings only in city areas that are suggested, or proven, to be problematic [10]. Interestingly, this type of suggestion is calling for a more autonomous solution, rather than a solution that would rely on the bus driver's skills. This indicates that autonomous solutions for bus safety are of interest to both bus operators and pedestrians.

The warning protocols and technology used in one or more of these proactive warning devices could be compatible with the pedestrian detection system, with the exception of the fixed-location device, as the focus will be on solutions to be integrated into the bus itself. The aforementioned statistics that the Federal Transmit Administration (FTA) produced based on survey responses indicate that a proactive warning system would indeed be helpful in deterring collisions between buses and pedestrians. Even though only 12% of the pedestrians surveyed felt that the turn warning kept themselves and others safe, the avoidance of any potential injury is a considerable improvement

2.6: Autonomous Braking Standards for Cars

Autonomous braking is a technology that many car manufacturers have begun to incorporate into their vehicles. The general term used to refer to autonomous braking features in automobiles is Automatic Emergency Braking (AEB). AEB systems are designed to monitor a vehicle's surroundings in real-time for potential crash situations. If a hazardous situation is detected, the car will promptly provide an audible and/or visual warning to the driver, and intervene to stop the car if the driver is suspected to be inattentive. Many AEB systems incorporate two types of automatic assistive braking: Dynamic Brake Support (DBS) and Crash Imminent Braking (CIB) [43]. DBS is designed to increase the braking of the vehicle in a situation where the driver is aware of the immediate potential crash and applies the brakes, but does not brake strong enough to stop the vehicle. CIB activates in the situation where the driver

fails to apply the brakes in the event of an imminent crash, and the vehicle's brakes are applied automatically. Most automobiles with AEB systems implement cameras, radar and laser proximity sensors to detect the distance of nearby objects and the speed at which they are approaching relative to the vehicle. The U.S. Department of Transportation National Highway Safety Administration (NHTSA) plans to make testing of AEB systems a standard procedure under the existing NHTSA 5-Star Safety Rating beginning with vehicles set to release for the 2018 model year [44].

AEB systems specifically created to avoid collisions with pedestrians are recognized and referred to by the NHTSA as Pedestrian Automatic Emergency Braking (PAEB) systems. The NHTSA has not currently determined and/or formally released any performance specifications for PAEB specifically, however it is stated to be "a promising technology that may be added to the 5-Star Safety Ratings list of recommended technologies in the future" [43].

The Insurance Institute for Highway Safety (IIHS) currently uses a standard testing procedure to evaluate the effectiveness of cars' make and model specific AEB systems in their ability to stop or slow a vehicle. The test consists of an engineer driving the vehicle directly toward a stationary inflatable target, used to replicate the rear of a stationary car. These tests only consider the AEB's performance in front-to-rear collisions. The stopping capability of the AEB is analyzed at speeds of 12 mph and 25 mph. Cameras are used to capture the driver's view of both the road and the dashboard lights in each test [45]. The deceleration of the vehicle is monitored and recorded, as well as whether the car indicates a forward collision warning to the driver. For each independent vehicle tested, the IIHS judges the effectiveness of the AEB system using a scoring system. The score table can be seen in Table 6:

Table 6: IIHS AEB Testing Score Chart. This chart was created to assess the effectiveness of AEB systems in stopping the vehicle. Higher score indicates greater stopping capability. [45]

	12 mph test			25 mph test				Forward collision warning
Speed reduction (mph)	less than 5	5 to 9	10 or more	less than 5	5 to 9	10 to 21	22 or more	n/a
Points	0	1	2	0	1	2	3	1

Using the points determined by the test scenario, the AEB system of a single car model is given a rating of either “basic” with 1 point, “advanced” with 2 to 4 points, or “superior” with 5 to 6 points [45].

Announced by the NHTSA and the IIHS on March 17, 2016, 20 automobile manufacturers committed to ensuring AEB systems would become a standard feature for new cars by September 2022 [44]. The commitment applies to vehicles 8500 pounds or less in weight. The AEB standard inclusion agreement has also been set for vehicles between 8,501 and 10,000 pounds, required before September 2025 [44]. These dates are expected to allow manufacturers to ensure proper functionality of AEB systems before fully launching the technology as a standard, as well as account for the time it may take to implement any recent AEB features that are still in development.

Considering the technologies used by automotive manufacturers for automatic braking in cars, a similar system with a much greater weight and stopping time consideration would be needed in order to bring a city bus to a full stop. The main focus of this project was incorporating a sensor system that could identify an event where a bus needs to stop, which could then be integrated into an automatic braking system in the future.

In addition to existing automatic braking standards in cars, research was conducted on how car manufacturers have developed blind spot detection technologies [46]. One example is the Volvo “Blind Spot Information System” (BLIS). BLIS works in Volvo vehicles by the following means: The car has two cameras in the rear view mirrors (one in each rear view mirror) and a computer connected to the cameras. If one of the cameras detect an object in the mirrors, this image will be transmitted to the computer. The computer will process the image from the cameras to determine if the object detected is a vehicle, and whether or not is is approaching the car. If that is the case, the door panel BLIS light will turn on and will stay on until the vehicle that was detected, either passes by the car or is out of range. An additional feature of BLIS is the Cross Traffic Alert (CTA) system which detects vehicles that could be approaching the car while the vehicle is driving backwards. This feature is activated automatically once the car switches gear into reverse [47]. The rise of blind spot detection systems in cars will aim to prevent highway fatalities, as the technology becomes more readily available in car purchases.

2.7: Blind Spot and Automatic Stopping

The project ultimately focused on one major factor that leads to pedestrian accidents: when the driver is unable to see a pedestrian in a blind spot caused by the structure of the bus. The proposed system needed to be equipped to deal with these blind spots, and in future revisions, ultimately automatically stop the bus to prevent an accident if the driver is unable to avoid the accident on their own.

Most public buses in the United States are manufactured without much driver assistance in monitoring blind spots beyond the bus's mirrors. A Washington Post report suggest that a blind spot of a bus can reach up to 6 feet in height [48]. The picture shown in Figure 10 describes the specific blind spot of a mid-sized bus. It was published by the Canadian society of insurance automobile in Montreal.

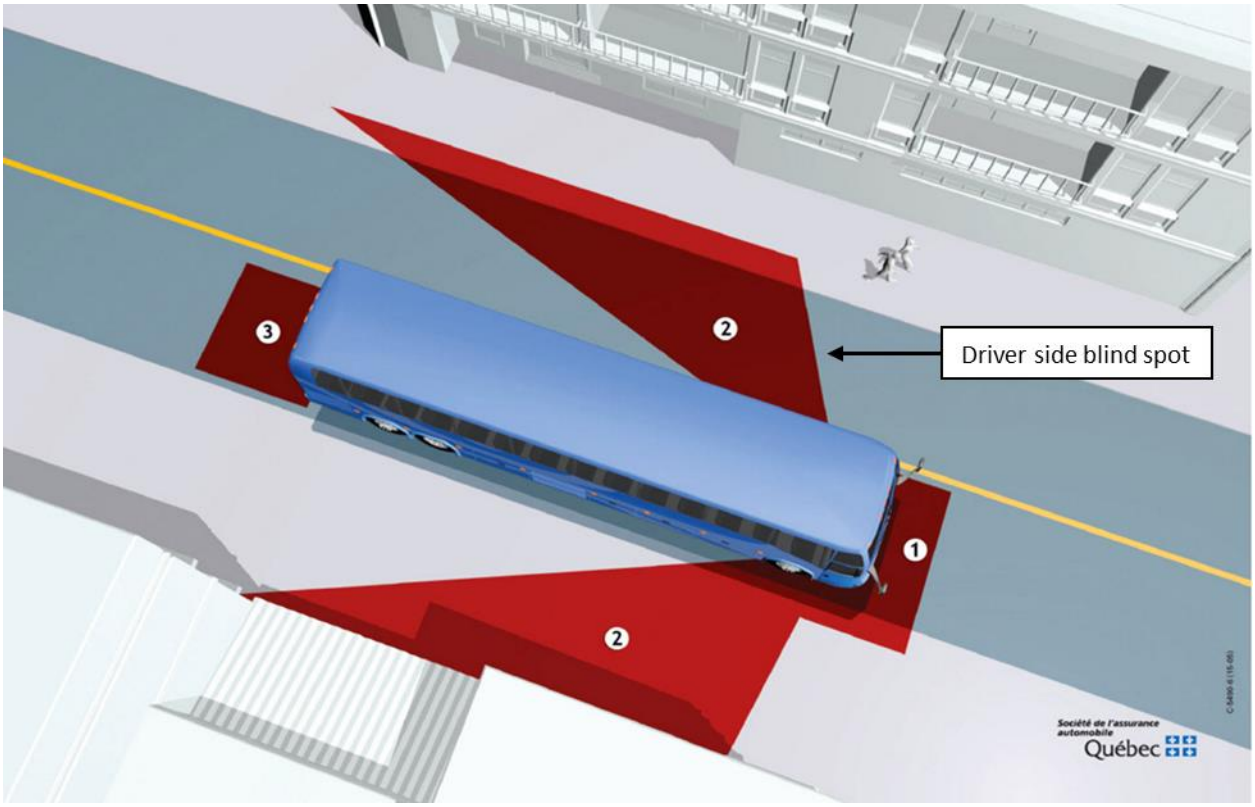

Figure 10: Blind Spot Critical Areas Present on the Sides, Front, and the Back of the Bus. Red areas are not visible by the bus driver. This project focused on the blind spots on the sides of the bus originating from the two front corners, especially the driver-side (left) corner. [49]

As for this project, the main concern was the blind spots originating from the corners of the bus, which corresponds to zones 2 to either side of the bus in Figure 10. As shown, the driver

has a very large part of his mirrors not covering any sides of the bus (red areas). This is where the system sensor will detect any pedestrians and alert the driver of obstacles, especially humans, around that area. It is estimated that the bus driver will have no vision using his mirror for the 4-5 meters to the right of his seat and the 9 meters at the back of his seat (which corresponds to Zone 2 at the driver's seat side) [49]. From the other side, the blind spot is slightly smaller with about 3-4 meters of blind spot wide and about 6-7 meters long (which corresponds to Zone 2 at the passenger seat side of the bus).

The blind spot was divided into the following zones: The potential threat zone, the danger zone and the critical action zone. In Figure 11, a detailed schematic of the defined blind spot of the bus is shown.

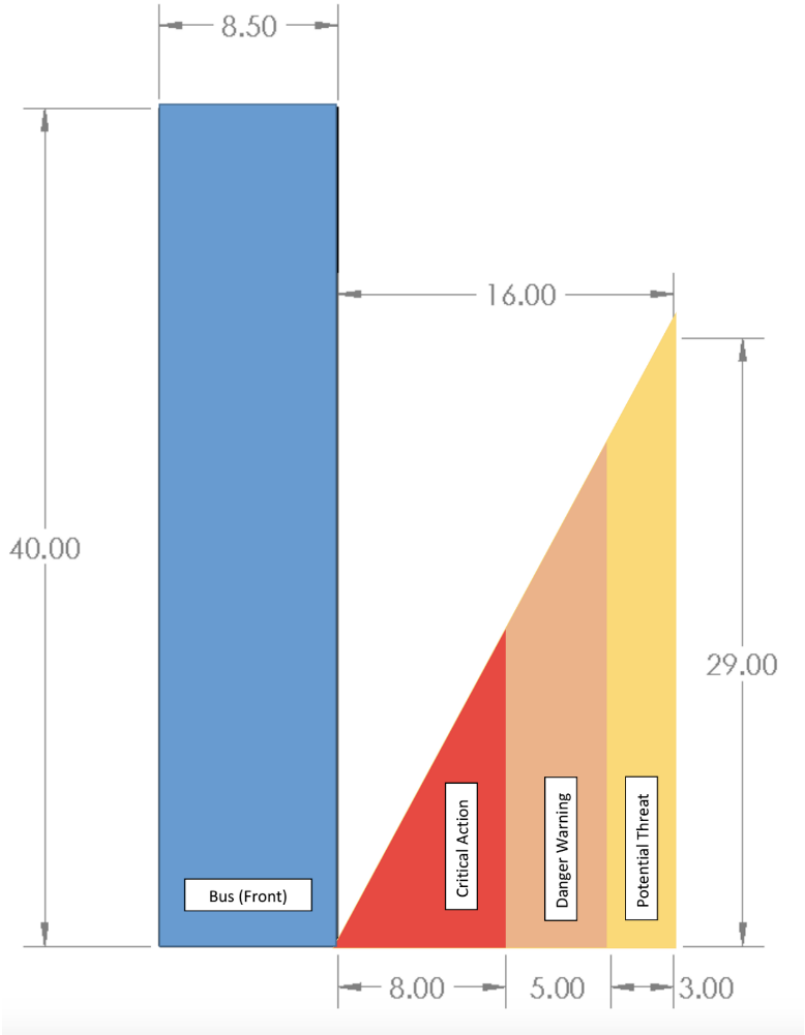

Figure 11: Blind Spot Zone Division (Dimensions are Represented in Feet). Area extends up to 16 Feet from the driver's side window. The three zones represent increasing level of danger to the pedestrian, as the distance between the pedestrian and bus narrows.

Each zone in the blind spot of the bus corresponds to a different level of system action because the distance between the pedestrian and the bus was a key factor in determining how fast the response needs to be. Starting from the potential threat zone, the systems sensors were able to detect that no obstacles were further than 13 feet away from the bus. The next two zones indicated increasing levels of danger. Understanding the level of danger in each zone of the blind spot allowed for the system to take the necessary degree of action.

Stopping the bus immediately was among the many factors that were considered in order to come to a full stop before hitting the pedestrian. One approach that can be implemented in a future revision, beyond the sensor integration achieved in this prototype, is to connect the systems FPGA data processing straight to the braking system of the bus. The decision making, managed by the system's central computing, was connected to all of the sensors in order to be able to determine whether the received information strictly implied that the encountered obstacle is a human. The overall human detection and central computing system could eventually trigger the braking system of the bus without the need of any intervention from the bus driver. In order to achieve such integration, there would have been interfacing between the sensing system and an automatic braking system much like Automatic Emergency Braking (AEB) systems in cars. If the driver does not take an action and the sensors were to still be sending data that assure that an accident will occur, the autonomous part of the system will take over the control from the driver and apply the brakes automatically. Even though this method may seem the safest and most efficient for pedestrians, coming to an immediate and complete stop might have severe consequences on the bus passengers, especially the ones that are standing up and not holding tight to anything. Research indicated that the average force that can be absorbed by a mid-age person is about 4.6 G's, or 4.6 times the force of gravity [50]. Studies have shown that a person should be able to withstand a short burst of 6-8 G's at maximum without any major injury to the body [50]. This amount of force, however, would not be ideal.

2.8: Technologies for Pedestrian Detection

In order to meet the requirements of the design different technologies were explored to determine what would be the most effective means of pedestrian detection for the overall system.

The use of embedded technologies in the scope of this project was implemented through the use of a Field Programmable Gate Array (FPGA) with parallel microprocessor capabilities

allowing for proper data fusion between the sensors. The ability to simultaneously run hardware functionality and logic in tandem with the software systems of a microprocessor created a level of flexibility that would be pivotal. An FPGA provided the necessary functionality and implementation capabilities for any sensor needed.

The use of the Zynq Evaluation and Development Board (Zedboard) was determined to provide better capabilities than the Nexys 4 DDR based on the considerations of the project. The Zedboard includes either a Xilinx Zynq-7000 series or Xilinx Zynq-7000S series All Programmable System on Chip (SoC) depending on the quality of the device chosen. The main difference is the 7000 series SoC contains a dual-core ARM Cortex A9 processor and the 7000S series has a single-core. All cost optimized devices have a Xilinx Artix-7 FPGA allowing for hardware and software programming within a singular device [51]. The Artix-7 in particular is designed with cost in mind, and allows for a low weight, low cost product that can be designed with the intention of large scale productions (i.e. every transit bus in New York). The mid-range devices utilize a Kintex-7, a higher end series-7 FPGA equivalent. The Kintex-7 is designed for a larger range of applications that can be expanded to larger scale applications such as advanced medical equipment. If cost was not a constraint there is also the Virtex-7 FPGA which is designed for high end FPGA applications in which the boards can have upwards of 2 million logic cells [52]. The cost alone of the Virtex-7 is upwards of $4,000 and the boards that utilized it cost well over $20,000 based on the prices available from Digikey [53].

Based on the cost constraints it was most beneficial to utilize the Xilinx Zynq-7020 as it met the necessary requirements for the project's design. This is the highest quality "cost efficient" series of board which utilizes the Artix-7, and the lowest quality of the mid-range boards start at over $1000 [51]. Spending over 40% of the available budget on the FPGA would not be the most cost effective solution. A detailed breakdown of the capabilities of the Zynq-7000 series family can be seen in Table 7.

Table 7: Zynq-7000 All Programmable SoC Family Product Table. This table lists a number of the SoC options available, with their associated specifications. A greater number of Flip-Flops, Lookup Tables (LUTs), and amount of RAM provides greater computing resources for the system design. [54]

		Cost-Optimized Devices						Mid-Range Devices			
	Device Name	Z-7007S	Z-7012S	Z-7014S	Z-7010	Z-7015	Z-7020	Z-7030	Z-7035	Z-7045	Z-7100
	Part Number	XC7Z007S	XC7Z012S	XC7Z014S	XC7Z010	XC7Z015	XC7Z020	XC7Z030	XC7Z035	XC7Z045	XC7Z100
Processing System (PS)	Processor Core	Single-Core ARM® Cortex™-A9 MPCore™ Up to 766MHz			Dual-Core ARM Cortex-A9 MPCore Up to 866MHz			Dual-Core ARM Cortex-A9 MPCore Up to 1GHz[1]			
	Processor Extensions	NEON™ SIMD Engine and Single/Double Precision Floating Point Unit per processor									
	L1 Cache	32KB Instruction, 32KB Data per processor									
	L2 Cache	512KB									
	On-Chip Memory	256KB									
	External Memory Support[2]	DDR3, DDR3L, DDR2, LPDDR2									
	External Static Memory Support[2]	2x Quad-SPI, NAND, NOR									
	DMA Channels	8 (4 dedicated to PL)									
	Peripherals	2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO									
	Peripherals w/ built-in DMA[2]	2x USB 2.0 (OTG), 2x Tri-mode Gigabit Ethernet, 2x SD/SDIO									
	Security[3]	RSA Authentication of First Stage Boot Loader, AES and SHA 256b Decryption and Authentication for Secure Boot									
	Processing System to Programmable Logic Interface Ports (Primary Interfaces & Interrupts Only)	2x AXI 32b Master, 2x AXI 32b Slave 4x AXI 64b/32b Memory AXI 64b ACP 16 Interrupts									
Programmable Logic (PL)	7 Series PL Equivalent	Artix®-7	Artix-7	Artix-7	Artix-7	Artix-7	Artix-7	Kintex®-7	Kintex-7	Kintex-7	Kintex-7
	Logic Cells	23K	55K	65K	28K	74K	85K	125K	275K	350K	444K
	Look-Up Tables (LUTs)	14,400	34,400	40,600	17,600	46,200	53,200	78,600	171,900	218,600	277,400
	Flip-Flops	28,800	68,800	81,200	35,200	92,400	106,400	157,200	343,800	437,200	554,800
	Total Block RAM (# 36Kb Blocks)	1.8Mb (50)	2.5Mb (72)	3.8Mb (107)	2.1Mb (60)	3.3Mb (95)	4.9Mb (140)	9.3Mb (265)	17.6Mb (500)	19.1Mb (545)	26.5Mb (755)
	DSP Slices	66	120	170	80	160	220	400	900	900	2,020
	PCI Express®	—	Gen2 x4	—	—	Gen2 x4	—	Gen2 x4	Gen2 x8	Gen2 x8	Gen2 x8
	Analog Mixed Signal (AMS) / XADC[2]	2x 12 bit, MSPS ADCs with up to 17 Differential Inputs									
	Security[3]	AES & SHA 256b Decryption & Authentication for Secure Programmable Logic Config									
Speed Grades	Commercial	-1			-1			-1			-1
	Extended	-2			-2,-3			-2,-3			-2
	Industrial	-1, -2			-1, -2, -1L			-1, -2, -2L			-1, -2, -2L

Notable features on the Xilinx Zynq-7020 ZedBoard can be seen in comparison to the Nexys 4 DDR in Table 8 [51]. The functionality available on this specific device will provide enough capacity for the application of this project, and the student price shall cost $300.

Table 8: Xilinx Zynq-7020 SoC Programmable Logic Features vs. Nexys-4 DDR. Based on these specifications, the Zedboard was chosen over Nexys 4 for its greater number of logic cells and DDR Memory (RAM). [55] [51]

Board	Logic Cells	LUTs	Flip Flops	Total Block Ram	DDR Memory	DSP Slices
ZedBoard	85,000	53,200	106,400	4.9 Mb	512MB DDR3	220
Nexys 4	31,700	63,400	126,800	4,860 kb	135 DDR2	240

The board that was compared to the Zedboard was the Nexys-4 DDR. It featured the same Artix-7 technology, but it lacked the capacity that would be required for the scope of the project. For example, the Zedboard consists of 85,000 logic cells and 4.9 Mb block Ram while the Nexys 4 has roughly 31,700 and 4,860Kb block ram. A notable benefit of the Nexys-4 over the Zedboard for the project application was the lower power consumption as the Nexys-4 requires a 5 volt (V) source, and the Zedboard requires a 12V source. However, from looking at examples of image processing there have been issues with memory [56]. The Nexys-4 has significantly less memory capabilities which could have led to issues during the implementation of the project. Because of this, the Zedboard was the clear choice as it does not fall short on memory capacity while maintaining the necessary peripherals and features. To further breakdown the differences between the board a comparison of the peripherals can be seen in Table 9 [55] [51].

Table 9: Zedboard & Nexys-4 DDR Peripheral Comparison. The Zedboard provides a greater number of PMOD IO ports, which is ideal for connections to external sensors. [55] [51]

Peripheral Component	Zedboard	Nexys 4
PMOD IO Ports	5	4
Push Buttons	7	5
Slider Switches	8	16
LEDs	8	16
Microprocessor Capabilities	Dual Core ARM Cortex A9 Processor	Microblaze IP Block (Inside FPGA)

The datasheet block diagram for the Zedboard can be seen in Figure 12, which showcases the various features for the processing system, multiplexed I/O, and programmable logic of the Zedboard including LEDs, push buttons, slider switches, and PMOD IO.

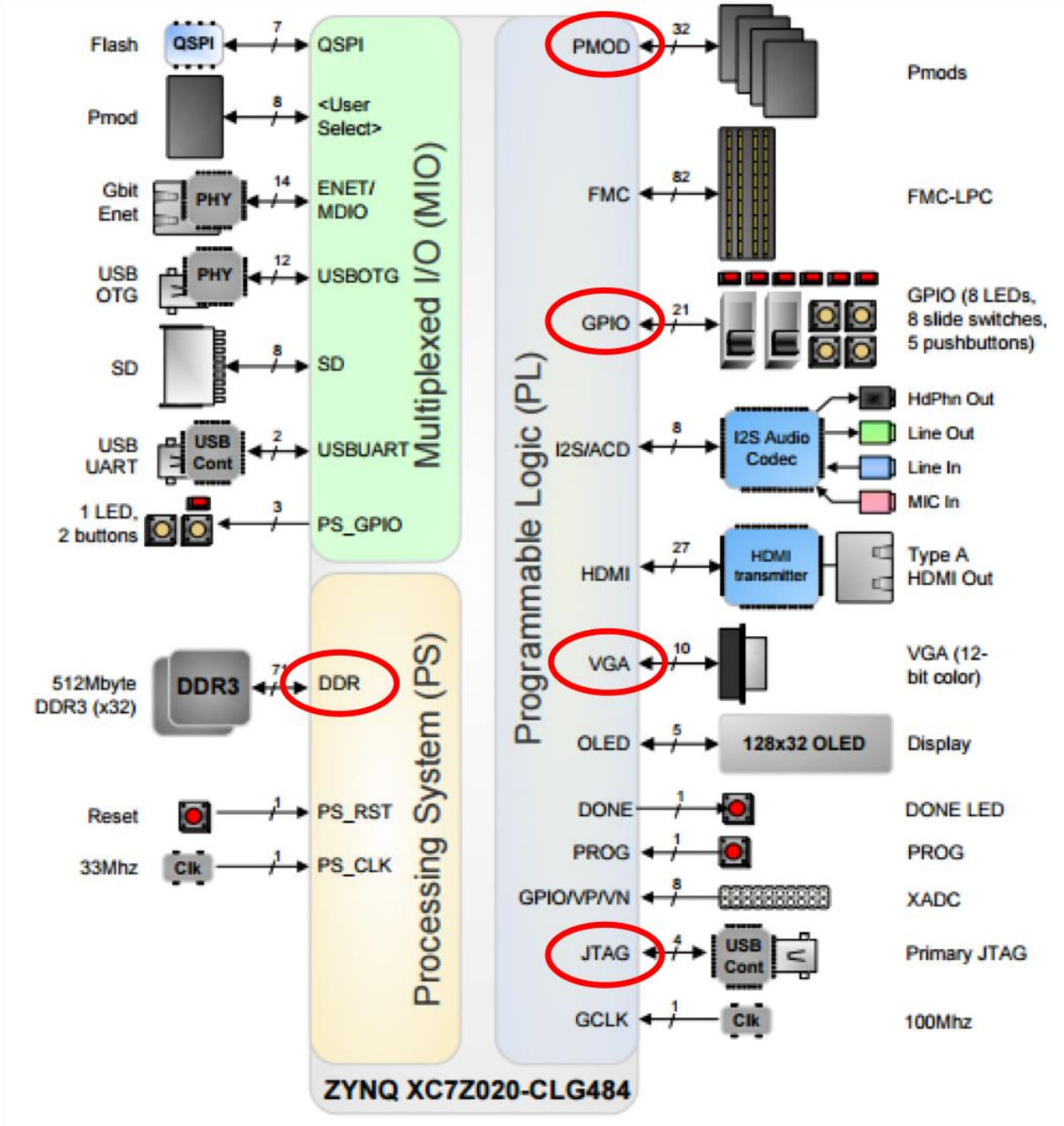

Figure 12: Zedboard Block Diagram. Integration of the Processing System, Programmable Logic and the Multiplexed I/O. This includes all of the hardware resources that are present on the Zedboard for user implementation. Hardware components utilized in this project are marked in red. [57]

The PMOD, GPIO, and VGA were components that the team previously learned to implement in courses, which further influenced the choice to consider this board for potential use. The JTAG interface allows for USB programming, and the DDR3 would provide more than enough memory for any processing the overall system may require [57].

Interfacing with the different sensors and peripherals that were chosen as a result of this project provided further functionality for the overall design. Possible sensors that were identified for use included radar, LIDAR, heat sensors and various cameras. The Zedboard provides ample computing power for the sensors investigated. For example, a radar application on the Zedboard would have the ability for real time continuous wave frequency modulation in which it would constantly scan the echo received and process the information [58]. This showed that the use of an FPGA, specifically the Zedboard, would be a more than viable option in properly connecting all the peripherals, making the correct logic based decisions, and decision making based on the data processed.

This project aimed to implement a partially autonomous bus. More specifically, this stage of the project focused on developing an effective sensing system that will detect if a pedestrian is too close to the side of the bus when it is making a turn. In order to develop an effective system, it was important to determine what kind of monitoring sensors to use. The sensors are the most critical feature of the system because they are essentially the “eyes and ears” that the autonomous system relies on to make effective decisions. The final system aimed to involve a combination of different types of sensors. This section focused on exploring the use of a camera for pedestrian detection.

Two main features for a pedestrian detection system is to have a zero probability of miss detection and a very small probability of false alarms. For example, if the person is in the blind spot, the system must always detect the person. However, it is acceptable to detect another object that is in the blind spot as long as the probability of false alarm is very small. Since not many moving objects have a face, the idea of face detection came to mind. If the detection system is able to detect that the object inside of the blind spot has a face it is probably detecting a person. Therefore, it was important to explore the use of cameras in face detection.

Face detection is implemented through some type of image processing software that can locate human faces in an image regardless of their position, scale, in-plane rotation, orientation, pose and illumination [59]. According to Ming-Hsuan Yang from Honda Research Institute in

California, the face detection can be very difficult due to the pose of the subject, the presence or absence of structural components, such as beards and glasses, the pedestrian's facial expressions, the orientation and the imaging conditions such as lighting, camera characteristics and resolution. As a result, the problems of face localization and facial feature extraction arise [59]. Since the face detection system is operated in real time, the search strategy, the speed and the precision are the main constraints for the effective pedestrian detection. The next part of the paper explores the concurrent face detection software.

There are many different face detection softwares available. The main differences are in terms of algorithms they implement, real-time video processing capabilities as well as operating languages that they are written in. The nature of such software will determine the type of hardware and resources that would be used in order to implement the face detection system. This section looks into two contemporary face detection software, Google Mobile Vision and OpenCV.

Google Mobile Vision Face API is a free Android and iOS library developed by Google software developers. The Google Face API finds human faces in photos, videos or live streams. The software is also capable of finding and tracking positions of facial landmarks such as the eyes, nose and mouth [60]. Therefore, it is obvious that this API could be used in the autonomous bus detection system to provide facial detection when an object is inside of the blind spot. The advantages of using Google Face API are that there is a lot of support and it is fairly easy to program because a high level object-oriented programming language, such as Android Studio, could be used. The immediate disadvantage of using the software is that it would require an Android device in order to implement. This means that the hardware that implements the camera would have to be capable of running Android OS. Usually, only consumer electronics, such as smartphones, run Android OS and it is clear that this is a very important limitation.

OpenCV stands for Open Source Computer Vision Library and it is developed under Berkeley Software Distribution (BSD) license which makes it free for academic and commercial use [59]. According to the official website [61], OpenCV is an open source computer vision and machine learning software library, with over 2,500 optimized algorithms, that can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements and track moving objects. It is obvious that this software library could be used to develop a pedestrian detection system. The advantages of using such software is that it has C++,

C, Python, Java and MATLAB interfaces and it supports Windows, Linux, Android and MAC OS. Furthermore, OpenCV is specifically designed for real-time vision applications which is exactly what the detection system for this project requires. Therefore, the hardware implementation can involve a Raspberry Pi running on Linux distribution and the face detection library can be implemented using Python programming language. Furthermore, since the library is licensed under BSD license the libraries are free to modify to suit the most optimized design. Similarly an FPGA can connect to a Raspberry Pi, and also run OpenCV software. This shows various implementation strategies that can be utilized in making the overall design function properly.

FPGA's have successfully been used for image processing and pedestrian detection in the past. There are numerous instances of FPGA based pedestrian detection that takes an image file and scans through it based on Histogram of Oriented Gradients (HOG). This method scans through an image using a box that slides from left to right and based on detectors of wanted and unwanted objects within the image it will create a detecting box around the desired objects. Usually this results in multiple boxes that detect the same object so this is averaged and the largest sample is taken [62]. Research into the OpenCV database showcased some implementations that had greater than 99% accuracy for person detection [63] However, this was not in real time, and scanned still images that came from a picture file, and not directly from a camera. The final design would require image processing within real time, and on a continuous basis from a camera feed. A recent MQP showcased a proof of concept for real time image and distance data processing that interfacing with an FPGA can provide [56]. The architectural design for the board would require an implementation of parallel processing capabilities. The programmable logic on the hardware would need to manage the peripheral sensors connected to the board and be able to transfer the image and sensor readings to the software functionality, the dual-core ARM Cortex A9 processor for the Zedboard, which would then handle the algorithms for the actual pedestrian detection.

2.9: Sensor Technologies Considered for System

For this project, it was necessary to analyze a certain number of sensors or technologies that could be used to detect human presence in the specified zone. Research for the appropriate technology brought about various considerations for possible technologies that could be

implemented for this project. Technology that was researched included: infrared, ultrasound, thermal sensor, sonar, LIDAR, and radar.

Radar, radio detection and ranging, is another sensor commonly found on autonomous vehicles and it has been used throughout the automotive industry for quite some time [64]. The major reason for the success story of automotive radar is its physical principle that offers unique performance features at reasonable costs. It continues to be used for collision avoidance systems such as for self-parking and automatic braking. Radar can be used to determine the proximity, range, speed, and size of an object [65]. Due to the radio frequency technology and solid-state design, radar has an advantage over other sensors because it is more reliable in fog, rain and snow [66]. In addition, it is capable of virtually looking through vehicles by exploiting reflections between the road surface and vehicle floor and hence makes the invisible visible [66]. Therefore, it was obvious that the radar technology might be essential to develop an effective sensing system for this project.

Radar technology has paved the way for vehicles to be autonomous. The introduction of semi-autonomous emergency braking and pre-crash systems was only possible by a dramatic improvement in radar technology and radar network architecture [8]. Those improvements include: a 250 meter multimodal range covering both long and short distances; Synthetic Aperture Radar (SAR), that is used to create images of objects in either 2 dimensional or 3 dimensional representations; and a high angular accuracy with a fast update rate of few 10's of milliseconds (ms) and a small latency of a few milliseconds [8] [67] [68].

It is not surprising that the radar technology is becoming a standard in autonomous vehicles. According to Digikey, the Advanced Driver Assistance Systems (ADAS) are increasing the adoption of 24GHz radar sensors in autonomous vehicles [69]; which has induced falling prices of radar sensors. Furthermore, ADAS that utilize radar sensing for adaptive cruise control and collision detection are becoming a requirement for car manufacturers to achieve the highest five star New Car Assessment Program (NCAP) safety rating in Europe [70]. It is obvious that these examples further enforce the idea of using radar when developing a detection system for automotive vehicles. The CDM324 is a relatively cheap 24GHz radar module available for purchase. The module's specifications are shown in Table 10.

Table 10: CDM324 Radar Module Specifications. These are the specifications of the radar module that was considered as one of the sensors for the initial prototype of the system. Despite the capabilities of this sensor, which made it acceptable for potential use, radar technology was ultimately not chosen in the final design in favor of LIDAR technology. [71]

CDM324 Specifications	
Transmission Frequency	Min: 24GHz Typ. 24.125GHz Max. 24.25GHz
Output Power	Typ. 16dBm
Antenna Shape	Horizontal: Typ. 80 degrees Vertical: Typ. 32 degrees
Side Lobe Suppression Ratio	Horizontal: Typ. 13dB Vertical: Typ. 13dB
Operating Temperature	Min. -20 degrees Celsius Max. 60 degrees Celsius
Operating Current	Typ. 30mA Max. 40mA
Pulse Width	10us
IF Output	Min. -300mV Max. 300mV
Size	25 x 25 x 7mm

This sensor can be interfaced with a microcontroller and by using the Doppler Effect it is possible to detect a moving object and its velocity [71]. Unfortunately, this system is not reliable for detecting slower speed objects [71]. Therefore, this radar sensor was not the best solution for pedestrian detection.

According to Digikey, the newest radars for autonomous automotive applications are operating at 77 GHz band of frequency and provide a longer range and higher resolution [69]. This type of technology allows for multiple object detection in real time [69]. Unfortunately, this technology is still in the development stage and there is not an affordable complete module that could be easily interfaced with the rest of the system. Since building such a sensing system could become a project of its own, an alternative technology was explored.

Light Detection and Ranging (LIDAR) is a common sensor found on many autonomous vehicles such as those used in Defense Advanced Research Projects Agency (DARPA) competitions [72]. This system provides a 3D or 2D reading of the field that its laser space covers [73]. Using the data from the reading, the processor is capable of implementing object identification, motion vector determination, collision prediction, and avoidance strategies [64]. Essentially LIDAR gives an autonomous vehicle obstacle and location perception.

Many autonomous vehicle companies choose LIDAR because it can provide a high resolution up to a range of 60 meters, and it can be integrated with other kind of sensors for more accurate measurements [72]. Low cost LIDAR sensors can only give a horizontal field range of 180 to 270 degrees, but only provide single layer measurements. The problem with this is that determining the heights of objects and the detecting the profile of the road can be difficult. To alleviate this problem many autonomous vehicles use LIDAR systems that have horizontal range of at least 270 degrees and 32 to 64 layer measurements [8] [72]. One of the most known LIDAR systems is Velodyne's HDL-64E with the most famous user being Google's self-driving car [67].

By using a high quality LIDAR system an autonomous vehicle can map out its surroundings and plan a route based off of this. This has been accomplished by an unmanned aerial vehicle (UAV) operating without GPS. Testing and research has shown that if a UAV or any other kind of autonomous vehicle was trapped in an area without GPS, laser scans of its surroundings can be used to plan a safe escape route [74]. LIDAR data can be mapped out on a Cartesian plane which can be processed by another system aboard an autonomous vehicle, it can create a 3D cloud [64]. The information from the LIDAR can be used to plot an object on a Cartesian plane [72].

There are a few drawbacks to using LIDAR. The first being that LIDAR is unable to recognize the object class [75]. LIDAR cannot tell the difference between a mailbox and a pedestrian. To combat this problem a more complex system with more sensors such as radar and cameras would be needed in order to confirm that an object is a pedestrian. Another drawback is that LIDAR measurements tend decrease as range increases. Experimental results showed that the height reading of an object was twice as inaccurate when the range was increased from three meters to seven meters [67]. The other drawback is that high quality LIDAR systems are very expensive. Velodyne's HDL-64E can cost as much as $80,000 with the next model down costing $40,000 and the lower quality model from the company costing $8,000 [67]. As previously stated, low cost LIDAR systems provide a narrower horizontal field and a single layer measurement which provides less details about the surrounding. Any company or team working on autonomous vehicle with a tight budget may find the use of LIDAR systems not feasible.

The current available LIDAR systems are expensive and very costly for a project like this one. However, there are some startups that are trying to push the new, low-cost, LIDAR systems on the market. According to IEEE Spectrum [76], the recent academic and industry research

focuses on making cheaper LIDAR sensors while decreasing their footprint. For example, a startup from Israel, Innoviz, is promising a "high-definition solid-state LIDAR" with better resolution and a larger field of view than the concurrent LIDAR sensors [76]. Furthermore, Innoviz is promising a per chip cost of 100 USD [76]. While this type of sensors seems like a viable option it is important to note that the working prototype has yet to be released in 2018 [76]. Therefore, this type of technology should be kept in mind for the future improvements of the sensing system.

A LIDAR sensor that has significant potential is the recently released Sweep LIDAR made by a California based startup, Scanse [77]. The sensor was released in March 2017, and it can be pre-ordered at relatively low cost of $350 [78]. According to Scanse [78], Sweep LIDAR sensor has scanning capabilities that allow on the fly adjustment of the rotation speed. When the rotation speed is slowed down, it is possible to extract more details about the surrounding. Similarly, fast reaction times are obtained by increasing the rotation speed [78]. More detailed specifications for this sensor can be seen in Table 11.

Table 11: Sweep LIDAR Sensor Specifications. Showing the range, the Field of View and the Electrical specifications. These were considered when choosing a potential LIDAR for the first prototype. The long 40m range and 360-degree field of view of this sensor show the strong capability of a LIDAR sensor for object detection. [78]

Specification	Value
Horizontal Field of View	360 degrees
Vertical Field of View	0.5 degrees
Physical	
Specification	**Value**
Weight	120 g (4.23 oz.)
Operating Temperature	-10 to 60 degrees Celsius
Storage Temperature	-40 to 80 degrees Celsius
Electrical	
Specification	**Value**
Power	5 VDC
Current Consumption	Up to 650 mA 450mA nominal
Measurement Performance	
Specification	**Value**
Range (75% reflective target)	40 m (131 ft)
Resolution	1 cm (.4 in)
Update Rate (75% reflective target)	Up to 1075 Hz

Another low cost LIDAR is RPLIDAR A2 manufactured by Slamtec. The RPLIDAR A2 is a 360 degree 2D laser which can scan to within a 6 meter range [79]. The sensor generates 2D point cloud data that can be used in localization and environment modeling. With a relatively low cost of $375 it was obvious that this LIDAR solution might be a viable option for this project. The scanning frequency of this device can be freely adjusted for more detail or faster detection, depending on the application. Furthermore, the device is specifically designed to have a long life span which is a vital characteristic for this project.

Figure 13: RPLIDAR A2 LIDAR Sensor. The sensor features a laser transmitter and receiver for determining object distance through reflections. The sensor's housing is able to rotate 360 degrees on the base for maximum field of view. [79]

Low-cost LIDAR sensing technology is still in the development phase; however, it does show a lot of potential. The current technologies proved to be very expensive; however, some startups are coming up with relatively cheap solutions. Table 12 demonstrates the comparison of the two LIDAR sensors.

Table 12: Sweep vs. RPLIDAR A2 Specifications. The specifications were analyzed and compared to determine which one would be the most efficient LIDAR sensor for the system. At only a slightly higher price, the SLAMTEC RPLIDAR A2 features a much faster refresh rate within a range acceptable for this application. [78] [79]

Features	Scanse	SLAMTEC
Model	**Sweep**	**RPLIDAR A2**
Price	$350	$375
Update Rate	500Hz	4000Hz
Recommended Scan Rate	2.5 Hz	10 Hz
Range Indoors	50m	0.15 – 6m
Range Outdoors	40m	
Range Accuracy	1-2% of distance	0.2% of distance
Range Resolution	1cm	1cm
Horizontal FoV	360 deg	360 deg
Vertical FoV	0.5 deg	N/A
Power	1W @ 5V	1.2W @ 5V
Weight	120g	170g
Interface	UART/USB	UART/USB
Operating Temp.	-10 to 60 C	0 to 45 C

Low-cost LIDAR solutions are becoming increasingly accessible. The Sweep LIDAR sensor made by Scanse could potentially be an appropriate choice for this application. However, this product is in the beta phase and it has yet to be perfected. If the Sweep sensor proves to be robust, it could become a vital part of the pedestrian sensing system for this project.

If used, any LIDAR sensor must prove to be very robust and immune to different mechanical vibrations and environmental conditions since it will be implemented on the side of a bus. Therefore, a solid-state LIDAR comes to mind. Solid-state LIDAR technology has no moving parts which enables very robust and reliable design at lower costs. This technology is still in its development stages and there are not many solutions on the market. However, LeddarTech, LIDAR sensing technology company from Quebec City, Canada, announced one of

the first solid-state LIDAR sensors available for purchase in September 2016 [80]. The LeddarVu8 LIDAR sensor has a high degree of modularity, which enables very flexible integration to meet the system's specifications [80]. LeddarVu8 is shown in Figure 14.

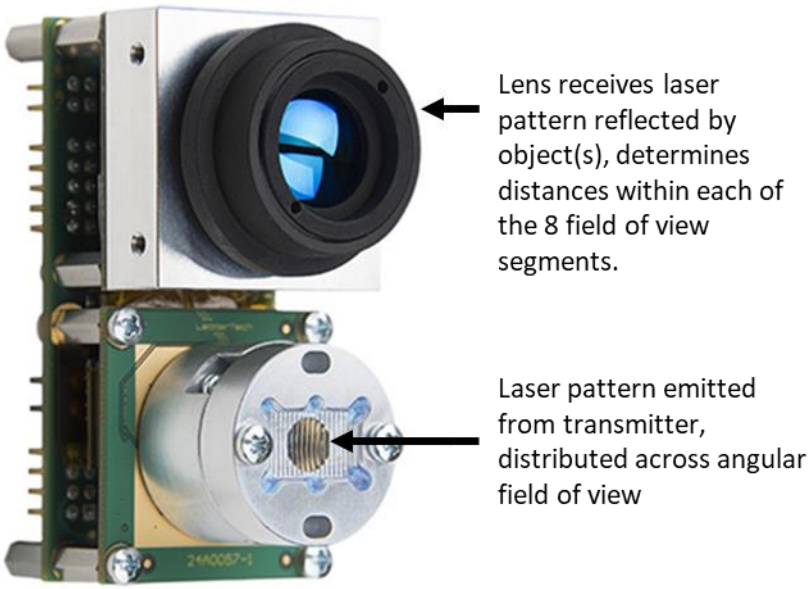

Figure 14: LeddarVu8 LIDAR. This is a solid state LIDAR sensor, featuring up to 100 degrees of horizontal Field of View (FoV). The FoV is divided into 8 segments and a distance is measured for the closest object in each segments. This device was chosen to be a part of the system for its robustness and individual segment readings. [80]

According to the manufacturer's website, LeddarVu8 "leverages powerful class-1 laser illumination and eight independent active detection elements into a single sensor, which results in rapid, continuous and accurate detection and ranging of objects" [81]. Furthermore, LeddarVu8 has an advantage over conventional LIDAR because it has no moving parts which makes it very robust [81]. Also, due to its modular design it is possible to choose the hardware that gives the best possible Field of View (FoV) for the desired application. Depending on the FoV, the sensor is capable of detecting objects up to 215 meters away with multi-object detection [81]. The LeddarVu sensor has an operating temperature of -40℃ to 85℃ which makes it great for the outdoors use [81]. This sensor is also immune to ambient light and works in all weather conditions [81]. It was obvious that this sensor had all the characteristics for the pedestrian

detection system. The only downside was the $475 starting price, which makes it more expensive than the previous LIDAR solutions [81]. Table 13 shows the LeddarVu8 specifications.

Table 13: LeddarVu8 Specifications. Showing the range, the Field of View and the Electrical specifications. These were considered when choosing a LIDAR for the first prototype. The capabilities of this sensor were impressive when compared to the other two aforementioned LIDAR sensor options, as it is a non-moving solid-state LIDAR technology. [81]

LeddarVu Specifications:	
Number of Segments	8
Vert9cap FoV options (degrees)	0.3, 3
Beams (degrees)	20, 48, 100
Interfaces	SPI, USB, CAN, Serial (UART/RS-485)
Wavelength	905 nm
Power supply	12V
Weight	14g
Detection range	0 - 40 m
Accuracy	5cm
Data refresh rate	Up to 100 Hz
Operating temp. range	-40 to 80 degrees C
Distance precision	6 mm
Distance resolution	10 mm
Power consumption	2 W

A cheaper alternative is LeddarOne, also manufactured by Leddar Tech [82], with a starting price at $170 [82]. This sensor is different from LeddarVu because it is only capable of a single point measurement [82]. However, due to its RS-485 communication standard option, in combination with MODBUS, it is possible to integrate multiple of such sensors within an RS-485 network [82]. It is obvious that this approach would require more focus into design of a multi-object detection system, which could become a very complex project of its own. Therefore, it was more feasible to choose the LeddarVu8 over the LeddarOne. LeddarOne specifications are shown in Figure 15 and Table 14.

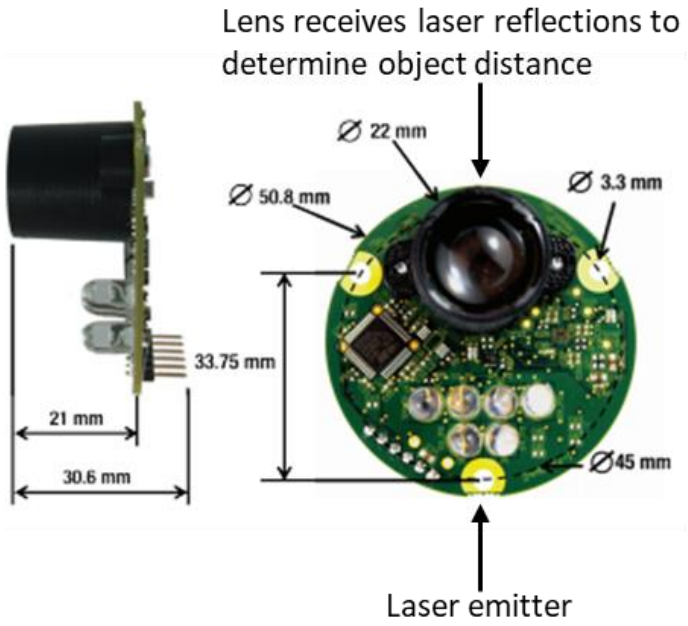

Figure 15: LeddarOne Sensor. This is solid state LIDAR sensor capable of single point measurement. This module was considered when choosing a LIDAR sensor for the first prototype [82]

Table 14: LeddarOne Specifications. Showing the range, the field of view, and the electrical specifications. These were considered when choosing a LIDAR for the first prototype. [82]

LeddarOne Specifications:	
Beam	3 degrees
Interfaces	3.3V UART or RS-485
Wavelength	850 nm
Power supply	5V
Diameter	60.8 mm
Weight	14g
Detection range	0 - 40 m
Accuracy	5cm
Data refresh rate	Up to 70 Hz
Operating temp. range	-45 to 80 degrees C
Distance precision	5 mm
Distance resolution	3 mm
Power consumption	1.3 W

2.10: Camera Sensor Technologies

Cameras can give a clear and accurate picture of a bus's surroundings. An important requirement for using cameras was that there must be a system on board that can process the images from the camera such as system that uses software such as OpenCV. Two types of cameras could be used, a standard camera that uses visible wavelengths or a thermal imaging camera. It was necessary for this image processing to be independent, as the driver having to look at a monitor hooked up to the live feed from the camera system would distract them from focusing on the road. This system needed to be designed as an aid, and limit the distraction caused by it.

A real-time vision algorithm could be created, using OpenCV, to detect and track pedestrians but there were many variables that would cause challenges. Some of the variables included clothing, human posture, lighting conditions, the background, etc. A common algorithm found in many object detection applications is the Histogram of Oriented Gradients (HOG) feature descriptor using a Linear Support Vector Machine (SVM) [83]. The HOG feature descriptor provides superior results because it highlights the head and shoulders clearly, the most relevant parts of the human body [84]. Studies have shown that a detection window of 64x128 pixels should be used in the algorithm [83].

The first step in this algorithm would be determining the histogram of oriented gradients to generate the features necessary for human detection. The features would be determined by solving for the histogram of oriented gradients of smaller rectangular regions of the detection window as it moves across the image. This process would scan through the entire image. One experiment used OpenCV's AdaBoost, a two-class classifier. AdaBoost is primarily used for system training and classification purposes. The training purpose allowed for old images to be stored and compared to new images to aid in the classification purpose which determines the identifier features that would need to be scanned for. AdaBoost was used to determine if a pedestrian was present and then Linear SVM was used to confirm the classification from AdaBoost. If AdaBoost and Linear SVM both confirmed a pedestrian was present then both detection windows used in AdaBoost and Linear SVM were merged. When compared to the default human detector in OpenCV this experiment had a faster processing rate but operated at a lower precision. The authors of the experiment noted that the precision rate could have been improved if their system was trained with more images contained with negative results.

OpenCV's default detector used 12,180 images with negative results compared to the experiment which used 7,308 images with negative results [85].

The use of thermal cameras was also a possibility. Specifically, far-infrared (FIR) camera systems have shown a level of quality and functionality that would be necessary for the project. It captures thermal radiations that is in the wavelength of 8 to 12 micrometers. The thermal radiation of human body falls within this range. When compared to close range infrared cameras, FIR cameras do not require additional illuminators and the impact of absorption and scattering of infrared signals due to fog is less severe. A FIR camera and standard camera can be used together to scan the same image when calibrated properly. Through research, an experiment was found, which was conducted using this setup in various conditions such as the morning, night, afternoon, and a rainy day. To calibrate the cameras, 20 ground-truth points where compared from the images from both cameras. The results showed that the thermal cameras was roughly 97 to 99 percent accurate during the daytime. During a rainy day, it was 99.63 percent accurate. The total processing time of this experiment was roughly 23 milliseconds [86]. This experiment used a different method compared to most human identification software that exists. In this experiment, the algorithm uses background subtraction of the thermal image and visible light that is apparent within the image to determine if a pedestrian is present within a particular region. Then the algorithm focuses on that region and determines if a pedestrian is present by using thermal imaging. If a pedestrian is detected across all of the stages then the algorithm merges the visible and thermal imaging positive images to give a clear and concise result [86].

Using standard and thermal cameras instead of other technologies to detect humans has its benefits. Camera sensors with the right detection and tracking software can be programmed to differentiate pedestrians from other objects. This is particularly useful in urban environments where there are many obstacles besides pedestrians [85]. The only drawback is that the type of algorithm, and computing power and architecture used can greatly influence the efficiency of the system along with the other variables mentioned earlier in this section [83].

2.11: Ultrasonic Sensors

Ultrasound technology is commonly used in automobile sensors to measure the position of objects in close proximity to the vehicle. The term "Ultrasound" refers to a high-frequency acoustic wave, generally above 20 kHz, which exceeds the human range of hearing [87].

Ultrasonic transducers are available as either transmitters, receivers, or transceivers. An ultrasonic transceiver is able to both transmit and receive ultrasonic waves. A matched pair of transmitter and receiver, or a single transceiver unit, can be used to transmit ultrasonic sound waves into a space and measure the waves that are reflected back. This enables ultrasonic technology to be used for measuring proximity of nearby objects. By calculating the change in proximity of a target object over time, ultrasonic sensors can also be used to measure an entity's speed. One advantage that ultrasonic transmitters and transceivers have in detecting object distance is that "ultrasonic sound waves can be produced with high directivity" [87]. By forming a direct beam of sound, as opposed to a wide-angle transmission, ultrasound sensors are effective in producing a precise linear distance reading. As shown in reference [88], there was a report released on the experimental testing of directivity patterns in piezoelectric ultrasonic transducers. Three transmitter-receiver pairs were tested individually in a tank of deionized water, with the transmitters fixed at various angles relative to the receivers, ranging from 0° to 10° in 0.1° increments. The operating frequencies of the three pairs were set at 2.25 MHz, 3.5 MHz, and 5 MHz. By testing each sensor at multiple angles under the same conditions, the amplitude of received waves could be measured to analyze the effect of transmission frequency on the directivity of ultrasonic sensors. The results of these experimental trials can be found in Figures 16, 17 and 18.

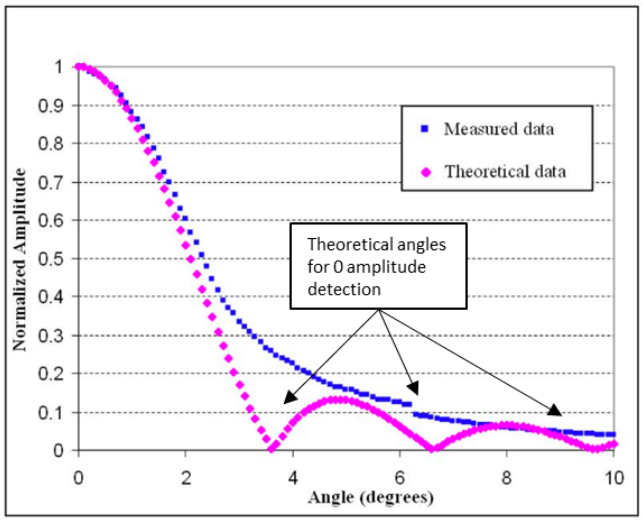

Figure 16: Ultrasound Directivity Test, at 2.25 MHz. A comparison between the actual measured data received from the Ultrasound sensor, and the theoretical predicted data. The theoretical zero amplitude "dips" indicate where the receiver was predicted to receive no signal from the transmitter, due to the directed angle. The behavior likely changes due to reflections of ultrasonic waves within the water tank. [88]

The normalized amplitude measurements of the 2.25 MHz sensor test shown in the Figure 16 demonstrated a relationship between the angle of the transmitter's placement relative to the receiver and the strength of the detected ultrasonic signal. As the transmitter is turned away from the receiver, the measured amplitude drops in magnitude. This reduction becomes most apparent after roughly three degrees.

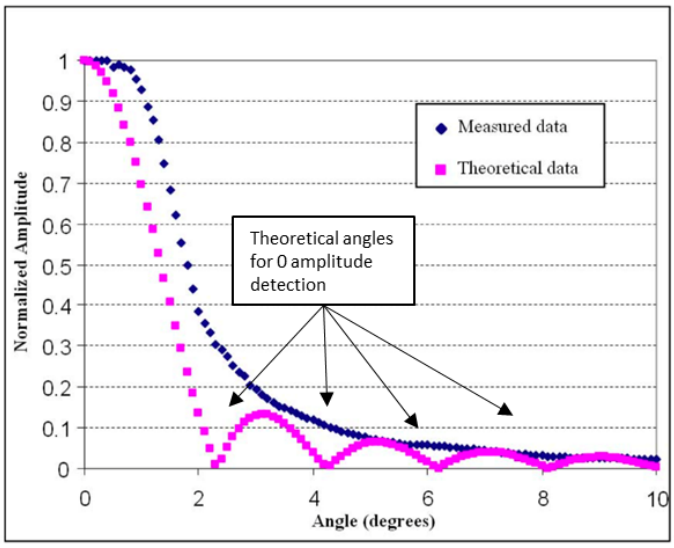

Figure 17: Ultrasound Directivity Test 3.5 at MHz. A comparison between the actual measured data received from the Ultrasound sensor, and the theoretical predicted data. The theoretical zero amplitude "dips" indicate where the receiver was predicted to receive no signal from the transmitter, due to the directed angle. The behavior likely changes due to reflections of ultrasonic waves within the water tank. [88]

The 3.5 MHz test revealed a sharper decrease in normalized amplitude as the transmitter was rotated. This graph indicates that an increase in frequency also increased the directivity of the ultrasonic beam, as the received signal was now weaker at narrower angles when compared to the 2.25 MHz trial. The normalized amplitude now shows most significant decrease at an angle just below two degrees.

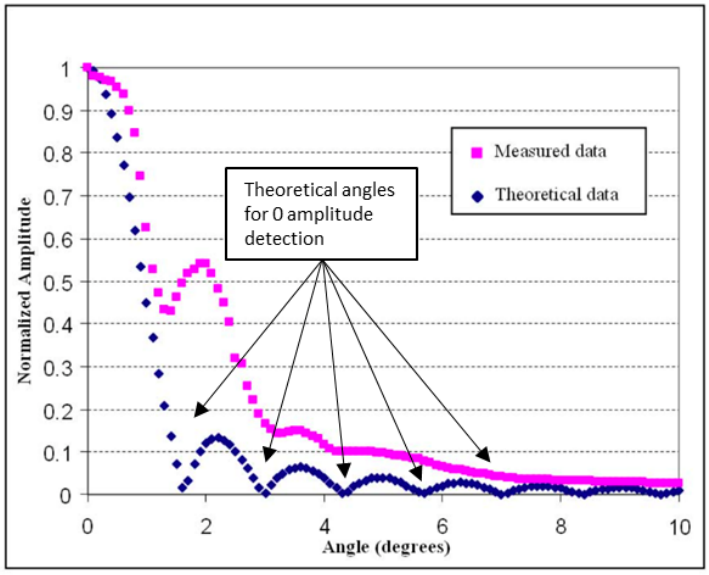

Figure 18: Ultrasound Directivity Test at 5 MHz. A comparison between the actual measured data received from the Ultrasound sensor, and the theoretical predicted data. The theoretical zero amplitude "dips" indicate where the receiver was predicted to receive no signal from the transmitter, due to the directed angle. The behavior likely changes due to reflections of ultrasonic waves within the water tank. [88]

The 5 MHz sensor test continued this trend, and further confirmed that an increase in frequency causes a more directional ultrasonic transmission. A rapid decrease in signal amplitude was observed at roughly one-degree rotation.

It can be concluded that the frequency of a chosen ultrasonic sensor would be important if in incorporating ultrasound technology into a pedestrian detection system. If ultrasound were to be used solely for finding the range between the bus and a target, a high frequency sensor may be useful because it could pinpoint a specific area and return a linear approximation. In the case where ultrasound were to be implemented for detecting objects in a conical angle, a lower frequency transceiver would most likely be the better option. However, the directivity of an ultrasound sensor would be best paired with another technology, designed to detect objects within a wide angle. A separate technology could first detect an object within close proximity of the bus, and then the ultrasound sensor could be used to more accurately determine the distance of the object in question.

One current application for ultrasound in the automotive industry is the acquisition of distance readings between a vehicle and surrounding structures. Ultrasonic sensors help to enable

autonomous and semi-autonomous features in cars, such as obstacle alerts for assisted parking. Reference [89] shows a report on multisensor integration techniques for an autonomous ground vehicle project. This study focused on the performance of both individual sensors and multisensor arrangements when used for the navigation of a vehicle among structured surroundings [89]. The autonomous vehicle interfaced with ultrasound transceivers amongst four other sensing technologies. The ultrasound sensors chosen were two MaxBotix MaxSonar-WRS 7384 ultrasonic sensors. The MaxBotix 7384 sensor can be used as a rangefinder, measuring the "time of flight" for sound waves transmitted towards and reflected back by objects within the sensor's line of sight [89]. Based on the time between transmission and reception of the reflected signal, the sensor then calculates and outputs a range reading. The two ultrasonic sensors were used to monitor the surroundings of the front-left and front-right sides of the vehicle. If an object were found in close proximity of the vehicle's path, these sensors would provide distance information to the remote motion controller that could adjust the steering accordingly. At fixed range tests from 2 to 7m, the ultrasonic sensors were able to detect objects, however, they became less accurate as the range increased [89]. The findings of this study indicated that ultrasound technology is most useful for obstacle detection applications, while other technologies, such as cameras and Global Positioning System, are better suited for determining the vehicle's navigation path.

Based on the studies mentioned, the team concluded that ultrasonic sensors were most practical for range finding applications, where a narrow, linear beam is used to determine the distance between the mounted location of the sensor and an object in its direct line of sight. When mounted on a bus, an ultrasonic sensor could be used to detect the distance between a person and the bus's surface. This information could be processed by the central computing system to determine if the person is in the dangerous zone of the bus's turn path. Multiple ultrasonic sensors were investigated for potential use in this project.

The MaxBotix MaxSonar series ultrasonic sensors are available in many different configurations for a wide range of applications. The MaxSonar series sensors are suitable for outdoor weather conditions with IP67 dust and water resistance ratings, and most are able to operate in a wide range of temperatures, as low as -40 °C and as high as 70 °C. The MaxSonar WRS 7384 model used in the aforementioned project was evaluated for potential use for the autonomous bus application, however, two major shortcomings were found. According to the

WRS 7384 datasheet, this sensor is rated for a theoretical maximum of 5 meters (~16.4 ft) in range, with a maximum refresh rate (data acquisition and serial output) of 6.67Hz [90]. These performance ratings presented two areas of concern to be considered in the consideration of an ultrasonic sensor choice: the range at which a sensor would be able to realistically detect pedestrians and the speed at which the range readings could be reported to the central data fusion and decision making system. While 5 meters is roughly the width of the zone to the side of the bus that the sensor technologies would aim to cover, it would be preferable to avoid limiting the maximum range of the ultrasonic sensor to only 5 meters. A longer theoretical maximum range would ensure that the sensor could cover such distance, with the capability of detecting targets further from the sensor depending on the configuration. For example, if the sensor were to be placed at an angle on the bus's side, rather than facing outwards perpendicularly, then a 5-meter range sensor would not be able to fully reach the 5-meter edge of the required zone. A longer-range sensor would provide the option of being placed in a greater number of configurations, in that it would be able to cover 5 meter from the bus at an angle. The refresh rate of the sensor was also a concern because the speed of the pedestrian detection system is an important factor in avoiding collisions. In the case of the WRS 7384, 6.67 Hz could pose latency issues. If the bus is taking a sharp turn into a side street as a pedestrian begins to cross, roughly 6.67 data samples per second may not be fast enough to provide ample warning in the case of an imminent crash. At this refresh rate, a theoretical maximum latency of 150 milliseconds would be added to the overall system when relying on the ultrasonic sensor's measurements.

The MaxBotix MaxSonar-WRLS 7363 ultrasonic sensor is very similar to the WRS 7384, yet it supports longer range transmission. The WRLS 7363 model can identify the distance of an object up to a maximum of 10 m (~32.8ft) away [91]. This distance would enable the sensor to cover critical areas along side of the bus from a wide range of positions and angles. However, this model also shares a flaw with the WRS 7384: a rather low refresh rate of 6 Hz for data acquisition and serial transmission. The WRLS 7363 could prove to be useful in a situation where range is significantly more important than speed. Such situation would only occur if the ultrasonic sensor would not be a primary sensing technology for detecting a pedestrian, and is instead used for determining a range reading for an already acquired target. This refresh rate issue brings concern to ultrasound's potential for an autonomous pedestrian detection system. However, there is one sensor manufactured by MaxBotix that could prove useful for this

application. Also within the MaxSonar series, the WR 7040 model makes use of I2C communication to support a 40 Hz data refresh rate, over six times faster than the 7363 model [92]. The I2C interface also enables the integration of multiple 7040 models with the use of only two wire lines, which would be helpful in a configuration consisting of more than one ultrasonic sensor. I2C would require different methods of integrating this sensor with the central data processing system than a similar serial peripheral interface device, however it is likely possible to achieve. The WR 7040 has an effective range of 7.65 m (~25 ft), with 1 cm resolution readings. An image of the sensor, as well as a beam pattern diagram from the official datasheet can be found in Figure 19.

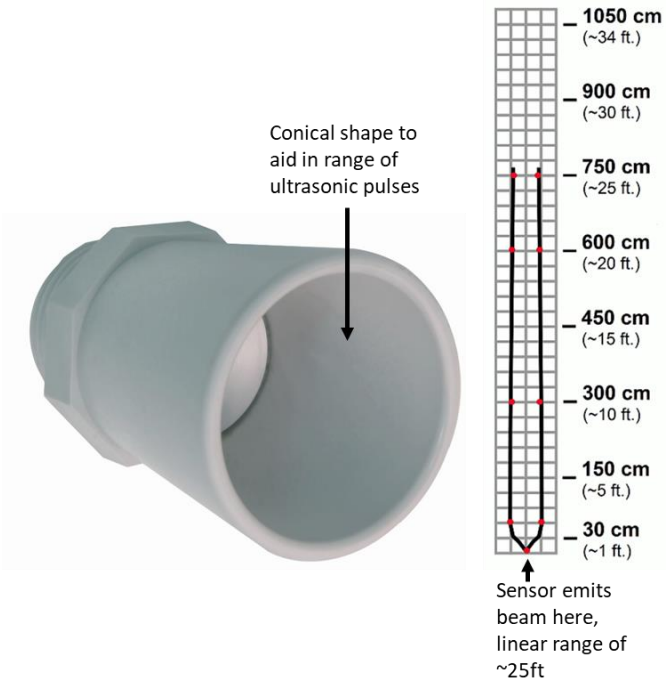

Figure 19: MaxBotix WR 7040 Ultrasonic Sensor & Beam Pattern. Pattern shows how the beam propagates from the sensor's opening. The sensor's conical shape assists in the directivity and range of the ultrasonic pulses. [92]

The range and refresh rate specifications of this sensor make it a better choice than the WRLS 7363 or the WRS 7384, as it supports just over ¾ of the range of the longest range WRLS 7363 at a significantly higher refresh rate for data acquisition. This sensor is capable of the convenient all-weather features that the MaxSonar series supports, which is beneficial to this application, where the sensor could be externally mounted on a city bus. The power requirements

of the WRLS 7363 is rather low, with an average current draw of 3.4 mA and 3.0V-5.5V operating voltage range. Conclusively, due to the capability of outdoor operation, suitable range, and comparatively fast data refresh rate, the ultrasonic sensor of choice for a pedestrian detection system would be the MaxBotix MaxSonar WR 7040.

2.12: GPS in Autonomous Vehicles

The primary sensor technology needed for any guidance system is location identification [65]. Global Positioning System has been used for many years to navigate the particular position of a vehicle. GPS technology has been successful in location accuracy on the order of one meter [65]. This project requires a greater degree of precision, and GPS is not a system that could work alone. Over the years, GPS has been able to evolve into more efficient subsystems that can be integrated into various designs. GPS can be produced on a chip (SoC) IC or multiple chips. These chips consists of an internal RF preamp for the 1.5-GHz GPS signal and an embedded, application-specific computed engine to perform the intensive calculations of the subsystems [64]. Typically, these antennas are on the roof of a vehicle with a low-noise amplifier RF preamplifier in order to locate the GPS circuitry in a more convenient location within the vehicle. In order to implement these chips the use of modular GPS solutions, for overall location awareness, are a good choice [65]. An example of a typical GPS module is the RXM-GPS-F4-T. It requires a single 1.8V supply at 33 mA, and has the ability to track up to 48 satellites simultaneously [64]. This is significant in that more channels will be able to allow the GPS to see and capture more data and ultimately yield better results and fewer dropouts. After it computes location based on the GPS received signal, it provides the output data to the system processor via serial interfacing. The industry-standard that needs to be followed is based on the National Marine Electronics Association. Another example of modular GPS technology is the Antenova M10478-A3. It includes wideband antenna to provide support for dropouts while providing simple UART-based interfaces with data rates up to 115.2 Kbits/sec for easy interfacing to virtually any standard microcontroller [65].

Even though GPS can be an essential factor in autonomous vehicles, it has performance drawbacks that need to be considered [64]. This includes the level of precision, as this project required a very small margin of error, and precision that is down to a fraction of a meter. External factors such as weather conditions, electronic noise sources, and mapping

inconsistencies can all interrupt the functionality of a GPS signal. In order to be implemented in autonomous vehicles it must act as a supplement to an inertial measurement unit (IMU). The IMU consists of a platform fixed to the vehicle with three gyroscopes and three accelerometers that individually connects to the orthogonal X, Y, and Z-axis [64]. The IMU is only capable of calculating the motion of a vehicle while GPS determines the initial location of a vehicle.

2.13: High Sensitivity Thermal Sensor Technologies

A thermal sensor is able to scan the temperature of an object by measuring the amount of atomic activity inside the object [93]. In case that there is no atomic activity in the object, the temperature is considered to be the coldest state of matter. As soon as there is atomic activity in an object and particles are moving, the thermal sensor is able to pick up this movement and translate this movement into a temperature [93]. The sensor would be continuously sending information to the microcontroller and would keep track of the temperature of the object that the sensor is scanning (in case no object is being scanned, the air temperature would be the temperature provided to the microcontroller). Therefore, in case of a change in the reading of the temperature, the sensor would pick up the temperature change immediately if the object is in the distance range of the sensor. While the object temperature is being scanned and if the temperature picked up by the sensor corresponds to the temperature of the human body, the microcontroller would receive this temperature reading from the thermal sensor and then alert the driver of any potential people in the blind spot.

2.14: Infrared (IR) Image Sensor

Infrared sensors have the capability of measuring the temperature of an object without being in physical contact with the object's surface. The IR sensor can be implemented in this project in order to scan blind spots and scan objects that are in the blind spot area range. There are three major types of IR sensors: high, medium and low resolution sensors. These sensors work as lasers that scan the heat of objects in a certain range and a certain angle (the range and angle scanning depends from every sensor). Infrared Sensors have an analog to digital converter which converts the heat scanned into a temperature. Based on the heat dissipated from the sensor, the device will convert the heat into a temperature reading. The temperature accuracy is based on the resolution. By conducting a market research, the team found sensors with an accuracy varying from 0.1 degrees Celsius, up to an accuracy of 4 degrees Celsius.

After taking the infrared sensing and the thermal sensing technologies into consideration, research on some possible sensors to be used was conducted and is shown in Table 15.

Table 15: IR Sensor Comparison Table. The MAX 30205 is more accurate (higher resolution) and power efficient in comparison to the other two.

	Cost	**Technology used**	**Accuracy**	**Operating temperatures**	**Voltage / Current Supply**	**Resolution (if applicable)**
D6T 44L06	$33-$52	Thermal Sensing	+/- 1.5 degrees Celsius	0 - 50 degrees Celsius	4.5-5.5 Volts 5mA	
MAX 30205	NA	Infrared image sensing	+/- 0.1 degrees Celsius	0 - 50 degrees Celsius	2.7-3.3V 600uA	16-Bit (0.0039 degrees Celsius)
ML8540 IR Sensor	NA	Infrared image sensing	NA	-38 – 85 degrees Celsius	5V 5mA	0.5 degrees Celsius

In Table 15, three sensors were analyzed and compared to each other by taking in consideration some of the most important factors and criteria that each of the sensors has to meet. It is notable from the table that option 2 (MAX 30205) is more accurate and power efficient than the other two sensors. It is notable that the prices for the MAX30205 and the ML8540 are missing. This is because the MAX30205 is available only for medical use, and the third sensor does not have a price listed on its official website as this sensor is still under development, therefore it is unknown as to how much of the limited budget this sensor would take. Whereas the D6T 44L06 sensor has a reasonable price range for this project. The Omron D6T sensor is a possible option to be considered due to its price. The specifications obtained from the data sheet meet the requirements for human sensing, as the average body human

temperature is in the range of 36.5 – 37 degrees Celsius. In Figure 20 is a picture of the D6T thermal sensor.

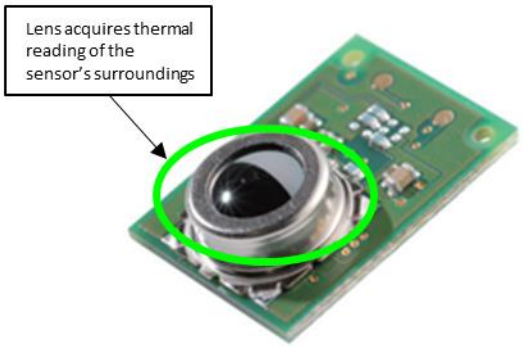

Figure 20: Omran D6T Thermal Sensor. As highlighted, the sensor's lens is used to acquire thermal readings of the surrounding environment. This can be used to distinguish between objects based on their temperature. [94]

Shown in Table 15 are the specifications in which this sensor operates and picks up heat from any objects. This would help in determining the architectural implementation of the sensors and their positioning on the bus. Therefore, the Omrah D6T was a possible sensor to be implemented after being analyzed in detail by all the members of the group.

From Table 15, where three possible options are listed, the team was limited by choosing the D6T sensor, since the MAX 30205 was available only for medical purposes and the ML8540 IR Sensor does not have a price listed since it is still under development according to MELEXIS. Therefore, it is difficult, planning on purchasing this sensor due to the limited budget. Therefore, the most suitable sensor to take in consideration would be the D6T. While reviewing the specs of this sensor, a problem was noticed that might occur while using this sensor during the winter time, since the sensor will not provide as accurate measurements below the freezing point of 0 degrees Celsius (32 degrees Fahrenheit), a solution had to be found that would solve this issue. In order to find a feasible sensor that would be capable of operating in temperatures below the freezing point, further research was conducted. The possible options that were found during the research phase are the following sensors shown in Table 16:

Table 16: MELEXIS Sensor Comparisons. The MLX90640 datasheet suggests that it is suitable for pedestrian detection. [95]

	Cost	Technology used	Accuracy	Operating temperatures	Voltage / Current Supply	Resolution (if applicable)
MLX90614 (Option 1)	$20	Infrared sensing	+/- 0.5 degrees Celsius	-40 - 125 degrees Celsius	4.5-5.5 Volts 2mA	0.14 degrees Celsius
MLX90621	$43	Infrared sensing	+/- 1 degrees Celsius	-40 – 85 degrees Celsius	2.6-3.3V 9mA	0.5 degrees Celsius
MLX90614 (Option 2)	$14	Infrared Sensing	+/- 0.5 degrees Celsius	-40 – 85 degrees Celsius	2.6-3.6V 2mA	16-Bit
MLX90640	$35	Infrared Sensing	+/- 0.5 degrees Celsius	-40 – 85 degrees Celsius	3.6-5V 14mA	16-Bit

The sensors shown in Table 16 use infrared technology for human sensing. The IR sensors above are manufactured by MELEXIS, from where the specifications for each of the sensors were obtained individually. Going through the data sheets for each of the sensors provided a better understanding of the specific capabilities of each of the sensors [95]. The MLX90640 is suitable for pedestrian detection and classification, but it is not specified as being suitable for blind spot detection. The MLX90621 has a higher price than the other sensors but has been used for presence detection and automatic blind spot detection [95]. The MLX90614 sensor also has a greater price, but has shown success with various presence detection applications [95]. This provides a number of possible infrared sensors that could be used for pedestrian detection within the application of this project. Specifically, infrared can help with

determining when a person has entered the “blind spot” and the bus would need to take corrective action.

2.15 Chapter Summary

This chapter discussed the research performed by the team to gain insight into both the human factors that play into the cause of bus accidents, as well as potential technologies to implement in creating a pedestrian detection system. Multiple computing solutions and sensor types were investigated. Although no exact components were chosen at this stage, the information found helped ultimately drive the final design considerations. Later sections including Chapters 3, 4, and 5 stem from the information that was garnered during this chapter.

3: Design Considerations

In planning the final idea for the system design there were important factors that needed to be considered in order to have the best possible system implementation. These factors were broken into two main categories for consideration; defining of the bus environment, and situation identification.

3.1: Bus Environment Definition & Situation Identification

In order to effectively create a system that could help prevent possible accidents through pedestrian detection, it was necessary to define the exact dimensions of the bus that would be used and the corresponding blind spots that result. Understanding the overall structure of the bus and what blind spot is created allowed for an optimal "blind spot" area that could be determined. The system would need to effectively cover this "blind spot" in order to have the most effective accident prevention. The Figure 21 corresponds to the dimensions of both the bus, and the crucial blind spot. The system would have different degrees of action based on the proximity of the pedestrian to the bus.

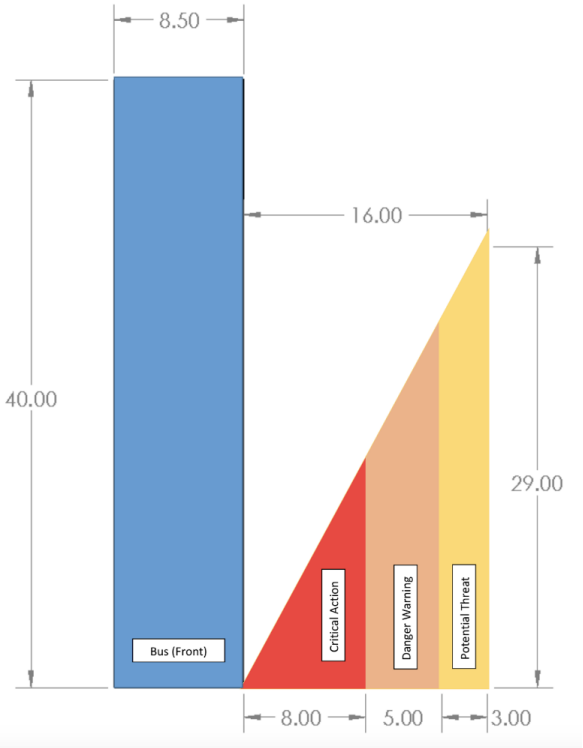

Figure 21: Bus Blind Spot with Indicated Zones of Detection (Units in Feet). Blind spot reaches up to 16 feet from the driver's side window. The three zones represent increasing level of danger to the pedestrian, as the distance between the pedestrian and bus narrows.

The blind spot was broken into three distinct zones including: Potential Threat, Danger Warning, and Critical Action. Each of the zones had the system react in a different manner. The "Potential Threat" zone is when a pedestrian will first be noticed by the system and be monitored accordingly. The system is aware of an object at this point, but does not take action. The next area of detection is the "Danger Warning" zone. This zone results in the proactive warning aspect of the system being applied so that the driver has the ability to take effective action without the automated response taking over. In a final design, if the bus driver is unable to take effective action in time the system will automatically take over. This will occur once a pedestrian has entered the "Critical Action" zone where if immediate course correction isn't taken then an accident would be inevitable.

A major factor to consider that ultimately drives the decision making is if the object detected is a pedestrian. The point of this project was to prevent accidents involving pedestrians, as they tend to be fatal. The bus will take more drastic measures if a pedestrian is to enter a crosswalk in comparison to if the system detects an inanimate object, such as a road sign. Other factors that were taken into consideration include the velocity and direction of the pedestrian. For example, if the pedestrian is walking away from the bus the corrective action could be as simple as slowing down the bus so that the pedestrian has time to pass into a safe distance. When a pedestrian begins, moving towards the bus the situation becomes increasingly complicated as the danger and possibility for an accident quickly escalates. If the pedestrian is traveling on a bicycle their velocity will thusly be higher. Because of this, the time to take corrective action is reduced. This can affect the decision-making as it could be within the best interest to forgo the proactive warning entirely and immediately allow the automated response to take over. The average walking speed of a human is 3.1 miles per hour (mph) while the average bike pedaling speed can be as high as 9.8 mph, over three times as fast [96] [97]. This cuts the time for the bus to react into a third. Similarly, it was important to analyze the general travel patterns of a city bus, especially in turning, to gain an understanding of how the blind angles and turn speed may vary. These factors were taken into consideration when finalizing sensor placements and the system's decision-making.

3.2: Sensing Technologies

Each of the sensors served a necessary purpose in accomplishing one of three major factors for the system. The two main factors were if there is an object in the "blind spot" and if that object is a person. The final factor was the ambient temperature under which the system is operating. This was required as each sensor has a different operating range, and certain temperatures can cause inaccuracies in particular sensors. Thusly, a temperature sensor would always be required for this system to determine which sensors should be utilized. Table 17 shows which sensors provide a possible solution for each main factor.

Table 17: Sensor Applications. At least one sensor for object detection and pedestrian identification to be implemented in the system.

Sensor	Object Detection	Pedestrian Identification
Radar	**X**	
Lidar	**X**	
Ultrasound	**X**	
Camera		**X**
Infrared Thermal Sensor		**X**

The benefit of utilizing an ambient temperature sensor is that it protects sensors from being damaged. The sensor would determine if the outside temperature is viable for the operating range of a sensor. In order to have an accurate system, the microcontroller would utilize only the sensors that are capable of operating in the current weather conditions where the bus is located. The temperature sensor could be placed anywhere on the bus as long as the temperature sensor is measuring the outside temperature. Once the temperature sensor sends this information to the Zedboard, the Zedboard itself would utilize only the sensors that are capable of operate in those weather conditions. Table 18 shows a comparison chart between two temperature sensors that were taken in consideration to be implemented.

Table 18: DHT11 and DHT22 Temperature Sensors Comparison Chart. The DHT22 provides a greater operating temperature range, as well as greater accuracy. The operating temperature range is critical, as a bus's exterior could be subject to sub-zero degrees Celsius.

	Price	**Operating temperatures**	**Accuracy**	**Power consumption**	**Refresh Rate**
DHT 11	$5.00	0 - 50 degrees Celsius	+/- 2 degrees Celsius	3-5 Volts 2.5 mA	1Hz
DHT 22	$9.95	-40 - 120 degrees Celsius	+/- 0.5 degrees Celsius	3-5 Volts 2.5mA	0.5Hz

3.3: Radar

As mentioned in the earlier section, radar uses radio frequency waves to detect motion. Radar is capable of motion detection, which includes the distance to as well as the speed and direction of the target. This means that radar would be able to determine where a certain object is in relation to the bus and whether the object is approaching to or moving away from the bus. The main advantage of radar over other sensors is that it uses electromagnetic waves, which do not require a medium to propagate. Due to this form of operation, radar is capable of penetrating through any type of weather conditions. Another advantage is that it works well in either nighttime or daytime. Additionally, the waves propagate at the speed of light, which makes the radar long range and allows for the Doppler Effect to be used to estimate the target's distance, velocity, and angular position. This property enables radar to distinguish fixed targets from moving targets. Lastly, radar can be used in various configurations and it is a very reliable sensor with a fast refresh rate when compared to other sensors. Also, solid-state radar technology is much cheaper than the solid-state LIDAR technology. One of the biggest disadvantages of radar is that it cannot distinguish and resolve multiple targets that are very close to each other. Another very important limitation of a RADAR is that it can sometime take couple of seconds to "lock on" which makes it unsuitable for real-time operating systems. In addition, radar does not work well with short range targets.

3.4: LIDAR

LIDAR works on a similar principle as Radar, except that it uses light in a form of a pulsed laser instead of a radio wave. LIDAR is used to provide a 2D or a 3D mapping of the sensor's surroundings by measuring the distances to the surrounding objects. LIDAR has a long range of detection and works in both dark and bright conditions. Due to the laser technology, LIDAR can provide better resolution than radar. LIDAR is also more available on the market than the radar technology. LIDAR is not as reliable in fog, rain or snow as radar. LIDAR usually has a much smaller operating temperature range than radar sensors. LIDAR modules are usually more expensive and are not as robust as the radar systems. Since it provides better resolution than radar, the raw data from the sensor is bulkier and takes longer to process. Because of this, LIDAR sensors usually have a slower refresh rate than radar sensors and require a higher processing power.

3.5: Ultrasound

As introduced earlier, ultrasonic sensors can be implemented for proximity and range detection of objects within the sensor's line of sight. Most ultrasound-based sensors include an ultrasonic transceiver that enables the transmission of high frequency sound waves, and can then receive the sound waves that are reflected back by an object. The time between transmission and reception of the sound waves allows the sensor to determine a distance reading. The primary advantage of ultrasonic sensors is their ability to give an accurate distance reading, often with a precise resolution (divisions of distance that the sensor can discern between). An ultrasonic sensor is best used fixed to a surface, where it can then determine range of objects relative to its position. The ultrasonic "beam" that is emitted by the sensor is highly directive, which is beneficial for applications where a straight, perpendicular distance reading is needed. Depending on the application, the directivity of ultrasonic sensors can also be a drawback. The linear, narrow angle at which the sensor is able to detect objects would not be ideal for detecting object presence within a wide space. This factor makes ultrasound less appealing for initial detection of objects within proximity. In the case of detecting pedestrians to the side of a city bus, ultrasonic sensors would be most applicable in a scenario where the sensor is focused on a specific area or angle from the bus's side, secondary to a different wide-angle technology to detect humans in the

general vicinity. Ultrasound would be able to identify the distance of an object from the bus, but the sensor would have to be directed to point at the specific object linearly.

3.6: Camera

Cameras are able to give a clear and accurate picture of the area surrounding a bus. The important part of using a camera is to not distract the bus driver; therefore, the system must be able to process images on its own. The camera is able to detect if the object would be a human or not. This is done by utilizing software such as OpenCV to detect and track pedestrians. The camera is able to have a high accuracy of identifying if an object is a human or not, and is then able to track the object. The combination of AdaBoost and Linear SVM are able to confirm the presence of a pedestrian faster than OpenCV but at a reduced accuracy. This system is looking to improve in the future. The camera has a slow computation rate, making it hard to implement within a fast decision operation. Most cameras often have operating temperatures that are not ideal for year round applications and the cameras that can survive in warmer or colder weather are higher in price. Cameras also have a problem with bright light, as the picture can be washed out and make collecting data extremely difficult. There are other variables such as clothing, human posture, and the background that may cause detection issues. There are too many niche variables that make the camera hard to utilize with a bus application within the scope of this project.

3.7: Infrared Thermal Sensor

The Infrared Thermal Sensor is a crucial sensor for this project, as it is able to detect the temperature that an object is emitting; therefore, it can distinguish between an object and a body. Depending on the accuracy and the resolution of the sensor, it can distinguish whether the body being scanned is a human or another body, such as an animal. If the object temperature corresponds to the temperature of the human body, then the sensor “confirms” that the scanned object was a human body. The advantage of using the Infrared Thermal Sensor is that is fairly inexpensive. Different IR sensors range between $20 up to $50. Budget is not the biggest concern of this project, but accuracy is. The IR sensors chosen to be used can operate in temperatures from -35 degrees Celsius, up to temperatures of 85 degrees Celsius. The accuracy of the IR sensor varies on the temperature that the sensor is operating at and also depending on the temperature of the object that is being scanned. The IR sensor is accurate up to +/- 0.5

degrees Celsius when the sensor is operating in temperatures between 0 - 50 degrees Celsius. One of the cons that the IR sensors carry is the fact that the distance range is not very large and does not cover the whole blind spot itself. Therefore, a certain number of IR sensor has to be purchased in order to have as much human detection coverage as possible.

3.8: FPGA Sensor Communications

The FPGA is capable of handling various forms of peripheral communications with the different sensors via Universal Asynchronous Receiver/Transmitter communications (UART), Serial Peripheral Interface (SPI), or Inter-Integrated Circuit (I2C). Peripheral Module connectors (PMOD) on the Zedboard support UART, SPI, and I2C via PMOD functionality and communication [51]. Xilinx offers peripheral drivers that support communications via PMOD directly to the ARM processors located on the Zedboard. The ARM processors will be first used to process the incoming data from the various sensors, before this processed data is sent to the FPGA. The ARM processor is capable of communicating with the FPGA via Advanced eXtensible Interface (AXI). This interface will need to be configured to meet the requirements of this design. Once the FPGA has the correctly processed data, the necessary programmable logic implementation will determine what decision needs to be made, and will select the appropriate output from the corresponding Look-up Tables (LUTs) based on this information. The FPGA can then output the data to the necessary system application that would provide the appropriate response. This includes connecting to the proactive warning system in order trigger alerts, and eventually connecting directly to the braking and turning system that will activate if the point of critical action is reached.

3.9 Chapter Summary

This chapter focused on discussing the advantages and disadvantages of the various components considered for the final design. This took into consideration the necessary information that would ultimately be used for making a final decision as to what sensors and technology would be used in meeting the needs of the project. This chapter also investigated factors that exist in the environment surrounding the bus and how they may affect the system's decision making process

4: Proactive Driver Alert

The tiered approach to the systems design was created so that as much control as possible would remain in the hands of the driver. A proactive system makes it so that before automated action is taken, the driver will be notified of possible situations that could lead to pedestrian accidents in order to make necessary course corrections. Possible alert methods were researched to determine an efficient and effective means of proactive driver alert.

4.1: Liquid Crystal Display

A Liquid Crystal Display (LCD) is an effective means of displaying detailed information including words and brief phrases. An LCD would be used in for the pedestrian detection and avoidance system to display a warning message to the driver indicating the issue. For example, when an object enters into the blind spot, it could start by displaying a message such as "WARNING" and could progressively blink at a faster rate to indicate how close the approaching pedestrian is in relation to the bus. The message could even be more detailed as to indicate the exact zone in which a pedestrian is detected, *i.e.* DETECTION, WARNING, CRITICAL. Driving of an LCD can be handled through the use of the FPGAs programmable logic functionality. Very often, the boards designed around an FPGA have a display interface directly implemented as a peripheral device on the board [98]. The design logic would include determining what to display, and when to display it based on the incoming data.

There are some possible issues that could arise from the use of an LCD. The biggest issue is the distraction that it would cause for the driver. In order to effectively read the LCD, the driver must take their eyes off of the road to check the warning message, read what it says, and then process the information to determine what course of corrective action should be taken. That process can easily take up more than the available time. This is an inherent drawback in the implementation of an LCD. Therefore, if an LCD were to be implemented, it would need to be done in a way where the driver could notice a change in information through peripheral vision, rather than direct attention.

4.2: Steering Wheel Vibration

Steering wheel vibrations could be an effective solution for alerting a driver. This offers a means of alert that does not disturb passengers, and would not require the driver to take their

eyes off the road. Similarly, if a driver is distracted, the vibration could serve as a means for them to refocus their attention while an LED or LCD could potentially go unseen by a driver if they are distracted. The amount of vibration could correspond to the distance of the detected pedestrian to the bus, and the level of danger that they pose to causing an accident. As a pedestrian becomes closer to the bus, and is traveling towards it at a faster rate, the steering wheel vibration could increase indicating the need for a higher level of caution or corrective action.

This would require the design of a steering wheel that has some form of vibrating motor controller built-in or attached to it, and potentially multiple vibration capsules so that each part of the steering wheel would vibrate during an alert. The vibration, however, does not provide any form of location information for where a pedestrian could be present in the blind spot. The utilization of a vibrating steering wheel would require some other form of driver notification on top of it. A vibrating steering wheel could be cascaded with a display to identify when critical action is necessary. The vibrations would allow the driver to keep his eyes on the road, but also know that there is an imminent threat and corrective action is required.

4.3: Dashboard LEDs

LEDs offer a simple, cost effective alerting option that can be easily implemented. LED functionality is built directly into the Avnet Zedboard that can be used, and for the purposes of testing will require no additional purchase. The Zedboard has eight LEDs which can be utilized [51]. This will eliminate the need for any additional data lines during the testing phases of the application. The actual use for driver alert could incorporate three different colored LEDs that would indicate the three different zones of detection. Green could indicate that something has been detected. Yellow could indicate that a pedestrian has entered into the next zone where corrective action might be necessary and the pedestrian poses a possible threat. Red would indicate that the pedestrian has entered a critical region and immediate action needs to be taken in order to avoid an accident. The light that is turned on would be determined by the programmable logic of the FPGA fabric. After the incoming data from the various sensors is broken down in the ARM processors and transferred to the FPGA fabric, the system would determine the level of alert will be needed.

The LEDs could be positioned on the dashboard of the bus, so that they would remain in the driver's vision without having to take their eyes off of the road. There could be two sets of the three LEDs. Each set of LEDs could correspond to either the right side or left side of the bus accordingly to provide as much information while remaining simplistic in the design. The use of different colors would also provide a quick insight into the current situation detected by the system. It is easier to process the simple color of an LED than having to read a word or phrase, and will be seen only by the driver and not affect the passengers in any way. The color scheme would also mimic that of a traffic signal, which is familiar to a driver. Other LEDs could also be incorporated on the mirrors of the bus, and even on the steering wheel to provide a very thorough alert to the driver. Taking these factors into account, one may see how the simple nature of an LED can provide an effective means of driver alert without requiring the driver to divert a large portion of his attention.

4.4: Sound Alarm

The use of sound for an alerting system could provide a distinct means for notifying a driver that a pedestrian has been detected by the sensors. A sound alarm could be easily implemented through the use of a piezoelectric buzzer which can be found on Digikey for as low as $1 [99]. The buzzer would be driven by the programmable logic of the FPGA fabric, which would indicate when it would need to make noise. It could be possible for the buzzer to use different tones to indicate the proximity of the pedestrian detected. Additionally, the sound from the buzzer could be heard by the passengers, and could be found to be a nuisance to them. The driver might also find the sound to be bothersome especially if it is constantly going off, as the sound needs to be distinct enough to get the attention of the driver. Therefore, sound would likely be implemented to trigger in only a critical situation. Utilizing a buzzer in only a critical situation would make it so that the driver knows that action is required immediately to avoid a possible accident. The addition of a buzzer on top of a visual would provide a means of cascading effects to have a greater degree of notification.

4.5: Driver Alert System Peripheral Combinations

The proactive alert would be best implemented by combining the methods discussed above. The methods have two distinct characteristics that would be the reason for selection. These are catching the attention of the driver, and effectively relaying the information to the

driver. The use of a sound alarm or steering wheel vibration represent the methods for catching the driver's attention. The LED and LCD methods represent the effective relaying of information. Having an overall alert system that incorporates one attention grabbing method, and a means of relaying the information will provide the most effective alert.

The use of steering wheel vibrations or a sound alarm can be used to effectively grab the attention of the driver. These are both alerts that would become bothersome to the driver if it constantly were going off for every slight detection. Also, it is more difficult to provide a clear description of the current situation that is detected by the system with these methods. This is why there is a need for the more information-based aspect on top of this. The attention grabbing would be utilized only when the situation has a higher level of threat. For an example, either a vibration or sound would be sent out when a pedestrian is passing through the danger warning zone and beginning to approach the critical action zone.

Overall, the sound alarm would be a better option for the implementation of the proactive driver alert. This is due to the simple and cost effective solution that it supplies. Steering wheel vibration can possibly be missed due to the innate shaking that is caused by the natural driving of a large bus. A more detailed breakdown of factors comparing the two methods can be seen in Table 19, consisting of the comparison of the specifications for a piezo electric buzzer and a vibration motor capsule.

Table 19: Comparison of Piezo Electric Buzzer & Vibration Motor Capsule. The lower cost and power requirement of the buzzer makes it a more efficient solution. Audible warning can be provided to the driver in the event of a potential collision. [99] [100]

Feature	Buzzer	Vibration Capsule
Cost in USD	1.38	4.99
Operating Temperature (Degrees Celsius)	-20 to 70	-30 to 70
Operating Voltage	Up to 30V peak to peak	2.2V to 3.6V
Current Consumption	Less Than 30 mA	250 mA Max

This breakdown shows that the buzzer would require less power overall due to the low current consumption, and provides flexibility in design by allowing for large operating range for the voltage. Also the cost for the vibration motor capsule does not include the cost associated with designing a steering wheel based around the vibration feature. The buzzer would be capable of working in all applications and will provide a simple means for effectively alerting the driver that would be more effective than the vibrating steering wheel.

The alert system will have an information-based display that will consist of either an LCD or a series of colored LEDs. For the purposes of testing the LED, a test application can be accomplished through the eight LEDs that are built directly into the Zedboard, while the LCD would require an external connection. The LEDs can be a good solution, but the driver may miss the blinking LED due to its small size on. The intensity of light that is emitted off an LED may also not be as apparent as compared to an LCD screen, possibly missed by the driver in the daylight. An LCD screen would act as a great attention grabber for the driver due its size. A single color image could be displayed on the entire LCD screen, making it easier for the driver to recognize what zone the pedestrian is in. The Zedboard comes with a VGA port by default, so installing an LCD monitor within the system should be feasible through the VGA interface.

4.6: Chapter Summary

This chapter focused on demonstrating methods that could be used for alerting the driver in a critical scenario. The details for each possible method were discussed as to show the considerations that were taken into account when determining an optimal solution for the proactive driver alert. The main concern was finding a solution that was not distracting nor bothersome to the driver, nor require them to take their eyes off the road.

5: Proposed Sensor Solutions

The team's research into the sensor options provided insight in creating different proposed solutions. Based on the project considerations, the best solution was determined based on the following four system implementations.

In order to identify a pedestrian that is located in the blind spot area of the bus, four different sensor implementations were designed. In these four different implementation strategies, different sensors capable of detecting movement and objects were considered, and other sensors were implemented that serve for human detection. The priority when implementing the sensors was the safety of the pedestrians. For the human detection process, it is crucial for the sensor implementation to detect whether the object in the blind spot is a person or not in order to prevent an accident, as well as to prevent false alarms in the case of an object being present instead of a human. In order to prevent a failure of the system, one of the components used in every implementation has to cover all (or most) of the blind spot area, and then other sensors will be used in order to determine whether the object is a human or not. It is important that every implementation designed can work in extreme weather conditions, such as temperatures below the freezing point and snow. After taking factors such as safety, accuracy and functionality of the system into consideration for all the above mentioned conditions, the four implementations were sketched to scale using Solidworks and AutoCAD. After each of the arrangements were sketched, advantages and disadvantages for each possible solution were taken into consideration to evaluate the design to be used for this project. In order to obtain a better insight of the four different implementations, each of them will be shown and described in detail.

5.1: Solution One: LIDAR Positioning, Heat detection, Face Detection

The first system implementation consists of three components: LIDAR, IR Sensors, and Camera. The combination of these three sensors has the capability of detecting an object, and determining whether or not the object is a human. Figure 22 shows a proposed arrangement of these sensors on a 40-foot city bus.

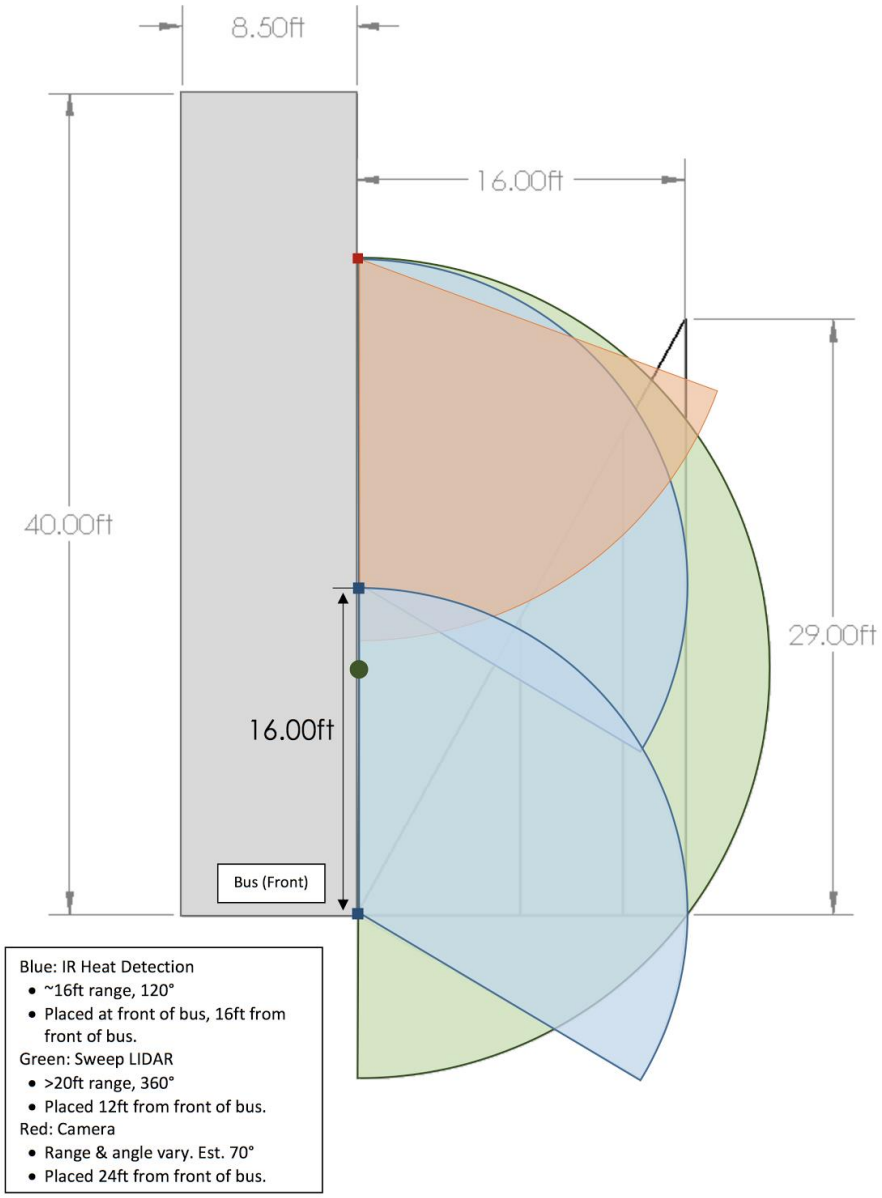

Figure 22: First Solution sensor Implementation. LIDAR, Camera and IR to be implemented in the system. Arranged for maximum coverage of the blind spot, with the camera positioned to view pedestrians near the front of the bus for detection of human faces.

In Figure 23, a flow chart of this proposed solution is shown, which describes the steps for how this implementation will detect the object and determine whether it is a human or not.

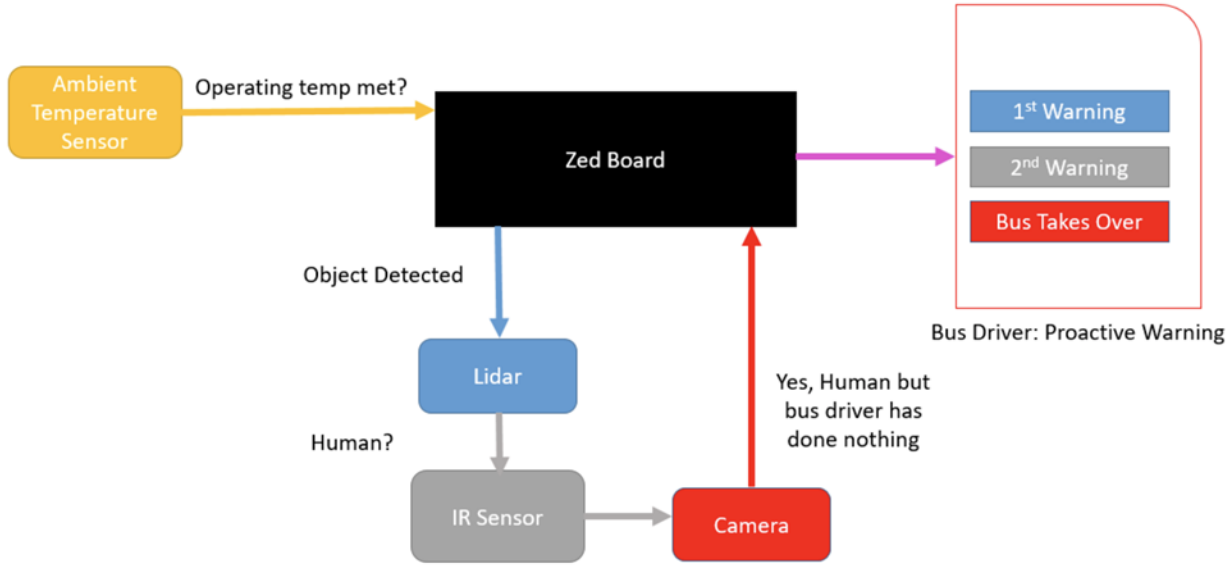

Figure 23: First Solution Data Flowchart. Data fusion on how the information will be processed in order to trigger an alarm and warn the driver. The detection of objects is handled by the LIDAR sensor (blue). If an object is detected in the blind spot, the IR sensor (gray) and camera (red) are then used to detect human body temperature and a human face to verify the object is a pedestrian.

Each of the four implementations has a temperature sensor, which measures the ambient temperature and can help the microcontroller determine which sensors and components this system should use under the current weather conditions. For the first proposed solution, as shown in Figure 23, the first component that will be used to scan the blind spot zone is the LIDAR, which has a field of view of 360 degrees and has a range up to approximately 140 ft. The LIDAR is the component in this implementation that is capable of covering the majority of the blind spot area, and it will be used to detect only whether there is a moving object or not. The LIDAR alone cannot distinguish between a human and another object. In order to determine whether the object detected by the LIDAR is human, IR Sensors will be used to scan the body temperature of the object. If the temperature read by the IR sensors corresponds to the temperature of the human body, a warning will be sent to the driver, alerting him that there is a person located in the blind

spot of the bus. Meanwhile, while the driver is being alerted of the presence of a pedestrian being in the blind spot, a camera is used for the human detection process in order to prevent false warnings. The camera is implemented with the purpose of being used for face detection. The reason why face detection is used after the IR sensor, even though both components are meant to sense human presence is the following: During many situations, people wear sunglasses, hats and during their winter time, some people cover their face as much as possible to prevent the cold. This can cause the face detection process to interpret incorrect information as to whether the object is a person or not. Therefore, the IR sensor is being used first, since every human body has heat that is being released. If the bus driver does not react even after the being alerted from the camera, which has sensed a human face, then the bus driver will not have any more control over the bus, and in further development processes, the bus has to be stopped automatically. This would be only if the bus driver does not react to any of the warnings provided by the Zedboard. This proposed solution has its advantages and disadvantages which will help in determining whether this solution is the most suitable for this application or not. A comparison chart that shows each of the advantages and disadvantages is seen in Table 20, shown below.

Table 20: Advantages and Disadvantages of the First Proposed Solution. Factors such as accuracy, data processing speed and durability taken into consideration.

ADVANTAGES	DISADVANTAGES
Image processing could be accurate	Potentially slow
Relatively inexpensive	Heavily affected by weather conditions

Since safety and accuracy are some of the major priorities while designing the implementation, and this solution may not be the most accurate due to the limitations of face detection, another proposed solution was designed with greater accuracy in mind.

5.2: Solution Two: Radar Positioning, Infrared Heat Detection, Ultrasonic Ranging

The second proposed solution for the implementation and component placement uses radar, infrared (IR), and ultrasound technologies for large coverage of the blind spot. The positions of the three components, Radar (Yellow), IR sensors (Blue), and Ultrasound (Pink), may be found in Figure 24.

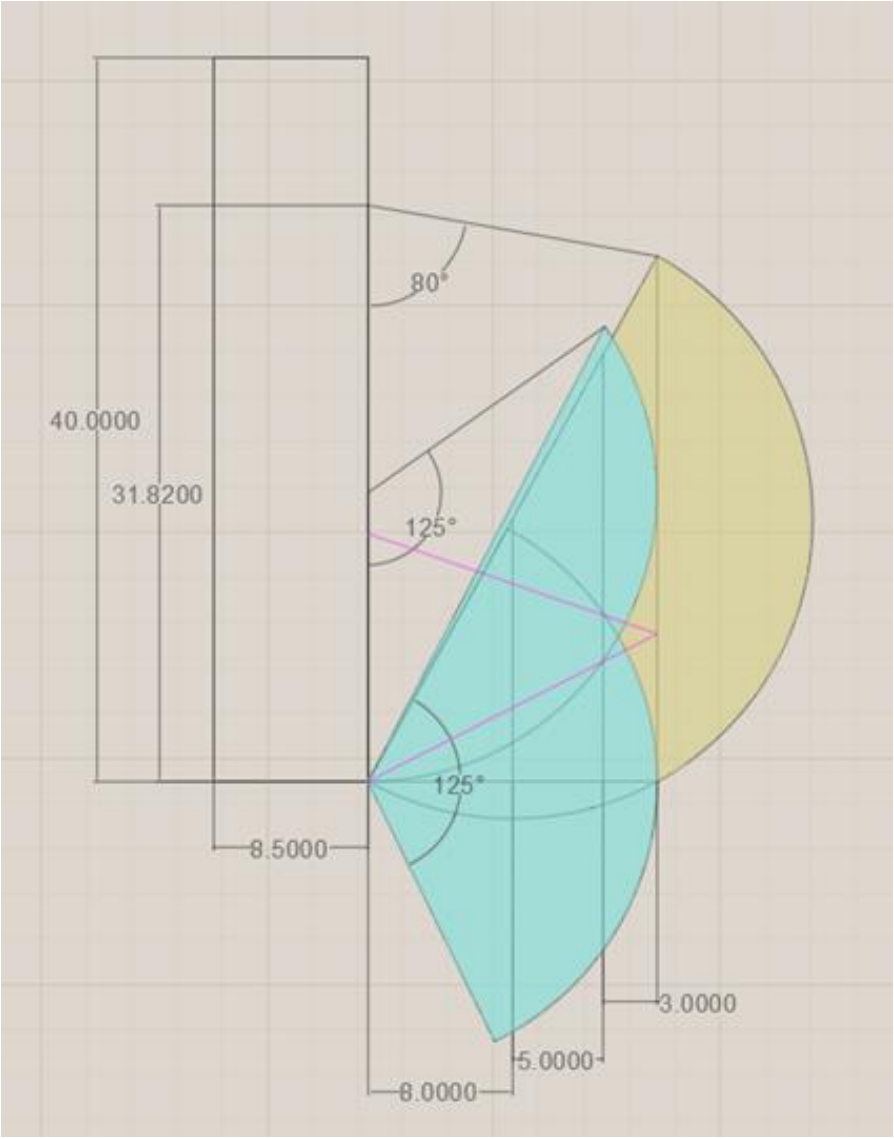

Figure 24: Second Solution Sensor Implementation. Radar (Yellow), IR (Blue) and ultrasound (Pink) to be implemented in the system. Radar and IR placed for best coverage of the blind spot through overlapping field of view. The ultrasonic sensors are positioned to cover the small area where the IR sensors and radar do not overlap.

The use of the radar, IR, and ultrasound sensors makes it possible for the system to go through two steps in order to detect whether or not there is a person in the blind spot area. The flowchart shown in Figure 25 represents the order of how the human sensing process will proceed.

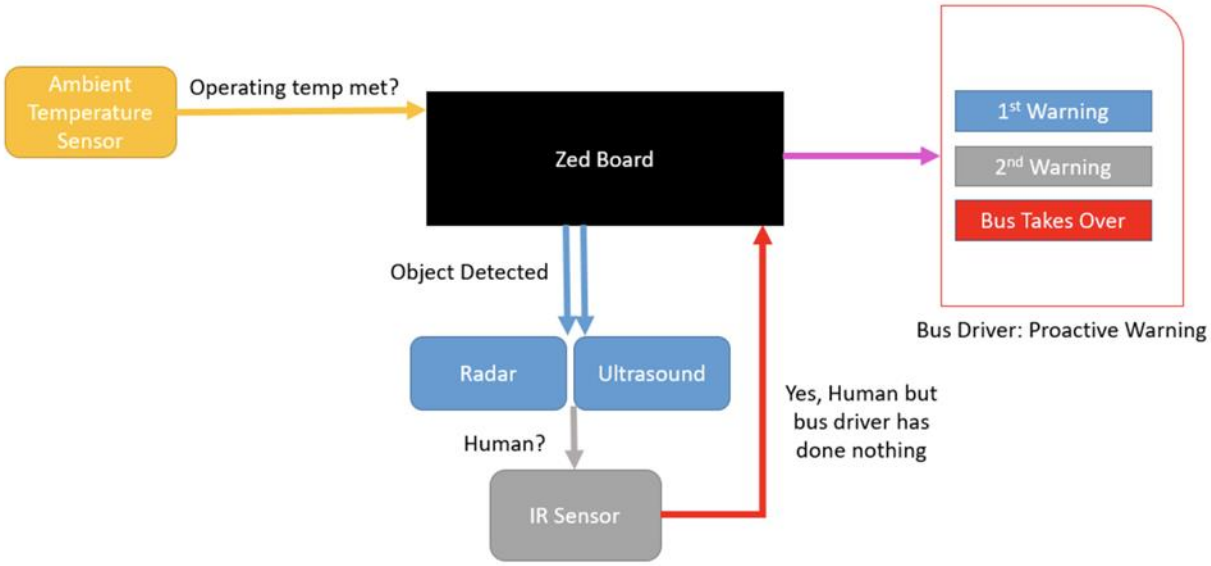

Figure 25: Second Proposed Solution Flowchart. Data fusion for how the information will be processed in order to trigger an alarm and warn the driver. The radar and ultrasonic sensors are used for detection of objects. If an object is detected, the IR sensors are used to sense if the object's temperature is roughly around human body temperature. This would indicate that the object is a pedestrian.

As also shown in the first proposed solution, the temperature sensor will help the microcontroller determine which sensors to use under the current weather conditions that the bus is driving in. The flowchart shows that two components are being used at the same time to detect whether there is an object located in the blind spot zone or not. The radar, as shown in Figure 24 above, is capable of covering the whole blind spot area because of its placement on the bus and its wide range and field of view. The radar is also capable of detecting if the object is moving, how fast it is moving, and in which direction it is moving. The ultrasonic sensor is placed at a fixed point and directed at an angle where the IR sensors are not capable of scanning. The ultrasonic sensor has a good detection range, which is capable of covering the length of the blind spot of the bus (but not with a wide FOV angle). These two components are able to alert the bus

driver that there is an object in the blind spot of the bus, as well as when the object is moving and either getting closer to the bus or further away. In order to detect whether to object sensed is a human or not, IR sensors are placed on the bus as shown in Figure 24. The IR sensor will scan the temperature of the object in question. If the temperature of the object corresponds to the temperature of the human body, the driver will be alerted that there is a human in the blind spot. This implementation tends to be accurate and functional in extreme weather conditions. Table 21 highlights the advantages and disadvantages of this proposed solution.

Table 21: Advantages and Disadvantages of the Second Proposed Solution. Factors such as Accuracy, data processing speed, and durability taken into consideration.

ADVANTAGES	DISADVANTAGES
Accurate, minimal system failure possibility	Might result in a false alarm
Durable in extreme weather conditions	Slow processing speed
Fast decision making	

This proposed solution will be able to detect objects in further distances of the blind spot zone, but will determine whether the object is a human or not only when it gets in the range of the IR sensor. The IR sensors cover a majority of the blind spot area, but for the area that it is not covering, the radar and ultrasound might trigger a false alarm in case that the object is not a human. The biggest advantage of this implementation is that the risk of an accident is likely very low, since the blind spot is completely covered and some of the areas are being overlapped, which will result in better coverage than the first proposed solution.

5.3: Solution Three: LIDAR and Radar Positioning, Infrared Heat Detection

The third proposed implementation incorporates three sensor technologies: infrared (IR) heat detection, LIDAR, and radar. This solution is aimed to have a wide area of coverage, with each sensor incorporating either a wide-angle field of view, long effective detection range, or a combination of the two. Taking these factors into consideration, a preliminary sensor arrangement was created using a 40ft bus model, with each sensor in specific areas to maximize coverage of the critical blind spot. A diagram of the placement of the sensors can be found in Figure 26.

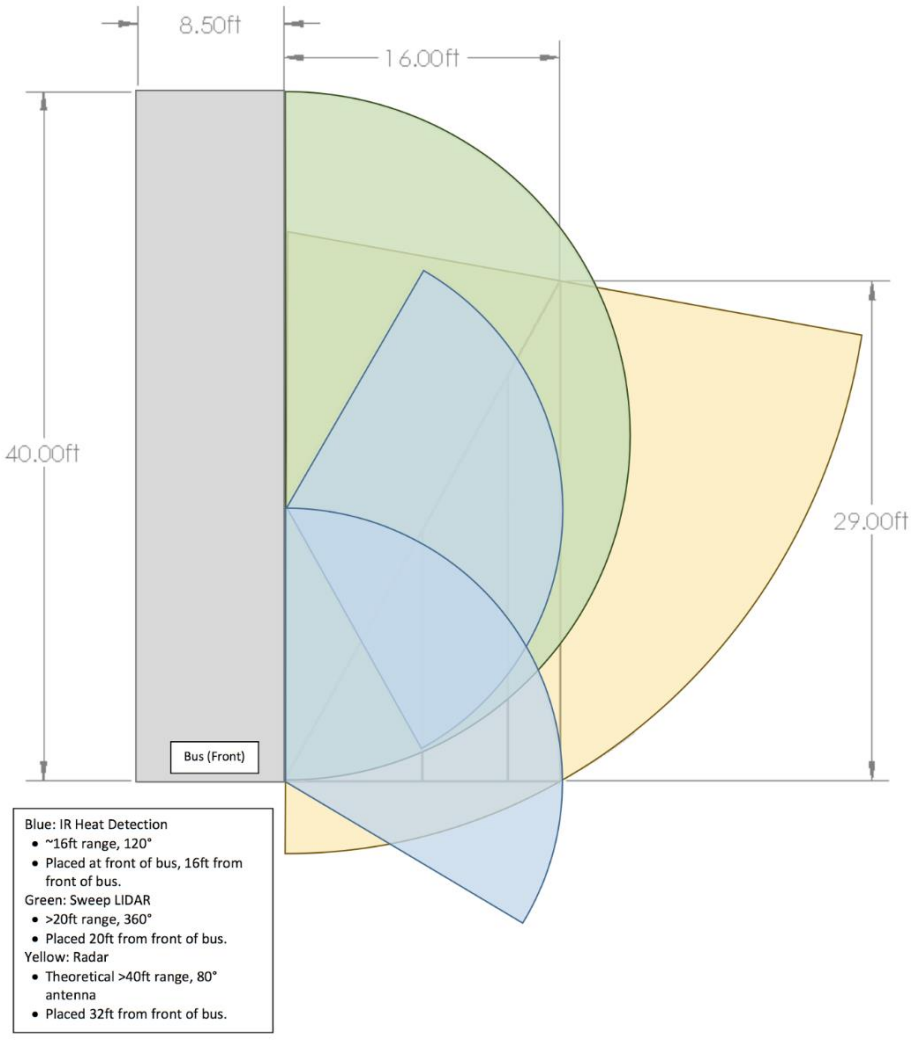

Figure 26: Third Solution Sensor Implementation. IR, LIDAR and Radar to be implemented into the system. This solution aimed to have a very wide area of coverage in the blind spot. All three of the sensors have a large theoretical field of view.

As shown in the placement diagram, the radar implementation that was researched earlier made use of an 80˚ directional antenna and had a theoretical range well over the size of the blind spot. Due to these specifications, the radar sensor in this design will be placed near the back of the bus, so that the entire blind spot area will be within an 80˚ angle of the antenna. With the range that radar is capable of, it will likely be the first source of position information for objects that enter the blind spot. Also incorporating a relatively long range, and the widest angle of detection for all sensors, the LIDAR system will also serve as a sensor for object position detection. The LIDAR placement in this configuration was chosen at the midpoint of the bus, so that the 180˚ sensing field would cover a long radius from the bus's center. The range of LIDAR in this diagram is estimated to be 20ft, yet it is theoretically capable of detecting objects at a longer distance. Because of the rotational operation of the researched LIDAR sensors, such as the Sweep, a small platform will likely need to be fixed to the bus so that the LIDAR can be mounted level, roughly parallel with the ground. The third sensor technology, IR heat detection, does not support the same long-range detection that radar or LIDAR would have, however, it maintains a wide 120˚ field of view. The advantage of IR in this implementation is that it can be used to detect whether or not an object is human by analyzing its temperature, after the radar and/or LIDAR have determined the object's position. The two IR sensors are positioned both at the front and midway down the bus to completely cover the Danger Warning and Critical Action zones. This enables the system to detect humans within the two highest risk zones, where either the driver or the autonomous system would have to take action in stopping the bus. These sensor placements are designed to aid the flow of information in the central computing system's decision making process, as shown in Figure 27.

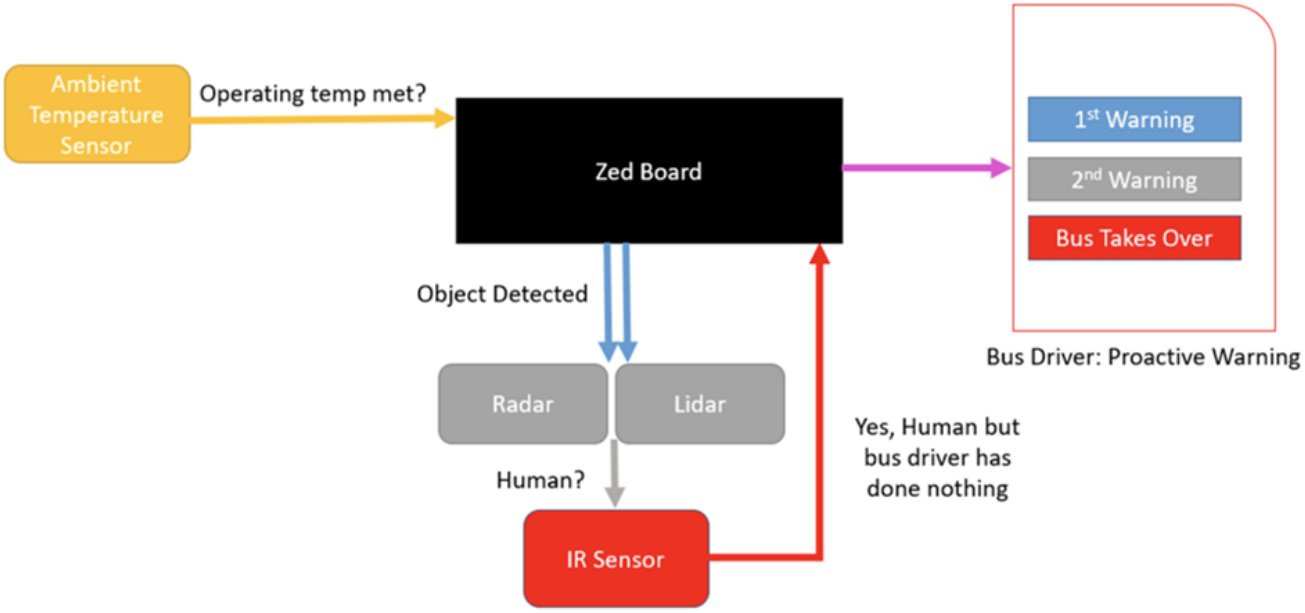

Figure 27: Third Solution Data Flowchart. Data fusion for how the information will be processed in order to trigger an alarm and warn the driver. The radar and LIDAR sensors are used to detect the position of objects within the blind spot. If an object is detected, the IR sensor is used to check if the object's temperature is roughly around human body temperature. This would indicate that the object is a pedestrian.

This flowchart outlines the way that the central processing system (implemented in the Zedboard) will determine the necessary action based on the data received from each sensor. An ambient temperature sensor will first determine if the outdoor conditions are suitable for each sensor's operating temperature range, in order to determine if the information from the sensor will be accurate. When operating temperatures are met, the system will focus on both the radar and LIDAR sensors to determine the position of nearby objects. If an object enters the blind spot area, and the radar and/or LIDAR detects the object crossing into the Danger Warning zone, the IR sensor will be engaged to analyze the temperature of the object in question. If the object is determined to be human based on body temperature readings, the driver will be alerted immediately and the system will send a signal to engage the brakes if the person crosses into the Critical Action zone. This proposed solution has both advantages and disadvantages to consider, as shown in Table 22.

Table 22: Advantages and Disadvantages of the Third Proposed Solution. Factors such as accuracy, data processing speed and durability taken into consideration.

ADVANTAGES	DISADVANTAGES
Ample coverage of blind spot by all sensor technologies.	LIDAR placement must compensate for rotational operation.
LIDAR and radar are both very capable in position tracking.	Determining whether to rely on Radar or LIDAR for definite object position adds complexity.
IR will likely be faster in detecting humans than image processing via cameras.	Relies solely on IR temperature readings to determine object is human.

While this implementation is advantageous in its ability to cover the blind spot with multiple sensors, the added complexities that arise from this setup are also a disadvantage. If the LIDAR and radar sensors are both to pick up an object, yet the location of the object is slightly different in each, it may be difficult to determine the object's exact location between the two sources. Additionally, the reliance on IR temperature readings will likely be faster than image processing via a camera to detect humans, yet temperature is subject to vary based on different people, as well as the outdoor conditions. This implementation has great potential, and these factors will have to be considered in the final decision.

5.4: Solution Four: Solid-State LIDAR Positioning, Infrared Heat Detection

This system implementation would consist of four components: an ambient temperature sensor, LeddarVu8 solid-state LIDAR module, servo motor, and an IR sensor. The combination of the LeddarVu8 and IR sensors together will have the capability of both object and human detection. A flowchart of this implementation is shown in Figure 28.

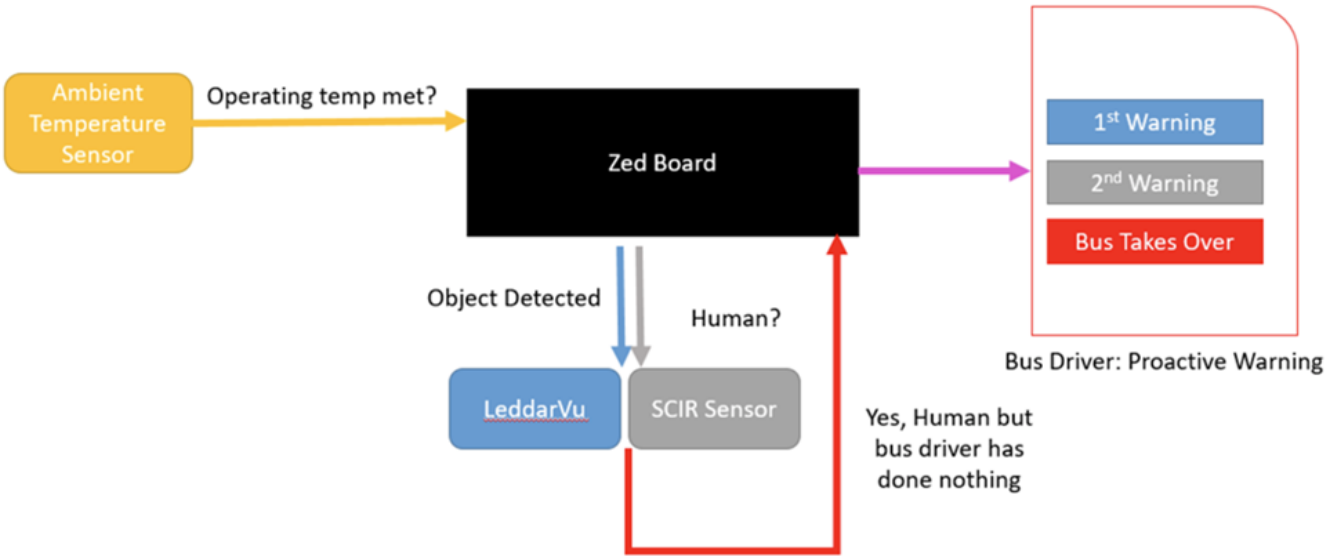

Figure 28: Fourth Solution Data Flowchart. Data fusion on how the information will be processed in order to trigger an alarm and warn the driver. The LeddarVu Solid State LIDAR sensor is used to detect the position of objects within the blind spot. If an object is detected, the IR sensor is used to check if the object's temperature is roughly around human body temperature. This would indicate that the object is a pedestrian.

In this solution, an ambient temperature sensor monitors the outside temperature, which is used to decide which sensors to use. For example, the LeddarVu8 has an operating temperature range from -40°C to 85°C. If the ambient temperature is outside of this range, the LeddarVu8 sensor reading would likely be inaccurate, and thus be ignored by the Zedboard. Note that this is done for every single sensor. Therefore, the ambient temperature sensor serves as a protection mechanism to ensure that all the sensors are used only within the operating temperature range specified by their manufacturer.

In this solution, the LeddarVu8 would be the main sensor due to its robust design, wide operating temperature range, long detection range and multi-object detection capability. Because of these advantages, the LeddarVu8 would be set-up in such a way to cover the blind spot in the most optimized way possible. This sensor can detect multiple objects and report the distance of each one of them. This enables the sensor to detect the closest object location as well as its velocity. Furthermore, if the closest object is inside the predefined blind spot, the sensor data can be used to estimate the trajectory. The distance of the closest object can be used to determine the

zone in which the object resides. Recall that there are three segments within the blind spot: Potential Threat, Danger Warning and Critical Action.

When the detected object is inside of the Potential Threat zone the “First Warning” is issued to the driver and this is a part of a proactive solution. This proactive solution would probably involve a non-intrusive alarm type, an LED array, for example. The LED array would light up in color green to indicate that something is in Potential Threat zone. This notification is non-intrusive and no action is taken at this point. Furthermore, using the input from the LeddarVu8, it is possible to determine the exact segment number in which the closest object resides. Recall that the LeddarVu8 sensor field of view is divided into eight segments, and each segment is capable of detecting the distance to the object within it. This is shown in Figure 29.

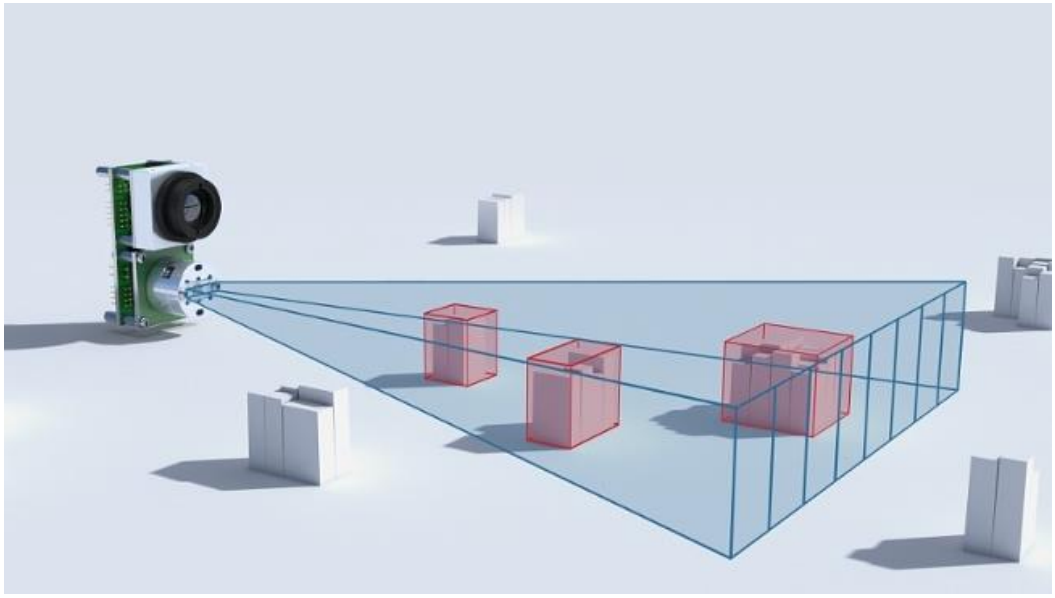

Figure 29: LeddarVu8 Field of View. This figure shows the field of view divided into the eight detection zones. The sensor is able to detect a distinct object within each.

The exact segment number, in which the closest object resides, is determined within the Zedboard software processing and this segment number is converted into a corresponding “servo” angle. Since the LeddarVu8 is capable of detecting the proximity of the object inside of the blind spot, an IR sensor controlled by a servo motor is used to determine if the object detected is indeed a person. For example, assume a 100 degree horizontal FoV (Field of View). Since the LeddarVu8 has eight segments, each segment will occupy 100/8 = 12.5 degrees. In

order to focus in the middle of a segment, the servo angle will be offset by 6.25 degrees compared to that of the LeddarVu8 segment. This is best illustrated with Figure 30.

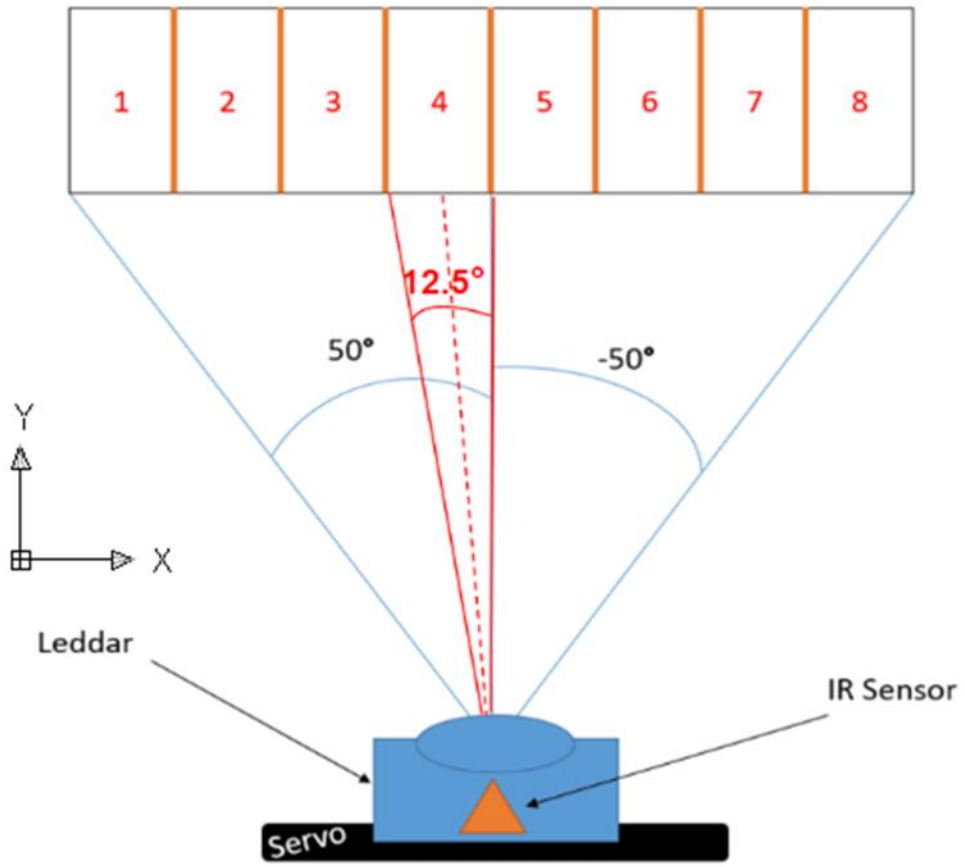

Figure 30: LeddarVu8 and SCIR Sensor Placement. This figure displays the area that the IR will cover while stationary. The IR sensor will rotate on a servo motor-controlled base to adjust the angle of view to be in line with the corresponding segment number of the nearest object detected by the LeddarVu.

From Figure 30, notice that the blue solid lines indicate the field of view for LeddarVu8 and red solid lines indicate the field of view for the IR sensor. Notice that the dashed red line indicates the 6.25 degree offset for the servo motor. The IR sensor will be fixed on the top of the servo motor and therefore this system will be referred to as a "Servo Controlled Infrared Sensor" (SCIR). The SCIR and LeddarVu8 will be aligned on the same z-axis. Based on the segment number in which the closest object resides, the Zedboard will convert this number to the appropriate servo angle, as shown in Table 23.

Table 23: Segment Number vs. Servo Angle. Representation of how much the servo must rotate to bring the IR to the desired position.

Segment #	Servo Angle in Degrees
1	43.75
2	31.25
3	18.75
4	6.25
5	-6.25
6	-18.75
7	-31.25
8	-43.75

Notice that the SCIR sensor, hence the IR sensor, must have a horizontal field of view (FoV) of at least 12.5 degrees to cover the whole segment in which an object is detected. Once the SCIR is pointing at the selected segment, a more precise object temperature is determined. Based on the temperature reading, the Zedboard will classify the object using criteria similar to that of the following: “No person detected”, “Person potentially detected”, and “Person detected”.

The system will keep monitoring the closest object distance to the bus and once the object is inside of the “Danger Warning” zone, the LED array would change to yellow as a part of the proactive solution. In case that the temperature reading from the SCIR sensor was not clear about whether the object is a person or not, the temperature scanning will be repeated to gain a better understanding. If the object keeps getting much closer to the “Critical Action” zone, the LED array would change to red color to indicate that the driver should take some action. Finally, if the detected object is determined to be a person and is inside of the “Critical Action”

zone the system would take control to either correct the course of the bus or bring the bus to a stop (in a final design with mechanical implementation).

Table 24: Advantages and Disadvantages of the Fourth Proposed Solution. Factors such as range and data processing speed were taken into consideration.

ADVANTAGES	DISADVANTAGES
Fast Refresh Rate	Use of mechanical parts for the SCIR system
Long Range	Detection Range affected by the object's reflectivity
Robust	Costly

This implementation is advantageous because it has the ability to cover the blind spot with only two sensors. Due to this factor, and that fact that both types of sensors have a relatively fast refresh rate, this system should prove beneficial in real-time use. Furthermore, due to the solid-state LeddarVu8 technology, this system is very robust to the outside mechanical and environmental conditions. Disadvantages of this system lie primarily in the SCIR system because it involves a servo motor. Due to of the moving parts inside of the servo motor, this system might prove sensitive to the mechanical noise. Also, the reflectivity of the objects will affect the detection range and its precision. All of these factors will be crucial when calibrating the final design.

5.5: Chapter Summary

This chapter focused on discussing the proposed full implementation solutions. It brought all of the material for each of the past chapter into fruition and consideration so that various potential solutions could be discussed. Bring everything into a big picture view made it so that the system implementation could be mapped out for the future application of this specification project. Each proposed solution had the broad details of solution as well as its advantages and disadvantages discussed.

6: Overall Design Choices

Based on the considerations for this project, it was decided that the best solution to implement was the 4th solution, which utilizes LeddarVu8 Solid state LIDAR and a servo controlled IR sensor. In comparison to the other three, this solution provided the greatest balance between a robust, effective, and efficient design. It did not have the drawbacks of a camera system that would require a latent intensive image processing application that would be difficult to implement. The sensors chosen are capable of working in any form of lighting, weather conditions, and have a wide enough temperature range that they can handle the range of temperatures present in the New England area. This solution met all the necessary requirements without any major drawbacks.

When combining the sensing portion with the FPGA and driver alert system, the overall system was planned to break down the blind spot into three distinct zones and make a decision based on which zone a pedestrian has been detected in. The Zedboard FPGA was chosen for its ability to provide sufficient capabilities for software and programmable logic, as well as sensor implementation and data fusion. The driver alert system chosen included an LCD screen to display warnings to the driver. The team was confident that this solution would provide the greatest degree of effectiveness on all fronts of implementation. The team's final design choices were based both on research performed for the proposed solution, as well as other factors that occurred throughout the design process.

6.1: Component Selection

The final choice for object detection was that of a LIDAR sensor. The LIDAR used is the LeddarVu8, a solid state LIDAR available from LeddarTech Inc. based out of Quebec City, Canada. This sensor was chosen for its ability to detect objects in real time with a large field of vision, separated into eight distinct segments. This enabled the detection of multiple objects within the range of the three distance zones defined earlier. Its reliability is marked by a high range of operating temperatures, a compact size, and high degrees of precision and resolution. The exact specifications can be seen in the background section on LIDAR sensors in Figure 14 and Table 13. The LeddarVu8 was also able to be directly interfaced with the Zedboard using SPI communications through PMOD connectors, and required 12 volt power like the Zedboard.

Taking this all into account, it was clear that the LeddarVu8 would provide the team with an effective means for object detection while having a manageable means of communication interfacing, power supply, and cost benefit.

Once an object is detected the next step would be to determine if the object is a human. An IR sensor was the most cost effective means for determining this in real time. The initial IR sensor choice was the Melexis MLX90621. This IR communicates via the I2C protocol and is able to generate a heat map of its surroundings for object detection. However, due to the complexity of the device and timing restrictions, it was deemed that it would not be possible to integrate this sensor within the allotted time. One of the major reasons was that this sensor utilizes the I2C control protocol control, whereas the LIDAR, the device central to the design, was configured utilizing SPI. With the use of PL logic programming on the Zedboard, custom controllers were designed in order to meet the needs of data acquisition, and because the MLX90621 utilizes I2C, this required an entirely separate custom controller that could not be an adaptation of the SPI design.

This I2C controller needed to set up a bidirectional data line (SDA) and a clock line (SCL) that matched the timing requirements for the IR sensor. It was necessary to configure the controller to specify the Zedboard as the master device, and have it set the IR sensor as the corresponding slave device. This would be accomplished by driving the SCL low and having the master device send out 2 sets of 8 bits. The first 8 bits consists of 7 bits for the slave address and an 8th bit determined a read or write function. The 2nd set of 8 bits determined what address within the slave the master would be either reading from or writing to. After every 8 bits there was an acknowledgement buffer that would be configured in the controller. This was the 9th clock cycle that pulls the SDA low to show that the byte has been properly accepted. Depending on the controller design, the master would be controlled for how much of the IR's data is read from either the full range, a particular column, a particular row, or a single pixel. However, it would be desired to have as much control and range as possible when receiving data from the IR sensor. The IR read data stream can be seen in Figure 31. This shows the complexity required for a read and the amount of data that would be coming in. The data line would require a sequence of addresses as outlined in Figure 31. To use the full detection range of the IR, this required a 40 word buffer for detection. Each word consisted of 2 bytes of data and represented the data from a single pixel of detection for the IR [101]. After setting up the controller to handle this timing and

pulling in this data, it would then be required to process all of the information in a real time, functional manner. Taking all of this into consideration, it was clear that on top of the complexity of the LIDAR and data fusion that still needed to be completed, it was in the best interest of the project to seek a simpler IR sensor implementation.

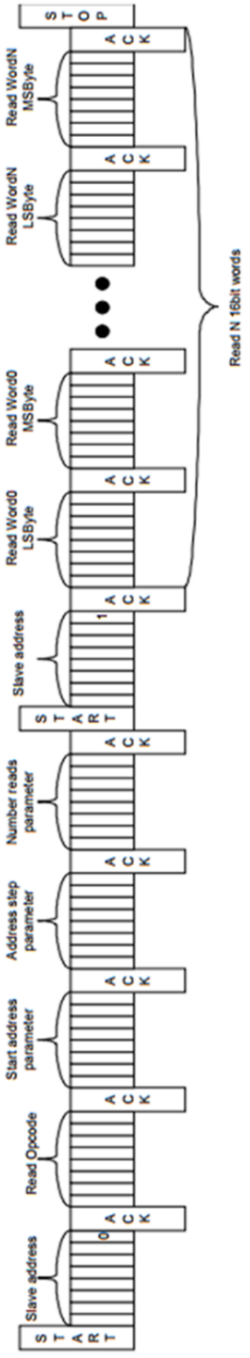

Figure 31: SDA Data Line for a Read Command on the MLX90621 IR Sensor. This transmission must be completed using the I2C protocol. The data packet consists of the addressing and read request data to be sent, as well as the corresponding response to be read in by the master. ***[101]***

The alternative choice for an IR solution that was capable of meeting the project requirements with a simpler means of integration was a passive IR sensor (PIR). The particular sensor that was determined would be best was the Panasonic Electric Works EKMC Series Long Distance Detection Type sensor as shown in Figure 32, along with its timing chart in Figure 33**.** This series of PIR sensors have a very simple function in that they give a digital signal of either high or low, and was pulled high if a person is detected by the sensor. This series of sensors were made specifically for detecting a human being. The sensor determines that a person has entered a space by observing the difference between the body temperature of the object and the ambient temperature of its surroundings. A threshold difference of 4 degrees Celsius is needed between the object and the environment. Also the PIR sensor has a 92 degree field of view which was substantially higher than the original IR [102].

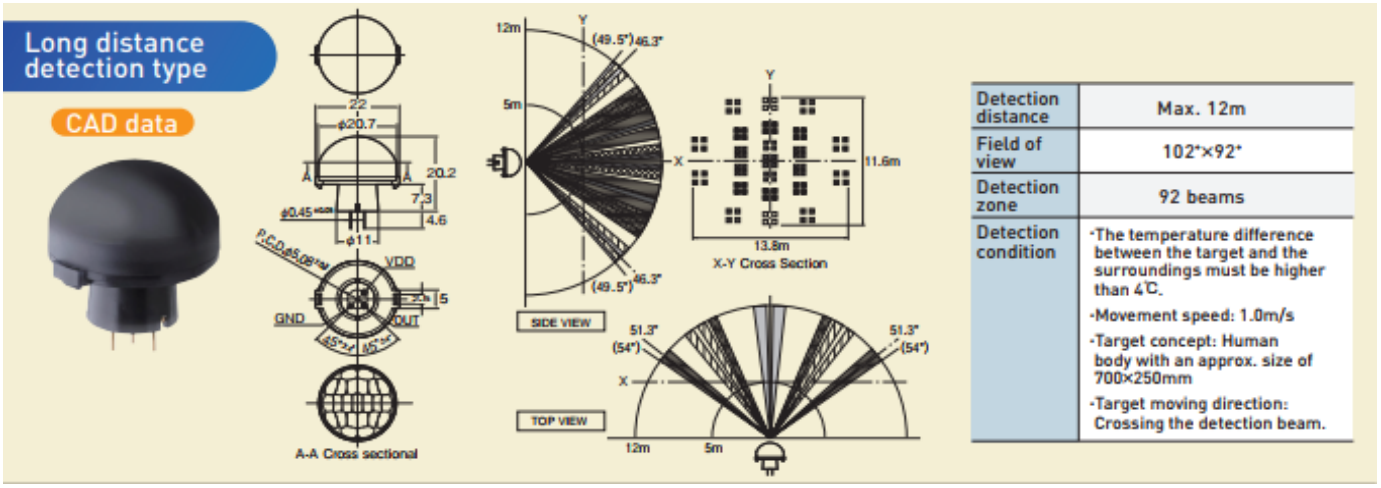

Figure 32: EKMC Series Passive Infrared Sensor, Field of View, range and electrical specifications shown. The X-Y cross section shows the points at which the sensor measures to determine when a moving object's temperature differs from that of its surroundings. [102]

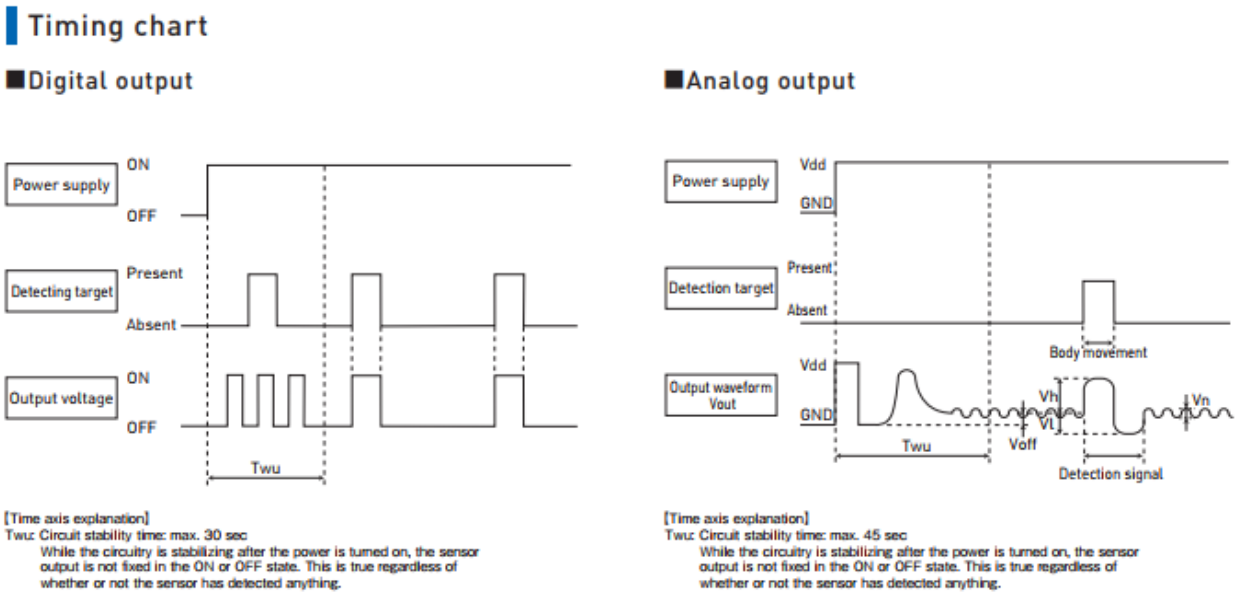

Figure 33: PIR Timing Chart. Representation of how the clock signal changes in the presence of a person. The three initial pulses within the "Twu" (warm-up) period shown on the left demonstrate the uncertainty present during the warm-up time. Once the sensor is warmed up, the presence of a human in the sensor's surroundings triggers a high pulse when the person moves into view. [102]

The sensor has the flexibility of being able to work in either a digital or analog output format. For this design, the digital output was chosen for compatibility with the digital PMOD connectors on the Zedboard. This allowed for a more manageable data fusion with the modules for the LIDAR.

Due to the lower level of complexity of the PIR sensor data output when compared to that of the Melexis sensor, this design choice simplified the required level of control to acquire infrared readings for the detection of a human. The data pin provides a "yes" or "no" answer for whether a person is within the sensor's field of view. When the pin is driven high, the object detected was determined to be human. This much simpler PIR sacrificed some control and the ability to manually analyze a heat map for a simple 1-bit logic. However, this was sufficient for the initial design to ensure that the system was able to accurately detect when a person was present in the sensor's view.

Initially, the IR sensor was going to be mounted on a servo so that it could be directed towards the LIDAR's zone of detection with the closest object. However, with the change in IR sensor and a now greater field of view, it was deemed that the cascading of two PIRs would

provide sufficient coverage of the space without the use of a servo to change the angle of view. Therefore, the PIR sensors were attached to the structure of the system just below each corner of the casing that contained the LIDAR and Zedboard. This two-sensor combination enabled a greater range for detection and essentially spread the field of view of the PIR. In Figures 34 and 35, the top and side view of the detection area can be seen, including how the two fields of view overlap for greater coverage. Figure 36 displays the overall disparity of the X-Y cross section for detection. Taking this disparity into account, one can see how overlapping the detection areas of the two separate sensors makes up for the areas in which each sensor is lacking a detection beam. The white space in the at the side and top views indicate the holes in the field of detection. The overlap helps to fill in these holes and expand the overall covered area.

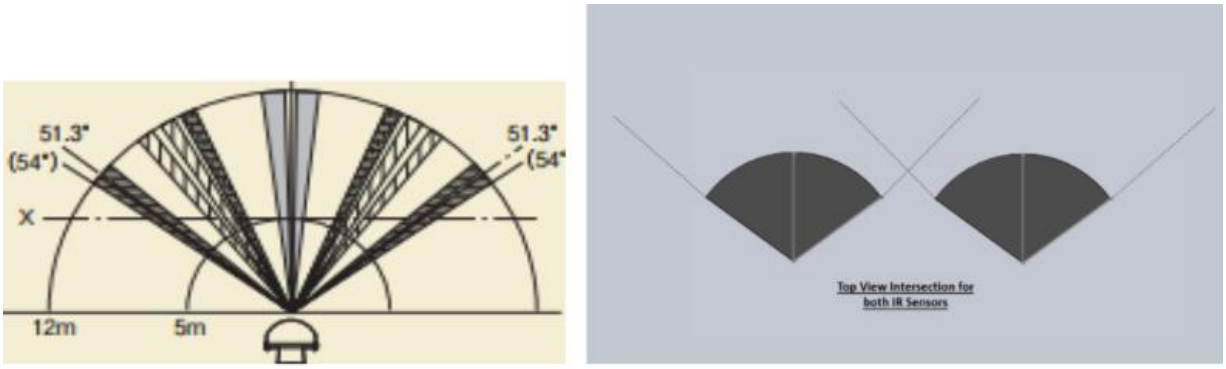

Figure 34: Top View of the IR Detection Area. The areas separated by gaps in the sensor's field of view show where the sensor measures to determine when a moving object's temperature differs from that of its surroundings. With two sensors positioned next to one another, as pictured on the right, the overall field of view of the IR arrangement is increased significantly. This overlap also accounts for the gaps in a single sensor's field of view. [102]

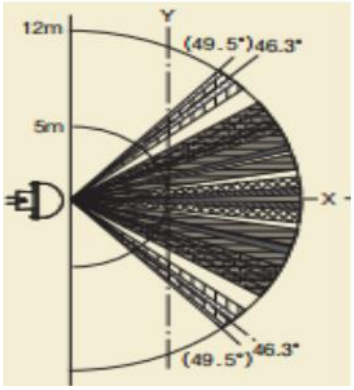

Figure 35: Side View of IR Detection Area. Detection gaps in the vertical field of view are smaller than that of the horizontal field of view. [102]

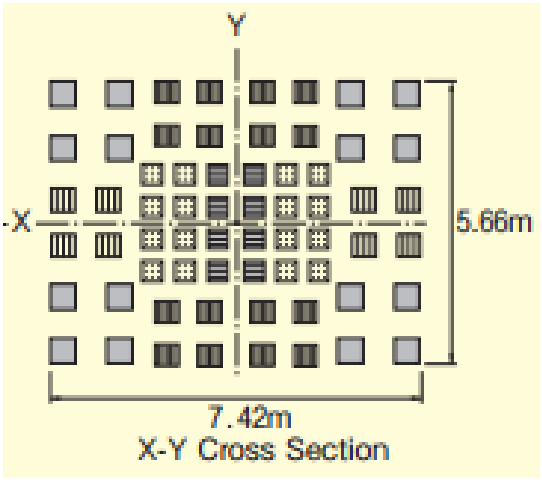

Figure 36: Cross Section View of the IR Detection Area. The X-Y cross section shows the points at which the sensor measures to determine when a moving object's temperature differs from that of its surroundings. [102]

The PIR detects from the top of the sensor's curved cap. Therefore, the sensors were mounted to the side of the bus structure so that the beams emit outward during operation. This arrangement has been attached to the structure as indicated in Figure 37.

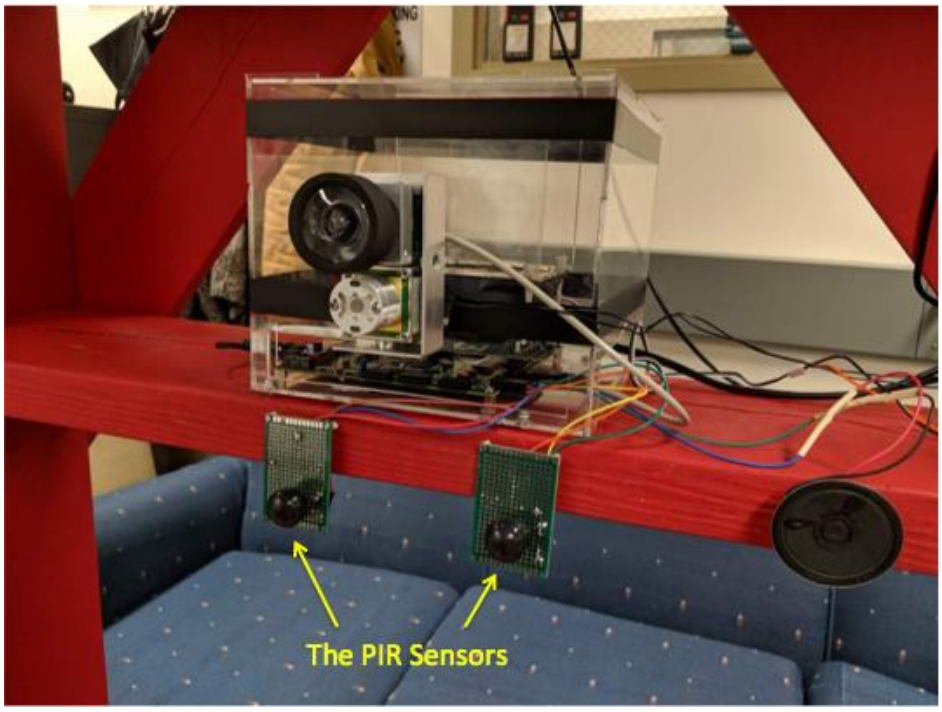

Figure 37: PIRs Attached to the Side of the Structure. PIRs connected to the Zedboard through the PMOD connectors. The PIRs are positioned to overlap their field of view, as planned.

The original design plan called for an accelerometer to measure the turning movement of the city bus. The PMODACL2 accelerometer was chosen based on its low voltage and current requirements as well as its SPI interfacing that could have been utilized in conjunction with the PMOD connectors on the Zedboard. The sensor's characteristics can be seen in Table 25.

Table 25: PMODACL2 Accelerometer Voltage, Current, & Interfacing Characteristics. The Zedboard is capable of providing the necessary power, and also utilizes SPI. [103]

Voltage	1.6 V - 3.5V
Current	2.5 µA
Interface	SPI

The PMODACL2 is an ultra-low power, three axis, micro-electronic mechanical system that consumes less than 2µA at a 100 Hz output data rate and 270nA when in a "motion triggered wake-up mode." The device is also capable of freefall detection as well as power saving features through its motion activated sleep and wake modes.

The accelerometer was relevant to the pedestrian detection and avoidance system as it determined various aspects for the actual motion of the bus. Having a three axis system determines the precise direction in which the bus is moving so that the system could make accurate judgements on the corresponding decision making. The turning of the bus is the critical time for the system to be in use as that is when there is the highest number of accidents. One of the main distinguishers is whether the bus is turning, or simply changing lanes. The accelerometer would have provided this information so that decision making can be made as to whether or not the bus is turning, or changing lines on a standard road or on the highway. Although this sensor was something that was important in a final design for a system of this make, it was not a critical design element for a proof of concept demonstration. Due to time constraints and complexity of the sensor for less than critical functional elements, it was deemed that it would not be necessary for the proof of concept.

Similar to the accelerometer, a temperature sensor, although necessary in a final design, was deemed not necessary in a proof of concept evaluation. However, the following considerations were taken into account to determine an effective temperature sensor and the role

it would have played in a final design. The DHT22 AM2303 temperature sensor would have been utilized so that the current operating temperature is within the specifications of each of the sensors in use. The purpose of this sensor would have been to ensure that the data provided from all operating sensors is as accurate as possible. This sensor has the following specifications as shown in Table 26.

Table 26: DHT22 AM2303 Sensor Specifications & Characteristics. The Zedboard's PMOD connectors are capable of providing the necessary power and data communication. [104]

Model	AM2303
Power Supply	3.3-6V DC
Output Signal	Digital signal via single-bus
Sensing element	Polymer humidity capacitor & DS18B20 for detecting temperature
Measuring range	humidity 0-100%RH; temperature -40~125 Celsius

As shown in Table 26, this sensor would receive sufficient power from the Zedboard, as the PMOD connectors can supply 3.3V and up to 100mA, while the DHT22 needs 1.5mA [104]. The compatibility with the system design, low operating current, and reliability of this sensor made it a cost effective choice. Although this would have been a helpful feature for the system, it was not a necessity due to the fact that the operating temperature of the sensors used provide a wide range. These temperatures are outlined below for each sensor.

The LIDAR can operate between -40 degrees Celsius, up to +85 degrees Celsius, according to its user guide [105].

The passive infrared sensor to be implemented can operate between -20 degrees Celsius, up to +60 degrees Celsius, according to its datasheet [102].

While the temperature sensor has a measuring range of negative 40 degrees Celsius, up to 125 degrees Celsius [104]. The original idea was to have the temperature sensor implemented in order to assure high accuracy of the MLX 90621 Thermal Infrared sensor due to its specifications. The MLX 90621 has operating temperatures between -40 degrees Celsius up to +85 degrees Celsius, but it has the highest accuracy (+-0.5 degrees Celsius) when it is operating between 0 degrees Celsius up to +40 degrees Celsius [101]. By implementing the temperature

sensor, it would be possible to ensure a high accuracy while using the MLX90621 and determining the temperature at which the sensor would have been operating.

Each of the datasheets for the sensors and other electrical components were analyzed to determine the maximum power rating which came out to be 56 watts (W) with the maximum current being 5.17 amps (A). These results corresponded to the use of the Zedboard, LIDAR, Melexis IR sensor, DHT22 temperature sensor, accelerometer and servo motor. There were three different operating voltages that were taken into account when designing the power supply: 12, 6 and 3.3 volts. The Zedboard required 12 volts and 4 amps maximum with a maximum power rating of 48 watts. On the Zedboard were five PMOD connectors that can supply 3.3 volts and 0.1 amps, which was sufficient for the accelerometer, IR and temperature sensor, so these sensors could be directly connected to the Zedboard [57]. The other device that needed 12 volts was the LIDAR which required 0.167 amps and the maximum power rating was 2.004 watts [105]. Lastly, the servo motor needed six volts and one amp with a power rating of 6 watts. Table 27 shows the power requirements for the original system.

Table 27: Detailed Chart Showing the Power Requirements for the Overall System Design. These calculations of the total power and current were used to determine the required power supply for the system.

Peripheral	**Voltage [V]**	**Current [A]**	**Power [W]**
Zedboard	12	4	48
LIDAR	12	.167	2.004
Accelerometer	3.3	.0000025	.00000825
IR	3.3	.009	.0297
Temperature Sensor	3.3	.0013	.00429
Servo Motor	6	1	6
Total Power [W]			56.03799825
Total Current [A]		5.1773025	

When determining the power supply needed for the system, many different approaches were considered. It was first determined whether the system could be powered by the alternator or any of the batteries on the bus itself. The alternator could provide the proper voltage and current requirements with just the need for a voltage regulation circuit. The signal from an alternator, however, could be very noisy, and once implemented into a bus this would need to be accounted for. Alternators are also very prone to brownouts. Another problem was knowing how the team may connect the pedestrian detection and avoidance system to the alternator, because a bus was not available to work on during the system development and testing. Determining the best solution for connecting the system to an alternator and properly powering it within the scenario of a bus-mounted design is something that will be discussed further in the recommendations section.

After conducting some research and a series of personal interviews with a former bus driver (who wished to stay anonymous), it was determined that Transdev uses mainly the Xcelsior XN60/40 and the Orion VII buses. The Orion VII contains a 12 volt battery, which could provide power to the system. Since buses are considered a public space, there are no 12 volt auxiliary (cigarette lighter) plugs that could be used, but newer buses are starting to use USB ports that allow passengers to charge their phones. If the fuses for the USB ports allow the amperage needed and output the correct voltage then the pedestrian detection system could theoretically be powered from the same source.

Since there was no available functioning bus to test the pedestrian detection system, it was determined there were two viable options: purchasing a large 12 volt battery, or selecting a power supply that connected to a wall outlet. After some consideration, it was determined that buying a power supply was better than a battery because the amount of testing done may have drained the battery quickly, which would require multiple batteries to be purchased and charged on rotation. Another consideration was that a stationary bus model would be used, which allowed easy access to a nearby wall outlet. The only detrimental aspect in selecting a power supply is that it is far less portable than a 12-volt battery. It was also determined that the power supply selected would need to have a power rating twice the calculated value of 56 watts and a current rating roughly 20% greater than the calculated current rating of 5.17 amps to allow headroom in the power specification. These two requirements provided a safety net in case the actual power drawn exceeded the calculated value.

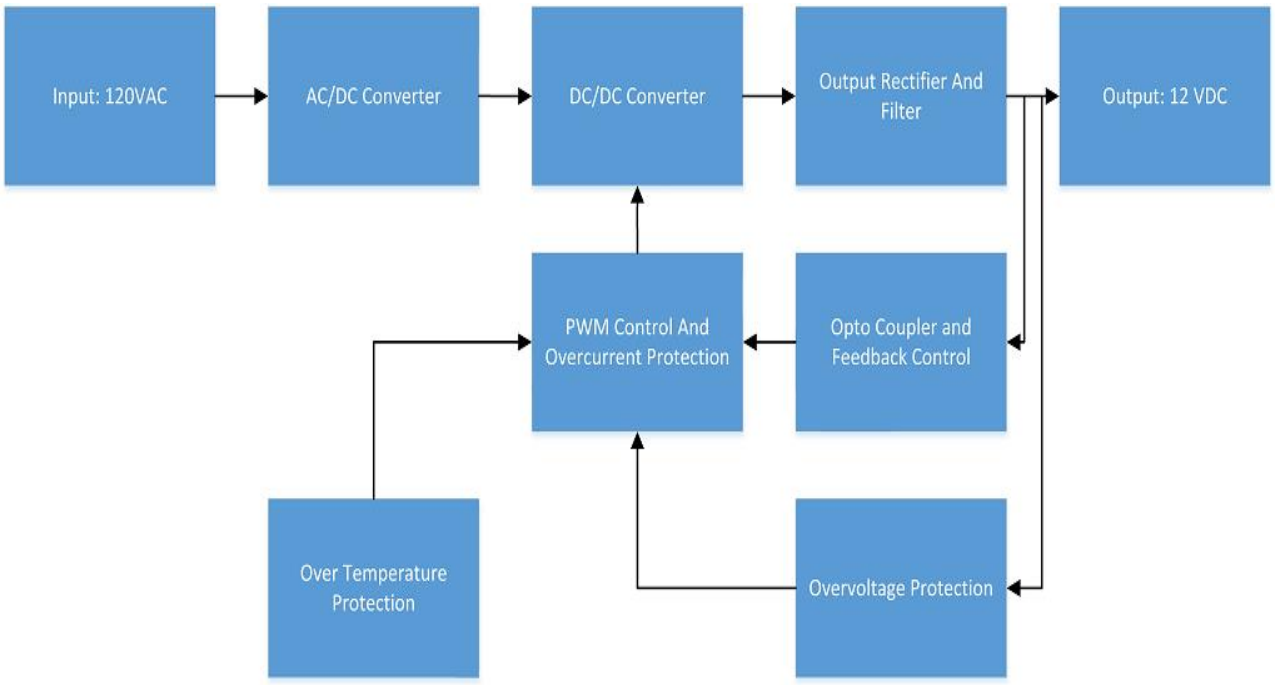

Figure 38: Delta Electronics PMC-12V100W1A Power Supply functional block diagram. The power supply takes in an AC signal and outputs a DC signal. The supply has built-in overcurrent, overvoltage, and temperature protection to ensure safety [106].

The power supply that was selected is the PMC-12V100W1A from Delta Electronics. A block diagram of this device's internals may be seen in Figure 38. It provides 100 watts, 12 volts and a maximum current of 8.33 amps. It converts the 120 volts AC signal from a common wall outlet and converts it to 12 volts DC [106]. These specifications provided almost twice the amount of power that was calculated and more than 20% of the calculated current rating. Another added benefit from the power supply is that it has two 12 volt output rails, which allows the servo motor to be powered by one rail with a step down converter to provide six volts, and all of the other components to be powered by the other rail. This simplifies the overall circuit by not having to attach a voltage regulator on one rail to step down the voltage while everything else on that rail needs 12 volts. The step down converter that would have been implemented for the system was the SMAKN DC/DC Converter, which takes in 12 volts and outputs six volts and a maximum of three amps [107]. The final design ultimately did not require this aspect due to the change in IR sensor not requiring a servo motor to manipulate the IR's orientation. However, if

in the future the need for an IR that did not have the range of detection of the PIR sensor was used, this converter could easily be implemented to meet the power requirements of a servo so that it could manipulate the orientation of an IR and effective increase its range.

The overall system requires an effective casing to neatly house the various sensors and Zedboard, provide room to connect everything to the power supply, and also be nonconductive. Due to these requirements, acrylic was utilized for the casing material, as it met all the necessary requirements, and could be effectively and flexibly implemented. More details on the design and construction of the casing will be discussed in the methodology.

The alert system was deemed to be most effective through the cascading of visual and audible alerts. The system has a progressive color display and sound system to alert the driver. Initially there were considerations for an external sound or visual alert for the pedestrians, but this was deemed ineffective due to potential annoyance factor, and that people may eventually become accustomed to it and filter it out with the rest of the city noise. The first driver alert is when there is no object detected in a zone of potential danger by the system and the monitor will display the color green. The second is when an object is detected in a zone of potential danger, the “Danger Warning” zone. At this time, the monitor will display the color yellow. The third is when there is a pedestrian within the “Critical Action” zone, this will display the color red on the monitor. This zone requires immediate action in order to avoid an accidental collision. An audible secondary alert, which will occur only in the critical action scenario, was implemented to emphasize the need for action. The cascading of sound and visual was deemed as an effective way of alerting the driver without distracting them. The sound might normally be found as annoying if it were to be triggered repeatedly, but the limited use only in critical situations prevents the noise from becoming a nuisance. The screen used for the pedestrian detection and avoidance system is a 16:9, 800x480 color LCD display monitor with a VGA output. This screen in particular was chosen as it provides a large enough visual, has a VGA connector that can be utilized in conjunction with the Zedboard, and was easily powered by the existing power supply.

Figure 39: 800x480 resolution VGA screen that was used to display a visual alarm. Screen turns green (No Danger), yellow (Danger Warning zone) or red (Critical Action zone) based on the location of the object.

Figure 40 shows the buzzer used for creating the audible alert for the driver. The buzzer was easily connected to the PMOD connectors on the Zedboard, and code could be written to drive it along with the rest of the alert system. This was the reasoning behind selecting this simple yet effective buzzer.

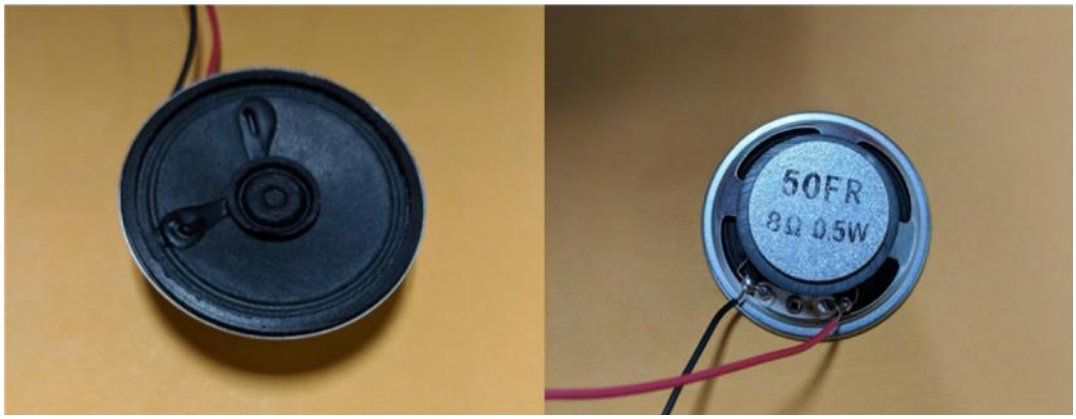

Figure 40: 8Ω, 0.5W Buzzer that was used to sound an audible alarm. Sounds when a human is detected in the Critical Action zone.

On a real city bus, the buzzer and the screen would be mounted in close proximity on the dashboard to provide the driver with a noticeable warning in the event of possible collision.

6.2: Chapter Summary

This chapter provided an overview of the design choices that the team determined to be best for creating an initial prototype and proof of concept of a bus-mounted pedestrian detection

system. By considering the advantages, disadvantages, and complexities of the components originally proposed, the team maintained use of most of the originally planned components, such as the Zedboard and LeddarVu8, while finding new replacement solutions where necessary, such as the passive infrared sensors. The final design plan consisted of the LeddarVu8 solid-state LIDAR sensor and two passive infrared sensors to transmit data samples to the Zedboard for the computation of object distance and human detection, and to drive the system's decision making. The plan for alerting the driver consisted of a VGA-connected LCD screen to flash different colors depending on the level of danger, along with an audible buzzer for critical action scenarios. A 12 Volt DC power supply was chosen to provide power to the overall system. All components were planned to be mounted on a wooden frame to mimic the side of a city bus. With the overall design choices selected, the team was then able to proceed into designing a functional integrated system.

7: Methodology & System Implementation

The pedestrian detection and avoidance system was designed based on certain criteria that the system had to meet. All components were chosen based on their functionality, efficiency and their cost. A diagram of the proposed sensor placement on a city bus, as well as an overall block diagram of the hardware connections can be seen in Figures 41 and 42. On a real-world city bus, rather than a scale model, the team would aim to place the system under the driver's side window. This placement would maximize the coverage of the bind spot to the side of the bus, due to the wide field of view of both the LeddarVu8 LIDAR sensor, and the passive infrared sensors. There were five separate functional sections in the system. Each of the functional sections shown in the connection block diagram can be seen with an individual color. The power supply blocks are shown with the green boxes and arrows, the programmable logic block is shown in orange, the sensor blocks are shown in purple, the warning system block in dark blue, and the cooling system in light blue. The power supply is a 120VAC-12VDC converter that has two 12 volt rails, with each rail supplying different devices. One of the rails was used to connect an extension cord on which the Zedboard, the LIDAR and warning screen were connected. In a junction box mounted to the system, these wires were neatly packaged and run through to maintain a neat configuration. The other 12 volt rail was used to wire the fan, which is used to cool the Zedboard.

While the LIDAR and the warning screen were powered up by the power supply, the information was transmitted through connections to the Zedboard. The LIDAR was connected to the Zedboard via PMOD A port using the LIDAR controller and the warning screen was connected to the VGA port using the VGA controller. The PIRs were both powered up through the PMOD B port of the Zedboard and they transmitted data through those connectors to the internal logic. Lastly, the buzzer was connected to Zedboard via PMOD B port.

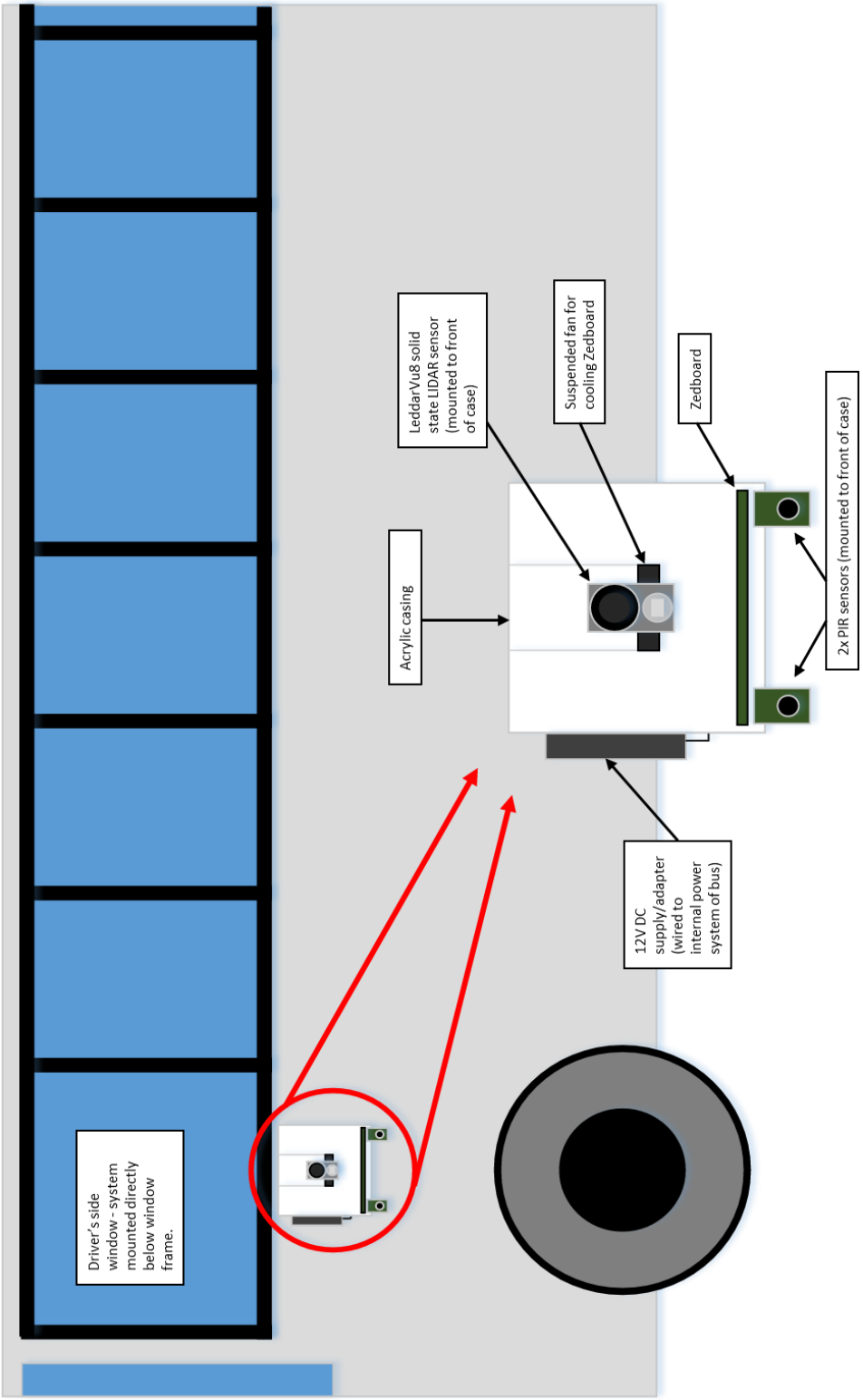

Figure 41: Proposed Placement of System on City Bus. This diagram shows how the team would work to place this system on a real-world city bus. By placing the system underneath the driver's side window, the field of view of both the LeddarVu8 and PIR sensors will give the greatest coverage of the blind spot.

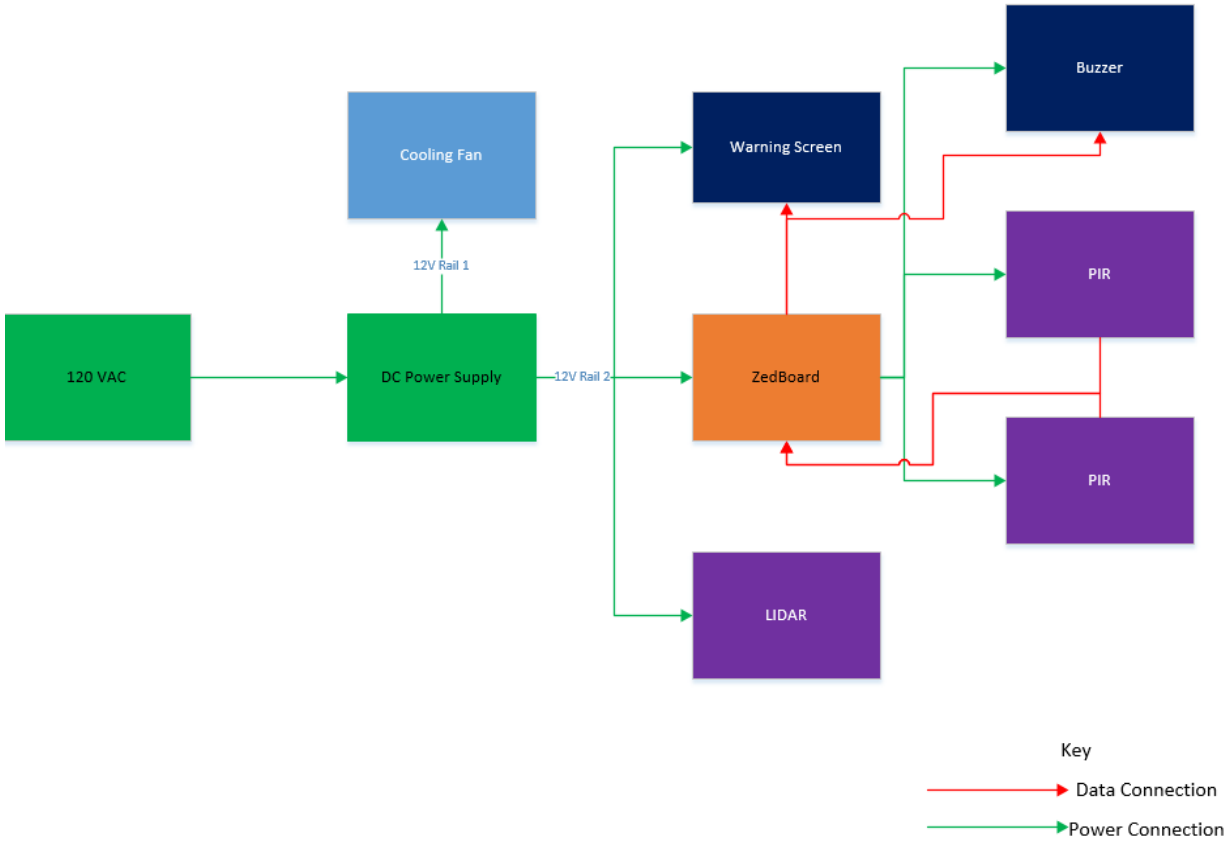

Figure 42: Top Level Connection Block Diagram. Representation of how the connections between the sensors and components are arranged. Colors distinguish between the five individual sections of the overall design: The power supply blocks are shown with the green boxes and arrows, the programmable logic block is shown in orange, the sensor blocks are shown in purple, the warning system block in dark blue, and the cooling system in light blue.

The system works by first determining if an object is less than eight feet away using the LIDAR, which provides distance readings to the Zedboard. If an object is not less than eight feet away, then the system checks to see if it is less than thirteen feet away. If the object is more than thirteen feet away, then the VGA controller would send a signal turning the warning screen green indicating that there is no danger. The screen would turn yellow if an object is within the range of eight to thirteen feet, the danger warning zone, of the system. When an object is within eight feet of the system, the critical action zone, then the PIRs would be triggered to see if the object is human. If a human is present, then the warning screen would turn red and the buzzer would

sound to notify the driver, if not then the screen would turn yellow. The overall flow of decision making based on sensor data can be seen below in Figure 43.

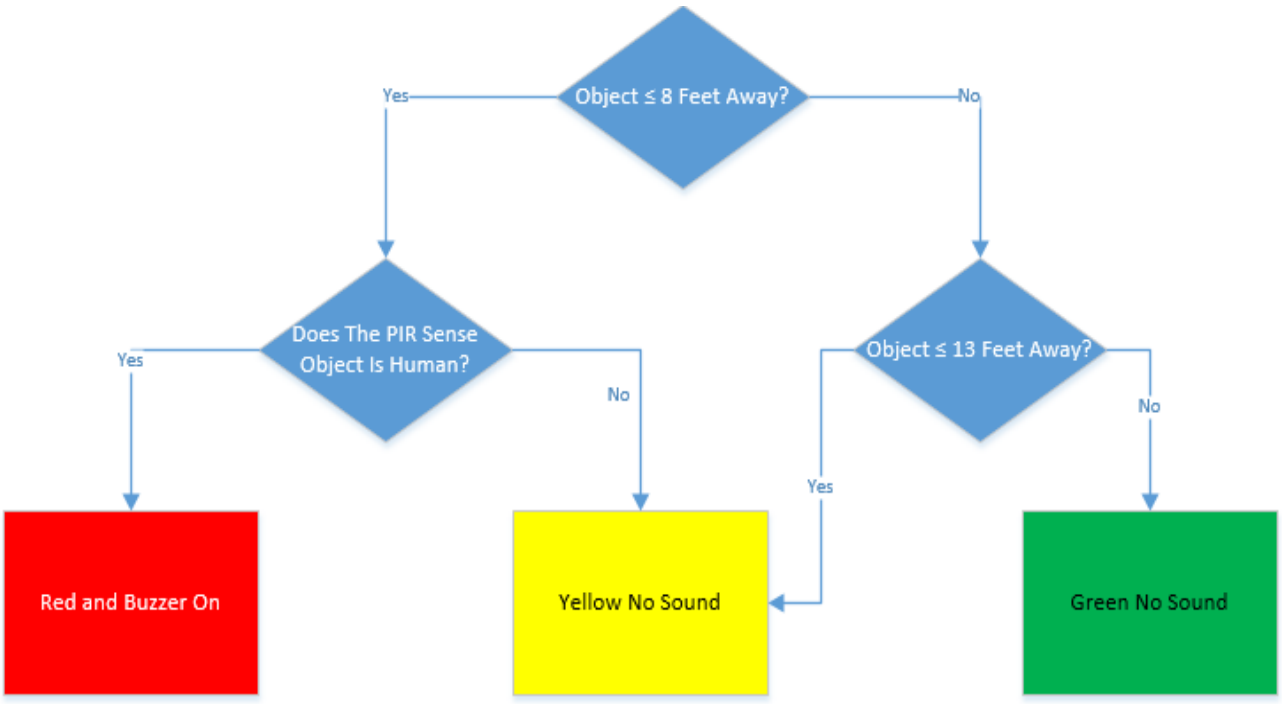

Figure 43: Logic Flow Diagram. The driver will be notified by the screen color changing to its respective color based on the decision made by the system. For example, if the object is less than 8 feet away and the PIR senses that the object is human, the VGA display will change to red and the buzzer will turn on.

In order to design the system, it was important to develop a detailed block diagram consisting of all of the logic modules with their corresponding connections. Figure 44 demonstrates the block diagram used when implementing the logic with Xilinx Vivado Design Suite on an Avnet Zedboard. The following sections discuss the design process for each of the logic modules, and will reference individual components of this overall block diagram throughout the chapter.

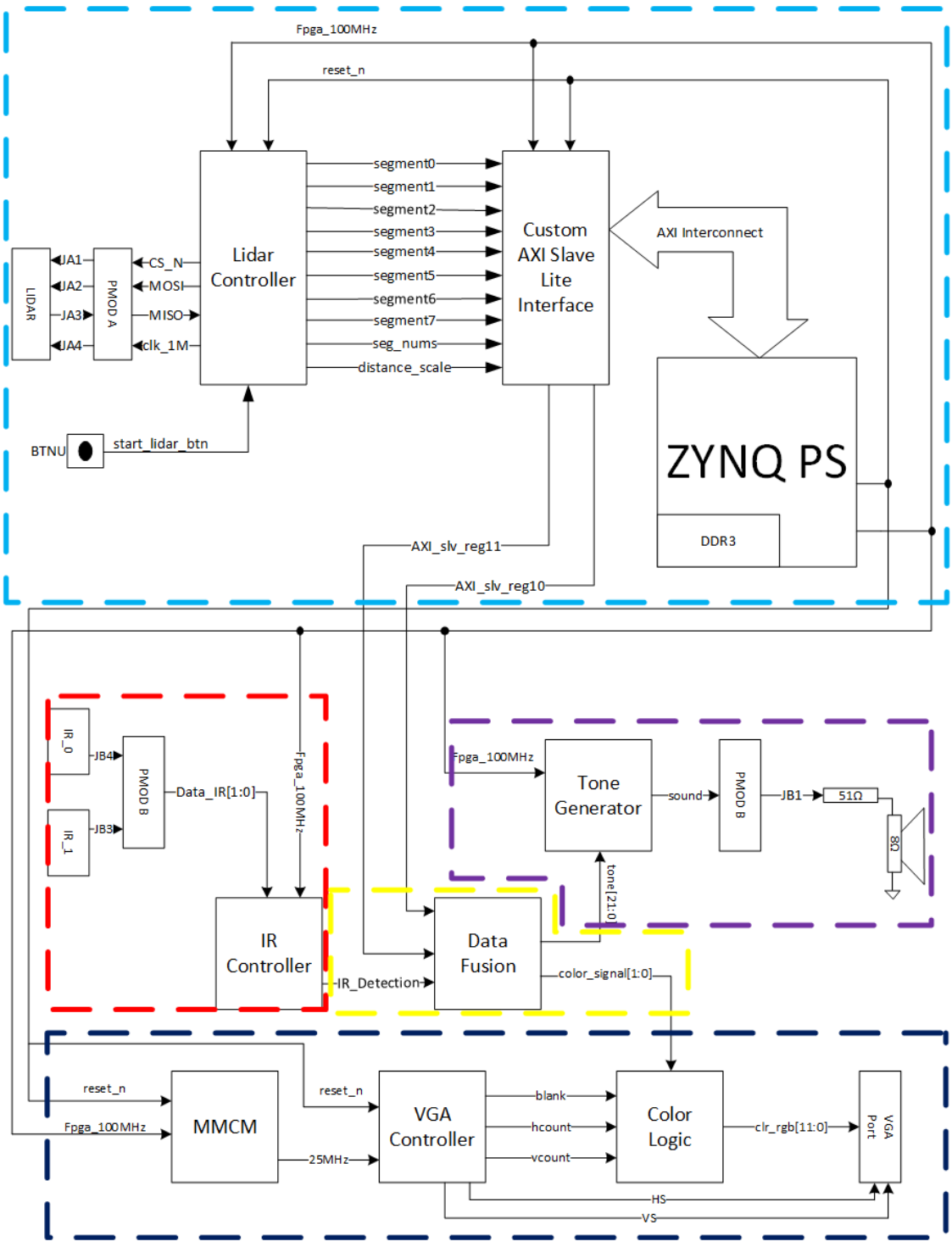

Figure 44: Detailed Hardware Block Diagram. This shows the complete block design with its corresponding modules and connections. Each module provides unique functionality that makes the top-level design possible. LIDAR, IR, Data Fusion, Audible Alarm and Visual Alarm logic implementations are outlined in light blue, red, yellow, purple and dark blue, respectively.

7.1: LIDAR

A critical factor that drove the team's decision to implement a LIDAR-based ranging system was the need for accurate and fast distance readings. The implementation of the LeddarVu8's segmented distance mapping and Zedboard's PS-PL communication aimed to meet this need by providing a system with precise acquisition of the sensor's surroundings and ample computing power to process incoming readings. This was accomplished by designing a custom SPI control driver in Verilog Hardware Description Language (HDL), along with a routine processing program in C. All custom code modules were written and compiled using the Xilinx Vivado Design Suite and Xilinx Software Development Kit (SDK). Figure 45 shows the logic from the detailed hardware block diagram that pertains to the LIDAR implementation.

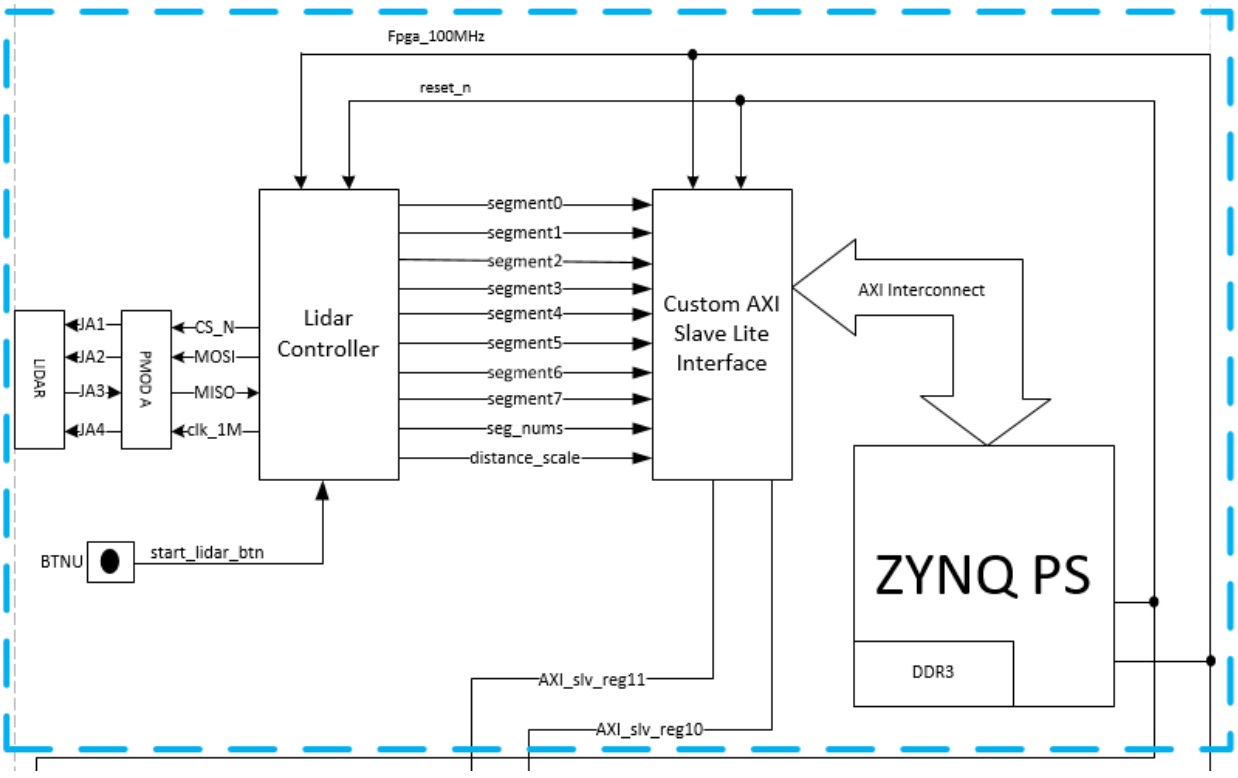

Figure 45: Detailed Hardware Block Diagram in Regards to LIDAR Implementation. The LIDAR sensor is connected through JA1-4 PMODA ports to the Lidar Controller. The Lidar controller utilizes an SPI protocol to retrieve raw distance readings from the LIDAR. These raw distance readings are passed to the ZYNQ PS via Custom AXI Slave Lite Interface. ZYNQ PS computes the actual distances in meters, compares them to the predefined zones and returns the results via AXI_slv_reg10 and AXI_slv_reg11. Note that BTNU is a push-button used to start the data acquisition.

The use of Verilog HDL enabled the creation of a logic-based driver specific to the timing and control signals required by the LeddarVu8 LIDAR sensor. The primary interface for communication between the LeddarVu8 module and the Zedboard was the Serial Peripheral Interface Bus (SPI) protocol. By design, the SPI protocol utilizes four primary control lines for data transmission: clock, chip select, Master Out Slave In (MOSI), and Master In Slave Out (MISO) as shown in Figure 46 below.

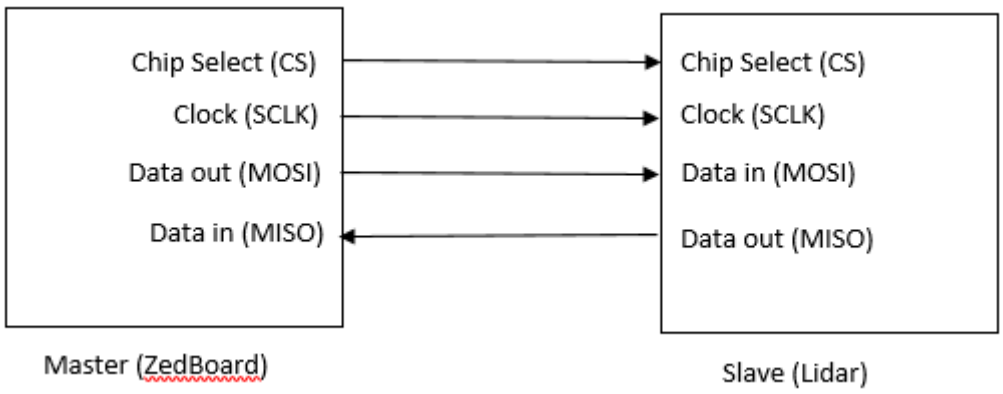

Figure 46: Four-Wire SPI Protocol is used to load data from LIDAR into the Avnet Zedboard registers. The protocol consists of a clock (provides timing), a chip select (used to select the device), MOSI (used to send commands/data to LIDAR from Avnet Zedboard), MISO (used to load/receive data into Avnet Zedboard from LIDAR)

When the master device would like to send data to or request data from the slave device, it must pull the chip select line low, transmit data serially at one bit per clock cycle, then complete the transmission by pulling chip select high once again. In the system design, the Zedboard was programmed to act as the master device, which requested data to be returned by the slave LeddarVu8 module. With respect to the general configuration of the standard SPI data lines, the LeddarTech's LeddarVu8 device was designed with its own specific protocol requirements that must be satisfied in order to obtain a stable data transmission from the device. As specified in the LeddarVu user guide, in order to receive data from the LIDAR sensor, a read data command must be issued from the master device that includes the "read" opcode, memory address at which the requested data is stored, and the size of the data packet that should be returned to the master in bytes [105]. There must then be a break between the time where the request is sent and when the master device expects a stream of data in return. This allows for the

LIDAR module to prepare the necessary data for transmission. Finally, the module will respond with the requested data in full, starting at the specified address for the number of bytes requested. A chronogram of the described data protocol from the LeddarVu user guide may be found in Figure 47.

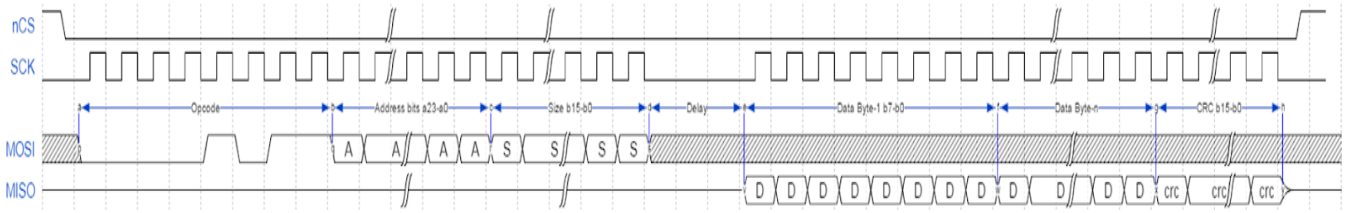

Figure 47: LeddarVu8 Read Data Protocol Chronogram [105]. This timing diagram illustrates the specific timing requirements for the LIDAR's SPI signals including clock, chip select, MOSI and MISO.

The transmission described above may be performed at a clock cycle rate of anywhere between 500 kHz and 25 MHz.

The factor in which the LeddarVu's implementation of the SPI protocol differs from other SPI interfaces is the break period between request and receive. The user guide specifies that this break period can be held for 1 millisecond, upon which chip select must remain low (active), yet the clock signal must be halted. This requirement set a specific design parameter when planning the control logic, as the timing constraint would have to be satisfied in order to collect data with consistent accuracy.

The LeddarVu stores all distance readings and configuration values in an arrangement of local memory, divided into multiple memory banks. The LeddarVu user guide provides a reference table of the memory banks with their associated addresses and sizes. Extracts from the table are as follows:

Table 28: LeddarVu8 Data Memory Banks [105]. This table demonstrates the LIDAR's memory map which is divided in four memory banks each dedicated to store specific type of data. For example, Bank 5 starts at 0x00400000 base address which is 128KB wide. This bank is read-only and stores the device information and constants data.

Bank Number	Start Base Address	Bank Size (KB)	Access	Description
0	0x00000000	1024	Read/Write	Configuration data
5	0x00400000	128	Read Only	Device information and constants
13	0x00500000	1024	Read Only	Detection list
19	0x00FFFB00	1	Read/Write	Transaction configuration

One critical location in the data memory that is regularly accessed for acquiring distance readings is the Detection List. The detection list holds the raw distance samples recorded by the LIDAR module, which are refreshed and overwritten at the same memory location for each of the eight segments. Tables of the detection list memory bank and detection structure format as shown in the LeddarVu User Guide may be found in Tables 29 and 30.

Table 29: LeddarVu8 Detection List Memory Bank. This is Bank 13 with 0x00500000 base address and it stores all the detection/acquisition data. For example, the raw detections are stored in detection list array starting at 0x0050000C. [105]

Offset	Length	Type	Description
0	4	uint32_t	Timestamp: in ms since power up
4	2	uint16_t	Number of detection (N)
6	2	uint16_t	Current percentage of light source
8	4	uint32_t	Acquisition options
12	N * detection structure size	Array of detection structure	Start of detection list array

Table 30: LeddarVu8 Detection Structure Format. Raw readings for one of the eight LIDAR segments are stored in the Detection Structure format as shown below. Table 29 shows that the raw reading of one segments consists of 12 bytes. Therefore, to get all 8 segment readings one has to start at 0x0050000C and read length of 96bytes. [105]

Offset	Length	Type	Description
0	4	uint32_t	Distance expressed in distance scale. To convert to meters, the distance must be divided by the distance scale.
4	4	uint32_t	Amplitude expressed in raw amplitude scale To convert the amplitude to count, it must be divided by the amplitude scale. Amplitude = Contents of this register/(Amplitude Scale Register + 8192)
8	2	uint16_t	Segment number
10	2	uint16_t	Bit-field detection flags: Bit 0: Detection is valid (will always be set) Bit 1: Detection was the result of object demerging Bit 2: Reserved Bit 3: Detection is saturated

The detection data for the eight segments are stored in the form of a common structure of 12 bytes, containing the raw distance reading, amplitude of sensor signal, segment number, and validity flags. To obtain a raw distance reading that includes one or more segments of the LIDAR sensor's full field of view, one can issue a read command to the base address of memory bank 13, with an addressing offset of 12 bytes and a size of n*12 bytes, where n is the number of segments to read, starting with segment seven (following 7 to 0 indexing) [105]. This directs the LeddarVu8 to return n*12*8 bits of data starting at the base address of the detection structure array, which ends with the last bit of the n'th segment's data. All distance acquisitions are initially transmitted by the LeddarVu8 module in a raw, unitless data format. In order to convert the raw data into a measurable format, the sample must be divided by a corresponding distance scale value. This scale value is stored in the Device Information and Constants memory bank (bank 5) at byte offset 354, and may also be read by issuing a read command to the LeddarVu8 module. With the above requirements taken into consideration, it was then possible to begin

developing a LIDAR sensor control driver through the FPGA hardware logic, written in Verilog HDL.

With full understanding of the data acquisition requirements, it was possible to develop the LIDAR controller logic block as shown in Figure 48 below.

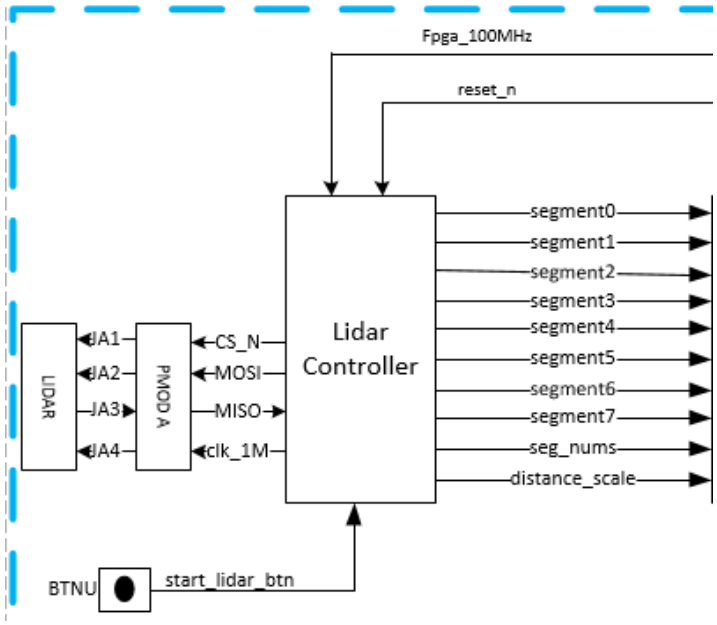

Figure 48: Block Diagram of LIDAR Controller. The LIDAR sensor is connected through JA1-4 PMODA ports to the Lidar Controller. The Lidar Controller utilizes an SPI protocol to retrieve raw distance readings including the distance scale from the LIDAR. The acquired sample of readings is then passed out to the Custom AXI Slave Lite Interface. Note that BTNU is a push-button used to start the data acquisition

In order to acquire regular segment distance samples in a routine format, the design for the LIDAR controller employed a digital logic based state machine utilizing both synchronous and combinational elements. The logic controlling the shift between states is set up as combinational logic, meaning that the status of the current state and next state will be checked and updated whenever any of the conditions listed in the "always @" sensitivity list are altered. This enables the use of push buttons and counter-based enable signals to change the value of the next state at the moment they are triggered. The combinational next state logic is located in the LIDAR controller module under Appendix A, beginning with the "always @ (current_state…" statement. The variables that the state machine dictates for the storage and manipulation of data are controlled within synchronous logic using non-blocking assignment, which allows for any changed data to be updated in parallel on the positive edge of each clock cycle. The synchronous

individual state logic may be found in the LIDAR controller module under Appendix A, beginning with the "always @ (posedge fpga_CLK, negedge reset)" statement.

Upon startup, the control system will first await user input from the Zedboard's push button BTNU. When the button is pushed operations will begin, the device then proceeds through a single branch of states to acquire and store the distance scale value from the LeddarVu module's information and constants bank. With the scale data then stored in logic, the system begins iteration through a looping series of states to sample distance readings. This allows for the distance scale constant to be recorded once at startup, then used for calculation of object ranges upon each acquisition. A diagram of the state machine flow may be found in Figure 49 below.

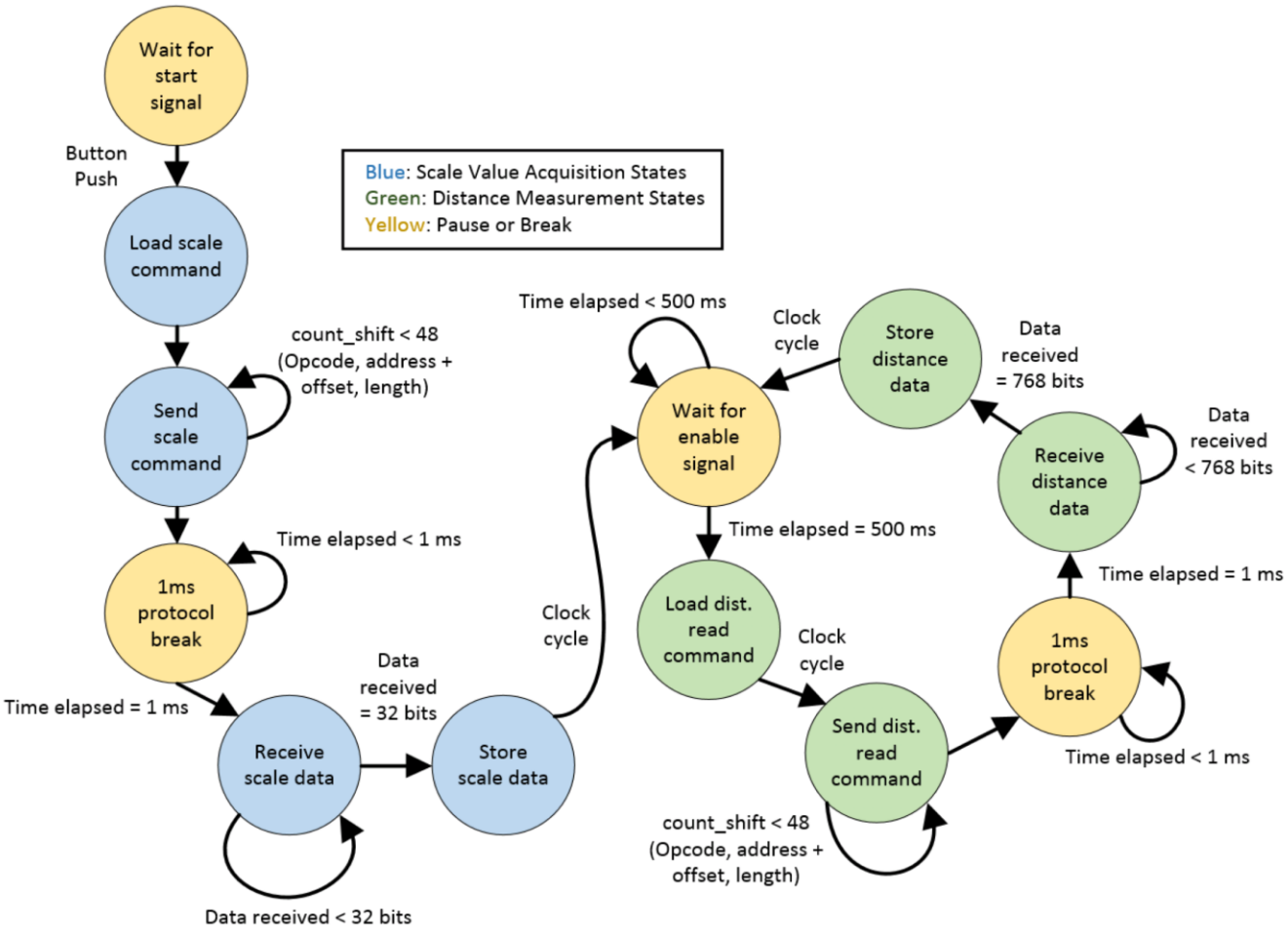

Figure 49: LIDAR Control State Machine Flow. This diagram illustrates the finite state machine (FSM) used to implement the LIDAR Controller logic, which meets the SPI timing constraints, and hence retrieves raw readings from the LeddarVu8.

The state machine begins in state_start, where the device waits for a high signal (1) from push button BTNU. Until this button signal is received, the next state logic will remain set to state_start. When the button is pressed by a user to initiate operation, the device then progresses to state_load_scale. This state lasts one clock cycle, where the synchronous logic prepares the necessary bits for issuing a read command at the distance scale value's address in the LeddarVu8 module's memory. This command contains the read opcode, bank 5 start address with distance scale offset (0x400162), and the 4 byte size. The command is loaded into a buffer, and is then sent out serially in the next state, state_scale_command. State_scale_command shifts the bits in the send buffer out one by one on the MOSI pin at a rate of 1MHz. This is done by waiting for the positive edge of a 1 MHz custom clock (clk_1M_posedge), generated by counting the 100 MHz FPGA clock cycles. In the next state, state_delay_scale, the state machine then utilizes a similarly generated 1 kHz clock counter to allow for the required 1 millisecond break between send and receive for the LeddarVu8's transmission protocol. State_reading_scale is where the distance scale value is read in serially from the LIDAR module. The data on the MISO pin is sampled at each 1MHz positive edge, then shifted left into scale_receive_buffer. After thirty-two 1MHz clock edges are counted, the machine shifts to state_store_scale, where a single clock cycle is used to store the data in scale_received_data, which will ultimately be used for the distance reading calculations.

At this stage, the device will enter the loop of reading distance samples, beginning with state_wait, to await the next signal of a 2 Hz clock generator. 2 Hz was chosen to match the default sample rate of the LeddarVu8 due to the process time of incoming data acquisitions. Upon receiving the enable signal, the state machine then shifts through the process of loading and transmitting the read command, waiting for the 1 ms break, and then receiving the LIDAR distance data. This is done in states state_load, state_read_command, state_delay, state_reading, and state_store reading in similar fashion to the state shifts performed to acquire the distance scale value. In this loop, the sent command requests 96 bytes of data (768 bits) from memory bank 13 at the detection list array offset (0x50000C). This allows for the detection data for all eight segments to be clocked in at 12 bytes per segment. In state_store_reading, the segment distance data is parsed and arranged so that the samples are prepared for division by the distance scale to convert to meters.

Once the data acquisition state machine was created, the implemented protocol was tested to ensure that reading requests were sent properly to the LeddarVu8, and that the module responded with the expected data. This testing was performed by sending a repeated read request for a constant with a known value, and then making sure the data returned by the LIDAR module matched the value expected. The value selected for this test was the LeddarVu8's onboard FPGA version number, which exists as a set of ASCII characters in the device's memory as "2572". The FPGA version is located in the Device Information and Constants memory bank (Bank 5) at offset 288 (Address 0x400120). This request was sent to the LIDAR module over the defined SPI protocol, and the returned reading was analyzed using an oscilloscope on the clock, chip select, and MISO lines. The results were captured from the oscilloscope for each individual ASCII character, as seen in the following figures:

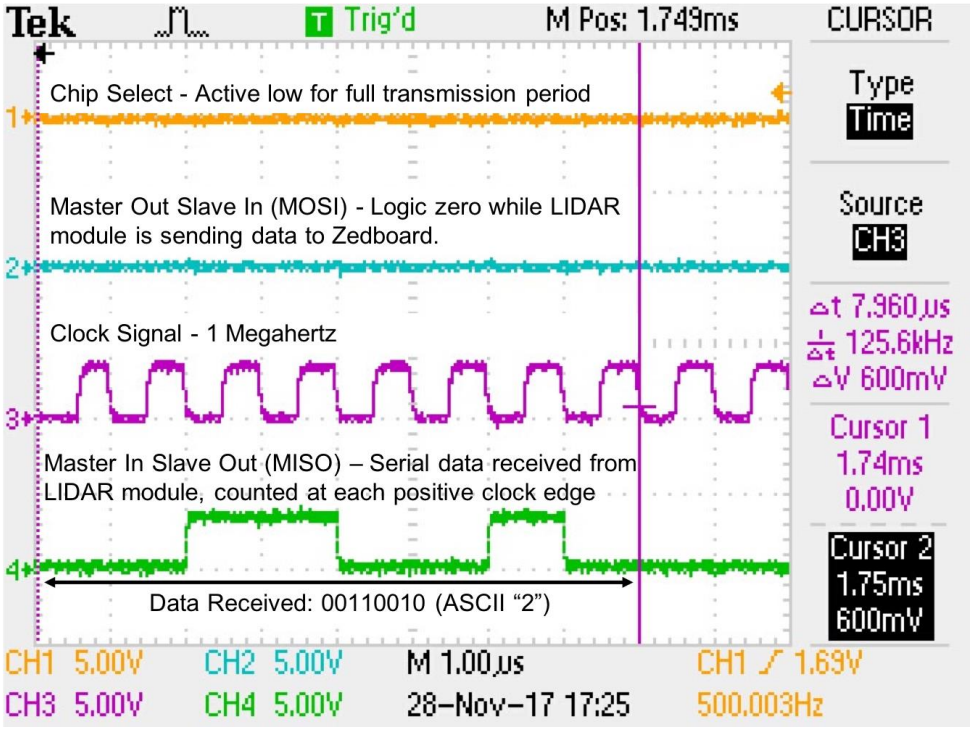

Figure 50: ASCII "2" From LeddarVu8 FPGA Version. The Zedboard receives the first ASCII character of the "2572" version number stored in the LIDAR module via SPI transmission.

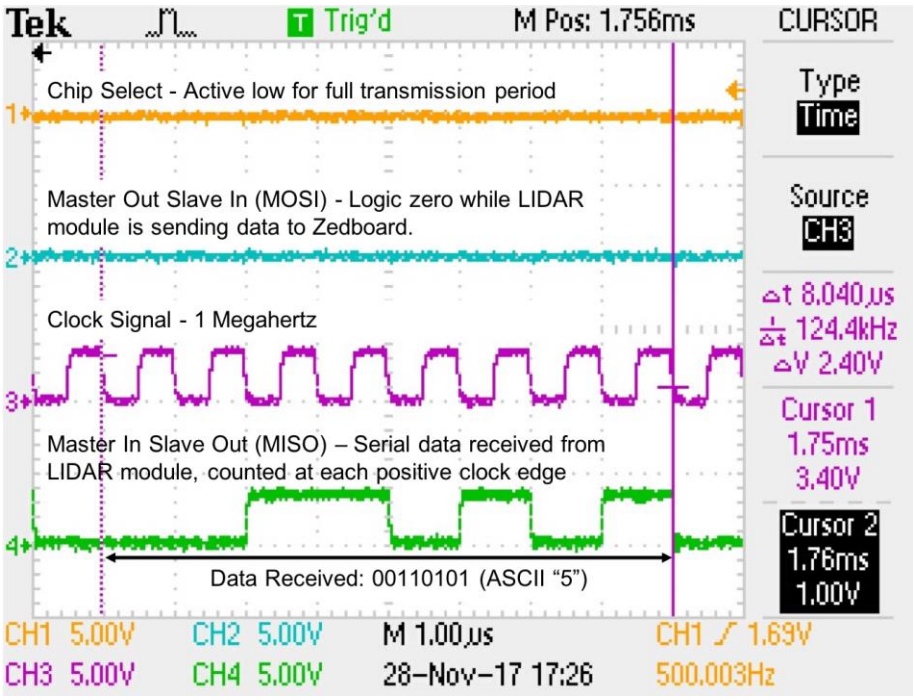

Figure 51: ASCII "5" From LeddarVu8 FPGA Version. The Zedboard receives the second ASCII character of the "2572" version number stored in the LIDAR module via SPI transmission.

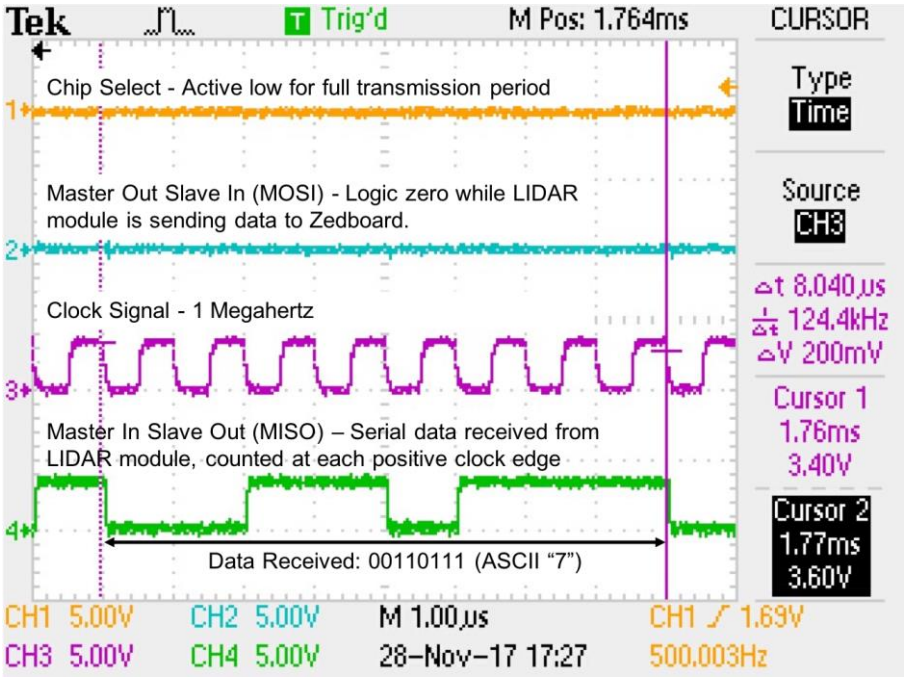

Figure 52: ASCII "7" From LeddarVu8 FPGA Version. The Zedboard receives the third ASCII character of the "2572" version number stored in the LIDAR module via SPI transmission.

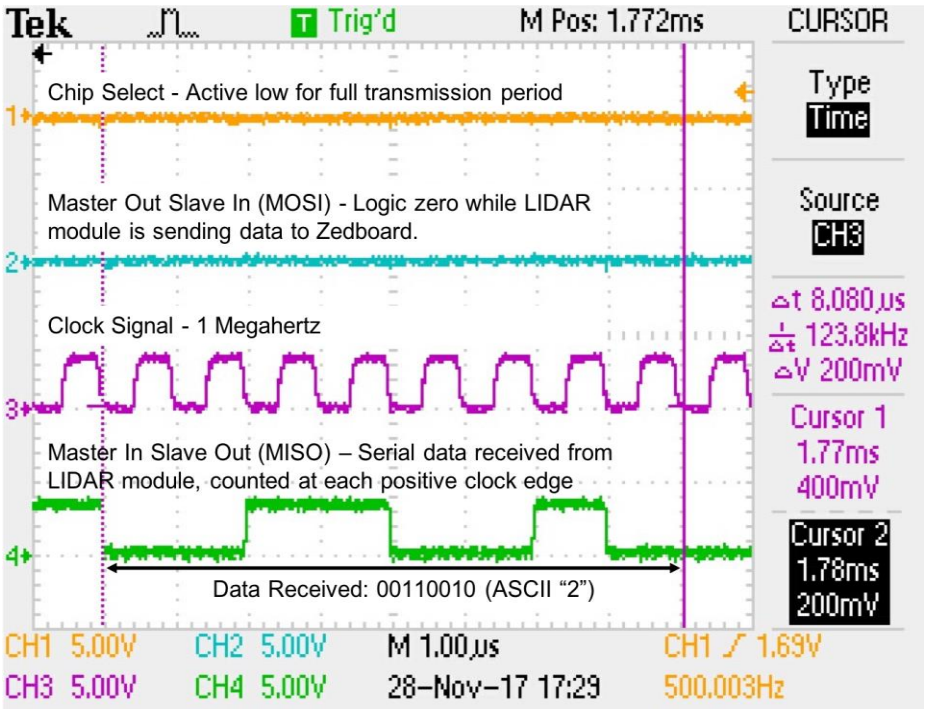

Figure 53: ASCII "2" From LeddarVu8 FPGA Version. The Zedboard receives the fourth ASCII character of the "2572" version number stored in the LIDAR module via SPI transmission.

The accurate reception and storage of the LeddarVu8 onboard FPGA version number confirmed that the SPI protocol designed for LIDAR data acquisition was successful. The constant value was received in the same format with every SPI transmission performed. With this test completed, the send command was readjusted to receive the scale and distance readings, as defined earlier. The next step in developing the LIDAR module control was to implement a means of computing scaled distance readings.

Due to the complexity of floating point division, the necessary mathematical operations for converting raw distance readings to meters could not be performed efficiently in real-time within the LIDAR control state machine. This limitation drove the decision to use the Zedboard's ARM-based processing system for distance computation, which separates the complex mathematical operations from the time-critical data acquisition performed by the programmable logic. The efforts performed to enable the passing of data between the programmable logic and the processing system involved a separate protocol, to be explained in detail shortly. Overall, the use of programmable logic via Verilog HDL successfully created a robust state machine to control LIDAR data acquisition.

With all data transmission to and from the LIDAR sensor controlled by the programmable logic (PL) state machine, it was necessary to implement a means of computing the sampled distances from the raw data acquired. The division of the stored raw binary data by the distance scale constant required floating point arithmetic to ensure that fractions of a meter were represented by decimal values. Such mathematical operations were performed using a C program on the embedded ARM core; however, it was necessary to establish a communication protocol that the ARM-based Processing System (PS) and the Programmable Logic could share. This full computation system was designed by instantiating the ZYNQ computation block IP available in Xilinx Vivado, as well as customizing Vivado's base Advanced Extensible Interface (AXI) protocol IP. With this in mind, it was possible to develop a second part of the detailed hardware diagram pertaining to LIDAR implementation, as shown in Figure 54 below.

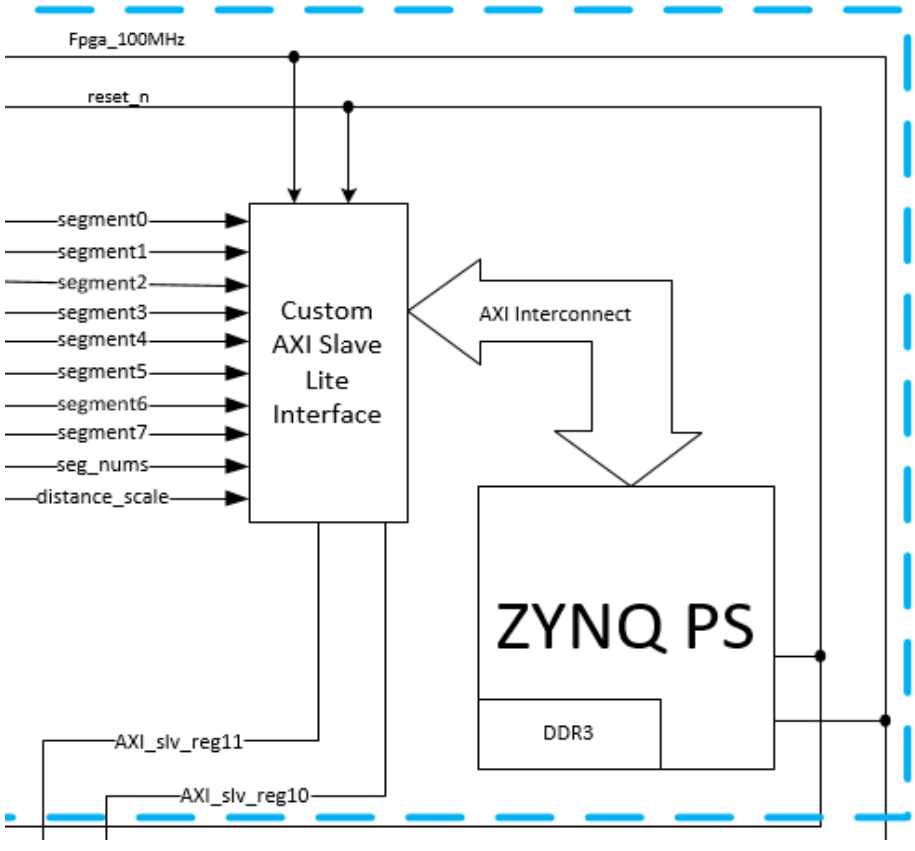

Figure 54: Block Diagram of ZYNQ PS with Custom AXI Interface. The raw distance readings from LIDAR controller are passed to the ZYNQ PS via Custom AXI Slave Lite Interface. ZYNQ PS computes the actual distances in meters, compares them to the predefined zones and returns the results via AXI_slv_reg10 and AXI_slv_reg11 to the Data Fusion logic module.

The following explanations will outline the process completed to utilize these modules and pass data over AXI communication.

In order to make use of Zedboard's ARM processors, it was necessary to create a ZYNQ Base System. Note that this approach was inspired by "Creating a Base System for the Zynq in Vivado" [108]. This was accomplished in Vivado by creating a new RTL project, targeting the "Zedboard Zynq Evaluation and Development Kit" board.

The first step was to add the Zynq Processor System (PS) and establish the minimal required connections. This was achieved by creating a new Block Design and inserting the "ZYNQ7 Processing System" from the IP catalog, as shown in Figure 55.

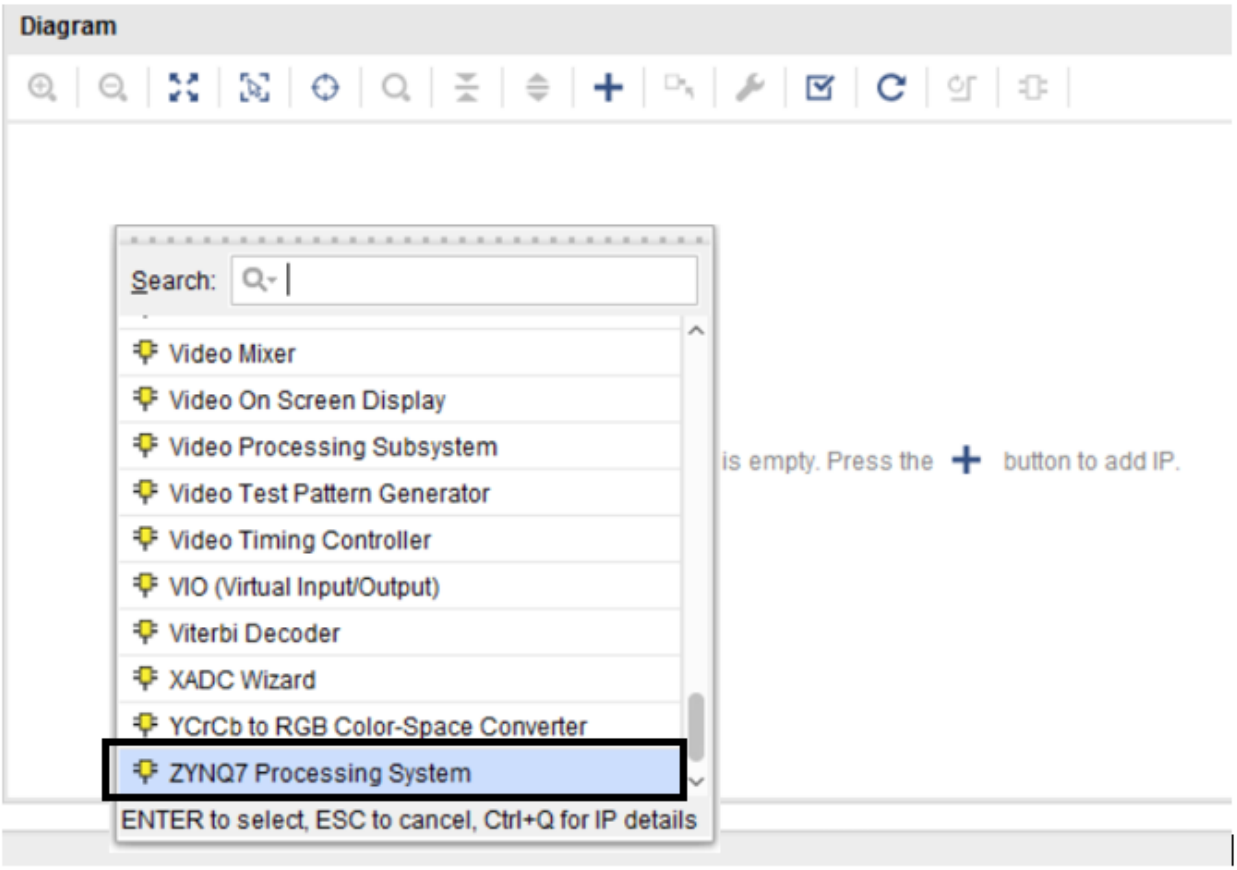

Figure 55: Illustrates how to add a "ZYNQ7 Processing System" IP to a Block Design in Vivado 2017.3 using the existing IP manager. "ZYNQ7 PS" is used to execute C code on ARM cores in order to convert raw distance readings to meters and compare the distance in meters to predefined zones such as Critical Action zone.

After successfully adding the IP, it was then possible to select “Run Block Automation”. This built in Vivado tool was used to ensure proper connection and pin assignment between the PS instantiation and external hardware, such as DDR and fixed IO as shown in Figure 56.

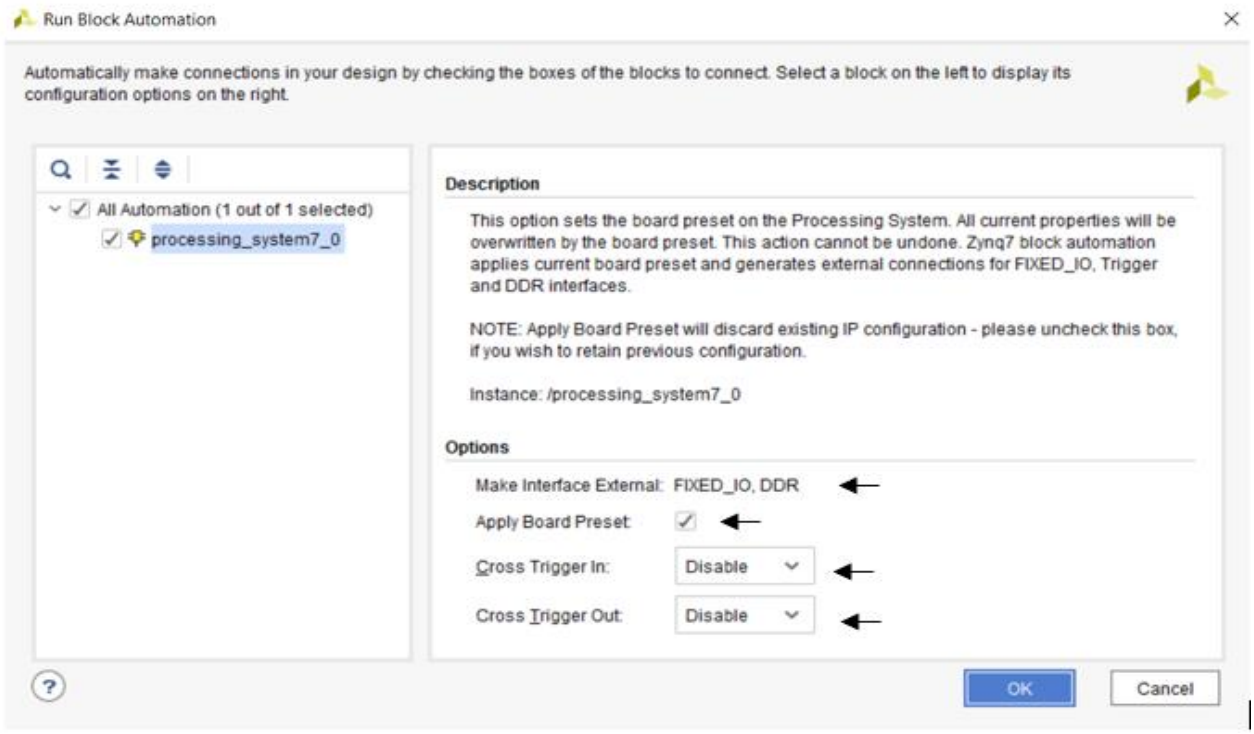

Figure 56: Run Block Automation Settings for ZYNQ7 PS. This is used to set up a basic version of ZYNQ7 (PS) which introduces external connections to DDR and FIXED_IO. With this it is possible to execute C code on one of the ARM cores of ZYNQ7 PS.

At this point, the block design was adjusted and the instance of the ZYNQ7 PS appeared with DDR and Fixed IO connected externally. It was then necessary to make a connection between the PL clock and the AXI General Purpose Master bus. ZYNQ7 PS was customized to enable the default 100MHz PL clock as shown in Figure 57 below.

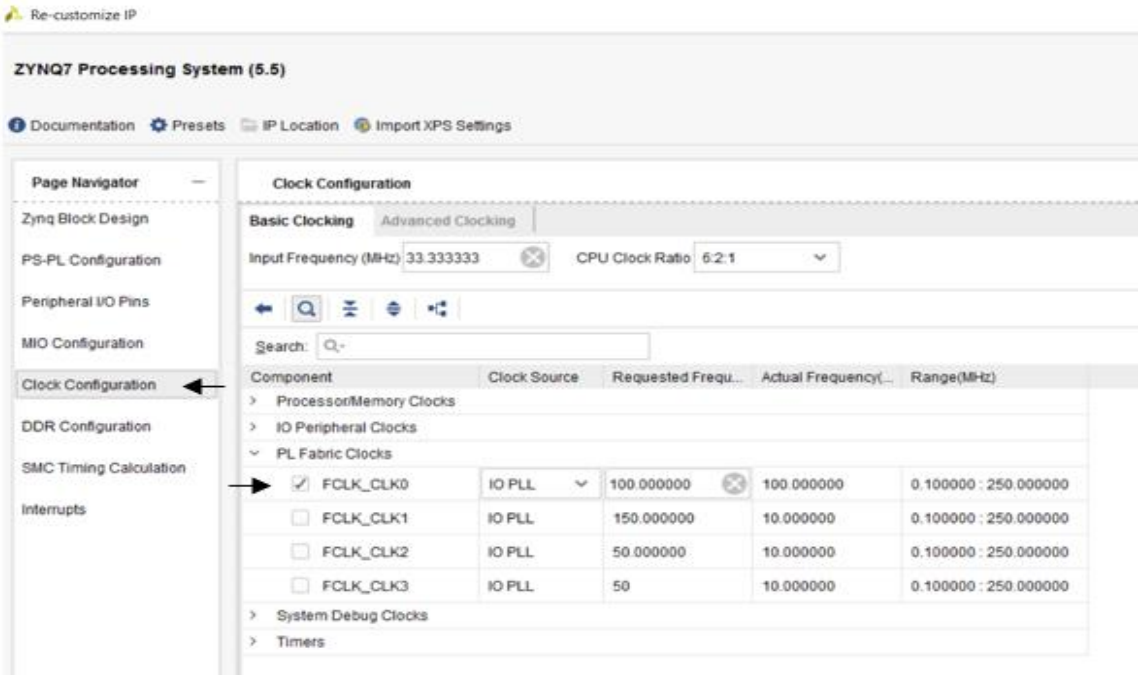

Figure 57: Re-Customize ZYNQ7 PS to Enable PL Fabric Clock. This clock is used to provide timing for all of the programmable logic (PL) including AXI interfaces.

Similarly, using the PS-PL Configuration, it was possible to enable General Purpose AXI Master interface 0 as shown in Figure 58.

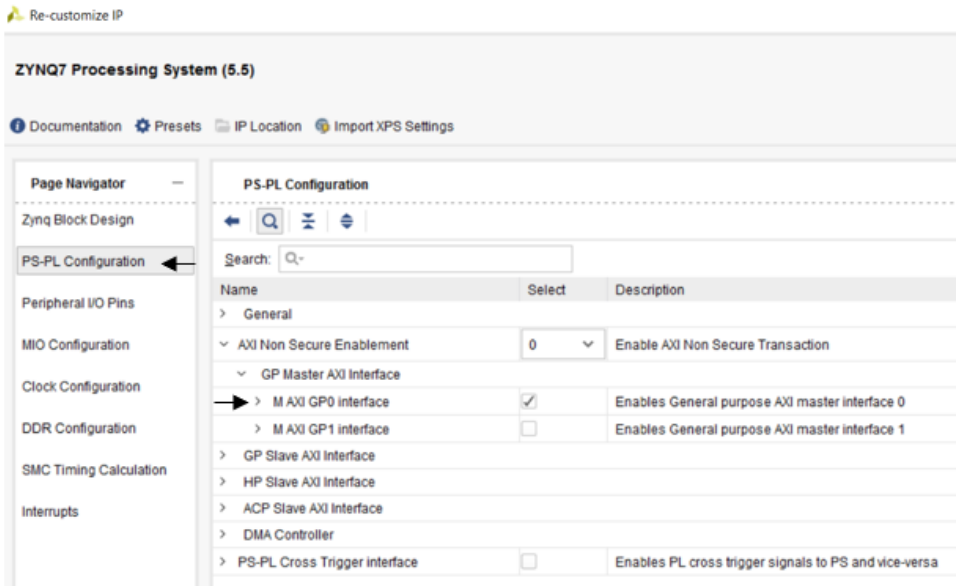

Figure 58: Enable Master AXI GP0 Interface. This interface is used to take in data from AXI Slave peripherals. This interface takes raw distance readings from Custom AXI Slave Lite interface and stores them in specified locations in DDR so that they can be accessed using ZYNQ7 PS via C code.

Finally, the 100 MHz PL fabric clock was connected to drive the General Purpose AXI master interface and the block design changed as shown in Figure 59.

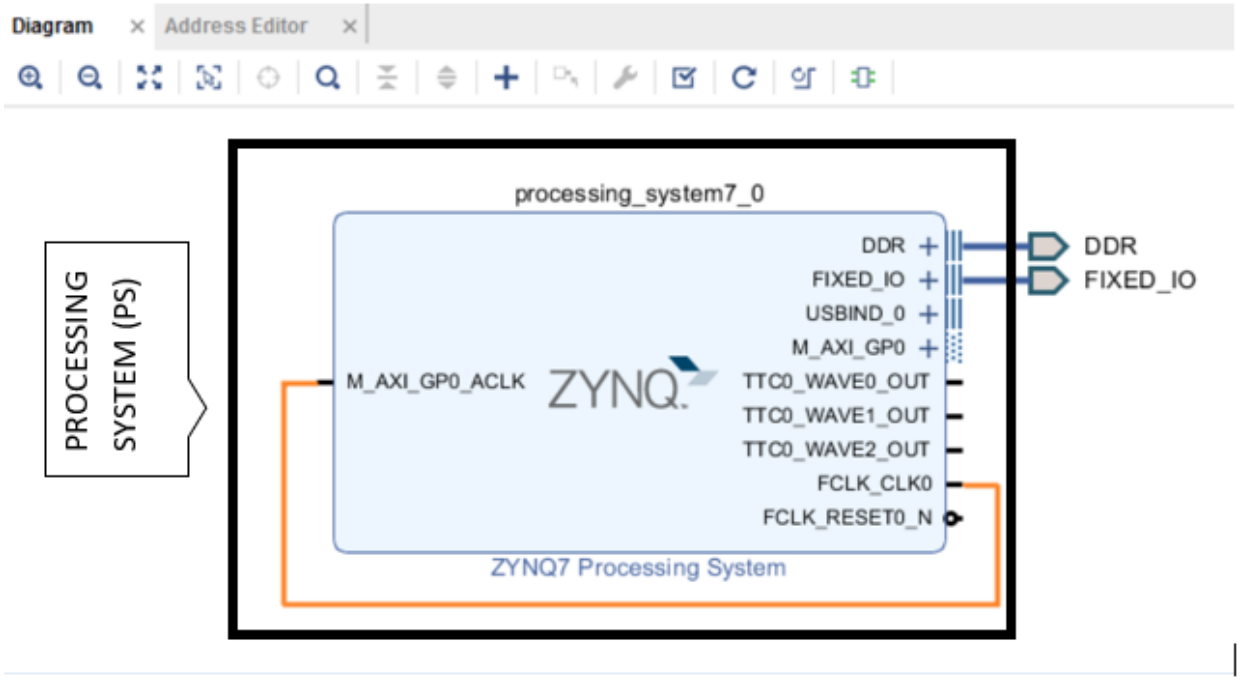

Figure 59: ZYNQ7 PS Block Design. ZYNQ7 PS consists of 2 ARM cores which are used to execute C - code. C code reads raw distance readings from DDR. The readings are converted to meters, compared against the distances of predefined zones such as Critical Action zone.

This point marked the successful instantiation of the Processing System. With the ZYNQ7 Processing System in place, the next stage of the project involved the integration of the existing Verilog HDL LIDAR controller into the block design from Figure 59.

A custom AXI IP block in Vivado was created which integrated the custom PL LIDAR controller. Just as with the instantiation of the Processing system, this approach was inspired by "Creating a custom IP block in Vivado" [109]. The first step was to create a new AXI4 peripheral using "Create and Package IP…" from Tools menu in Vivado. When adding the AXI4 interface the Custom AXI IP was configured as AXI Lite type. The interface was set up as a Slave to the ZYNQ7 PS with sixteen 32-bit registers as shown in Figure 60. These registers were used to pass information between the ZYNQ7 processing system and the custom programmable

logic. Sixteen registers were chosen to prepare an ample number of lines for the transmission of all eight segments' distance data, the distance scale constant, and the zone detection signals.

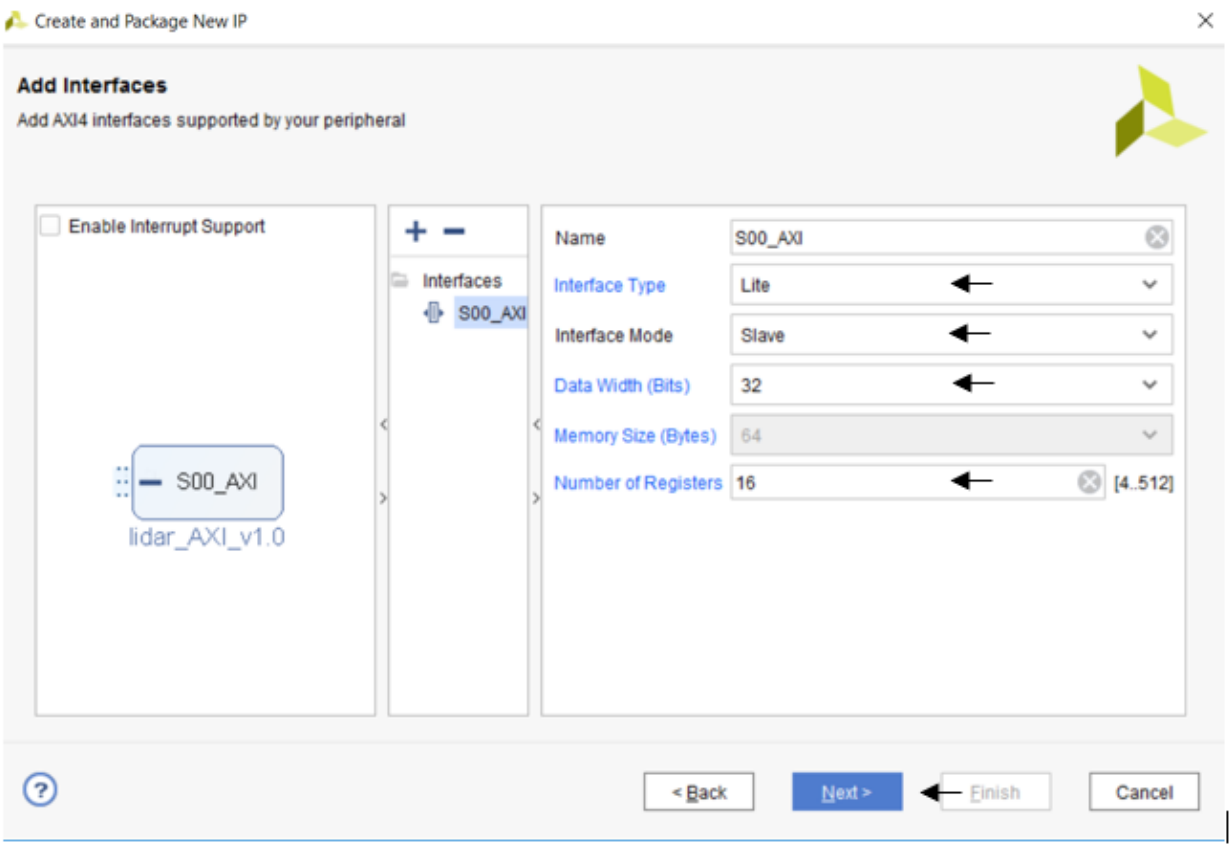

Figure 60: Add AXI Slave Lite Interface with 16 32-bit registers. Defines AXI Custom interface to be a slave to the ZYNQ7 PS with 16 32-bit registers which are used to read/write data to DDR.

It was important to choose "Edit IP" as a "Next Steps" option in order to modify the automatically generated AXI peripheral. It was then possible to edit the AXI Lite Slave peripheral to customize the sixteen I/O slave registers by connecting the inputs and outputs of a PL top module containing instantiations of the device hardware control modules.

In order to do this, it was necessary to first create a top module for the PL LIDAR controller code, which is shown in Appendix A as pl_lidar_top_module. The pl_lidar_top_module instantiates the custom lidar_controller module which is also shown in Appendix A. Notice that the lidar_controller module prepares raw detections of 8 segments each 35 bits wide where bits [2:0] contain the unsigned 3-bit segment number and bits [34:3] store the

raw distance reading as an unsigned 32-bit integer. For example, segment_0 stores one raw distance reading in segment_0[34:3] and the corresponding segment number is stored in segment_0[2:0]. It is also important to notice that lidar_controller also prepares a 32-bit unsigned integer distance scale which is used to convert the raw distance reading to a measured distance in meters. Therefore, pl_lidar_top_module parses and prepares the distance_scale, segment distance readings and segment numbers as 32 bit outputs to be used as inputs to 32-bit AXI slave registers. For example, the module outputs one of the raw 32-bit distance readings (segment_0[34:3]) via segment0 output and the corresponding segment number via seg_nums[2:0] output. Since pl_lidar_top_module prepared all of the data needed to perform the conversion to meters, the next step involved integrating this custom logic with the Vivado generated AXI Slave interface.

In order to add the custom PL logic to AXI Lite Slave interface, the automatically generated lidar_axi_v1_0_S00_AXI module had to be modified as shown in Appendix B. The first change was to add the port connections under "/Users to add ports here/" comment. As mentioned previously, the data was to be passed from pl_lidar_top_module as 32 bit inputs, including the 8 raw distance readings, a distance scale, and segment numbers. The second change to lidar_axi_v1_0_S00_AXI was to instantiate the pl_lidar_top_module and make the connections as shown under "// Add user logic here". The last set of changes involved ensuring that inputs from the pl_lidar_top_module were stored in the appropriate registers as shown under "// Address decoding for reading registers". Note that only the first 10 registers out of 16 (slv_reg0 through slv_reg9) were used to implement the passing of LIDAR data from the PL to the PS. Two additional registers, slv_reg10 and slv_reg11, were later utilized for passing data from PS to PL when objects are present in either the danger warning or critical action zone. The values passed to these two registers are used for displaying the appropriate warning which is dependent on detected object positioning. Details of the warning display logic will be discussed in later sections.

Finally, the topmost module, lidar_axi_v1_0 of the Custom AXI Slave Lite interface, had to be modified to add the user ports for the custom PL logic as shown in Appendix B. This brought the user ports to the highest module in the block, where they then could be set external to connect to hardware pins. After all of these changes were made, the custom IP was repackaged via "Package IP" wizard and it was time to add it to the block design with ZYNQ7 PS.

After successfully creating the custom lidar_axi_v1_0 IP, the next step was to add it to the existing block design via “Add IP”. Before running “Block Automation” to establish connections between the customized blocks, it was important to specify the external ports so that Vivado is aware of the ports connecting to physical I/O on the Zedboard. The final block design is shown in Figure 61 below.

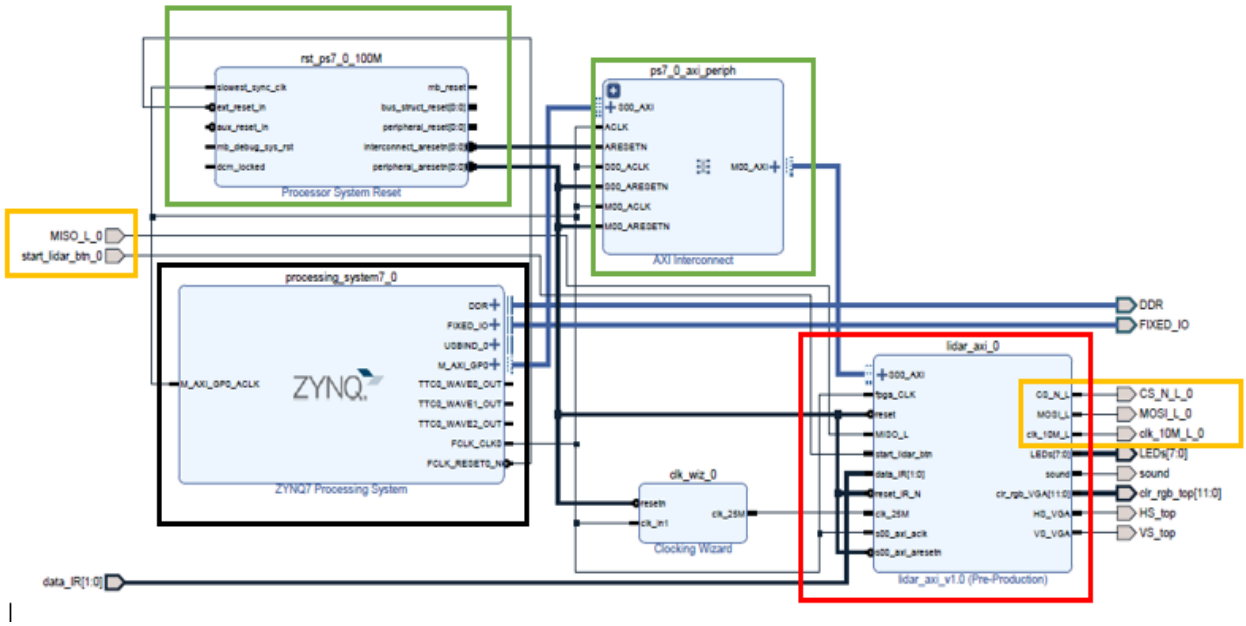

Figure 61: Final Block Diagram as seen in Vivado. Outlined are ZYNQ7 PS (black), Custom AXI interface (red), Custom AXI’s LIDAR external inputs and output (orange), AXI Interconnect and PS Reset (green).

As shown in Figure 61, the customized AXI IP is highlighted by the red box and the external ports associated with the lidar_controller module are outlined by orange. The ZYNQ7 PS block is outlined in black. The connections that were made automatically via Vivado’s Block Automation are boxed in green. The Block Automation ensured that the Master AXI General Purpose interface of ZYNQ7 was connected appropriately with the Custom AXI Slave block via the AXI Interconnect block. From Figure 61, it can also be seen that there are other ports which are related to the passive infrared sensor, VGA display control and buzzer logic. There is also a Mixed Mode Clock Manager (MMCM) IP to control clocking for the VGA display timing. The

infrared sensor, screen, and buzzer logic found within these blocks will be discussed in later sections.

After all of the connections were established, it was important to create the Hardware Description Language (HDL) wrapper by right-clicking on the block design within Sources as shown in Figure 62. The HDL wrapper ensures that the entire design is factored into the synthesized and exported hardware data.

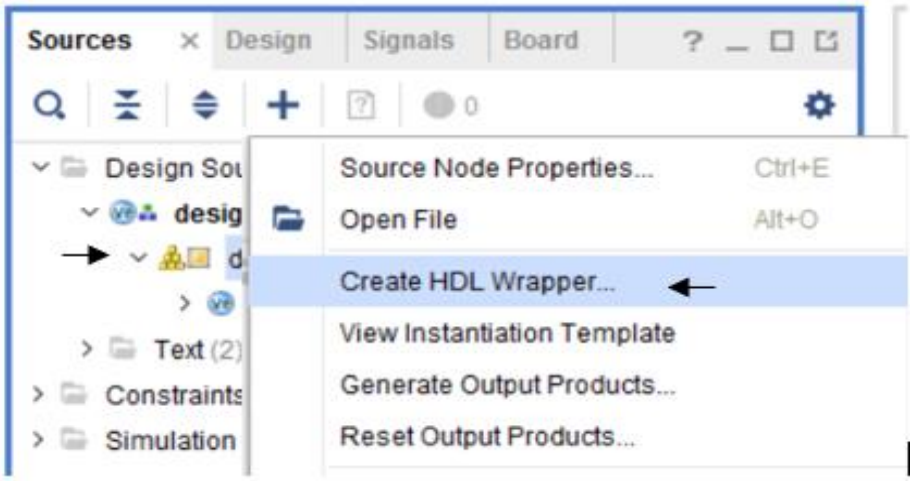

Figure 62: Create HDL Wrapper. This creates a Verilog file that instantiates the top level hardware logic with its appropriate connections. This file is used to synthesize, implement and generate a bitstream of the design.

After creation of the HDL wrapper, the bitstream was generated. The hardware design and bitstream could then be exported for programming the PL to the FPGA fabric of the Zedboard device, as shown in Figure 63.

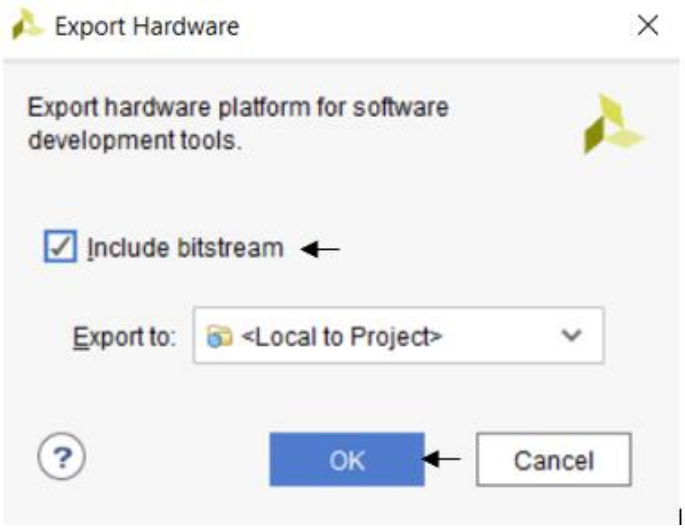

Figure 63: Export Hardware with Bitstream. This bitstream is used to program the FPGA of Avnet Zedboard via Xilinx Software Development Kit (SDK). SDK implements the custom PL design (bistream), with C-code executing on one of the ARM cores of ZYNQ7 PS.

Finally, the Xilinx Software Development Kit could then be launched and used as the coding environment for developing C code. All written C code runs on the ZYNQ PS for the calculation and processing of distance data sampled and acquired by the PL.

The full C code portion of the project is shown in Appendix D. One critical element in this C code is the pointer to the base address of the memory location where transmissions over AXI are performed. Writing and reading to this location are what make the PS-PL communication possible. The following line of code shows the definition of the pointer to this location:

```
//pointer to base address of PS-PL memory location shared over AXI.
Xuint32 *baseaddr_p = (Xuint32 *)XPAR_LIDAR_AXI_0_S00_AXI_BASEADDR;
```

The proper definition of this pointer was created with reference to “Creating a Custom IP Block in Vivado” [109]. An example of this pointer’s use can be seen in the passing of the distance scale value. The distance scale that was read from the LIDAR device was transmitted to the PS from the PL over slv_reg_9. Therefore, to read the distance scale in the C code, the following line of code was used to dereference an offset of 9 from the base address:

```
distance_scale = *(baseaddr_p+9);
```

Similarly, to read the raw distance from segment0 on slv_reg_0, the following line of code was used:

```
distance_seg0 = *(baseaddr_p+0);
```

In order to convert the raw distance readings to meters, the raw data held in each distance_seg variable is divided by the distance scale. The following line was used to perform the division operation for segment zero. The use of floating point arithmetic preserves the decimal value of the distance reading in meters:

```
segMeters[0] = ((float)distance_seg0) / ((float) distance_scale);
```

The variable segMeters is a global array that holds 8 different distance readings from their corresponding segments, once converted to meters.

In order to read all of the active segment numbers, transmitted across in a single register, the following line of code was used:

```
seg_nums = *(baseaddr_p+8);
```

This operation reads in the seg_nums value that exists as an output from the the PL logic, where each individual segment number collected is written to this output in a set of three bits, as described earlier.

Finally, in order to associate the segMeters[0] reading in meters to a corresponding segment number the following line of code was used:

```
distanceSegNums[0] = (int) (seg_nums & SEG0_NUM);
```

Where distanceSegNums is a global array that holds 8 segment numbers, and SEG0_NUM0 is a bitmask used to extract the corresponding segment number from seg_nums. This array is used for printing the segment readings to the console. Note that the remaining 7 out

of 8 distance readings with corresponding segment numbers are read via C code in the same fashion as described above for that of segment zero.

The acquisition and conversion of the distance readings were tested using a serial console output. The LIDAR sensor was placed in a hallway to test the distances of the walls and objects surrounding the system. A screenshot of the serial console output is pictured in Figure 64.

```
Getting reading....
Reading ready
Distance Scale : 0x00010000
Segment Nums Read : 0x00FAC688
Seg7 Distance Reading : 0.583588         Segment: 7
Seg6 Distance Reading : 1.452530         Segment: 6
Seg5 Distance Reading : 1.611374         Segment: 5
Seg4 Distance Reading : 1.837112         Segment: 4
Seg3 Distance Reading : 1.574203         Segment: 3
Seg2 Distance Reading : 1.842300         Segment: 2
Seg1 Distance Reading : 3.787949         Segment: 1
Seg0 Distance Reading : 3.922760         Segment: 0
End of test
```

Figure 64: Console Output from LIDAR Sensor Hallway Test. Segments 6 through 2 are directly facing a wall approximately 1.8 meters away from the sensor's center point. Segments 0 and 1 point towards another open space through a nearby doorway. Segment 7 detects a person standing roughly half of a meter away.

The segment distances printed the console updated in real-time. If an object was placed in segment 7, for example, and then was moved further away, the distance value displayed on the screen would increase to track this change. This test showed that the LIDAR was successful in mapping the surroundings of the system, picking up individual readings in each of the eight field of view segments and converting them to meters using the distance scale value.

Once the C code successfully completes the conversion of all segment distances to meters, it then writes corresponding flag values for each segment to the memory locations associated with AXI registers slv_reg10 and slv_reg11. This allows for the C code running in the PS to provide a signal to the PL logic when an object is detected within either the Critical Action zone or the Danger Warning zone. For example, if an object is found to be within the critical action zone, a flag is written to offset 11 of the AXI base address with a logic OR operation. The n'th bit, starting with bit zero, corresponds to the n'th segment. The line of code that performs this write is as follows:

```
*(baseaddr_p+11) |= (flagToWrite << segNum);
```

When the PL side running on the FPGA detects that a flag value of 1 has been written to AXI slv_reg11, the logic is designed to interpret this as an object present in the Critical Action zone, and can process this data to provide a warning output when appropriate. More details on the warning logic will be described in later sections.

This concludes the implementation of both the hardware logic controller for LIDAR data acquisition, as well as the custom PS-PL communication to process distance samples. The designed state machine logic enabled data transfer to and from the LeddarVu8 modules, compliant with its unique SPI protocol. Furthermore, an AXI Stream Lite interface was used to integrate this logic with the ZYNQ7 processing system, which then stored the raw data from the PL logic into DDR memory. Finally, an embedded C code application was designed to process the raw data stored in memory and convert it to meters using floating point operations. These efforts resulted in a design that could produce accurate distance readings from the data acquired by the LIDAR sensor, and set position flags to be integrated with other modules in the system.

7.2: PIR Sensor

The Passive Infrared Sensor (PIR) was the second critical part of the project, which enabled the detection of moving human beings within the LIDAR's field of view. To set up the interfacing for the EKMC Series PIR, the Zedboard's programmable logic was used to write a simple controller for reading from the sensor. As with the LIDAR control design, all custom code modules were designed using Vivado and the Xilinx SDK.

Initially, the IR sensor was to be mounted on top of a servo motor so that it could rotate and effectively scan any of the areas being detected by the LIDAR sensor. With the choice to switch to the PIR, this was no longer needed due to a significant increase in the field of view. Therefore, it was decided to utilize two of the PIR sensors so that a larger area of detection could be covered. The IR sensors were soldered into small protoboards and 22 gauge wire was used to make a connection to the to the PMOD located on the Zedboard. It was important to pull-down the PIR data line to keep the current below 100uA as specified by the data sheet, as well as ensure data line stability [102]. This was accomplished by adding a 51k ohm pull-down resistor

between the data line and the ground, which ensured that the output current was below 100uA at around 65uA. This is shown in a simplified diagram in Figure 65 below.

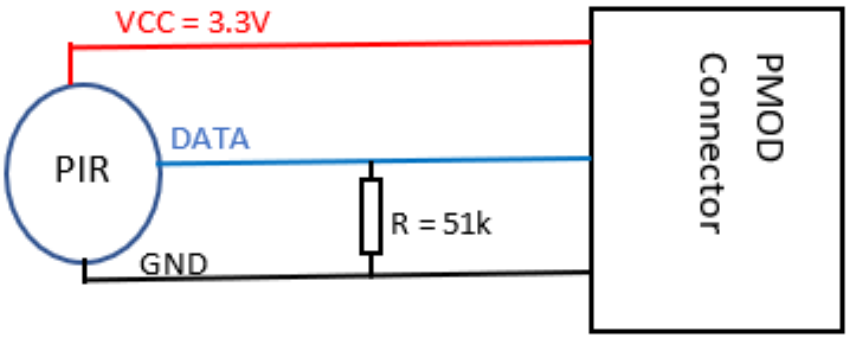

Figure 65: Connecting PIR to Zedboard PMOD. PIR is powered via 3.3V of Zedboard's PMOD connector. The DATA pin of the PIR is pulled-down using a 51kΩ resistor to keep the data current below 100uA which ensures data line stability.

Just as with the LIDAR sensor control, the use of Verilog HDL enabled the creation of a logic-based driver specific to the timing and signals required by the PIR sensor. This turned out to be a much simpler protocol than the SPI protocol implemented for the aforementioned LIDAR. The EKMC PIR has only one data line, which outputs either logic low (target absent) or logic high (target present). Before implementing this data acquisition controller, it was important to understand the sensor's timing diagram. Figure 66 below shows the timing diagram for the EKMC Series PIR used in this design.

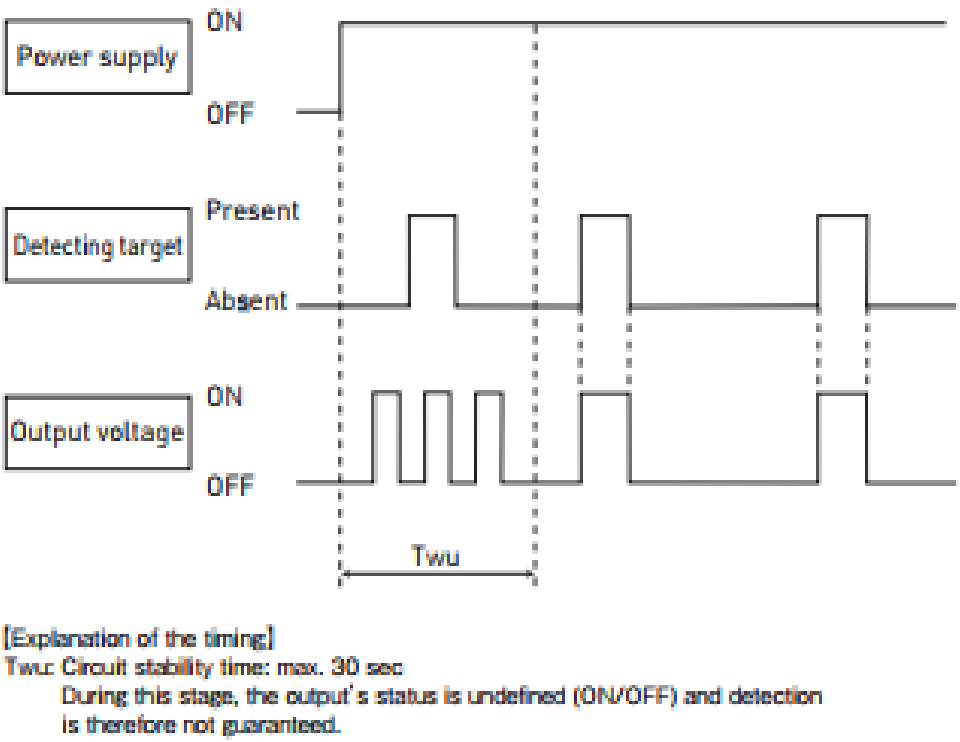

Figure 66: Timing Diagram of EKMC Series PIR illustrates that the stabilization time (Twu) is apparent during the initial power up phase. During Twu (warm-up) stage, the output is undefined and detection is not guaranteed. This was considered when testing. [102]

From Figure 66, one may see that a major consideration for the PIR is that it has a circuit stabilization time, during which the output is undefined and valid detections are not guaranteed. This stabilization time is only apparent during the initial power up phase for the PIR (EKMC PIR). The maximum stabilization time is 30 seconds, which was taken into the account before doing any experiments and tests. This understanding of the timing diagram was critical when designing the PIR controller within programmable logic (PL) and performing test runs.

With full understanding of the PIR sensor's data acquisition requirements, it was possible to develop the IR controller logic block, as shown in Figure 67.

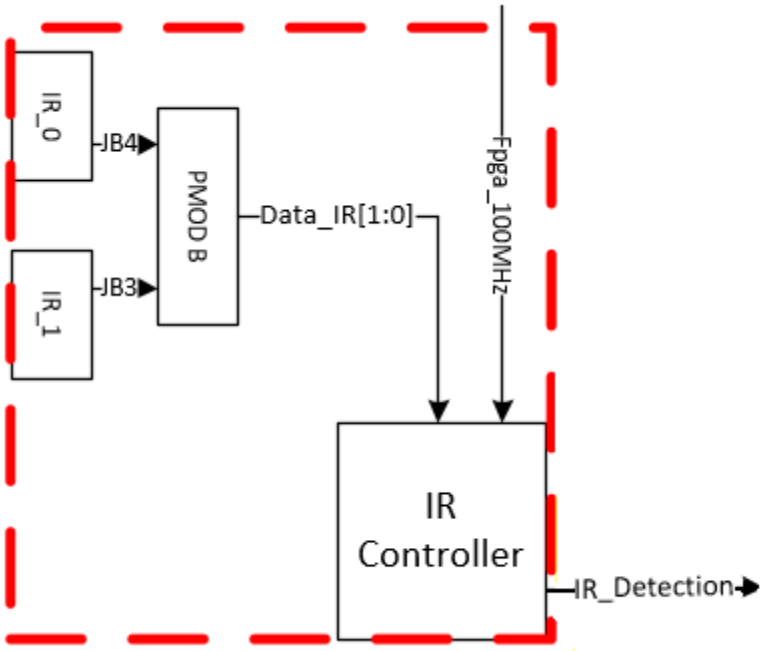

Figure 67: Block Diagram of IR Controller. Two PIR sensors (IR_0 and IR_1) are connected through JB3-4 PMODB ports to the IR Controller. At each rising edge of the 100MHz clock, the IR Controller reads the data pins from the two PIR sensors and outputs the IR_Detection signal, which indicates a valid/invalid detection.

The primary interface for communication between the PIR module and the Zedboard was a data input checked at the 100MHz rate of the PL clock frequency. Due to the use of two PIR sensors in this final design, a valid detection was considered when either one or both of the sensors were triggered. To accomplish this functionality, two PIR inputs were ORed as shown in the IR_top verilog module from Appendix C.

```
      //detection for sound output determined from IR & Lidar
      always @ (posedge clk_100M) begin
              IR_detection <= IR_data[0] | IR_data[1];  //Two IR sensor
signals ored for single detection
          end
```

From the code above one may see that on every rising edge of the PL fabric clock, the IR_detection signal was updated based on the two PIR inputs: IR_data[0] and IR_data[1]. The IR_detection signal was used as a decision flag to signal a valid detection. For example, if a person was in the field of view of either one or both of the sensors, the IR_detection signal

would be set high. Similarly, if the person was not in the field of view of either sensor, or an inanimate object was present, the IR_detection signal would stay low and the detection ignored.

In order to check that both sensors were working properly, they were first tested individually using the same approach, except that the IR_detection was assigned to only one of the IR_data inputs at a time. Then, the IR_detection signal was tied to an LED output to prove the sensor operated properly through a visual cue. If IR_detection was valid, the LED would light up. Once the PIR was proven to be working, the control logic was instantiated in the IR_top module, connecting the IR control to warning display logic, in place of the LED indication used for testing. This allowed for the PIR to be connected to alarm system logic that incorporated a VGA display and buzzer. Based on the output of the PIR, along with the distance acquired by the LIDAR sensor, the color of the screen changes and the buzzer sounds. The logic driving this alert system will be introduced later. The warning system is a method of alerting the driver to show that there is potential danger, and this was the beginning of the data fusion for the PIR into the overall system of pedestrian detection and driver notification.

7.3: Data Fusion between LIDAR and PIR

In sections 7.1 and 7.2, it was shown how the data was obtained from the LIDAR and PIR modules. This section discusses the data fusion between LIDAR and PIR sensors that was used to signal the alarm system. More specifically, this section introduces the logic design that provides control signals for the buzzer and the warning display.

Section 7.1 showed that the Custom AXI module (Appendix B) took raw distance readings from the LIDAR module and stored them in DDR memory. These readings were then accessed from DDR via a C program (Appendix D) running on ZYNQ7 processing system. Using the C program, the raw distance readings were then converted to appropriate floating point readings in meters. The distance readings of each of the 8 segments (segments 0 to 7) were compared against the predefined thresholds for “critical action” and “danger warning” zones, and the results of these comparisons were then stored to the DDR memory via slvreg11[7:0] and slvreg10[7:0], respectively. For example, if the LeddarVu8 segments 0, 1 and 5 measured distances below “critical action” threshold then the slvreg11[7:0] would take on a binary value of 0b00100011 . Similarly, if the segments 1, 3, and 4 measured distances below the “danger warning” zone then the slvreg10[7:0] would take on a value of 0b00011010. Therefore, if all of

the distance readings for all 8 segments were beyond both range thresholds, then slvreg10[7:0] and slvreg11[11:0] registers would take values of 0b00000000. Hence, if the value of either one of the registers was greater than 0, it would indicate that there was at least one segment that located an object within a threshold range.

Section 7.2. showed that a valid PIR detection was defined by an ORed signal between the two PIR sensors. This signal was labeled as IR_detection in IR_top module (Appendix C) and it is an active high signal. For example, if either one or both of the PIR sensors detect a moving human body, the IR_detection signal is pulled high.

With the full understanding of the LIDAR signals passed from the Custom AXI module, as well as the IR_detection signal passed from the IR controller module, it was possible to develop the data fusion logic block, as shown in Figure 68 below.

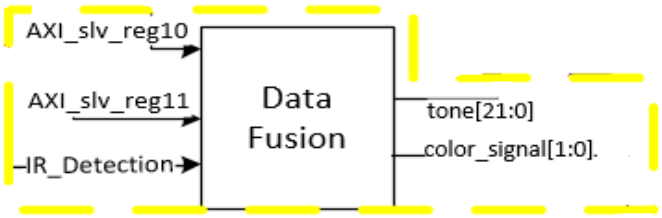

Figure 68: Block Diagram of Data Fusion. Data Fusion module takes results from LIDAR and PIR logic and fuses them to generate control signals for the Alarm logic. Tone and color_signal output signals are used to set the tone and color for the Audible and Visual Alarm logic, respectively.

The signals from Sections 7.1 and 7.2 were put into a combinational logic design within the IR_top module (Appendix C) as shown in the code below.

```
always @ (AXI_slv_reg10, AXI_slv_reg11, IR_detection, tone) begin
        if((AXI_slv_reg11 > 0) && IR_detection) begin   //if object in
critical zone, IR's detect person
            color_signal = 2'b10;   // show red
            tone = 22'd30000; //play sound
        end
        else if (AXI_slv_reg10 > 0) begin       //if object is in danger
warning zone
            color_signal = 2'b01;   //show yellow
            tone = 22'd0000;  //turn of buzzer
        end
        else begin
            color_signal = 2'b00;   //show green
            tone = 22'd0000; //turn off buzzer
        end
    end
```

From the “if” statement written above, one may observe that if there was some object within the “critical zone” (AXI_slv_reg11>0), and the PIR also detected a human (IR_detection), the color_signal is set to choose red for the VGA display and the tone is set to turn on the buzzer. Otherwise, the next statement checks if the object is in “danger warning” zone. If so, the module sets the display color to yellow and turns off the buzzer. Finally, if nothing is within the two zones, color_signal is set to display green on VGA display and buzzer is kept off.

Now that the control signals for VGA color and buzzer tone were shown, it is now possible introduce the usage of these signals to control the VGA driver logic and the tone generator logic as demonstrated in Sections 7.4 and 7.5, respectively.

7.4 VGA Controller and Color Logic

In order to show the visual warnings on a VGA screen, it was necessary to understand the signals involved with controlling the VGA lines, so that a color logic module for setting images could be designed. To reduce design complexities, a VGA controller from a known source was implemented.

The VGA controller module used was originally designed by Ulrich Zoltán from Digilent Inc. [110], as shown in `vga_controller_640_60` from Appendix C, and is used to control the graphics of the VGA display. It is written in VHDL. This module is often referenced and used for academic study at WPI, so it was a design that the team was comfortable with. Note that the reset signal input was modified to be active-low to agree with the active low peripheral reset of ZYNQ7 processing system. The controller contains the logic to generate the synchronization signals, horizontal and vertical pixel counter, and a video disable signal, which ensure proper protocol for displaying images of 640x480 resolution at 60Hz over the VGA port. The `vga_controller_640_60` also provides horizontal and vertical counters for the currently displayed pixel, as well as a blank signal that is active when a pixel is not inside the visible screen and the color outputs should be reset to zero. The block diagram of the VGA controller module can be seen in Figure 69. A 25 MHz clock is used as an input for this module, which is generated using a Mixed-Mode Clock Manager (MMCM) IP block. HS is the horizontal sync pulse outpin pin to the VGA port monitor, while VS is the vertical sync pulse. Hcount is the horizontal count of the display screen, while vcount is the vertical count to the active video on the display screen. Blank is active when pixels are not in the visible area. The blank signal is

delayed one pixel clock period (40ns) from where the pixel leaves the visible screen, according to the counters, to account for pixel pipeline delay. This delay occurs because it takes time from when the counters indicate current pixel should be displayed to when the color data actually arrives at the monitor pins (memory read delays, synchronization delays).

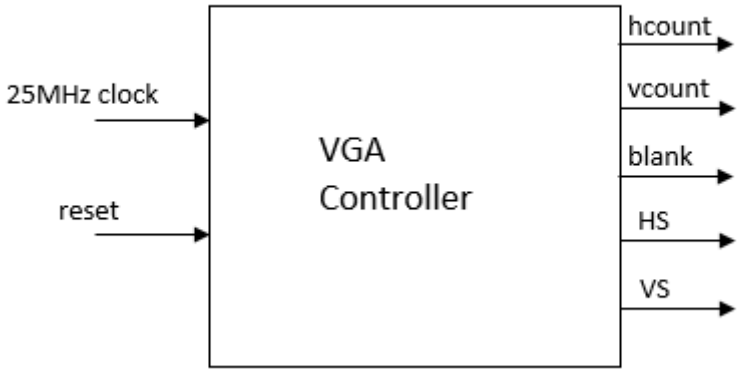

Figure 69: Block Diagram of The VGA Controller. This module takes a 25MHz clock from the MMCM and generates output signals used to drive a display via VGA port. HS and VS are horizontal and vertical sync signals passed straight to the VGA port and they take care of the horizontal and vertical pixel synchronization/timing. hcount, vcount, and blank are signals related to the current pixel position and these are used to control the graphics/color on the display.

The current pixel horizontal (hcount) and vertical (vcount) counts are passed with the blank signal to the color_logic module from Appendix C. The color_logic module from Appendix C takes these signals together with color_signal mentioned in Section 7.3 and decides on the output color by setting the RGB signals (clrg_rgb) on the VGA connector. For example, if the color_signal chosen is red, the RGB signals would get set to red as shown in code from color_logic module from Appendix C:

```
//display a color to fill the screen based on the input from IR_top
always @ (blank, color_signal, clr_rgb) begin
    if (blank == 0) begin
        case(color_signal)
            2'b01 : clr_rgb = YELLOW;
            2'b10 : clr_rgb = RED;
            default: clr_rgb =  GREEN;
        endcase
```

```
    end
    else begin
        clr_rgb = BLACK;    //if blank signal =1, display no color.
    end
end
```

The horizontal sync (HS) and vertical sync (VS) together with RGB signals (clr_rgb) are used to drive the VGA color graphics. The Zedboard allows 12-bit color video output through a standard VGA connector. Therefore, the clr_rgb output is a 12 bit output. The VGA connection pins along with their mapping to Zynq I/O pins are shown in Table 31.

Table 31: Table showing the VGA connector pin numbers and descriptions and their corresponding Zynq pins on the Avnet Zedboard. These are crucial when routing VGA signals from programmable logic to external pins of the VGA connector. [57]

VGA Pin	Signal	Description	Zynq Pin
1	RED	Red video	V20, U20, V19, V18
2	GREEN	Green video	AB22, AA22, AB21, AA21
3	BLUE	Blue video	Y21, Y20, AB20, AB19
4	ID2/RES	Formerly monitor ID bit 2	NC
5	GND	Ground (HSync)	NC
6	RED_RTN	Red return	NC
7	GREEN_RTN	Green return	NC
8	BLUE_RTN	Blue return	NC
9	KEY/PWR	Formerly key	NC
10	GND	Ground (VSync)	NC
11	ID0/RES	Formerly monitor ID bit 0	NC
12	ID1/SDA	Formerly monitor ID bit 1	NC
13	HSync	Horizontal sync	AA19
14	VSync	Vertical sync	Y19
15	ID3/SCL	Formerly monitor ID bit 3	NC

The final block design used to control the VGA color warning graphics is shown in Figure 70.

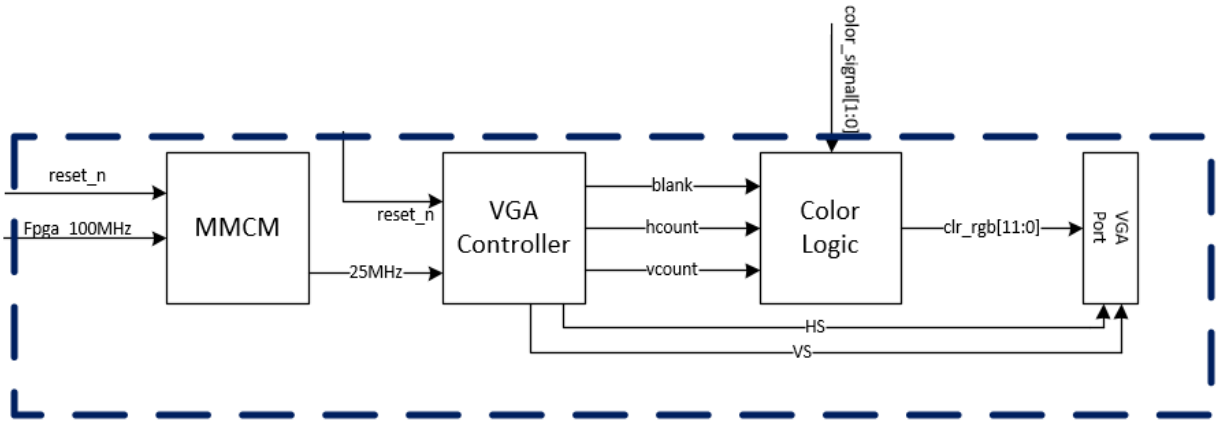

Figure 70: Block Diagram of a Visual Alarm System. The MMCM module takes a 100MHz PL clock and creates a synchronous 25MHz clock used to drive the VGA controller. VGA Controller takes a 25MHz clock and generates output signals that drive a display via VGA port. HS and VS are horizontal and vertical sync signals passed straight to the VGA port. They control the horizontal and vertical pixel synchronization/timing of the VGA display. hcount, vcount, and blank are signals related to the current pixel position and these are used to control the graphics/color on the display via Color Logic module. Color Logic module changes the color based on the control signal from the Data Fusion block, color_signal.

From Figure 70 one can see that the VGA Controller is supplied a 25MHz clock via MMCM. The MMCM takes a 100MHz clock from PL fabric and creates a synchronous 25MHz clock. This method of stepping-down clock rate via MMCM is a common practice for generating clean clock signals from the stock FPGA source clock. The MMCM was instantiated and configured via Vivado's included IP catalog. The VGA controller takes the 25 MHz input clock and generates five output signals that are used for the VGA display protocol: blank, hcount, vcount, hsync and vsync. The signals blank, hcount, and vcount give information about the current pixel position and are passed to the color logic module. The color logic module takes these inputs with the color_signal input and it outputs the appropriate 12-bit RGB value in order to set the graphics on the VGA display. Hsync and vsync are used as synchronous pulse signals to display the RGB value, according to the VGA protocol.

7.5: Tone Generator

For the purpose of sounding an audible alarm warning on a buzzer, it was necessary to develop a tone generator logic module.

A buzzer is used by system to alert the driver in sync with the VGA display colors on the LCD screen. The buzzer used for this system was the 8 ohm model introduced in Section 6, connected to JB1 of PMOD ports of the Zedboard. A current limiting resistor of 51 ohms was connected in series with the buzzer in order to keep the current at around 56mA. This was necessary in order to satisfy the PMOD's power ratings. The final block diagram of this portion of the design is shown in Figure 71.

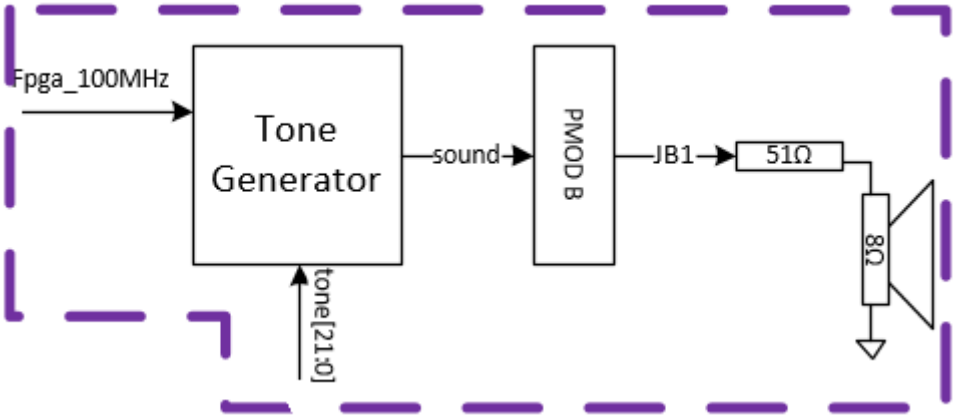

Figure 71: Block Diagram of Audible Alarm. Tone Generator takes a 100MHz clock and generates a sound signal based on the tone signal. tone is a control signal that comes from the Data Fusion module and is used to set the sound frequency. The output sound signal is fed into an 8Ω buzzer through a 51Ω current-limiting resistor via JB1 PMODB port.

Tone generator is a simple module that counts the positive edges of a 100MHz programmable logic (PL) clock to a create a pulse width modulated (PWM) signal for the purpose of driving the 8 ohm buzzer. The tone (counter) is a control variable that is used to specify the logic high duration of the PWM signal. This way it is possible to control the audible frequency of the buzzer, depending on the duty cycle of the output square wave. Recall that the tone (counter) signal was supplied by the data fusion logic as explained in Section 7.3. For example, if the data fusion logic decided that the detected object is potentially a human inside of the Critical Action zone, it would set the tone signal to a specified value, thus sounding an alarm. The detailed code is shown in the `tone_generator` module under Appendix C.

7.6: Final Block Design in Vivado

The final hardware layout block design, as it appeared in Vivado after all of the modules were implemented, can be seen in the Figure 72 below.

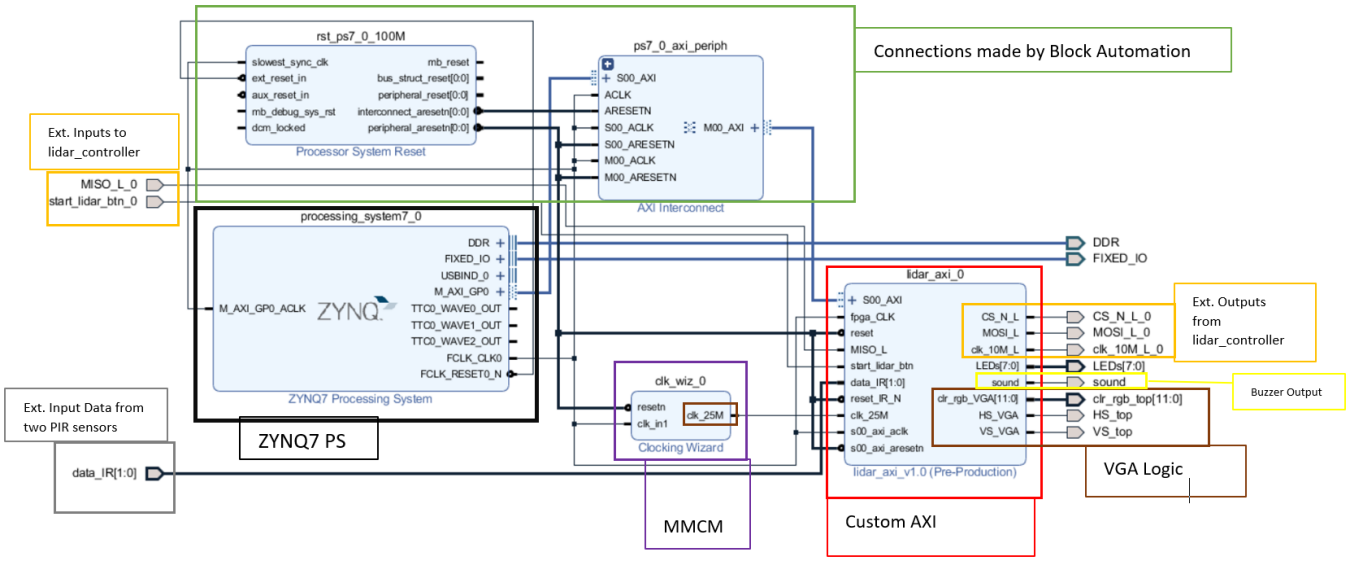

Figure 72: Final Block Design as seen in Vivado 2017.3. Outlined are ZYNQ7 PS (black), Custom AXI interface(red), AXI Interconnect and PS Reset (green), Custom AXI's LIDAR external inputs and output (orange), Custom AXI's PIR external inputs (gray), Custom AXI's Audible Alarm external output (yellow), Custom AXI's Visual Alarm external input and outputs (brown), MMCM (purple). Note that all custom programmable logic modules, such as Lidar Controller, IR Controller, Data Fusion, VGA Controller, Tone Generator, were instantiated within the Custom AXI Interface (red).

As shown in Figure 72, the customized AXI IP is highlighted by the red box, which contains all of the custom logic such as LIDAR controller, IR controller, VGA controller, Tone generator, and Data Fusion. The external ports associated with the LIDAR controller, IR controller, VGA controller and Tone Generator module are outlined by orange, gray, brown and yellow, respectively. The ZYNQ7 PS block is outlined in black. The connections that were made automatically via Vivado's Block Automation are boxed in green. Block Automation ensured that the Master AXI General Purpose interface of ZYNQ7 was connected appropriately with the Custom AXI Slave block via the AXI Interconnect block. There is also a Mixed Mode Clock Manager (MMCM) IP to control clocking for the VGA display timing.

The custom AXI interface establishes a protocol to acquire raw distance readings from the LIDAR using the LIDAR controller module and passes these readings via AXI Interconnect

into DDR memory. The ZYNQ7 PS reads the raw distance readings from DDR, converts the raw readings to meters, compares the distances against predefined zones (“e.g. Critical Action zone”), and stores the results to DDR. The custom AXI IP reads the results from DDR memory and passes them to Data Fusion PL (instantiated within Custom AXI IP). The PIR data is read using the IR controller module, which passes the detection signal to Data Fusion PL. Based on these input signals, the Data Fusion logic decides the color and the tone frequency for the VGA controller and Tone Generator, respectively.

This concludes the methodology related to the hardware and software design of the project. The next section focuses on the power supply implementation to provide energy to all of the physical components.

7.5: Power Supply

The original plan for the power supply was to have two different output voltages, one with 12 volts and the other with 6 volts using a step down converter. Due to design and component changes, the servo motor was no longer needed, which allowed both power supply rails to be used for 12 volt output with no step down converter. In addition to the Zedboard and LIDAR, a new power requirement was determined to account for the added cooling fan and warning screen. With the addition of these two devices, the new power requirement totaled at roughly 80 watts. The power supply still provided at least 20 percent more power than needed. All of the sensors, except the LeddarVu8 LIDAR, were powered by the Zedboard directly, since it supplied the necessary 3.3 volts through the PMOD connectors. Figure 73 represents the wiring diagram for the system implemented with the different voltages accounted for.

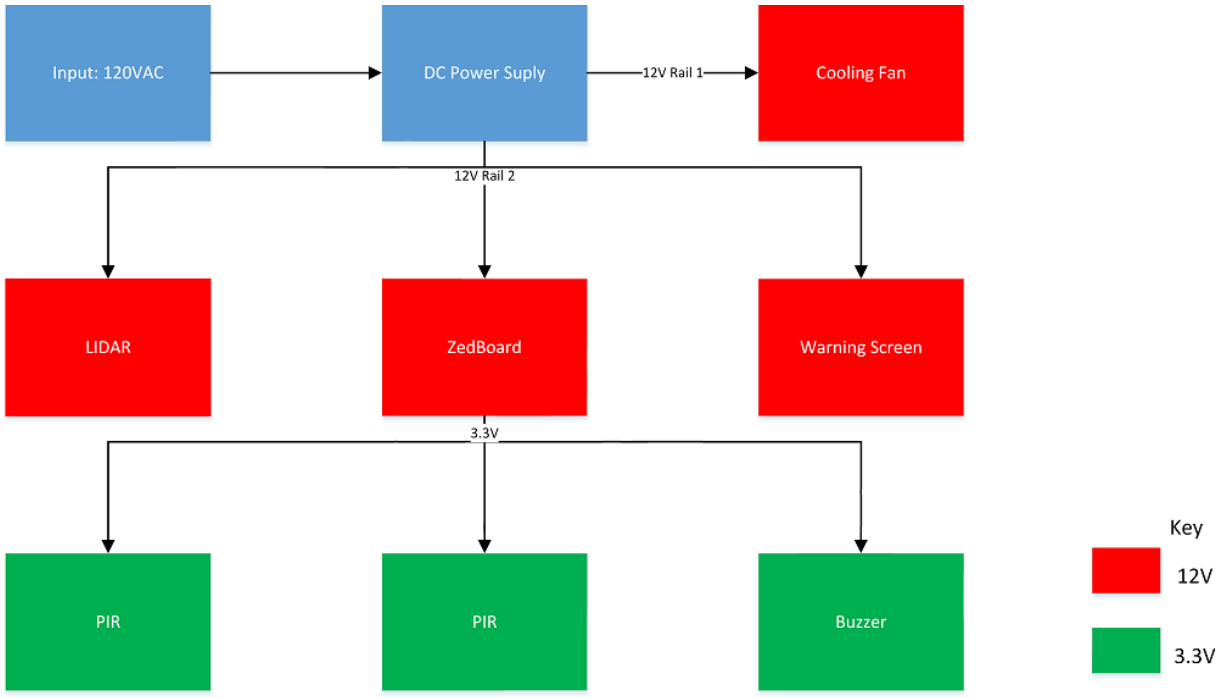

Figure 73: Wiring Diagram for the System Implemented. The red boxes represent devices receiving 12 volts and green represents devices receiving 3.3 volts. 3.3 volts is provided directly from the Zedboard's PMOD ports. The blue boxes represent the external AC power source, and its conversion to DC by the power supply.

The wire gauges used were 16, 18 and 22. To connect the wires to the power supply, spade connectors and solder were used to safely attach them together. The power supply had screws that clamped down on a piece of wire or metal that would be attached to the power supply with terminal connectors, as shown in Figure 74. These were used to make safe contact between the power supply and wire. The wire and spade connectors were crimped onto each other. For additional safety, heat shrink was placed near the connection, which helps in preventing stress on the connector and the possibility of wires becoming exposed and touching each other.

Figure 74: Connections at the Power Supply Terminals. Fork crimp connectors were used to provide a reliable connection. Heat shrink is used to ensure isolation.

To connect the power supply to the wall outlet, an existing 16-gauge wire with a male and female end was used. The female end was cut off and the wire was stripped in order to connect the spade connectors. The power supply was then connected to the wall outlet and it was left alone for 20 minutes while monitored using a volt meter for any problems. The power supply ran smoothly during that time period and there were no heat issues.

The next step was connecting the first output channel that would be used to supply 12 volts to the Zedboard, LIDAR and warning screen. The outputs of the power supply were tested with a multimeter to ensure that they output 12 volts. After both rails gave the expected 12-volt output (shown in Figure 75), an extension cord with three female sockets replaced this setup. The male end of the extension cord was cut off and the wire was stripped down to be connected to the terminal connectors, and then to the power supply. After it was connected to the power supply, a multimeter was used to test the female sockets and they output the expected approximate 12 volts as shown in Figure 76.

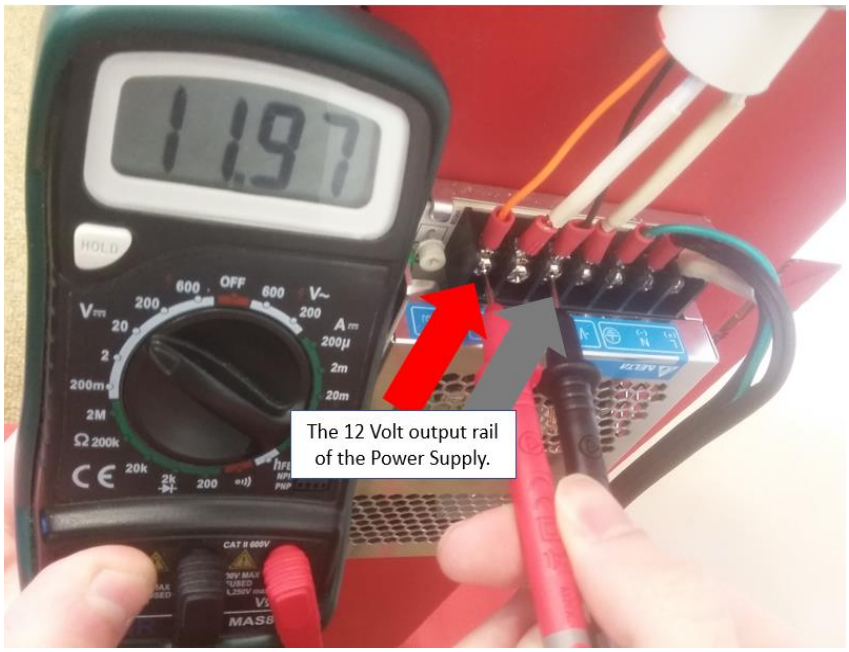

Figure 75: Testing of the 12 Volt Output Rails. Each output rail was tested and output roughly 12 volts. The red multimeter probe was connected to the positive terminal of the power supply and the black probe was connected to the negative terminal of the power supply.

Figure 76: Testing the Female Socket Plugs from the Extension Cord Running from the Power Supply. It provided roughly 12 volts. The red multimeter probe was connected to the positive terminal of the female socket plug and the black probe was connected to the negative terminal.

To connect the LIDAR sensor, Zedboard, and warning screen to the extension cord, additional wire was used along with a two RC jacks (same size) and a two-pin terminal connector (known as a phoenix connector). The warning screen and Zedboard both required an RC jack, while the LIDAR sensor needed the terminal connector. One of the RC jacks had two strands of wire provided, which was used for the Zedboard. For the warning screen, the power adapter, AC to DC converter, and step down converter were cut off from the wire and a two prong male plug was soldered on in its place. Additional heat shrink tubing was placed over the soldered wire for safety. For the LIDAR sensor, a three pin rocker switch was needed to allow the Zedboard to initialize before turning on the LeddarVu8 module. A piece of wire with a male plug and two separate strands of 22 gauge wire were used for the LIDAR sensor. The two wires with the positive polarity were twisted together and crimped on to a connector, while the wires with the negative polarity were connected to the two other pins, as shown in Figure 77.

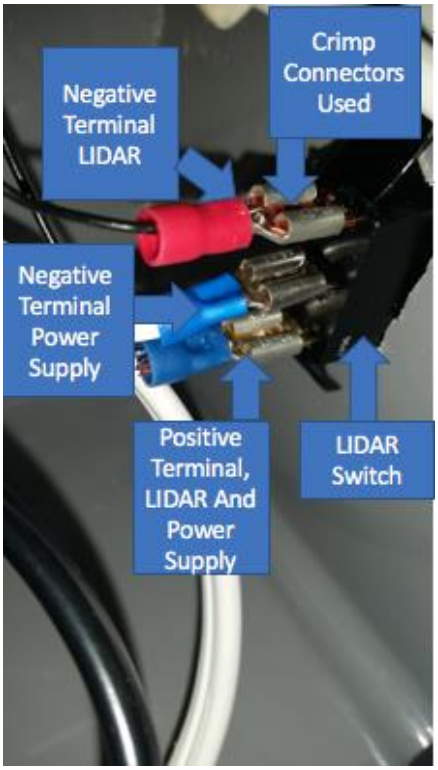

Figure 77: Crimp Connector Used for the Rocker Switch. 22 AWG wires crimped inside the junction box. The positive pin of the switch handled the positive output rail of the power supply and LIDAR while the other two pins handled the negative terminals of the LIDAR and power supply.

After creating the necessary wires for each component, each wire was individually tested to ensure that it output 12 volts. First, the wire for the Zedboard was plugged into the extension cord and a multimeter was attached to the RC jack. As shown in Figure 78, the RC jack output 12 volts for the Zedboard. Initially, a multimeter was attached to the terminal connector going to the LIDAR and it output 12 volts. Further testing was done after the switch was installed as shown in Figure 79. Testing at the switch terminals confirmed that the wire still output 12 volts. Lastly, the wire for the warning screen was tested using the same method as the wire for the Zedboard. As shown in Figure 80, the RC jack output 12 volts for the warning screen.

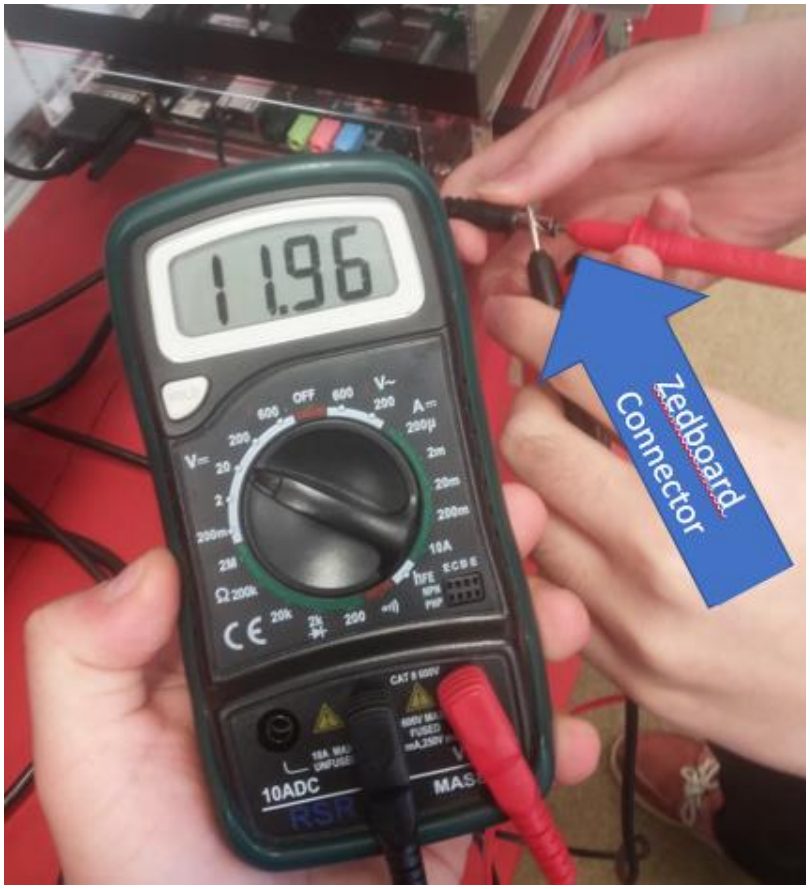

Figure 78: Testing the RC Jack for the Zedboard. The output was roughly 12 volts. The red probe of the mulimeter was inserted into the inner diameter of the RC jack while the black probe was connected to the outer diameter.

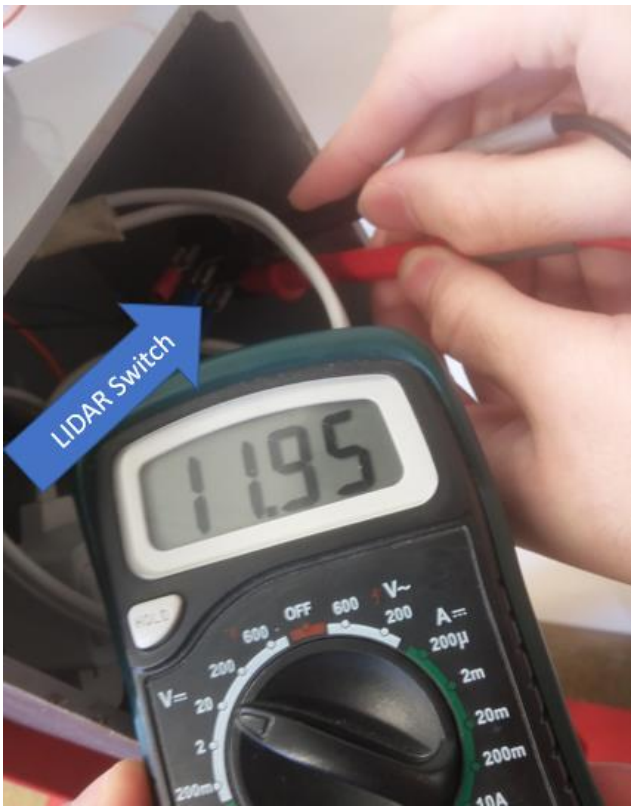

Figure 79: Testing the LIDAR Switch. It outputted roughly 12 volts as expected. The red probe was connected to the positive pin of the switch which handled the positive polarity of the LIDAR and power supply while the black probe was connected to ground.

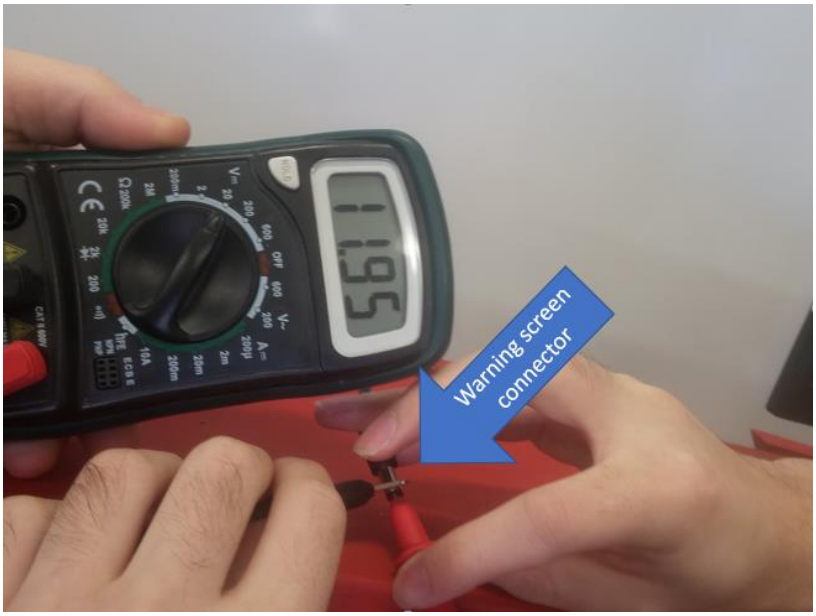

Figure 80: Testing the RC Jack for the Warning Screen. The output was roughly 12 volts. The red probe was connected to the inner diameter while the black probe was connected to the outer diameter of the RC jack.

It was decided that a small extension cord would be used instead of wire nuts because it would be easier to remove or add components whenever needed. Due to this modularity, it was much faster using an extension cord rather than building a circuit on one rail that can safely power the LIDAR, warning screen, and Zedboard. A junction box was used to house all of the connections from each component to the extension cord. This was done to ensure that the connections would be safe from any outside elements. A cutout for the rocker switch was made in the junction box as seen in Figure 81.

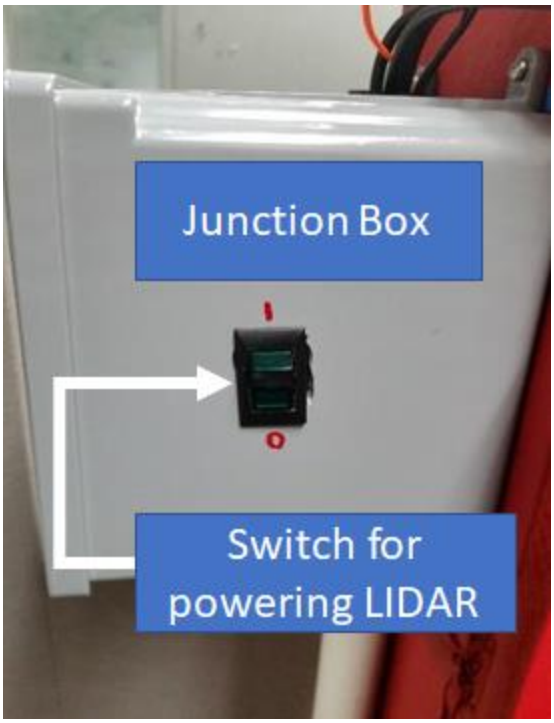

Figure 81: Switch Used For the LIDAR. Provides time to power the LIDAR after the Zedboard is initialized. A cutout for the switch was made using a drill and a filer.

The cooling fan for the system received its own dedicated power rail for simplicity. Two separate strands of wire would run from the power supply to the three pin connector of the cooling fan. The third pin of the fan was not needed as it was the data pin to control when to run the fan and when not to. For the implemented system, the fan would stay on the whole time so this feature was not needed. It was also determined that that would be the simplest approach for powering the fan, as converting the fan's three pin connector into the standard wall outlet plug

was unnecessary. Before connecting the fan, the wires were tested to verify it would receive 12 volts, which was the case, as shown in Figure 82.

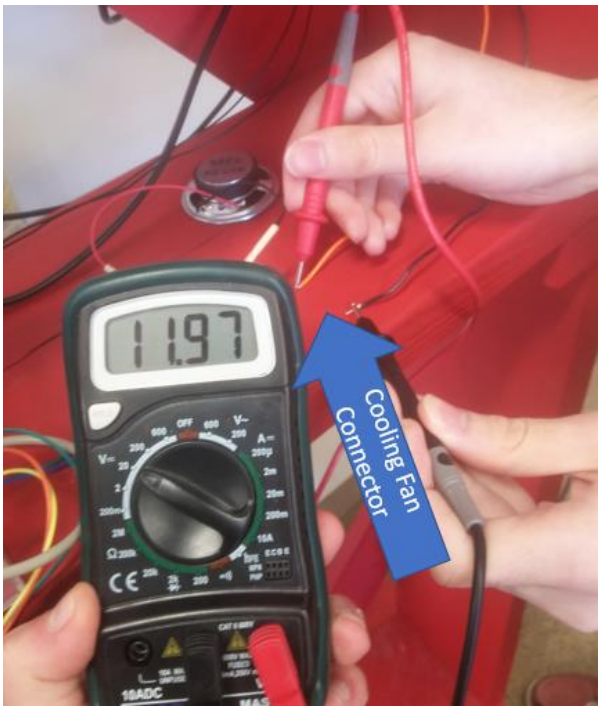

Figure 82: Testing the Wires Supplying Power to the Cooling Fan. Output of roughly 12 volts. The orange wire carried the positive polarity for the fan while the black wire handled the negative polarity.

When testing the output rail for the LIDAR sensor, warning screen, and Zedboard, each device was tested individually to minimize any problems, such as all three devices burning out. First, the Zedboard was connected to the power supply and it ran for 10 minutes without showing any issues. Next, the LIDAR module was tested with a data connection to a laptop, and the Zedboard connection was removed. Again, the LIDAR sensor ran successfully for 10 minutes with no issues. Lastly, the LIDAR was taken out and the warning screen was powered on. This process was repeated multiple times and there were no issues. In all of these cases, the LIDAR and Zedboard were simply plugged in to ensure they powered up, no computations or measurements were taken. After this the LIDAR and Zedboard were powered together with both components performing under load; the LIDAR was actively taking measurement readings and the Zedboard was performing the necessary computations. Lastly, the warning screen was plugged in while the other devices were running and all three components operated successfully.

With one of the power supply rails working correctly, the LIDAR, warning screen and Zedboard were disconnected and the cooling fan was plugged into the other rail. The fan worked with no problem when connected to the power rail. All devices worked as expected.

The power supply and junction box used had existing mounting holes. Six wood screws were used to mount the power components. When placing these two items on the bus frame, it was determined that they should be placed on the side, where the wires are not overly stressed. This was determined based on the wire lengths available used, and the placement of the sensors. Six pilot holes were drilled where the devices would go and then they were drilled in to place. A piece of PVC piping was used to help manage the wires going from the power supply to the junction box, this helped in regards to safety and aesthetics (Figure 83).

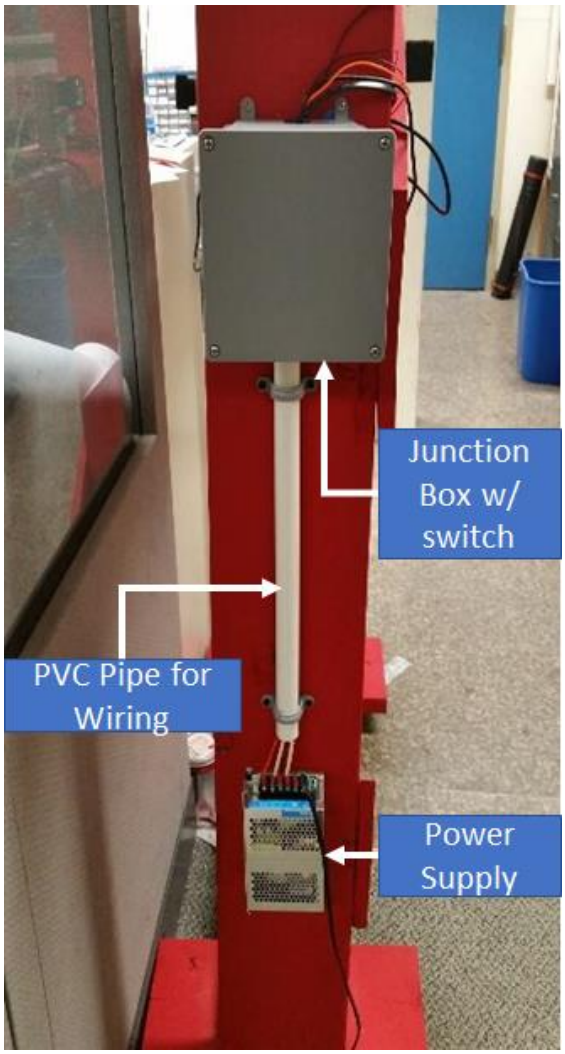

Figure 83: Power Supply, PVC and Junction Box Mounted onto the Frame. PVC was used to help manage the wiring and protect the cables from any outside elements. The junction box was to protect the connections from any outside elements and for safety considerations.

With the power system now fully set up, the casing could be built to house all of the sensitive electronic components.

7.6: Casing

The casing of the detection system houses the Zedboard, LIDAR sensor and the fan. To form a spacious encapsulation to hold these devices securely, a Solidworks model was designed. The model consists of four sides, a bottom, and a top compartment. Based off of Figure 84, each of the four sides of the model consists of a 1” x 5.67”, rectangular cut out to allow accessibility to the development board and allow air flow from the fan. The bottom, shown in Figure 85 below, has four hexagon cutouts that will allow the standoffs on the Zedboard to be placed securely. The top consists of an extruded part specifically made to hover a computer fan over the Zedboard for cooling. Figure 86 shows the top compartment in a Solidworks model. With all parts of the case assembled together in Solidworks as shown in Figure 87, the final dimensions are 6.50”x 7.50” x 7.50”.

Figure 84: Solidworks Model of Casing Side Part with Rectangular Cutout. The dimensions of the cutout are 6.50” x 7.50”. A cutout of 1” x 5.67” was made to help manage airflow and for accessibility to the Zedboard.

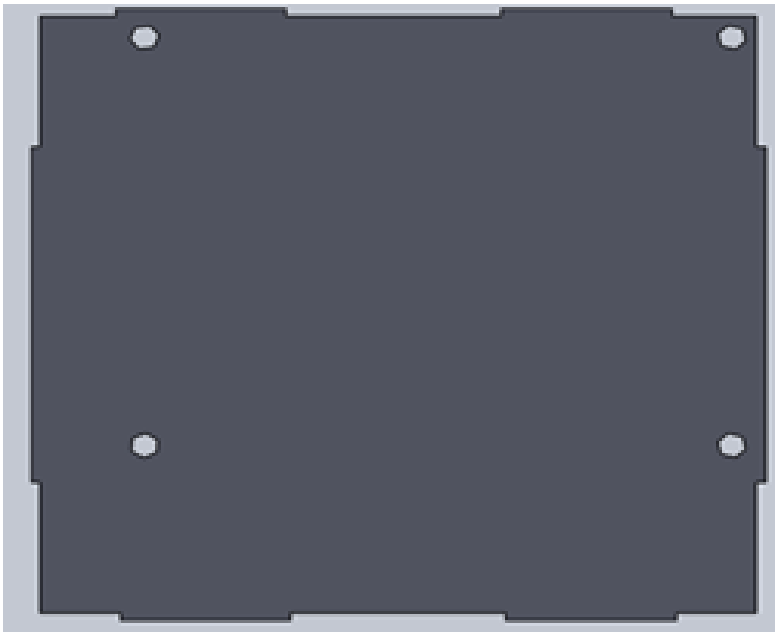

Figure 85: Solidworks Model of Bottom Casing Part with Four Hole Cutout Features. The dimensions for this part are 7.50" x 7.50". The four holes are for the standoffs needed for the Zedboard.

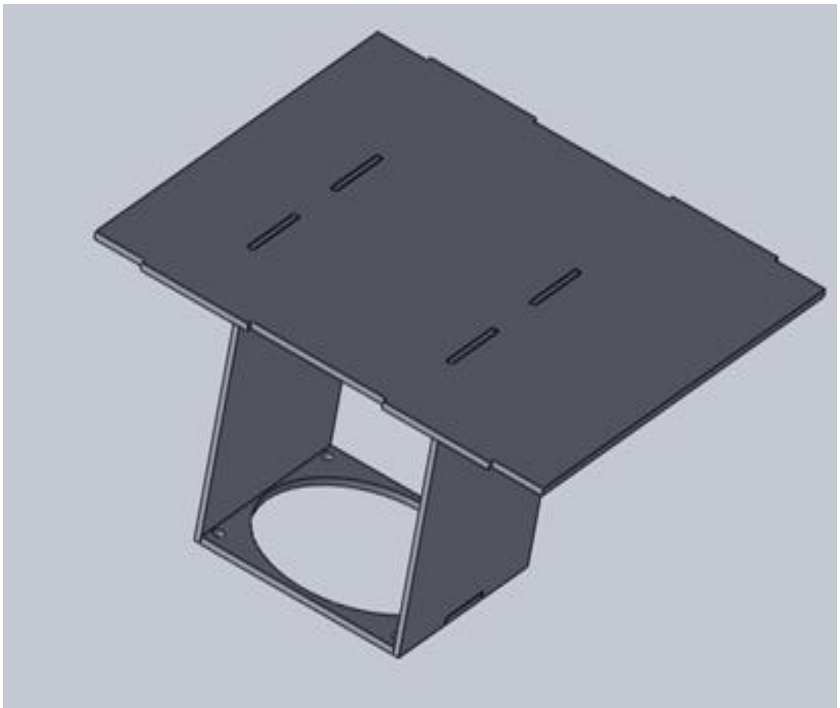

Figure 86: Solidworks Model of Top Compartment with Holder for Fan. The dimensions for the top compartment are 7.50" x 6.37". The circular component serves as a mounting bracket for the fan.

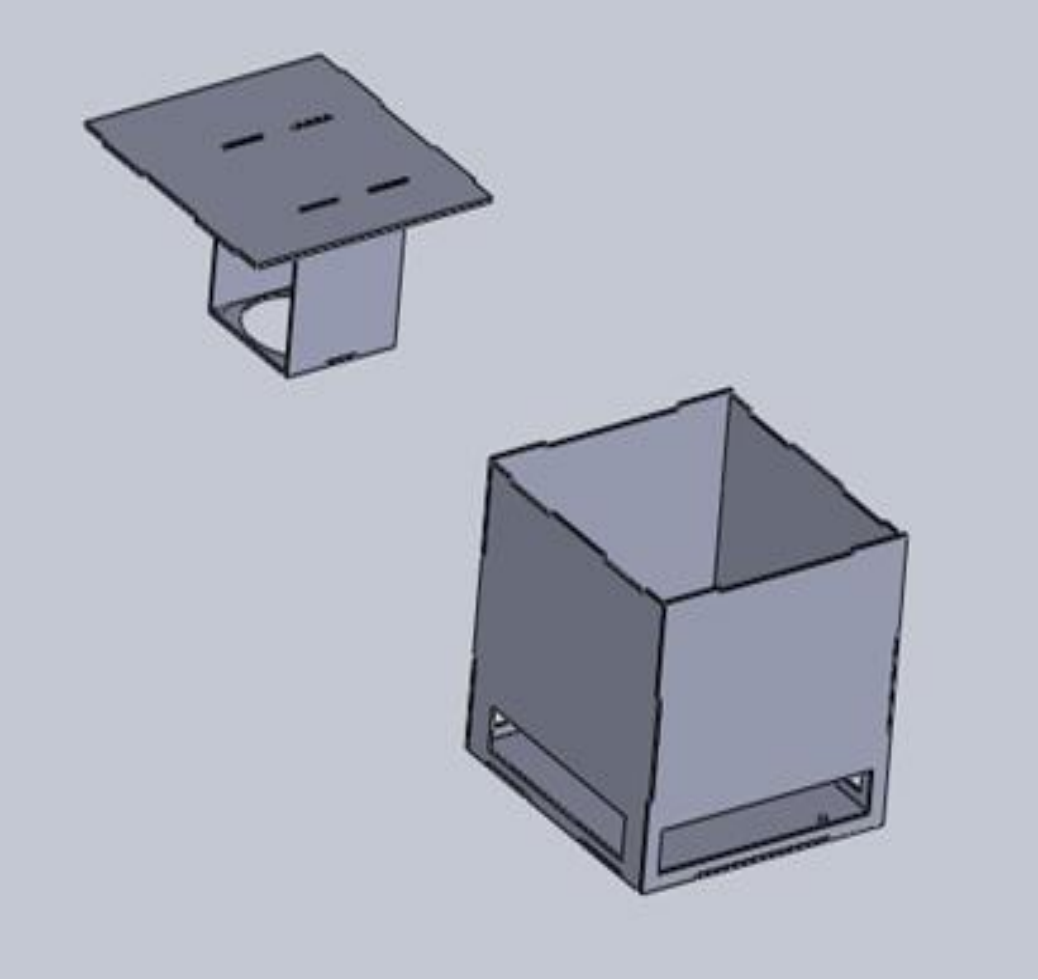

Figure 87: Solidworks Model of Acrylic box with All Parts Assembled Together. The box was designed to be assembled and disassembled quickly if needed.

Acrylic is the material used for each part of the case. This material has many advantages that are beneficial to the final design. It is capable of being milled, cut, and machined easily with common wood tools, unlike glass or metal [111]. Overall, it is easy to work with and durable enough to hold the Zedboard, LIDAR sensor, and fan in place. Figure 88 and 89 below both show how the devices are positioned using the available casing.

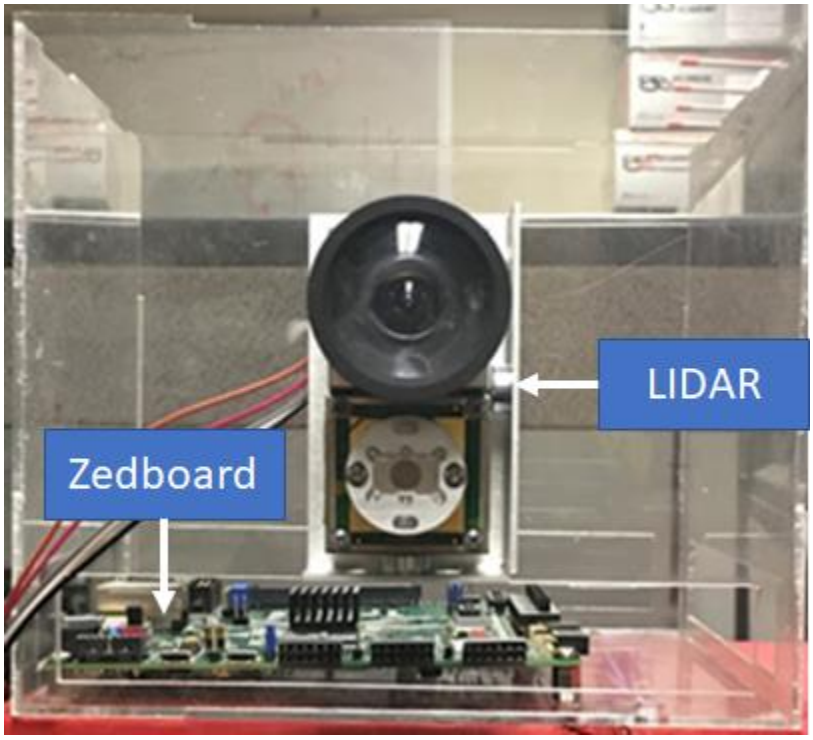

Figure 88: Acrylic Container Containing the Zedboard and LIDAR Sensor. The LIDAR and Zedboard were mounted with standoffs into the Acrylic container. The cutouts in the side panel of the acrylic box allow for easy access to the Zedboard and for wire management.

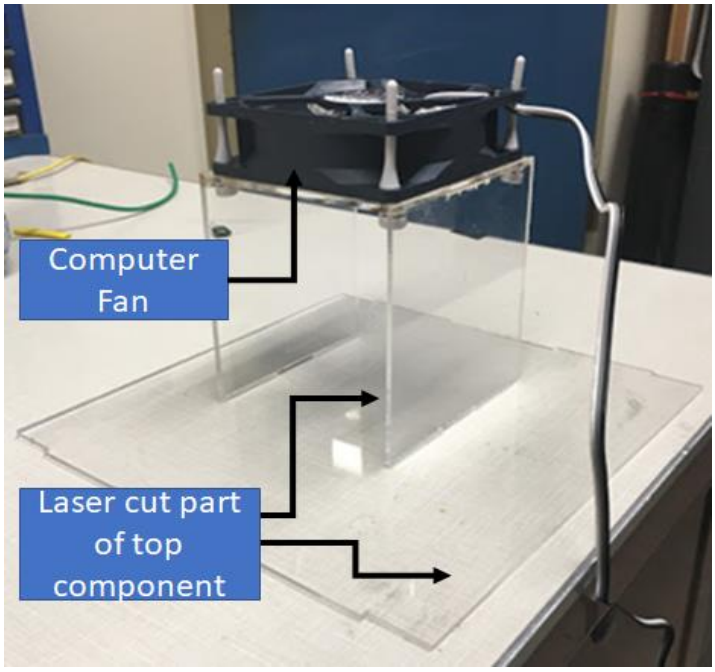

Figure 89: Acrylic Top Casing Part with Fan Component Attached. Rubber standoffs were used to attached the fan to the casing.

The actual test of the system is based off the placement of each device. In order to mount the acrylic box securely to properly perform tests of the overall system, a wooden structure was created, which is shown in Figure 90.

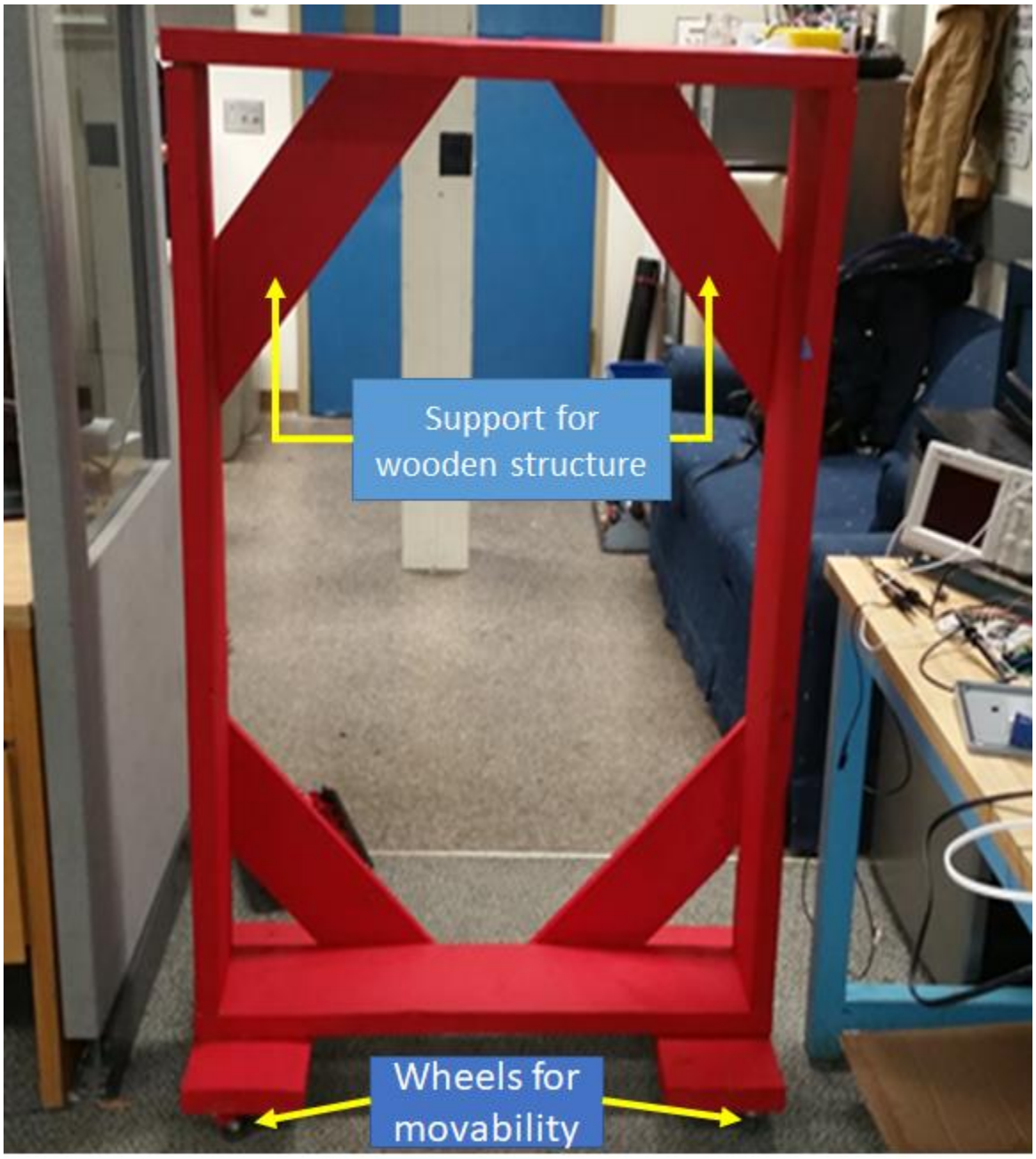

Figure 90: Wooden Structure Used to Handle All Components of the Autonomous Bus System. The wooden structure was designed to be mobile and support the implemented design. A shelf was added to hold the acrylic box holding all of the necessary components and to provide a mounting point for the PIRs.

For placement of devices in the acrylic box, the LIDAR required drilling of holes in one side of the casing. The LIDAR sensor was mounted by four 440 standoffs that are ½" long. Figure 91 shows the placement of the sensor on the side of the box. As for the Zedboard, it was prepackaged with standoffs, therefore; screws were attached onto the bottom of the acrylic to ensure no type of movement.

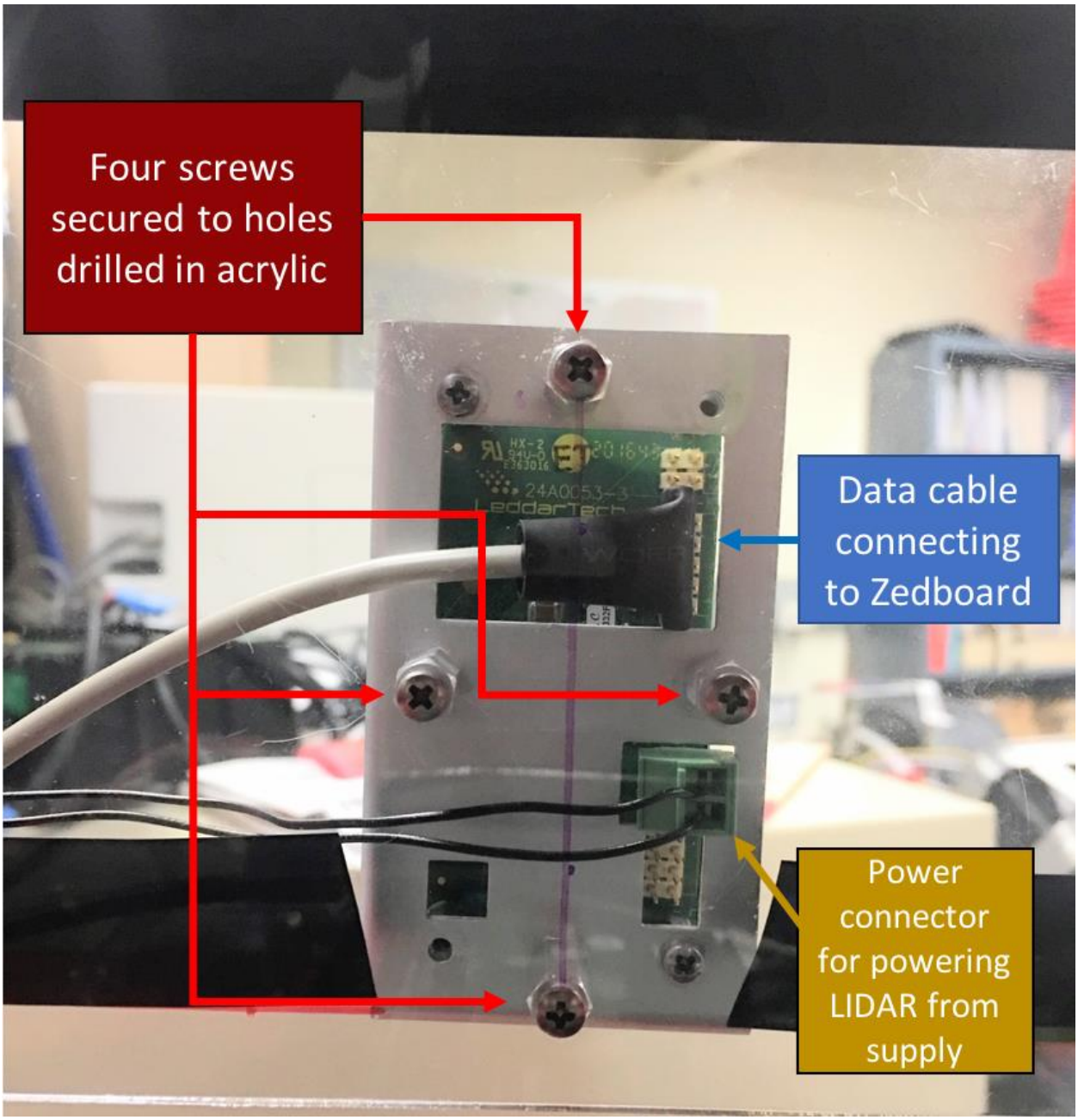

Figure 91: LIDAR Sensor Mounted onto the Acrylic Box (Rear View) with 440 standoffs, 1/2" Long. Four holes were drilled out to allow for the LIDAR sensor to be easily mounted.

A shelf was created to provide a designated height for the LIDAR to detect objects or humans reasonably. Once every individual part was pieced together in the acrylic case, the power supply and the VGA screen were then mounted onto the structure as well. Figure 92 shows the final implementation of all components needed for the detection system.

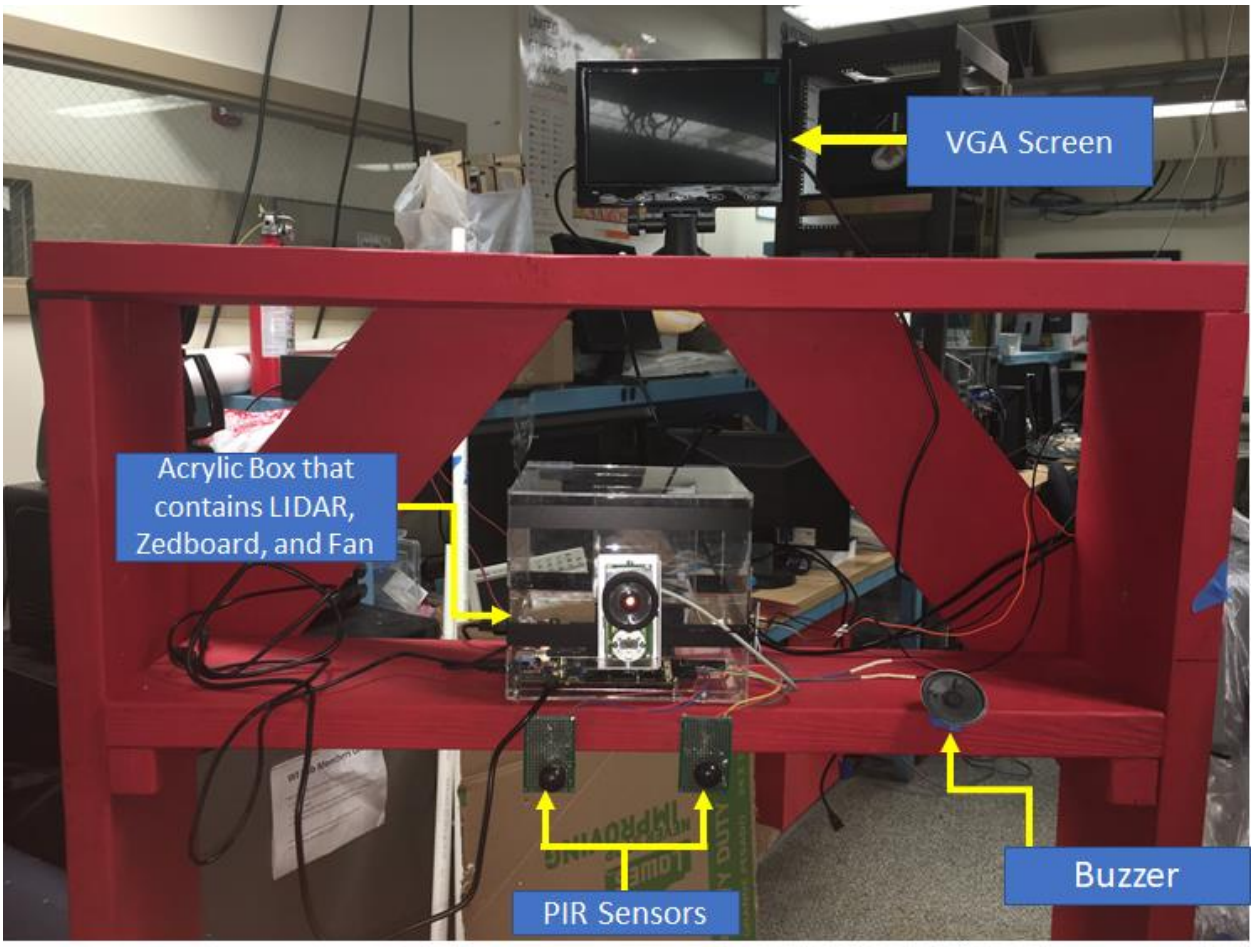

Figure 92: Final Implementation of the Components with the Acrylic Box, the Two PIR Sensors, a VGA Screen and a Buzzer Mounted onto the Wooden Structure. This final implementation was used to test the accuracy and response of the system.

With the entire system assembled, the cooling fan was tested to ensure that the Zedboard would remain cool during system testing. As a precaution, it was necessary to keep the Zedboard from overheating so that no components would get damaged. To test the effectiveness of the cooling fan, the entire system was powered on for an hour with the ambient temperature inside the box being taken every ten minutes. The temperatures taken throughout the test are shown in Table 32.

Table 32: Temperatures Recorded from Inside the Acrylic Box. The temperature was taken every ten minutes and it remained consistent over the course of 60 minutes.

Time Elasped (minutes)	Temperature (degrees Celsius)
0	24
10	23.9
20	23.8
30	23.9
40	23.4
50	23.8
60	23.9

The temperature remained fairly constant around 23.8 degrees Celsius. During the development phase, the Zedboard did not show any signs of overheating, however, the team wanted to implement this cooling system to be sure that heat issues would not occur over time. These results showed that it would be unlikely for any overheating to occur with the fan in place.

With the components, casing, and structure now fully assembled, the team was able to begin creating a plan for overall testing procedures.

7.7: Stationary Bus Model Planning

Creating a system to simulate the bus turning was necessary. Rather than creating a moveable 40-foot structure, the team worked on creating a stationary bus model. Objects would move around the stationary bus model to test the effectiveness of the pedestrian detection and avoidance system. Figure 93 explains the idea and movement of an object around the stationary bus model. The yellow triangle is the object, while the bus is represented by the olive green box. Figure 94 shows how the blind spot area rotates as the bus turns towards the pedestrian.

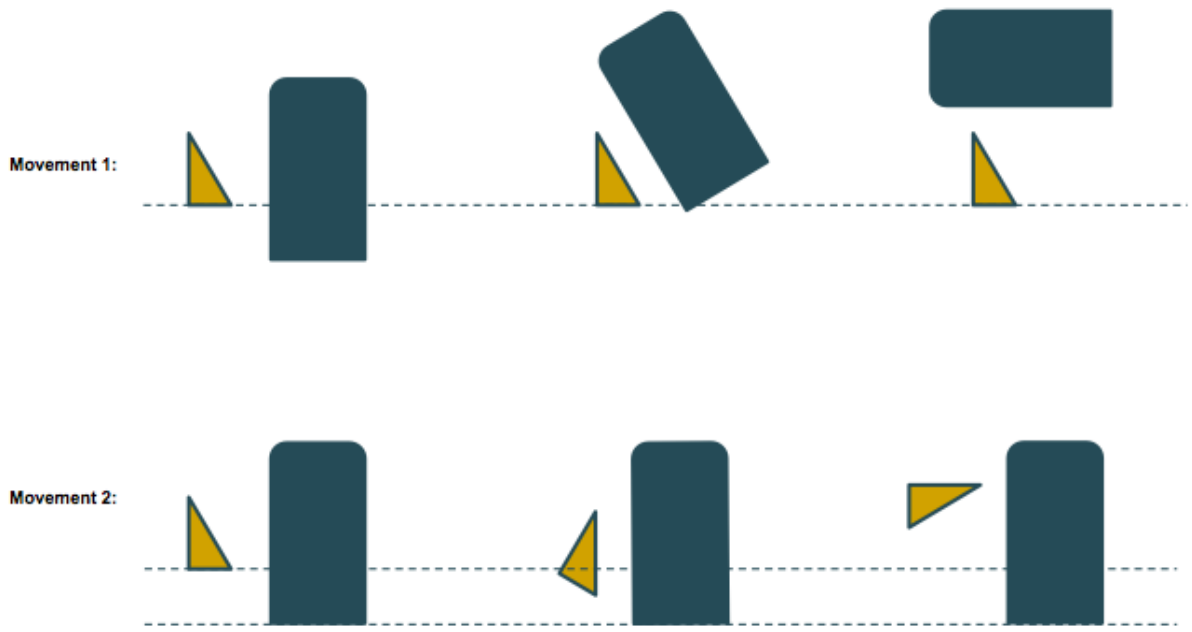

Figure 93: Stationary Bus Model Movement. Showing the change in location between the pedestrian and the bus during a bus turn. The yellow triangle represents the pedestrian while the olive green box represents the bus. Movement 1 represents a bus turning towards a pedestrian while Movement 2 represents the pedestrian's movement in relation to the bus.

In Figure 93, there are two types of movements. Movement 1 depicts how a bus would make a left turn in relation to the stationary object. This was used to create Movement 2, in which the object moves in relation to the stationary bus. This was useful because the team was not able to acquire an actual bus, and needed to have a test scenario that could be controlled by the team. With the stationary bus model, the team was able to keep the structure stationary and map out a path for one of the team members to walk. This was necessary for testing the system to get accurate results with the absence of a real bus.

With the stationary bus model selected, the next stage was determining where on the bus the blind spot is in relation to the pedestrian in the crosswalk. Figure 94 depicts the blind spot of the bus (shown in red) and how it would move with the bus.

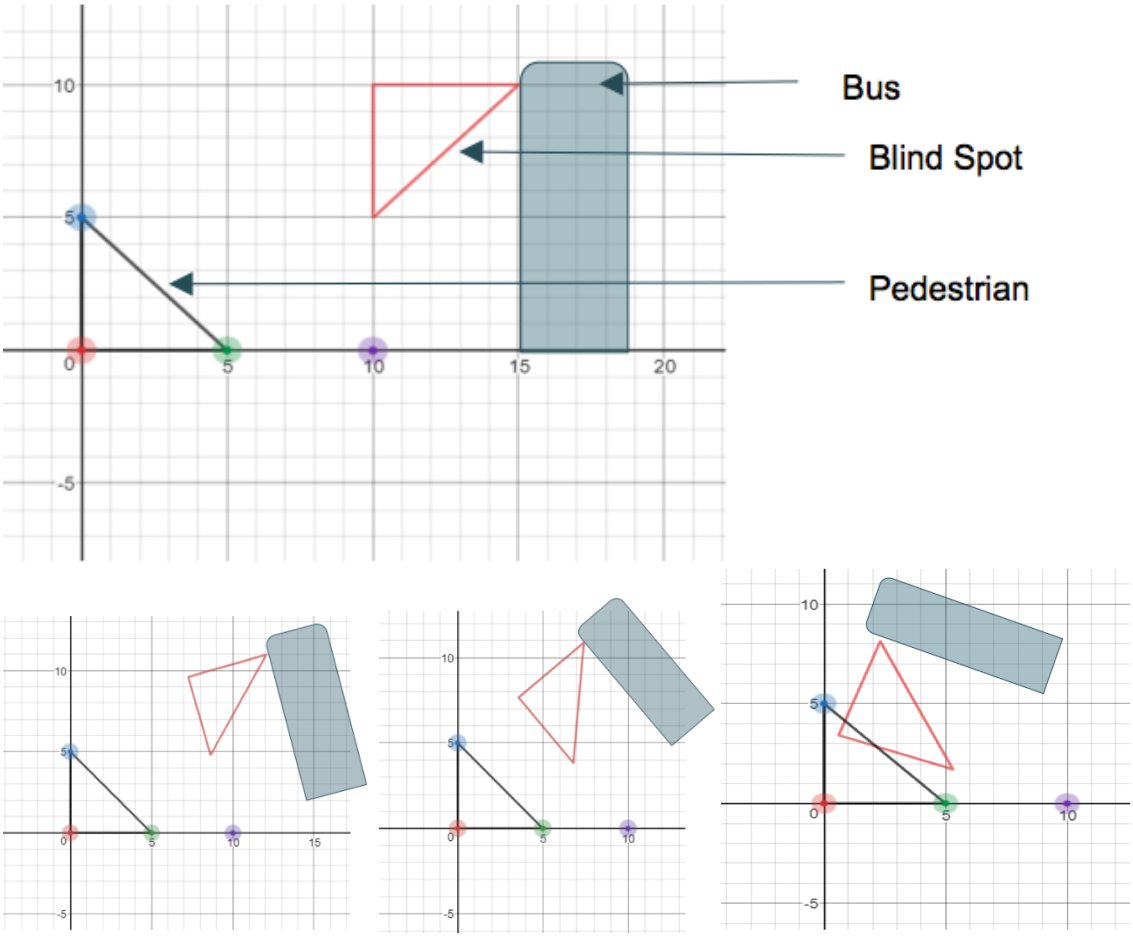

Figure 94: Representation of a Bus Turning by a Pedestrian. Representation of the change in location between the bus and the pedestrian by the pedestrian's field of view.

Figure 94 is a more detailed version of Movement 1 within Figure 93. The blind spot attached to the bus moves in sync with the bus around the left turn. The pedestrian is not in the blind spot until the bus has completed about ¾ of the left turn. This means that, when the bus is straightening out and going forward, the bus driver is not aware of the pedestrian in the possible danger zone.

In order to test the system, the team created a way for the pedestrian to walk into the blind spot of the bus. This meant reversing the movement of the bus in relation to the pedestrian within the sidewalk. In Figure 95, the pedestrian is now moving in relation to the bus. Here, the red triangle represents the pedestrian. The bus is labeled along with its blind spot, shown in black.

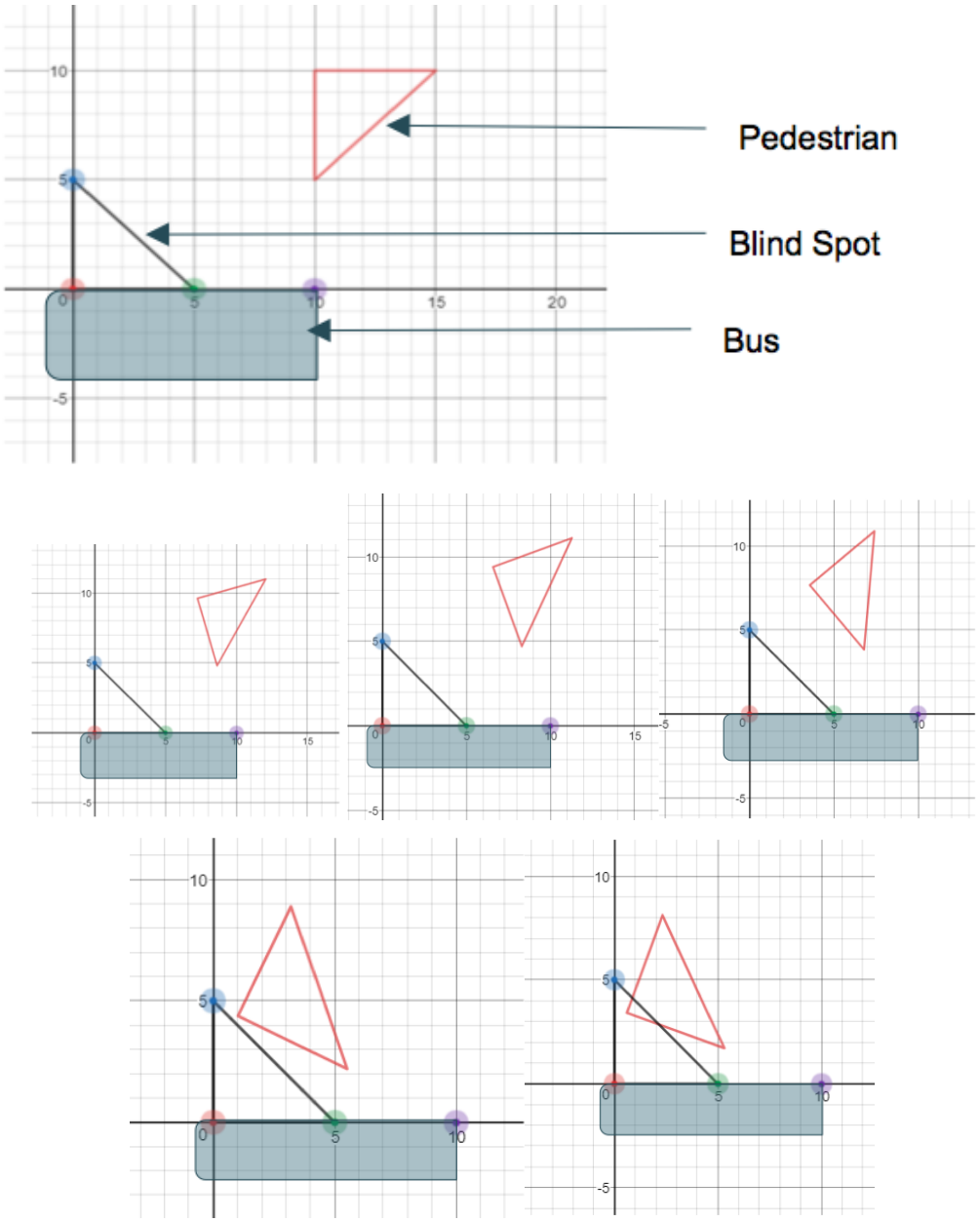

Figure 95: Reverse Bus and Pedestrian Movement. Change in location between the bus and the pedestrian while the pedestrian enters the blind spot. The bus remains stationary while the pedestrian (red triangle) moves into the blind spot of the bus.

The team used this stationary bus model for system level testing as shown in section 8, where the pedestrian walked up to the bus in the same manner. The pedestrian followed a certain path leading up to the blind spot of the bus, mimicking the bus turn in relation to the pedestrian.

In order to have a deeper look at the movement of the tires of the vehicle, the Ackermann Steering model was taken into consideration. The intention of Ackermann geometry is to avoid

the slip of rear tires sideways when following the turning path around a curve. The geometrical solution to this is for all wheels to have their axles arranged as radii of circles with a common center point. As the rear wheels are fixed, this center point must be on a line extended from the rear axle. Intersecting the axles of the front wheels on this line requires that the inside front wheel is turned at a greater angle than the outside wheel. The equations in Figure 96 are used for the rotation transformation with the back axle of the bus as the center of rotation. Figure 97 outlines the parameters needed for the Ackermann model. T is tread, and L is the wheelbase. Tread is the distance between the center point of the tires in width, while L indicates the length.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta. \end{aligned}$$

Figure 96: Equations Used for the Ackermann Steering Model. These equations were used to model the path of the bus with an ideal center tire on the back axle of the bus.

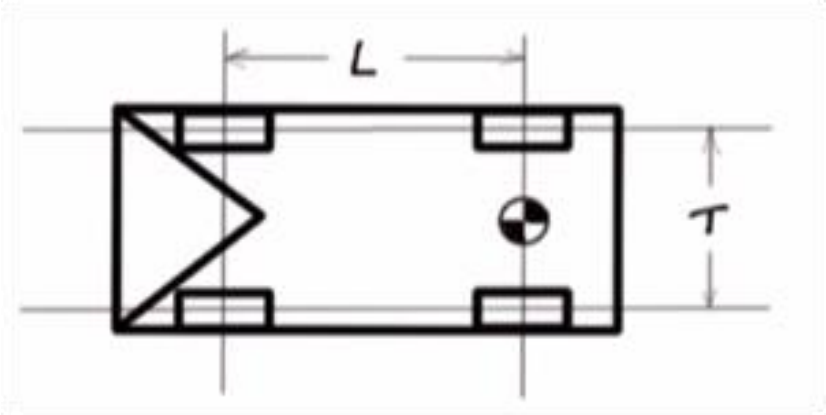

Figure 97: Image Showing the Parameters of the Ackermann Model. The back axle center point is represented by the black and white circle. The parameter L represents the length from center point of both of the axles while T represents the distance between the two tires on the same axle. [112]

When the bus is moving, the path of the rear axle center point is measured. The rear tires may not travel in the same path as the front tires, as seen in Figure 98. To make up for this, the rear axle is replaced with an ideal tire that would correspond to the path of the front axle, as seen in Figure 99. When going straight, a line through the center matches with the front and the rear.

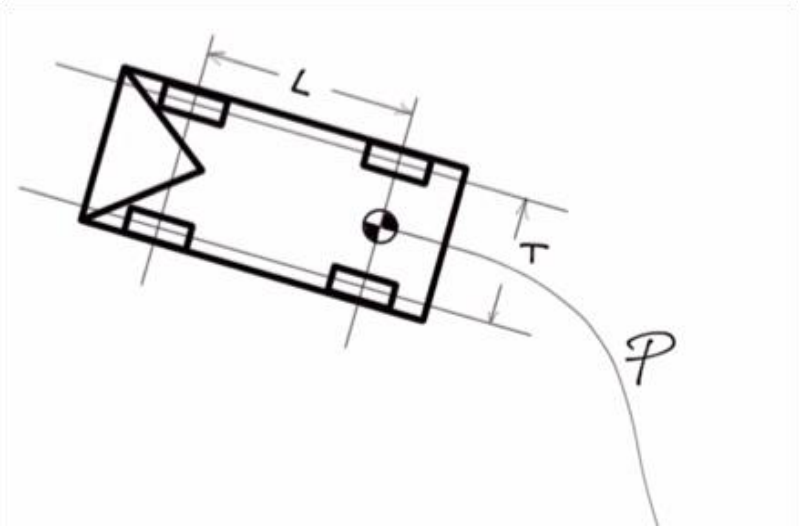

Figure 98: Image of the Rear Axle Following a Different Path from the Front Axle. This models a standard turn by most automobiles where the rear axle is fixed. [112]

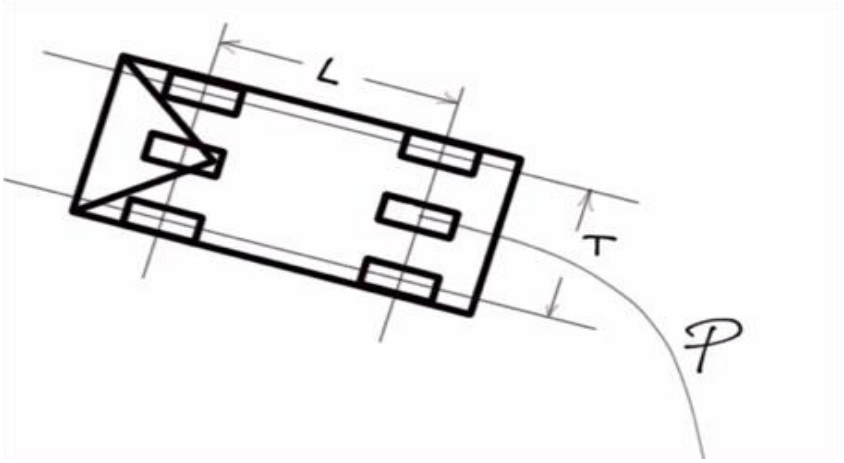

Figure 99: The Ideal Tire in the Front and Rear Axle. In this case, the ideal front and rear tire follow the same path. [112]

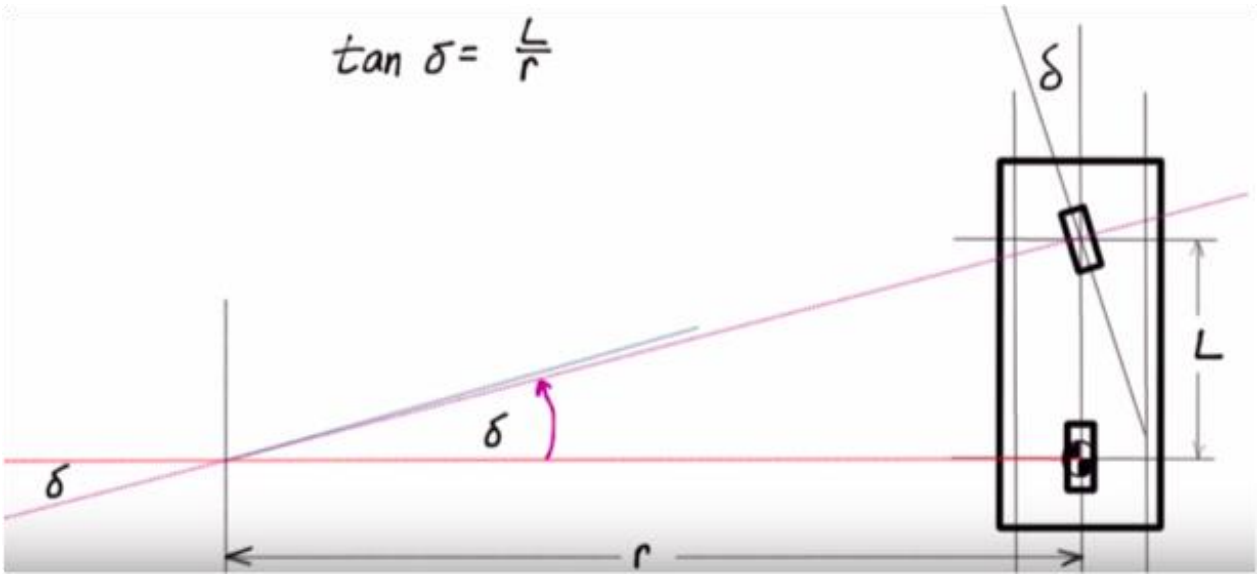

Figure 100: Equations and Parameter Needed for the Ideal Tire Model. "L" is the distance between the axles of the front and rear sets of tires. "r" is the distance from the pedestrian to the rear axle center. "δ" represents the angle from the pedestrian to both of the bus axles. Using these parameters, the movement of the bus can be simulated. [112]

Table 33: Results from Each Individual Equations in the Ideal Tire Model. The parameter "t" was determined from the time it would take a bus to make a left turn in the middle of the intersection. The parameter "r" was determined based off the turning movement of the bus. With these values, "dθ/dt", "δ", and velocity was determined with tan(δ)= L/r.

r (distance between pedestrian and the center-back axle of the bus)	**t (time in seconds while the bus is turning)**	$d\theta/dt$ **(the front wheel rate of change of angle**	**δ (Angle from the pedestrian to both bus axles)**	**V(m/s) (Velocity of the bus in relation to defined parameters)**
6	1	36	80.54	6
5.5	2	9	81.31	2.75
5	3	4	82.09	1.67
4.5	4	2.25	82.87	1.125
4	5	1.44	83.66	0.88
3.5	6	1	84.45	0.58
3	7	0.73	85.24	0.43

Table 33 shows the necessary equation and parameters used for the ideal tire model. Figure 100 shows the results from the equations used. The parameter L was selected to be 40 feet, the standard size of a bus. The distance between the pedestrian and the center back axle was predetermined along with the corresponding time in seconds of the bus executing a turn. To get the front wheel rate of change of angle, the following equation in Figure 101 was used.

$$\frac{d\theta}{dt} = \frac{V}{t} * \tan\delta$$

Figure 101: Rate of Change Equation for Front Wheel Angle. Used to express the rate at which the front wheel angle (θ) changes as an expression of velocity and the angle from both bus axles.

Figure 102 shows a scaled down version of the reverse pedestrian movement path using the results from Table 33. Styrofoam blocks were used to outline the bus and its blind spot. The model below was scaled 1:10. The value of r and δ were used from Table 33 to outline the movement of the pedestrian. A protractor was carefully used to mark the angle for the next block to be placed for each new value of r.

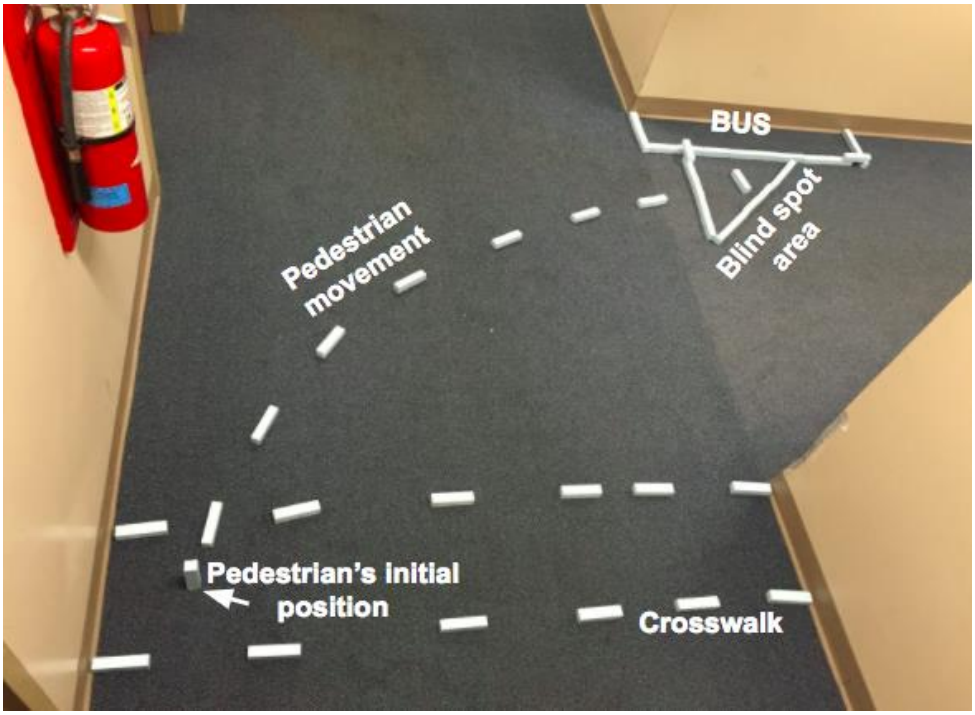

Figure 102: Scaled Down Reverse Pedestrian Movement Model. Foam blocks are used to show the path of the pedestrian's movement into the bus's blind spot area. This is a movement of the pedestrian, instead of the bus, to mimic the bus making a left turn.

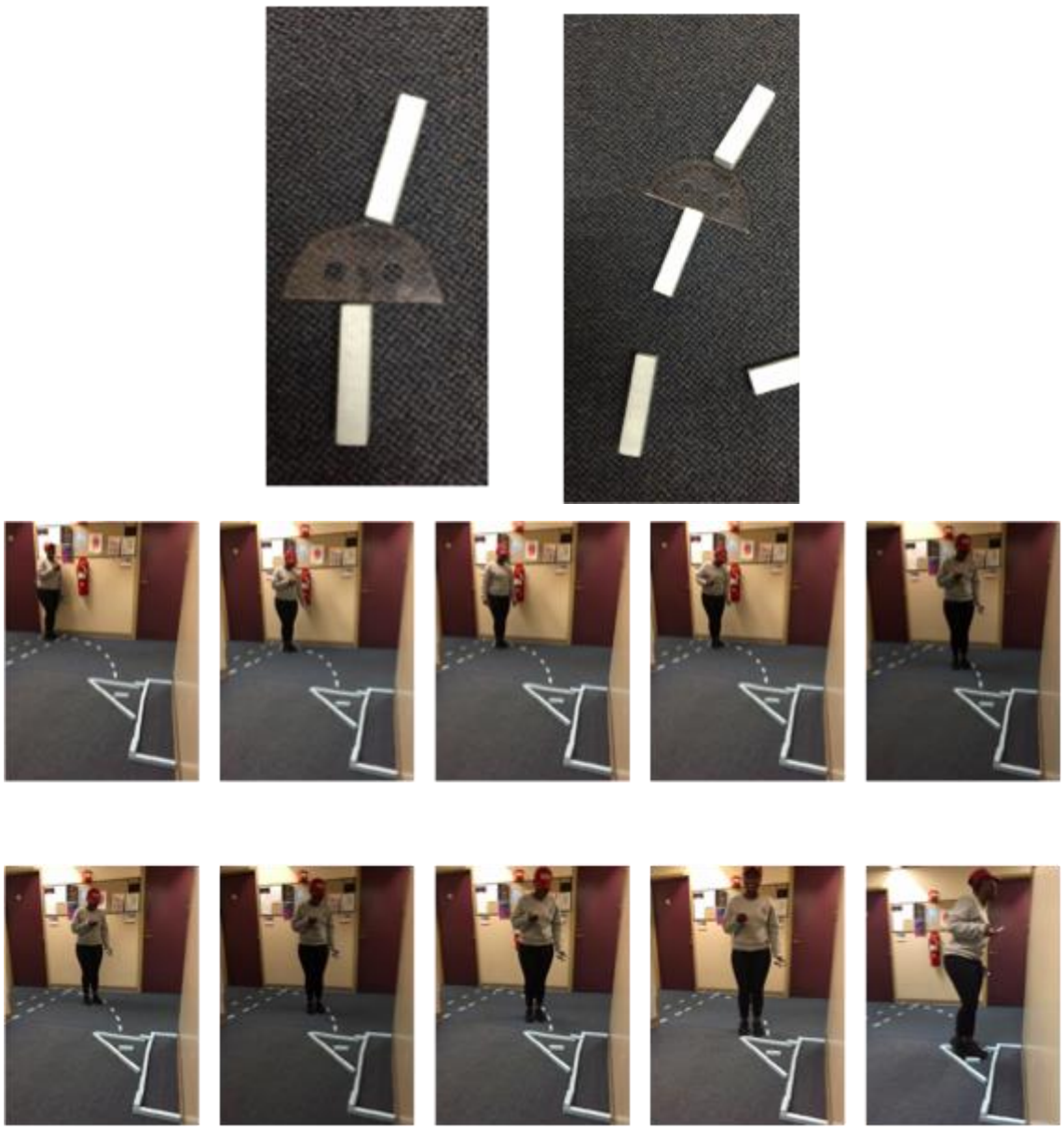

Figure 103: Construction and Execution Of The Model. A protractor was used to map out the angle from the pedestrian to both bus axles. Using foam blocks, the test subject was able to follow the path into the blind spot area of the bus.

Figure 103 shows how the model in Figure 102 was constructed and tested. The pedestrian is to be detected by the system once he/she enters the blind spot area (the triangle) to warn the driver.

7.8: Chapter Summary

This chapter provided a detailed explanation of the methodologies used in making this project a reality, including the design of the sensor logic and software, power system, casing and structure assembly, and reverse-movement stationary bus model testing plan. Hardware logic modules were created to control the acquisition of distance data and human detection from the LeddarVu8 and passive infrared sensors, respectively. The distance data was passed over an implemented AXI bus to be computed and checked against predefined thresholds in the Zedboard's processing system. This determined when an object was within the blind spot area surrounding the bus. Signals were passed accordingly back to the programmable logic, which was fused with the received passive infrared sensor data to determine when a human was in danger and alert the driver via visual and audible warning. The power supply was established to maintain a 12 volt connection with all the electrical components at their required current level, and mounted to a wooden frame for optimal wiring. An acrylic casing housed the sensitive devices and secured them in place, while providing cooling to avoid any potential heat issues. Finally, a plan was devised for testing the system from a stationary location by replacing the turn path of the bus with an equivalent movement by the pedestrian. With all of these components in order, the team was ready to begin testing the overall system.

8: Testing & Results

Once all of the components were mounted on the bus model frame, the LIDAR sensor and Passive Infrared sensors (PIR) were tested to ensure accurate detection of a pedestrian. In this testing scenario, the accuracy of the system was tested as well by determining if false alarms would be a potential issue. The testing of the stationary bus model consisted of two parts, the first one was to assure the detection (and differentiation) between objects and humans. The second part was testing of the accuracy of the system. In the background, it was determined that the average blind spot for city buses reaches between 13-16 ft (3.96-4.88m) from the bus itself, as shown in Figure 104.

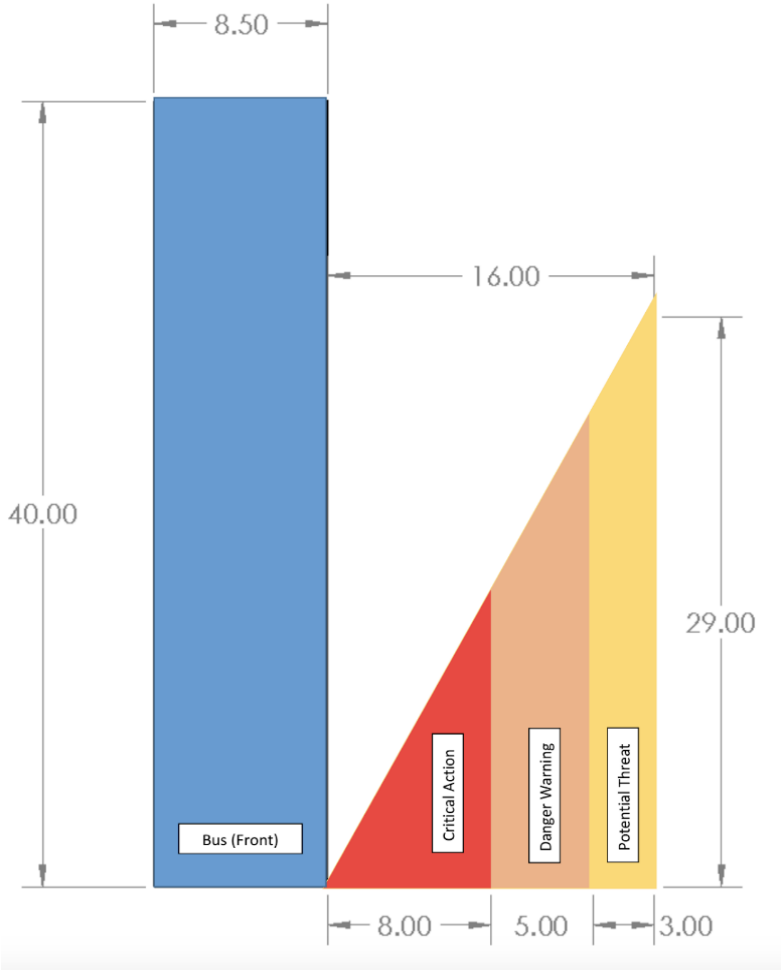

Figure 104: Bus Blind Spot with Indicated Zones of Detection (Units in Feet). Blind spot reaches up to 16 feet from the driver's side window. The three zones represent increasing level of danger to the pedestrian, as the distance between the pedestrian and bus narrows.

Two zones are being utilized for the system designed. The first one is the critical action zone, which reaches up to 8 ft (2.44m) from the bus. The second zone is the danger warning zone that starts after the critical action zone and is 5 ft wide, reaching up to 13 ft (3.96m) from the bus itself. After the danger warning zone, the area around the bus is clear indicating that there is no danger, but the system has the ability to still detect and monitor the distance of objects in the potential threat zone pictured.

The testing was performed based on the system implementation and its data fusion. The LIDAR sensor is always performing object detection during normal operation. In case that no object is detected in either the critical action or danger warning zones, the area will be considered clear and the driver will be notified of that by the screen displaying green. In the case of an object being detected in the danger warning zone (regardless of being an object or a human) the driver would be alerted by the screen turning yellow. The testing of the critical action zone is more complex than the danger warning zone, due to the fact that the difference between human and object will be made in this most critical zone. If an object detected is within the critical action zone, and has been determined to be human based on the PIR readings, the driver would be notified by the screen flashing red and the buzzer beeping, indicating a potential accident. If the IR readings do not determine that the object is human, the detected object is said to be inanimate or non-human, and the screen remains yellow.

In order to assure full functionality of this part of the system, a variety of tests were performed in the critical action and in danger warning zones, as well as beyond the danger warning zone, to assure that no false alarms are triggered by movement outside of those zones. The testing was performed in an open, wide area and the danger zones were marked by tape on the ground as shown in Figure 105.

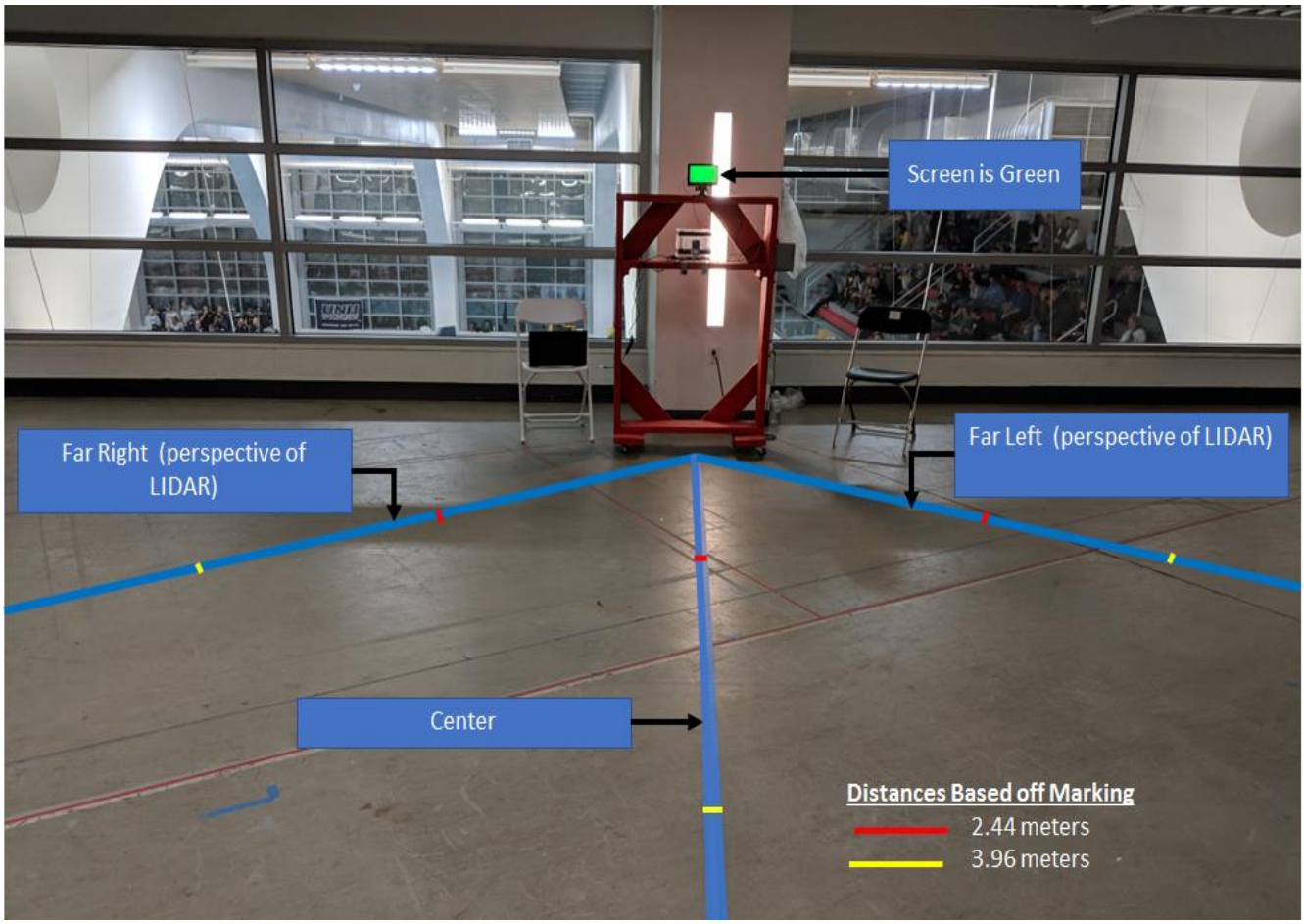

Figure 105: Testing Setup of the Indicated Blind Spot Zones for the Transit Buses. The 100-degree field of view of the LeddarVu8 is mapped out on the floor in blue with a center line originating from the sensor. The critical action zone and danger warning zone are highlighted in red and yellow markings, respectively. This setup is used to test the detection of objects and pedestrians in each zone.

As shown in Figure 105, the tape in the center is aligned with the LeddarVu8 LIDAR sensor, while the two tape strips on the sides are the extreme edges of the 100-degree field of view (FOV) angle of the LIDAR sensor. The tests were performed in various locations, both on the thresholds of each zone and within each zone. The variety of tests performed were as follows:

1. People standing and walking through the zones and in different areas of the zones.
2. Sliding/rolling objects through the different zones.
3. Having objects and humans in the blind spot area to see how the system reacts.

For the first test, the main focus was on the thresholds in order to assure that the system can detect whether the object is in the critical action, danger warning, or in none of the zones. This test can be seen in Figure 106, where the subject is standing on the outer edge of the danger warning zone. In this case, the screen turned yellow.

\

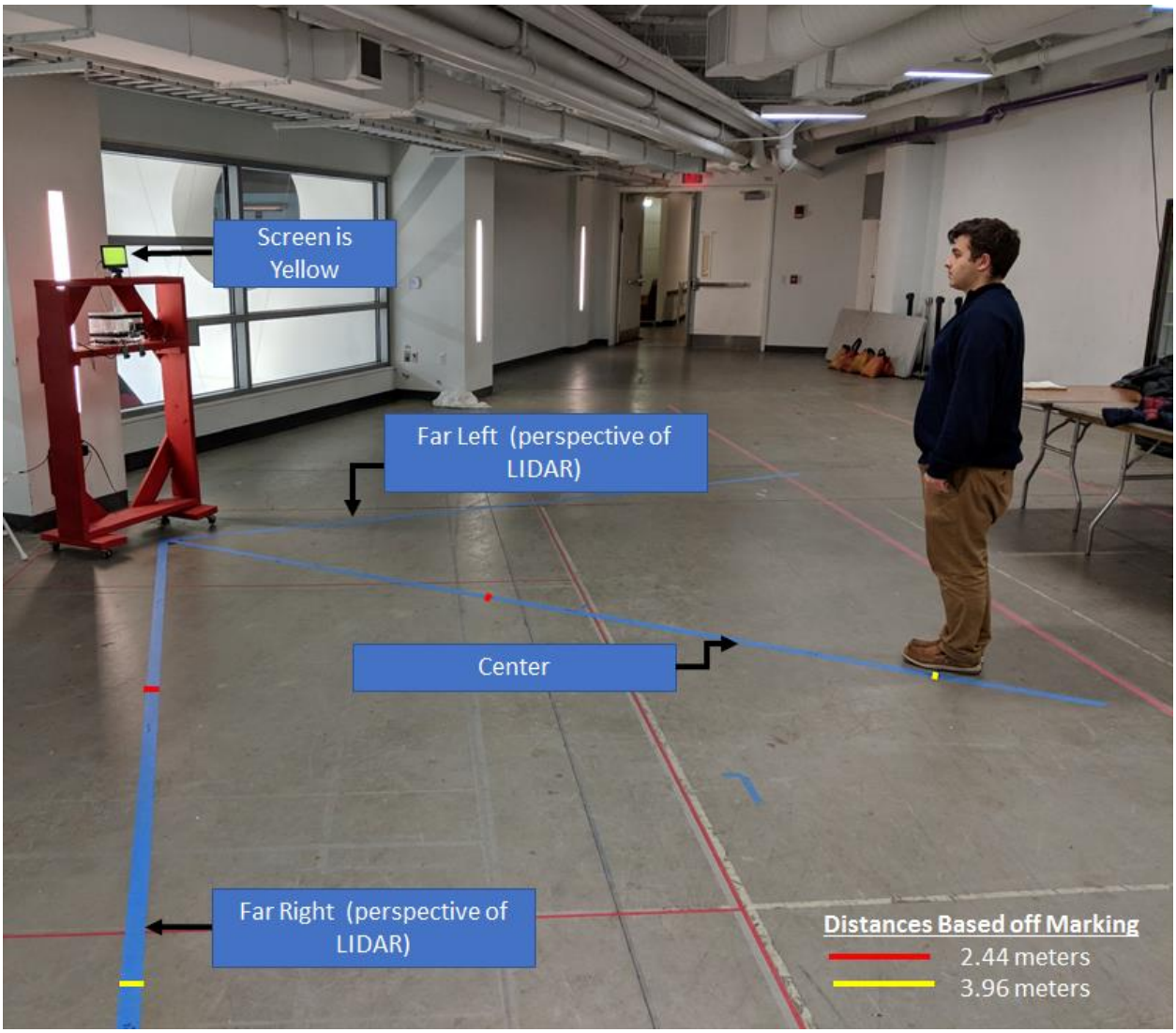

Figure 106: Distance Threshold Test at Danger Warning Line. Test subject is at the 3.96 meter mark, which is the outer threshold of the danger warning zone. As shown, the screen turns yellow because the test subject is in this trigger zone.

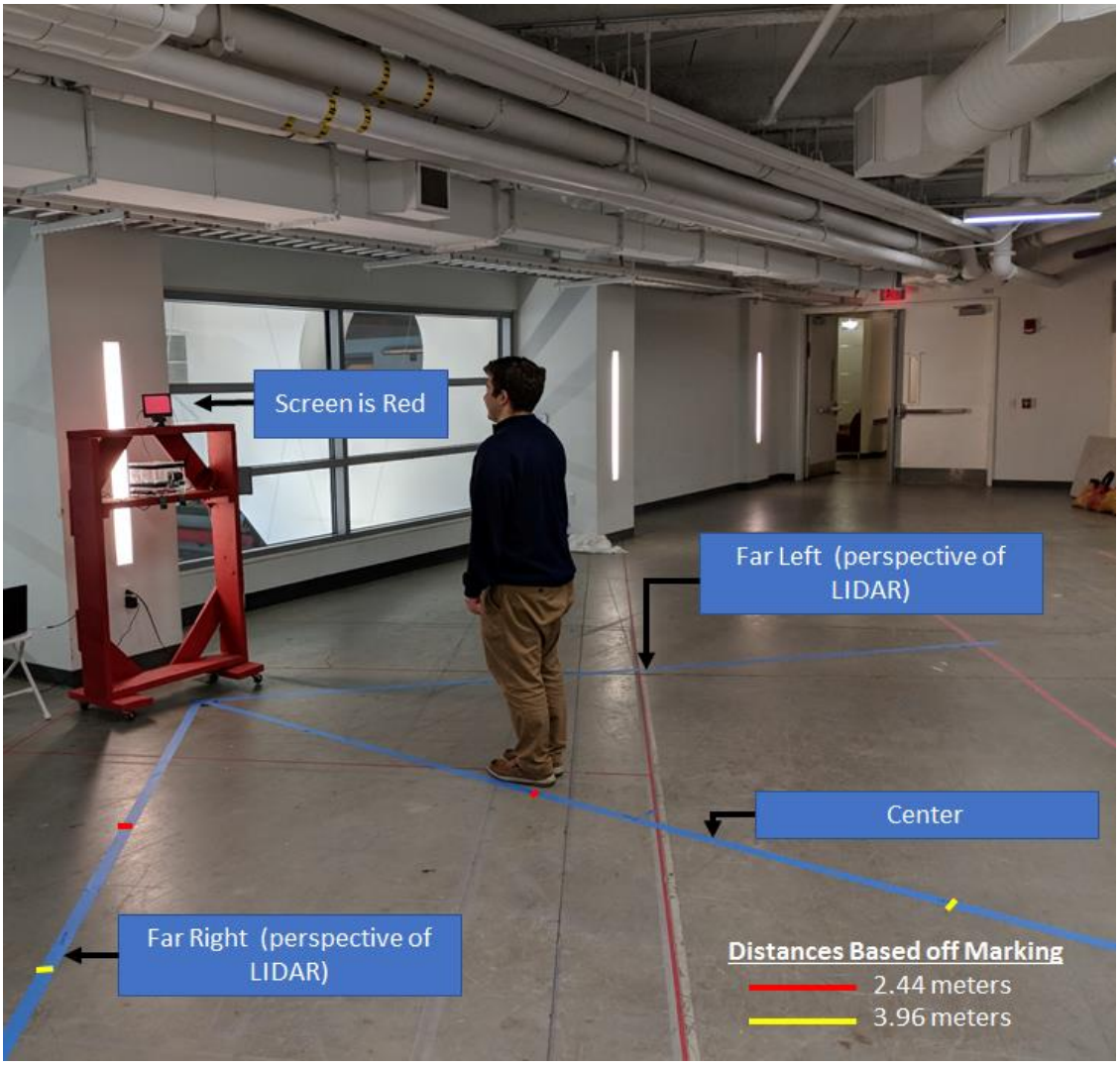

Figure 107: Distance Threshold Test at Critical Action Line. Test subject has moved into the critical action zone, which is past the red mark. As shown, the screen turns to red because he has been detected as a human entering the critical action zone, as opposed to an inanimate object.

After the person moved in to the critical action zone as shown in Figure 107, the screen turned red and buzzer started sounding as expected. The critical action and the danger warning zone thresholds were marked on the tape with a marker at the distances defined as before at 8 ft (2.44m) and 13 ft (3.96m). Other testing was performed, such as rolling a whiteboard through the danger warning zone and the critical action zone, as well as out of any of the detection zones multiple times. Once the whiteboard was rolled by a person (who was standing outside of the detection zones) through the danger warning zone, the LIDAR picked up the moving object, therefore the screen turned yellow indicating that there is an object within the threshold. The

same rolling whiteboard test was done within the critical action zone, and the screen turned yellow due to the fact that the object was detected, yet the PIR sensors did not detect a body that was emitting heat in the critical action zone. After the testing with the whiteboard was completed, the same tests were performed with a human, walking through the different zones and the results were the following:

1. Walking through the danger warning zone turned the screen yellow.
2. Walking through the critical action zone turned the screen red and the buzzer played sound.

An additional test was performed by using the whiteboard as well as having a person in the detection zones to see how the system would behave and the decision it would make based on having an object and a human in the blind spot areas. Four tests were performed which were set up in the following way:

1. The person and the board were in the critical action zone.
2. The person was in the danger warning zone and the board was in the critical action zone.
3. The person was in the critical action zone and the board was in the danger warning zone.
4. The person and the board were in the danger warning zone.

The data fusion of the system works as following: Once an object is detected in either of the two zones, the LIDAR will pick up that object and will notify the Zedboard about the object detected and the screen will turn yellow. In the case of an object being detected in the critical action zone (whether it is a human or an object), the IRs will be triggered to check if there is a heat emitting body in either the critical or the danger zone. Therefore, the following results were obtained by the four different test setups. For the first scenario, since both the human and the object were in the critical action zone, the PIR sensors were triggered and detected the heat emitted by the human body. The screen turned red and the buzzer sounded. In the second scenario, the board was located in the critical action zone, therefore the IRs were triggered to detect a heat emitting object, which in this case was the human body in the danger warning zone, therefore the screen turned red and the buzzer was sounding. Even though the board was technically the object within the critical action zone, the alarm was played because the human detected in the danger warning zone could potentially be injured by a real-life bus setting. For

the third scenario, since the person was in the critical action zone and their body temperature was detected by the PIR sensors, the screen turned red with the buzzer sounding. In the last testing setup, both of the objects were in the danger warning zone, therefore only the LIDAR sensor was detecting the object's distance and the PIR sensors were not triggered. Therefore, the screen turned yellow due to the human and the board being in the danger zone. These four testing strategies ensured the full expected functionality of the system, and then more technical tests could be performed.

The accuracy (in terms of distance) of the system was tested in order to check for objects/people being located at measured points (the critical action zone and danger warning zone thresholds). This was done in order to evaluate the accuracy of the LeddarVu8 LIDAR sensor by viewing the distance readings measured once it detects an object in its FOV. The tests were performed at the following positions:

1. The right extreme FOV angle of the LIDAR sensor.
2. The center of the LIDAR sensor's FOV.
3. The Left extreme FOV angle of the LIDAR sensor.

The tests consisted of a person standing at the critical action zone threshold and the danger warning zone threshold, ten times at each of the 3 different test locations. Once the person was at the marked thresholds, the distance calculated from the LIDAR readings was recorded, and a table with 10 data points for every test position was generated. The distances were read using a laptop computer configured to read the processing system's print commands via a Universal Asynchronous Receiver/Transmitter (UART) serial connected console. The average of each set of 10 distance readings was taken once all the data was acquired. The actual distances for each threshold line were compared to the measured values to determine a final idea of the system's accuracy, as shown in Tables 34 and 35. From the data obtained, it is notable that the system at each threshold had a standard deviation of less than 10 centimeters. This tolerance can be considered negligible, relative to the scale of the area that is covered by the system.

Table 34: Danger Warning Zone Test Values Obtained for the Threshold. 10 trials were performed at the far left, far right, and center of the LeddarVu8's field of view to record the distance at which a test subject was detected within the danger warning threshold (3.9624 meters). The data was found to be very consistent with the expected value across all tests for each position, indicating that the LIDAR sensor readings were accurate for this distance.

Trial	**Far Left (3.9624m)**	**Center (3.9624m)**	**Far Right (3.9624m)**
1	4.018	3.929	4.011
2	3.956	3.878	4.016
3	3.990	3.893	3.949
4	3.986	3.852	4.005
5	3.992	3.915	4.11
6	4.004	3.907	4.01
7	3.97	3.916	4.13
8	4.016	3.917	4.108
9	4.034	3.899	4.055
10	4.021	3.905	4.05
TOTAL AVERAGE	3.9987	3.9011	4.0444
STANDARD DEVIATION	0.02443	0.0223	0.05732

Table 35: Critical Action Zone Test Values Obtained for the Threshold. 10 trials were performed at the far left, far right, and center of the LeddarVu8's field of view to record the distance at which a test subject was detected within the critical action threshold (2.4384 meters). The data was found to be very consistent with the expected value across all tests for each position, indicating that the LIDAR sensor readings were accurate for this distance.

Trial	Far Left (2.4384m)	Center (2.4384m)	Far Right (2.4384m)
1	2.241	2.436	2.399
2	2.44	2.39	2.42
3	2.46	2.424	2.383
4	2.47	2.385	2.47
5	2.41	2.375	2.47
6	2.45	2.48	2.474
7	2.5	2.436	2.474
8	2.511	2.411	2.474
9	2.475	2.41	2.415
10	2.522	2.368	2.474
TOTAL AVERAGE	2.4479	2.4115	2.4454
STANDARD DEVIATION	0.08018	0.03410	0.03657

To test the overall movement as if the system were to be implemented on a 40-60 ft Transdev city bus, the reverse pedestrian model was created. Instead of moving the wooden structure relative to the pedestrian, the test consists of moving the pedestrian relative to the movement of a bus making a left turn onto an adjacent lane. The reverse pedestrian model was derived from the Ackermann Steering Model introduced earlier, used to solve the angular distance from the wheels on the inside and outside of a given turn of a vehicle.

The team was able to take the stationary model in Figure 102 in Section 7, and plan out the pedestrian movement in relation to the wooden structure and the blind spot detection zone. To outline the path for the pedestrian for unit testing, the angles calculated in Table 33 were used. The path was carefully marked using a protractor and was joined using a blue tape on the

ground for the pedestrian to follow. Figure 108 shows the path in relation to the blind spot. For the test, the team designated one person to follow the path from the crosswalk to the detection zone of the LIDAR and two PIR sensors, as shown in Figure 109. Measurements were taken at the point where the screen turns yellow upon the danger warning threshold, as well as when the screen turns red upon the critical action threshold. These tests were repeated ten times, recording each time the subject caused the screen to change yellow and red. The measurements for the two thresholds were averaged. The data is organized in Table 36. The average measurement distance for when the screen turns yellow given the test subject has followed the mapped out pedestrian path is 3.6186 meters. The average measurement distance for when the screen turns red when the subject continues on the pedestrian path is 2.3188 meters. These distances were calculated from the second segment on the LIDAR sensor. It is also notable that the overall standard deviation for both test trials were less than 11 centimeters. Since this value is very similar to the deviation found in the distance readings test, it shows the consistency of the LIDAR sensor's ability to determine the distance of an object in its distinct eight segment field of view.

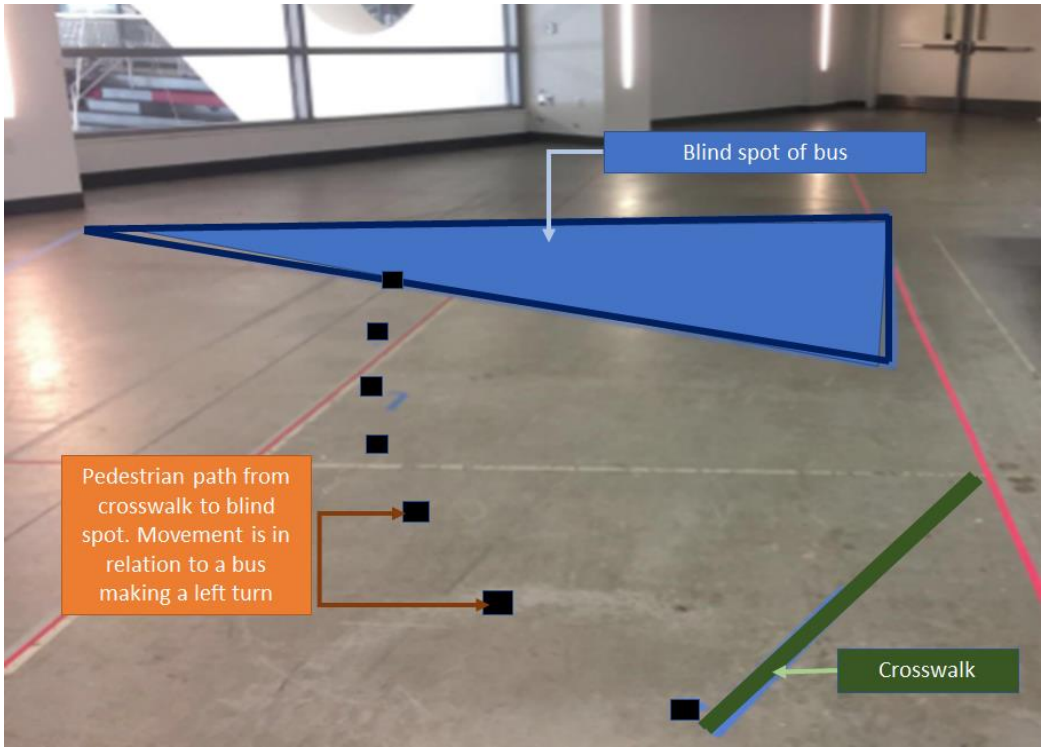

Figure 108: Pedestrian Movement Mapped on Floor. Beginning at the crosswalk, pedestrian movement is shown through black rectangles on floor leading to the blue triangular "blind spot". Depending on where the pedestrian is standing, the screen on the bus system will either stay green, or turn yellow or red.

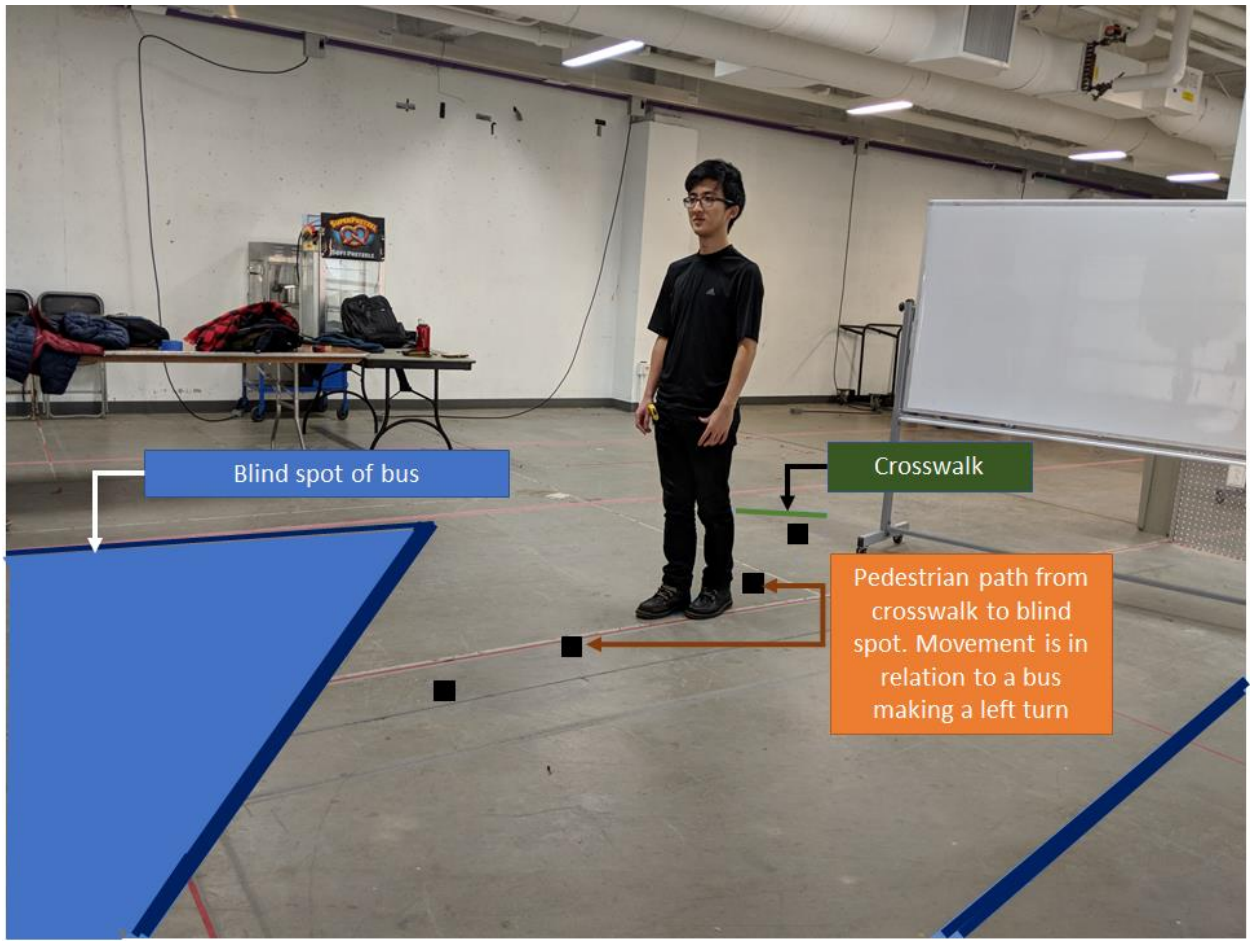

Figure 109: Test Subject Follows Mapped Movement of the Pedestrian Path with Respect to the Bus. Walking slowly along the path starting from the crosswalk, the subject stops at this point because the screen from the autonomous bus system has turned yellow. Subject will continue until the screen turns red and then repeat the process from the beginning.

Based off the obtained results shown in Table 36, the team determined that the autonomous bus system is capable of detecting a human being given the event that an actual bus is making a left turn whilst a pedestrian is about to enter, or within, the crosswalk. This test is significant in that it provides evidence that this system is capable of providing the needed feedback for the driver of the bus to take necessary precaution in either stopping or slowing down the bus.

Table 36: Trial Results of the Reverse Movement Model. 10 trials were performed where a test subject moved into the blind spot area, following the planned path to replicate the turn of a bus. The data was found to be very consistent across the trials, indicating that the system was able to detect a pedestrian entering each zone with a sufficient reaction time.

Trial	Distance When Yellow	Distance When Red
1	3.435	2.313
2	3.558	2.22
3	3.715	2.388
4	3.695	2.424
5	3.623	2.435
6	3.676	2.352
7	3.443	2.278
8	3.616	2.262
9	3.752	2.29
10	3.673	2.226
TOTAL AVERAGE	3.6186	2.3188
STANDARD DEVIATION	0.10925	0.07787

Chapter Summary

This chapter provided an overview of all the testing that was implemented to test the sensor accuracy. There were two types of testing conducted, with the first set of tests consisting of tracking stationary objects or humans in the different zones of the blind spot. The goal of this test was to validate the distances in which the system takes the appropriate danger warning and critical action responses. It was determined that the LIDAR and PIR were able to identify objects

entering different zones of the blind spot with accurate distance readings, as well as whether an object was a human or not. The crossing of distance thresholds was represented on the warning screen, with the addition of the buzzer sounding when a human was detected in the critical action zone. The second set of testing was done to determine if the system was able to detect humans in different zones of the blind spot during a turning scenario. To do that, a reverse pedestrian model was used, which consisted of moving the pedestrian into the "blind spot" of the stationary bus structure to mimic the turning of a bus into a cross street. The recorded results from the reverse pedestrian model were consistent with the data from the stationary human testing in that the system was able to detect pedestrians entering each of the defined zones. Therefore, the pedestrian detection and avoidance system was able to detect a human in the blind spot of a turning bus and alert the driver as intended.

9: Conclusion

In conclusion, the system was effective while detecting an object, determining if that object was a human, and kick starting an effective alert system respective to how close the pedestrian is to the “bus”. The system does not generate a false alarm if the object detected is not a human, which is ideal for a transit bus since there are a number of animate and inanimate objects the bus passes by on the road. Our goal was to only alert the driver if the object is a human within the detection zone, and based on the tests and results in the section above, and the performance of the system, the team feels that the project effectively met the proof of concept requirements for the sensor application of an automated pedestrian detection and alert system. The project as a whole was a success, but there are still steps that can be taken in the future to better solidify the project as a whole and eventually push it towards becoming a system that can be implemented on all buses currently in operation for Transdev. The first steps in furthering the automated pedestrian detection system would be to complete a more fully implemented design. This would require the additional implementation of an accelerometer, temperature sensor, and an IR that could provide more information and control.

The most important of these is the accelerometer. Adding this sensor would provide the ability to track the motion of the bus. For one, this allows the system to determine the axis of motion and make decisions based on how the bus is moving. The system would be able to make more accurate decisions, as it would know whether or not the bus is traveling at slow speeds on standard city streets or at high speeds on the freeway. If there is detected motion in the y direction, which indicates right and left motion for the bus, the system can determine if it is a simple lane shift at higher speeds or a turn during low speeds. At low speeds, having detected motion in the y direction will allow the system to know that it is turning and interpret the other sensor’s data accordingly. As testing is further completed for this application, the motion detection could be calibrated and the code adapted based on the testing data from the accelerometer during right and left hand turns. Knowing what the incoming data from the accelerometer looks like during a left turn, and that it is a problem area for accidents, the system could potentially enter a state of increased awareness, whereas it would be in a more idle state during other movement scenarios. An increased awareness state would mean that the system can both alert the driver and utilize an automated stopping functionality earlier as it knows that there is a higher likelihood of an accident during this maneuver by the bus. Being able to detect that

the bus is traveling at higher speeds would typically indicate that the vehicle is traveling at highway speeds. Knowing that this is the case, the system could be idle during higher speeds so that it is not unnecessarily using power. Conclusively, the implementation of an accelerometer would provide the system the ability to track the motion of the bus and give a further degree of control.

The temperature sensor is another sensor that provides a higher degree of control. Due to the potential of sub-zero or abnormally hot temperatures, while the system utilizes components that have operating temperature restrictions, this sensor would provide a safety measure. If any of the sensors temperature limits are reached, this could provide a functionality so that the decision making logic does not utilize that particular piece of the system, and is instead more aware of data received by sensors within a range of normal operation.

The original infrared sensor that was going to be utilized in the design was that of the Melexis MLX90621. This IR sensor has the functionality to create a heat map. Being able to implement this would allow for a more accurate measurement and image disparity. The current PIR implementation relies exclusively on a detection as to whether or not an object is moving greater than 1 meter per second and has a temperature difference of greater than 4 degrees Celsius from its environment. For the case where a person is standing still, the movement of the subject relative to the bus's movement would trigger the PIR sensor. A potential hole in the design right now would be if an individual is standing still and the bus is moving extraordinarily slow. The MLX90621 has the ability of recording the temperature of an object. This greater degree of control makes it so that the system can determine not only the temperature difference between an object and its environment, but also determine if a detected object is human, other animal, or a heated inanimate object. The MLX90621 could also be cascaded with the PIR sensor, as the PIR sensor has a greater detection range and could provide another layer of detection for the system.

One of the biggest objectives for the future of this system would be to integrate the electrical and sensor computing alongside the mechanical turning and braking of a bus. This is a part of the overall design that is out of the scope of the current team's qualifications as a group of electrical and computer engineers. Turning and braking would have to be implemented based on the location of an individual detected and based on the severity and likeliness of impact. This would also require a significant amount of programming in conjunction with the mechanical

implementation to coordinate the decision making based on the digital signal, so that it could have the appropriate mechanical operation. This project, as proof of concept, provided the correct implementation for a simulated design of a bus. The calibrations and rigging for the sensor system would have to be incorporated into a real bus, so that it is able to receive accurate and calibrated readings as well as protect the system. The power would also have to be implemented to correctly run off of the alternator of the bus at whichever output it is able to provide. Issues could occur on this front, as the power signal needs to be stable, which may require various converters to ensure power consistency as the source from the bus could have a potentially unclean signal. All of this would need to be taken into account, and appropriately overcome so that the system could be brought to a larger scale. The Zedboard is also a test and development board. The system on the bus would likely benefit from a custom circuit design and FPGA specifically for this one sole purpose. This would cut down power consumption as only the necessary functionality would be utilized.

In terms of the alert system, there is also potential for improvement on this front. The LeddarVu8 ships with an included Windows application that shows the eight detection zones as well as where the closest object is located in these zones. A similar information panel could be implemented on a displayed screen with the color of the warning. This would make it so that the simple color information is cascaded with the eight segments shown, providing a greater degree of information to the bus driver. This addition would not be a major distraction either, as it could be designed to be a basic and easy to read configuration that maintains the three colors for warnings alongside the eight detection zones.

Overall, the future goals for this project would be for it to meet its full potential that was out of reach given the time constraints for this one team, and for the system to be expanded to a stage that is capable of fully automated sensing and accident avoidance for the bus.

Appendix A: Lidar Control Modules

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//
// Company: Worcester Polytechnic Institute
// Engineers: Dario Martinovic, Michael Padberg
//
// Create Date: 11/19/2017 12:59:46 PM
// Module Name: pl_lidar_top_module
//
// Description: Top module to control passing of data and I/O between lidar
//              controller and higher control modules.
//////////////////////////////////////////////////////////////////////////////////


module pl_lidar_top_module(

    //Physical board I/O
    input fpga_CLK, //Zedboard clock
    input reset, //BTND
    output CS_N_L,  //chip select line
    output MOSI_L,  //Master out slave in data line
    output clk_1M_L,   //Clocking signal to lidar device
    input MISO_L,   //Master in slave out data line
    input start_lidar_btn, //BTNU

    //Outputs for PS connection over AXI.
    output [31:0] distance_scale_out,
    output [31:0] segment7, //distance reading for segments 7-0
    output [31:0] segment6,
    output [31:0] segment5,
    output [31:0] segment4,
    output [31:0] segment3,
    output [31:0] segment2,
    output [31:0] segment1,
    output [31:0] segment0,
    output [31:0] seg_nums  //segment nums corresponding to distance
readings.
                            //seg_nums[2:0] corresponds to segment0 distance
reading, etc.

    );

    wire [31:0] scale_received_data;    //Scale data from lidar control
module

    //Segment distances - raw data from lidar control module
    wire [34:0] segment_7,      //MSB byte = Distance MSB byte, LSB byte =
Segment #
                segment_6,
                segment_5,
                segment_4,
                segment_3,
                segment_2,
```

```
                segment_1,
                segment_0;

    //distance scale value passed to PS over AXI for distance conversion to
meters
    assign distance_scale_out =
{scale_received_data[7:0],scale_received_data[15:8],scale_received_data[23:16
],scale_received_data[31:24]};

    //individual segment distance readings passed to PS over AXI. Raw sample
data.
    assign segment7 =
{segment_7[10:3],segment_7[18:11],segment_7[26:19],segment_7[34:27]};
    assign segment6 =
{segment_6[10:3],segment_6[18:11],segment_6[26:19],segment_6[34:27]};
    assign segment5 =
{segment_5[10:3],segment_5[18:11],segment_5[26:19],segment_5[34:27]};
    assign segment4 =
{segment_4[10:3],segment_4[18:11],segment_4[26:19],segment_4[34:27]};
    assign segment3 =
{segment_3[10:3],segment_3[18:11],segment_3[26:19],segment_3[34:27]};
    assign segment2 =
{segment_2[10:3],segment_2[18:11],segment_2[26:19],segment_2[34:27]};
    assign segment1 =
{segment_1[10:3],segment_1[18:11],segment_1[26:19],segment_1[34:27]};
    assign segment0 =
{segment_0[10:3],segment_0[18:11],segment_0[26:19],segment_0[34:27]};
    assign seg_nums = {8'b0, segment_7[2:0],
segment_6[2:0],segment_5[2:0],segment_4[2:0],segment_3[2:0],segment_2[2:0],se
gment_1[2:0],segment_0[2:0]};

    //Lidar control logic instance
    lidar_controller lidar_controller_inst (
    .reset(reset),
    .fpga_CLK(fpga_CLK),
    .MISO(MISO_L),
    .CS_N(CS_N_L),
    .clk_1M_L(clk_1M_L),
    .MOSI(MOSI_L),
    .start_lidar_btn(start_lidar_btn),
    .scale_received_data(scale_received_data),
    .segment_7(segment_7),
    .segment_6(segment_6),
    .segment_5(segment_5),
    .segment_4(segment_4),
    .segment_3(segment_3),
    .segment_2(segment_2),
    .segment_1(segment_1),
    .segment_0(segment_0)
    );

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////
// Company: Worcester Polytechnic Institute
// Engineers: Dario Martinovic, Michael Padberg
//
// Create Date: 11/19/2017 01:14:27 PM
// Module Name: lidar_controller
//
// Description:  Programmable logic based state machine for controlling
//                       data transmission to from LeddarTech LeddarVu8.
//////////////////////////////////////////////////////////////////////////////////
//////


module lidar_controller(

    //Physical board I/O
    input reset,    //BTND
    input fpga_CLK, //Zedboard clock
    input MISO,     //Master in slave out data line
    input start_lidar_btn, //BTNU
    output CS_N,    //Chip select lane
    output clk_1M_L,   //Clock output to lidar device
    output MOSI,        //Master out slave in

    //Data output to PS over Axi
    output reg [31:0] scale_received_data,  //scale data to PS
    output reg [34:0] segment_7, //MSB byte = Distance MSB byte, LSB byte =
Segment #
                      segment_6,
                      segment_5,
                      segment_4,
                      segment_3,
                      segment_2,
                      segment_1,
                      segment_0
    );

    //state machine
    reg [3:0] current_state,
              next_state;

    //Parameters for state labels
    parameter [3:0]
              //read distance scale states
              state_start = 4'b0000,
              state_load_scale = 4'b0001,
              state_scale_command = 4'b0010,
              state_delay_scale = 4'b0011,
              state_reading_scale = 4'b0100,
              state_store_scale = 4'b0101,
              //read data states
              state_wait = 4'b0110,
              state_load = 4'b0111,
              state_read_command = 4'b1000,
              state_reading = 4'b1001,
```

```
              state_delay = 4'b1010,
              state_store_reading = 4'b1011;
             // state_stop = 4'b1011;

    //Lidar control opcodes for commands
    parameter [7:0] LIDAR_READ_OPCODE = 8'h0B,
                    LIDAR_WRITE_OPCODE = 8'h02,
                    LIDAR_CE_OPCODE = 8'hC7;
    //LeddarVu8 memory bank base addresses
    parameter [23:0] BANK_0_START_ADDRESS = 24'h000000,  //Configuration Data
               BANK_5_START_ADDRESS = 24'h400000,  //Device Information and
constants
               BANK_13_START_ADDRESS = 24'h500000, //Detection list
               BANK_19_START_ADDRESS = 24'hFFFB00; //Transaction
configuration

    //Offsets applied to memory bank addresses for data access
    parameter [23:0] FPGA_Version_Offset = 24'h120; //Register offset of FPGA
Version in Bank 5
    parameter [23:0] Detection_List_Array_Offset = 24'hc; //Register offset
of Detection List Array in Bank 13
    parameter [23:0] Distance_Scale_Offset = 24'h162; // Register offset of
Distance Scale in Bank 5

    //used to generate 1kHz clock
    reg [16:0] count_100M_1K; //count from 0 to 99999
    parameter [16:0] MAXIMUM_COUNT_1K = 100000;
    wire clk_enable_1K;


     //used to generate 2Hz clock
    reg [25:0] count_100M_2;
    parameter [25:0] MAXIMUM_COUNT_2 = 26'd50000000;
    wire clk_enable_2;


     //used to generate 1MHz clock
    parameter [6:0] MAXIMUM_COUNT_1M = 7'd100;
    reg [6:0] count100M_1M;
    wire clk_1M_posedge;
    reg clk_1M_enabled = 1'b1;

    //send buffer used to send data out on MOSI pin
    reg [47:0] send_buffer;
    reg [7:0] count_shifts_W = 8'h00;


    //receive buffer used to load in lidar response
    reg [767:0] receive_buffer;
    reg [31:0] scale_receive_buffer;
    reg [9:0] count_shifts_R = 10'h0;
    reg [6:0] scale_count_shifts_R = 7'b0;


    //sequential logic for FSM
    always @ (posedge fpga_CLK, negedge reset) begin
       if(reset == 1'b0)begin
```

```
            current_state <= state_start;
        end
        else begin
            current_state <= next_state;
        end
    end

    //combinational next state logic
    always @
(current_state,clk_enable_1K,count_shifts_W,count_shifts_R,scale_count_shifts
_R,clk_enable_2,start_lidar_btn)begin
         case(current_state)
             state_start: begin
                 if(start_lidar_btn)     //wait for start button signal
                     next_state = state_load_scale;
                 else
                     next_state = state_start;
             end
             state_load_scale: begin
                 next_state = state_scale_command;
             end
             state_scale_command: begin
                 if(count_shifts_W == 8'h30)
                     next_state = state_delay_scale;
                 else
                     next_state = state_scale_command;
             end
             state_delay_scale: begin
                 if(clk_enable_1K)
                     next_state = state_reading_scale;
                 else
                     next_state = state_delay_scale;
             end
             state_reading_scale: begin
                 if(scale_count_shifts_R == 7'h20)   //hex 20 = 32
decimal.  Count in 32 bits.
                     next_state = state_store_scale;
                 else
                     next_state = state_reading_scale;
             end
             state_store_scale: begin
                     next_state = state_wait;
             end
             state_wait: begin   //wait for enable signal from counter
                 if(clk_enable_2) //acquire_distance_data
                     next_state = state_load;
                 else
                     next_state = state_wait;
             end

             state_load:begin
                 next_state = state_read_command;
             end

             state_read_command:begin
                 if(count_shifts_W == 8'h30)
                     next_state = state_delay;
```

```
                else
                    next_state = state_read_command;
            end

            state_delay:begin
                if(clk_enable_1K)  //1ms has elapsed
                    next_state = state_reading;
                else
                    next_state = state_delay;
            end

            state_reading:begin
                if(count_shifts_R == 12'h300) //hex 300 = 768 decimal.  Count
in 768 bits.
                    next_state = state_store_reading;
                else
                    next_state = state_reading;
            end

            state_store_reading:begin
                next_state = state_wait;
            end

            default:
                next_state = state_wait;
        endcase
    end

    //individual state logic
    always @ (posedge fpga_CLK) begin
        case(current_state)
            state_start: begin  //first state, wait for start button signal
                count_shifts_W <= 8'b0;
                count_shifts_R <= 10'b0;
                scale_count_shifts_R <= 7'b0;
                clk_1M_enabled <= 1'b1;
                send_buffer <= 48'b0;
                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                receive_buffer <= 768'b0;

                segment_7 <= segment_7;
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end
            state_load_scale: begin     //load scale read command into send
buffer
                clk_1M_enabled <= 1'b1; // enable clk_1M
                send_buffer <= {LIDAR_READ_OPCODE,BANK_5_START_ADDRESS |
Distance_Scale_Offset,16'h0004};
                //send_buffer <= {LIDAR_READ_OPCODE,BANK_5_START_ADDRESS |
FPGA_Version_Offset,16'h0020};
```

```
                count_shifts_W <= 8'b0;
                count_shifts_R <= 10'b0;
                scale_count_shifts_R <= 7'b0;

                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                receive_buffer <= 768'b0;

                segment_7 <= segment_7;
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end
            state_scale_command: begin  //send scale read command
                if(clk_1M_posedge) begin
                    count_shifts_W <= count_shifts_W + 8'b1;
                    send_buffer <= {send_buffer[46:0], 1'b0};
                end
                else begin
                    count_shifts_W <= count_shifts_W;
                    send_buffer <= send_buffer;

                end

                clk_1M_enabled <= 1'b1;
                count_shifts_R <= 10'b0;
                scale_count_shifts_R <= 7'b0;
                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                receive_buffer <= 768'b0;

                segment_7 <= segment_7;
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end
            state_delay_scale: begin    //delay between command and data
reading
                clk_1M_enabled <= 1'b0; //disable clock output
                count_shifts_W <= 8'b0;
                count_shifts_R <= 10'b0;
                scale_count_shifts_R <= 7'b0;
                send_buffer <= 48'b0;
                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                receive_buffer <= 768'b0;

                segment_7 <= segment_7;
                segment_6 <= segment_6;
```

```
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end
            state_reading_scale: begin  //read in lidar response
                if(clk_1M_posedge) begin
                    scale_count_shifts_R <= scale_count_shifts_R + 8'b1;
                    scale_receive_buffer <= {scale_receive_buffer[30:0],
MISO}; //receive and shift left
                end
                else begin
                    scale_count_shifts_R <= scale_count_shifts_R;
                    scale_receive_buffer <= scale_receive_buffer;
                end

                clk_1M_enabled <= 1'b1;
                count_shifts_W <= 8'b0;
                count_shifts_R <= 10'b0;
                send_buffer <= 48'b0;
                scale_received_data <= scale_received_data;
                receive_buffer <= 768'b0;

                segment_7 <= segment_7;
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end
            state_store_scale: begin    //store the scale value
                clk_1M_enabled <= 1'b1;
                scale_receive_buffer <= scale_receive_buffer;
                scale_received_data <= scale_receive_buffer[31:0];
                count_shifts_W <= 8'b0;
                count_shifts_R <= 10'b0;
                send_buffer <= 48'b0;
                scale_count_shifts_R <= 7'b0;
                receive_buffer <= 768'b0;
                segment_7 <= segment_7;
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end

            state_load: begin   //load the distance reading command into send
buffer
//                send_buffer <= {LIDAR_READ_OPCODE,BANK_5_START_ADDRESS |
FPGA_Version_Offset,16'h0020};
```

```
                send_buffer <= {LIDAR_READ_OPCODE,BANK_13_START_ADDRESS |
Detection_List_Array_Offset,16'h0060};
                clk_1M_enabled <= 1'b1; // enable clk_1M
                count_shifts_W <= 8'b0;
                count_shifts_R <= 10'b0;
                scale_count_shifts_R <= 7'b0;
                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                receive_buffer <= 768'b0;

                segment_7 <= segment_7;
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end
            state_read_command: begin   //send distance read command
                if(clk_1M_posedge) begin    //bit shifted on each positive 1M
clock edge
                    count_shifts_W <= count_shifts_W + 8'b1;
                    send_buffer <= {send_buffer[46:0], 1'b0};
                end
                else begin
                    count_shifts_W <= count_shifts_W;
                    send_buffer <= send_buffer;
                end

                clk_1M_enabled <= 1'b1;
                count_shifts_R <= 10'b0;
                scale_count_shifts_R <= 7'b0;
                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                receive_buffer <= 768'b0;

                segment_7 <= segment_7;
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end
            state_delay:begin   //delay between command and data reading
                clk_1M_enabled <= 1'b0; //disable clk_1M
                count_shifts_R <= 10'b0;
                scale_count_shifts_R <= 7'b0;
                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                count_shifts_W <= 8'b0;
                send_buffer <= 48'b0;
                receive_buffer <= 768'b0;

                segment_7 <= segment_7;
```

```
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end
            state_reading:begin //read in segment distance data from lidar
                clk_1M_enabled <= 1'b1;
                scale_count_shifts_R <= 7'b0;
                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                count_shifts_W <= 8'b0;
                send_buffer <= 48'b0;

                segment_7 <= segment_7;
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;

                if(clk_1M_posedge) begin
                    count_shifts_R <= count_shifts_R + 8'b1;
                    receive_buffer <= {receive_buffer[766:0], MISO};
//receive and shift left
                end

                else begin
                    count_shifts_R <= count_shifts_R;
                    receive_buffer <= receive_buffer;
                end
            end
            state_store_reading: begin  //store the distance readings
                clk_1M_enabled <= 1'b1;
                scale_count_shifts_R <= 7'b0;
                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                count_shifts_W <= 8'b0;
                count_shifts_R <= 10'b0;
                send_buffer <= 48'b0;
                receive_buffer <= receive_buffer;
                //received data is parsed for each segment.
                segment_7 <=
{receive_buffer[767:736],receive_buffer[698:696]};
                segment_6 <=
{receive_buffer[671:640],receive_buffer[602:600]};
                segment_5 <=
{receive_buffer[575:544],receive_buffer[506:504]};
                segment_4 <=
{receive_buffer[479:448],receive_buffer[410:408]};
                segment_3 <=
{receive_buffer[383:352],receive_buffer[314:312]};
                segment_2 <=
```

```
{receive_buffer[287:256],receive_buffer[218:216]};
                segment_1 <=
{receive_buffer[191:160],receive_buffer[122:120]};
                segment_0 <= {receive_buffer[95:64],receive_buffer[26:24]};
            end
            state_wait:begin    //wait for enable signal from counter
                clk_1M_enabled <= 1'b1;
                count_shifts_W <= 8'b0;
                count_shifts_R <= 10'b0;
                receive_buffer <= 768'b0;
                send_buffer <= 48'b0;
                scale_receive_buffer <= 32'b0;
                scale_received_data <= scale_received_data;
                scale_count_shifts_R <= 7'b0;

                segment_7 <= segment_7;
                segment_6 <= segment_6;
                segment_5 <= segment_5;
                segment_4 <= segment_4;
                segment_3 <= segment_3;
                segment_2 <= segment_2;
                segment_1 <= segment_1;
                segment_0 <= segment_0;
            end


        endcase
    end

    assign MOSI = send_buffer[47];  //serial output is sent out from MSB of
send buffer.

    //chip select is pulled low during command & data transmission.
    assign CS_N = !(current_state == state_read_command || current_state ==
state_delay || current_state == state_reading ||
                    current_state == state_scale_command || current_state ==
state_delay_scale || current_state == state_reading_scale );

    //creates a 1KHz clock
    always @ (posedge fpga_CLK) begin
        if (count_100M_1K == MAXIMUM_COUNT_1K - 1)
            count_100M_1K <= 0;
        else
            count_100M_1K <= count_100M_1K + 1'b1;
    end

    assign clk_enable_1K = (count_100M_1K == MAXIMUM_COUNT_1K - 1); //clocked
at 1KHz

   //creates a 2Hz clock
    always @ (posedge fpga_CLK) begin
        if (count_100M_2 == MAXIMUM_COUNT_2 - 1)
            count_100M_2 <= 0;
        else
            count_100M_2 <= count_100M_2 + 1'b1;
    end
```

```
    assign clk_enable_2 = (count_100M_2 == MAXIMUM_COUNT_2 - 1); //clocked at
2Hz


    //creates a 1MHz clock
    always @ (posedge fpga_CLK) begin
        if (count100M_1M == MAXIMUM_COUNT_1M - 4'b1)
            count100M_1M <= 0;
        else
            count100M_1M <= count100M_1M + 4'b1;
    end

    assign clk_1M_L = (count100M_1M < 5) && clk_1M_enabled;  //50% duty cycle
clock at 1MHz rate
    assign clk_1M_posedge = (count100M_1M == 0);

endmodule



`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////
/////
// Company: Worcester Polytechnic Institute
// Engineers: Dario Martinovic, Michael Padberg
//
// Create Date: 02/06/2018 09:03:32 PM
// Module Name: LED_warning
//
// Description: A module created to display data on the eight zedboard LEDs.
//              LEDs indicate presence of objects within danger warning zone
//////////////////////////////////////////////////////////////////////////////
/////


module LED_warning(
    input clk,
    input [7:0] LEDdata,    //eight bit input for eight leds
    output reg [7:0] LEDs   //LED outputs
    );

    always @ (posedge clk) begin

        LEDs <= LEDdata;    //assign 8 input 1 or 0 to LEDs.
    end

endmodule
```

Appendix B: AXI Protocol Files – Standard Xilinx IP Files Modified for Integration with Lidar Controller Logic

```
`timescale 1 ns / 1 ps

	module lidar_axi_v1_0 #
	(
		// Users to add parameters here

		// User parameters ends
		// Do not modify the parameters beyond this line


		// Parameters of Axi Slave Bus Interface S00_AXI
		parameter integer C_S00_AXI_DATA_WIDTH	= 32,
		parameter integer C_S00_AXI_ADDR_WIDTH	= 6
	)
	(
		// Users to add ports here
	  input fpga_CLK,
	  input reset, //BTND
	  output CS_N_L,
	  output MOSI_L,
	  output clk_1M_L,
	  input MISO_L,
	  input start_lidar_btn, //BTNU
	  output [7:0] LEDs,

	  //-----------IR-----------------
	  input [1:0] data_IR,
	  input reset_IR_N,
	  input clk_25M,
	  output sound,
	  output [11:0] clr_rgb_VGA,
	  output HS_VGA,
	  output VS_VGA,
	  //-------------------------------

		// User ports ends
		// Do not modify the ports beyond this line


		// Ports of Axi Slave Bus Interface S00_AXI
		input wire  s00_axi_aclk,
		input wire  s00_axi_aresetn,
		input wire [C_S00_AXI_ADDR_WIDTH-1 : 0] s00_axi_awaddr,
		input wire [2 : 0] s00_axi_awprot,
		input wire  s00_axi_awvalid,
		output wire  s00_axi_awready,
		input wire [C_S00_AXI_DATA_WIDTH-1 : 0] s00_axi_wdata,
		input wire [(C_S00_AXI_DATA_WIDTH/8)-1 : 0] s00_axi_wstrb,
		input wire  s00_axi_wvalid,
		output wire  s00_axi_wready,
		output wire [1 : 0] s00_axi_bresp,
		output wire  s00_axi_bvalid,
```

```
            input wire  s00_axi_bready,
            input wire [C_S00_AXI_ADDR_WIDTH-1 : 0] s00_axi_araddr,
            input wire [2 : 0] s00_axi_arprot,
            input wire  s00_axi_arvalid,
            output wire  s00_axi_arready,
            output wire [C_S00_AXI_DATA_WIDTH-1 : 0] s00_axi_rdata,
            output wire [1 : 0] s00_axi_rresp,
            output wire  s00_axi_rvalid,
            input wire  s00_axi_rready
      );
// Instantiation of Axi Bus Interface S00_AXI
      lidar_axi_v1_0_S00_AXI # (
            .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),
            .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH)
      ) lidar_axi_v1_0_S00_AXI_inst (
            .S_AXI_ACLK(s00_axi_aclk),
            .S_AXI_ARESETN(s00_axi_aresetn),
            .S_AXI_AWADDR(s00_axi_awaddr),
            .S_AXI_AWPROT(s00_axi_awprot),
            .S_AXI_AWVALID(s00_axi_awvalid),
            .S_AXI_AWREADY(s00_axi_awready),
            .S_AXI_WDATA(s00_axi_wdata),
            .S_AXI_WSTRB(s00_axi_wstrb),
            .S_AXI_WVALID(s00_axi_wvalid),
            .S_AXI_WREADY(s00_axi_wready),
            .S_AXI_BRESP(s00_axi_bresp),
            .S_AXI_BVALID(s00_axi_bvalid),
            .S_AXI_BREADY(s00_axi_bready),
            .S_AXI_ARADDR(s00_axi_araddr),
            .S_AXI_ARPROT(s00_axi_arprot),
            .S_AXI_ARVALID(s00_axi_arvalid),
            .S_AXI_ARREADY(s00_axi_arready),
            .S_AXI_RDATA(s00_axi_rdata),
            .S_AXI_RRESP(s00_axi_rresp),
            .S_AXI_RVALID(s00_axi_rvalid),
            .S_AXI_RREADY(s00_axi_rready),
            .reset(reset), //BTND
        .fpga_CLK(fpga_CLK),
        .CS_N_L(CS_N_L),
        .MOSI_L(MOSI_L),
        .clk_1M_L(clk_1M_L),
        .MISO_L(MISO_L),
        .start_lidar_btn(start_lidar_btn),  //BTNU
        .LEDs(LEDs),
          //-----------IR-----------------
        .data_IR(data_IR),
        .reset_IR_N(reset_IR_N),
        .clk_25M(clk_25M),
        .sound(sound),
        .clr_rgb_VGA(clr_rgb_VGA),
        .HS_VGA(HS_VGA),
        .VS_VGA(VS_VGA)
        //------------------------------

      );

      // Add user logic here
```

```
	// User logic ends

	endmodule



`timescale 1 ns / 1 ps

	module lidar_axi_v1_0_S00_AXI #
	(
		// Users to add parameters here

		// User parameters ends
		// Do not modify the parameters beyond this line

		// Width of S_AXI data bus
		parameter integer C_S_AXI_DATA_WIDTH	= 32,
		// Width of S_AXI address bus
		parameter integer C_S_AXI_ADDR_WIDTH	= 6
	)
	(
		// Users to add ports here
	  input fpga_CLK,

	  input wire [31:0] distance_scale_out,
	  input wire [31:0] segment7,
	  input wire [31:0] segment6,
	  input wire [31:0] segment5,
	  input wire [31:0] segment4,
	  input wire [31:0] segment3,
	  input wire [31:0] segment2,
	  input wire [31:0] segment1,
	  input wire [31:0] segment0,
	  input wire [31:0] seg_nums,

	  input reset, //BTND
	  output CS_N_L,
	  output MOSI_L,
	  output clk_1M_L,
	  input MISO_L,
	  input start_lidar_btn, //BTNU
	  output [7:0] LEDs,


	  //-----------IR-----------------
	  input [1:0] data_IR,
	  input reset_IR_N,
	  input clk_25M,
	  output sound,
	  output [11:0] clr_rgb_VGA,
	  output HS_VGA,
	  output VS_VGA,
	  //--------------------------------
```

```
		// User ports ends
		// Do not modify the ports beyond this line

		// Global Clock Signal
		input wire  S_AXI_ACLK,
		// Global Reset Signal. This Signal is Active LOW
		input wire  S_AXI_ARESETN,
		// Write address (issued by master, acceped by Slave)
		input wire [C_S_AXI_ADDR_WIDTH-1 : 0] S_AXI_AWADDR,
		// Write channel Protection type. This signal indicates the
		// privilege and security level of the transaction, and whether
		// the transaction is a data access or an instruction access.
		input wire [2 : 0] S_AXI_AWPROT,
		// Write address valid. This signal indicates that the master
signaling
		// valid write address and control information.
		input wire  S_AXI_AWVALID,
		// Write address ready. This signal indicates that the slave is
ready
		// to accept an address and associated control signals.
		output wire  S_AXI_AWREADY,
		// Write data (issued by master, acceped by Slave)
		input wire [C_S_AXI_DATA_WIDTH-1 : 0] S_AXI_WDATA,
		// Write strobes. This signal indicates which byte lanes hold
		// valid data. There is one write strobe bit for each eight
		// bits of the write data bus.
		input wire [(C_S_AXI_DATA_WIDTH/8)-1 : 0] S_AXI_WSTRB,
		// Write valid. This signal indicates that valid write
		// data and strobes are available.
		input wire  S_AXI_WVALID,
		// Write ready. This signal indicates that the slave
		// can accept the write data.
		output wire  S_AXI_WREADY,
		// Write response. This signal indicates the status
		// of the write transaction.
		output wire [1 : 0] S_AXI_BRESP,
		// Write response valid. This signal indicates that the channel
		// is signaling a valid write response.
		output wire  S_AXI_BVALID,
		// Response ready. This signal indicates that the master
		// can accept a write response.
		input wire  S_AXI_BREADY,
		// Read address (issued by master, acceped by Slave)
		input wire [C_S_AXI_ADDR_WIDTH-1 : 0] S_AXI_ARADDR,
		// Protection type. This signal indicates the privilege
		// and security level of the transaction, and whether the
		// transaction is a data access or an instruction access.
		input wire [2 : 0] S_AXI_ARPROT,
		// Read address valid. This signal indicates that the channel
		// is signaling valid read address and control information.
		input wire  S_AXI_ARVALID,
		// Read address ready. This signal indicates that the slave is
		// ready to accept an address and associated control signals.
		output wire  S_AXI_ARREADY,
		// Read data (issued by slave)
```

```
        output wire [C_S_AXI_DATA_WIDTH-1 : 0] S_AXI_RDATA,
        // Read response. This signal indicates the status of the
        // read transfer.
        output wire [1 : 0] S_AXI_RRESP,
        // Read valid. This signal indicates that the channel is
        // signaling the required read data.
        output wire  S_AXI_RVALID,
        // Read ready. This signal indicates that the master can
        // accept the read data and response information.
        input wire  S_AXI_RREADY
);

// AXI4LITE signals
reg [C_S_AXI_ADDR_WIDTH-1 : 0]      axi_awaddr;
reg   axi_awready;
reg   axi_wready;
reg [1 : 0]       axi_bresp;
reg   axi_bvalid;
reg [C_S_AXI_ADDR_WIDTH-1 : 0]      axi_araddr;
reg   axi_arready;
reg [C_S_AXI_DATA_WIDTH-1 : 0]      axi_rdata;
reg [1 : 0]       axi_rresp;
reg   axi_rvalid;

// Example-specific design signals
// local parameter for addressing 32 bit / 64 bit C_S_AXI_DATA_WIDTH
// ADDR_LSB is used for addressing 32/64 bit registers/memories
// ADDR_LSB = 2 for 32 bits (n downto 2)
// ADDR_LSB = 3 for 64 bits (n downto 3)
localparam integer ADDR_LSB = (C_S_AXI_DATA_WIDTH/32) + 1;
localparam integer OPT_MEM_ADDR_BITS = 3;
//------------------------------------------------
//-- Signals for user logic register space example
//------------------------------------------------
//-- Number of Slave Registers 16
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg0;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg1;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg2;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg3;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg4;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg5;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg6;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg7;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg8;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg9;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg10;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg11;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg12;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg13;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg14;
reg [C_S_AXI_DATA_WIDTH-1:0]  slv_reg15;
wire   slv_reg_rden;
wire   slv_reg_wren;
reg [C_S_AXI_DATA_WIDTH-1:0]   reg_data_out;
integer      byte_index;
reg    aw_en;
```

```
// I/O Connections assignments

assign S_AXI_AWREADY    = axi_awready;
assign S_AXI_WREADY     = axi_wready;
assign S_AXI_BRESP      = axi_bresp;
assign S_AXI_BVALID     = axi_bvalid;
assign S_AXI_ARREADY    = axi_arready;
assign S_AXI_RDATA      = axi_rdata;
assign S_AXI_RRESP      = axi_rresp;
assign S_AXI_RVALID     = axi_rvalid;
// Implement axi_awready generation
// axi_awready is asserted for one S_AXI_ACLK clock cycle when both
// S_AXI_AWVALID and S_AXI_WVALID are asserted. axi_awready is
// de-asserted when reset is low.

always @( posedge S_AXI_ACLK )
begin
  if ( S_AXI_ARESETN == 1'b0 )
    begin
      axi_awready <= 1'b0;
      aw_en <= 1'b1;
    end
  else
    begin
      if (~axi_awready && S_AXI_AWVALID && S_AXI_WVALID && aw_en)
        begin
          // slave is ready to accept write address when
          // there is a valid write address and write data
          // on the write address and data bus. This design
          // expects no outstanding transactions.
          axi_awready <= 1'b1;
          aw_en <= 1'b0;
        end
        else if (S_AXI_BREADY && axi_bvalid)
            begin
              aw_en <= 1'b1;
              axi_awready <= 1'b0;
            end
      else
        begin
          axi_awready <= 1'b0;
        end
    end
end

// Implement axi_awaddr latching
// This process is used to latch the address when both
// S_AXI_AWVALID and S_AXI_WVALID are valid.

always @( posedge S_AXI_ACLK )
begin
  if ( S_AXI_ARESETN == 1'b0 )
    begin
      axi_awaddr <= 0;
    end
  else
    begin
```

```
            if (~axi_awready && S_AXI_AWVALID && S_AXI_WVALID && aw_en)
              begin
                // Write Address latching
                axi_awaddr <= S_AXI_AWADDR;
              end
          end
      end

      // Implement axi_wready generation
      // axi_wready is asserted for one S_AXI_ACLK clock cycle when both
      // S_AXI_AWVALID and S_AXI_WVALID are asserted. axi_wready is
      // de-asserted when reset is low.

      always @( posedge S_AXI_ACLK )
      begin
        if ( S_AXI_ARESETN == 1'b0 )
          begin
            axi_wready <= 1'b0;
          end
        else
          begin
            if (~axi_wready && S_AXI_WVALID && S_AXI_AWVALID && aw_en )
              begin
                // slave is ready to accept write data when
                // there is a valid write address and write data
                // on the write address and data bus. This design
                // expects no outstanding transactions.
                axi_wready <= 1'b1;
              end
            else
              begin
                axi_wready <= 1'b0;
              end
          end
      end

      // Implement memory mapped register select and write logic generation
      // The write data is accepted and written to memory mapped registers
when
      // axi_awready, S_AXI_WVALID, axi_wready and S_AXI_WVALID are asserted.
Write strobes are used to
      // select byte enables of slave registers while writing.
      // These registers are cleared when reset (active low) is applied.
      // Slave register write enable is asserted when valid address and data
are available
      // and the slave is ready to accept the write address and write data.
      assign slv_reg_wren = axi_wready && S_AXI_WVALID && axi_awready &&
S_AXI_AWVALID;

      always @( posedge S_AXI_ACLK )
      begin
        if ( S_AXI_ARESETN == 1'b0 )
          begin
            slv_reg0 <= 0;
            slv_reg1 <= 0;
            slv_reg2 <= 0;
            slv_reg3 <= 0;
```

```
            slv_reg4 <= 0;
            slv_reg5 <= 0;
            slv_reg6 <= 0;
            slv_reg7 <= 0;
            slv_reg8 <= 0;
            slv_reg9 <= 0;
            slv_reg10 <= 0;
            slv_reg11 <= 0;
            slv_reg12 <= 0;
            slv_reg13 <= 0;
            slv_reg14 <= 0;
            slv_reg15 <= 0;
          end
        else begin
          if (slv_reg_wren)
            begin
              case ( axi_awaddr[ADDR_LSB+OPT_MEM_ADDR_BITS:ADDR_LSB] )
                4'h0:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 0
                      slv_reg0[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                    end
                4'h1:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 1
                      slv_reg1[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                    end
                4'h2:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 2
                      slv_reg2[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                    end
                4'h3:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 3
                      slv_reg3[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                    end
```

```
                4'h4:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 4
                      slv_reg4[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                    end
                4'h5:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 5
                      slv_reg5[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                    end
                4'h6:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 6
                      slv_reg6[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                    end
                4'h7:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 7
                      slv_reg7[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                    end
                4'h8:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 8
                      slv_reg8[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                    end
                4'h9:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 9
```

```
                    slv_reg9[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                  end
                4'hA:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 10
                      slv_reg10[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                  end
                4'hB:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 11
                      slv_reg11[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                  end
                4'hC:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 12
                      slv_reg12[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                  end
                4'hD:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 13
                      slv_reg13[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                  end
                4'hE:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
                      // Respective byte enables are asserted as per write
strobes
                      // Slave register 14
                      slv_reg14[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                  end
                4'hF:
                  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-
1; byte_index = byte_index+1 )
                    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
```

```
                    // Respective byte enables are asserted as per write
strobes
                    // Slave register 15
                    slv_reg15[(byte_index*8) +: 8] <=
S_AXI_WDATA[(byte_index*8) +: 8];
                  end
              default : begin
                          slv_reg0 <= slv_reg0;
                          slv_reg1 <= slv_reg1;
                          slv_reg2 <= slv_reg2;
                          slv_reg3 <= slv_reg3;
                          slv_reg4 <= slv_reg4;
                          slv_reg5 <= slv_reg5;
                          slv_reg6 <= slv_reg6;
                          slv_reg7 <= slv_reg7;
                          slv_reg8 <= slv_reg8;
                          slv_reg9 <= slv_reg9;
                          slv_reg10 <= slv_reg10;
                          slv_reg11 <= slv_reg11;
                          slv_reg12 <= slv_reg12;
                          slv_reg13 <= slv_reg13;
                          slv_reg14 <= slv_reg14;
                          slv_reg15 <= slv_reg15;
                        end
            endcase
          end
      end
    end

    // Implement write response logic generation
    // The write response and response valid signals are asserted by the
slave
    // when axi_wready, S_AXI_WVALID, axi_wready and S_AXI_WVALID are
asserted.
    // This marks the acceptance of address and indicates the status of
    // write transaction.

    always @( posedge S_AXI_ACLK )
    begin
      if ( S_AXI_ARESETN == 1'b0 )
        begin
          axi_bvalid  <= 0;
          axi_bresp   <= 2'b0;
        end
      else
        begin
          if (axi_awready && S_AXI_AWVALID && ~axi_bvalid && axi_wready &&
S_AXI_WVALID)
            begin
              // indicates a valid write response is available
              axi_bvalid <= 1'b1;
              axi_bresp  <= 2'b0; // 'OKAY' response
            end                   // work error responses in future
          else
            begin
              if (S_AXI_BREADY && axi_bvalid)
                //check if bready is asserted while bvalid is high)
```

```
                  //(there is a possibility that bready is always asserted
high)
                  begin
                    axi_bvalid <= 1'b0;
                  end
              end
          end
      end

      // Implement axi_arready generation
      // axi_arready is asserted for one S_AXI_ACLK clock cycle when
      // S_AXI_ARVALID is asserted. axi_awready is
      // de-asserted when reset (active low) is asserted.
      // The read address is also latched when S_AXI_ARVALID is
      // asserted. axi_araddr is reset to zero on reset assertion.

      always @( posedge S_AXI_ACLK )
      begin
        if ( S_AXI_ARESETN == 1'b0 )
          begin
            axi_arready <= 1'b0;
            axi_araddr  <= 32'b0;
          end
        else
          begin
            if (~axi_arready && S_AXI_ARVALID)
              begin
                // indicates that the slave has acceped the valid read
address
                axi_arready <= 1'b1;
                // Read address latching
                axi_araddr  <= S_AXI_ARADDR;
              end
            else
              begin
                axi_arready <= 1'b0;
              end
          end
      end

      // Implement axi_arvalid generation
      // axi_rvalid is asserted for one S_AXI_ACLK clock cycle when both
      // S_AXI_ARVALID and axi_arready are asserted. The slave registers
      // data are available on the axi_rdata bus at this instance. The
      // assertion of axi_rvalid marks the validity of read data on the
      // bus and axi_rresp indicates the status of read
transaction.axi_rvalid
      // is deasserted on reset (active low). axi_rresp and axi_rdata are
      // cleared to zero on reset (active low).
      always @( posedge S_AXI_ACLK )
      begin
        if ( S_AXI_ARESETN == 1'b0 )
          begin
            axi_rvalid <= 0;
            axi_rresp  <= 0;
          end
        else
```

```
          begin
            if (axi_aready && S_AXI_ARVALID && ~axi_rvalid)
              begin
                // Valid read data is available at the read data bus
                axi_rvalid <= 1'b1;
                axi_rresp  <= 2'b0; // 'OKAY' response
              end
            else if (axi_rvalid && S_AXI_RREADY)
              begin
                // Read data is accepted by the master
                axi_rvalid <= 1'b0;
              end
          end
      end

      // Implement memory mapped register select and read logic generation
      // Slave register read enable is asserted when valid address is
available
      // and the slave is ready to accept the read address.
      assign slv_reg_rden = axi_aready & S_AXI_ARVALID & ~axi_rvalid;
      always @(*)
      begin
            // Address decoding for reading registers
            case ( axi_araddr[ADDR_LSB+OPT_MEM_ADDR_BITS:ADDR_LSB] )
              4'h0   : reg_data_out <= segment0;    //pass segment distances
              4'h1   : reg_data_out <= segment1;    //0-7 and distance scale
              4'h2   : reg_data_out <= segment2;    //to PS over AXI
              4'h3   : reg_data_out <= segment3;
              4'h4   : reg_data_out <= segment4;
              4'h5   : reg_data_out <= segment5;
              4'h6   : reg_data_out <= segment6;
              4'h7   : reg_data_out <= segment7;
              4'h8   : reg_data_out <= seg_nums;
              4'h9   : reg_data_out <= distance_scale_out;
              4'hA   : reg_data_out <= slv_reg10;   //10 and 11 reserved
              4'hB   : reg_data_out <= slv_reg11;   //for zone detection
              4'hC   : reg_data_out <= slv_reg12;   //signals
              4'hD   : reg_data_out <= slv_reg13;
              4'hE   : reg_data_out <= slv_reg14;
              4'hF   : reg_data_out <= slv_reg15;
              default : reg_data_out <= 0;
            endcase
      end

      // Output register or memory read data
      always @( posedge S_AXI_ACLK )
      begin
        if ( S_AXI_ARESETN == 1'b0 )
          begin
            axi_rdata  <= 0;
          end
        else
          begin
            // When there is a valid read address (S_AXI_ARVALID) with
            // acceptance of read address by the slave (axi_aready),
            // output the read dada
            if (slv_reg_rden)
```

```
              begin
                axi_rdata <= reg_data_out;     // register read data
              end
          end
      end

      // Add user logic here

      //lidar top modue instance
    pl_lidar_top_module pl_lidar_top_module(
        .fpga_CLK(fpga_CLK),
        .reset(reset), //BTND
        .CS_N_L(CS_N_L),
        .MOSI_L(MOSI_L),
        .clk_1M_L(clk_1M_L),
        .MISO_L(MISO_L),
        .start_lidar_btn(start_lidar_btn), //BTNU
        .distance_scale_out(distance_scale_out),
        .segment7(segment7),
        .segment6(segment6),
        .segment5(segment5),
        .segment4(segment4),
        .segment3(segment3),
        .segment2(segment2),
        .segment1(segment1),
        .segment0(segment0),
        .seg_nums(seg_nums)
        );

    //LED warning instance
    LED_warning LED_warning_inst (.clk(S_AXI_ACLK), .LEDdata(slv_reg10[7:0]),
.LEDs(LEDs));

    //IR control top module instance
    IR_top IR_top_inst(.IR_data(data_IR),   //input from IR sensor (HIGH if
there is a detection)
        .AXI_slv_reg10(slv_reg10[7:0]),     // Danger warning detection
        .AXI_slv_reg11(slv_reg11[7:0]),     // Critical action detection
        .sound(sound),                      //output to speaker/buzzer
        .vga_reset_N(reset_IR_N),           // input active low reset coming
from PS
        .clr_rgb_top(clr_rgb_VGA),         // output that define the color
display of the VGA
        .HS_top(HS_VGA),             // Hsync output to the VGA
        .VS_top(VS_VGA),             // Vsync output to the VGA
        .clk_25M(clk_25M),         // input 25 MHz
        .clk_100M(fpga_CLK)          //input 100MHz - FPGA/AXI Clock
    );

      // User logic ends

      endmodule
```

Appendix C: Infrared Sensor, VGA Controller, Buzzer Controller (VGA .vhd File from Digilent, Inc.)

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////
// Company: Worcester Polytechnic Institute
// Engineer: Kazim Shaikh
// Integration by: Dario Martinovic, Michael Padberg
//
// Create Date: 01/21/2018
// Module Name: IR_top
//
// Description: Top module for controlling IR sensor integration with VGA
//              color display and sound buzzer.
//
//////////////////////////////////////////////////////////////////////////////////
//////


module IR_top(


    input [1:0] IR_data,               //input from IR sensor (HIGH if there
is a detection)
    //data loaded from ps AXI transmission
    input [7:0] AXI_slv_reg10,      //high if object is in danger warning
zone
    input [7:0] AXI_slv_reg11,      //high if object is in critical action
zone
    output sound,                   //output to a speaker/buzzer
    output reg [1:0] color_signal,  //color selection for color_logic_inst

    //VGA controller signals
    input vga_reset_N,           // active low reset coming from PS
    output [11:0] clr_rgb_top, // takes values that define the color display
of the VGA
    output HS_top,             // Hsync output to the VGA
    output VS_top,             // Vsync output to the VGA

    //input clock signals
    input clk_25M,         // 25 MHz from MMCM
    input clk_100M          //100MHz - FPGA/AXI Clock

);

    wire blank_wire;           // a blank signal to the VGA

    wire [10:0] hcount_wire;   // hcount output to the VGA
    wire [10:0] vcount_wire;   // vcount output to the VGA

    reg [21:0] tone;           // tone for buzzer

    reg IR_detection;          // Combined data from two IRs
```

```
    //detection for sound output determined from IR & Lidar
    always @ (posedge clk_100M) begin
        IR_detection <= IR_data[0] | IR_data[1];  //Two IR sensor signals
anded for single detection
    end

    always @ (AXI_slv_reg10, AXI_slv_reg11, IR_detection, tone) begin
        if((AXI_slv_reg11 > 0) && IR_detection) begin   //if object in
critical zone, IR's detect person
            color_signal = 2'b10;   // show red
            tone = 22'd30000; //play sound
        end
        else if (AXI_slv_reg10 > 0) begin       //if object is in danger
warning zone
            color_signal = 2'b01;   //show yellow
            tone = 22'd0000;
        end
        else begin
            color_signal = 2'b00;   //show green
            tone = 22'd0000;
        end
    end

    // Frequency Divider
    tone_generator tone_generator_inst( .clk(clk_100M),
        .counter(tone),
        .sound(sound)
    );

    // This module is creating outputs that will be manipulated by the color
logic module to display
    // graphics on the monitor.  VGA controller is Digilent IP.
    vga_controller_640_60 in3 (.pixel_clk(clk_25M),
                               .rst(vga_reset_N),
                               .HS(HS_top),
                               .VS(VS_top),
                               .hcount(hcount_wire),
                               .vcount(vcount_wire),
                               .blank(blank_wire));

    //RGB color logic for VGA port
    color_logic color_logic_inst (.blank(blank_wire),
                                  .vcount(vcount_wire),
                                  .color_signal(color_signal),
                                  .hcount(hcount_wire),
                                  .clr_rgb(clr_rgb_top));


endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////
// Company: Worcester Polytechnic Institute
// Engineer: Kazim Shaikh
// Edited by: Dario Martinovic
//
// Create Date: 01/21/2018
// Module Name: tone_generator
//
// Description: Module to generate sound output on buzzer when critical
action
//              should be taken.  Person in critical action zone.
//
//////////////////////////////////////////////////////////////////////////////////
//////

module tone_generator(
    input clk,  //clock input
    input [21:0] counter,   //count signal
    output reg sound    //output sound
    );

    wire sound_next;
    wire sound_invert;

    reg [21:0] num;
    wire [21:0] next_num;

    always@(posedge clk) begin
        if(num == counter)
        begin
            sound <= sound_invert;
            num <= 22'd0;
        end
        else
        begin
            sound <= sound_next;
            num <= next_num;
        end
    end

    assign next_num = num + 22'b1;
    assign sound_next = sound;
    assign sound_invert = ~sound;

endmodule




`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//////
// Company: Worcester Polytechnic Institute
```

```
// Engineer: Kazim Shaikh
// Edited By: Dario Martinovic, Michael Padberg
//
// Create Date: 01/21/2018
// Module Name: color_logic
//
// Description: This module takes in five input signals from the vga
controller
//              and gives an output to the VGA monitor.
//
//
//////////////////////////////////////////////////////////////////////////////
/////

module color_logic(
input blank,                // input from vga_controller. It determines if
pixels are visible on the screen
input [10:0] vcount ,       // input from vga_controller. It describes the
vertical location of a pixel from 0 to 479.
input [1:0] color_signal,   // input from IR_top for color display based on
object position.
input [10:0] hcount ,       // input from vga_controller. It describes the
horizontal location of a pixel from 0 to 649.
output reg [11:0] clr_rgb   // output describing red, green and blue
    );

//Color parameters
parameter YELLOW = 12'hF0F;
parameter RED = 12'h00F;
parameter BLACK = 12'h000;
parameter WHITE = 12'hFFF;
parameter GREEN = 12'hF00;


//display a color to fill the screen based on the input from IR_top
always @ (blank, color_signal, clr_rgb) begin
    if (blank == 0) begin
        case(color_signal)
            2'b01 : clr_rgb = YELLOW;
            2'b10 : clr_rgb = RED;
            default: clr_rgb =  GREEN;
        endcase
    end
    else begin
        clr_rgb = BLACK;    //if blank signal =1, display no color.
    end
end

endmodule
```

*****Vga_controller_640_60.vhd file from Digilent, Inc.*****

```
----------------------------------------------------------------------------
-- vga_controller_640_60.vhd
----------------------------------------------------------------------------
-- Author : Ulrich Zoltán
--          Copyright 2006 Digilent, Inc.
----------------------------------------------------------------------------
-- Software version : Xilinx ISE 7.1.04i
--                    WebPack
-- Device           : 3s200ft256-4
----------------------------------------------------------------------------
-- This file contains the logic to generate the synchronization signals,
-- horizontal and vertical pixel counter and video disable signal
-- for the 640x480@60Hz resolution.
----------------------------------------------------------------------------
--  Behavioral description
----------------------------------------------------------------------------
-- Please read the following article on the web regarding the
-- vga video timings:
-- http://www.epanorama.net/documents/pc/vga_timing.html

-- This module generates the video synch pulses for the monitor to
-- enter 640x480@60Hz resolution state. It also provides horizontal
-- and vertical counters for the currently displayed pixel and a blank
-- signal that is active when the pixel is not inside the visible screen
-- and the color outputs should be reset to 0.

-- timing diagram for the horizontal synch signal (HS)
-- 0                          648    744           800 (pixels)
-- ------------------------|______|-----------------
-- timing diagram for the vertical synch signal (VS)
-- 0                                  482    484  525 (lines)
-- ----------------------------------|______|-------

-- The blank signal is delayed one pixel clock period (40ns) from where
-- the pixel leaves the visible screen, according to the counters, to
-- account for the pixel pipeline delay. This delay happens because
-- it takes time from when the counters indicate current pixel should
-- be displayed to when the color data actually arrives at the monitor
-- pins (memory read delays, synchronization delays).
----------------------------------------------------------------------------
--  Port definitions
----------------------------------------------------------------------------
-- rst               - global reset signal
-- pixel_clk         - input pin, from dcm_25MHz
--                   - the clock signal generated by a DCM that has
--                   - a frequency of 25MHz.
-- HS                - output pin, to monitor
--                   - horizontal synch pulse
-- VS                - output pin, to monitor
--                   - vertical synch pulse
-- hcount            - output pin, 11 bits, to clients
--                   - horizontal count of the currently displayed
--                   - pixel (even if not in visible area)
-- vcount            - output pin, 11 bits, to clients
--                   - vertical count of the currently active video
--                   - line (even if not in visible area)
-- blank             - output pin, to clients
```

```
--                   - active when pixel is not in visible area.
----------------------------------------------------------------------------
-- Revision History:
-- 09/18/2006(UlrichZ): created
----------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- simulation library
library UNISIM;
use UNISIM.VComponents.all;

-- the vga_controller_640_60 entity declaration
-- read above for behavioral description and port definitions.
entity vga_controller_640_60 is
port(
   rst         : in std_logic;
   pixel_clk   : in std_logic;

   HS          : out std_logic;
   VS          : out std_logic;
   hcount      : out std_logic_vector(10 downto 0);
   vcount      : out std_logic_vector(10 downto 0);
   blank       : out std_logic
);
end vga_controller_640_60;

architecture Behavioral of vga_controller_640_60 is

----------------------------------------------------------------------------
-- CONSTANTS
----------------------------------------------------------------------------

-- maximum value for the horizontal pixel counter
constant HMAX  : std_logic_vector(10 downto 0) := "01100100000"; -- 800
-- maximum value for the vertical pixel counter
constant VMAX  : std_logic_vector(10 downto 0) := "01000001101"; -- 525
-- total number of visible columns
constant HLINES: std_logic_vector(10 downto 0) := "01010000000"; -- 640
-- value for the horizontal counter where front porch ends
constant HFP   : std_logic_vector(10 downto 0) := "01010001000"; -- 648
-- value for the horizontal counter where the synch pulse ends
constant HSP   : std_logic_vector(10 downto 0) := "01011101000"; -- 744
-- total number of visible lines
constant VLINES: std_logic_vector(10 downto 0) := "00111100000"; -- 480
-- value for the vertical counter where the front porch ends
constant VFP   : std_logic_vector(10 downto 0) := "00111100010"; -- 482
-- value for the vertical counter where the synch pulse ends
constant VSP   : std_logic_vector(10 downto 0) := "00111100100"; -- 484
-- polarity of the horizontal and vertical synch pulse
-- only one polarity used, because for this resolution they coincide.
constant SPP   : std_logic := '0';

----------------------------------------------------------------------------
```

```
-- SIGNALS
-------------------------------------------------------------------------

-- horizontal and vertical counters
signal hcounter : std_logic_vector(10 downto 0) := (others => '0');
signal vcounter : std_logic_vector(10 downto 0) := (others => '0');

-- active when inside visible screen area.
signal video_enable: std_logic;

begin

   -- output horizontal and vertical counters
   hcount <= hcounter;
   vcount <= vcounter;

   -- blank is active when outside screen visible area
   -- color output should be blacked (put on 0) when blank in active
   -- blank is delayed one pixel clock period from the video_enable
   -- signal to account for the pixel pipeline delay.
   blank <= not video_enable when rising_edge(pixel_clk);

   -- increment horizontal counter at pixel_clk rate
   -- until HMAX is reached, then reset and keep counting
   h_count: process(pixel_clk)
   begin
      if(rising_edge(pixel_clk)) then
         if(rst = '0') then
            hcounter <= (others => '0');
         elsif(hcounter = HMAX) then
            hcounter <= (others => '0');
         else
            hcounter <= hcounter + 1;
         end if;
      end if;
   end process h_count;

   -- increment vertical counter when one line is finished
   -- (horizontal counter reached HMAX)
   -- until VMAX is reached, then reset and keep counting
   v_count: process(pixel_clk)
   begin
      if(rising_edge(pixel_clk)) then
         if(rst = '0') then
            vcounter <= (others => '0');
         elsif(hcounter = HMAX) then
            if(vcounter = VMAX) then
               vcounter <= (others => '0');
            else
               vcounter <= vcounter + 1;
            end if;
         end if;
      end if;
   end process v_count;

   -- generate horizontal synch pulse
   -- when horizontal counter is between where the
```

```
   -- front porch ends and the synch pulse ends.
   -- The HS is active (with polarity SPP) for a total of 96 pixels.
   do_hs: process(pixel_clk)
   begin
      if(rising_edge(pixel_clk)) then
         if(hcounter >= HFP and hcounter < HSP) then
            HS <= SPP;
         else
            HS <= not SPP;
         end if;
      end if;
   end process do_hs;

   -- generate vertical synch pulse
   -- when vertical counter is between where the
   -- front porch ends and the synch pulse ends.
   -- The VS is active (with polarity SPP) for a total of 2 video lines
   -- = 2*HMAX = 1600 pixels.
   do_vs: process(pixel_clk)
   begin
      if(rising_edge(pixel_clk)) then
         if(vcounter >= VFP and vcounter < VSP) then
            VS <= SPP;
         else
            VS <= not SPP;
         end if;
      end if;
   end process do_vs;

   -- enable video output when pixel is in visible area
   video_enable <= '1' when (hcounter < HLINES and vcounter < VLINES) else
'0';

end Behavioral;
```

Appendix D: Processing System C Code

```
/******************************************************************************
**
*
* Copyright (C) 2009 - 2014 Xilinx, Inc.  All rights reserved.
*
* Permission is hereby granted, free of charge, to any person obtaining a
copy
* of this software and associated documentation files (the "Software"), to
deal
* in the Software without restriction, including without limitation the
rights
* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
*
* Use of the Software is limited solely to applications:
* (a) running on a Xilinx device, or
* (b) that interact with a Xilinx device through a bus or interconnect.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
* XILINX  BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
* WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
* OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*
* Except as contained in this notice, the name of the Xilinx shall not be
used
* in advertising or otherwise to promote the sale, use or other dealings in
* this Software without prior written authorization from Xilinx.
*
******************************************************************************
*/

/*
 *   ps7_uart is configured to 115200
 *   Authors: Michael Padberg & Dario Martinovic
 */

#include "platform.h"
#include "xbasic_types.h"
#include "xparameters.h"
#include "xil_printf.h"
#include <stdio.h>

#define NUMSEGMENTS     8
#define DISTANCETHRESHOLD 1.0
#define DANGERWARNTHRESHOLD   3.9624 //3.9624 meters
#define CRITACTIONTHRESHOLD   2.4384 //2.4384 meters
```

```
//function prototypes
void testSegmentThreshold(float dist, int segNum);
void checkSegmentThresholds(float dist, int segNum);
void delay_time();

//Global Variables

//pointer to base address of PS-PL memory location shared over AXI.
Xuint32 *baseaddr_p = (Xuint32 *)XPAR_LIDAR_AXI_0_S00_AXI_BASEADDR;

//bitmasks
const Xuint32 SEG7_NUM = 0x00E00000;
const Xuint32 SEG6_NUM = 0x001C0000;
const Xuint32 SEG5_NUM = 0x00038000;
const Xuint32 SEG4_NUM = 0x00007000;
const Xuint32 SEG3_NUM = 0x00000E00;
const Xuint32 SEG2_NUM = 0x000001C0;
const Xuint32 SEG1_NUM = 0x00000038;
const Xuint32 SEG0_NUM = 0x00000007;

//raw readings from shared PS-PL memory
Xuint32 distance_scale = 0;
Xuint32 distance_seg7 = 0;
Xuint32 distance_seg6 = 0;
Xuint32 distance_seg5 = 0;
Xuint32 distance_seg4 = 0;
Xuint32 distance_seg3 = 0;
Xuint32 distance_seg2 = 0;
Xuint32 distance_seg1 = 0;
Xuint32 distance_seg0 = 0;
Xuint32 seg_nums = 0;

//segment distances converted to meters.
float segMeters[NUMSEGMENTS];

//segment numbers parsed from seg_nums reading.
int distanceSegNums[NUMSEGMENTS];

int main()
{

      init_platform();

      //begin operating loop
      while(1)
      {

            xil_printf("Getting reading....\n\r");

            //Short delay to give the Lidar some time to acquire distance
data before next computation
            delay_time(2500000);

            xil_printf("Reading ready\n\r");
```

```
            //get raw distance readings from shared PS-PL memory
            distance_scale = *(baseaddr_p+9);
            distance_seg7 = *(baseaddr_p+7);
            distance_seg6 = *(baseaddr_p+6);
            distance_seg5 = *(baseaddr_p+5);
            distance_seg4 = *(baseaddr_p+4);
            distance_seg3 = *(baseaddr_p+3);
            distance_seg2 = *(baseaddr_p+2);
            distance_seg1 = *(baseaddr_p+1);
            distance_seg0 = *(baseaddr_p+0);
            seg_nums = *(baseaddr_p+8);

            //print scale value for conversion
            xil_printf("Distance Scale : 0x%08x \n\r", distance_scale);

            //print seg_nums output from memory
            xil_printf("Segment Nums Read : 0x%08x \n\r", seg_nums);

            //Calculate Distance Reading in Meters
            segMeters[7] = ((float)distance_seg7) / ((float) distance_scale);
            segMeters[6] = ((float)distance_seg6) / ((float) distance_scale);
            segMeters[5] = ((float)distance_seg5) / ((float) distance_scale);
            segMeters[4] = ((float)distance_seg4) / ((float) distance_scale);
            segMeters[3] = ((float)distance_seg3) / ((float) distance_scale);
            segMeters[2] = ((float)distance_seg2) / ((float) distance_scale);
            segMeters[1] = ((float)distance_seg1) / ((float) distance_scale);
            segMeters[0] = ((float)distance_seg0) / ((float) distance_scale);

            //Decode seg nums for printing using bitmask
            distanceSegNums[7] = (int) (seg_nums & SEG7_NUM) >> 21;
            distanceSegNums[6] = (int) (seg_nums & SEG6_NUM) >> 18;
            distanceSegNums[5] = (int) (seg_nums & SEG5_NUM) >> 15;
            distanceSegNums[4] = (int) (seg_nums & SEG4_NUM) >> 12;
            distanceSegNums[3] = (int) (seg_nums & SEG3_NUM) >> 9;
            distanceSegNums[2] = (int) (seg_nums & SEG2_NUM) >> 6;
            distanceSegNums[1] = (int) (seg_nums & SEG1_NUM) >> 3;
            distanceSegNums[0] = (int) (seg_nums & SEG0_NUM);

            //check all segment readings for distances within object
detection threshold
            for(int i=0; i<NUMSEGMENTS; i++)
            {
                  checkSegmentThresholds(segMeters[i], distanceSegNums[i]);
            }

            // Distance Reading
            printf("Seg7 Distance Reading : %f \t Segment: %i \n\r",
segMeters[7], distanceSegNums[7]);
            printf("Seg6 Distance Reading : %f \t Segment: %i \n\r",
segMeters[6], distanceSegNums[6]);
            printf("Seg5 Distance Reading : %f \t Segment: %i \n\r",
segMeters[5], distanceSegNums[5]);
            printf("Seg4 Distance Reading : %f \t Segment: %i \n\r",
segMeters[4], distanceSegNums[4]);
```

```
            printf("Seg3 Distance Reading : %f \t Segment: %i \n\r",
segMeters[3], distanceSegNums[3]);
            printf("Seg2 Distance Reading : %f \t Segment: %i \n\r",
segMeters[2], distanceSegNums[2]);
            printf("Seg1 Distance Reading : %f \t Segment: %i \n\r",
segMeters[1], distanceSegNums[1]);
            printf("Seg0 Distance Reading : %f \t Segment: %i \n\r",
segMeters[0], distanceSegNums[0]);

            xil_printf("End of test\n\n\r");

      } //end while(1) operation loop

      return 0;

} //end main




/*
 * Tests segment distances to see if they are below the 1m testing distance
threshold.
 * Segments lower than threshold are flagged in DDR memory to pass to PL.
 *
 * @param dist          The distance in one segment from the LeddarVu,
converted to meters.
 * @param segNum  The segment number associated with the distance reading.
 */
void testSegmentThreshold(float dist, int segNum)
{
      Xuint32 flagToWrite = 0x00000001;

      if( dist < DISTANCETHRESHOLD )
      {
            *(baseaddr_p+10) |= (flagToWrite << segNum);    //left shift by
segment number to write flag.
      }
      //flag written high if distance in segment is below threshold
      else
      {
            *(baseaddr_p+10) &= ~(flagToWrite << segNum);   //flag written
low if distance in segment is above threshold
      }
}




/*
 * Checks segment distances to see if they are below the Critical Action and
 * Danger Warning zone thresholds.
 * Segments lower than threshold are flagged in DDR memory to pass to PL.
 * Critical Action - slv_reg11
```

```
 * Danger Warning - slv_reg10
 *
 * @param dist          The distance in one segment from the LeddarVu,
converted to meters.
 * @param segNum  The segment number associated with the distance reading.
 */
void checkSegmentThresholds(float dist, int segNum)
{
      Xuint32 flagToWrite = 0x00000001;

      if( dist < CRITACTIONTHRESHOLD )
      {
            *(baseaddr_p+11) |= (flagToWrite << segNum);    //Critical action
written to slv_reg11
            *(baseaddr_p+10) |= (flagToWrite << segNum);    //Danger warning
written to slv_reg10

      //left shift by segment number to write flag.
      }
      //flag written high if distance in segment is below threshold
      else
      {
            *(baseaddr_p+11) &= ~(flagToWrite << segNum);   //Critical action
flag written low

            if( dist < DANGERWARNTHRESHOLD )
            {
            //Danger warning written to slv_reg10
                  *(baseaddr_p+10) |= (flagToWrite << segNum);    //left
shift by segment number to write flag.
            }
            //flag written high if distance in segment is below threshold
            else
            {
                  *(baseaddr_p+10) &= ~(flagToWrite << segNum);   //Danger
warning flag written low

            //flag written low if distance in segment is above threshold
            }
      }

}


/*
 * Software polling delay.
 *
 * @param t Number of iterations to poll for delay.
 */
void delay_time(long t)
{
      long i;
      for(i= t; i>0; i--)
      {
      }
}
```

Bibliography

[1] National Highway Traffic Safety Administration, "Traffic Safety Facts 2014," 2014. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812261. [Accessed 24 Mar. 2017].

[2] CBS New York, "MTA bus drivers union: Flawed bus design creates danger for pedestrians," 22 Apr. 2015. [Online]. Available: http://newyork.cbslocal.com/2015/04/22/mta-bus-design-blind-spot-pedestrians/. [Accessed 30 March 2017].

[3] E. Stolte, "Union argues Edmonton bus blind spot could hide 19 pedestrians," 07 Oct. 2016. [Online]. Available: http://edmontonjournal.com/news/local-news/union-argues-edmonton-bus-blind-spot-could-hide-19-pedestrians. [Accessed 27 March 2017].

[4] L. Hanley, "Before slamming bus drivers," New York Post, New York City, 2015.

[5] L. Bult and J. Stephansky, "Bus crushes woman to death in Queens: witnesses," *New York Daily News,* pp. http://www.nydailynews.com/new-york/queens/bus-crushes-woman-death-queens-witnesses-article-1.2097369, 30 Jan. 2015.

[6] MTA, "MTA Bus Safety Symposium," MTA, 2016.

[7] C. DiMaggio a. M. Durkin, "Child Pedestrian Injury in an Urban Setting:Descriptive Epidemiology," 1 January 2002. [Online]. Available: http://au4sb9ax7m.search.serialssolutions.com/?ctx_ver=Z39.88-2004&ctx_enc=info%3Aofi%2Fenc%3AUTF-8&rfr_id=info%3Asid%2Fsummon.serialssolutions.com&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.atitle=Child%2Bpedestrian%2Binjury. [Accessed 20 Apil 2017].

[8] J. Dickmann, "Automotive radar the key technology for autonomous driving: From detection and ranging to environmental understanding," *2016 IEEE Radar Conference (RadarConf)*, pp. 1-6, 2016.

[9] Volvo, "Volvo Buses Unveils Life-Saving Safety Technology For Unprotected Road-Users," Volvo, 20 Sept. 2016. [Online]. Available: http://www.volvobuses.com/en-en/news/2016/sep/volvo-buses-unveils-life-saving-safety-technology-for-unprotected-road-users.html. [Accessed 29 March 2017].

[10] Federal Transmit Administration, "Evaluation of Transit Bus Turn Warning Systems for Pedestrians and Cyclists," 2015. [Online]. [Accessed Apr 2017].

[11] M. Hikita, "An introduction to ultrasonic sensors for vehicle parking," MA Business Ltd , 12 May. 2010. [Online]. Available: http://www.newelectronics.co.uk/electronics-technology/an-introduction-to-ultrasonic-sensors-for-vehicle-parking/24966/. [Accessed 25 Mar. 2017].

[12] SchoolBusInfo.com, "Frequently Asked Questions (FAQ)," SchoolBusInfo, 2017. [Online]. Available: http://schoolbusinfo.com/faq.asp. [Accessed 2 May 2017].

[13] M. MacDonald, "Get a CDL," New York State DMV, 16 Febuary 2014. [Online]. Available: https://dmv.ny.gov/commercial-drivers/get-cdl-%E2%80%9Cs%E2%80%9D-endorsement-school-bus-drivers. [Accessed 26 March 2017].

[14] H. New, "New York State Commercial Driver's Manual," New York State DMV, 04 May 2016. [Online]. Available: https://dmv.ny.gov/commercial-drivers/new-york-state-commercial-drivers-manual. [Accessed 26 March 2017].

[15] H. Mulders, T. Meijman, M. Boersma, G. Groen, R. Bullinga, M. Kompier, "Absence behaviour, turnover and disability: A study among city bus drivers in the Netherlands," *Work & Stress,* vol. 4, no. 1, pp. 83-89, 27 September 1990.

[16] E. Jaffe, "The Very Mortal Life of City Bus Drivers," CityLab, 5 September 2012. [Online]. Available: https://www.citylab.com/transportation/2012/09/very-mortal-life-city-bus-drivers/3166/. [Accessed 27 March 2017].

[17] Canadian Broadcasting Corporation, "Rally for slain bus driver demands protection from 'heinous attacks'," 17 Feb 2017. [Online]. Available: http://ic.galegroup.com/ic/ovic/NewsDetailsPage/NewsDetailsWindow?disableHighlighting=&displayGroupName=News&currPage=&dviSelectedPage=&scanId=&query=&source=&prodId=&search_within_results=&p=OVIC%3AGIC&mode=view&catId=&u=mlin_c_worpoly&limiter=&display-q. [Accessed 7 Apr 2017].

[18] Anonymous, "FMCSA Proposes Standards for Truck & Bus Drivers," *Professional Safety,* vol. 61, no. 8, p. Aug, 2016.

[19] B. Hoye, "Are shields the answer? Fatal stabbing has bus drivers calling for safety barriers," 18 Feb. 2017. [Online]. Available: http://www.cbc.ca/news/canada/manitoba/bus-driver-safety-barriers-shields-winnipeg-1.3985473. [Accessed 20 Mar. 2017].

[20] L. Evans, "Confessions Of A NYC Bus Driver," Gothamist, 5 November 2014. [Online]. Available: http://gothamist.com/2014/11/05/mta_bus_driver_interview.php. [Accessed 29 March 2017].

[21] S. Miller, "Senate Passes Bill to Prevent Arrests of Bus and Taxi Drivers Who Kill," STREETSBLOG NYC, 6 June 2015. [Online]. Available:

http://nyc.streetsblog.org/2015/06/23/senate-passes-bill-to-prevent-arrests-of-bus-and-taxi-drivers-who-kill/. [Accessed 28 March 2017].

[22] D. Bernhardt and B. Hoye, "Bus driver dies after stabbing at University of Manitoba," 14 Feb. 2017. [Online]. Available: http://www.cbc.ca/news/canada/manitoba/transit-driver-attack-university-manitoba-winnipeg-1.3981581. [Accessed 30 Mar. 2017].

[23] M. Chhabria, A. Rao, U. Krishnaswamy, "Excessive sleepiness, sleep hygiene, and coping strategies among night bus drivers: A cross-sectional study," *Indian Journal of Occupational & Environmental Medicine,* vol. 20, no. 2, pp. 84-87, 2016.

[24] Victoria News Staff, "New Safety Doors Aim to Protect Transit Drivers," Victoria News, Victoria, CA, 2017.

[25] C. Stayton, J. Mandel-Ricci, P. McCarthy, K. Grasso, D. Woloch, B. Kerker, L. Nicaj, "Bicyclist Fatalities in New York City: 1996–2005," 7 April 2009. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/15389580802641761. [Accessed 7 Apil 2017].

[26] S. Gursten, "Scary secret from a bus driver: The deadly blind spot on transit buses," 31 Jan. 2015. [Online]. Available: http://www.michiganautolaw.com/blog/2015/01/31/deadly-blind-spot-buses/. [Accessed 29 Mar. 2017].

[27] American Public Transporation Association Bus Safety Committee, "Bus-pedestrian countermeasures," 2013.

[28] J. Higgins, "Bus Procurement Update," Capital Program Oversight Committee, 2017.

[29] New York City Department of Transportation, "The New York City Pedestrian Safety Study & Action Plan," New York City Department of Transportation, New York City, 2010.

[30] SeegerWeiss LLP, "Left-Turning Buses and Bus Accidents," SeegerWeiss LLP, [Online]. Available: http://www.seegerweiss.com/personal-injury/accidents/bus-accidents. [Accessed 2017].

[31] IbisWorld, "Anonymous Public Transportation in the US," 2011. [Online]. Available: http://www.ibisworld.com/industry/retail.aspx?indid=1159. [Accessed 2017].

[32] Anonymous, "Companies and Markets: Truck & Bus Manufacturing in the US," *M2 Presswire,* 2009.

[33] C. Thompson, "Businessinsider.com," 19 July 2016. [Online]. Available: http://www.businessinsider.com/mercedes-future-bus-city-pilot-photos-features-2016-7/#mercedes-didnt-have-to-start-from-scratch-with-the-future-bus-the-company-is-a-

pioneer-in-autonomous-trucks-so-the-company-was-able-to-build-upon-its-highway-pilot-syste. [Accessed 2 March 2017].

[34] EvoBus, "The Mercedes-Benz Future Bus With City Pilot," [Online]. Available: https://www.evobus.com/en/layer/Mercedes-Benz-Future-Bus-with-CityPilot/. [Accessed 2017].

[35] DAIMLER, "The Mercedes-Benz Future Bus," DAIMLER AG, 2017.

[36] Claitec, "Pedestrian Alert System (PAS)," [Online]. Available: https://www.claitec.com/en/portfolio/pedestrian-alert-system-pas/. [Accessed 6 Apr 2017].

[37] J. Rose, "TriMet's 'Talking Buses': Listen to New Pedestrian Warning (Audio)," 2013. [Online]. Available: http://www.oregonlive.com/commuting/index.ssf/2013/10/trimets_talking_buses_listen_t.html. [Accessed 06 Apr 2017].

[38] J. Rose, "TriMet's talking buses: Portland Neighborhoods Complaining About New Pedestrian-Warning Technology's Noise," 2014. [Online]. Available: http://www.oregonlive.com/commuting/index.ssf/2014/04/trimets_talking_buses_portland.html. [Accessed 06 Apr 2017].

[39] J. Rose, "Portland Buses Try Pedestrian-Alert Technologies," 2014. [Online]. Available: http://www.seattletimes.com/seattle-news/portland-buses-try-pedestrian-alert-technologies/. [Accessed 06 Apr 2017].

[40] Harsco Rail, "Safe Turn Alert 2.0," 2017. [Online]. [Accessed 06 Apr 2017].

[41] Clever Devices, "TurnWarning™ Pedestrian Warning System," Clever Devices, 2017. [Online]. [Accessed Apr 2017].

[42] R. Altstadt, "With “Distracted Walking” on the Rise, TriMet Teams Up With The Federal Transit Administration to Enhance Pedestrian Safety," TriMet News, 2013. [Online]. Available: http://news.trimet.org/2013/10/with-distracted-walking-on-the-rise-trimet-teams-up-with-the-federal-transit-administration-to-enhance-pedestrian-safety/. [Accessed 06 Apr 2017].

[43] NHTSA, "Safety Technologies," 2017. [Online]. Available: https://www.nhtsa.gov/equipment/safety-technologies#5431. [Accessed 17 Apr 2017].

[44] IIHS, "U.S. DOT and IIHS announce historic commitment of 20 automakers to make automatic emergency braking standard on new vehicles," 2016. [Online]. Available: http://www.iihs.org/iihs/news/desktopnews/u-s-dot-and-iihs-announce-historic-

commitment-of-20-automakers-to-make-automatic-emergency-braking-standard-on-new-vehicles. [Accessed 17 Apr 2017].

[45] IIHS, "Front crash prevention tests," 2017. [Online]. Available: http://www.iihs.org/iihs/ratings/ratings-info/front-crash-prevention-tests. [Accessed 17 Apr 2017].

[46] Volvo, "support.volvocars.com.uk," Volvo, 2017. [Online]. Available: http://support.volvocars.com/uk/cars/Pages/owners-manual.aspx?mc=Y555&my=2015&sw=14w46&article=4da5e00011e70e1bc0a801e800212255. [Accessed 7 April 2017].

[47] A. Jefferson, "proctorcars.com," 15 April 2015. [Online]. Available: http://www.proctorcars.com/how-do-blind-spot-monitors-work/. [Accessed 7 April 2017].

[48] C. O'Neill Grace, "First Ride On The School Bus," 16 Apr 1996. [Online]. Available: https://www.washingtonpost.com/archive/lifestyle/wellness/1996/04/16/first-ride-on-the-school-bus/ef679aaa-b0c8-4b3f-bb74-cb52e34d0e9d/?utm_term=.4bf7bfcc5939. [Accessed 27 Mar 2017].

[49] Société de l'assurance automobile du Québec, "Visibility Around Heavy Vehicles," 2017. [Online]. Available: https://saaq.gouv.qc.ca/en/road-safety/behaviours/blind-spots/visibility-around-heavy-vehicles/. [Accessed 30 Apr 2017].

[50] P. Tyson, "All About G Forces," 2017. [Online]. Available: http://www.pbs.org/wgbh/nova/space/gravity-forces.html. [Accessed 30 Apr 2017].

[51] Xilinx, "ZedBoard Zynq-7000 ARM/FPGA SoC Development Board," 2017. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/1-elhabt.html. [Accessed 30 March 2017].

[52] FPGA Blog, "Xilinx Virtex-7, Kintex-7, and Artix-7 FPGA Families," 2013. [Online]. Available: http://fpgablog.com/posts/7-series-fpga/. [Accessed 30 March 2017].

[53] Digikey Electronics, "Embedded - FPGA Product," [Online]. Available: https://www.digikey.com/products/en?FV=ffece041. [Accessed 30 March 2017].

[54] Xilinx, "Zynq-7000 All Programmable SoC Family Product Tables and Product Selection Guide," 2016. [Online]. Available: https://www.xilinx.com/support/documentation/selection-guides/zynq-7000-product-selection-guide.pdf. [Accessed 30 March 2017].

[55] Digilent: A National Instruments Company, "Nexys 4 DDR Reference," [Online]. Available: https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/start. [Accessed 30th March 2017].

[56] J. Decusati and G. Gauthier, "FPGA Based Real-Time Slam MQP," Worcester Polytechnic Institute MQP Database, Worcester, Massachusetts, 2017.

[57] Avnet: Electronics Marketing, "ZedBoard Hardware User's Guide, Version 2.2 Datasheet," 27 January 2014. [Online]. Available: http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf. [Accessed 30 March 2017].

[58] Altera, "Implementing Digital Processing for Automotive Radar Using SoCs," 2013. [Online]. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01183-automotive-radar-socfpga.pdf. [Accessed 30 Mar 2017].

[59] M. Yang, "Recent Advances in Face Detection," Honda Research Institute Mountain View, San Jose, 2014.

[60] Google Developers , "Face Detection Concepts Overview," Google, Inc. , 2017. [Online]. Available: https://developers.google.com/vision/face-detection-concepts. [Accessed 07 Apr. 2017].

[61] OpenCV.org, "OpenCV Library," OpenCV, [Online]. Available: http://opencv.org/. [Accessed 07 April 2017].

[62] A. Rosebrock, "pyimagesearch," 9 November 2015. [Online]. Available: http://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/. [Accessed 30 March 2017].

[63] S. Bauer, U. Brunsmann and S. Schlotterbeck-Macht, "FPGA Implementation of HOG-based Pedestrian Recognition System," Faculty of Engineering Aschaffenburg University of Applied Sciences, Aschaffenburg, Germany, 2009.

[64] B. Shweber, "The Autonomous Car: A Diverse Array of Sensors Drives Navigation. Driving, and Performance," mouser.com, Mansfield, 2017.

[65] J. Gabay, "Sensors for Self-Driving Vehicles," DigiKey, 2016.

[66] M. Barnard, "Tesla & Google Disagree About LIDAR — Which Is Right?," Sustainable Enterprises Media, Inc, 2016. [Online]. Available: https://cleantechnica.com/2016/07/29/tesla-google-disagree-lidar-right/. [Accessed 27 Apr. 2017].

[67] B. Berman, "Lower-cost lidar is key to self-driving future," SAE International, 11 February 2015. [Online]. Available: http://articles.sae.org/13899/. [Accessed 07 April 2017].

[68] M. Aeberhard, S. Zuther, M. Schmid, J. Dickmann, K. Dietmayer, M. Muntzinger, "Automotive Pre-Crash System with Out-of- Sequence Measurement Processing. IEEE Intelligent Vehicles Symposium," IEEE, 2010.

[69] European Editors, "Radar Sensing for Driverless Vehicles," Digi-Key's European Editors, 02 Nov. 2016. [Online]. Available: https://www.digikey.com/en/articles/techzone/2016/nov/radar-sensing-for-driverless-vehicles. [Accessed 29 Apr. 2017].

[70] European New Car Assessment Programme, "Euro NCAP Rating Review 2015," March 2016. [Online]. Available: http://www.euroncap.com/. [Accessed 29 Apr. 2017].

[71] Limpkin's blog, "Making the Electronics for a 24GHz Doppler Motion Sensor," Tindie, Inc. , 22 Feb. 2017. [Online]. Available: http://www.limpkin.fr/index.php?post/2017/02/22/Making-the-Electronics-for-a-24GHz-Doppler-Motion-Sensor. [Accessed 29 April 2017].

[72] C. Rernandez, R. Dominguez, D. Fernandez.-Liorca, J. Alonso, M. Sotelo, "Autonomous navigation and obstacle avoidance of a microbus," *International Journal of Advanced Robotic Systems* , vol. 10, no. 4, 2013.

[73] C. Premebida, "A lidar and vision-based approach for pedestrian and vehicle detection and tracking," in *IEEE Intelligent Transporation Systems Conference*, 2007.

[74] S. Lai, X Dong, J. Cui, B. Chen, "Autonomous navigation of UAV in foliage environment," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 259-276, 2016.

[75] S. Kwon, "A low-complexity scheme for partially occluded pedestrian detection using LIDAR-RADAR sensor fusion," in *IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications*, 2016.

[76] E. Ackerman, "Cheap Lidar: The Key To Making Self-Driving Cars Affordable," 22 Sep. 2016. [Online]. Available: http://spectrum.ieee.org/transportation/advanced-cars/cheap-lidar-the-key-to-making-selfdriving-cars-affordable. [Accessed 07 Apr. 2017].

[77] E. Ackerman, "Sweep Is a $250 LIDAR With Range of 40 Meters That Works Outdoors," 06 Apr. 2016. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/robotics-hardware/sweep-lidar-for-robots-and-drones. [Accessed 07 Apr. 2017].

[78] Scanse, "Scanse," Scanse LLC, 2017. [Online]. Available: http://scanse.io/. [Accessed 07 Apr. 2017].

[79] T. Huang, "RPLIDAR - Slamtec - Leading Service Robot Localization and Navigation Solution Provider," Slamtec.com, [Online]. Available: http://www.slamtec.com/en/Lidar.. [Accessed 17 April 2017].

[80] S. Higgins, "Spar3D," Diversified Communications, 2017. [Online]. Available: http://www.spar3d.com/news/lidar/vu8-475-solid-state-lidar-adapts-needs/. [Accessed 07 Apr. 2017].

[81] LeddarTech, "LeddarVu Solid-State LiDAR Platform," Leddartech, 2017. [Online]. Available: http://leddartech.com/modules/leddarvu/. [Accessed 26 Apr. 2017].

[82] LeddarTech, "LeddarOne Single-Element Sensor Module," LeddarTech, 2017. [Online]. Available: http://leddartech.com/modules/leddarone/. [Accessed 26 Apr. 2017].

[83] A. Chavan, S. Yogamani, "Real-time DSP implementation of pedestrian detection algorithm using HOG features," 2th International Conference on ITS Telecommunications, 2012.

[84] S. Molina-Giraldo, H. Insuasti-Ceballos, C. Arroyave, J. Montoya, J. Lopez-Villa, A. Alvarez-Meza, G. Castellanos-Dominguez, "People detection in video streams using background subtraction and spatial-based scene modeling," 20th Symposium on Signal Processing, Images and Computer Vision (STSIVA), 2015.

[85] S. Hilado, E. Dadios, L. Gan Lim, E. Sybingco, I. Marfori, A. Chua, "Vision based pedestrian detection using histogram of oriented gradients, adaboost & linear support vector machines," TENCON 2012 IEEE Region 10 Conference,, 2012.

[86] J. Lee, J. Choi, E. Jeon, Y. Kim, T. Le, K. Shin, H. Lee, K. Park, "Robust pedestrian detection by combining visible and thermal infrared cameras," Sensors , 2015.

[87] L. M. Tamaki, "Pedestrian Accidents: Representative Cases," Minamtimaki, 2017.

[88] S. Umchid, "Directivity Pattern Measurement of Ultrasound Transducers," *International Journal of Applied Biomedical Engineering*, vol. 2, no. 1, pp. 39-43, 2009.

[89] M. Panicker, "Multisensor data fusion for autonomous ground vehicle," in *Conference on Advances in Signal Processing*, 2016, pp. 507-512.

[90] MaxBotix, Inc., "HRXL-MaxSonar®- WRS Series," MaxBotix, Inc., Brainerd, MN, 2012.

[91] MaxBotix, Inc., "HRXL-MaxSonar®- WR Series," MaxBotix, Inc., Brainerd, MN, 2012.

[92] MaxBotix, Inc., "I2CXL-MaxSonar®- WR/WRC Series," MaxBotix, Inc., Brainerd, MN, 2012.

[93] S. Morgan, "sciencing.com," 2017. [Online]. Available: http://sciencing.com/heat-sensors-work-4613395.html. [Accessed 21 April 2017].

[94] Mouser Electronics, "D6T-44L-06 Omron Electronics | Mouser Europe," 2017. [Online]. Available: http://www.mouser.com/ProductDetail/Omron-Electronics/D6T-44L-06/?qs=IQwVXVmkiFL3XeQ%252bJMEhQw%3D%3D. [Accessed 7 April 2017].

[95] Melexis, "Far Infrared Sensor Array at High Speed with Low Noise (16x4 RES)," Melexis, 2017. [Online]. Available: https://www.melexis.com/en/product/MLX90621/Far-Infrared-Sensor-Array-High-Speed-Low-Noise. [Accessed 7 April 2017].

[96] K. Charles, "What Speed Should I Put the Treadmill at for Jogging?," Livestrong, 20 Sept 2010. [Online]. Available: http://www.livestrong.com/article/252045-what-speed-should-i-put-the-treadmill-at-for-jogging/. [Accessed 2 May 2017].

[97] "Copenhagen City of Cyclists," City of Copenhagen, 2011. [Online]. Available: http://www.cycling-embassy.dk/wp-content/uploads/2011/05/Bicycle-account-2010-Copenhagen.pdf . [Accessed 2 May 2017].

[98] Silica: An Avnet Company, "Zynq Workshop for Beginners: Version 1.0 (Zedboard - Vivado 2014.1)," 2014. [Online]. Available: https://products.avnet.com/opasdata/d120001/medias/docus/3/SILICA_Xilinx_Zynq_ZedBoard_Vivado_Workshop_ver1.0.pdf. [Accessed 21 April 2017].

[99] "Digikey Electronics Piezo Electric Buzzers," [Online]. Available: https://www.digikey.com/products/en/audio-products/buzzer-elements-piezo-benders/160. [Accessed 30th April 2017].

[100] "Digikey Electronics Vibrating Motor Controller," [Online]. Available: https://www.digikey.com/products/en?mpart=28822&v=149. [Accessed 30th April 2017].

[101] Melexis, "MLX90621 16x4 IR Array Datasheet," Melexis , Nashua, NH, 2016.

[102] Panasonic Industrial Devices, "PaPIR EKMC (VZ) Series," Panasonic, Newark, NJ, 2018.

[103] Digilent, Inc., "PMODACL2 Reference Manual," Digilent, Inc., Pullman, WA, 2018.

[104] T. Liu, "Digital-output relative humidity & temperature sensor/module DHT22," Aosong Electronics Co. Ltd.

[105] Leddartech Inc., "LeddarVu and Configurator User Guide," Leddartech, Inc., Quebec City, CA, 2017.

[106] Delta Electronics, "PMC Panel Mount Power Supply, 12V 100W 1 Phase / PMC-12V100W1A," Delta Electronics, 2017.

[107] SMAKN, "SMAKN DC/DC Converter 12V to 6V/3A," SMAKN, 2017.

[108] J. Johnson, "Creating a Base System for the Zynq in Vivado," FPGA Developer, 2014.

[109] J. Johnson, "Creating a Custom IP Block in Vivado," FPGA Developer, 2014.

[110] U. Zoltán, "vga_controller_640_60.vhd," Digilent, Inc., 2006.

[111] "Benefits of Acrylic Displays," Independent Retailer, 2010.

[112] J. Sprinkle, "How and Why to use the Ackermann Steering Model," 2016.

[113] G. Monteiro, U. Nunes, P. Peixoto, C. Premebida, "A lidar and vision-based approach for pedestrian and vehicle detection and tracking," *2007 IEEE Intelligent Transportation Systems Conference,* pp. 1044-1049, 2007.

[114] Slamtec, "Slamtec," Shanghai Slamtec Co., Ltd, 2017. [Online]. Available: http://www.slamtec.com/en/Lidar. [Accessed 07 Apr. 2017].

[115] B. Zheng, "Next Generation Automotive Architecture modeling and exploration for autonomous driving," in *IEEE Compyter Society Annual Symposium*, ISVLSI, 2016, pp. 53-58.

[116] T. Vanderblit, "Bus Driver Appreciation Day: A fitting tribute to one of the most stressful jobs in the world," Slate Magazine, 26 January 2011. [Online]. Available: http://www.slate.com/articles/life/transport/2011/01/the_most_stressful_job_on_the_planet.html. [Accessed 30 March 2017].

[117] O. Tuzel, F. Porikli and P. Meer, "Pedestrian Detection via Classification on Riemannian Monifolds," Institute of Electrical and Electronic Engineers , 2008.

[118] V. Terenina, "blog.seon.com," 30 November 2016. [Online]. Available: http://blog.seon.com/2016/11/30/technology-and-strategies-that-save-lives-eliminating-blind-spots-on-transit-buses/. [Accessed 2 March 2017].

[119] V. Tambellini and M. B. Cahill, "Market Insight Series," March 2016. [Online]. Available: https://www.thetambellinigroup.com/wp-content/uploads/2016/04/Market-Insights-Series_Banner-by-Ellucian-Customer-Survey-Results.pdf.

[120] J. Steinberg, "Inc," 25 May 2016. [Online]. Available: http://www.inc.com/paul-metselaar/notes-to-my-younger-self-the-2-famous-quotes-that-taught-me-everything.html.

[121] G. Seo, "Challenges in implementing enterprise resource planning (ERP) system in large organizations : similarities and differences between corporate and university environment," Massachusetts Institute of Technology: Sloan School of Management, Cambridge, 2013.

[122] K. Rutledge, "ipdusa.com," 13 June 2011. [Online]. Available: https://www.ipdusa.com/techtips/10086/what-is-blis-blind-spot-information-system. [Accessed 7 April 2017].

[123] RSmart, "PR Newswire," 3 September 2013. [Online]. Available: http://www.prnewswire.com/news-releases/erp-systems-have-cost-colleges-and-universities-5-to-10-billion-222316221.html.

[124] M. Perlin, "Evolven," 17 September 2012. [Online]. Available: https://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html.

[125] J. Lemelson &. R. Pedersen, "GPS vehicle collision avoidance warning and control system and method," 2002.

[126] S. Neuendorffer, T. Li and D. Wang, "Zync-7000 OpenCV Application," Xilinx Reference, 2015.

[127] K. Negi, K. Dohi, Y. Shibata and K. Oguri, "Deep Pipelined One-Chip FPGA Implementation of a Real-Time Image-Based Human Detection Algorithm," in *International Conference on Field-Prgrammable Technology*, 2011.

[128] L. B. Mehlinger, "Indicators of succesfful enterprise technology implemenations in higher education," Morgan State University, 2006.

[129] L. Maiello, "Key Steps to Managing Left-Turning Buses and Pedestrian Safety," METRO, 2015.

[130] J. Maida, "Bussiness Wire: A Berkshire Hathaway Company," 19 October 2016. [Online]. Available: http://www.businesswire.com/news/home/20161019005435/en/Top-5-Vendors-SIS-Market-Higher-Education.

[131] Jeff, "Merit Pages," 24 August 2015. [Online]. Available: http://www.meritsolutions.com/business-insights/risks-associated-with-erp-implementations/.

[132] J. Cui, "Autonomous Navigation of UAV in Foliage Environment," Springer Science & Business Media, Dordrecht, 2016.

[133] F. Ashframe, "LinkedIn," 1 May 2010. [Online]. Available: https://www.slideshare.net/ashfame/handling-web-servers-of-high-traffic-sites.

[134] Workday, "Workday," [Online]. Available: https://www.workday.com/en-us/applications/why-workday.html?wdid=enus_ws_hm_wdherotext_wds_wd_web_17.0084.

[135] Worcester Polytechnic Institute: Office of the Registrar, "Worcester Polytechnic Institute," 28 March 2017. [Online]. Available: https://www.wpi.edu/offices/registrar/course-registration.

[136] Volvo, "Volvo.custhelp.com," 2017. [Online]. Available: http://volvo.custhelp.com/app/answers/detail/a_id/9561/~/blind-spot-information-system-(blis)-and-cross-traffic-alert-(cta). [Accessed 7 April 2017].

[137] OkSolar, "Technical LEDs Led Color Chart," OkSolar, [Online]. Available: http://www.oksolar.com/led/led_color_chart.htm. [Accessed 30th April 2017].

[138] Federal Motor Carrier Safety Adminisitration, "Pocket Guide to Large Truck and Bus Statistics," FMCSA, Washington, 2016.

[139] WRSH, "New York Pedestrian Knockdown Lawyer," WRSH, New York, 2017.

[140] "New York Bus Driver Salaries," Salary.com, 31 March 2017. [Online]. Available: http://www1.salary.com/NY/Bus-Driver-salary.html. [Accessed 29 March 2017].

[141] Maxim Integrated, "maximintegrated.com," 2017. [Online]. Available: https://www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/MAX30205.html. [Accessed 7 April 2017].

[142] LAPIS Semiconductor, "lapis-semi.com," 2017. [Online]. Available: http://www.lapis-semi.com/en/semicon/sensor/ir-sensor.html. [Accessed 7 April 2017].

[143] Urban Transport News, Greater Cleveland Regional Transit Authority.

[144] Fisher & Talwar, "fishertalwar.com," 2014. [Online]. Available: http://www.fishertalwar.com/bus-accident-statistics/. [Accessed 7 Apri 2017].

[145] "Department of Labor," New York State Department of Labor, [Online]. Available: https://labor.ny.gov/stats/olcny/bus-driver.shtml. [Accessed 27 March 2017].

[146] "Coronary Heart-Disease and Physical Activity of Work,” Coronary Heart-Disease and Physical Activity of Work," Science Direct , [Online]. Available:

http://www.sciencedirect.com/science/article/pii/S0140673653914950. [Accessed 27 March 2017].

[147] S. Morgan, "How Do Heat Sensors Work?," Sciencing, 2017. [Online]. Available: http://sciencing.com/heat-sensors-work-4613395.html. [Accessed 21 April 2017].