AUTONOMOUS MAPPING ROBOT

A Major Qualifying Project Report:

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

_____

Jonathan Hayden

_____

Hiroshi Mita

_____

Jason Ogasian

Date: April 21, 2009

Approved:

_____

R. James Duckworth, Major Advisor

_____

David Cyganski, Co-Advisor

Abstract

The purpose of this Major Qualifying Project was to design and build a prototype of an autonomous mapping robot capable of producing a floor plan of the interior of a building. In order to accomplish this, several technologies were combined including, a laser rangefinder, ultrasonic sensors, optical encoders, an inertial sensor, and wireless networking to make a small, self-contained autonomous robot controlled by an ARM9 processor running embedded Linux. This robot was designed with future expansion in mind.

Acknowledgements

There are many people who deserve thanks for their help with this project. First we would like to thank our advisors for this project: professors Duckworth and Cyganski, without whom this project would not have been possible at all. We would also like to thank Tom Angelotti in the ECE shop for all of his help, as well as the many other members of Worcester Polytechnic Institute's Department of Electrical and Computer Engineering who provided assistance along the way. Finally we would like to thank the Department of Electrical and Computer Engineering at WPI for providing funds to purchase the laser rangefinder, a major contributing factor to the success of this project.

Table of Contents

Abstract I

Acknowledgements II

Table of Figures V

Table of Tables VI

1. Introduction 1

2. Background 2

2.1 CMU Mapping Robot 3

2.2 MobileRobots Inc. MapperBot 4

2.3 Centibots 4

2.4 Conclusion 5

3. Overall Design 6

3.1 Design Requirements 6

3.2 System Overview 7

4. Robot Base 9

5. Microprocessor 10

5.1 Architecture and Features 10

5.2 Embedded Operating System 11

6. Custom PCB 12

6.1 Power Supply 14

6.1.1 Method 14

6.1.2 Design 15

6.1.3 Battery Compartment 17

6.2 Wireless Device 18

7. Motor Controller 18

8. Sensors 19
8.1 Optical Encoders 20
8.2 Accelerometer/Gyroscope 23
8.3 Laser Range Finder 26
8.3.1 Sensor Evaluation 27
8.4 Ultrasonic Sensor Array 29
8.4.1 Sensor Evaluation 31
9. Software 34
9.1 Communication Protocol 34
9.2 Robot Code 35
9.2.1 Device Drivers 35
9.2.2 Timer Scheduler 35
9.2.3 Navigation Algorithm 36
9.3 Host Application 37
10. Final Results 39
10.1 Capabilities and Test Results 40
10.2 Quantitative Performance Analysis 42
11. Recommendations 43
Appendix A – Load Currents 44
Appendix B – Power Supply 45
Appendix C – Daughter Board Schematic 46
Appendix D – Daughter Board Layout 47
Appendix E – Ultrasonic Sensor Measurements 48
References 50

Table of Figures

Figure 1 - CMU 3D Mapping robot 3
Figure 2 - MobileRobots Inc. MapperBot (http://www.mobilerobots.com/smMapperBot.gif) 4
Figure 3 - Centibots (http://www.ai.sri.com/centibots/pictures/robots-dec-2003/index.html) 5
Figure 4 - Overall System Block Diagram 7
Figure 5 - Max '99 Robot Base. Adapted from Zagros robotics website. (https://www.zagrosrobotics.com/shop/item.asp?itemid=523) 9
Figure 6 - SAM9-L9260 Development Board from Olimex 11
Figure 7 - Final Daughter Board Layout 12
Figure 8 - Top and Bottom of Populated Daughter Board 13
Figure 9 - Block Diagram of the Power Supply 16
Figure 10 Battery compartment close up 17
Figure 11 - Lantronix MatchPort b/g 18
Figure 12 - WIRZ #203 Motor Controller Mounted to Robot Deck 19
Figure 13 - QRB1134 operation 21
Figure 14 - Quadrature encoder operation 21
Figure 15 - Hokuyo URG-04LX Laser Range Finder (http://www.hokuyo-aut.co.jp/02sensor/07scanner/urg_04lx.html) 26
Figure 16 - Visual data provided by vmon application 28
Figure 17 - SRF05 Timing Diagram, Mode 2 30
Figure 18 - Ultrasonic Sensory Array Layout, Bottom View 31
Figure 19 - Robot Base with Ultrasonic Sensor Array 31
Figure 20 - Error of the Ultrasonic Sensor Plotted 33
Figure 21 - Software Architecture Block Diagram 34
Figure 22 - Code Flow Diagram of the Host Application 38
Figure 23 - Screenshot of the Graphical User Interface (this figure will be updated later) 39
Figure 24 - Fully assembled robot 40
Figure 25 - Left: robot-generated floor plan; right: actual floor plan 41
Figure 26 - Quantitative analysis of a hallway in the map 42
Figure 27 - Schematic of the Power Supply 45
Figure 28 - Daughter Board Schematic 46
Figure 29 - Layout of the Daughter Board 47

Table of Tables

Table 1 - Binary outputs for quadrature encoder 22
Table 2 - IMU parts decision matrix 24
Table 3 - IMU decoder truth table 25
Table 4 - Laser rangefinder data points and scatter plot 29
Table 5 - Load Current of Each Device 44
Table 6 - Power Supply Parts List 45
Table 7 - Accuracy and precision measurement of the SRF05 Ultrasonic Sensor 48
Table 8 - Angular Range Measurement of the SRF05 Ultrasonic Sensor 49

1. Introduction

Robots are becoming more and more common in our daily lives showing up in the form of everything from children's toys, to robotic vacuum cleaners, to home security robots; robots have been doing automated tasks in factories for decades. With the ever-increasing speed and power of digital systems coupled with the continuously expanding field of robotics, it is becoming more practical to build custom robotic systems with a degree of flexibility and freedom that was once impossible, giving robots the ability to communicate wirelessly or to act autonomously. Now, instead of robots simply performing menial tasks such as repetitive jobs at factories, robots can perform jobs once thought to be reserved for humans, without the risk a of danger to a human life. One such job that would be much safer for a robot to take on than a human is that of search and rescue. Such a robot, if it could perform its duties safely and reliably, would be of immense value to any rescue operation. The robot prototype designed and built for this major qualifying project was meant to be a starting point for such a robot; due to time and budget constraints building a fully functional search and rescue robot was not feasible.

The robot that was designed and built for this project was intended to have some of the basic components needed to develop a fully functional search and rescue robot. While such a robot would need many complicated systems all interfaced together the design of the robot presented in this paper focused on what was thought to be one of the core functionalities needed, autonomous mapping. The goal of the autonomous mapping robot was to be able to allow the robot to know its location relative to a known starting point and to be able to construct a floor plan of the surrounding structure on a wirelessly connected computer. As a proof of concept of this functionality the goal for this project was to have the robot make a map of the third floor of the Atwater Kent electrical and computer engineering building at Worcester Polytechnic Institute. Such a feature would be of vital importance to any rescue team using the robot because it would allow them to know the nature of the location before ever sending in a human, which would be a great asset especially in potentially dangerous situations.

By designing and building the autonomous mapping robot many steps were taken towards the design of a fully functional search and rescue robot. This design provides a solid base for further development of the robot by future students. The autonomous mapping robot was built around an embedded processor system running a custom Linux kernel. This processor was used to control the many systems incorporated into the design of the robot. The major functional blocks of this robot are its base – the platform on which all the systems were mounted, the processor, the power circuitry, the

motor controller, and finally the sensors, including optical wheel encoders, an inertial measurement unit, ultrasonic sensors, and a laser rangefinder. Each of these individual systems was interfaced through the processor using various drivers and programs, while the mapping and location data were sent to the graphical user interface using a custom communications protocol designed for an 802.11 wireless connection to the host computer.

Each of the individual functional blocks of the overall mapping robot will be discussed in great detail in the following chapters beginning with a discussion of existing autonomous mapping robots. After that will be the overall design of the robot including requirements and a functional overview. The next topic will be the base chosen for the robot followed by a chapter about the ARM9 microprocessor and its operating system. From there the paper goes on to discuss each of the sensors used on the robot including the laser rangefinder for gathering mapping data, the ultrasonic sensors for navigational and obstacle avoidance purposes, the optical wheel encoders used for elementary dead reckoning techniques, and the accelerometer and gyroscope which were used to make an inertial measurement unit for the robot to aid in the dead reckoning algorithm. The next chapter goes on to discuss the motor controllers used for the robot's drive wheels, followed by a detailed description of the power supply for the robot. The next few chapters will focus on the software side of the robot outlining the communications protocol written to pass data between the client (robot) and the host machine, the code design and program flow for the programs running on the robot, the graphical user interface necessary for displaying the mapping data and robot position, and finally the navigation algorithm chosen to control the robot's actions. The final two chapters share the results of this project as well as some recommendations for any further work to be done on the robot.

2. Background

One important step in designing a new mapping robot was to study and understand existing autonomous mapping robots as well as to attempt to find possible improvements to these established designs. This paper examines three separate existing autonomous mapping robots to see the different methods used to achieve their goals. The information presented concerns mapping robots from Carnegie Mellon University, Mobile Robots Inc., and the Centibots project. Different methods were used in each project however, there were some constant themes that provided a good outline for a new mapping robot.

2.1 CMU Mapping Robot

Carnegie Mellon University has built several different types of mapping robots to map both indoor and outdoor terrain. On the CMU robotics website, they present information on two of their mapping robots, which are both capable of mapping in three dimensions. The first of these is designed to map a mine, while the second robot is designed to map a 3D texture map.

Two robotic systems were used in their research of mapping mines. The first type was a cart that had to be manually walked through the mine in order to map the mine. The cart was equipped with four 2D laser range finders to provide information on the mine cross section ahead of the cart and the ceiling structure above the cart. The second type they have developed is a tele-operated device constructed from the chassis of two ATVs. This robot was equipped with two laser range finders, one directed forward and the other pointed towards the ceiling. In order to map out an accurate 3D map, the researchers developed a complicated algorithm which was explained in their report. The Groundhog, as the robot is known, was not equipped with an odometer or an inertial sensor to obtain the robot's global coordinates, however their algorithm allows them to calculate an estimate of the robot's location by taking two consecutive scans and calculating the relative displacement. Though the robot used to do their mapping does not exhibit autonomy like the one presented in this paper, their mapping algorithm and strategies are great resource to us.(6)

The 3D texture mapping robot designed by students at CMU was equipped with a home-made panoramic camera and two laser range finders. The panoramic camera was constructed by simply combining a camera with a convex mirror. This allowed the camera to record its surroundings without having to move it. They achieve their goal by taking data points using laser range finders and combining that data with the images taken with the camera.(3)

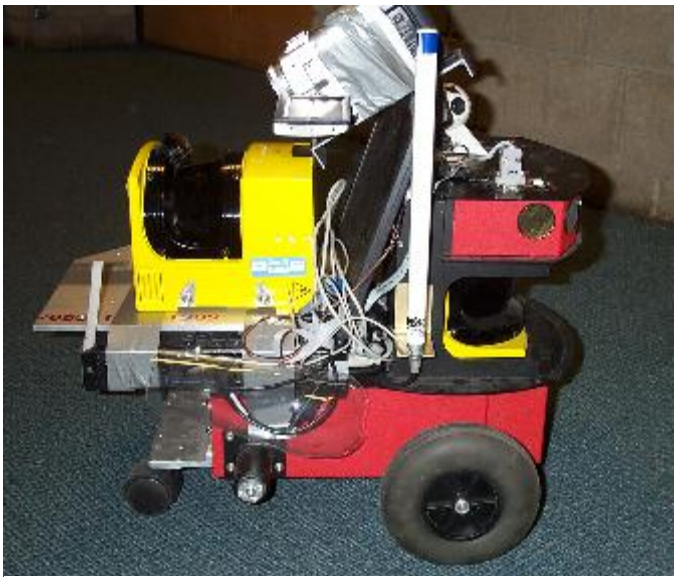


Figure 1 - CMU 3D Mapping robot

Both of these robots were successful in constructing 3D maps and were great examples to examine. The only concern is the cost of these robots. One of the constant themes seen in these robots

was the use of laser rangefinders to gather the data needed in order to make the actual map. While multiple lasers were needed in order to build 3D maps this would not be the case for this robot. Another valuable resource gained from these projects was the algorithms which were developed for use with each of these sophisticated robots.(7)(8)

2.2 MobileRobots Inc. MapperBot

This robot was built with the purpose of constructing 2D maps of an area, having been first guided through the area by a human. This robot featured WiFi control and tracking, as well as a capable laser rangefinder. Optionally, a user-controlled camera may be added to this robot allowing a human operator to see exactly what the robot sees as it drives. The informational page about this robot boasts the ability to map a 30,000sq ft building in 30 minutes.(9)

Figure 2 - MobileRobots Inc. MapperBot (http://www.mobilerobots.com/smMapperBot.gif)

The base of this robot was equipped with a pair of drive wheels and a single castor, acoustic close range sensors, a laser rangefinder, and an 802.11b radio. This robot used the laser rangefinder to generate map data, which it then saved as a JPEG compressed image file for record keeping. The robot was also able to accept commands via the WiFi connection for moving around its environment. The website does not include pricing information for the robot. Many of the goals of this robot are shared in this project. This robot also employs the use of a laser rangefinder to do its mapping. It also uses ultrasonic sensors in order to achieve its short-range navigational needs. Wireless data transfer, along with the ability to control the robot through this link, was also intriguing characteristics of MobileRobots' MapperBot.

2.3 Centibots

The Centibots Project was a project intended to "demonstrate by December 2004, 100 robots mapping, tracking, guarding in a coherent fashion during a period of 24 hours." [10] The project members included SRI International, Stanford University, University of Washington and ActivMedia.

This project used 97 ActivMedia Amigobots and 6 ActivMedia Pioneer 2 AT robots. Each of the Pioneer 2 AT robots was equipped with a laser range finder (a SICK 200), an on-board computer,

odometer, sonar, camera and an Inertial Navigation System. Each of the Amigobots was equipped with an onboard computer, sonar, wireless USB and a USB camera. The sonar on both robots was used for navigation purposes only, and data collection was done with both the camera and the laser rangefinder.

The main goal of this project was to have 100 robots work together. The first wave of robots (the pioneer 2 AT) mapped the building or area and sent the map wirelessly to each of the other robots. The Amigobots then move in an optimal way into the building or the area to sense intruders. These robots were capable of communicating with one another and performing the tasks more effectively. Each robot was also equipped with wireless networking equipment allowing the formation of an ad-hoc network between all of the robots.

Figure 3 - Centibots (http://www.ai.sri.com/centibots/pictures/robots-dec-2003/index.html)

These robots used existing mapping techniques to utilize the data from their laser rangefinders [11]. Similar to the MapperBot these robots also employed the use of ultrasonic sensors to gather the needed data for short range navigational purposes. This project also had a good amount of interesting and useful information concerning robots and wireless communication. This project, along with the references it lists was a valuable source of information to for this project.

2.4 Conclusion

Using successful robots as a starting point, research for the autonomous mapping robot presented in this paper has been divided into several important areas. Each of these existing robots has shown that laser range finders are almost universally used for mapping applications because they currently provide the best accuracy over medium to long ranges. Additionally, short-range sensors, such as ultrasound sensors, are a common solution for navigation and obstacle avoidance. Each of the aforementioned robots communicates wirelessly with a host computer, which then generates a map from the data provided by the robot(s). From these projects it was also determined that a powerful processor will be needed to interpret the sensor data and navigate the robot. By studying each of these existing mapping robots several themes were discovered, many of which were incorporated into the design for this

autonomous mapping robot. Even though not all of the techniques used by these other robots will be used they provided a solid understanding of different ways these goals can be accomplished.

3. Overall Design

This chapter introduces the design requirements and the functional overview of the autonomous mapping robot. The design requirement of this project was determined by the project goal of mapping the third floor of the Atwater Kent Laboratories located at WPI. This goal was chosen to act as a controlled proof of concept of the robot's capabilities. By mapping a room which can be easily accessed it is possible to test the accuracy of the sensor readings from the robot as well as to compare the generated map to available blueprints of the location. After presenting the requirements the robot must meet a functional overview of the robot introducing each device needed for the robot to complete its core functions is provided.

3.1 Design Requirements

The following design requirements were determined by the team members in order to achieve the project goal of being able to map the third floor of Atwater Kent as well as to help equip the robot to be able to map other similar areas:

1. The robot needed a durable chassis to hold all of the necessary sensors and drive equipment.
2. An efficient power supply to power all the devices for at least 1~2 hours was required.
3. A fast microprocessor was needed to control all the devices and process their incoming data.
4. The robot needed to have an accurate range sensor to collect mapping data.
5. The robot had to have some way to collect orientation data to place itself in the map.
6. A host computer had to be equipped with software capable of receiving data from the robot and plotting the map from the given data.
7. The robot requires a wireless module to communicate wirelessly with the host computer.
8. The host computer must have a graphical user interface to display the collected data properly in a useful manner.

Design decisions were made in order to achieve the design requirements above. The following subchapter will give an overview of the robot's design along with an explanation of each component in the robot.

3.2 System Overview

This subchapter discusses the overall system design of the autonomous mapping robot along with a brief discussion on each of the components. Arguably the most important component of the robot is the AT91SAM9260 microprocessor, due to the fact that it is the central point of integration between all of the onboard equipment the robot possesses. The processor used on the robot is built into a development board which was purchased from Olimex, which is discussed in greater detail in Chapter 5 of this report. The processor was also required to be able to perform all of the on-board computing for navigational purposes as well as communicate with the host machine which runs a graphical user interface to plot the mapping data. Connected to the microprocessor is the motor controller, laser range finder, an array of ultrasonic sensors, accelerometer, gyroscope, optical encoders and the wireless module. Figure 4 provides a simple overall system diagram of the robot and, the connections used for each of the components, and the interface with the host system used to display the map.

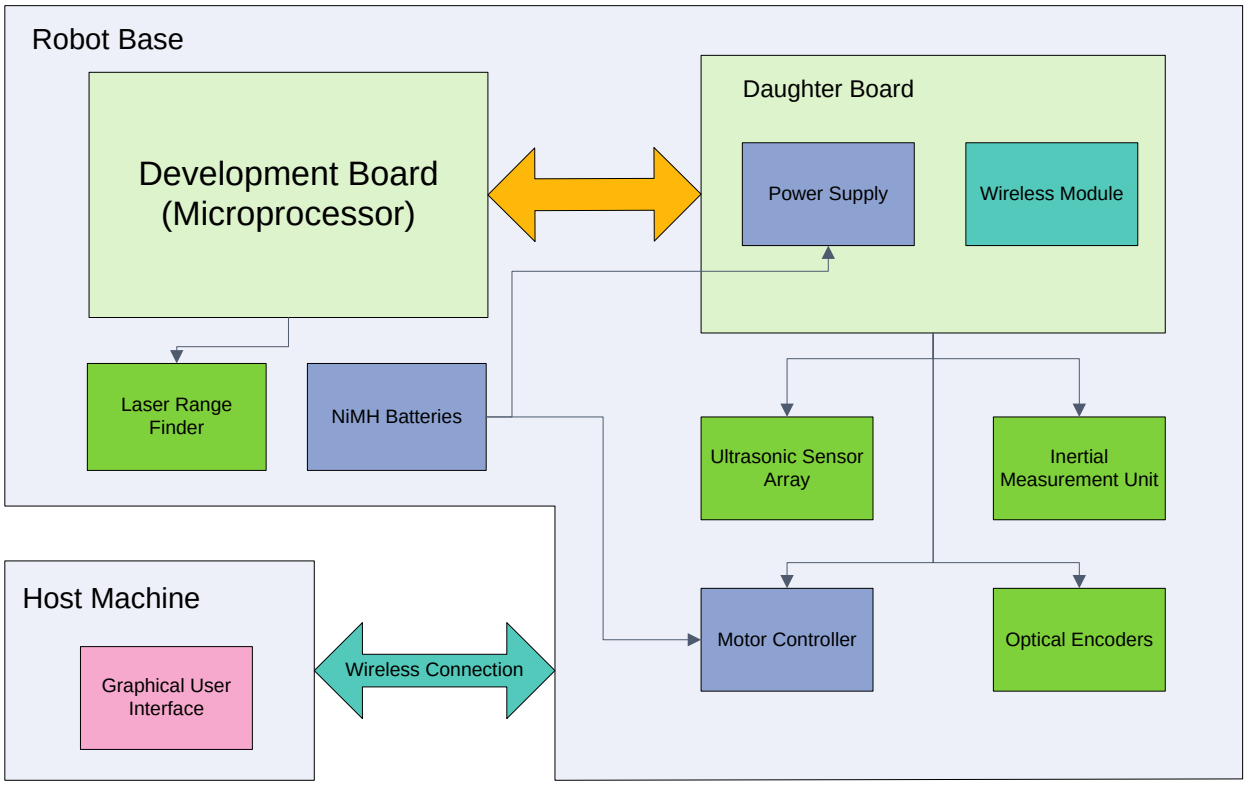


Figure 4 - Overall System Block Diagram

From the above diagram it can be seen that the laser rangefinder is the only sensor connected directly to the microprocessor development board. The laser range finder is the primary sensor of this robot used to collect mapping data. The URG-04LX laser range finder manufactured by Hokuyo was

selected, which was connected to the development board via USB. The data retrieved from the laser range finder data is combined with the orientation and position data in order to produce the map.

The rest of the sensors used on the robot are interfaced with the development board through a custom printed circuit board which contains the power circuitry for the robot as well as the wireless module used for communication with the host machine. The robot is accessed via an ad-hoc network allowing communication between the host and robot.

An array of ultrasonic sensors was used for the robot's navigational purposes. The robot is equipped with five ultrasonic sensors spaced around the robot to allow effective implementation of the navigation algorithm as well as adequate obstacle avoidance capabilities. The microprocessor makes decisions on where to move next or to stop based on the data returned from the ultrasonic sensors. These sensors are all interfaced with the development board through the custom PCB.

The robot is equipped with a WIRZ #203 motor controller which was provided with the robot chassis which was ordered from Zagros Robotics. The motor controller is controlled by a Pulse Width Modulated (PWM) signal where the duty cycle of the PWM signal determines the rotation rate of each wheel. The motor controller is connected to the SAM9260 microprocessor timer-counter which generates the PWM signal for each of the wheel to control the robots movement through the custom PCB.

The inertial measurement unit, made from an accelerometer and a gyroscope, along with the optical encoders, is used to collect orientation and position data. The accelerometer and the gyroscope are connected to the microprocessor via a serial peripheral interface (SPI) bus. The accelerometer is meant to provide the robot with acceleration data to determine distance traveled and the gyroscope provides angular acceleration data used to compute the heading of the robot. Due to some operating system and software issues which are discussed later these parts are not currently being utilized by the system. The robot was also equipped with two optical encoders on each wheel which provide information on the direction the wheel is turning and the distance it has covered. These systems together allow the use of dead reckoning to determine the robot's location.

Data collected by each sensor are all sent to the microprocessor where preliminary data processing is performed for the robot navigation. The data is stored into data structures headed with a message ID to identify what the data type is and is then sent to the host computer where the data is displayed by a graphical user interface.

Each of the functional subsystems presented in Figure 4 above are interfaced with the robot through the processor, allowing control of the motors, data gathering and processing from the multitude of sensors present, and wireless communication module. In the following chapters each of the aforementioned modules is discussed in greater detail.

4. Robot Base

The choice of a base for the robot was of great importance to the overall design since it dictates the size and weight constraints for the rest of the components. At the beginning of the project it was decided that purchasing a pre-made robot chassis would allow the focus to remain on the computer engineering aspects of the robot without having to worry about the mechanical engineering work needed to design an effective base. A search of existing robot bases yielded a large number of possible bases. Generally these bases were comprised of a simple chassis, two drive wheels and occasionally came equipped with an embedded processor. The most important aspects were the size, features, and price of the base. One of the goals of this project was to make the robot as small and manageable as possible while still maintaining sufficient size to carry all of the necessary sensing equipment. The absolute largest size would be 75cm in any dimension to ensure that the robot would always be able to fit through standard doorways. A reasonable starting estimate for the robot size envelope was decided to be a 40cm cube. The design of the robot meant that it would move by means of electric motors running a set of drive wheels; bases which include drive motors and controllers were desirable, allowing the focus to be on the design and implementation of the sensing and processing aspect of the robot. Therefore, bases which included any advanced circuitry (such as a microprocessor or wireless radio) were not considered. The price range for bases considered for this robot spanned $100-$400. Simple, well-made bases from Zagros Robotics were decided to be worth the extra design effort of purchasing and putting together a motor, motor controller, and wheel setup that work well together.

Figure 5 - Max '99 Robot Base. Adapted from Zagros robotics website. (https://www.zagrosrobotics.com/shop/item.asp?itemid=523)

The base that was chosen was the Max '99 Robot Kit from Zagros Robotics, Figure 5. This robot chassis included three round platforms, two drive wheels and two castors, as well as motors and motor controllers. The motor controller provided was the WIRZ #203. Power and current consumption of the

motors can be seen in the table in **Error! Reference source not found.**A. This 12 inch diameter base was easily within the size constraints for the robot and was capable of supporting all of the necessary electronics and sensors with a maximum recommended payload of 35lbs. The price of this base was reasonable at $179. After researching the necessary parts it was found that this base would be only slightly cheaper to make with the help of the mechanical engineering department Worcester Polytechnic Institute and it was decided that the most efficient use of both time and money was to purchase the chassis from Zagros Robotics.

5. Microprocessor

The choice of microprocessor is important for overall robot design since it dictates the electrical design of the entire system as well as the complexity of the software which can run on the robot. Linux support for the architecture was also important, as an embedded operating system would provide structure and low-level functionality for the robot. The ARM family of microprocessors are well supported under Linux and enjoy nearly ubiquitous use in devices like cellular phones and portable music players. Widespread use and support for the embedded operating system of choice indicated that an ARM microprocessor would be a good fit. The ARM9 series was in the speed range of interest (100s of MHz), which would be fast enough to run the embedded operating system and leave extra processing power for the robot application. The processor must also expose enough general purpose I/O (input/output) pins and embedded peripherals to interface with the sensors required for the robot. With these constraints in mind, the microprocessor chosen for the robot was the AT91SAM9260 ARM9 from ATMEL, a 200 MIPS processor clocked at 180MHz.

5.1 Architecture and Features

The AT91SAM9260 has a host of built-in peripherals which make it a good fit for the design. The peripherals used by the robot include timer-counters, parallel I/O controllers, USB, SPI, MAC/PHY, and an AIC (advanced interrupt controller). A development board based on the AT91SAM9260, the SAM9-L9260 from Olimex (See Figure 6), was chosen to gain access to a useful sensor evaluation and test environment early in the development process.

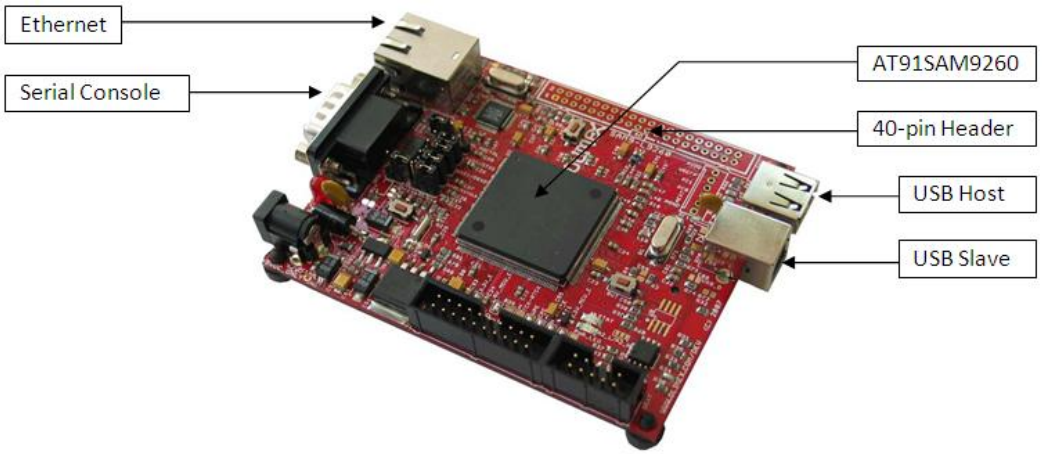


Figure 6 - SAM9-L9260 Development Board from Olimex

Fitted with an external 64MB of RAM and 512MB of flash memory, the development board is capable of running embedded Linux. The development board also houses an RJ-45 jack for network communications, an RS-232 port for console access to the operating system, and USB host and slave ports for programming the board and running USB peripherals, respectively. The development board also provides access to a 40-pin header, (of which 35 are I/O pins) connected to the processor.

5.2 Embedded Operating System

The embedded operating system allows the navigation and mapping code to communicate with the host computer application over Ethernet and manages local processes efficiently. The robot runs the Linux kernel version 2.6.28.4 with a Debian userland. The kernel has been upgraded several times throughout the development of the robot to include features such as USB serial emulation (used by the laser rangefinder) as well as support for the onboard real time clock (to supply the robot with a consistent date and time at startup). Additionally, the kernel source has been slightly modified to include userspace support for the SPI bus on which the gyroscope and accelerometer operate.

The operating system can be accessed through a serial console, as well as over an SSH connection. Most of the development is done by cross compiling source code on a host machine and then transferring the binary to the robot via SCP, but the robot has a sane build environment as well and simple code may be compiled there. More complicated code takes considerably longer to compile on the robot. The embedded operating system makes communication like this easy so that the focus can remain on developing device drivers and application code for the robot.

6. Custom PCB

A printed circuit board was designed as a daughter board to the development board housing the microprocessor using the board layout software CadSoft EAGLE. The original idea was to use the daughter board as a way to mount and use the MatchPort Wireless Radio which was chosen for the robot's wireless communications, but eventually expanded in scope to encompass many other functions. The design grew to include the power supply design, circuitry which had been present only on a prototype board, as well as connectors for the various sensor systems on the robot to allow them to interface with the processor through this board.

The circuitry on the daughter board was designed around an original prototype through-hole daughter board that was used for testing parts. The board features a two section power supply, controller circuitry for the ultrasonic array, and a level shifter between the processor and the SPI bus. Additionally, a number of connectors for the various sensor arrays were added, as well as the MatchPort Wireless Radio and associated RJ-45 connector. See **Error! Reference source not found.** for the daughter board connected to the development board and Appendix C for the complete daughter board schematic.

The ground and V_{dd} planes have been turned off to make the layout more visible in **Error! Reference source not found.**, but the bottom layer includes a V_{dd} plane (3.3V) and the top layer includes a ground plane. Data traces are 10mil. As mentioned above, the power supply traces were widened from the original design. The 12V lines are 100mil, and the 3.3V and 5V lines are 70mil. See Appendix D for a larger version of the layout depicted in Figure 7.

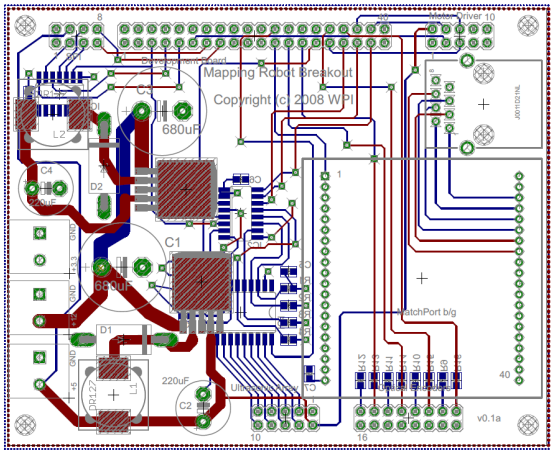


Figure 7 - Final Daughter Board Layout

Due to board size constraints imposed by the 'lite' version of CadSoft EAGLE (maximum size of 3.2"x4"), the board layout had to be fairly dense. The power supply was laid out first, and the rest of the parts added later. The layout of the power supply circuitry was modified slightly to fit the rest of the components and connectors. One of the biggest challenges was to carefully fit all of the connectors around the perimeter of the board. Both sides of the board have pads for parts as shown in Figure 8. The board was fabricated at BatchPCB. Upon arrival, the board was populated and tested.

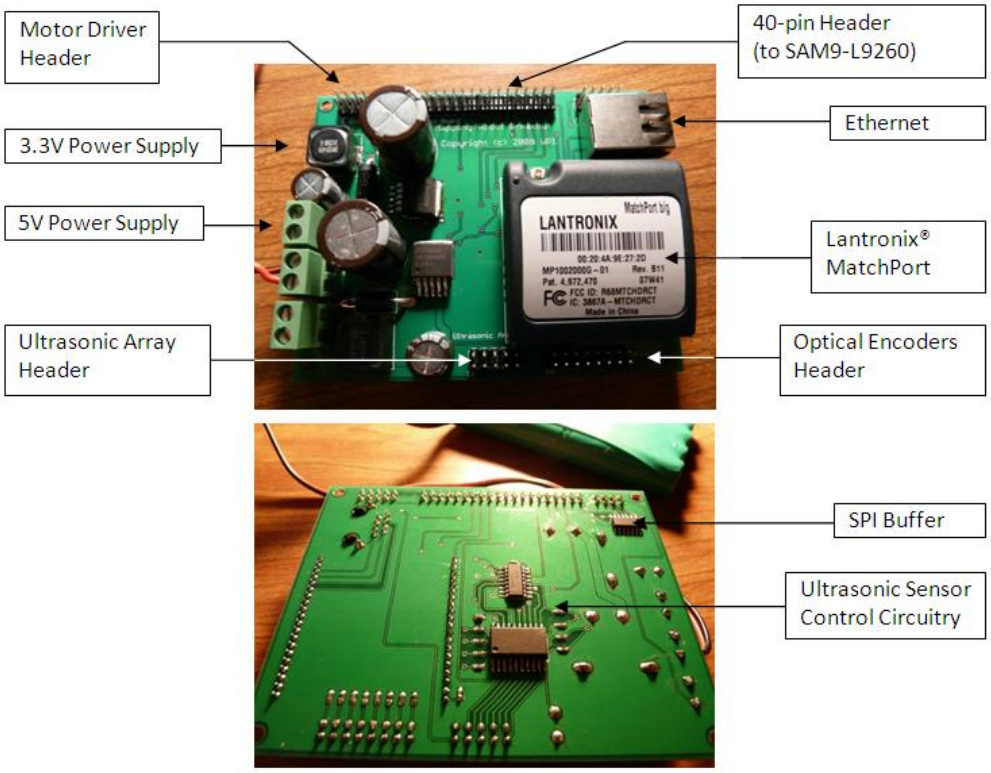


Figure 8 - Top and Bottom of Populated Daughter Board

Several changes have been made to the schematic since the original population of the board. Two of the GPIO pins which connect the processor to the optical encoders were changed to different GPIO pins during debugging of the optical encoders. The first modification was to cut the traces for these lines and install jumpers. The second modification was the result of testing the SPI bus and ultrasonic array together. The data line for the ultrasonic array was moved to a different GPIO pin, again the trace was cut and a jumper was installed. Lastly, a jumper was installed between the processor connector and one of the GPIO pins on the Lantronix radio module to be configured as a kill switch for the robot. The updated schematic appears in Appendix C.

6.1 Power Supply

The power supply for a mobile robot system is a critical component of the design. In most cases, mobile robots are powered by limited power supplies such as batteries or solar panels. With an inefficient power supply design, the battery life will be significantly shortened. Also, the power circuitry must be carefully designed for robots that are equipped with electric motors since they create noise in the power line which could cause damage to delicate components. Therefore, efficient power circuitry with proper voltage isolation or filtering is required.

6.1.1 Method

Robots require various voltage levels due to variety in components. This particular robot required three different voltage levels which were 12V, 5V and 3.3V. The list of each component's voltage requirements can be found in Appendix A. To meet the requirements of generating clean supply of various voltages, a power supply with two NiMH batteries and two switching regulators was designed. There were several options of DC-DC voltage regulation methods which were the fly-back regulator, linear regulator and the switching regulator. Each method has its own advantages and disadvantages which will be discussed in this chapter.

The fly-back regulator is a voltage conversion method that requires coupled inductors, or a transformer. The fly-back regulator provides voltage isolation which is useful for isolating the motor from other components. The disadvantage of the fly-back regulator is that it is far more complex than the other two voltage regulation methods discussed in this chapter and it is difficult to design a stable fly-back regulator [5]. Though the voltage isolation was an attractive property of the fly-back regulator, due to time constraints the decision was made not to implement the fly-back regulator, but instead install a separate battery pack for the motor in order to isolate the motor power line from delicate components.

The linear regulator is the simplest method out of the three DC-DC voltage regulation methods. There are many linear regulator IC chips available, usually with three pins corresponding to input, ground, and output. Simply connecting these pins to the battery and ground will produce the desired voltage on the output pin. Another appealing aspect of the linear regulator is that the output DC voltage is very clean, but the tradeoff is that it is a very inefficient method of regulation especially when the voltage drop and the load current are large [6]. For a mobile robot, it is desired that the power

circuitry to be efficient, therefore the use of a linear regulator for large voltage drop of 12V to 5V and 3.3V that will be needed for this application will not be suitable for this application.

The power loss of a linear regulator can be calculated using $\mathbf{Ploss} = (\mathbf{V_{in}} - \mathbf{V_{out}}) * \mathbf{I_{load}}$ Equation 1. It can be seen that greater the voltage drop, more power is wasted. Therefore for large voltage drops the linear regulator is not the best solution. On the other hand, if the voltage drop needed is small and the load current is small, the power loss is not as significant, making the linear regulator more appealing because of its lower price, easy implementation and clean supply.

$$P_{loss} = (V_{in} - V_{out}) * I_{load} \qquad \text{Equation 1}$$

A more efficient regulator is the switching regulator. Switching regulators typically have an efficiency of 80%~90% [6]. The disadvantages of using switching regulators are that they are expensive compared to linear regulators and more difficult to implement. Switching regulators use a pulse width modulation (PWM) that switches the source on and off feeding current little by little giving switching regulators its high efficiency. The switching regulator uses the characteristics of an inductor to down-convert the voltage which requires a careful layout design on the board. Though the switching regulator is harder to implement than the linear regulator, there are chips available that make the implementation much simpler. Due to its high efficiency a switching regulator was selected for the voltage regulation method.

6.1.2 Design

In order to achieve the goal of designing a high efficiency power supply, two switching regulators were implemented to produce the desired voltages, 5V and 3.3V. Two 12V NiMH rechargeable batteries were chosen as the power source for its lower cost and fairly high capacity of 2200mAh. To achieve voltage isolation for the motor, a separate battery pack was used for the motor. This choice was made over the fly-back converter because the use of a separate battery pack was a simpler solution than to implement the fly-back regulator. The choice of using switching regulators over linear regulators was made by looking at the amount of voltage drop required and the current consumption for each voltage level.

The table in Appendix A shows the current consumption of each component in the robot. As can be seen from the table, approximately 950mA is needed at maximum for 5V which is not suitable for linear regulators since the power loss would approximately be 2.09W. Therefore a switching regulator was selected for the 12V to 5V conversion. For the 3.3V conversion there were two options. The first option was to use a linear regulator from the 5V output of the switching regulator, and the second

option was to use a switching regulator from the 12V source. If a linear regulator from the 5V output is used, the power loss would have been 0.763W with devices running at their typical load current ratings and 1.30W at its maximum ratings. Though the voltage drop required is small, since there are many components that run on 3.3V drawing a fairly large amount of current, the power loss was still high. Therefore a switching regulator was selected for the 3.3V regulation as well.

The final design of the power supply contains two switching regulators converting a 12V source from a NiMH battery to 5V and 3.3V. Figure 9 shows the block diagram of the power supply circuitry. With this design, the efficiency of the power supply should be approximately 80%~90% according to the specification sheet of the switching regulator chip. Parts were selected carefully, and for the switching regulator IC, the LM2592HV series by National Semiconductors was selected. The LM2592 is capable of supplying 2A of current which was needed to have enough head room for each voltage level. Only four discrete components were required for operation of this chip which was also another reason why the LM2592 chip was selected.

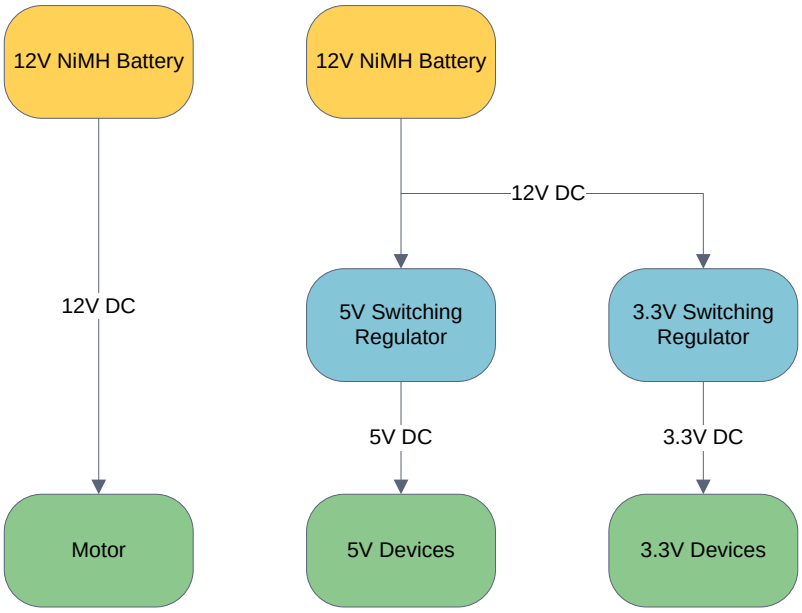


Figure 9 - Block Diagram of the Power Supply

The schematic of the power circuitry and the parts list can be found in Appendix B. The component values were selected carefully by reading the specification sheet of the switching regulator IC. Shielded inductors were selected to prevent magnetic field interruptions as much as possible. After

the schematic was drawn, the board layout was designed using EAGLE. The daughter board layout can be found in Appendix C. The component placements were carefully chosen since it can affect the performance of the switching regulator greatly. Thick traces were drawn to make sure the trace can withstand high current flowing through the power lines and screw type terminal block connectors were selected for the input and output connection for flexibility.

6.1.3 Battery Compartment

The battery compartment is an addition that was made to clean up the wiring of the robot as well providing safety features to the robot. An aluminum die cast casing was used for the battery compartment. The battery compartment contains fuses to protect the circuitries as well as a battery indicator to notify the user that the battery is running low. The battery compartment also has a three way switch that can switch the robot on, off and into charging mode. Figure 10 is a close up picture of the battery compartment.

Figure 10 Battery compartment close up

The outside switches are used to switch the system on, off and to charging mode. The batteries are chargeable without taking the batteries out by connecting the charger to the barrel plug input next to the switches and flipping the switch up. The two LED sequences indicate the battery life of each battery. When fully charged all LED's are lit, and when the battery starts to run low, the LED starts to turn off one by one.

6.2 Wireless Device

The robot design dictates a wireless communication device because autonomous operation would not be feasible with a tether to the host computer. The wireless radio should support standard wireless Ethernet (Wi-Fi) as well as the ability to bridge to a wired connection. Bridging to the existing development board wired connection would provide efficient integration of wireless communication into the design and take advantage of the embedded operating system present on the development board. The alternative was a device for which serial drivers would have to be written. While coding a serial-based device driver would have been possible, bridging to the existing wired connection generally boasted faster transfer rates, and was much easier to implement.

The wireless module chosen for the robot was the Lantronix MatchPort, which provided both serial and bridged connections as well as a simple web server and configurable I/O pins. The daughter board was designed with connectors to accept the MatchPort, along with an RJ-45 connector with internal magnetics. A standard cross-over Ethernet cable is used to bridge between the development board wired connection and the connection to the wireless radio on the daughter board. Configuring the radio amounted to upgrading the firmware over tftp, and configuring the radio via a telnet connection. After configuration, the wireless module bridges the wired connection to an ad-hoc wireless network which is the basis for wireless communication.

Figure 11 - Lantronix MatchPort b/g

7. Motor Controller

The motor controller, a WIRZ #203 provided with the base from Zagros Robotics, translates I/O signals generated from the microprocessor into the voltage levels required by the drive motors. The translation requires the connection of two power supplies. One 5V supply is required for the quad-H driver and inverter logic, a separate 12V supply provides power to the motors. By controlling the duty cycle of the

input signals (PWM), the microprocessor is able to dictate the voltage delivered to each of the drive motors independently, thereby controlling turn speed. Additionally, the motor controller accepts logic levels which dictate the polarity of the voltage delivered to each motor, thereby controlling their direction. A single timer-counter peripheral with two outputs on the microprocessor is coded to provide two same-frequency, different duty-cycle PWM signals to the motor driver. Two parallel I/O lines are used to control each motor's direction. Control of the entire drive system is outlined in the following snippet of pseudo-code:

```
driveRobot (direction, duty_cycle)

        Configure I/O pins
        Set direction pins (for each motor)
        Set max duty cycle (for each motor)
        Ramp up from 40% duty cycle to maximum
        Wait for event
        Ramp down to 40% duty cycle
```

The navigation code may access the duty cycle for each wheel independently, so the developer has direct control over the size and duration of turns as well as forward and reverse speed. Variations in motor construction that would ordinarily lead the robot astray while expecting to travel in a straight line may be compensated for by a combination of distance and orientation sensors.

Figure 12 - WIRZ #203 Motor Controller Mounted to Robot Deck

8. Sensors

Without an array of different sensors a robot becomes nothing more than a mobile computer incapable of accomplishing tasks based upon its current surroundings. In order to navigate and map its

surroundings, as well as determine its current location, the robot needed to be equipped with a suite of sensors able to provide the necessary data to perform the task of mapping. In order to be able to estimate its position the robot utilizes both optical wheel encoders and a homemade inertial measurement unit to allow dead reckoning techniques to be used. For the purposes of gathering mapping data a laser rangefinder was chosen, this was based on the fact that this is one of the most common ways to gather this type of data, as was discussed in the background chapter. Another common sensor to see on a robot is an ultrasonic sensor, an array of five of these were employed in order to provide the robot with the necessary information to navigate its immediate surroundings and to avoid obstacles. When used together all of these sensors provide the robot with the information necessary to complete the task of autonomous mapping.

8.1 Optical Encoders

Optical encoders are another means of collecting distance and angular direction information from the robot. The sensors that were used in this project were reflective optical encoders, meaning they emit infrared light and activate when a reflection is present. This allows the sensor to differentiate between reflective and non-reflective surfaces. In this project the optical encoders were used as a secondary measurement of the robot's current heading and the distance it has traveled. This information is sent to the processor where it can be checked against the data generated by the IMU. For instance, the IMU may indicate that the robot is in a state of constant acceleration when it is actually at rest. It is also possible that the accumulation of error in either the gyroscope or the accelerometer could cause erroneous data to be sent to the host. A simple test of the output from the wheel encoder can correct this and reset the output of the IMU to the proper values.

The Fairchild Semiconductor QRB1134 reflective optical encoders used for this project are a fairly simple device consisting of an infrared light emitting diode and a phototransistor placed adjacent to one another inside of a plastic housing, as shown in Figure 13 taken from the sensor's datasheet. A phototransistor is simply an NPN bipolar transistor with its base region exposed to sources of light. The light shining on the base region of the transistor acts as the voltage that would normally be applied to this location, causing the phototransistor to amplify changes in the amount of light currently hitting its surface.

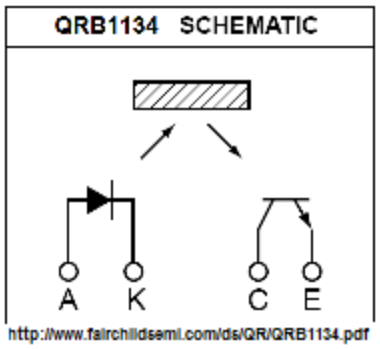


Figure 13 - QRB1134 operation

Reflective optical encoders work by reading IR reflections from a reflective (white) surface, such as a pinwheel pattern on the rotating wheel, like the one shown in Figure 14 which causes them to return a series of pulses. This information would be sufficient if the robot moved in only one direction. However, since the wheels are capable of spinning both backwards and forwards another implementation must be used to check the direction of rotation. For this application the use of a quadrature encoder is needed. A quadrature encoder works by combining the output from two encoders and comparing their respective logic levels, in fact for this project each quadrature encoder is made up from two QRB1134 sensors connected parallel to each other. Figure 14 shows the basic operation of a quadrature encoder.

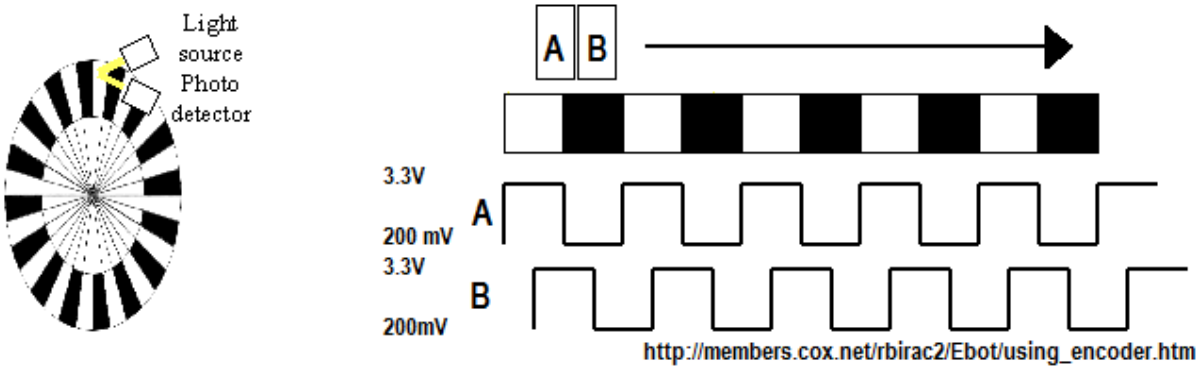


Figure 14 - Quadrature encoder operation

It can be seen in Figure 14 that for one direction of rotation sensor A will change states before sensor B, and rotating the other direction will cause sensor B to lead in the output signal. By comparing the current bit value to the previous bit value of this out of phase pattern the direction of rotation of the wheel can be determined. Table 1 below shows the binary outputs from this quadrature encoder for both clockwise and counterclockwise rotation.

Table 1 - Binary outputs for quadrature encoder

Clockwise		Counterclockwise	
0	0	0	0
1	0	0	1
1	1	1	1
0	1	1	0

From this table one can see that using this pinwheel scheme will cause only one bit value to change for each event. This is paramount to error reduction as it will keep the sensors from changing many bit values in too short of a time which can lead to unexpected values and cause deeply embedded errors when programming.

For their implementation each of the optical encoders was connected to a general purpose IO pin on the processor. These pins are registered as interrupt lines in the kernel by means of a kernel module (wpi_oenc) which acts as the driver for these devices. The module keeps a buffer of the transitions from the encoders, and the direction of those transitions. By multiplying this value by the width of one of the stripes on the pinwheel, 7.5 mm which is the smallest resolution available, the distance traveled can be calculated. Using equations taken from *A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators* by G.W. Lucas the value of theta, the angle in radians that the robot has turned, can be found. The equation used was fairly simple:

$$\Theta = (So - Si) / WB \qquad \text{Equation 2}$$

Where S_o and S_i are the distance the outer wheel traveled and the distance the inner wheel traveled respectively and WB is the wheel base of the robot (246.0625 mm). Also calculated are x and y coordinates indicating the robot's position relative to its starting location. These are calculated by multiplying the distance traveled by the cosine of theta and the sine of theta to get x and y respectively. After all of this information has been calculated, a global structure is incremented by these values, thereby keeping accumulated position data from the starting point. The heading of the robot, however, is not incremented, but saved as theta. Note that for the code theta needed to be scaled up in order for it to be stored as an integer in C. Storing all global data as the integer datatype simplifies communication between the robot and the host computer.

This data is then used to supplement the data coming from both the accelerometer and the gyroscope. It can be seen that by using these sensors in combination with the IMU circuitry much more

accurate results can be gathered providing a much more accurate estimation of the current position of the robot.

8.2 Accelerometer/Gyroscope

In all autonomous motion devices there is a need to know the relative location of the autonomous object. This is critical for the autonomous mapping robot as it will need to efficiently navigate through a room or building making sure it examines all possible areas. In order to get the necessary information about the robot's current to the navigation algorithm a set of sensors capable of providing both direction and distance information were needed.

In many cases distance and direction calculation can be done with simple odometry techniques, using encoders to count wheel rotation. With this data the current location can be determined trivially. However, this approach lends itself to several sources of error. Wheel slip is one of the main causes of error leading to inaccurate data measurements. If a wheel on the robot were to slip on the surface it were driving on extra rotations would be measured and the distance and/or direction data would be incorrect.

Another method other than odometry that avoids the issues brought on by wheel slip is the use of an inertial measurement unit (IMU). This is a device that incorporates an accelerometer and a gyroscope which work together to determine the current relative position of the robot. IMUs, while usually more accurate than simple wheel encoders, also suffer from accumulating error. This can be due to gyroscope drift which happens over time as a result of noise and even periodic drift from the rotation of the earth (Odometry: Calibration and Error Modeling, D. Bradley). However, there are ways of compensating for these kinds of error; see the section on the optical wheel encoders used for more information on this topic.

For the purposes of this project it was decided that purchasing a separate accelerometer and gyroscope would be the best way to increase our performance while keeping the budget to a reasonable level (commercially available IMUs can cost upward of $400). Table 2 shows a comparison made of some of the various accelerometers and gyroscopes that were considered for this project. The most important measure for the sensors was their level of sensitivity since this, combined with their resolution, yielded a measure of the overall performance of the sensor. Another important factor was the sensor's interface. It was preferred to find sensors with a digital output such as I^2C or SPI. Many of the accelerometers were

available with this option; however digital output gyroscopes were much less common. This was one of the factors that lead to the choice of the Melexis MLX90609.

Another challenge impeding the choice of the IMU sensors was the fact that they were nearly all offered only as SMT or LGA packages, making them difficult to use for prototyping. This problem was solved by choosing parts that have been broken out to easily accessible pins by Sparkfun.com. This will allow the use of these more precise parts during prototyping, as well as remove the need to purchase more solder boards that would only be used during prototyping and testing.

Table 2 - IMU parts decision matrix

Accelerometer	MFG	Resolution	Sensitivity	Voltage	Interface	Price
LIS3LV02DQ	ST	±2g/±6g	2g - 1024 LSB/g	3V	I2C/SPI	$43.95
			6g - 340 LSB/g			
SCA3000-D01	VTI	±2g	1333 LSB/g	3V	SPI	$44.95
Gyroscope	**MFG**	**Resolution**	**Sensitivity**	**Voltage**	**Interface**	**Price**
ADXRS150	Analog Devices	150°/s	12.5mV/°/sec	5V	Analog	$69.95
ADXRS300	Analog Devices	300°/s	5mV/°/sec	5V	Analog	$69.95
MLX90609	Melexis	±75°/s	12.8LSB/°/sec	5V	SPI	$59.99
		±150°/s	6.4LSB/°/sec			
		±300°/s	3.2LSB/°/sec			
IDG-300	InvenSense	±500°/sec	2mV/°/sec	3V	Analog	$74.95
Combo Board	**MFG**	**Resolution**	**Sensitivity**	**Voltage**	**Interface**	**Price**
ADXL203/ADXRS150	Analog Devices	±1.7g ±150°/s (10-bit ADC)	1024 LSB/g 15 mV/°/sec	5V in 2.5V out	Analog	$124.95
ADXL320/ADXRS150	Analog Devices	±5g (10-bit ADC)	1.95 mV/LSB	5V in 2.5V out	Analog	$109.95

The table above shows several possibilities for sensors including two boards with a combination of an accelerometer and gyroscope. These, however provided less performance than any of the parts looked at separately at a higher price. The two parts that were chosen were the Melexis MLX90609 angular rate sensor and the VTI Technologies SCA3000 accelerometer, each mounted on a board with break-out pins for more efficient implementation during prototyping. The MLX90609 has a resolution of ± 150°/sec. This measurement is of some importance as it gives an upper bound on how fast the robot will be able to turn and still get accurate data from the gyroscope. It also provides 6.4LSB/°/sec with this resolution which should be satisfactory for the purposes of this project. The SCA3000 has a rating of 7.501e-4 g/LSB. To put that in more practical terms this means that if the robot were to operate for 100

seconds continuously, it would accumulate a total error of +/- 120 feet, which is significantly less than other accelerometers that were considered for this role. As mentioned above, this error can be kept in check through the use of other means of odometry.

Communication with each of these devices was performed using serial peripheral interface (SPI). Programming for SPI use in Linux user space was different from any experience anyone in the project group had previously had with SPI. First it required the development of a new kernel module. An issue that arose from the use of the SPI interface was that there was only one SPI chip select line available from the development board. In order to work around this, the devices were selected using the single chip select line, a general purpose IO line, and a decoder as described in Table 3 below. As can be seen in the table when the accelerometer was selected by setting the GPIO line low, the outputs of the decoder would follow the input it received from the SPI chip select line; the same was true for setting the GPIO line high - selecting the gyroscope.

Table 3 - IMU decoder truth table

E (Always lo)	**A_0** (SPI_CS)	**A_1** (GPIO)	**O_0** (to accel.)	**0_2** (to gyro.)	**Notes**
HI	X	X	HI	HI	Cannot happen: E tied LO
LO	LO	LO	**LO**	HI	Accel. selected and activated
LO	HI	LO	**HI**	HI	Accel. selected not activated
LO	LO	HI	HI	**LO**	Gyro. selected and activated
LO	HI	HI	HI	**HI**	Gyro. Selected not activated

Once communications with each device were established they each needed to be calibrated to scale the output by the device's sensitivity and to remove built in biases. For the accelerometer this is done by placing the robot in a "zero-g" orientation (meaning the axes of interest are experiencing no acceleration forces) and subtracting out the bias; this value is saved as an offset that can be applied to all measurements coming in after this value is stored. The mlx90609 angular rate sensor was calibrated in a similar way, subtracting out the static offset from each measurement. In order to keep these calculated offsets in the correct range as the sensors experience both drift and temperature change the must be calculated every so often. In order to accomplish this calculation the robot will be stopped approximately every 3 meters. At this point laser rangefinder data will be taken and the axes of the

accelerometer will be zeroed. This will prevent unbound error growth in the sensors, thereby keeping the directional and distance data as accurate as possible.

Currently the inertial measurement unit system has been built and tested. It performs as expected with one down side: the communication with the individual devices via the SPI bus with the spidev Linux driver is not behaving as needed. The main problem stems from the fact that the chip select line, which activates each of the individual parts, remains in its active state for much longer (approximately two orders of magnitude) than is necessary to simply communicate with the device. These nearly 200ms communication cycles make scheduling the sensor firing order (discussed in chapter 9) impossible to accomplish in an amount of time that would be able to provide useful data. For this reason the IMU has been left out of the final design due to time constraints placed on the project. This will obviously have a negative effect on the positional data provided to the navigation algorithm since the error in the optical encoders cannot be kept in check by another system.

8.3 Laser Range Finder

The laser range finder is the primary mapping sensor for the robot. The most important factor when generating a map is the accuracy and precision of the data collected by the sensors. Laser range finder measure distance by measuring the time taken by a laser beam to be reflected off the target. The accuracy is usually on the order of millimeters, which is more than adequate for mapping purposes. Though laser range finders are suitable for mapping, there were some issues with them as well. Most laser range finders consume a lot of power and are heavy, which makes them difficult to install on a mobile robot. Therefore a light-weight and efficient URG-04LX laser scanner sensor manufactured by Hokuyo was selected for this project.

Figure 15 - Hokuyo URG-04LX Laser Range Finder (http://www.hokuyo-aut.co.jp/02sensor/07scanner/urg_04lx.html)

The URG was particularly designed for robot applications. At the Aichi Expo 2005 in Japan, more than half of the robots presented at the Expo used the URG-04LX [1]. The appealing aspects of this laser range finder are low power consumption and light weight. The required power source for this device is 5V DC , drawing only 500mA and weighing approximately one pound. The range of the URG sensor is 60mm to 4095mm with a field of vision of 240° incrementing

approximately by 0.36° per measurement. The accuracy of the sensor is ±10mm when the object is 60 to 1000mm away and 1% of the distance when the object is farther than 1000mm.

The URG can communicate with the host via an USB connection or an RS232 link. For this robot the USB connection was used because it provides faster transfer rate. The URG-04LX laser range finder communicates with the host through Hokuyo's SCIP communication protocol. Hokuyo provides a library with API functions to simplify initialization and communication with the sensor.

The laser range finder driver was written using the API to initialize the sensor and to perform a full range scan. Each full range scan was stored in a buffer containing 726 data points of magnitude data and angle measurements that are retrieved by another API function call that converts indices to radians. Before the laser range finder driver was written, the device was tested using vmon, an application provided by the manufacturer.

8.3.1 Sensor Evaluation

The Hokuyo laser came with a CD containing evaluation software. This included an application called vmon which could be run in a Windows environment. Installing the driver for the laser connection was simple and a README from Hokuyo provided all of the necessary information to use the software. The vmon application shows a visual representation of the data that the laser is sending back in real time and can also be used to export a directory of Microsoft Excel spreadsheets containing numerical data from ten sequential sweeps of the laser. Figure 16 shows the visual representation of the data given by vmon

Table 4 shows a sample of the corresponding output data values and a scatter plot of the x and y coordinates.

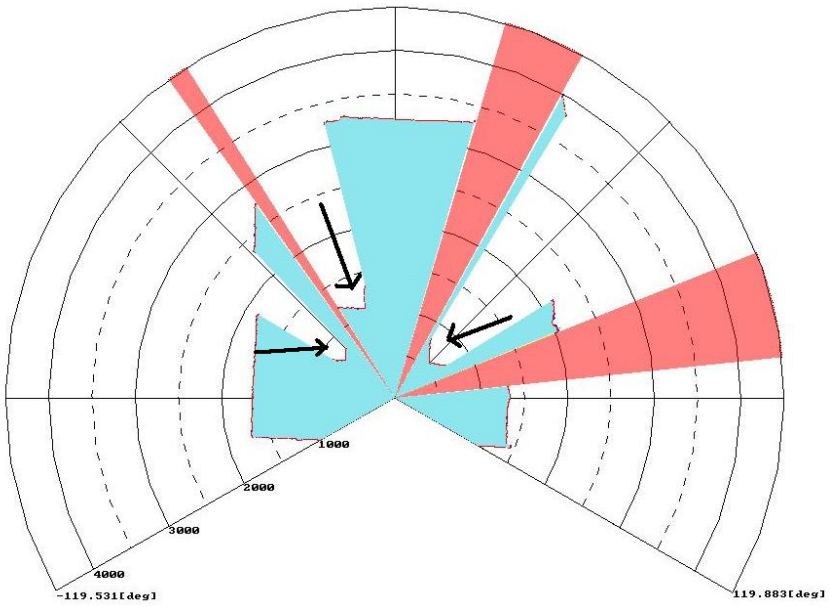
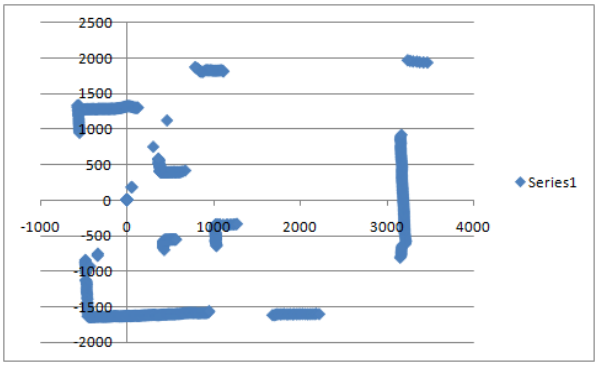


Figure 16 - Visual data provided by vmon application

In the image above the laser was placed at the center of a grid laid out on a table in the lab. Blue areas represent information received by the laser while the pink areas show where the laser's range was exceeded. The three arrows indicate boxes that were placed in the laser's scanning range for testing purposes. In an x-y coordinate system the corners of these boxes were placed at: (40cm, 1000cm) (-40cm, 40cm) and (60cm, 40cm). Note that in the image above the view is as if the laser were being looked at from below so, although two boxes appear to have negative x coordinates, they were actually to the right of the laser when looking in the same direction it was facing. The actual measurements returned in vmon for each box respectively were: (997.4, 38.0cm) (41.9cm, 41.9cm) and (56.0, 42.3). By analyzing several more data points from many tests it was found that this laser is indeed accurate to within the +- 10mm as specified by the manufacturer. Table 4 shows some sample data and a scatter plot of all of the x and y coordinates from the test described above.

Table 4 - Laser rangefinder data points and scatter plot

Angle (rads)	X-coord. (cm)	Y-coord. (cm)	Angle (deg)
-0.80994	530.5326	-557.236	-46.4063
-0.80381	536.093	-556.202	-46.0547
-0.79767	542.4289	-555.908	-45.7031
-0.79153	551.1003	-557.905	-45.3516
-0.7854	557.6243	-557.624	-45
-0.77926	563.3831	-556.511	-44.6484
-0.77313	570.6523	-556.815	-44.2969
-0.76699	1685.746	-1624.8	-43.9453
-0.76085	1705.818	-1624.08	-43.5937
-0.75472	1723.619	-1620.98	-43.2422



The format of the data in the above table is: Angle in radians, x coordinate, y coordinate, angle in degrees.

The data in the table above corresponds to -45° where the edge of one of the boxes can clearly be seen in the highlighted row. By making a simple scatter plot of the x and y coordinates in Microsoft Excel a crude 2-dimensional map can be constructed from the data provided by vmon. In the scatter plot it can be seen that there are a few extraneous data points that do not correspond to an actual object in the laser's view. This was most likely due to the fact that the angle that those points are at was an area in the lab where the laser was pointed at a lab bench with a very inconsistent surface.

8.4 Ultrasonic Sensor Array

The SRF05 ultrasonic sensor is capable of interfacing with the host processor in one of two modes. The first mode separates the input and output data, whereas the second mode allows for the use only one line for bidirectional communication (See Figure 17). The second mode was chosen to reduce the wiring complexity, and to reduce the number of built-in timer-counter peripherals necessary from the ARM9 microprocessor.

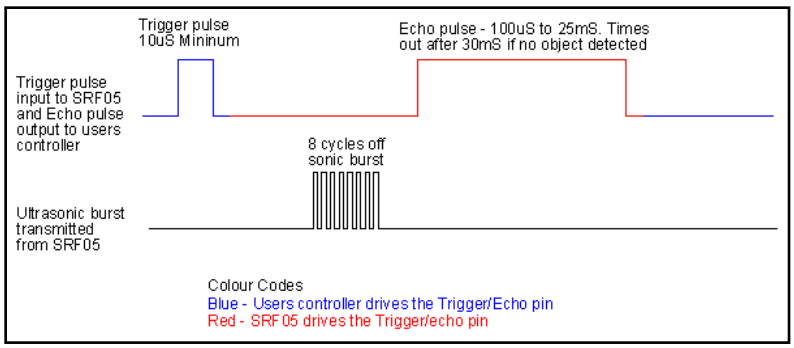


Figure 17 - SRF05 Timing Diagram, Mode 2

Since multiple sensors are necessary, the data lines from each of 5 sensors are multiplexed to a single timer-counter on the ARM9 microprocessor. Furthermore, the 5V logic levels from the ultrasonic sensors must be level shifted to 3.3V logic levels. This is accomplished with a 4051 analog multiplexer to multiplex the bidirectional data between the sensor array and the microprocessor as well as a 5V tolerant bidirectional buffer, which acts as a level shifter. See Appendix C for the circuitry designed to accomplish this task. The code to retrieve distance data from the sensor array is outlined in following pseudo-code snippet:

```
pollSRF(sensorID)
       Configure IO pins
       Set inhibit pin low (multiplexer active)
       Set address lines (Select the proper sensor)
       Set direction pin (Processor -> Sensor)
       Send an 11µs pulse
       Set direction pin (Sensor -> Processor)
       Measure received pulse
       Set inhibit pin high (multiplexer in active)
```

The ultrasonic sensors are mounted on the underside of the middle deck of the robot according to Figure 18. This placement allows for easy wall-following, with two sensors at each side, and good obstacle avoidance in front of the robot.

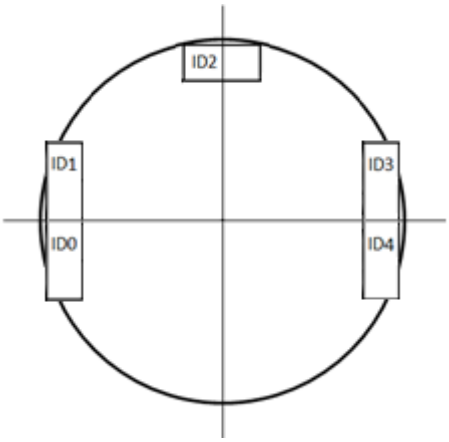


Figure 18 - Ultrasonic Sensory Array Layout, Bottom View

Each sensor is mounted to the deck of the robot with a custom-made bracket. The brackets are composed of aluminum angle stock with the appropriate holes drilled to match the sensors, as well as a triangular hole pattern for fastening to the deck of the robot. See Figure 19 for an image of the robot with all 5 sensors installed.

Figure 19 - Robot Base with Ultrasonic Sensor Array

8.4.1 Sensor Evaluation

The SRF05 ultrasonic sensor is responsible for obstacle detection and robot navigation. Though the accuracy of the ultrasonic sensor is not a critical element for mapping purposes, it was still important to

understand its capabilities and any potential problems it might introduce into the overall design. Two experiments were conducted to examine the accuracy, precision and angular range of the sensors. The first table in Appendix shows the results of these experiments.

In order to examine the accuracy of the ultrasonic sensors a piece of tape was used to mark the distance from the sensor to the wall. Next, an ultrasonic sensor was placed on each distance mark and a measurement was taken using a simple program that triggered the ultrasonic sensor one hundred times sequentially. The program output the measurement of each of the hundred runs, and also output the average of the hundred runs after terminating. As the ultrasonic sensor was run one hundred times each of the measurements output was examined to observe any possible glitches or unexpected measurements in the output. The numbers in the table are the average of the one hundred measurements made. In order to measure the consistency of the ultrasonic sensor, the program was run ten times for each of the measurements made.

It was found that the ultrasonic sensor is very accurate for most distances. Though accuracy is important, the precision and consistency of the sensor was more valued for obstacle detection and navigational purposes. As long as the sensor is consistent it is possible to make some adjustments to the code or to the sensor itself to account for measurement error. The sensor had the least precision at 5cm, where the maximum value and the minimum value ranged 3.18mm and the least accuracy at 30cm where the percent error was 1.3%. This was high enough precision and accuracy for obstacle detection and navigation purposes for the robot. It was also found that as the sensor gets closer to the wall, its accuracy decreases. Up to 60cm, the percent error of the sensor was below 0.3% but at 50cm it increased to 0.785%. Overall the SRF05 ultrasonic range sensor meets the requirements for accuracy and consistency for this project.

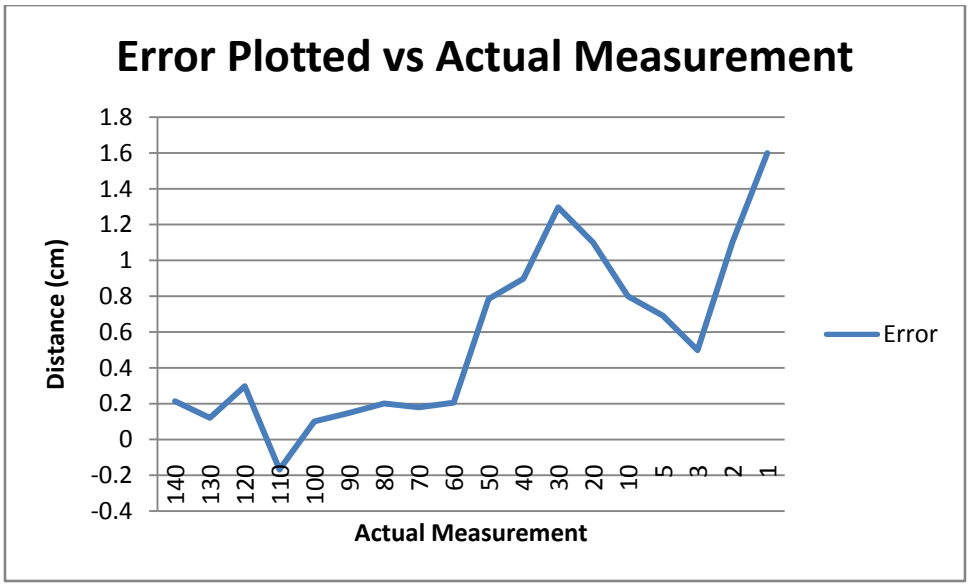


Figure 20 - Error of the Ultrasonic Sensor Plotted

Figure 20 shows the error of the ultrasonic sensors plotted versus the actual measurement. It can be seen that the maximum error is 1.6cm at 1cm. Up to 60cm the ultrasonic sensor is accurate up to +-0.2cm. As the object gets closer the sensor becomes less accurate.

The angular range of the ultrasonic sensors is important data for this application since it determines the placement of the sensors on the robot. In order to measure the angular range of the ultra sonic sensors the robot was placed 80cm away from the wall and a board was slowly moved at a fixed distance in front of the robot. The angle at which the sensor detected the board was measured using a protractor and a string. The angles with the board were measured at 20cm, 40cm and 60cm. When the measurements were made, it was found that there was a region where the sensors detected both the board and the wall. Therefore it was decided to make two measurements where the sensor stably detected the board (inner bound) and where the sensor stably detected the wall (outer bound). The table in Appendix shows the result of this experiment. From these results it can be said that objects within plus or minus ten degrees from the ultrasonic sensor will be detected consistently.

For most of the measurements taken the performance of the ultrasonic sensors was better than expected. By conducting this evaluation, the capabilities of the ultrasonic sensors as well as their problems were better understood. The experimentation was valuable since it opened up many options. It was found that the sensors are fairly accurate and consistent, which yielded the option of

supplementing the laser range finder information for the mapping process with data gathered by the ultrasonic sensors.

9. Software

The software of this system is comprised of three main parts, which are the communication code, robot code and the host application. The communication code is used to transmit commands and data between the robot and the host application. The communication code contains the server code and the client code where each code lies in both the robot code and the host application as shown in Figure 21 The robot code contains the server/client code, sensor timer scheduler, sensor drivers and the navigation algorithm. The host application holds the server/client code and the graphical user interface. Figure 21 shows the basic code flow of the entire system.

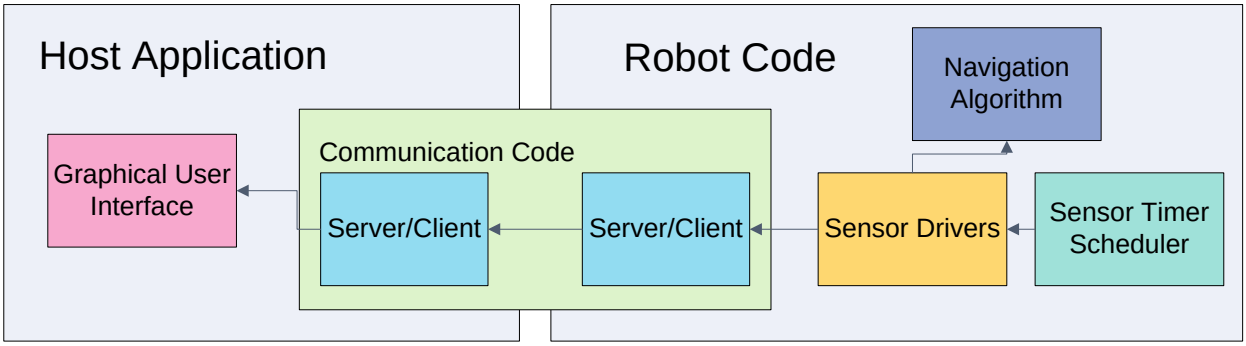


Figure 21 - Software Architecture Block Diagram

9.1 Communication Protocol

The communication protocol is the bridge between the robot and the host machine. The protocol uses the sockets API and is designed around using a message ID to identify different types of data. Important data structures are stored globally in the robot application. These include a structure for holding each of the major sensor data, as well as a combined structure for holding the adjusted heading and position data, calculated on the robot side. Each structure has as the first element an integer specifying the message ID.

When the structure is sent over the network, the first piece to arrive on the receiving end is the message id. The receiving end can then use a lookup table to determine the message length and the appropriate storage structure. The receiving end waits in a loop until all of the data has been sent.

Since the size of the data structure is known from the lookup table, the loop can use the comparison of the total number of bytes received to the total number of bytes expected. As the data is being received, it is stored in a buffer with size determined by the message ID. When the buffer has been filled (all expected bytes have been received), the buffer is then copied to an allocated data structure which matches the structure from the sending side. The sending side also loops through the send process to allow for the data to be split over multiple packets.

9.2 Robot Code

The robot code is separated into device drivers, the timer scheduler, communications, and navigation. Each piece is crucial to the autonomous operation of the robot. The device drivers are the lowest level code, and are responsible for abstracting the physical world, as experienced through sensors (ultrasonic sensors, optical encoders) and actuators (motor driver), into clean, efficient function calls where the data can be made available to the application layer. The timer scheduler sits between the driver and application, and is responsible for keeping accurate time between sensor reads, interrogating sensors at known intervals, and scheduling the sending of map data to the host machine.. The navigation code is on the application level and is responsible for higher-level decision making based on the sensor data. Communication code is also used by the application level, which allows the robot to communicate with the host machine.

9.2.1 Device Drivers

The device drivers communicate with the development board hardware in a number of ways. The ultrasonic sensors and motor controller drivers map out a page in user space from /dev/mem and manipulate this page in order to control the contents of the microprocessor registers. The optical encoders use a custom kernel module (wpi_oenc) which treats the encoders as interrupt sources. The kernel module updates the direction and number of transitions on each new interrupt and buffers the information until it is read through an ioctl call. The laser rangefinder uses an API based on the cdc-acm kernel module, which emulates a serial connection over USB. The gyroscope and accelerometer both use SPI. The SPI bus is controlled via a custom kernel module which allows for access to the Linux kernel SPI API. Each major device or sensor has its own source and header file.

9.2.2 Timer Scheduler

Sensor reads are scheduled with the custom kernel module (wpi_tcsched). This module controls a timer/counter which it uses to generate interrupts. On an interrupt, a flag is set and then copied from

kernel space to userspace. The main code waits for a flag change and then services the appropriate device based on the flag. Since the time of the sensor reads in known, data can be integrated with respect to time, which is necessary for calculating position from acceleration data (accelerometer) and heading from angular velocity data (gyroscope). Additionally, the scheduler can be used to finely adjust the frequency of sensor reads to strike a balance between gathering all the necessary information, but not wasting system resources.

9.2.3 Navigation Algorithm

The navigation algorithm is needed for the robot to navigate through the building and to avoid obstacles. The robot uses a "wall follower" algorithm which is a popular navigation algorithm for robots navigating in an unknown environment. The "wall follower" is an algorithm that has the robot follow the wall to its right (or left) until it returns to its original position. This algorithm works in any environment as long as all the walls are connected. The navigation algorithm for this project was written based on the ultrasonic sensor data and the wheel encoder data.

The navigation algorithm lies in the ultrasonic sensor polling service routine. The algorithm starts by polling the ultrasonic sensor depending on the sequence array which holds ultrasonic sensor IDs in the order to be fired. The sequence array was necessary to avoid firing physically close ultrasonic sensors too closely in time, a condition which is known to create false readings when one sensor detects echos generated by another sensor. Depending on which sensor was fired, the algorithm calls on a check function. There are four check functions for this algorithm which are bump check, right check, parallel check and center check.

The bump check function is called whenever the front sensor (sensor ID 2) is fired. This function checks the value returned by the front ultrasonic sensor, and stops the robot if the distance measured is closer than 90cm. After stopping the function decides on which way for the robot to turn by checking the previous sensor values on the right and left of the robot. The right check function is called on whenever the right sensor (sensor ID 0) is fired. This function checks if the right wall is lost or not by checking if the sensors return value. If the right wall is lost, it tells the robot to turn right to follow the right wall.

The parallel check function is called when both of the right sensors are fired (sensor ID 0 and 1). This function checks if the robot is parallel to the wall to its right by comparing the values of the two right sensors. If the difference between the two sensors is large, the robot adjusts itself until the robot is

parallel to the right wall. The center check function is called when the left sensor (sensor ID 4) is fired. The center check function checks whether the robot is centered relative to the walls on its side. This function is only called when the robot is parallel to the right wall and when the robot sees a wall to its right and left. When the function is called the robot takes the right and left ultrasonic sensors measurement to determine the center point. The robot then checks if its current position is near the center point. If it isn't, the robot moves toward the center.

The individual functions of the navigation algorithm works properly, but for certain situations the navigation algorithm seems to fail. Further testing and debugging will be needed in order to determine the exact reason for the failure in certain situations. The navigation algorithm can be improved and one whole research project can be spent on writing an effective navigation algorithm for this robot. If time allowed, it would be interesting to use the laser range finder data for navigational purposes along with the ultrasonic sensor and the wheel encoders.

9.3 Host Application

The host application was designed to receive data from the robot and to display it in a manner that is easy to comprehend for the user as well as sending commands to the robot when necessary. The host application was programmed using a combination of C and Tcl/Tk. Tcl is a scripting language and Tk is a popular Graphical User Interface (GUI) toolkit used with Tcl. The host application contains four parts, server code, client code, data processing code and the GUI, each running on its own thread. The server runs constantly in the background waiting for data from the robot. Once data is received, the server notifies the data processing thread. The data processing thread then processes the data so that the GUI can properly display the data in a useful fashion. Figure 22 is a flow diagram of the host application.

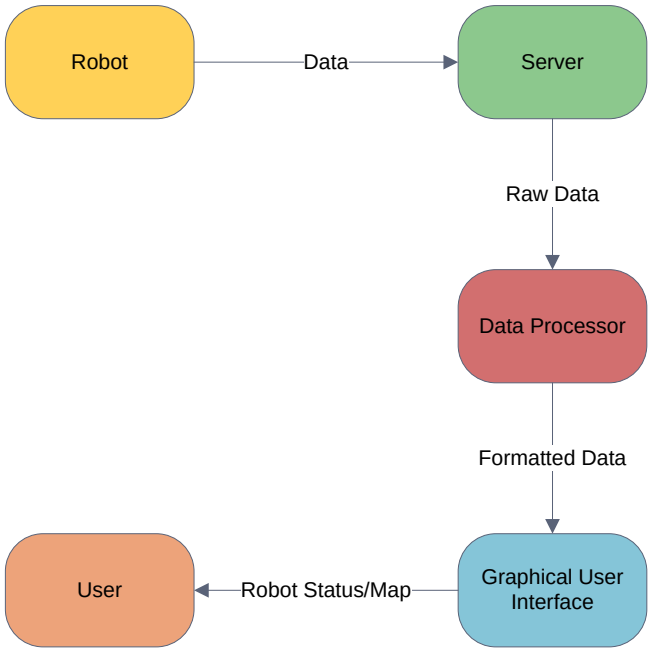


Figure 22 - Code Flow Diagram of the Host Application

The server code, client code and the data processing code were written in C and the GUI code was written in Tcl/Tk. Though it was possible to perform the data processing in Tcl, it was decided to add the data processing thread because C is faster at performing calculations. The data processing thread and the Tcl/Tk code communicates through linked variables. Tcl provides a useful API function that links a variable in C to a variable in Tcl. Whenever a change on one side is detected; the changes are reflected on the other side.

The server is responsible of receiving the data from the robot and storing it in the appropriate data structure. Each data received is headed with a message ID which is used to identify what type of data is being received. Once the data is stored in the appropriate data structure, it notifies the data processor. The data processor then takes the message ID to call on the appropriate function to calculate the values needed for the GUI to display the information properly. The data processor stores the calculated values in the linked variables and then notifies the GUI that the data is ready to be displayed for the user.

The GUI provides the user an easy way to view the robot's current status and the data being collected as well as providing the user some control over the robot when needed. The GUI displays the map, a log of robot's action and the robot's current status. The GUI runs a custom event handler where it traces a variable and whenever a change is detected, it updates the proper data depending on the

message ID. The message log is updated every time new data is received. Figure 23 is a screenshot of the GUI.

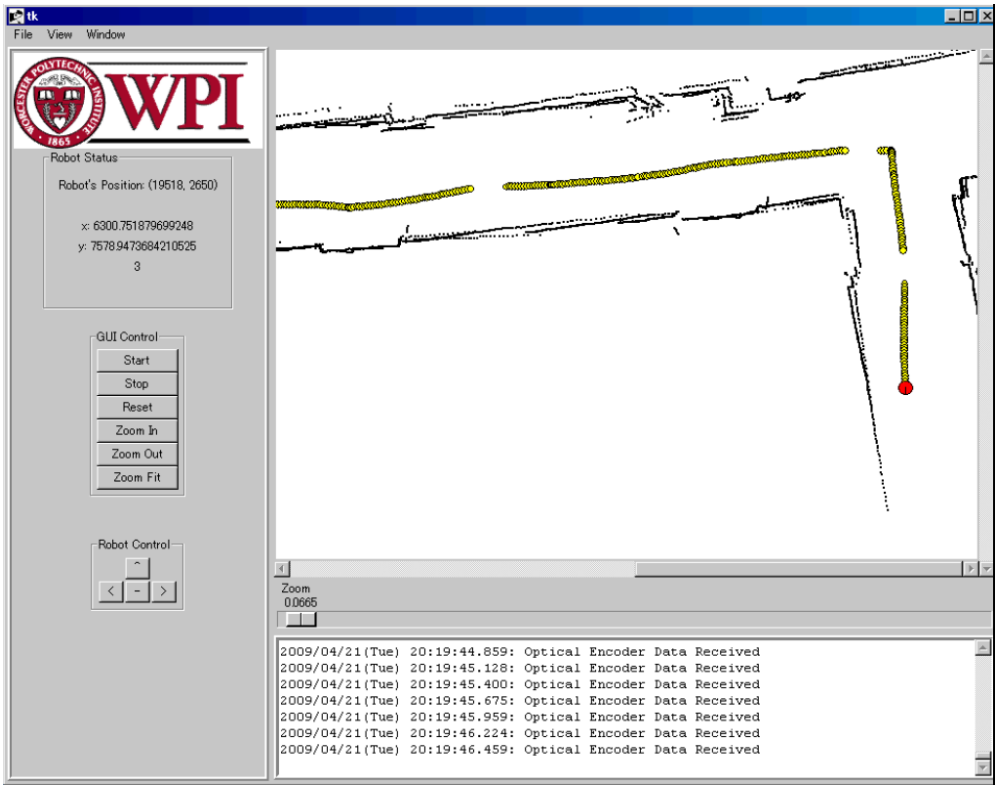


Figure 23 - Screenshot of the Graphical User Interface (this figure will be updated later)

The red circle in the figure above indicates the latest position of the robot and the small yellow circles indicate previous positions of the robot. The line in the red circle indicates the heading of the robot and the black dots are objects detected by the laser range finder. The robot's position is displayed in x-y coordinates in millimeters where (0, 0) is defined as the original start position of the robot and zero degrees defined as the robot's heading at the start. When the zoom is set to 1, each pixel on the map display corresponds to 1mm. The GUI allows you to click on the laser data points to view the coordinate of the data point.

10. Final Results

At the beginning of this project the goal was to build a prototype of a robot with the ability to autonomously navigate the interior of a building and produce a floor map of the areas it navigated through. After integrating all of the various sensors and running numerous tests on the robot to

calibrate and debug the navigation code the final results of the project were able to be determined. Figure 24 shows the robot fully assembled in the same state it was in when it ran its final test.

Figure 24 - Fully assembled robot

10.1 Capabilities and Test Results

The overall outcome of this project was a positive one. The final prototype of the robot is capable of creating a two-dimensional map of its surroundings as it navigates autonomously through them. This allowed for a partial fulfillment of the original goal which was set for this project: to map the third floor of Atwater Kent Laboratories at WPI. Figure 25 shows a screen shot of the final map that was created as well as an actual floor plan of the area for comparison.

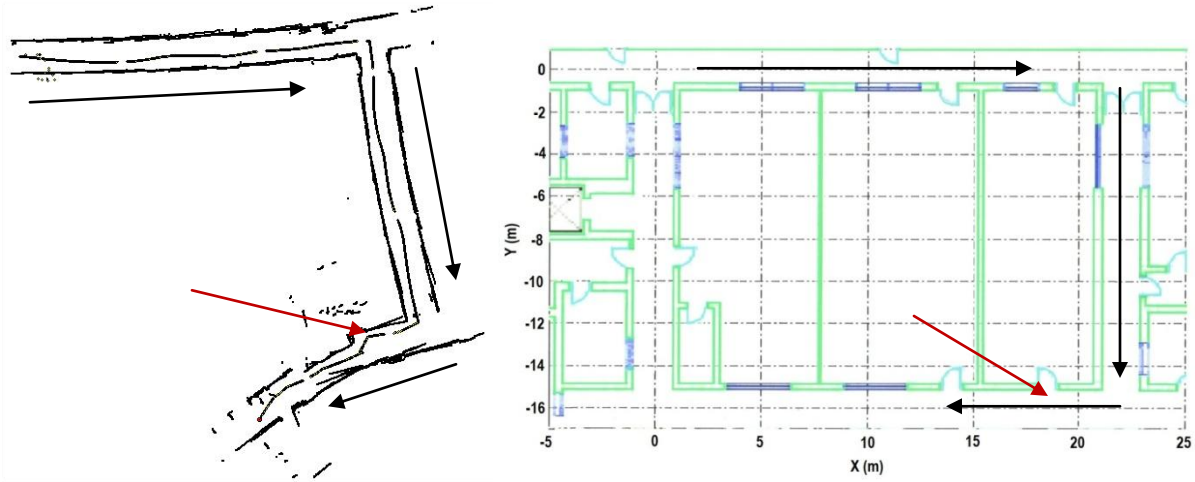


Figure 25 - Left: robot-generated floor plan; right: actual floor plan

As can be seen from the above images the robot was unable to complete a map of the entire third floor. This was due to what is believed to be a combination of some minor bugs in the navigation algorithm and a build up of error in the position data provided by the optical encoders. In order to make the above map the robot followed the directions shown by the black arrows. The slight curvature in the hallways shows the skew that was being accumulated by the position data. At the point indicated by the red arrow the robot became disoriented as it found itself next to an open door and with a chair immediately in front of it; this caused the robot to spin in a 360° circle before continuing. This circle caused the state of the navigation algorithm to change eventually causing the robot to lose its sense of direction, rendering the rest of the map unusable. The performance of the robot in this final test, however, showed that despite not having each individual module of the robot working perfectly; it was still able to do what it was originally built to do, with some limitations. Had the inertial measurement unit been fully implemented, allowing more reliable distance and angle data to be available to the navigation algorithm it is believed that the results would have been far better.

Other positive results that came out of the final test of the robot were that the navigation algorithm's centering functionality worked very well in the first two hallways traveled, as can be seen from the small amount of wavering in the series of points running through the corridor that represent the robot's path. Also effective was the functionality of the navigation algorithm to try to keep the robot parallel to the right wall as often as possible. This too worked very well for the first two hallways traversed.

10.2 Quantitative Performance Analysis

The effectiveness of this autonomous mapping system can also be measured in a quantitative manner. Using the X and Y coordinates stored from each point of the laser scan it becomes possible to compare the generated map to the actual dimensions of the floor.

The accuracy of the laser rangefinder was probably more than would be needed for a simple floor plan mapping system. The problems that arose were due to the inconsistencies in the position data as mentioned above. The image below describes some detailed measurements from the generated map.

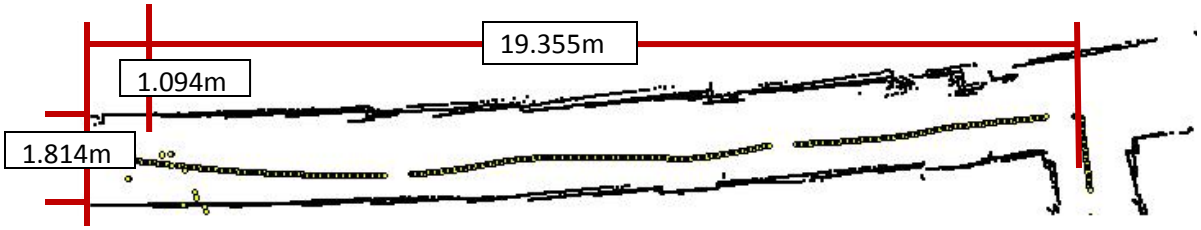


Figure 26 - Quantitative analysis of a hallway in the map

In the image above it is shown that the length of the hallway traveled by the robot was measured in the graphical user interface to be 19.4m long. This distance, in reality is very close to the actual length of the corridor which is almost exactly 20m. The calculation of the width of the hallway was also quite accurate, coming out to be 1.814m; while the actual width of the hallway at this location was measured as 1.828m. The final measurement needed to tell the whole story was a measurement of the accumulated positional skew which causes the hallways to appear somewhat curved in the maps that are generated. In order to measure the skew the difference was taken between the x-coordinate at the start of the hallway and the x-coordinate at the point where the robot made its right turn; these two should ideally be approximately equal. Using this technique the skew induced by the inaccuracy of the optical encoder-provided position data was found to be about 1.1m. When divided into the total length traveled down this hallway that yielded a skew percentage of about 5.6%which is quite high for such an application. Again, however, with a working IMU it is believed that this error could be drastically decreased. This amount of skew does not present too much of a visual disruption to the map for a single hallway, however when added together over the course of trying to map an entire floor it becomes much more significant.

Regardless of the small setbacks that were experienced while attempting to accomplish the goal of mapping the entire floor it can be seen from the results presented in this chapter that the overall outcome of this project was a positive one. The original goal of building a robot with the ability to

autonomously map the interior of a building was met. The next chapter will discuss some recommendations for possible future MQP groups to improve the overall design of the robot and to help it become a more effective search and rescue system.

11. Recommendations

Given the general success of this project it is only natural that more work be done on it in the future to add an extended set of capabilities to the existing system. Before this can be done some minor work would need to be completed to improve the current design, including laying out a custom PCB for the processor and also getting the serial peripheral interface with the IMU parts to work as needed with the system to prevent to decaying quality of the positional data. Once these tasks are complete many opportunities for added functionality will be available. Some recommendations for future work on the robot are described in this section.

This project was originally meant to be developed into a robot capable of performing complex search and rescue operations to help protect human lives. One of the most important features that should be added next to such a system would be a human detection capability and alert system. This would allow the robot to navigate itself through an area to locate a possible victim and alert the operator, while also providing a map of the area it has traversed leading to the victim.

After implementing the ability to locate a victim of some sort of hazardous situation another improvement to the robot that would greatly improve its practicality would be to redesign the drive capabilities to allow the robot to traverse more difficult terrain. One idea would be to attempt to make a two-wheeled balancing design that would allow the robot to climb over some obstacles.

With the addition of the proposed minor upgrades, along with the major capability extensions suggested in this chapter the robot would have the opportunity to become a very practical and useful device. All of these capabilities would allow this autonomous robot to become a system with the possibility of saving human lives or, at a minimum, reducing risk to search and rescue personnel in hazardous situations.

Appendix A – Load Currents

Table 5 - Load Current of Each Device

Device	Voltage	Typical Load Current	Max Load Current
AT91SAM9260	1.8V	-	100mA
Motor Driver	5V	-	70mA, 33mA
Gyroscope	5V	16mA	20mA
Laser Range Finder	5V	500mA	800mA
Ultrasonic Sensor	5V	-	4mA
Wireless Module	3.3V	360mA	550mA
Accelerometer	3.3V	-	650µA
Optical Encoders	3.3V	20mA	50mA
Bidirectional Buffer	3.3V	-	10µA
Analog Multiplexer	3.3V	-	40µA
Dual RS232 Xceiver	3.3V	8mA	12mA
Switching Regulator		0.9mA	1.4mA
Voltage	1.8V	3.3V	5V
Max	100.01mA	762.7mA	943mA
Typ		448.7mA	639mA

Appendix B – Power Supply

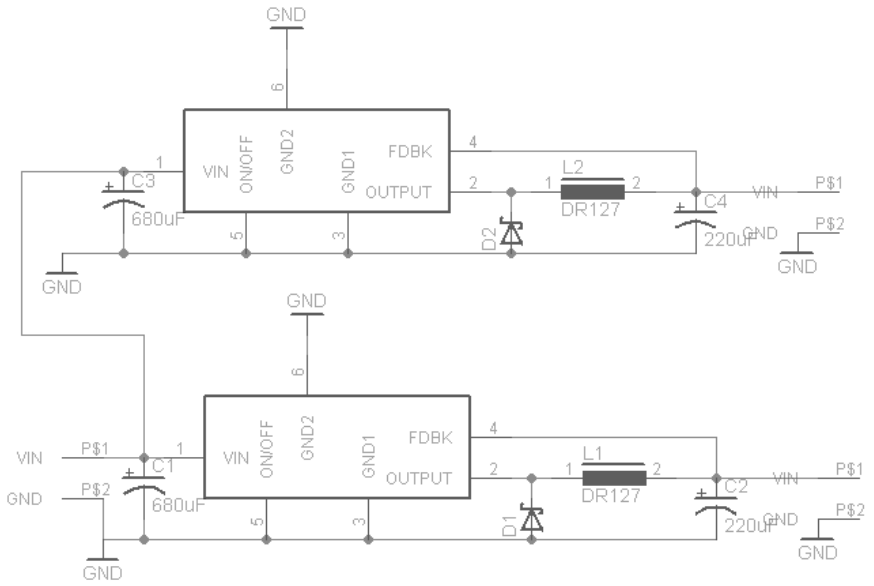


Figure 27 - Schematic of the Power Supply

Table 6 - Power Supply Parts List

#	Description	Value	Manufacturer
C1	Input Capacitor	680µF 50V	Nichicon
C2	Output Capacitor	220µF 25V	Nichicon
L1	Inductor	33µH	Coiltronics
D1	Diode	3.3A 60V Schottky	Vishay IR
	Switching Regulator	DC-DC 5V Step Down	National Semiconductor
	Switching Regulator	DC-DC 3.3V Step Down	
C3	Input Capacitor	680µF 50V	Nichicon
C4	Output Capacitor	220µF 25V	Nichicon
L2	Inductor	50µH	Coiltornics
D2	Diode	3.3A 60V Schottky	Vishay IR
	Switching Regulator	DC-DC 3.3V Step Down	National Semiconductor

Appendix C – Daughter Board Schematic

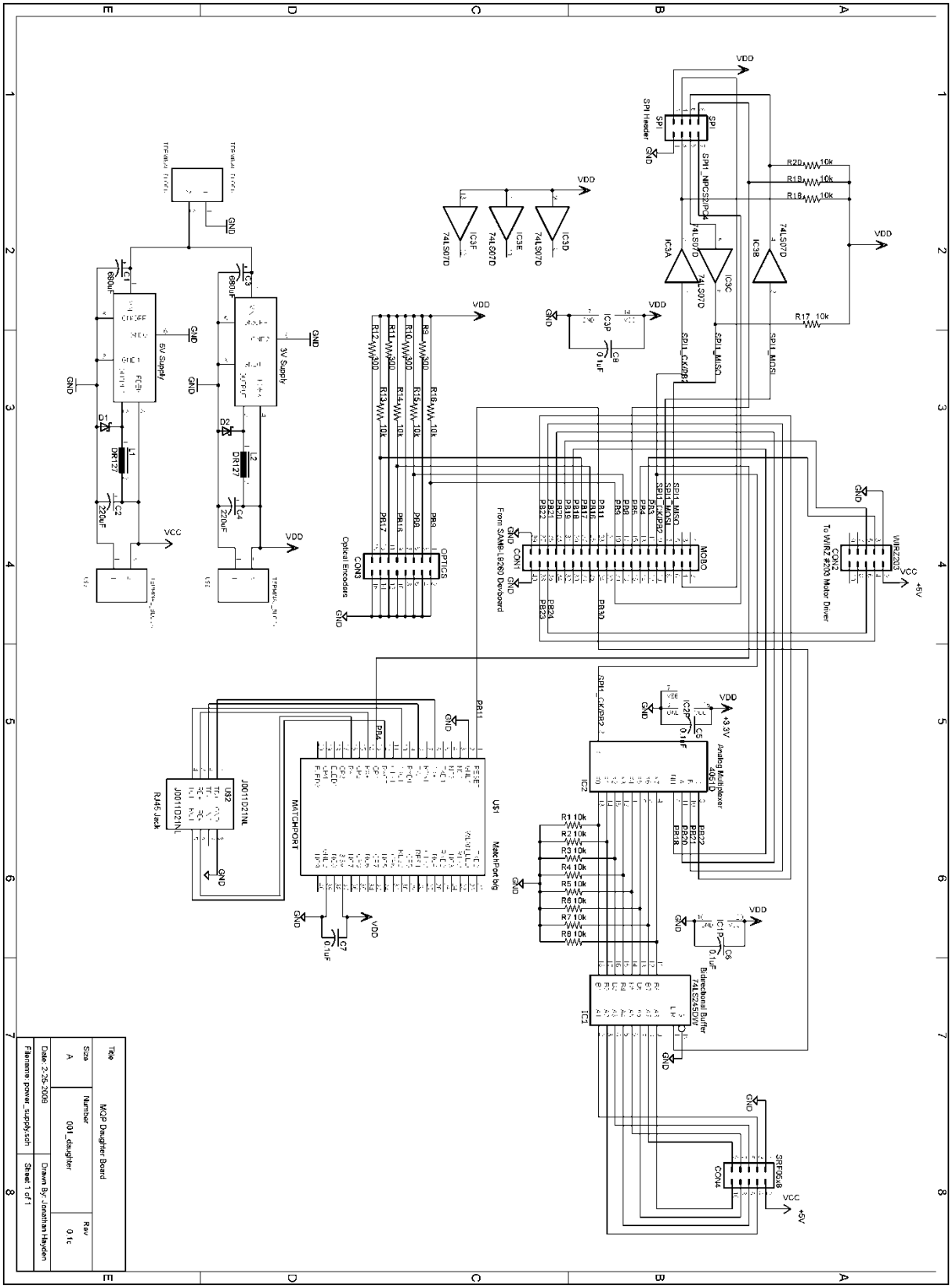


Figure 28 - Daughter Board Schematic

Appendix D – Daughter Board Layout

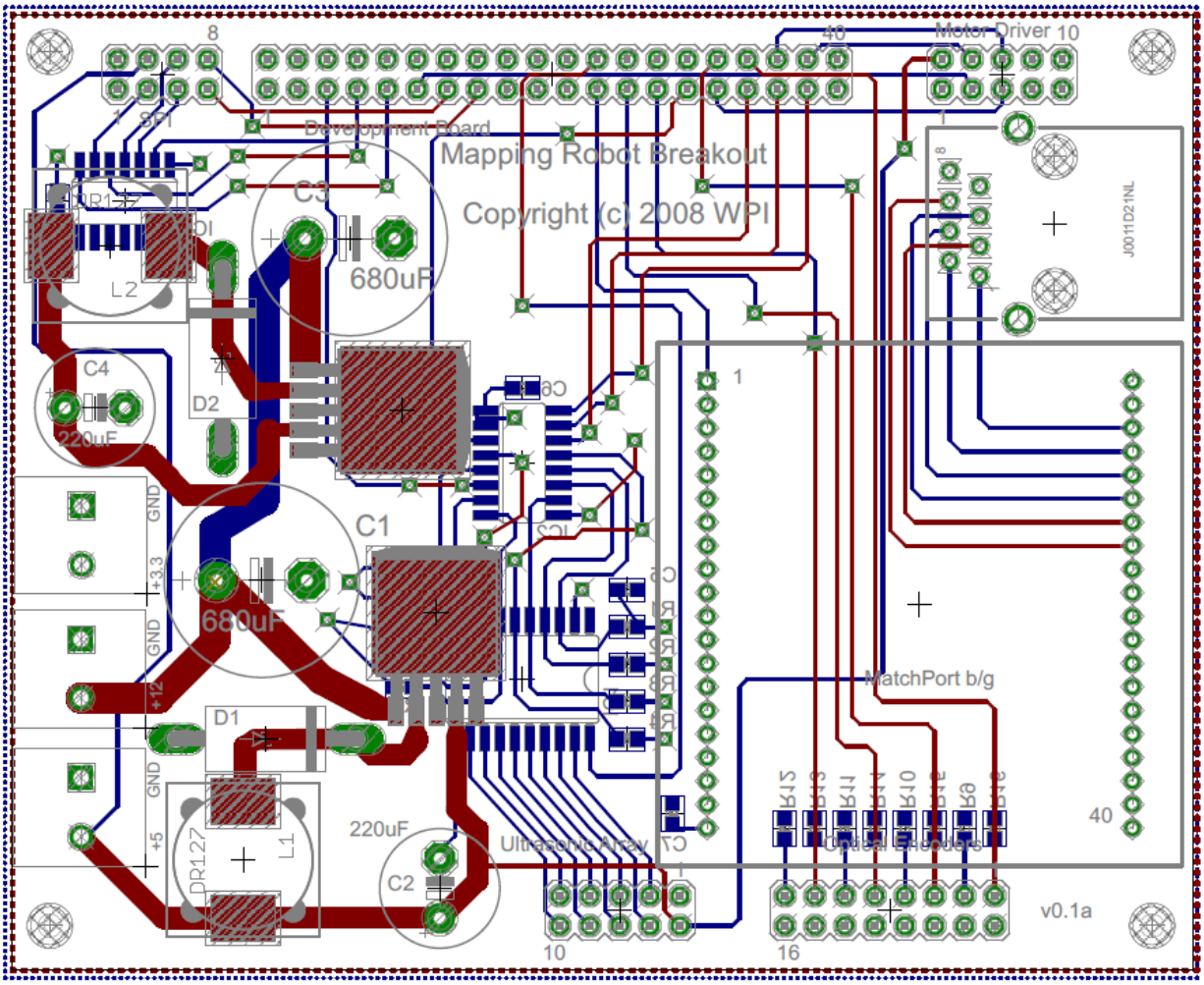


Figure 29 - Layout of the Daughter Board

Appendix E – Ultrasonic Sensor Measurements

Table 7 - Accuracy and precision measurement of the SRF05 Ultrasonic Sensor

Actual Measurement	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10	Average	Range	%Error
140	140.165	140.21	140.18	140.223	140.253	140.245	140.208	140.265	140.192	140.2	140.214	0.1	0.21%
130	130.11	130.128	130.118	130.15	130.149	130.158	130.144	130.165	130.108	129.976	130.121	0.189	0.12%
120	120.3	120.3	120.299	120.3	120.295	120.299	120.3	120.289	120.295	120.299	120.298	0.011	0.30%
110	109.683	109.809	109.847	109.73	109.853	109.869	109.886	109.878	109.883	109.881	109.832	0.203	0.17%
100	100.105	100.104	100.1	100.101	100.102	100.106	100.105	100.094	100.1	100.093	100.101	0.013	0.10%
90	90.15	90.153	90.148	90.156	90.151	90.142	90.141	90.146	90.145	90.148	90.148	0.015	0.15%
80	80.198	80.201	80.204	80.201	80.201	80.2	80.201	80.201	80.2	80.201	80.201	0.006	0.20%
70	70.2	70.2	70.131	70.112	70.165	70.196	70.2	70.199	70.199	70.198	70.18	0.088	0.18%
60	60.205	60.208	60.204	60.209	60.2	60.203	60.206	60.212	60.207	60.206	60.206	0.012	0.21%
50	50.766	50.793	50.794	50.787	50.787	50.781	50.783	50.784	50.785	50.791	50.785	0.028	0.79%
40	40.898	40.9	40.897	40.897	40.898	40.897	40.903	40.899	40.897	40.897	40.898	0.006	0.90%
30	31.296	31.297	31.295	31.299	31.299	31.295	31.297	31.296	31.297	31.298	31.297	0.004	1.30%
20	21.101	21.1	21.1	21.1	21.1	21.101	21.101	21.1	21.101	21.1	21.1	0.001	1.10%
10	10.8	10.8	10.8	10.8	10.8	10.8	10.8	10.8	10.8	10.8	10.8	0	0.80%
5	5.6	5.604	5.615	5.655	5.659	5.847	5.918	5.675	5.695	5.648	5.692	0.318	0.69%
3	3.5	3.497	3.498	3.5	3.498	3.499	3.499	3.499	3.499	3.5	3.499	0.003	0.50%
2	3.102	3.102	3.101	3.102	3.102	3.103	3.1	3.1	3.101	3.102	3.102	0.003	1.10%
1	2.6	2.6	2.6	2.6	2.6	2.601	2.6	2.601	2.6	2.6	2.6	0.001	1.60%
Total												0.056	0.58%

*All measurements in cm.

Table 8 - Angular Range Measurement of the SRF05 Ultrasonic Sensor

	Left Inner	Outer	Right Inner	Outer
20cm	15	18	16	18
40cm	10	15	10	15
60cm	15	16	18	19

*All measurements in degrees.

References

1. **HOKUYO AUTOMATIC CO., LTD.** URG | Photo sensor | Products. *HOKUYO AUTOMATIC CO.,LTD.* [Online] Hokuyo Automatic Co. LTD., 2007. [Cited: April 3, 2008.] http://www.hokuyo-aut.jp/02sensor/07scanner/urg.html.

2. **RobotShop Distribution inc.** Hokuyo URG-04LX Scanning Lazer Rangefinder. *RobotShop.ca.* [Online] 2003-2008. [Cited: April 3, 2008.] http://www.robotshop.ca/home/suppliers/hokuyo-en/hokuyo-urg-04lx-laser-rangefinder.html.

3. **SICK Sensor Intelligence.** Laser Measurement Sensors - LMS-200. *SICK Sensor Intelligence.* [Online] 2008. http://www.mysick.com/eCat.aspx?go=DataSheet&Cat=Gus&At=Fa&Cult=English&Category=Produktfinder&ProductID=9168.

4. —. Winning DARPA Team uses SICK Laser Measurement Sensors on Vehicle. *SICK Sensor Intelligence.* [Online] 2008. http://www.sick.com/gus/about/news/archive/listing/lmssensors/en.html.

5. **Basso, Christophe.** How to deal with Leakage Elements in FLYBACK Converters. *Motorola Semiconductor Application Note.* [Online] http://www.eetasia.com/ARTICLES/2000DEC/2000DEC08_AMD_AN4.PDF?SOURCES=DOWNLOAD.

6. **Kolanko, Frank.** Tech Tutorial: Linear Regulators vs. switchers for automotive applications. *DesignLine.* [Online] 6 12, 2006. http://www.automotivedesignline.com/189400333;jsessionid=R2OCXTO3UGBKGQSNDLQCKH0CJUNN2JVN?printableArticle=true.

7. **National Semiconductor.** *Switching Regulators.* http://www.national.com/appinfo/power/files/f5.pdf : s.n.

8. **Simpson, Chester.** *Linear and Switching Voltage Regulator Fundamentals.* http://www.national.com/appinfo/power/files/f4.pdf : National Semiconductor.

9. **Bradley, David M.** *Odometry: Calibration and Error Modeling.* Robotics Institute Carnegie Mellon University : s.n.

10. **Wikipedia.** Wireless network. *Wikipedia.* [Online] March 25, 2008. [Cited: March 29, 2008.] http://en.wikipedia.org/wiki/Wireless_networks.

11. —. Wi-Fi. *Wikipedia.* [Online] April 3, 2008. [Cited: April 3, 2008.] http://en.wikipedia.org/wiki/Wi-Fi.

12. —. Personal Area Network. *Wikipedia.* [Online] March 25, 2008. [Cited: April 1, 2008.] http://en.wikipedia.org/wiki/Personal_Area_Network.

13. —. ZigBee. *Wikipedia.* [Online] March 18, 2008. [Cited: April 1, 2008.] http://en.wikipedia.org/wiki/Zigbee.

14. —. Bluetooth. *Wikipedia.* [Online] April 4, 2008. [Cited: April 4, 2008.] http://en.wikipedia.org/wiki/Bluetooth.

15. **Technology Questions and Answers.** *Technology Questions and Answers.* [Online] http://expertanswercenter.techtarget.com.

16. **The Seattle Robotics Society.** RF Radio Modules. *The Seattle Robotics Society.* [Online] October 11, 2004. [Cited: March 28, 2008.] http://www.seattlerobotics.org/encoder/200304/donc/RF%20Radio%20Modules.htm.

17. **Active Robots.** Hokuyo Laser & Infrared Range Finders. *Active Robots.* [Online] April 4, 2008. [Cited: April 4, 2008.] http://www.active-robots.com/products/sensors/hokuyo.shtml.

18. **Carnegie Mellon University.** 3D Mapping robot project. *3D Mapping robot project.* [Online] http://www.cs.cmu.edu/~thrun/3D/.

19. —. CMU's robotic mine mapping project. *3D Mapping robot project.* [Online] http://www.cs.cmu.edu/~thrun/3D/mines/index.html.

20. **S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, W. Whittaker.** A System for Volumetric Robotic Mapping of Abandoned Mines. *3D Mapping robot project.* [Online] http://www.cs.cmu.edu/~thrun/papers/thrun.mine-mapping.pdf.

21. **Y. Liu, R. Emery, D. Chakrabarti, W. Bugard, S. Thrun.** Using EM to Learn 3D Models of Indoor Enviornments with Mobile Robots. *3D Mapping robot project.* [Online] July 2001. http://www.cs.cmu.edu/~thrun/papers/thrun.3D-EM.pdf.

22. **CARMEN-Team.** Program with Carmen. *Carmen Robot Navigation Toolkit.* [Online] November 27, 2007. http://carmen.sourceforge.net/program_carmen.html.

23. **MobileRobots.** Robotic Floor Plan Mapping System. *MobileRobots.* [Online] January 11, 2008. http://www.mobilerobots.com/DXmapping.html#Measure_Time.

24. —. Advanced Robotics Interface for Applications. *MobileRobots.* [Online] September 2006. [Cited: April 7, 2008.] http://www.activrobots.com/SOFTWARE/aria.html.

25. **International, SRI.** Centibots: The 100 Robots Project. *Centibots Project Home Page.* [Online] 2002-2004. [Cited: March 28, 2008.] http://www.ai.sri.com/centibots/index.html.

26. **K. Konolige, C. Ortiz, R. Vincent, A. Agno, M. Eriksen, B. Limketkai, M. Lewis, L. Briesemeister, E. Ruspini, D. Fox, J. Ko, B. Stewart, L. Guibas.** *Centibots Large Scale Robot Teams.* s.l. : DAPRA Software For Distributed Robotics: Tech Report, 2002.

27. Integrated networking solutions for OEMs . [Online] Quatech, 2008. http://www.quatech.com/products/embedded.php.

28. Embedded Wireless Networking. [Online] Lantronix, 2007. http://www.lantronix.com/device-networking/embedded-device-servers/wiport.html#features.

29. **Electronics, SparkFun.** *SparkFun Electronics Web site.* [Online] 2005. [Cited: April 18, 2008.] http://www.sparkfun.com/commerce/advanced_search_result.php?keywords=inertia+sensor&x=0&y=0 .

30. **MEMSENSE.** Memsense-nIMU-. *IMUs and Triaxial Intertial Sensors.* [Online] 2006. [Cited: April 14, 2008.] http://www.memsense.com/products/product/moredetails/display.php?product_id=1.

31. **MicroStrain.** Inertial Measurement Unit and Vertical Gyro. *MicroStrain: Orientation Sensors.* [Online] 2008. [Cited: April 18, 2008.] http://www.microstrain.com/inertia-link.aspx?gclid=CMOk_oDD3ZICFQQIFQodkVt8_A.

32. **Analog Devices.** Analog Devices ADIS16006. *Analog Devices, Inc.* [Online] 1995-2008. http://www.analog.com/en/prod/0,,764_800_ADIS16006%2C00.html.

33. —. Analog Devices ADIS16255. *Analog Devices, Inc.* [Online] 1995-2008. http://www.analog.com/en/prod/0,,764_801_ADIS16255%2C00.html.

34. **Zagros Robotics.** SRF05 - Ultra-Sonic Ranger Technical Specification. *Zagros Robotics.* [Online] [Cited: April 20, 2008.] https://www.zagrosrobotics.com/Index.asp.

35. **The Semantic Robot Vision Challenge.** The Semantic Robot Vision Challenge. *The Semantic Robot Vision Challenge.* [Online] http://www.semantic-robot-vision-challenge.org/.

36. **RoboGames.** RoboGames. *RoboGames.* [Online] http://robogames.net/.

37. **Rensselaer Polytechnic Institute.** Topological Mapping with Sensing-Limited Robots. *Computer Science ar RPI.* [Online] http://www.cs.rpi.edu/~beevek/topological.html.