Automatic Vehicle Recharging Station (AVRS)

A Major Qualifying Project
Submitted to the Faculty of the
WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the
Degree of Bachelor of Science
By:

Matthew Fortmeyer **Nikolas Gamarra** **Robert O'Brien** **Jacob Remz**

Project Advisors:
Craig Putnam and Jie Fu

Date: 4/25/2019

Abstract

The goal of this project is to recharge autonomous electric vehicles, thus automating one of the last steps preventing a level 5 autonomous vehicle from being completely independent from human assistance. For this project we are using an ABB industrial robotic arm with a custom end effector. Our tooling has force sensing and a tool changer, allowing for switching between a pneumatic flap opener and two charging port standards. To help us achieve this goal we implemented custom computer vision and point cloud processing software. System integration was done using ROS-Industrial. Ultimately, while we were able to plug in our test vehicle using the ABB arm, there were issues with latency as well as with the reliability of the real time path planning features of ROS-Industrial.

Executive Summary

With major developments in vehicle autonomy and a rise in the popularity of electric vehicles, the need for an automatic charging solution is on the horizon. As it currently stands, autonomous vehicles still require human intervention for recharging or refueling. To ensure level 5 autonomy will be able to operate continuously without human interaction (outside of as passengers), automatic vehicle recharging stations will need to be developed. In order to solve this problem, we developed an autonomous charging station using an ABB IRB 1600-6/1.45 industrial robotic arm. Our system uses a ROS-Industrial framework to articulate the arm, along with computer vision and point cloud data to locate the charging port. The system is designed to work with two of the most popular charging standards used in the U.S. electric vehicles today, Tesla and J-1772. The system was able to reliably open the charging port flap from a variety of angles and was able to plug in the charger, but with a smaller range of tolerance.

Table of Contents

Introduction

Electric Vehicles

The automotive industry has been dominated by combustion engines for a century, but recently electric vehicles (EVs) have become increasingly common (Sierzchula, 2014). EVs are not a recent invention, but only became feasible market alternatives around 2010. This is largely due to limitations in speed and driving range, which are often sacrificed in order to make EVs competitively priced compared to combustion engine cars. Unlike refueling internal combustion engine (ICEs) vehicles, charging EVs can be a lengthy process and requires specific charging stations to be worked into the infrastructure.

Charging options for EVs are currently limited, and the low availability of charging stations reduces the effective range of the vehicles. While public charging stations exist, the most reliable method of charging is in-home. Infrastructure is continuously being built to better support EVs, but regardless, charging can be a lengthy process. Different EVs support various maximum charging speeds, but there are two common charging levels publicly available and a third emerging (Saxton, 2011). First is Level 1 charging, which can be sourced from any ordinary outlet, but only charges at around 1.5kW/h. Depending on the vehicle, this provides around the equivalent of 4-5 miles of range per hour. Level 2 charging, which is supported on all modern EVs, charges at 6-8kW/h, provides at least 25 miles of range per hour. These first two stations can be set up in private homes and can be installed in public areas for convenient charging, but still require multiple hours to fully charge a vehicle. Level 3 charging, also called DC fast charging, is not supported by all EVs, but provides around 40 miles of range per 10 minutes charging. These level 3 stations first appear to solve the issue of high charging times but are much more expensive and cannot be regularly installed in homes. Alternative charging methods do exist, but are still far from public availability. In addition, significant research and development has gone into creating more efficient batteries, primarily lithium ion batteries, but the unstable market and unsure future of EVs stunted manufacturing until recent years (Sierzchula, 2012). The real driving force for EVs has been environmental concerns and resulting policy. Numerous countries have implemented policies to reduce vehicle emissions and have established or increased tax credits for owning an electric vehicle. These factors, among others, have promoted the use and development of EVs.

As the demand for transportation continuously increases, vehicle emissions become a greater and greater environmental concern (Sierzchula, 2014). EVs provide a very clear solution, despite the barriers that have limited the spread of the emerging technology. These include reluctance to leave behind ICEs, higher cost and lower performance of a current EVs compared to ICEs, and uncertainty of the technology’s trajectory. However, EVs are reaching the point of undeniable market success and saturation (Becker, 2009). As seen in Figure 1 below, sales in 2018 are projected to reach more than twice the sales in 2017, and this rate of growth is expected to continue (Statista, 2018). However, in order to support this growth, charging options need to become more widespread. While development of autonomous driving began initially with ICEs, as they were easily obtainable and reliable, it is safe to say EVs will be the dominant vehicle type in the future.

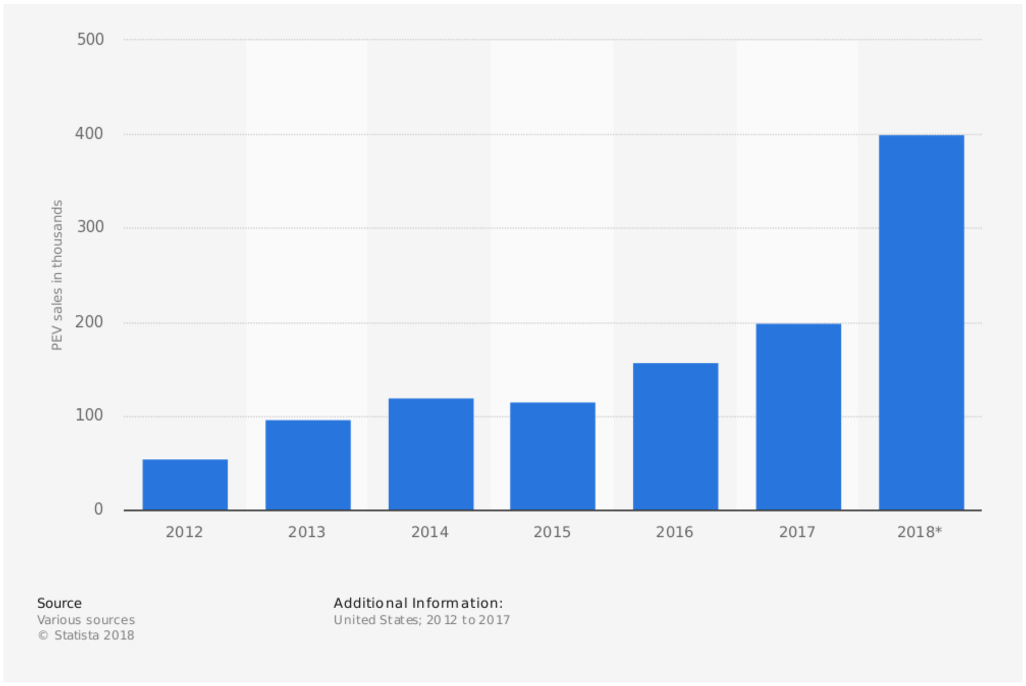


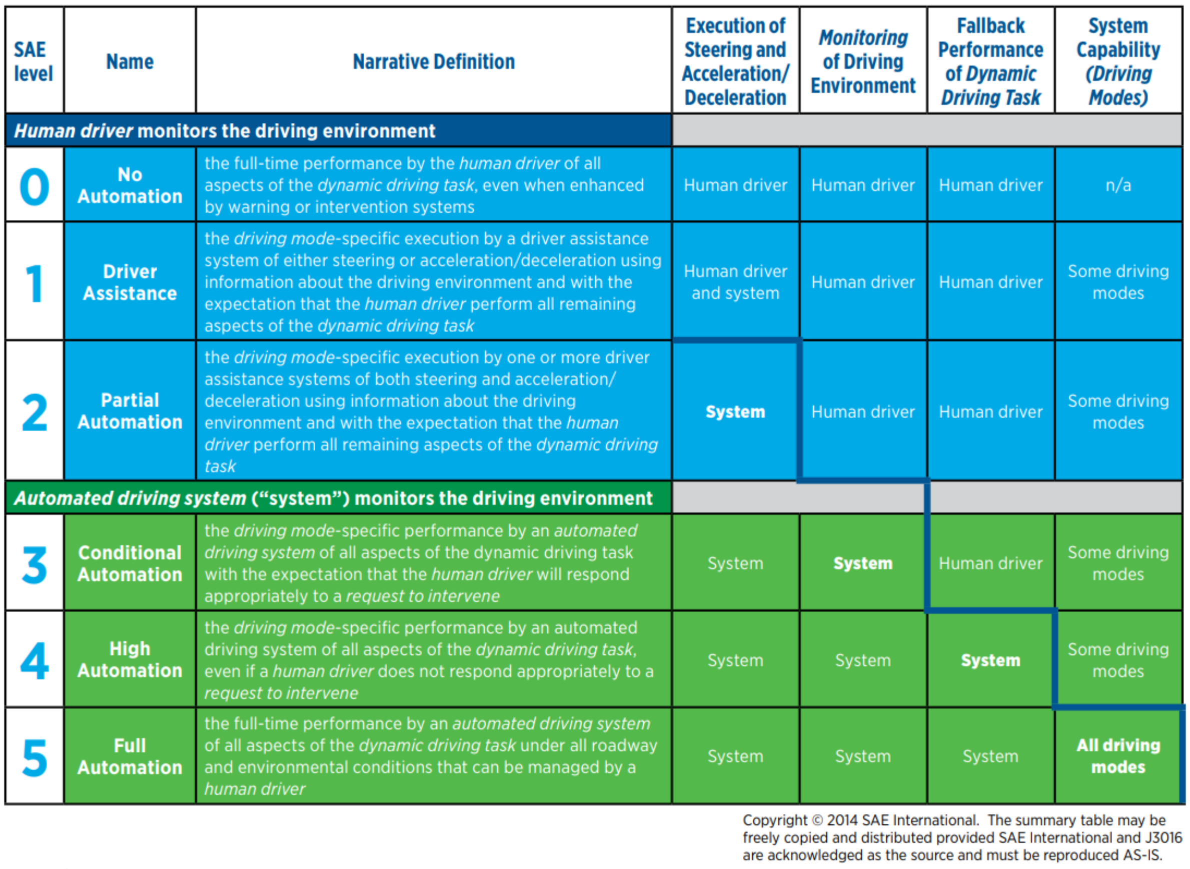
Figure 1: Electric Vehicle Sales in the US from 2012-2018

Autonomous Vehicles (AVs)

While many experts disagree on how soon we may be seeing mass adoption of autonomous vehicles, nearly all agree that vehicle autonomy will greatly affect the transportation industry in the 21st century. In their article "From Horseless Carriage to Driverless Car" (Irving Wladawsky-Berger, 2018) the Wall Street Journal remarks on how this technology could be as transformative as the adoption of the car in the early 20th century. Just as the leap from horses to cars redefined the transportation industry and brought with it numerous societal benefits and costs, the adoption of autonomous vehicles has the potential to do the same.

One of the most obvious and significant potential benefits from full vehicle autonomy would be increased safety. In 2017 alone U.S. vehicle fatalities were estimated to total over 40,000, and a full 94% (Bomey, 2018) of these can be attributed to human error. Thus, eliminating the potential for human error would be the single most effective way to reduce road fatalities. In the paper “Preparing a Nation for Autonomous Vehicles: Opportunities, Barriers and Policy Recommendations”, Daniel J. Fagnant (2015) states, the number of lives saved per year could reach nearly 22,000 at 90% AV adoption. Another less obvious societal benefit of AVs is economic savings that stem from reduced congestion, reduced need for parking, reduced rate of crashes, fuel savings, and insurance savings. In total these savings were estimated to be approximately $3100 per AV per year at 90% adoption. This would result in approximately $355.4 billion in savings across the U.S. economy and at only 10% adoption could still result in annual economic benefits in the range of $27 billion nationwide. While many barriers to full adoption of AVs remain, automakers have made it clear that they think autonomous vehicles will be a part of the future through their investments and research into the field. These investments are coming from new tech startups (Optimus Ride, Zoox), tech giants (Google/Waymo, Apple), and traditional auto manufacturers (Ford, GM, Toyota, and many others). Ford has recently begun to invest in its own autonomous vehicle research subsidiary in order to stay competitive with companies such as Tesla, Uber, and Waymo (Colias, 2015). Toyota alone has made a $500 million partnership with Uber (Bensinger, 2018). In 2016, GM spent $581 million to acquire self-driving car startup, Cruise Automation (Walker, 2018).

In order to help provide a standardized frame of reference for what the definition of an autonomous vehicle is, SAE (Society of Automotive Engineers) defined the five levels of vehicle autonomy in their International Standard J3016.



SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* (“system”) monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 2. Levels of Vehicle Autonomy

This standard provides a logical framework for what constitutes various levels of vehicle autonomy and can be seen in *Figure 2 (Acosta, 2018)*. As of 2018 several companies such as Tesla and BMW offer Level 2 autonomy in their vehicles as a form of advanced driver assistive technologies. Many companies such as Uber, Lyft and Waymo are also testing level 3 autonomy prototypes on public roads. These vehicles are essentially fully autonomous but have a driver present as a fallback in the case of an emergency. One of the most interesting potential uses of autonomous vehicles is the use of Level 5 autonomy fleets of vehicles operated by rideshare companies. Such companies could provide end-to-end transportation to their users without the need for a human intervention or operation. An autonomous fleet system like this would greatly benefit from some means of recharging without help from humans.

Fleets

Autonomous vehicles are still a new technology, but a few companies are already exploring how they can be deployed in large numbers to satisfy the needs of consumers. Some are already developing fleets of autonomous vehicles that can function fully independently. For example, Uber is currently creating a fleet of autonomous vehicles that will be used as a part of their ridesharing service. In this scenario, riders would not require any human interaction in order to use the service. Pickup, navigation, drop-off, and payment would all be handled in one automated process, which would increase efficiency and could cut down on labor costs. As of 2017, the company planned on acquiring a fleet of up to 24,000 autonomous vehicles (Camhi).

Using fleets of autonomous vehicles to power ridesharing services also allows for more flexible implementations. Autonomous fleets greatly expand the ability of services to offer dynamic ride-sharing options, where the service will group riders with others who are travelling a similar path. While it may result in a slight increase in ride duration, it makes the potential cost to riders much lower. In a study done by researchers at the University of Texas, they found that consumers did not mind a small increase in travel time if it meant that they didn't have to pay as much (Gurumurthy, Kockelman, 181). After analyzing ride-sharing data in the city of Orlando, Florida they found that nearly 60% of single person trips could be shared with less than five minutes of additional travel time. This is due to the fact that most riders only used the service for very short trips, usually around five miles. A distribution of trip lengths can be found in figure 3 (Gurumurthy, Kockelman, 184). The study also estimated that one autonomous vehicle which employed a dynamic ride-sharing system could replace six standard vehicles currently used by ride-sharing services. If implemented correctly, fleets of autonomous vehicles could not only cut down on labor costs, but also reduce the number of vehicles needed by ride-sharing companies.

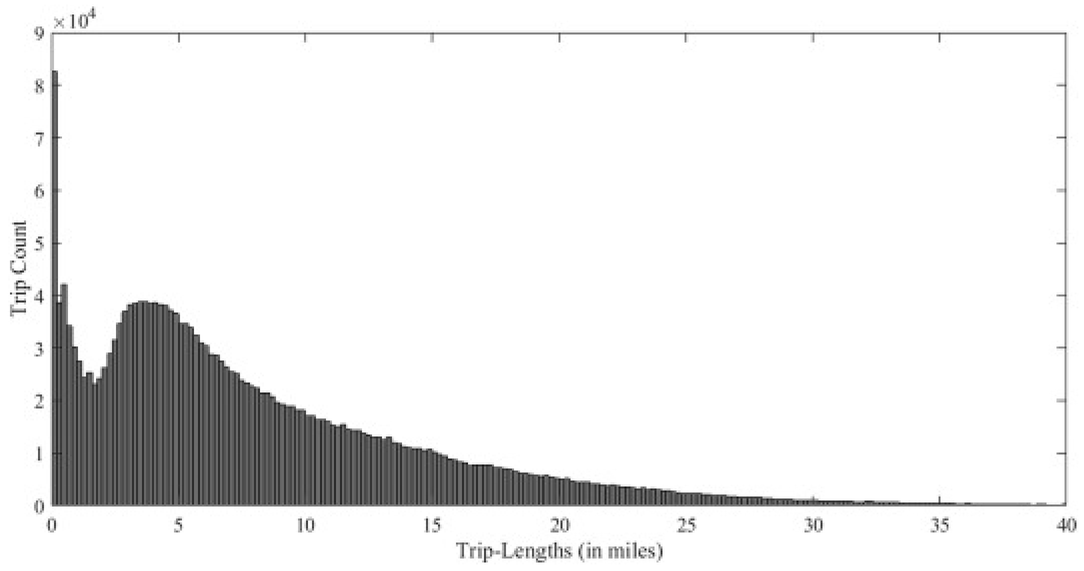


Figure 3. Average Trip-Length

While autonomous fleets would be cost effective for ride-sharing companies, it is also important to note the areas in which they would increase costs. The previously mentioned study found that an autonomous vehicle travels 161 miles per day on average and can travel up to 301 miles in one day in extreme cases (Gurumurthy, Kockelman, 182). This is significantly more distance than the average non-autonomous vehicle and is far more taxing on the vehicle. It is likely that vehicles used in this type of system would require frequent maintenance and would probably not have the longevity of a non-autonomous vehicle.

Background

It is likely that autonomous vehicles will take an increasing role in society in the near future. As they do, more technologies will be developed alongside, such as autonomous charging. Currently, the ideal fleet system requires no regular human interaction until charging is considered. While automatically plugging in and charging EVs has been achieved before, it is far from becoming a dependable or viable market solution. Additionally, the future of such a pervasive autonomous system must be considered before it can be implemented. Removing the human element from industry entirely is not universally accepted, although many argue for the overall societal benefits of automation.

Solutions of Autonomous Charging

Autonomous vehicle refueling for Internal Combustion Engines is a technology that has been around since the late 90s but has been through various iterations since then (Lamm, 1998). Designs have ranged from a continuous joint arm with a relatively small work envelope, to overhead gantry-based systems that could access cars in a multitude of positions and locations (Ziegler, 2016). Existing designs for automatic electric charging have been various arm designs with plugs built into them, battery swapping, and inductive charging. The first of these design types, robotic arms, has so far been limited to designs compatible with only one type of charging standard available, while more than 5 competing standards exist worldwide (Edelstein, 2014). Another method, battery swapping, is plagued by two major issues, cost and lack of standardization (Voelcker, 2014). The battery in an EV is usually the most expensive component, so battery ownership becomes a major concern, especially if batteries are constantly being swapped out and there is no guarantee that any car will receive the same battery more than once. Car manufacturers all also have proprietary car designs for integrating batteries into their vehicles making a quick universal battery swapping system quite difficult to achieve and implement. By contrast, wireless charging doesn't suffer from the problem of standardization nearly to the same extent, but instead is held back by energy efficiency (Genovese, Ortenzi, and Villante, 2015; Onar and Jones, 2015).

Of all the designs that have been created and tested, few have reached the market. The first of these designs to reach market serviced gasoline vehicles and was installed beside a standard fuel pump. This design utilized an industrial arm enclosed in a metal box that would retract to expose the arm after the car pulled up to the pump; the arm would then reach out to open the gas flap, unscrew the gas cap, remove the gas pump from the fueling stand, fuel the vehicle, then put everything back to the way it was originally (The Automated Refueling Process, 10/4/18). Another design in development is that of PowerHydrant, a SCARA arm-based system that utilizes a proprietary plug type and boasts the ability to service multiple vehicles parked in multiple locations (Leary, 2016).

Social Implications of Autonomy

Ethical implications are an important consideration in any engineering project. Without a code of ethics, an engineer is just one who applies science to solve a problem regardless of whether that solution benefits society as a whole or just a few individuals. In order for a society to achieve the best possible outcome from engineers, it is in our collective best interest for us to hold our engineers to a set code of ethical standards in the same way the American Medical Association has established a code of ethics for medical professionals. For this reason, the National Society of Professional Engineers (as well as several other professional engineering societies) have established Fundamental Canons of ethics for engineering. The first of these being (NSPE, 2007): *Hold paramount the safety, health, and welfare of the public*.

When considering the safety and health of the public, vehicle automation is an obvious step toward a better future. In 2017 alone, U.S. vehicle fatalities were estimated to total over 40,000, and a full 94% (Bomey, 2018) of these can be attributed to human error. Thusly, eliminating the potential for human error would be the single most effective way to reduce road fatalities. However, not all ethical considerations can be so simple. The potential implications of this canon are very large as the “the welfare of the public” has many potential interpretations. For instance, is automation that results in elimination of jobs in the best interest of the public welfare? This section will cover this and other social implications of ever increasing autonomy.

First it must be established that automation itself is not the enemy. Since its beginning, mankind has been striving to produce more for less effort. From the agricultural revolution, to the industrial revolution, to the robotic revolution; humans have been automating their lives to make them easier. All previous technological revolutions created more jobs than they eliminated. This shouldn’t be taken for granted when it comes to the robotic revolution. For the first time in human history, automation has begun to outpace job growth by a very significant margin. According to David Rotman (2013), “It’s one of the dirty secrets of economics: technology progress does grow the economy and create wealth, but there is no economic law that says everyone will benefit.” This can be seen most clearly when looking at graphs of worker productivity overlaid with employment.

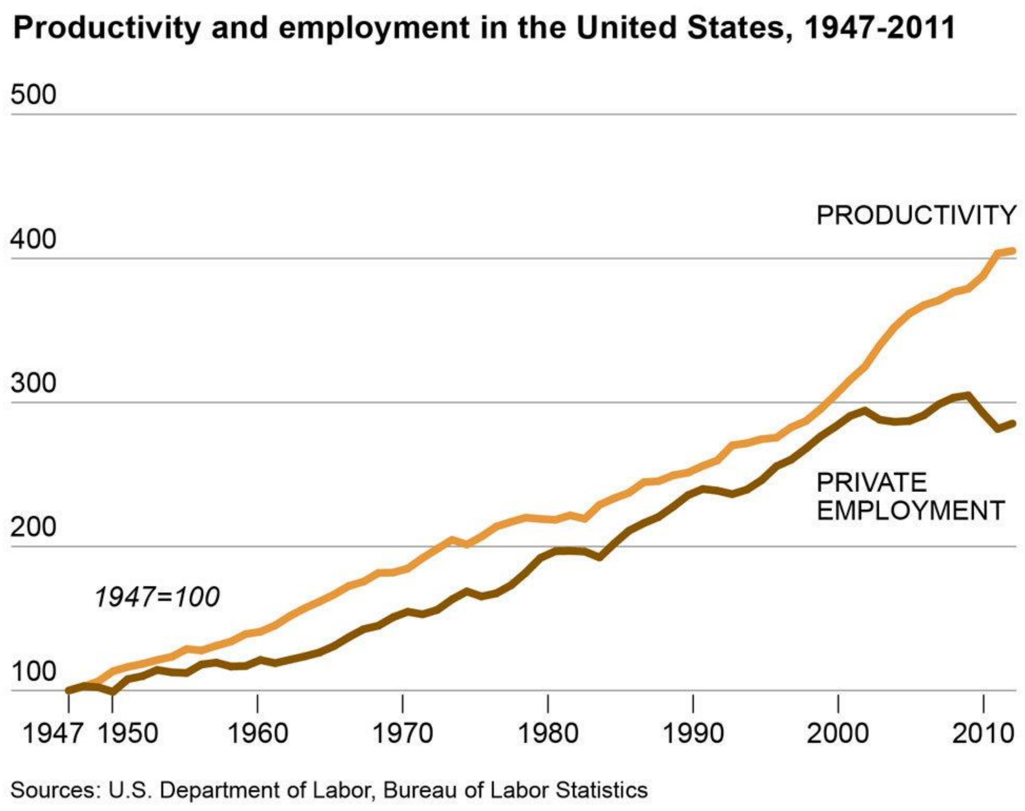


Figure 4. The Great Decoupling

For most of U.S. history these two lines have continued alongside each other in an almost parallel fashion. Then in 2000, employment began to stagnate yet productivity continued to increase, as seen in *Figure 4 (Brynjolfsson, NYT)*. Although economists are split on whether this should be attributed to automation or not, the ever increasing gap between these graphs is a reality we have to face as the technology behind automation continues to advance and exacerbates this problem.

Automation isn't going away, and we shouldn't want it to. As stated before, automation enables us to produce more for less effort. However, we as a society need to begin thinking now about how the rewards of that productivity are going to be distributed when large portions of the population are unemployable. Or as stated by MIT professor Erik Brynjolfsson (2015):

> It’s time to start discussing what kind of society we should construct around a labor-light economy. How should the abundance of such an economy be shared? How can the tendency of modern capitalism to produce high levels of inequality be muted while preserving its ability to allocate resources efficiently and reward initiative and effort? What do fulfilling lives and healthy communities look like when they no longer center on industrial-era conceptions of work? How should education, the social safety net, taxation, and other important elements of civic society be rethought?

We as a society don’t have the answers to these questions right now but as automation technology continues to advance ever more rapidly, it will be increasingly important for engineers, politicians, and other members of society to begin having these conversations.

In regard to the ethical implications of this project, there is not currently much potential for this technology to cause significant job displacement as there is currently very little economic incentive to implement it. Economic viability of the project will be discussed in greater depth in the results section. It should also be noted that there isn’t currently a significant number of people working as vehicle rechargers. In the future as autonomous vehicle technology continues to develop, we must remain cognoscenti that such industries may not create as many jobs in supporting roles at previous technological advances have.

Methodology

Design Methods

Project Objectives and Goals

Autonomous vehicles are slowly moving towards becoming the standard for private transportation. As this standard progresses it will require infrastructural changes to accommodate for the technologies. Namely, the charging of driverless vehicles which currently require human interaction. Our project seeks to provide an automatic charging solution for electric vehicles using an industrial robotic arm. However, this solution, the automatic vehicle recharging station (AVRS) is not being designed with market feasibility in mind. We are designing a safe and repeatable solution for both the academic value and to take a step towards making this technology more achievable.

In order to accomplish this task, we designed an end effector and developed controls for an ABB IRB 1600 1.45/6 arm. This end of arm tooling (EOAT) provided the capability to open the charging port and any present flaps, identify the type of charging port, and then plug in the corresponding charger. We limited our scope to two types of chargers: the Tesla proprietary charger and the J1772. Additionally, a communication protocol between the 'vehicle' and arm were simulated in order to transmit necessary information.

Necessary Behaviors

The first part of our design process was outlining the behaviors the system would have to go through as part of its operation. This outlining of behaviors served as a guide for our design, ensuring that we included any functionality that may not have been immediately obvious and allowed us to identify the manipulators required for each step. Table 1 below details each action taken and specifies a completion condition if needed. The overall objectives can be generalized as the following:

- Identifying a vehicle
- Locating the charging port
- Opening the port
- Determining the pose of the port
- Plugging in the charger
- Removing the charger at a specified time and
- Closing the port

The design of our end effector was formulated from the logic represented in this table. Once the potential manipulators for each task were identified we compared similarities to find the best set of tooling with the most overlap.

Table 1

SUBSYSTEM	**ACTION**	**Comments/Success Condition**
	Connect to vehicle comms	connection message
Identify car (Coms msg)	Identify port type	pull from database based on car type and account info
	Identify flap location	

End effector	Flap opens slightly or all the way (done initially by vehicle)	Flap must be opened all the way
	Identify if port has latch cover or not	Open latch cover
	Verify port type (should already be known based on database (license plate, RFID, QR, ect...)	
	Determine port pose (location and orientation)	
	Plug in port	Tesla, J1772, CHAdeMO
	When to unplug (Appendix B)	After elapsed time?
		Sense the charging amount?
		user/AV based “unplug at this time”
		Unplug at scheduled time?
	Depress release button on plug (if it exists)	
	Remove plug from port	
	If latch cover, close latch cover	
	Close flap	
Return to waiting location		

End of Arm Tooling

In order to fulfill all required actions our EOAT requires a minimum of two end effectors: the mechanism used to open and close the charging port/gas flap, as well as an additional end effector for charging the vehicle. As the scope for this project only included two charging standards, the creation of one end effector for each standard resulted in a total of three end effectors having been created. As part of the requirements for this project, force feedback was determined to be necessary for safe control of the tooling during operation near the vehicle. Without force sensing integrated into the EOAT, the arm could potentially push into the vehicle causing damage to itself or the vehicle or even harm bystanders before the arm itself detected a problem. To obtain usable force feedback data we designed a custom force sensing package, utilizing three 200kg load cells (TAS501) distributed in a 120 degree pattern, each of these load cells connects to a load cell amplifier (HX711) which in turn communicate with the control system via an Arduino Mega running a ROS node. This force sensing package, as shown in Figure 4, is mounted directly to the tool flange of the ABB IRB 1600. The force sensing package was designed in the shape of a hexagon to accommodate the three load cells without increasing the overall size of the structure by a too large a margin. Five of the six faces of the force sensing package were made available for sensor mounting, the sixth and bottom face was designed specifically to facilitate mounting of the Arduino Mega. Two sensors were mounted to the exterior of the force sensing package, the first of which was a 1080p webcam mounted to the face opposite the Arduino Mega and the second was a VL6180 Linear Range Finder mounted to one of the faces adjacent to the Arduino Mega. These sensors were used for computer vision and to detect how close the tooling was on final approach to the vehicle respectively. The female side of the tool change was mounted to the side of the force sensing package opposite to the tool flange of the robot such that the force sensing package acted as a Z offset of the tool flange.

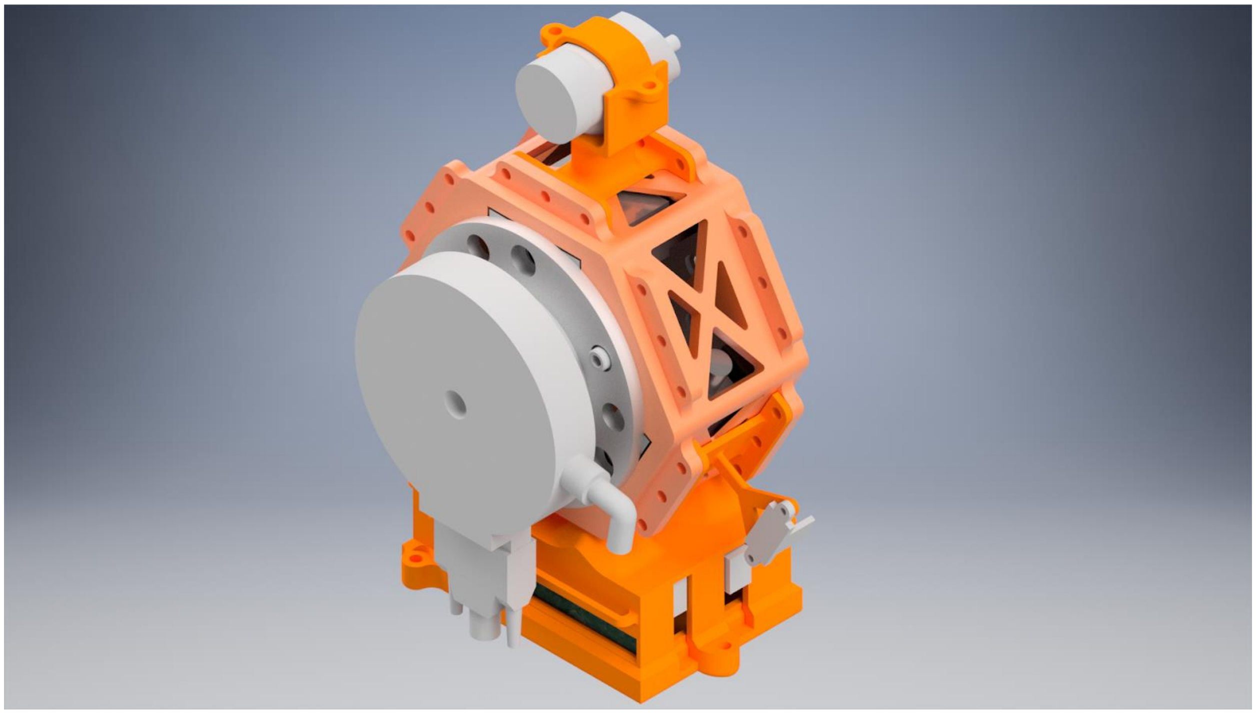


Figure 5: Force Sensing Package

The use of a tool change in this project allows for the active tool of the EOAT to be interchangeable without having to remount different equipment to the arm. The tool change used in this project has four (4) pneumatic pass-through lines and 15 pins of electrical connectivity available to pass information through a given set of connected tool change plates. These two features allowed for the vacuum cup to be utilized easily, without having to worry about pneumatic lines being connected and disconnected, and for the vacuum sensor mounted to the same tool to be mounted without the need for wireless communications to read its values.

Our first interchangeable tool was the pneumatic vacuum cup, shown in Figure 5, used to both open and close the charging port flap. This tooling was composed of a 2.5 bellow Hithane vacuum cup with a diameter of 42mm (EMI 1661) mounted to a 35mm stroke suspension (EMI

6828) supplied with vacuum by a venturi (EMI 362). The decision to use a vacuum cup with greater than half a bellow was to enable suction on contoured surfaces with a reduced need for accurate orientation on approach. Hithane was chosen due to its tendency to not leave any marks on the surfaces it interacts with. A 35mm stroke suspension provided the tooling with greater compliance ensuring the robot would be able to passively prevent a crash with the exterior of the vehicle. The vacuum necessary to use the vacuum cup was generated by a venturi, which generates suction using the Venturi effect, supplied with 90 psi via the pneumatic pass through of the tool change. Once the vacuum is generated and the vacuum cup has made contact with the vehicle and forms a seal, the vacuum sensor (EMI 1931) mounted at the base of the suspension sends a signal to the Arduino Mega communicating the state (sealed or not sealed) of the vacuum.

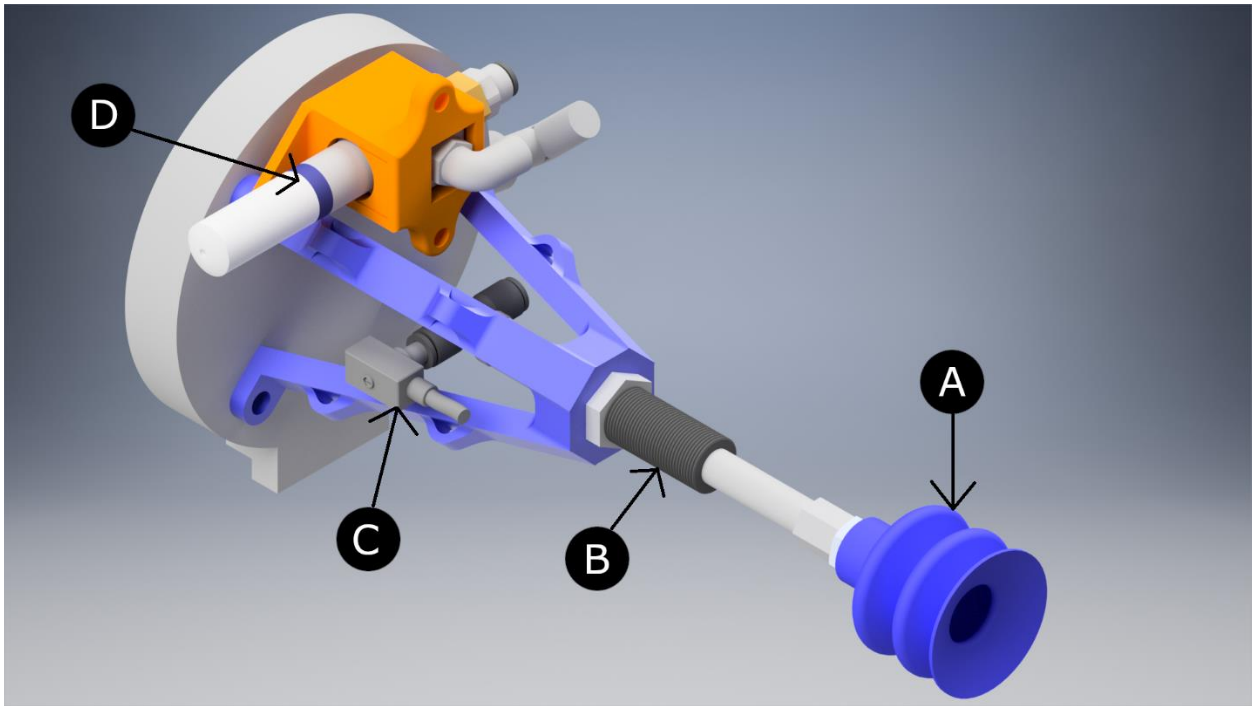


Figure 6: Pneumatic Vacuum Cup Tool: A) 2.5 Bellow Hithane Vacuum Cup, B) 35mm Stroke Suspension, C) Vacuum Sensor, D) Venturi; note pneumatic tubing not shown

The second and third interchangeable tools designed for this project are the Tesla and J1772 male charging plug emulators, pictured in Figures 6 and 7 respectively. Both of these tools are mostly 3D-printed and are designed such that when plugged into the test vehicle there is clearance between the port flap and the force sensing package. As shown in Figure 6 the Tesla tool utilizes a J1772 to Tesla adapter so that when plugged into the test vehicle one half of the connection is a genuine connector, increasing the validity of our testing. The J1772 tool still meets this testing standard, regardless of its connector being entirely 3D-printed, due to the J1772 to Tesla adapter being mounted within the test vehicle to provide a J1772 female port.

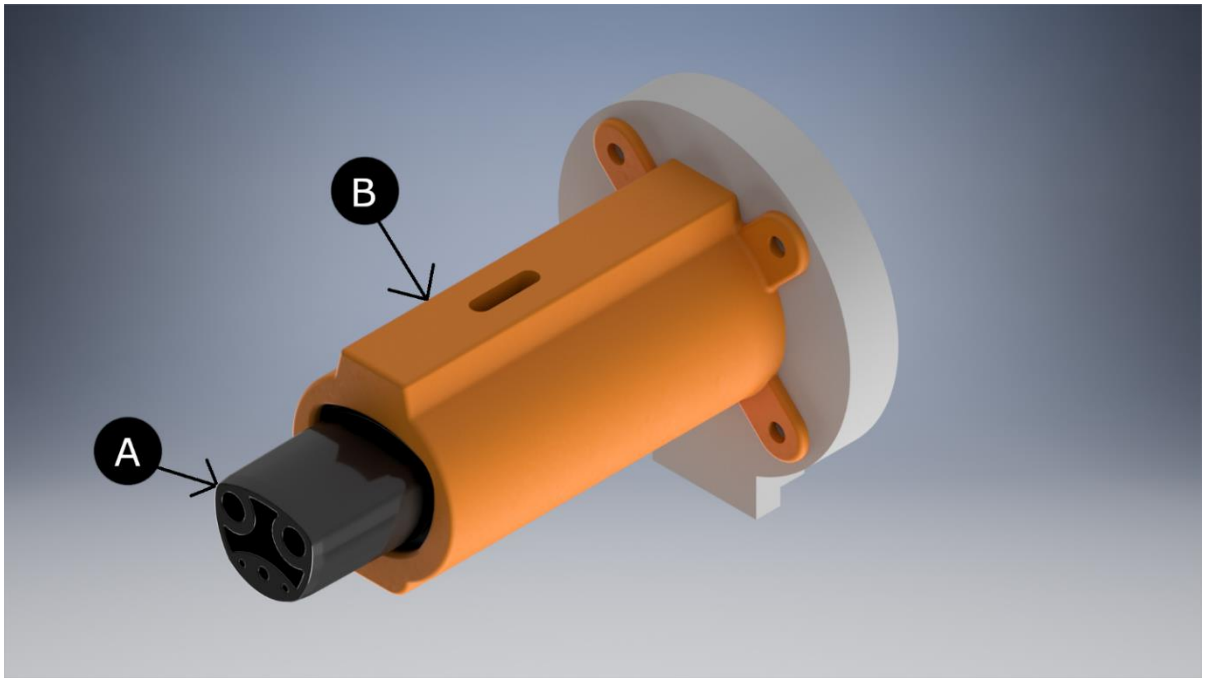


Figure 7: Tesla Male Tool: A) J1772 to Tesla Adapter, B) 3D-printed mounting structure

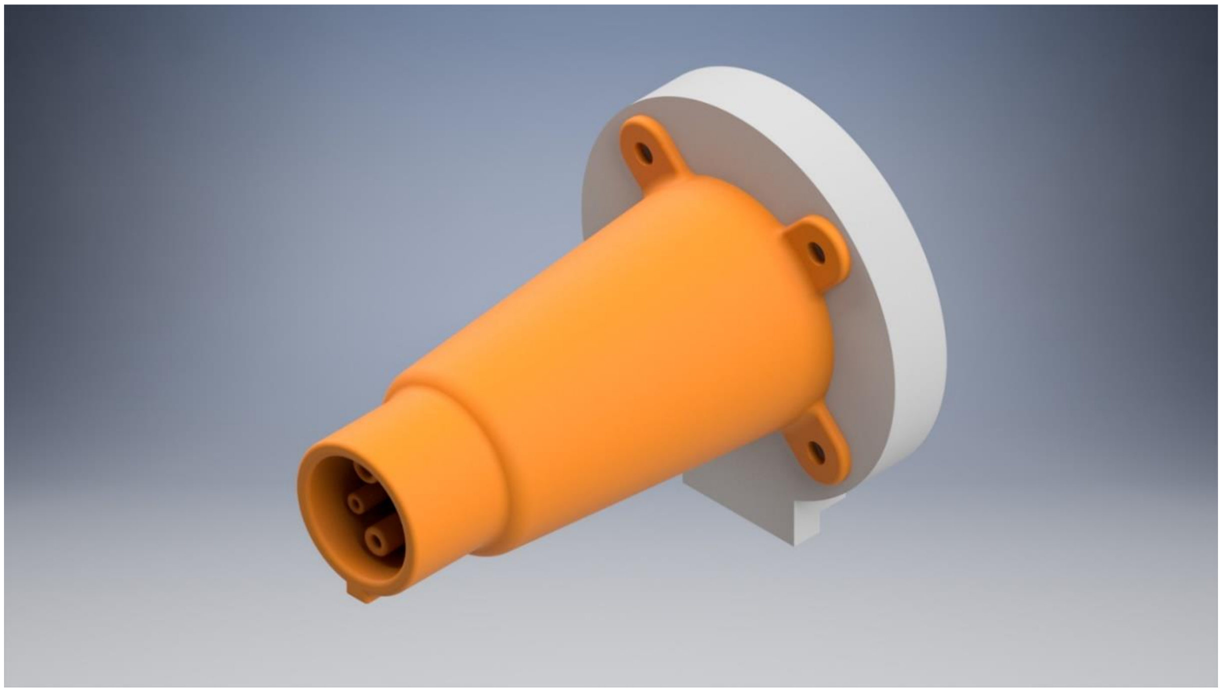


Figure 8: J1772 Male Tool

Kinematics

In order to gain a more complete understanding of the control of the robotic arm we are using we derived its DH parameters. As with many 6-DOF robotic arms we found that there were several ways to represent the DH parameters depending on how much the robot is simplified. To verify that our parameters were correct we plugged them into RoboAnalyzer, a software capable of creating 3D wire frame models based on DH parameters as seen in Figure 8. Our DH parameters (as seen in Table 2) are in a complete format so that it can be used to recreate the wire frame model. Measurements were pulled from the ABB IRB 1600 Product specification as seen in Figure 9. Some of the parameters can be changed to simplify calculations further. The values marked in red can be set to zero as they cancel each other out. The values marked in orange can also be set to zero, but the offsets will need to be added back into the final resulting transformation matrix.

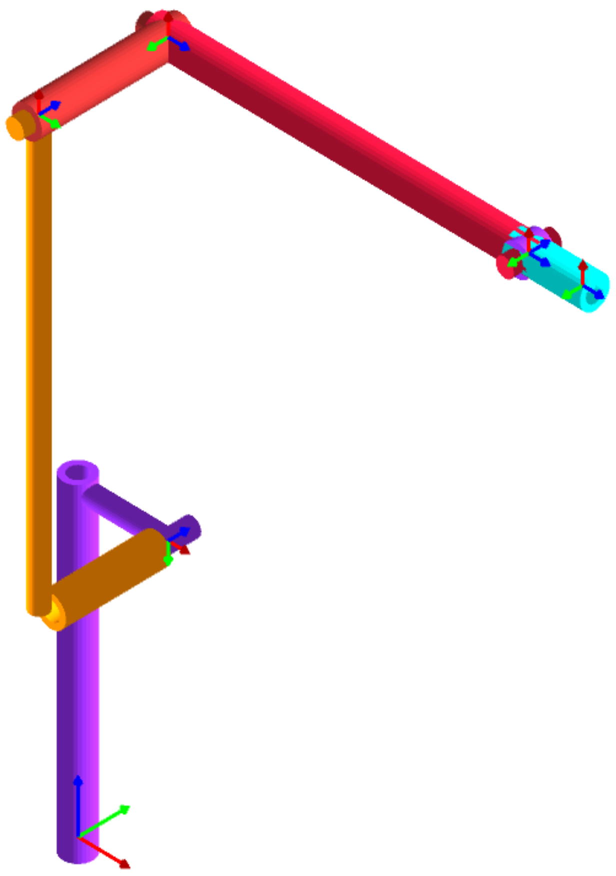


Figure 9: ABB IRB 1600 Wire Frame Model

Table 2: D-H Parameters

Link i	d_i (mm) (z Disp. / Link Length)	a_i (mm) (x Disp. / Link Offset)	α_i (deg) (x Rot. / Link Twist)	θ_i (deg) (z Rot. / Joint Angle)
1	**486.5**	**150**	**-90**	**θ1***
2	**-215**	**700**	**0**	**θ2*-90**
3	**215**	**0**	**-90**	**θ3***
4	**600**	**0**	**90**	**θ4***
5	**0**	**0**	**-90**	**θ5***
6	**65**	**0**	**0**	**θ6***

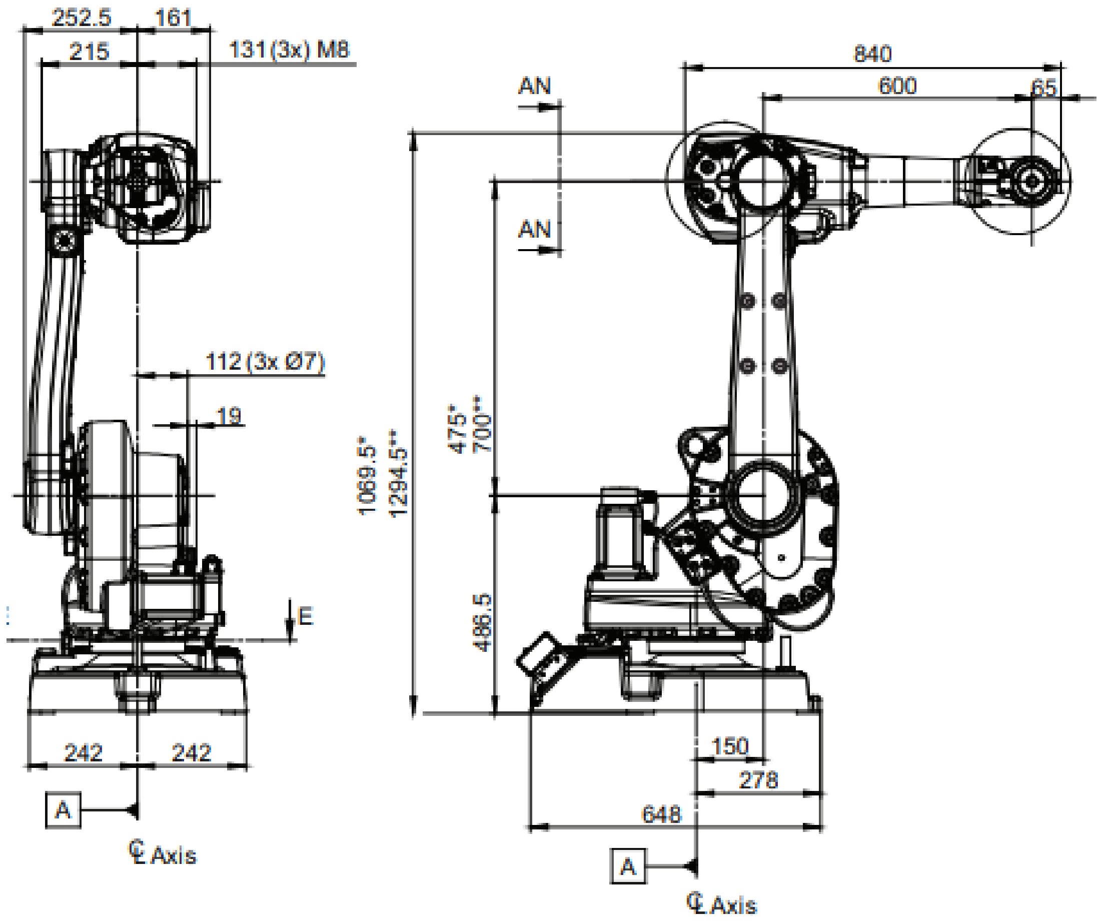


Figure 10: ABB IRB 1600 Measurements

System Architecture

We decided to build our system using ROS. The main reason behind this is that the highly modular nature of ROS lends itself well to integrating a variety of different components into our system such as the ABB controller, our EOAT microcontroller, CV camera, Microsoft Kinect, and simulated car computer. In order to plan our ROS nodes and topics we created a flow chart of our system as seen in Figure 10. A complete ROS generated chart of nodes and topics can be seen in the Appendix 1. The main state machine node was coded in Python using a ROS library called SMACH that allows nodes to be directly integrated into the machine. Each individual task the robot had to accomplish was defined and laid out in a flow chart which was then used to form the states themselves. Any given state can connect to a number of ROS nodes, so states were divided into logical processes rather than individual tasks. For example, in order to change between tools the robot must move to the tool change fixture, put down its current tool, and then pick up the requested tool. While this process involves 3 distinct actions, using SMACH userdata a single ‘ToolChange’ state was created to handle the entire process. The state receives the current tooling and desired tooling and then performs the change using hard coded poses based on the set location of the fixture held in a ROS node. The state machine we designed is shown in Figure 11. Each state has a single transition but have various error catching methods. For example, the Open Flap state will not transition until suction is detected. Otherwise it will continue to move forward in small steps after approaching the flap. However, if the force sensing package detects that the tooling is being pushed into an object it will stop all motion regardless of if suction is detected. The process of writing the states themselves is described in the testing methods section below, as it is dependent on the entire system architecture.

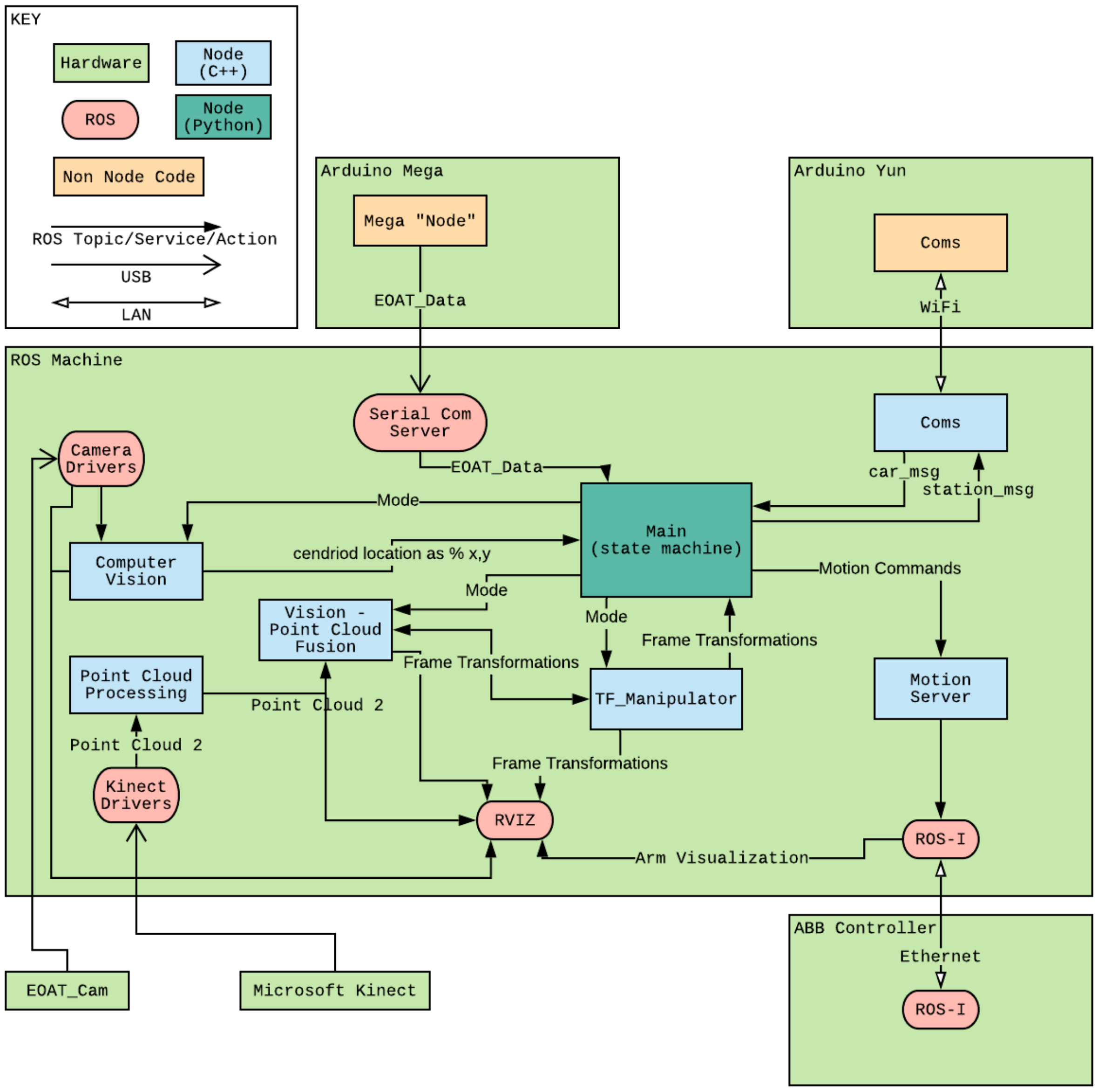


Figure 11: System Architecture

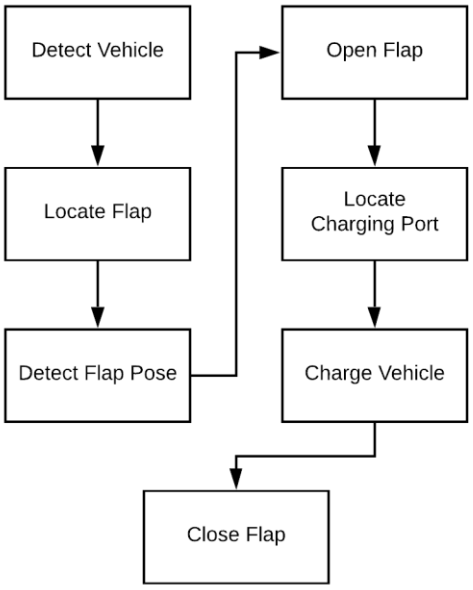


Figure 12: State Machine

Machine Vision (CV and point cloud)

Locating the vehicle quarter panel and charging port was a two-step process with redundancy. The panel was initially located using an openCV stereo camera mounted on the arm. This camera uses traditional CV to get the X and Y coordinates of the charging port flap in the camera's frame of reference. These coordinates were then combined with data from a point cloud provided by a Microsoft Kinect in order to add depth as well as 3D space orientation into the pose.

Computer vision was used in order to find the charging port flap’s location relative to the rest of the car. We used a ROS library to have our mounted webcam take an image of the car, which is then then converted into an OpenCV image so that it can be processed. Our CV program then analyzes said image to look for the location of the charging port flap and sends it back to our state machine as a percentage along the x and y axis of the camera. The state machine the uses this information to zero the camera in on the flap. The location of the camera once zeroed is obtained using forward position kinematic and saved for later use.

When looking at an image of the car, there is no reasonable way to distinguish the surface of the charging port flap from the surface of the rest of the car since they are the same color, and made of the same material. Instead of searching for the charging port flap itself, our program looks for the small gap around the charging port flap. The algorithm looks through the matrix of color values in the image, and finds colors that are black (or close to black). It then looks in the area surrounding that point to see if it is part of a black circular ring.

This solution is viable for the scope of our project, but there are situations where it will not work. The program is only looking for round gas flaps, so it wouldn't work on a car with a square gas flap. Also, the algorithm we are using specifically searches for a circular black ring surrounded on both sides by a non-black surface. This would not work if the car it's looking at is black. This problem could be avoided by affixing a brightly colored indicator to the gas flap, and searching for that instead if the car's color is determined to be too dark. If the algorithm runs and doesn't find a ring of the expected size, it can look for the easily located indicator, and compute the x and y coordinates based on that.

In order to get more information about the location of the charging port flap we are also using point cloud data gathered by a Microsoft Kinect. The Kinect was selected because it is relatively affordable, accurate, and easy to integrate into ROS. A Kinect driver node connects to the device and publishes a point cloud to a ROS topic. Two ROS nodes then clean and interpret the data using Point Cloud Library (PCL). A helper node called Point Cloud Processing then cleans the data by cropping out unnecessary points, down sampling the data, removing outliers, and correcting frame transformations. The cleaned point cloud is then published to an new ROS topic.

Next the Vision Cloud Fusion Node processes the point cloud into useful information for the robot motion planning nodes. The Fusion node subscribes to the saved zeroed-in position of the camera to obtain x and y position information provided by the computer vision node and to the cleaned point cloud published by the previous point cloud node. Using the x and y information from the camera, a diamond shaped cross section is created in the x y plane approximately the size the charging port flap. This shape is then projected along the z axis creating a rectangular prism that is used to cut out a subsample of the point cloud representing the surface of the car. This subsample is now an approximation of the surface of the charging port flap and can be used to gain information about its position in the z axis and the orientation of the normal of the surface in 3D space (see Figure 12 and 13). The z position is determined by finding the centroid of the flap sub-sample. The orientation of the flap is found using PCL's planar segmentation algorithm. This algorithm searches a given point cloud for a planar approximation within a certain tolerance. The object returned by this algorithm has fields for the slope of all three axis in the form "ax + by + cz + d = 0". Using the arctangent trig function, the roll, pitch, yaw can be determined and can be used to create a quaternion representing the orientation. Now that we have complete information about position and orientation we can create a frame transformation representing the coordinate system of the charging flap. The transformation is then published as a ROS topic and can be used by motion planning nodes as a target.

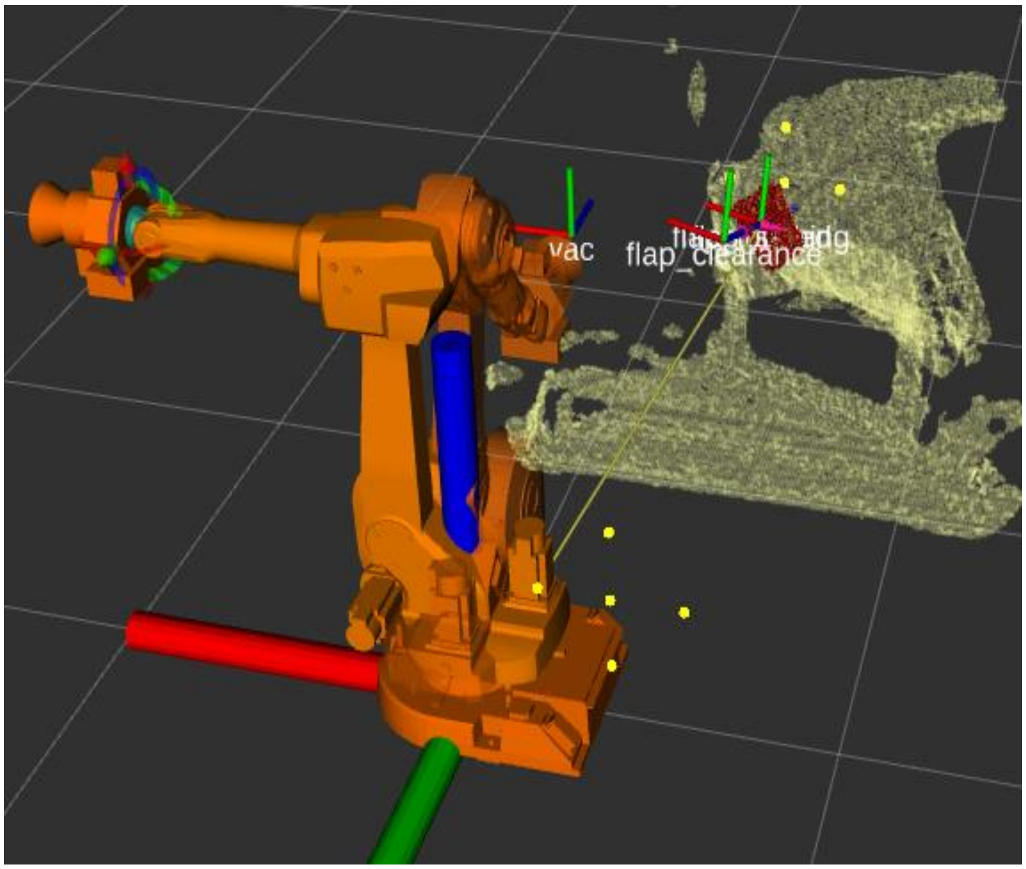


Figure 13: Point cloud representation of rear passenger quarter panel

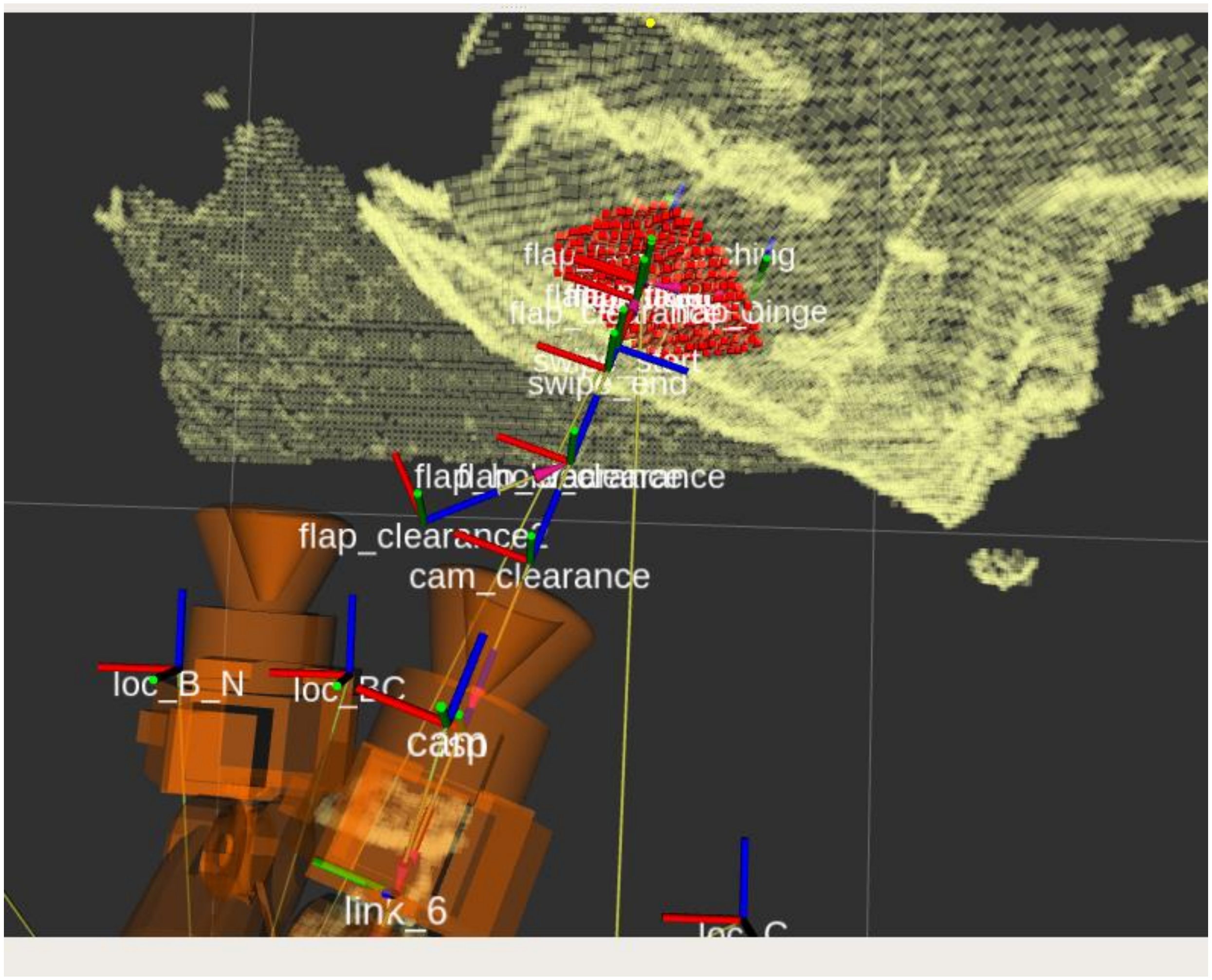


Figure 14: Point Clout with full set of frame transformations

Force sensing

The end of arm tooling has three built-in load cells to allow for dynamic force sensing. These load cells had to be integrated into a ROS node in order to get meaningful data from them. Each of the three load cells were connected to a load cell amplifier mounted to a breakout board on an Arduino mega. The mega then communicated the load cell information to ROS via an on-board ROS node over a USB cable.

Wireless Communications

As an additional functionality, communications between the robot and the vehicle were simulated using an Arduino Yún. Using the Wi-Fi capabilities of the Yún, simple packets of information were sent back and forth to ROS. The Yún sent information about the vehicle being charged including the model, charging port type, charging amperage, battery percentage, and whether the vehicle is finished charging. Basic profiles for different simulated vehicles can be selected with push buttons connected to the Yún. The robot then sends out a signal when it is ready to charge in order to open the charging port. While the vehicle is 'connected' and charging, another push button can be pressed to signal that the vehicle has finished charging. This then triggers the state machine to begin the process of unplugging and closing the flap. All of these communications were handled through a ROS node that read wireless serial messages from the Yún. The data was then published as a ROS topic so that other nodes could interpret and act on it. The Arduino code that accomplished this can be found in Appendix X.

Testing Methods

Workspace

The industrial robotic arm utilized in this project is an ABB IRB 1600-6/1.45 a 6 degree of freedom (DoF) arm with revolute joints. A model of the workspace and the functional range of the arm is shown below in Figures 14 and 15. The arm successfully connecting with the charging port flap can be seen in Figure 16. Also shown is the approximate position of the 'vehicle'. The rear passenger quarter panel of a 2014 Chevy Volt was used to simulate an electric vehicle. The internals of the gas port were repurposed to mount the female ends of the two charging standards: Tesla and J1772. A knee-level wooden platform with light fence mirrors surrounds the ABB arm, although it is capable of reaching beyond it with our tooling attached. Given the physical size of our end of arm tooling, the arm's reach is extended by roughly 0.385 meters giving a total range of 1.835 meters.

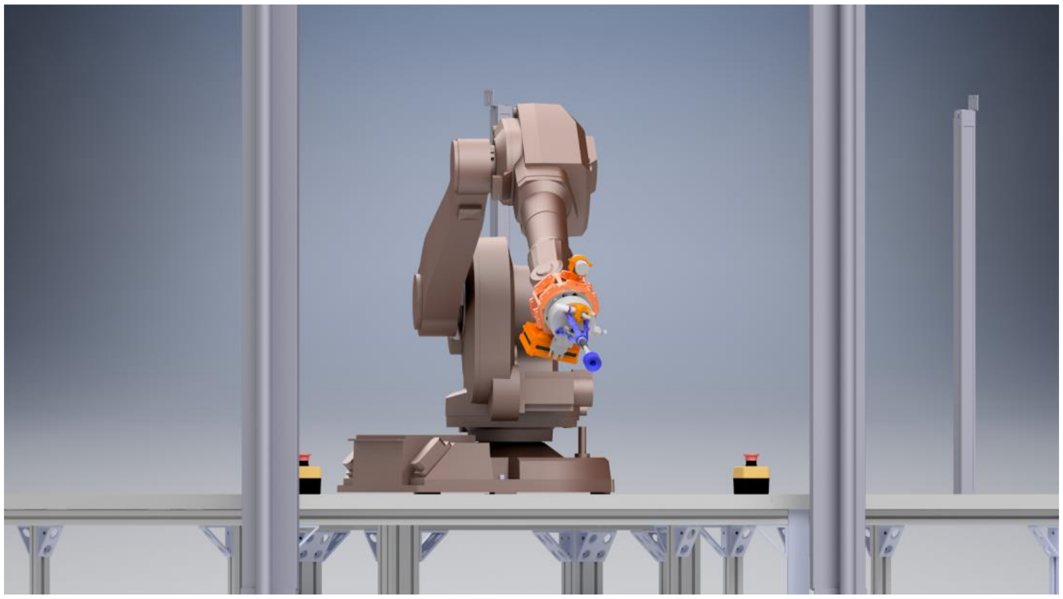


Figure 15: Head on view of ABB with pneumatic vacuum cup tooling inside workcell

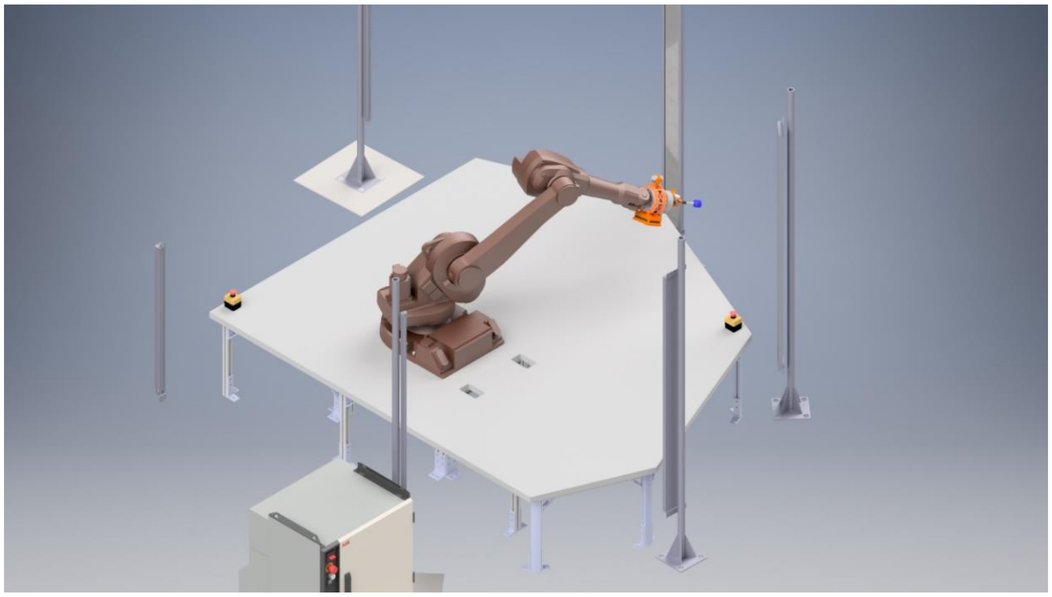


Figure 16: Isometric view of ABB with pneumatic vacuum cup tooling inside work cell

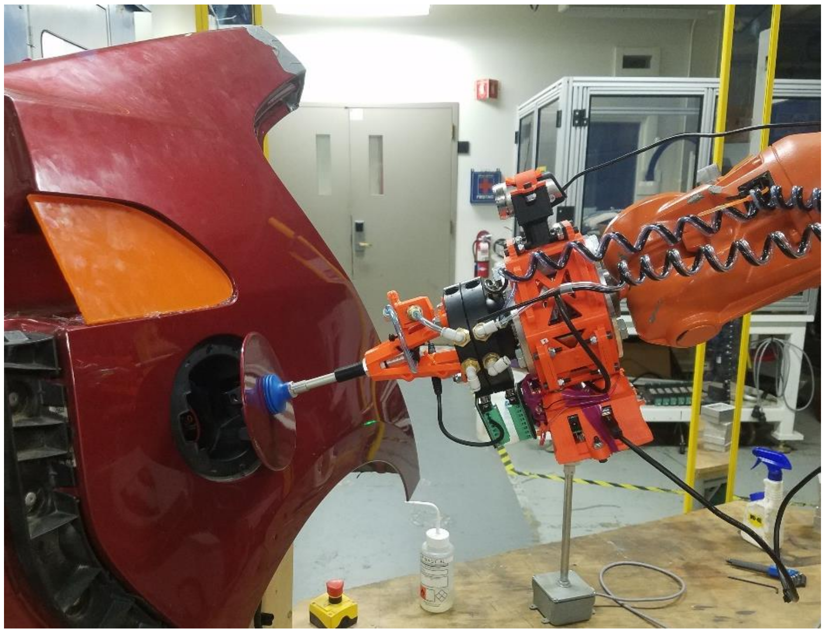


Figure 17: Quarter panel in work cell during flap opening procedure

Arm Control

Once the overall setup was completed we were able to begin directly writing code to open the flap and plug in chargers. Each state in the state machine was written utilizing the ROS services to locate the vehicle and flap, generate poses, and move to them. This was centered upon the helper function ‘get_pose_from_tf’ that listened to a ROS topic publishing transforms. It took two inputs, the base frame of the arm and the desired frame, and output two arrays containing the translation and rotation needed to move the arm to the desired frame. A goal object was then created from those two arrays and published to the motion client, moving the arm. The blocking call ‘client.wait_for_server()’ is called after each sent goal which prevents the machine from continuing, and potentially sending another goal, until the motion is completed.

Many states, such as ‘get_tool’, are almost entirely comprised of this process to move the arm. States such as ‘Find2DFlap’ and ‘Find3DFlap’ do move the arm but their primary function is to generate relevant poses using services by the process described in the CV and point cloud sections above. These poses become part of the tf topic which constantly publishes pose information, allowing new locations to be input to ‘get_pose_from_tf’

Results

While workplace and ROS setup took up the majority of this project, we did complete the intended task and have plenty of results to be discussed. Unlike prior sections, which were broken up by topic, this chapter is organized chronologically with respect to features as they were implemented.

Seurobot Things: Getting the action server working/URDF

During the second term of this MQP (B18) three members of the project were also enrolled in RBE4815: Industrial Robotics and as part of the class all three were part of the same final project group. The work that was completed toward the Industrial Robotics final project provided an additional opportunity to resolve issues with the ROS-Industrial setup used to control the ABB arm that would later become useful for this MQP.

For both this MQP and our industrial robotics project we wanted to be able to write the main control and decision-making code in Python. The reason for this was that Python offered several advantages in ease of programming and inclusion of useful libraries like ROS SMACH and easy reading of image files. However, the ROS Move-it libraries that have been implemented for the ABB IRB 1600 in ROS-Industrial do not have implementations in Python. In order to overcome this issue, we created our own ROS action server/client pair. Action lib is the highest level inter node communication standard. It allows for the sending of custom messages from a server to a client, feedback to the client during server-side execution and a final response by the server to the client. Our server was implemented in C++ and used a custom message of the structure (x, y, z, r, p, y, w, tool, mode). During execution the server responds with its percent completion and once done a Boolean of if the motion was successfully completed. Calls to this server can be simply and easily be made in Python. In addition to enabling our industrial robotics final project and our MQP we hope that a polished version of our action server ROS node can become a resource to future projects that want to use ROS with the ABB IRB 1600.

When we first got the ABB IRB 1600 moving to specific locations provided by our main ROS node we quickly realized that there was a very large error between where we told the arm to go and where it moved to. Initially we thought this was simply due to moving the robot in joint space as opposed to Cartesian space so we developed the ability to select between joint space (Move J in RAPID) and cartesian space (Move L in RAPID) motion commands. After implementing these useful modes however we still had significant error preventing us from moving horizontally which was very important for our industrial project. After much debugging by measuring the error in the z axis at different x, y locations we realized that this error was always greater in the plane of the second and third joints of the robot. This behavior suggested that perhaps there was a difference between the physical robot and how the robot was defined in software. Upon further investigation we realized that the URDF (Universal Robotic Description Format) file which defines the locations for each of the joints of the robot was for the ABB IRB 1600-6/1.2 that had a reach of 1.2 meters, while the arm we are using is the 6/1.45 version with a 1.45 meter reach. The two models are identical except for the length of the second link. Correcting the URDF to have the correct length for link two resolved our inaccuracy issues.

Point Cloud and CV

To maximize the efficiency of the vehicle flap localization process we used the camera located on our EOAT to identify the XY coordinate of the centroid of the flap. This centroid is then used to determine the location of the flap within the point cloud via frame transformations.

Once the flap is localized in the point cloud we were able to determine the pose the vacuum cup tooling should use to approach to properly open the flap of the vehicle.

The CV program that located the vehicle flap had consistent and accurate results. The variance in the program was not enough to throw off the position of the suction cup to the point where the flap would not open. In repeated trials of the flap opening behavior, angles of up to 45 degrees off the camera plane could be introduced and the flap would still be successfully detected and opened. Angles this large would cause an approximately 1.25-inch error in how centered the suction cup was on the charging port flap. Such variance did not cause issues as the 2.5 bellow suction cup provided considerable tolerance when opening the flap. The returned location was almost always within a couple of centimeters of the actual location. In order to combat any outliers that might happen, the CV service looped a number of times, and computed a rolling average of the flap location. The only factor that presented a potential accuracy problem was visual noise and glare in the image, but this could be tuned out to some degree by applying certain blurs and filters to the image. Figure 17 shows an example of the CV's approximation of the flap and the image used.

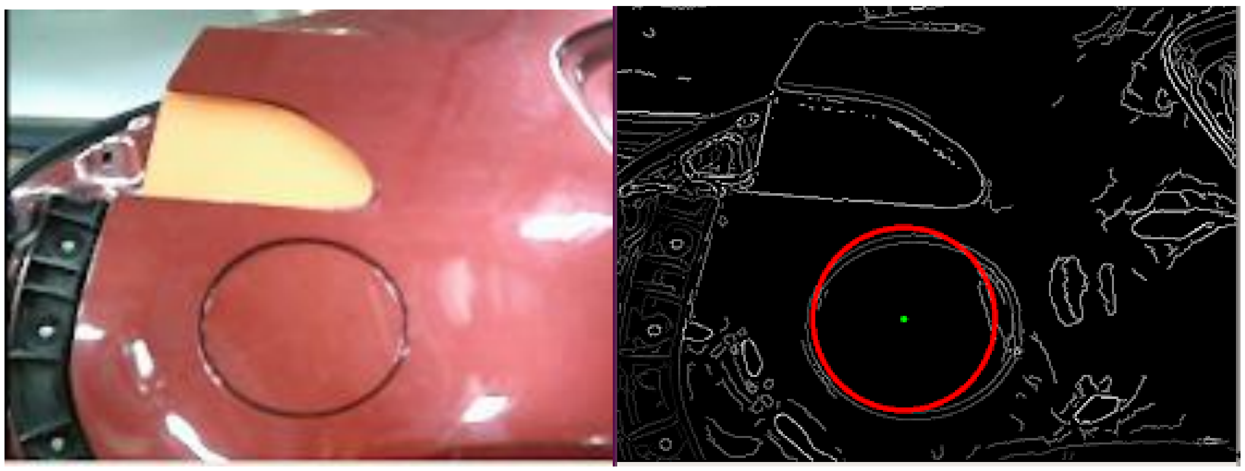


Figure 18: Computer Vision Flap Locating

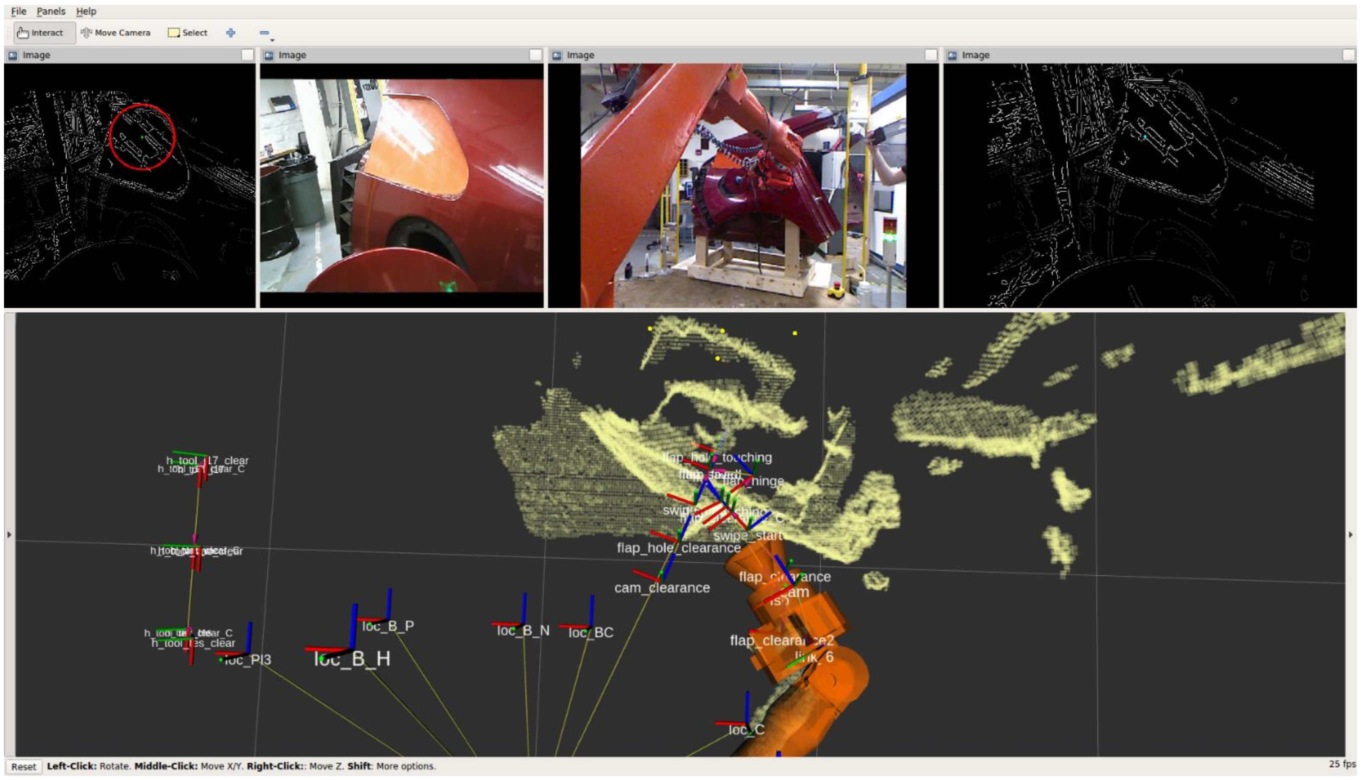


Figure 19: Opening Flap Frame Transformations

The pose and depth information provided by our fusion of our 2D-vision and our point cloud data were able to consistently locate the flap and its pose. Angles for roll, pitch, and yaw proved to be within 3 degrees on average when measured in four significantly different locations. This proved to be more than enough for the flap opening operations, which we were able to accomplish with very high accuracy. With the quarter panel roughly perpendicular to the Kinect it could locate the flap with 100% accuracy, across dozens of test runs, and was able to open it with 80% accuracy. The frame transformations used for clearance and touching targets can be seen in Figure 18. As the quarter panel was angled further away from the Kinect the flap location remained highly accurate but the arm was not able to successfully open the flap as consistently. At an angle of roughly 60 degrees, the robot was only able to open the flap at a 50% rate.

Plugging in the charger itself had a smaller success rate however, as the tight tolerances of the actual charging port proved to cause reliability issues. The flap opening had significantly more tolerance, as the size of the suction cup allowed it to bend if the arm had to move at an odd angle. The port itself had almost no tolerance, and succeeded in only 60% of tested vehicle locations.

Peripherals: Communications and Force Sensing

Autonomous vehicles can communicate with other peripheral devices and in a 'real' situation, an autonomous charging station would be able to get information and identification from an approaching vehicle. Different vehicles may not have the same communication protocols, but we decided to simulate a vehicle using an Arduino Yún to represent this potential function. Packets of information were transmitted using onboard Wi-Fi that was then received by the ROS node coms_node and published to the topic coms_msg. This message consisted of the vehicle type, the port type, the maximum charging level supported, the battery percentage (which was randomly generated), and whether the flap can open by itself or not. Most of the fields in these packets were placeholders to demonstrate that the information could be transmitted, but the charger type was utilized in the control logic. As a secondary feature, the Yun was later used to trigger the state machine unplugging the vehicle and closing the flap to add further realism.

Due to budget constraints, all of our testing was conducted on a quarter panel from a 2014 Chevy Volt. This meant that we only had to handle a single type of flap. However, different vehicles can have not only different sized and shaped flaps, but also varying opening mechanisms. While our design had no way to account for this, it is potentially possible to utilize vehicle communications to convey this. This would require cooperation from manufacturers and is a more far-reaching assumption. Our system as it currently functions may be able to account for varying flap shapes and locations, but if it were to receive any form of location from the vehicle it could better find the needed poses.

One of the main issues we encountered during this project is the latency of communicating between the Linux computer and the two Arduinos used to calculate the force sensing and emulate the vehicle. This latency was not a concern with the vehicle emulation communication, as the messages sent between the vehicle and the Linux PC are infrequent and are not time sensitive. The force sensing messages, on the other hand, are time sensitive as they were intended to be used as a safety feature to ensure the arm did not exert too much force on the vehicle in any given direction. Initially, we were concerned that any force messages sent from the Arduino Mega to the computer would be too delayed to be useful, but once the ROS publisher and subscriber were initialized messages were received 3-4 times a second. This is an acceptable amount of latency for our purposes, but it still took the arm roughly a second to react and stop moving and is not quite up to industrial safety standards. To solve this latency issue with the force sensing one possible solution would have been to program the Arduino to output a digital high signal when a force greater than desired is detected, this signal would be then input to a circuit connected to the robots E-STOP functionality, essentially preventing the arm from ever reaching dangerous forces.

Physical Testing

Once the state machine was set up and the various services written preliminary testing was conducted to open the flap. We immediately noted a significant amount of latency between each motion of the arm. Every time a goal is sent a transform is calculated then sent to the action client which then sends the motion commands to the arm. ROS topics and services are known to have lots of overhead, so the layers of nodes needed to perform a simple motion expectedly takes around 3-4 seconds to even begin. In order to account for these delays after every move command, regardless of the state, the command ‘motion_client.wait_for_server()’ was called to block the machine from continuing until the arm had completed the trajectory.

As is often the case with robotic arms, we also ran into issues with singularities causing inconsistent and unpredictable motions. The arm was usually started in the ‘all zeros’ position, which itself is a singularity, and when moving between the typical vehicle locations and tool change station ran directly through a singularity. Both situations were able to be avoided by moving with respect to joints, rather than linearly with respect to the tool, to intermediary locations. We were able to consistently move the arm in a more indirect path around the singularities whenever starting or going back and forth from the tool change station. It was also discovered that it is not currently possible to toggle the pneumatics on our arm through ROS. This meant that they had to be manually toggled while the arm was running, which is less than ideal but unavoidable.

Additional assumptions and limitations

There were a handful of further limitations and assumptions that had to be made in addition to those already mentioned. The most significant assumptions are related to the charging port flap and charger. First and foremost, it is assumed that the charging port has the same orientation as the flap itself. Our CV is unable to get an accurate pose of the port once the flap is opened because it is all black and is not distinguishable enough to find any features. This is not an issue for some models of vehicles but is not universal. Similarly, not all flaps are the same and may have different opening mechanisms. Our solution is serviceable for this particular case but again is not applicable to all vehicles.

We also have a limited range of detection for the quarter panel itself. The Kinect is statically mounted and while it has a wide angle of vision can only get accurate poses if the vehicle is within a limited cone. This effectively gives about a foot of variance which is not entirely unreasonable. It could be assumed that an autonomous vehicle would be able to drive up and park in a specified location with about a foot of variance.

Overall, while our project is not economically viable in the present, through our results we have demonstrated that such a system is feasible. We have also highlighted some of the difficult aspects of the project that would need to be further developed in order to fully realize this technology.

Economic Analysis

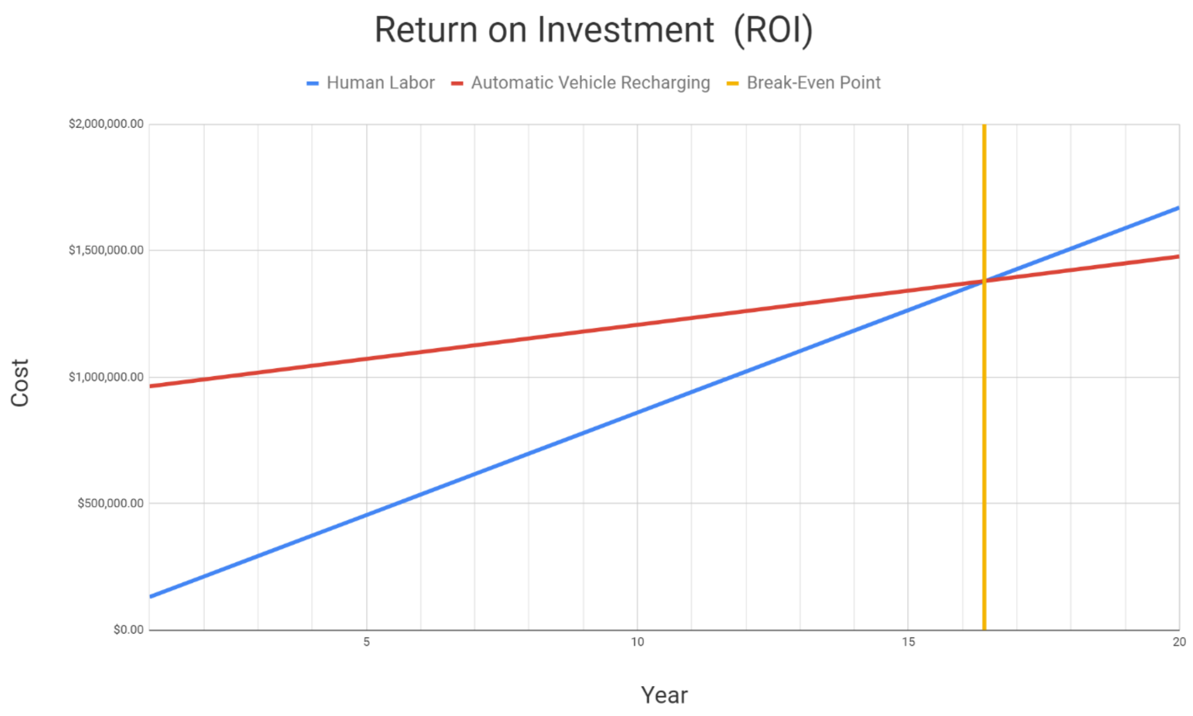
As initially expected, there is very little economic incentive to implement such at technology in the current day. Autonomous vehicle technology is not currently developed to a state where there would be demand for high throughput autonomous charging solutions. Furthermore, the estimated development costs that would be required to develop such a product to market quality are much too high to provide a return on investment within a reasonable timeframe. As such, this project should be viewed as a proof of concept and exercise in practicing the relevant skillsets that would be required to develop such a project. Such skills include developing computer vision software, point cloud processing software, and doing motion planning for industrial robotic arms. Such skills are applicable to many robotic applications.

In order to determine the economic feasibility of bring the product to market we did a return on investment calculation (Table 3 and Chart 1) and found that it would take over sixteen years to reach the breakeven point when compared to human labor doing the same task. As such we do not feel it is currently worth pursuing our implementation as a market solution.

Table 3: ROI Calculations

Expense	Calculation	Total
Yearly Cost of labor	$15.0*(135%)*(8*2*250) (Wage)*(Overhead)*(work hours)	$21,015.00
Cost of level 3 charging station	Source	$50,000.00
Software Engineer Wages	Source	$76,000.00
Cost of robot setup	(cost of robot)+(Cost of level 3 charging station)+(software engineer wages*four employees over two years)	$958,000.00
Robot Operating Cost	Cost of electricity per day for industrial robot	$5.00
Cost of Robot over Time	(Robot Setup Cost - Yearly Cost of labor)+(Robot Operating Cost)*(135%)*(8*2*250)*(Year)	N/A
Cost of Labor over Time	$15.0*(135%)*(8*2*250) * (Year)	N/A

Chart 1: Return on Investment



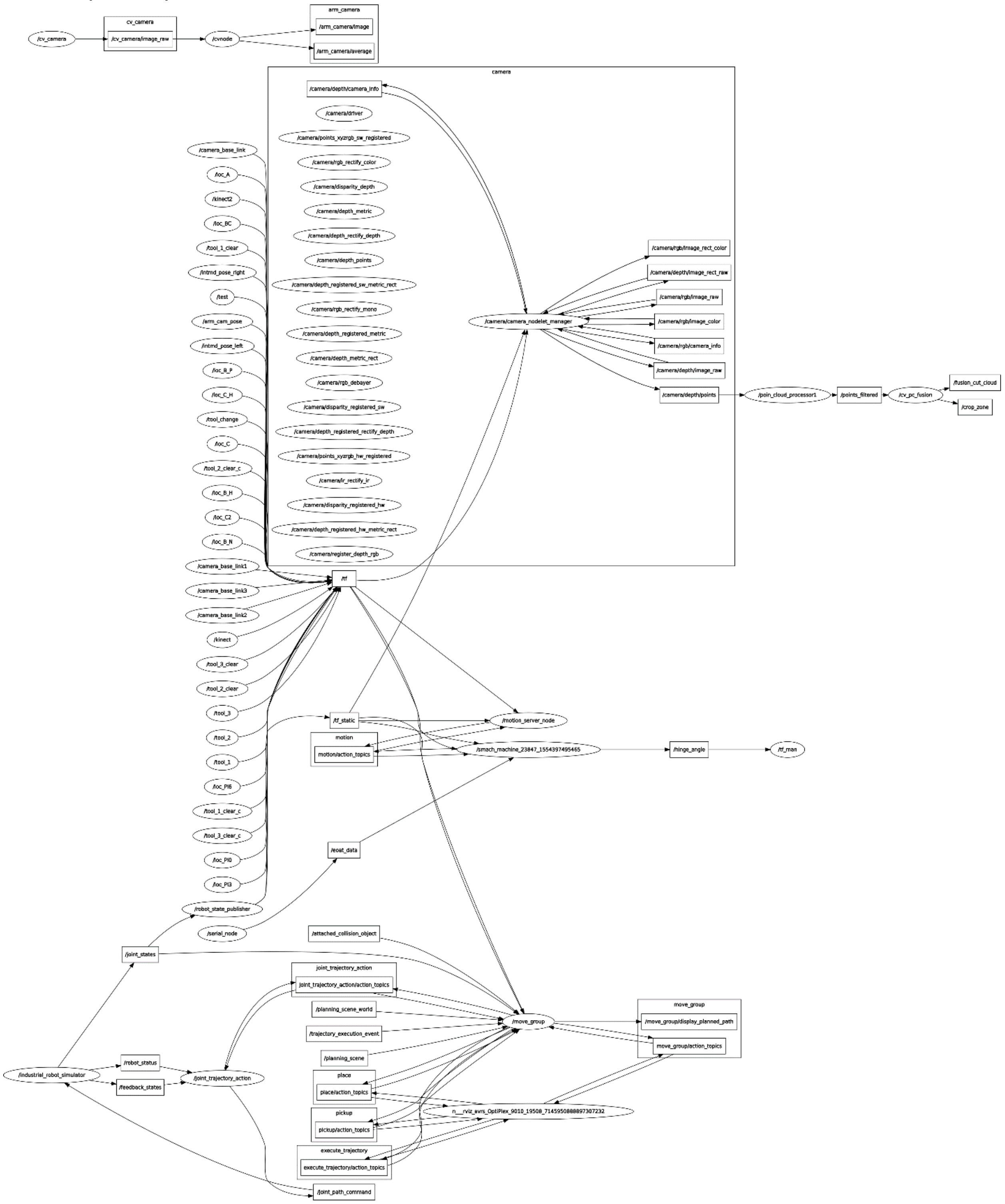
Conclusions and Future Recommendations

While we satisfactorily completed our initial objectives, there is further work that could be done to build upon and improve our implementation. The most obvious expansion would be adding compatibility for more charging standards than just J-1772 and Tesla, namely the CHAdeMO charging standard. Having more than one quarter panel for testing would greatly increase the robustness of test cases, particularly for CV and point cloud location which were tuned specifically for our single quarter panel. As mentioned in the previous section, the charging port flap varies in both size and location between vehicles and our system could be expanded upon to allow for a variety of vehicles. To accommodate for a wider variety of vehicles and potentially identification of the charging ports themselves a higher resolution camera could be used. We used a web cam which, while serviceable and budget compliant, was certainly not the best option.

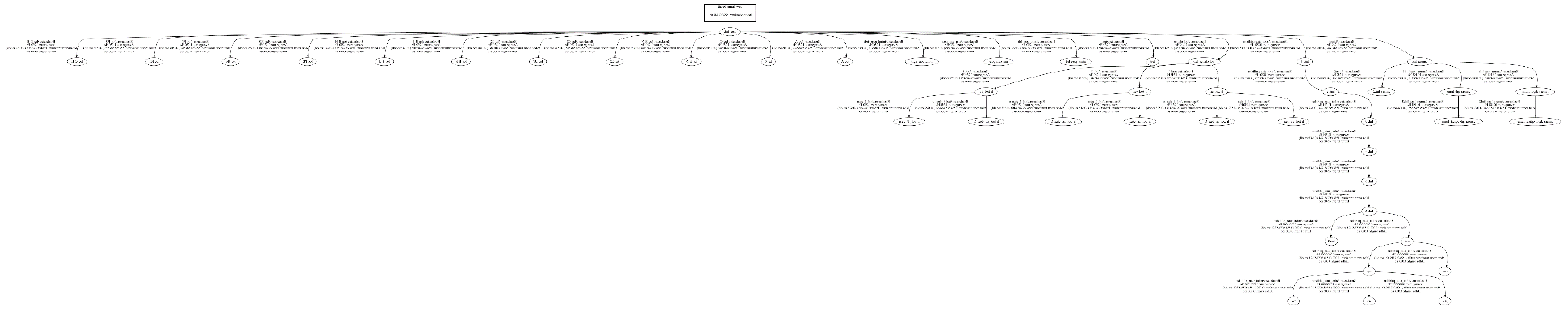
Another area of the project that could be taken further is the safety features built into the system. We implemented force sensing that prevented the arm from applying a dangerous amount of force to anything that may be in the workspace. However, this was the only notable safety feature other than the light fence installed around the arm in Washburn shops. The already functioning point cloud functionality could be used to identify and locate human bodies or any unexpected objects other than the vehicle that enter the arm's workspace while it is running. Additionally, it may be possible for the vehicle to communicate with the station whether any doors are ajar for any reason. Our implementation does not have any ways to handle unexpected interference other than halting all movement after already contacting something incorrectly; a 'real' version of this technology would require more robust sensing and safety features like those mentioned.

Appendix A: ROS Graphs

1. ROS Topic Graph



2. ROS Transform Graph



Appendix B: Charging Scenarios

The following is a list of potential charging scenarios that are being considered in Table 1

1. User has vehicle plugged in to charge fully
    a. User decided to leave early, needs unplug before completion
    b. User leaves car plugged in beyond necessary charging time to allow for vehicle to draw power from plug for auxiliary behaviors (climate control, etc..)
    c. Vehicle is fully charged and plug is removed
2. User has vehicle plugged in with intention of leaving at time xx:xx
    a. User decided to leave early, needs unplug before completion
    b. User specified plugged in time leaves car plugged in beyond necessary charging time, allowing for vehicle to draw power from plug for auxiliary behaviors (climate control, etc..)
    c. User specified unplug time unplugs car while not fully charged
3. User has vehicle plugged in for specified amount of time XX minutes
    a. Time is elapsed
    b. User wanted vehicle unplugged before time is fully elapsed
4. User has vehicle plugged in with intention of being unplugged at a certain charge
    a. Charge amount is reached
    b. User decided to leave early, unplug before completion

References

Acosta, L., Blanco, R., Haber, R., Villagra, J., Naranjo, J., Navarro, P., Sánchez, F. (2018). *Automated driving*

Becker, A. T., Sidhu, I., & Tenderich, B. (2009, Aug 24). *Electric vehicles in the united states: A new model with forecasts to 2030*. University of California, Berkeley.

Bensinger, G., & Dawson, C. (2018, Aug 28). Toyota investing $500 million in uber in driverless-car pact. *Dow Jones Institutional News*

Bomey, N. (2018, Feb 15). U.S. vehicle deaths topped 40,000 in 2017, national safety council estimates. *USA Today*

Camhi, & Jonathan. (2017, Nov 22). Uber to buy up to 24,000 volvo SUVs for self-driving fleet. *Business Insider*

Critical reasons for crashes investigated in the national motor vehicle crash causation survey : Traffic safety facts: Crash stats; 2015 ASI 7766-19.56; DOT HS 812 115. (2015). Washington, DC: NHTSA National Center for Statistics and Analysis.

David Rotman. (2013, May 15). How technology is destroying jobs. *MIT Technology Review*.

Davis, M. R. (2018, Mar 26). Electric cars vs. gas cars: Comparing maintenance &amp; battery costs. *Ez-Ev*.

Erik Brynjolfsson, & Andrew McAfee. (2012, 12/11/). Jobs, productivity and the great decoupling. *The New York Times*.

Erik Brynjolfsson, & Andrew McAfee. (2015, June 1). Will humans go the way of horses? *Foreign Affairs*.

Fagnant, D. J., & Kockelman, K. (2015). Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations.*Transportation Research: Part A: Policy and Practice, 77*, 167-181.

Ford hives off self-driving operations. (2015, Jul 24). *Dow Jones Institutional News*

Fuelmatics: Leading technology for true automation. Retrieved from http://fuelmatics.com

Genovese, A., Ortenzi, F., & Villante, C. (2015). On the energy efficiency of quick DC vehicle battery charging. *World Electric Vehicle Journal, 7* (EVS28 International Electric Vehicle Symposium and Exhibition)

Gurumurthy, K. M., & Kockelman, K. M. (2018). Analyzing the dynamic ride-sharing potential for shared autonomous vehicle fleets using cellphone data from orlando, florida. *Computers, Environment and Urban Systems, 71*, 177-185.

Lamm, M. (1998) Full service without a smile. *Popular Mechanics*, *175*, 62.

Nathan Bomey. (2018, 2/15/). U.S. vehicle deaths topped 40,000 in 2017, national safety council estimates. *USA Today*.

National Society of Professional Engineers. (2011). Code of ethics for engineers (2007). Retrieved from http://ethics.iit.edu/ecodes/node/4098

Number of plug-in electric vehicle sales in the united states from 2012 to 2018 (in 1,000s). (2018). Retrieved from https://www.statista.com/statistics/801263/us-plug-in-electric-vehicle-sales/

Onar, C. O., & Jones, P. T. (2015). *Wireless charging of electric vehicles* Oak Ridge National Laboratory.

Power hydrant. Retrieved from http://www.powerhydrant.com

Saxton, T. (2011). Understanding electric vehicle charging. Retrieved from https://pluginamerica.org/understanding-electric-vehicle-charging/

Sierzchula, W., Bakker, S., Maat, K., & van Wee, B. (2012). The competitive environment of electric vehicles: An analysis of prototype and production models. *Environmental Innovation and Societal Transitions*, *2*, 49-65.

Sierzchula, W., Bakker, S., Maat, K., & van Wee, B. (2014). The influence of financial incentives and other socio-economic factors on electric vehicle adoption. *Energy Policy, 68*, 183-194.

Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems (2014). Status: Current version. Revision #: GXGFHFAAAAAAAAAA.

U.S. vehicle deaths topped 40,000 in 2017, national safety council estimates. (2018, 2/15/). *USA Today*

Voelcker, J. (2009). Better place video showing EV battery-pack quick swap: We're not convinced. *Green Car Reports,* Retrieved from https://www.greencarreports.com/news/1020699_better-place-video-showing-ev-battery-pack-quick-swap-were-not-convinced

Voelcker, J. (2014). Standardized electric-car battery swapping won't happen: Here's why. *Green Car Reports,*

Walker, J. (2018). The self-driving car timeline – predictions from the top 11 global automakers. Retrieved from https://www.techemergence.com/self-driving-car-timeline-themselves-top-11-automakers/

Wladawsky-Berger, I. (2018, Jun 1,). From horseless carriages to driverless cars -- are you ready? *The Wall Street Journal* Retrieved from https://blogs.wsj.com/cio/2018/06/01/from-horseless-carriage-to-driverless-car/

Ziegler, C. (2016, June 27,). Tesla owner makes the 'solid metal snake' self-charging system that elon musk promised. *The Verge* Retrieved from https://www.theverge.com/2016/6/27/12040026/tesla-solid-metal-snake-elon-musk-charger