Applications of Robotics: A Quadrupedal Robotic Solution to Aid in Search and Rescue Operations

Alfred W. Martinez III
Electrical Engineering
California State University, Sacramento
alfredmartinez555@gmail.com

Kristian Ornelas
Electrical Engineering
California State University, Sacramento
krisornelas85@gmail.com

Edgar Granados
Computer Engineering
California State University, Sacramento
edgargranados153@gmail.com

Marcus Huston
Computer Engineering
California State University, Sacramento
marcushuston@csus.edu

Table of Contents

Table of Figures:

Table of Tables:

Executive Summary

Our project is a semi-autonomous robotic quadruped to help aid in search and rescue and other operations. In this report we provide a detailed explanation of the work performed and our experience with this project over our timeline from Fall 2019 to Spring 2020. Our goal for our project was centered around the idea to help workers in the field of search and rescue, and to help prevent those people being in dangerous situations. The inspiration behind the creation of a robotic quadruped was looking at the world's problems and understanding that there are disasters throughout the world. Once we had a better understanding of the societal problem, we had to figure out a way to solve the problem or best help solve the problem. Since disasters, whether it be natural or not natural are common throughout the world and not preventable, we wanted to focus on something that is achievable and can work to help when those disasters occur.

Our milestones consisted mainly of the creation of our features. Once we were able to complete each feature one at a time, we knew that we were one step closer to completing our design. Some of the big milestones were completing the first prototype as well as the second prototype to show the progress we had made as well as giving us inspiration to work towards a final product. Also, big milestones were the December showcase because of the valuable feedback given for the rest of our project and what we needed to do in order to succeed with our final design. Each aspect of our robotic quadruped had risks. Though most of the project consisted mainly of software, the hardware parts of the project had the most risks. This is because the software errors would only be compiling errors and mostly from just testing the software if it works. Our hardware is a different problem though since the robotic quadruped is a movable object, parts move a lot and they could break with random falls or become short circuited without much notice. There is also the issue with building our second prototype is the amount of power consumption as well as stress on the parts. After diagnosing and finding the broken part, it could take days or weeks to replace our items. All and all, we did not run into many issues throughout the product that set us back to far. The main issue that we ran into was the lack of time to work on and test our robotic quadruped since it was so complex and a big learning curve for a lot of the software.

Many factors have affected the ability to 'roll-out' the project as a fully formed product, namely the resources that are available for use as well as the time to completely get the project fully operational. All measurable metrics have been met throughout the project's timeline. To improve on our design, we believe that the main features are solid and would just need more refinement. More things need to be added to software to be more consumer friendly and more inviting as well as the capabilities of the product. Hardware wise, we could look at using different boards and microcontrollers to make the robotic quadruped cheaper as well as smaller. Also, more fabrication of better integration of our robotic quadruped could be made to create a nicer more robust robotic quadruped.

Overall, this document walks you through how we got the idea to solve the societal problem as well as the tasks it took to complete the robotic quadruped. The product specifications will be discussed to allow the reader to provide insight about the cost, purpose, and ability to compare the project to similar products. Throughout the report, you will also find how we broke down each part and combined them into one product. Also, this document provides the details about the various robotic industries to provide context as to how our robotic quadruped will fit in the real world

MegaByte: A Robotic Dog to Aid in Search and Rescue Operations

Alfred Martinez III, Kristian Ornelas, Edgar Granados, Marcus Huston
Department of Electrical & Electronic Engineering, Computer Engineering Program, CSU Sacramento
6000 J Street, Sacramento, CA, USA

***Abstract*— As disasters continue to happen every year throughout the world, the increase in search and rescue workers continue to rise as well as the danger they put themselves into. We aim to create a design to help lower the chances of injury and keep more people safe by creating a semi-autonomous robotic quadruped. This document is intended to summarize our Fall 2019 and Spring 2020 semesters, in terms of the technical work we have completed. The paper starts by explaining the societal problem and followed with the design idea as a proposed solution. In addition, we highlight the funding that was required to make this project possible as well as the cost it will take to build the robotic quadruped. We also discuss the project milestones and their significance, as well as the work breakdown structure, which includes the risk assessment and task assignments for each team member. Included, we have documentation for our design, which includes the hardware/software block diagram and schematics. We have also included any mechanical drawings and renders that were used in our project. Additionally, we included our test plan along with the test results. This report also touches on some of the possible applications our robotic design could be used for. This includes specifications outlining a relative cost of manufacturing and developing, analysis into the current Robotics Industry, as well as the top competitors in the industry. This document has information about the problem statement to the design and all the way to the finished product of our senior design project.**

***Index Terms*— Robotics, Quadruped, Search and Rescue, Disaster Worker, Robot, Open Source, ROS, DC Motor, Semi-Autonomous, Machine Vision, PID Control**

I. Introduction

During the first semester of Senior System Design, our group was tasked with designing a certain product that would be able to tackle a real-world societal issue. Initially, like many other groups, we struggled to bring about a realistic solution that would successfully be able to address one of societies' problems. Nonetheless, when a team from a different engineering department (Mechanical Engineering) came in to present their vision for their senior design project, our team became very intrigued of the possible project that could be developed with both teams working together. The ME team proposed an intercollegiate collaborative project in which the electrical/computer engineering team would assist in designing the control and feedback system of a quadrupedal robot. Robotics has a wide variety of applications in the real world; in fact, in today's modern day, you can find an application of a robotic system in nearly everything imaginable. Whether it'd be in the medicine, logistics, or manufacturing field; robotics has shown to be very impactful in everyday human life, and as further development is done, will become even more essential in dealing with societies' necessities. Thus, for our team finding a solution for a societal issue no longer became a problem. It was choosing which societal issue our project would best be able to solve. In the end, our team settled on designing a robotic quadruped that would be able to operate semi-autonomously to aid in search and rescue operations. Search and Rescue Workers, Disaster Relief Workers, and Firefighters are all typically put into highly dangerous situations which prevents them from helping victims as thoroughly as they would otherwise like to. The idea of a semi-autonomous robotic quadruped is to help save lives of victims and search and rescue teams that put their lives in danger every day. In our endeavor to design our quadruped robot, our team ultimately ended up creating the control and feedback system for a small model robotic system that would be able to sustain its walking movement, through the help of sensor modules. It was originally planned that the final deployable prototype would be a large-scale model, roughly 3 feet tall, 100+ lbs., and able to move semi-autonomously; however due to the worldwide

coronavirus pandemic, our development time was impacted, and our final deployable prototype was a small-scale model system and the control/walking algorithm for a large-scale model leg. This document will present the design and technical information that was conducted by our team in our attempt to design a semi-autonomous quadrupedal robot.

II. Societal Problem

A. Fall/Spring Societal Problem

1) Introduction

This section studies the situations as of Fall 2019 in Disaster Relief and Search and Rescue, S.A.R., efforts around the globe and attempts to find areas where improvements can be made based on existing models and what is achievable and economical. Several branches of SAR Robotics were studied, these were sorted into Aerial, Ground, and Marine. Furthermore, these categories were divided into disaster relief, search and rescue, firefighting, and crisis management.

2) Natural/Non-Natural Disaster Statistics

This section presents all the necessary statistics regarding the amount of disasters that occur throughout the world, indicating that there is a problem.

a) *Wildfires*

This subsection provides numerous statistical datasets regarding wildfires. Wildfires are a form of ground search and rescue; also, other forms of fires can devastate urban areas to include them in urban search and rescue. A recent example would be "The Camp Fire" in Paradise, CA. They had to evacuate 52, 000 people throughout the city and surrounding area. The fire lasted for two weeks and burned across 153,000 acres.

(1988-2017)

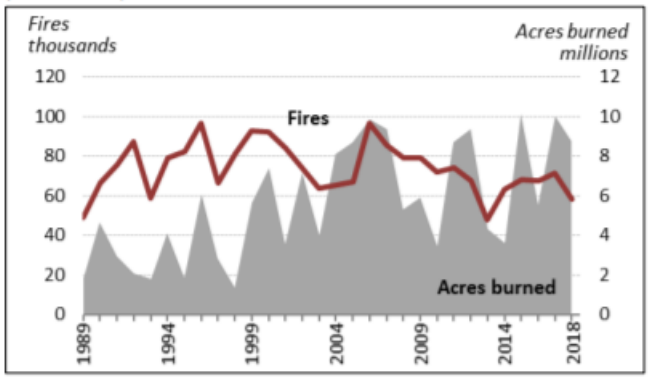


Figure 2.1. Number of wildfires and acres burned in the U.S. from 1988 to 2017. Reprinted from "Wildfire Statistics."

Source: Please refer to [27]

About 14,000 residences were destroyed and about 18,000 buildings in total. Multiple California Urban Search and Rescue (US&R) Task Forces and human remains canine search teams were deployed from across California to assist law enforcement with the search for, and recovery of, victims missing throughout Paradise and other towns devastated by the fire. Throughout the two-week span, the wildfire killed 85 people. Even two weeks after the fire, search and rescue crews were still trying to find hundreds of people.

	2014	2015	2016	2017	2018
Number of Fires (thousands)					
Federal	13.0	13.8	12.6	15.2	12.5
FS	*6.8*	*7.1*	*5.7*	*6.6*	*5.6*
DOI	*6.1*	*6.6*	*6.8*	*7.3*	*7.0*
Nonfederal	50.6	54.4	55.2	56.4	45.6
Total	**63.6**	**68.2**	**67.7**	**71.5**	**58.1**
Acres Burned (millions)					
Federal	2.15	7.41	3.00	6.3	4.6
FS	*0.87*	*1.92*	*1.25*	*2.9*	*2.3*
DOI	*1.24*	*5.47*	*1.70*	*3.3*	*2.3*
Nonfederal	1.4	2.72	2.51	3.7	4.1
Total	**3.60**	**10.13**	**5.51**	**10.0**	**8.8**

Tab. 2. 1 Table format of Fig. 2.1 Forest Service (FS) and Department of Interior (DOI).

Source: Please refer to [12]

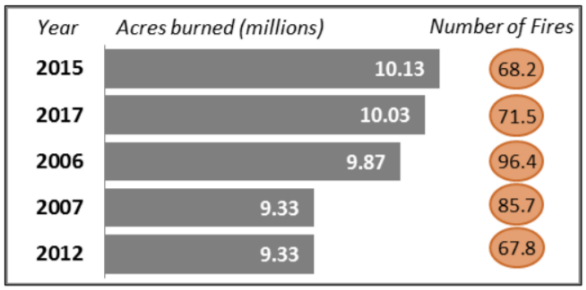


Figure 2.2. Top five years with the largest number of acres burned since 1960 in U.S.

Source: Please refer to [27]

	2015	2016	2017	2018
Personnel				
FS Firefighters	10,000	10,000	10,000	10,000
DOI Firefighters	3,997	4,129	4,514	4,492
Losses				
Firefighter Fatalities	13	12	14	19
Structures Burned	4,636	4,312	12,306	25,790

Tab. 2.2 Number of personnel and losses for FS and DOI

Source: Please refer to [12]

Up to this point, all the data represented occurs within the U.S., but wildfires are an issue experienced worldwide. Fig. 2.5 shows the 2017 wildfire mapping of the burned and unburned areas of Pedrógão Grande, located in Portugal.

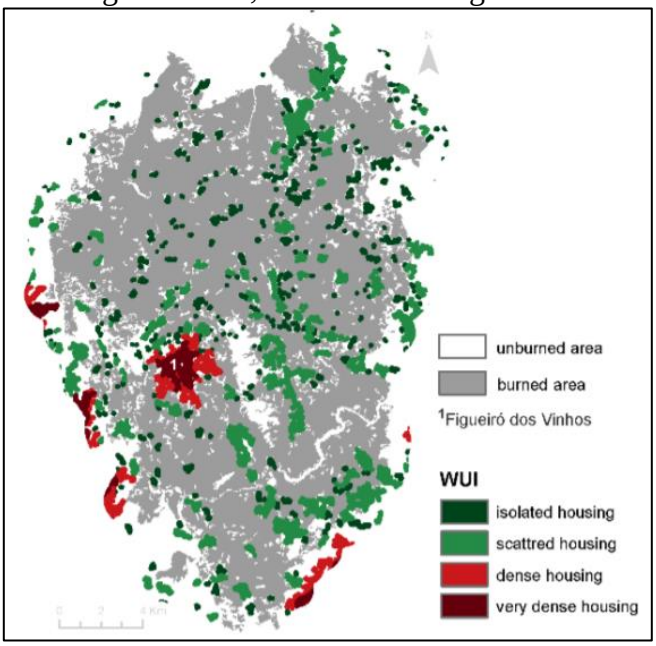


Figure 2.3. Burned/Unburned mapping of Pedrógão Grande in 2017

Source: Please refer to [14]

b) Earthquakes

This subsection presents statistical data regarding earthquakes and other relevant earthquake-related information.

One of the most recent big earthquakes was the 2011 Tohoku 9.0 earthquake which later caused a Tsunami to hit the city. The combined total of confirmed deaths and missing is more than 16,000. Due to the earthquake and tsunami, the Japanese government ordered over 100,000 troops, a total of 236 aircrafts and 50 vessels/cargo planes were deployed to aid in the field. This incident is classified under the form of water and urban search and rescue.

Worldwide Earthquakes 2000–2016

Magnitude	2000	2001	2002	2003	2004	2005	2006
8.0+	1	1	0	1	2	1	2
7–7.9	14	15	13	14	14	10	9
6–6.9	146	121	127	140	141	140	142
5–5.9	1344	1224	1201	1203	1515	1693	1712
Estimated Deaths	231	21357	1685	33819	298101	87992	6605

2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
4	0	1	1	1	2	2	1	1	0
14	12	16	23	19	12	17	11	18	16
178	168	144	150	185	108	123	143	127	130
2074	1768	1896	2209	2276	1401	1453	1574	1419	1550
708	88708	1790	226050	21942	689	1572	756	9624	

Tab. 2.3 Worldwide Earthquakes from 2000 to 2016

Source: Please refer to [3]

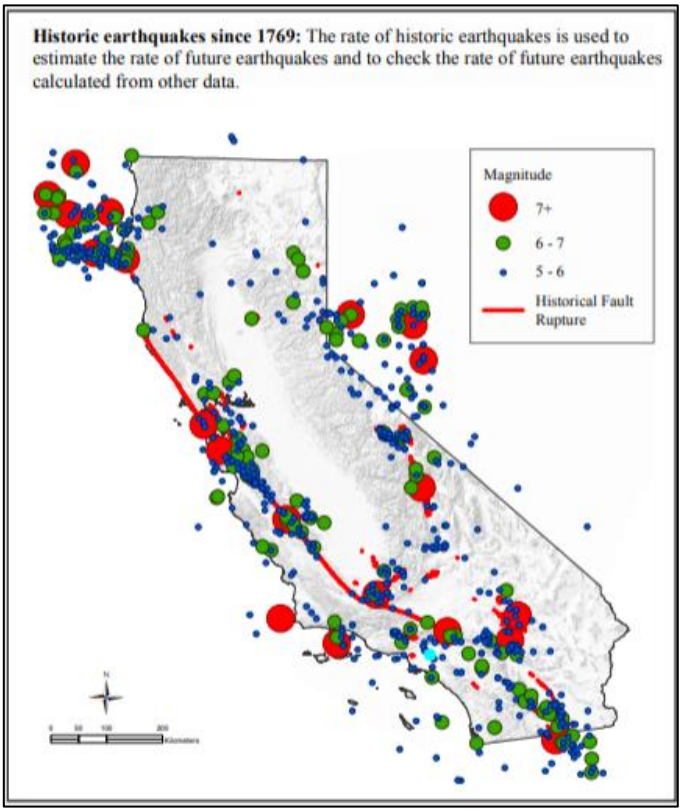


Figure 2.4. Map of historic earthquakes since 1769

Source: Please refer to [3]

Figure 2.5 below shows the map of the epicenter of the 2011 earthquake in Japan. The earthquake had a magnitude of 9.0 and was the largest recorded earthquake in Japan in over 100 years. The earthquake cost more than $300 billion which makes it the most expensive disaster.

Figure 2.5. 2011 Tohoku earthquake epicenter map

Source: Please refer to [20]

c) Tsunamis/Hurricanes/Floods

Subsection C provides data related to tsunamis, hurricanes, and floods, respectively.

Recently, Texas has been hit with record breaking rainfall during the Tropical Storm Imelda which has caused flash flooding. They have seen up to 42 inches of rain within a three-day period. This storm has left at least two people dead and has had rescue crews with boats scrambling to reach stranded drivers and families trapped in their homes during the relentless downpour. There has been a combination of about 1,000 high-water rescues and evacuations to get people to shelter. The classification of the Tropical Storm Imelda falls under the categories of urban search and rescue as well as water search and rescue.

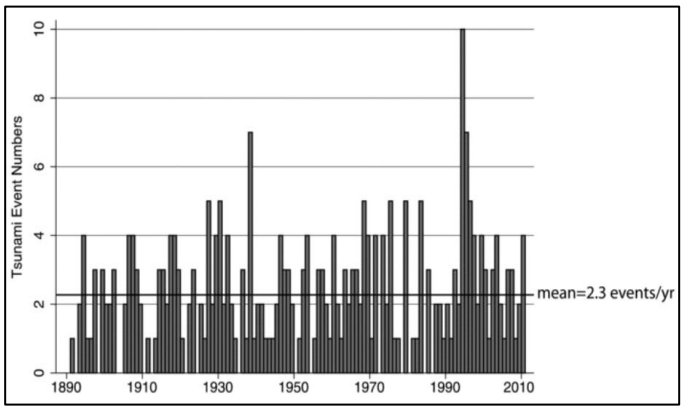


Figure 2.6. Global tsunami events from 1890 to 2010

Source: Please refer to [20]

Year	Total hurricanes (1)	Made landfall as hurricane in the U.S.	Deaths (2)
1999	8	2	60
2000	8	0	4
2001	9	0	42
2002	4	1	5
2003	7	2	24
2004	9	6 (3)	59
2005	15	7	1,518
2006	5	0	0
2007	6	1	1
2008	8	4 (4)	41
2009	3	1 (5)	6
2010	12	0	11
2011	7	1	44
2012	10	1 (6)	83
2013	2	0	1
2014	6	1	2
2015	4	0	3
2016	7	3	36
2017	10	4	147
2018	8	2	48

Tab. 2.4 Number of hurricanes and hurricane related deaths

Source: Please refer to [30]

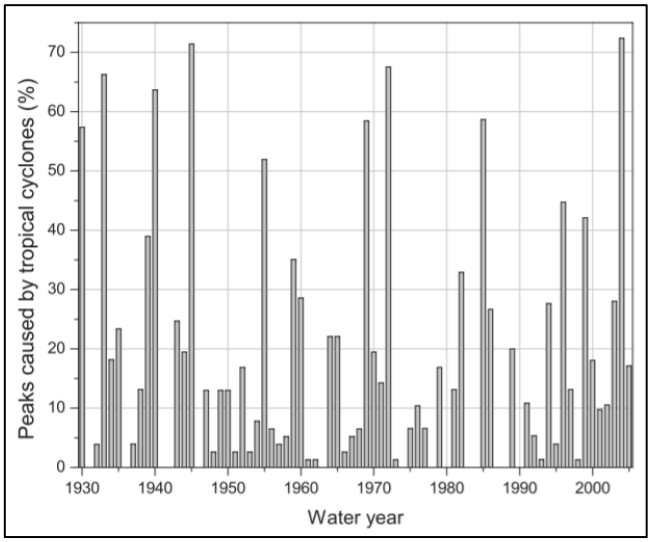


Figure 2.7. Annual percentage of flood peaks in the Easter United States

Source: Please refer to [30]

d) Avalanches

Avalanches are a form of mountain search and rescue. The largest avalanche in history was on May 30th, 1970 off the coast of Peru. An earthquake caused the north slope of Mt. Huascaran to collapse. The avalanche ran for almost 11 miles and estimates that 20,000 people were killed in the event. Statistics show that about 25% of victims were killed by the trauma of the event. Of those that survive the trauma, the first 15 minutes is crucial for survival. The search for victims must start immediately if they are buried. In the present day, ski patrols, helicopters and mountain rescue teams are sent out to search and rescue the buried victims. For avalanche rescue teams, they usually have a first team that can travel light and move quickly to locate and uncover buried victims. After that depending on the severity, they will transport the victim via the ski patrol or by helicopter. Overall the plan for search and rescue teams has evolved tremendously.

3) Search and Rescue Statistics

People may go missing for a variety of reasons, which brings the need of various search and rescue groups. Search and rescue can be broken down into four different categories. The first being ground search and rescue. Ground search and rescue is the search for persons who are lost or in distress on land. The next is Mountain Rescue which is the search and rescue operations specifically in rugged and mountainous terrain. Urban Search and Rescue is the location and rescue of persons from collapsed buildings or other urban and industrial entrapments. Water search and rescue is the capability to coordinate and conduct water search and rescue response efforts for all hazards involving water. All of these include locating, accessing, medically stabilizing, and freeing victims from their specific areas.

4) Current Safety Equipment Needs

With an increasing number of deployment missions by search and rescue personnel, the need of equipment and supplies has also seen a jump in demand. For instance, the year 2017, was the most destructive year in California in terms of property loss/damage, as more than 9,133 fires burned across the state. That year alone saw an unprecedented

number of search and rescue deployments; which utilized so much of California's Fire and Rescue resources, that it required additional assistance from 10 other states. Having said that, this section aims to discuss the most common problems that SAR personnel face when deploying on missions; as well as, the resources needed by workers and victims.

a) Closer Look into The Needs and Risks of California Firefighters

Most of the largest fires in California have taken place within the past 20 years. These frequent occurrences have led to the need of more fire combatant equipment. When dealing with wildfires, firefighters must come equipped with fire resistant pants and shirts, a helmet, eye protection, gloves, leather boots, and fire shelter. On the ground, they must be prepared to contain the spread of the fire by common means such as digging "fire lines", cutting down trees and bushes, or creating "backfires" to deprive the main fire of fuel.

Even with all this equipment aimed to protect and maintain the safety of firefighters, firefighters still run the risk of developing respiratory or carcinogenic issues in the future. The International Agency for Research on Cancer (IARC) classified occupational exposure as a firefighter as possibly carcinogenic to humans. In Fent et al.'s article, "Contamination of Firefighter Personal Protective Equipment and Skin and the Effectiveness of Skin decontamination," he analyzed the health effects associated with contaminated fire equipment utilized by firefighters. Due to the low maintenance of fire suits, as laundering them is only commonly performed once or twice a year, toxins can accumulate on the suits and could transfer onto the skin of firefighters.

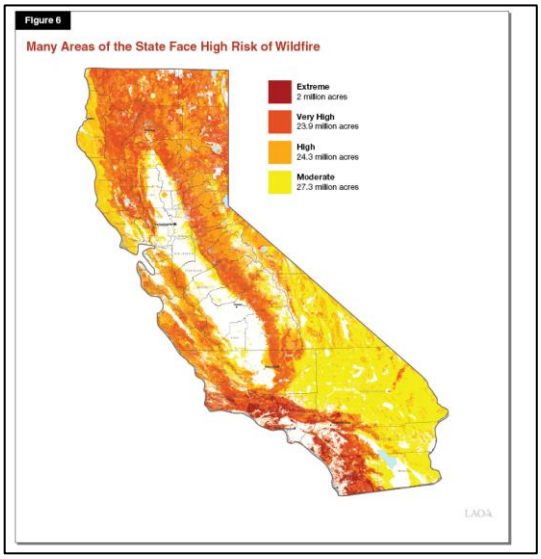


Figure 2.8. Potential Areas with High Risk of Wildfires.

Source: Please refer to [27]

Material	Reference: Gold et al[23]	Treitman et al[24]	Lowry et al[26]
Carbon monoxide	3–1000	15–5000	0–15 000
Hydrogen chloride	18–150	1–200	0–40
Hydrogen cyanide	0·02–5	0·1–5	0–40
Formaldehyde acetaldehyde	NA	NA	1–15
Nitrogen dioxide	0·02–0·89	0·2–10	NA
Carbon dioxide	NA	1000–60 000	NA
Benzene	NA	0·2–150	500–1200*
Particulates	4–750	20–20 000	NA

All concentrations in ppm except particulates which are in mg/m^3.
*Reported as total hydrocarbons.
NA = Not available.

Tab. 2.5 Table demonstrating the most common chemical substances firefighters interact with.

Source: Please refer to [27]

As the upward trend in natural disasters continues, disaster relief individuals will be more susceptible to becoming exposed to hazardous materials. New risk management techniques need to be developed and implemented; in order to maintain the health and safety of SAR workers.

b) Quantifying the Disaster: The Health and Economical Effects Influenced by Natural Disasters in the General Population

Oftentimes, when a natural disaster occurs such as a catastrophic storm, flood, hurricane, wildfire etc. the priority many times is to act upon the current situation. Similarly, this is the same when

researchers analyze the health consequences of natural disasters; as the research focuses primarily on the populations impacted and the immediate aftermath. Prochaska and Peters (2019) argue that there has been very little research done on the health effects of natural disasters in older adults over a long period of time. The article proposes correlations between exposure to these events and the long-term development of cancerous diseases in older adults. As discussed before, the current trends show an increase in occurrences of severe weather events in the United States; with, this increase has also led to an increased exposure to natural disasters by the general population.

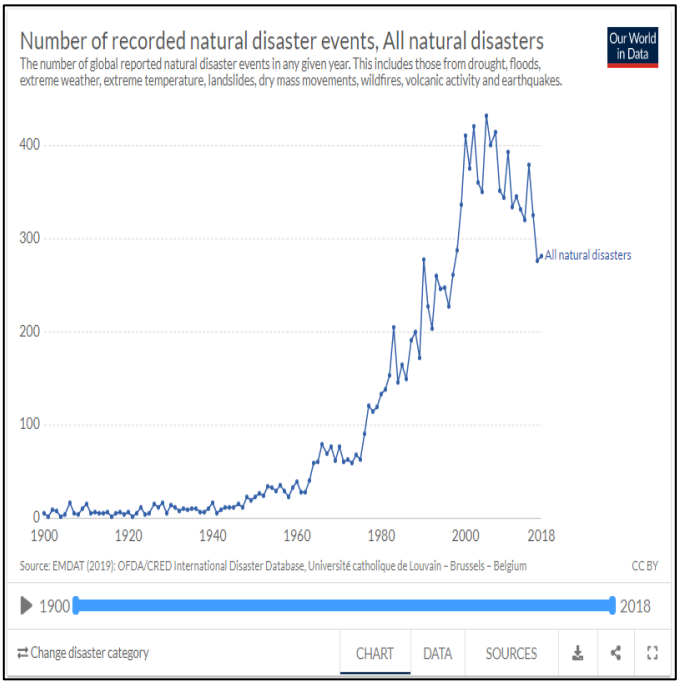


Figure 2.9. Number of Natural Disasters from 1900- 2018

Source: Please refer to [5]

Some of the most common natural disasters that Americans are exposed to are flooding and wildfires. In the case of wildfires, potential risks of cancer can be associated with the exposure of particulate matter in the air. For instance, smoke from wildfires contain high levels of ash and particulate matter (PM2.5) that remains suspended in the air. The specific health concern is the harmful airborne PM associated with forest fires. These smaller airborne particles, specifically PM2.5, have been found to embed and accumulate in the lungs resulting in respiratory diseases such as lung cancer.

When it comes to flooding, there can be a variety of toxins and carcinogens that become present in the water. Storm surges can create situations where water meets hazardous materials; which in turn, may flow into bodies of water used by people.

Economic strain may also be an apparent consequence of a natural disaster. For instance, according to Munich Re, in 2018 the US saw an 82-billion-dollar loss due to natural disaster events that had occurred.

Natural Catastrophe Losses In The United States, 2018

(Based on perils)

As of March, 2019	Number of Events	Fatalities	Estimated Overall Losses (US $bn)	Estimated Insured Losses (US $bn)*
Severe Thunderstorm	56	66	18.8	14.1
Winter Storms & Cold Waves	9	26	4.2	3
Flood, Flash Flood	20	49	2.6	1.2
Earthquake & Geophysical	2		0.5	0.4
Tropical Cyclone	5	107	30.4	15.6
Wildfire, Heat Waves, & Drought (ongoing drought condition without loss estimation for the half year)	16	107	25.4	18
Totals	**108**	**355**	**$81.9**	**$52.3**

Source: © 2019 Munich Re, NatCatSERVICE; Property Claim Services (PCS®)*, a Verisk Analytics® business. As of March 2019.

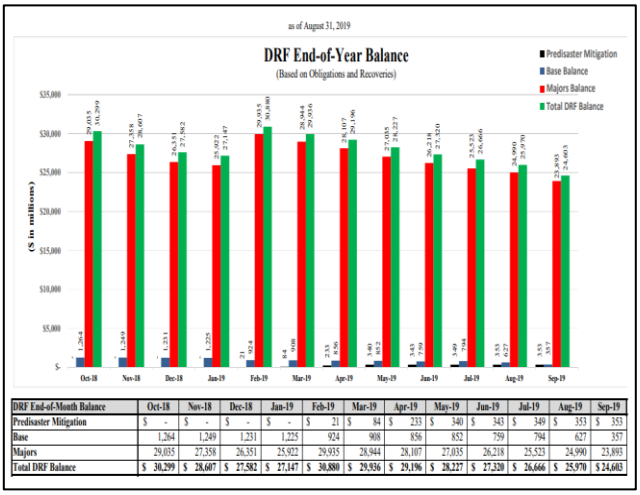
Tab. 2.6 Natural Catastrophe Losses in the US 2018

Source: Please refer to [5]

However, the worst loss for the United States' economy was in 2017 where the US saw a total loss of 307 billion dollars due to 16 events that cost more than $1 billion each (Amadeo, 2019). Economic impacts like these have a tremendous impact on low-income communities as it takes much longer for them to recover; since they have less access to resources.

c) Relief Efforts

During search and rescue operations, the main objective of a team is to rescue the largest number of people in the shortest time, while also mitigating the risk factor. The immediate priority after a natural disaster is providing emergency first aid and medical services to injured persons. In most cases first responders consist of doctors, emergency workers, and police units. Currently, the United States has provided a total of 270 billion dollars in disaster relief efforts as of August 2019.



DRF End-of-Month Balance	Oct-18	Nov-18	Dec-18	Jan-19	Feb-19	Mar-19	Apr-19	May-19	Jun-19	Jul-19	Aug-19	Sep-19
Predisaster Mitigation	$ -	$ -	$ -	$ -	$ 21	$ 84	$ 233	$ 340	$ 343	$ 349	$ 353	$ 353
Base	1,264	1,249	1,231	1,225	924	908	856	852	759	794	627	357
Majors	29,035	27,358	26,351	25,922	29,935	28,944	28,107	27,035	26,218	25,523	24,990	23,893
Total DRF Balance	$ 30,299	$ 28,607	$ 27,582	$ 27,147	$ 30,880	$ 29,936	$ 29,196	$ 28,227	$ 27,320	$ 26,666	$ 25,970	$ 24,603

Figure 2.10. End of Year Cost of Disaster Relief

Source: Please refer to [24]

5) Possible Solutions

Currently, Rescue Teams and Firefighters are put into highly stressful and erratic environments. The proposed, main solution to this problem is to add more robotic technology to help analyze dangerous situations, speed up the searching processes when disasters do happen, react to fires more rapidly and from more dynamic approaches, and give all this information to the teams in the situations so that they can help people in need with better assurance of their own safety. Currently, robotic models are used as live streams that feed information to a remote location out of danger. Eventually, the goal would be to allow autonomous or semi-autonomous robots to replace workers and to allow greater environmental control over any disaster type situation. This requires highly intelligent robotics as well as teams that are trained to operate them.

a) Search and Rescue Robots

Modern search and rescue workers are equipped with a multitude of tools to evaluate and interact with the dangerous situations they are placed in. Robotic technology is a broad topic on its own, but in general most current robots are limited to searching and not so much rescuing. This is caused by robots not having the level of sensitivity and dynamic motion control needed to analyze and transport victims in an efficient and safe manner. Many robots in use will be sent out to locate and send visual/audio feed to an operator that can then send that information to rescue workers. This alone is still a vital piece to rescue situations when victims have a limited time to be rescued. Thermal cameras, range finders, location tracking systems, and 3D mapping systems are all currently being used to varying degrees and being constantly improved.

One example of a current SAR robot is Japan's KOHGA3. This tracked robot is built to analyze damaged buildings whenever Japan is hit with an earthquake. The aim being to protect inspectors and workers as well as provide rescue workers with more eyes and, therefore, greater efficiency.

Figure 2.11. KOHGA3 Ground Robot for Search and Rescue and Vacant Building Inspection

Source: Please refer to [17]

Another example of current robotic technology falls under snake-like robotics. These newer designs are light, thin, and can fit into narrow areas rescue workers cannot readily access. These types of robots, still being developed, could one day help rescue workers locate victims trapped under rubble piles in broken down buildings or miners stuck in caves and mines.

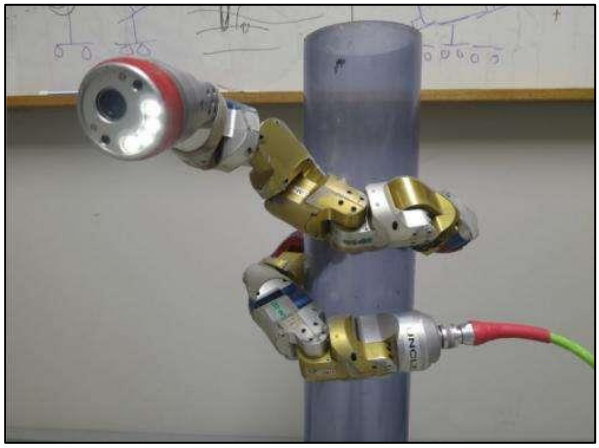


Figure 2.12. Robot Snake, Biorobotics Laboratory Carnegie Mellon University

Source: Please refer to [7]

Likely the most difficult of all the forms of robotics is the walking/running type robots that emulate humans and walking animals. It is very easy to imagine the usefulness of futuristic walking robots. Dogs, cats, horses, and mules have all been worked alongside humans for millennia and having robotic counterparts to match these aspects is not simply science fiction. Robotic quadrupeds have the ability to traverse rough terrain quickly, ever more efficiently, and can relay data just the same as any of the other robots discussed. Quadrupeds are being thoroughly researched and developed; many companies and universities are developing their own walking robots under different specifications which leaves hundreds, possibly thousands, of design ideas across the globe. Boston Dynamic, Honda, MIT, Stanford, and many others have customized prototypes.

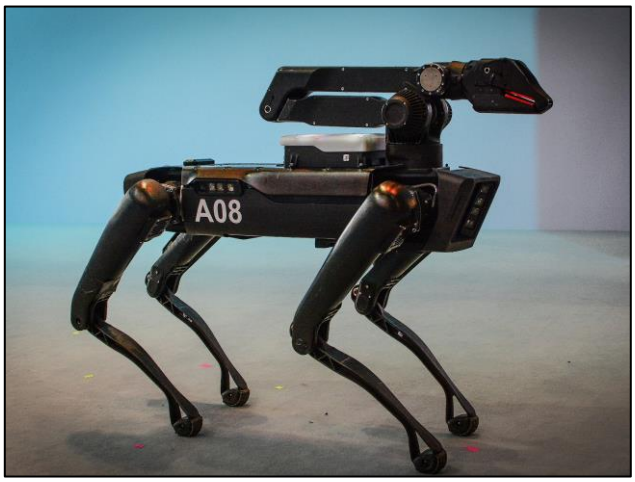


Figure 2.13. Boston Dynamics' SpotMini

Source: Please refer to [19]

Figure 2.14. Honda's Asimo

Source: Please refer to [16]

These robots can be highly versatile in the future working in all areas of society not exclusively search and rescue.

b) Firefighting Robots

Robotic Firefighting Systems typically entail a remote-controlled vehicle with fire suppression technology attached to it, like a water hose. They can travel into areas deemed unsafe for humans such as a collapsing building or closer to a fire that would be too hot for a firefighter. They also have the added benefit of being able to detect voices, body signatures and map out hotspots to avoid walking over collapsible spots.

Figure 2.15. Thermite Fire Fighting Robot

Source: Please refer to [29]

Adding Autonomy to these robots would free up Firefighters to worry about people in rooms and trapped in various locations while the Robots analyzed and hosed down the fires. In the future, firefighting robots could be deployed in wildfire locations with teams of "Hotshots" to automate ditch digging, logging, setting backfires, or carrying additional equipment.

Figure 2.16. Prescribed fire in eastern Washington, United States

Source: Please refer to [27]

c) Drone Technology

Aerial drones can be used as standalone units or in a mini fleet to record terrain and battle fires from above. Currently, drones are being tested to track wildfire movement in large areas that are incredibly difficult to navigate, but, in the future, it is possible to have a team of drones dropping mapping hot zones and movement while other drone drop water and extinguishing material over the flames while firefighters tackle other angles. The same approach could be used in large buildings and skyscrapers to attack fires from the winds using hoses carried by drones.

Figure 2.17. Firefighting Drone

Source: Please refer to [32]

Figure 2.18. Disaster Relief Drone

Source: Please refer to [32]

A group of aerial drones could be set up to map the progress of a fire and track its movement.

6) Societal Problem Conclusion

Natural Disasters are a consistent issue across the globe especially for impoverished communities. This report's aim was to study common forms of natural disasters and their statistical side effects including fatalities, suburban devastation, and economic impact on specified regions. Following this, the report aimed to find solutions to these disaster situations by improving safety standards for firefighters, rescue workers, and disaster relief workers using robotics. The general notion being that creating safer conditions for rescue workers is tantamount to improving rescue effort efficiency as well and increasing relief effort responses. By looking at firefighting robots and drones, adding a small, well-trained team to a wildfire effort could increase overall coordination, tracking, and possibly aid in stopping wildfires and saving thousands of acres of forest. Estimated with increased performance over the next few decades and higher agility robots, one day, perhaps, only a small team of robotic technicians could be putting out entire forest fires with no personnel in direct conflict with the flames.

III. Design Idea

A. Fall Design Idea

1) Brief Description

Fall semester brought a unique opportunity for the team to classify exactly what aspects of the robotics project they wanted to tackle. Topics ranging from

Motor Control to Dynamic Motion to Machine Vision were discussed and ranked to achieve a list of what best suited the needs of the team and project. The search and rescue robot we decided to take further would be designed in two groups: a large model and a small. Each would give the opportunity to tackle different aspects at different paces.

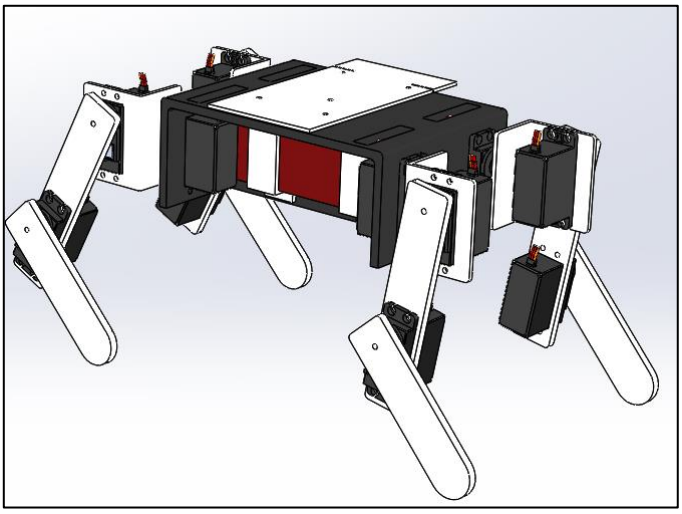


Figure 3.1 Small Scale CAD Model of Robotic Dog

Source: Please refer to [6]

The small model would be built as a bare-bone design of a quadrupedal robot using 12 Servo Motors to achieve the 3-DOF per leg as requested in the final design. By using small Servo Motors, the team could begin to easily control the angles of the limbs and move into the realm of robotic locomotion without waiting for the construction of the much larger design.

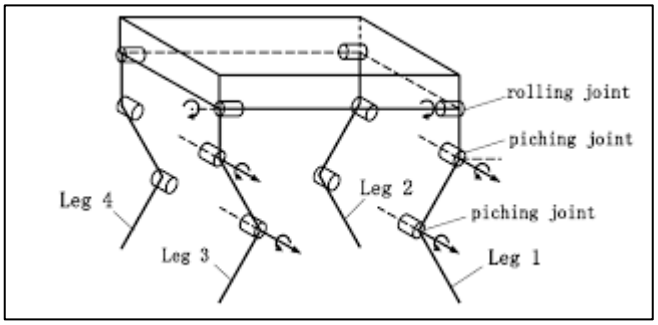


Figure 3.2 Small Model Joint Placement Diagram

Source: Please refer to [21]

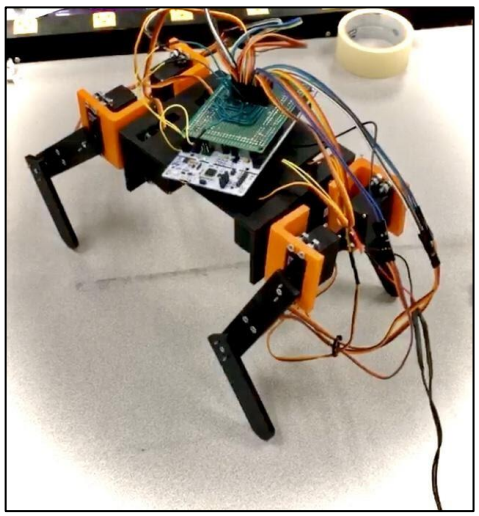


Figure 3.3 Small Model Prototype Design

Source: Please refer to [6]

The larger model would take much longer to construct due to the complexity of its mechanical design. The design would also use a series of more complex electronics to move its limbs. This included: BLDC Motors connected to spinning rods, encoders that measure the RPM of the Motors in both directions, limit switches that prevent motors from crashing passed their limits, and distance sensors that measured the angle of the limbs externally. This was, of course, still a work in progress itself; other solutions included absolute encoders to measure the exact position of the motor along its rod and potentiometers in the joints.

2) Problem Reiteration

The first subjects tackled in the Fall semester were robotic locomotion using Servo Motors and Walking Gaits. The mechanical engineers helped us find several sources of robotic dogs from online 3D models. The next step would be to take these models and create a CAD model that was modified and printed to our specifications. With the model built, we wanted to have the small model built, coded, and walking in a basic motion by the end of the semester.

3) Idea Uniqueness

To be clear, this project was similar in many aspects to other projects built in on other campuses, companies, and hobbyists. It is difficult to say what makes this project specifically unique over other projects. For this team, this project was selected due to its unique goals and challenges that may offer

great opportunities to learn realistic robotics design processes and techniques.

4) Feature Set & Measurable Metrics

The Fall Semesters Feature Set and Measurable Metrics were one of the first challenges faced with this kind of project. The two teams decided that this project would not be completed but be a multi-year project. This meant that we had to first decide what we could realistically accomplish by year's end and what was most important for us to focus on.

At the start of the semester, we originally had planned to split into two pairs. One team would focus on the kinematics, servo motors, and walking motion of the robot; the other team would focus on working in Linux to work on the cameras and computer software for guiding the robot through visual and distance sensors. This slowly fell apart as we found out the level of difficulty that the cameras brought. This eventually was dropped out of the project for the next semester students or club members to take over.

Our feature set was built as a list of what we believed was achievable by the Fall semester's end with caution in mind:

a) Small Model:

- Control the servo motors using embedded C programming language.
- Finishing kinematic model of the servo robot legs.
- Controlling the angles of the joints to a certain degree accuracy.
- Making the Servo Robot walk in a straight line for at least three meters to a certain linear accuracy.

b) Large Model:

- Complete a 3D printed mount that will hold the BLDC Motors and the rotary encoders.
- Connect a motor to a motor controller and access the hardware through software.
- Control the speed and rotary positions of a motor to a certain accuracy.

B. Spring Design Idea

1) Brief Description

With the Fall Semester completed along with most of the original goals and plans, it was time to plan a course of action for the Winter Break and the Spring Semester through to the end of the school year. By the showcase in the Fall, we had a walking prototype with the small dog and were able to control the BLDC Motors to some degree. With this in mind, we gained more realistic insight into what was achievable given the time restrictions and course work from the previous semester. For the Spring Semester we would focus on building a single, large leg instead of the full robot. We would also focus on controlling the small dog using an embedded computer instead of a microcontroller to allow integration with more modern features. These included wireless control over Wi-Fi, Python language, and camera accessibility to expand the robot further throughout the Winter and Spring. For the Spring, we also had all the parts assembled and redesigned for a single leg of the large model. This included two moving joints, the sensors to control them, and the power to maintain them.

2) Problem Reiteration

This semester's goals were the extension of the Fall's semester with further steps in some areas. We wanted to further our ability to make the small prototype walk using the servo motors. We hoped to mount one or two cameras on top of this model and use them as a form of input data to the robot, as well as adding an IMU to make the prototype have balancing feedback features. Basically, the plan for the small dog in the spring semester was to continue working on the walking gaits and locomotion while adding cameras and feedback sensors to make the robot more responsive and advanced. For the large model, the Spring semester goals were reduced from building the entire robot dog to just building a single leg and mounting that to a testing platform. This would allow us to test motor speed, angular control, torque control, power consumption, and mechanical stability. All of which were highly important to be tested before we could go further in the mechanical

design. Some of this did not last to fruition due to campus shut down and lab closure.

3) Feature Set & Measurable Metrics

The features planned for this semester were more realistic than the previous semester. Each team member was to undertake only one main feature and work exclusively on that to ensure everyone was working hard but not overworking.

The Spring Feature Set was more compact and more achievable; however, it has since been reduced further due to campus shutdown:

a) Small Model:

- Controlling the motors and joints using an embedded computer instead of a microcontroller.
- Attaching an IMU to the body platform and implementing feedback into the walking software to increase balance.

b) Large Model:

- Implementing power control system for the leg to operate for a certain length of time and begin testing power consumption.
- Controlling the angles of the joints via the BLDC Motor Controllers in software.

IV. Funding

A. Mechanical Aspects

1) Mechanical Component Breakdown

Tab. 4.1 ME Funding

Bill of Materials	MECH	Total Cost:	$1,767.99
PART DESCRIPTION	QTY	UNIT COST	TOTAL COST
3D printing filament	20	$20.00	$400.00
1/4 Aluminum sheet 36" x 36"	2	$160.00	$320.00
Ballscrew support bearings	8	$22.75	$182.00
250 mm ballscrew, thighs	8	$20.59	$164.72
Linear rail leg supports (rod to be cut in half)	4	$37.35	$149.40
200 mm ballscrew, hip joints	4	$33.99	$135.96
30mm ball bearings	8	$9.00	$72.00
30 mm steel tubes, 2.0mm x 300mm, hip supports	4	$17.28	$69.12
3 x 1 x 36 Aluminum beam, backbone	1	$52.68	$52.68
40 x 40 Aluminum Ext 91"	1	$49.14	$49.14
10mm ball bearings	12	$3.00	$36.00
20x60 Aluminum Ext 800mm	2	$17.30	$34.60
20x40 Aluminum Ext 700mm	2	$14.95	$29.90
Wheel bearings for ball screw support	2	$10.99	$21.98
Assorted hex screws	1	$19.99	$19.99
20 x 20 Aluminum Ext 600 mm	2	6.79	$13.58
3/8 threaded rod 10ft	1	$7.98	$7.98
3/8 washer 25 piece	1	$3.45	$3.45
3/8 hex nut 25piece	1	$2.97	$2.97
T-nuts 1 x M3 1 x M4 lot of 20	2	$1.26	$2.52

Source: Please refer to [6]

B. *Electrical Aspects*

1) *Electrical Component Breakdown*

Tab. 4.2 EEE/CPE Funding

Bill of Materials		Grand Total:	$2,504.70	
PART DESCRIPTION	QTY	UNIT COST	TOTAL COST:	Club or Team Funded
Brushless DC motor	12	$85.85	$1,030.20	Club
O Drive 3.6 (24V version)	6	$129.00	$774.00	Club
NVIDIA Jetson Nano	1	$99.00	$99.00	Club
Time of Flight Sensors	2	$50.00	$100.00	Team
I2C MUX	1	$13.50	$13.50	Team
Rotary Encoder	12	$24.25	$291.00	Club
IMU	1	$5.00	$5.00	Team
Nucleo STM Board	1	$12.00	$12.00	Team
Servo Motors	12	$15.00	$180.00	Team

Source: Please refer to [6]

V. Project Milestones

A. *Cardboard Model*

The first iteration of our robot design was made of cardboard, glue, and tape. This was simply a brainstorming model and it allowed the mechanical engineers on the team to communicate with the electrical and computer engineers on how best to begin this project. The cardboard model was used for roughly the first month of the Fall Semester and ended when the 3D model was designed, and printing began. The greatest lessons learned from this model were: (1) controlling servo motors through microcontrollers, (2) using while loops and time delays properly, and (3) the restrictions on the range of motion on the joints. This was also an opportunity to gain basic understanding on how to properly assign pins and use an ARM M4 microcontroller.

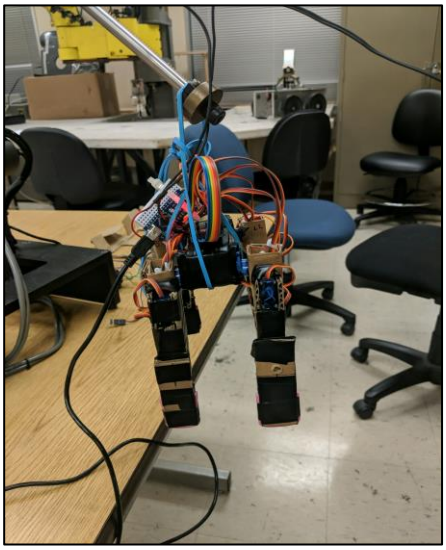


Figure 5.1 Small Cardboard Model Prototype

Source: Please refer to [6]

B. *3D Printed Prototype (Part 1)*

The miniature prototype was the most used model throughout the course of the project. It facilitated the most learning because of its small, carriable size and simplistic design. It allowed us to cut passed the steps of difficult motor control and higher power consumption and practice robotics with less risk.

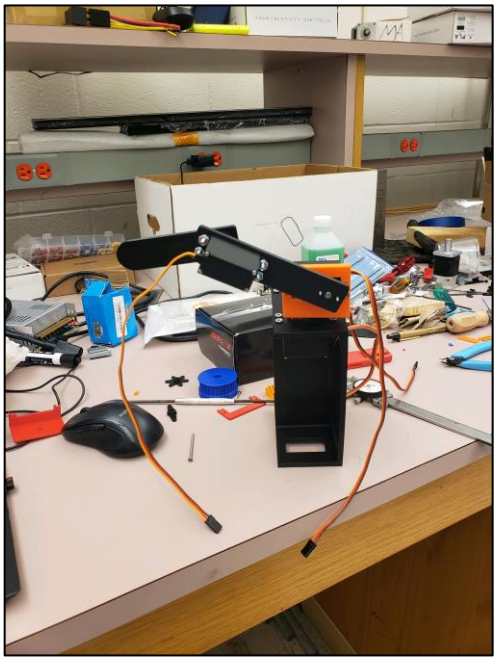


Figure 5.2 Construction of the 3D Printed Small Model

Source: Please refer to [6]

The miniature prototype was designed to operate using 12 high torque servo motors placed directly in each joint, called ‘Direct Drive’ in robotics. This

meant that the motors were in a situation that would allow them to be damaged easily if not used properly, but it allowed them to be as light as possible with the largest range of motion possible. Alternatives to this design we considered including using cabling, springs, struts, or gears coupled between the motors and the joints.

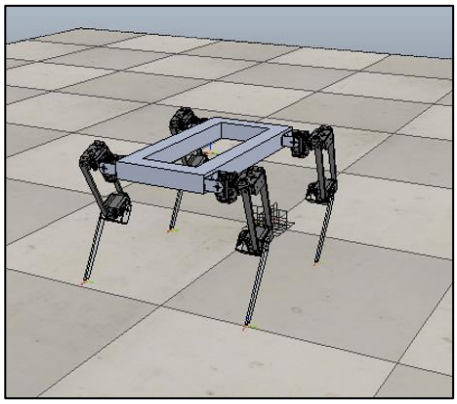


Figure 5.3 Diagram representation of our 12 Servo Dog

Source: Please refer to [6]

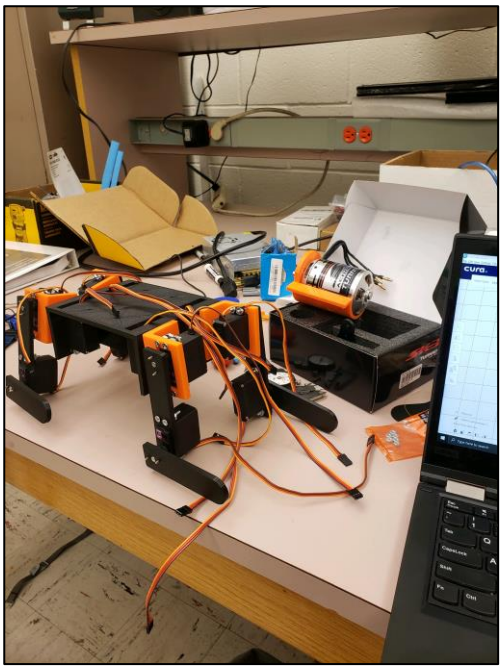


Figure 5.4 3D Printed Small Scale Dog Model

Source: Please refer to [6]

All these options had specific benefits, but the largest common drawback was the added weight, design time, and build time that caused them to each be left out. If a second robot was to be built, these features would be added to increase the robots impact reaction when walking on hard surfaces.

One major aspect of this prototype was the microcontroller itself. The first thing that had to be concurred was controlling a servo motor from the adjustable systick timers and clocks within the ARM Cortex M4 chip itself. The team decided to use a M4 Microcontroller over a hobbyist Arduino board because they have much more capabilities. These include faster clock speeds, lower level port and peripheral control, and lower power consumption. The main drawback of these more advanced chips are the added difficulties caused by the lack of support online.

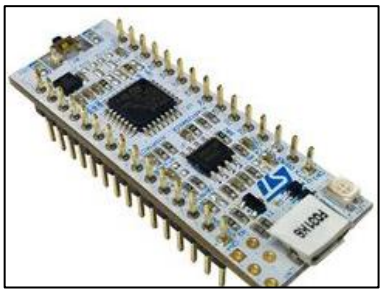


Figure 5.5 STM32 Nucleo Board

Source: Please refer to [4]

Figure 5.6. STM32 Cortex M4 Specifications

Source: Please refer to [4]

Most people who make recreational drones or robots use either Arduino Microcontrollers or Raspberry Pi boards. The ARM Cortex chips, however, are industry standard across a range of products for being cheaper, more compact, and faster. They are used in everything from common household electronics to automobiles to expensive medical supplies. All of which influenced our decision to use the more advanced chips for our project.

The STM32 MCU was used to control our smaller prototype consisting of 12 Servo Motors and a UART Tx/Rx Setup to print out the data onto a terminal in a PC. After Researching and Trial and Error, it was found that the PWM Channels were control on the ARM Cortex through a series of General Purpose & Advanced Timers. Each with a set of 3 to 4 Registers that controlled the PWM channels separately. On the next page is a diagram of the Chip used with all the pins initialized to their correct pins.

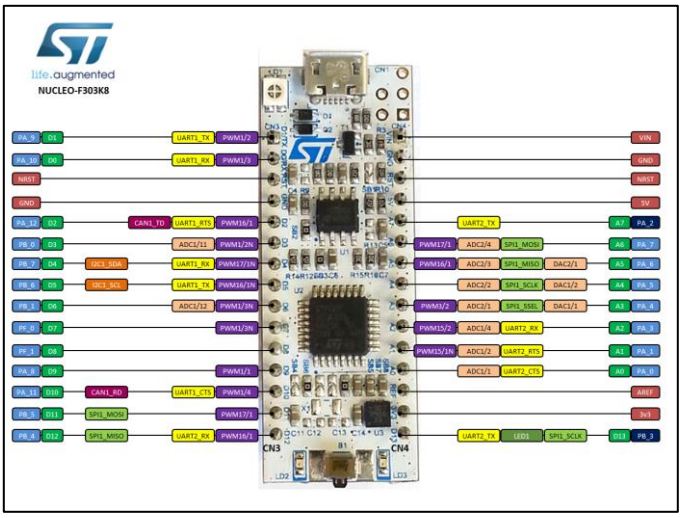


Figure 5.7 STM Nucleo Board Pinout

Source: Please refer to [4]

We also began to consider which programming languages were more useful than others for the tasks we laid out for ourselves to achieve. The main breakdown was a split between timing reliability, lowest level power consumption, simplicity and support, and the ability for rapid prototyping. The primary languages chosen were: (1) Python, (2) C/C++, (3) MATLAB, even (4) Verilog. As absurd as some of these may sound, if the project could be fragmented correctly these could serve individual purposes within the main body of the project to bring together a fully operational robot. The thinking from the start of the semester was that we would make the robot's walking gaits and peripheral control in either Verilog or C/C++. Then we would simultaneously begin working on the camera systems in MATLAB or Python.

C. 3D Printed Prototype (Part 2)

By early to mid-October we had a fully built model with the 12 servo motors properly controlled by the ARM Cortex Microcontroller. The chip selected was mounted onto a PCB Prototyping board which was then mounted onto the center of the robot; for power supply we used a cable power supply that fed the motors 7.5V and fed into an inverter that supplied the chip with 5V. With the clocks and pins set up correctly, we had a fully operational platform with angular control of the 12 joints independently of one another. The next steps to follow would be to use the 12 moveable joints in uniform to create more advanced motion.

At this point, we moved onto the walking patterns and angular control of the robot. This involved more trigonometry and calculus. The robotics community refers to this as Kinematics. Robotic Kinematics applies geometry to the study of the movement of multi-degree of freedom kinematic chains that form the structure of robotic systems. By controlling the links and angles of the body we moved onto walking gait equations. First, we worked at converting the angle control of the joints to position control in 3D space. This meant that we could control the position of the four feet. Next, we worked at creating a circular pattern of moving the foot through a series of positions that would mimic walking. After a lot of trial and error, the walking gaits were perfected to a degree suitable for showcasing.

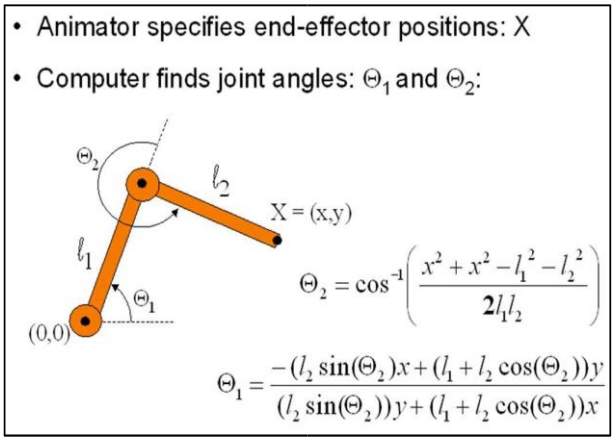


Figure 5.8 2D Inverse Kinematic Nonlinear Equations

Source: Please refer to [28]

Above is a diagram demonstrating the Inverse Kinematics Equations used to get the robot to walk properly. Specifically, the goal of these equations is to convert the X (x, y) position of the foot compared to the hip joint, to two angles for the joints. This would give us the ability to control where the feet are in vector space while letting the function solve for the angles to give the robot joints. Of course, this took considerable time and effort to fully understand, implement, and test, but it all seemed to work perfectly besides the standard deviations in the 3D print model and the looseness of the servo threads.

1) Walking Gaits and Locomotion

Walking the dog was undoubtedly the hardest part of the project last semester. At this point each of the feet could be positions on demand as well as the corresponding angles in each of the joints. Now, torque, speed, and angular momentum all came into play. To make the robot walk, prototypes and products from other schools, studies, or companies had to be studied to gain a basic understanding of what has been attempted in the past. It was found that many of the robots built began with studying dog walking patterns first. By searching the internet, we were able to find many examples of diagrams and videos showing how exactly dogs walk over flat and uneven terrain. These would become the base for our experimentation throughout the process.

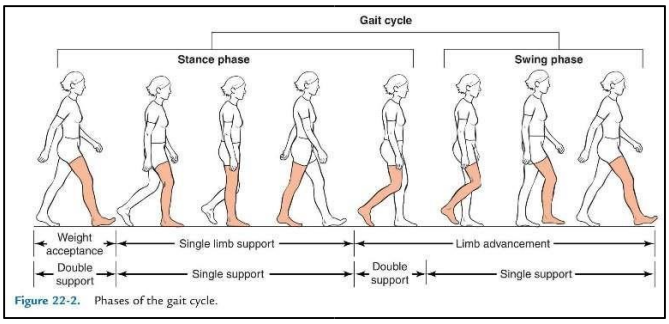


Figure 5.9 Gait Cycle

Source: Please refer to [1]

It was found that using elliptical patterns for the feet, you could effectively cover ground if the robot was well balanced and the diagonal legs moved simultaneously. The diagram above demonstrates human bipedal motion. It is visible that each leg moves opposite to the other. It is also obvious that the legs both move in a cyclic pattern mimicking the ellipse shape when walking straight.

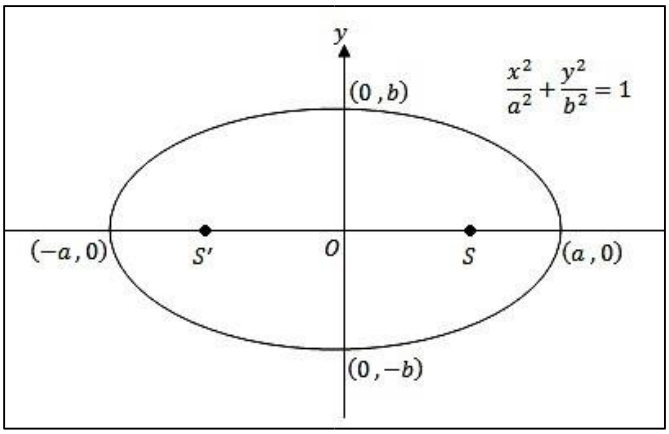


Figure 5.10 End-Effector Ellipse

Source: Please refer to [17]

An ellipse, shown above, was used to mimic the changing position of the foot throughout its walking gait. These equations were drafted into a useable, embedded C program and added as another layer to the previous segments of code. The same variable names: h, k, a, & b; were used to keep a consistent understanding between the code and the pure, mathematical diagrams above them. By use of trigonometry and algebra, we were able to output constant values for the X and Y position along the ellipse to be fed into the servo motor inverse

kinematics equations. Once cyclic motion was achieved in the code, it was time to make the robot walk. To achieve this, the legs vertical to each other had to start with a phase shift of 180 degrees to ensure they were perfectly opposite to one another. Finally, to have the robot continuously walk, the main while loop would continuously call a function to step through the ellipse at a constant rate.

With the robot walking in a basic walking pattern, we had an open-loop system. The system walked almost correctly but would simply continue walking if it was tipped over. The next step was to begin practicing feedback systems and implementing them into the walking equations to make the walking more accurate and responsive. For these steps, each team member tried to implement their version and research the next steps for closed loop responsiveness. These added feedback sensors will be discussed in the sections below.

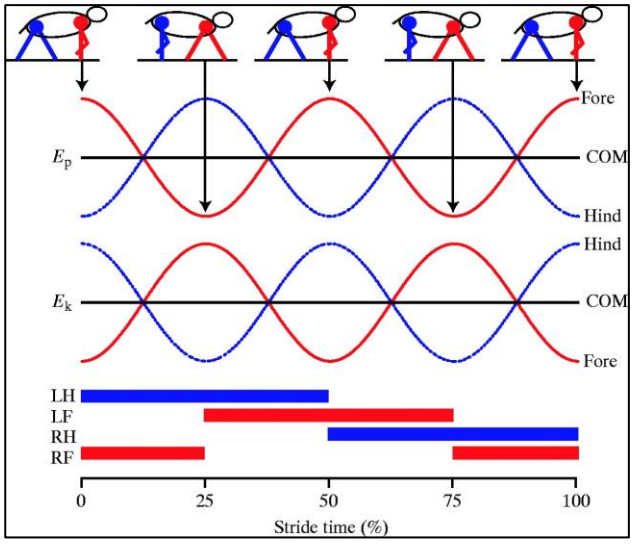


Figure 5.11 Robotic Gaits

Source: Please refer to [17]

D. 3-Phase Brushless DC Motor Control

The motors we used were massive in size and difficult to interface with for first time users. With no background knowledge in the use of 3-phase motors, the team struggled to first learn what size was necessary for the torque we needed. And we were unsure exactly how much power the motors would require or how to measure the consumption to begin the research. Throughout the course of the year, most of these concerns were studied and researched to determine how 3-phase motors operated and why exactly they were more effective for our needs than brushed DC motors, stepper motors, or servo motors.

For us, the lighter weight, higher torque, and greater efficiency really drove home the idea that they best fit our needs. 'Field Oriented Control', a developer term which refers to controlling Brushless DC Motors in practical applications. This meant studying how the three-phase sine waves from the motor controller boards would be impacting the motors and the noise given off by using high power sine waves in small locations. This also meant learning how to control the motors by mounting encoders on each one for position control as well as the voltage, current, and torque requirements for our goals.

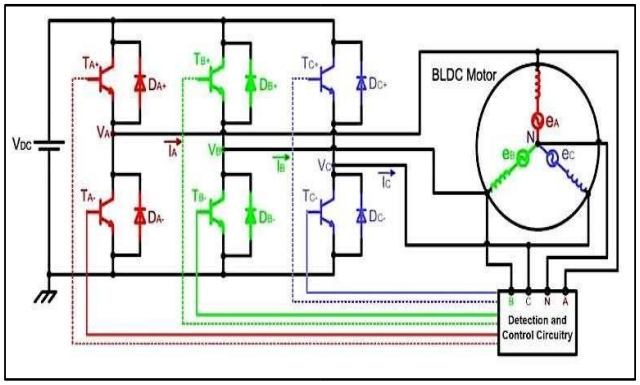


Figure 5.12 3-Phase BLDC Theory

Source: Please refer to [2]

Above is a diagram of the basic BLDC motor control layout. By manipulating the transistors rise and fall time and rapidly switching between the three phases of the BLDC's coils, the motor is given unique advantages. These include greater efficiency than other motors, higher torque output, and, therefore, they can replace bigger motors that output the same results.

The following graph shows the motor spinning at a specified RPM (shown in pink) and Voltage. Gradually, the torque, or strain, on the motor is increased making the motor work harder to maintain its original goal speed. To maintain this set speed, the motor controller will increase the current sent into the motor's three set of coils (shown in red). As these two factors were changing, an efficiency coefficient was being measured to calculate the different between the expected, theoretical, and the actual

output power usage. The trend shows a gradual decline in efficiency as the torque is increased, but it is very shallow and maintains very steady efficiency throughout the course of the test.

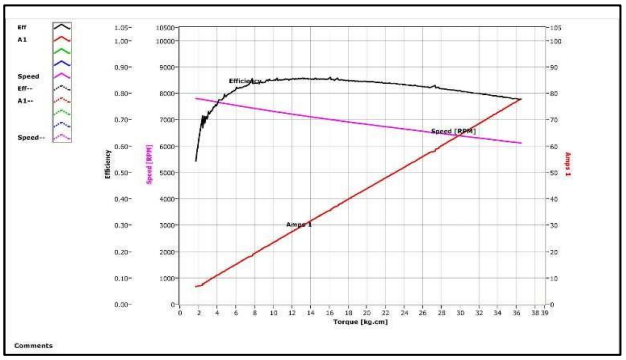


Figure 5.13 Turnigy Aerodrive SK3 Brushless DC Motor Efficiency: Rotational Accuracy Chart

Source: Please refer to [2]

Our test procedures for the Fall semester stipulated that we would aim to control one or two motors using the motor control board selected by the Fall semester senior showcase. This was achieved to some degree because we had a prototype of the motor mount designed and built by the mechanical engineering team. There were issues with the design however; the design did not hold the motor correctly enough to prevent friction between the rotating motor and the surface of the mount. Secondly, the motor encoder was not mounted to an accurate enough degree to prevent error. These two issues combined led to the motor controller constantly receiving corrupt data that led to failure in our testing. Still, at low speed, the motors were controllable in both position control mode and velocity control mode, meaning we were successful in our efforts. It is also important to note that the motors were being controlled on an embedded SOC, NVidia's Jetson Nano, for its small form factor and reasonable strength over other similarly sized products.

E. Battery Management System

After researching the motors and motor control boards, it became necessary to power them properly so they could meet our specific current requirements. To this aim, the team began studying different types of batteries. This included the different battery chemistries available on the market as well as the specifiers that were relevant to our project. Important terms included: Charge / Discharge Rate, Capacity, Voltage, and Cell Count. The two forms of batteries used and tested included Deep Cycle Lead Acid and Lithium Polymer Batteries; both having specific qualities that pertained to our mission.

Deep Cycle Lead Acid Batteries were the first choice because they were the cheapest option, had a very high discharge rate, and came in the correct cell count / voltage required for our mission. The downsides to the Lead Acid Batteries were their large size and weight, their voltage drop-off after 50% usage, and their low recharge cycle rate. This meant that the lead acids were great for testing but for the final product, Lithium Polymer Batteries would be selected.

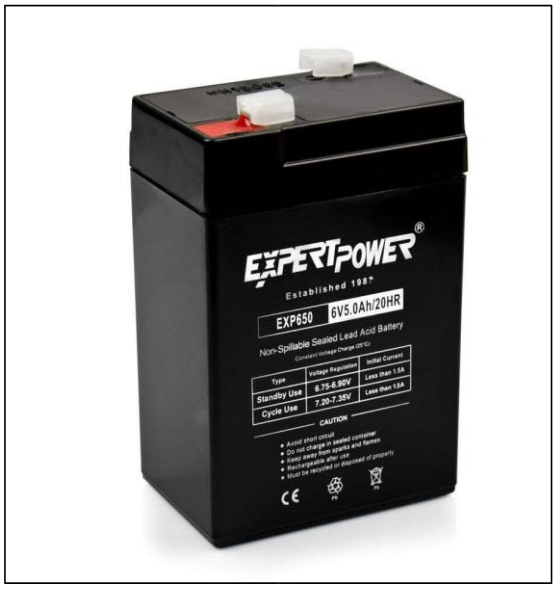


Figure 5.14 6 Volt Lead Acid Battery

Source: Please refer to [31]

Lithium Polymer Batteries are the ideal choice for most RC applications and fit perfectly into our scope. They have the highest energy density of all the forms available on the market currently but come at a much higher sticker price. They are sold in the form of common cell counts. Each cell being built to maintain a voltage of 3.3V. For our robot to operate in the torque and force ranges theorized, we will need to use Lithium Polymer batteries that can output roughly 24 Volts and maintain constant current up to 15-20 Amps. This is no cheap or simple task on its

own. Currently there is a simplistic design being used to maintain the correct voltage.

Figure 5.15 22.2 Volt Li-Po Battery

Source: Please refer to [31]

In the Spring, the team increased their goal to controlling the motors by adding several feedback sensors. These included limit switches to keep the motors from running off past their assigned range of motion and time-of-flight sensors that would allow the motors to be controlled through distance measurements assigned by the user.

F. ODrive Development and Interfacing

The BLDC motors that our design will utilize run on three phases alternating current and it is supplied by the ODrive Motor Controller. The ODrive will provide pulses of current to the motor windings; which in turn, will control the speed and torque of the motor. The ODrive already comes with equipped communication interfaces that can be 6 used to control several aspects of the controller, which ultimately affect the control of the motors. By using a communication interface and some of the pre-written commands, we can effectively control variables, such as: supplied current, supplied voltage, maximum rpm, maximum torque etc. of each motor.

Figure 5.16 Turnigy Aerodrive SK3 Brushless DC Motor

Source: Please refer to [2]

The first step of this task required the team to implement the correct parameters in the ODrive firmware, so that the ODrive could properly control the motors. For instance, if we want our motor to spin at a specific fixed RPM, then we needed to confirm that our motor is spinning at the desired speed. The ODrive may need to know factors such as the type of motor, the number of pole pairs, and the motor KV to accurately calculate the appropriate parameters to drive the motor. Once we have effectively configured the controllers, we can then drive the motor at a specific RPM using some of the ODrive's prewritten commands via the Python Library. Now, by using an external device such as a tachometer, we can determine the RPM accuracy of the ODrive motor controller with that of the tachometer. If there is noticeable difference in the desired RPM with the actual RPM; the ODrive Controller also comes with other parameters, we can adjust to drive the motor at a much more accurate speed. However, since development and support for the ODrive firmware is consistent, the controller should drive the motor reliably to the desired speed.

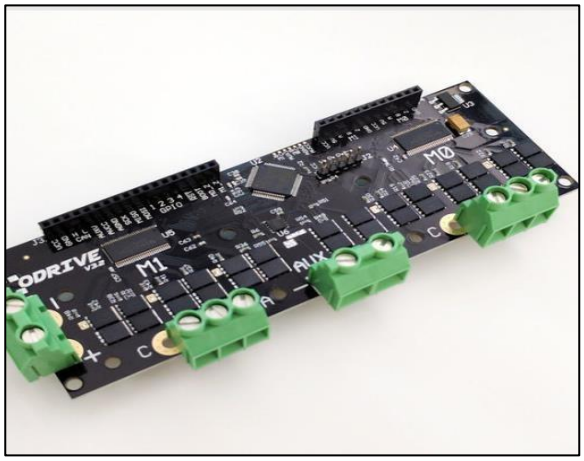


Figure 5.17 The ODrive Motor Controller v.3.5

Source: Please refer to [2]

A. *System Communication Protocols*

The entire control system initially used the SPI Communication protocol, running at over 1 MHz Each ODrive would receive the same readings from the IMUs simultaneously and perform the appropriate calculations based on their local parameters i.e. the positions of each motor they are controlling. In order to accomplish this task, the very first thing that the team did was set up an MCU as a master device that would take readings and transmit them overusing SPI. This is a very trivial step if it is done using STM's supported IDE known as STM32CubeMX, as all we need to do is assign certain pins to the SPI clock, MOSI, MISO, and GND. The CS (Chip Select) pin would not be used in this design, as all ODrives would receive the same IMU readings, and they would perform their own "local" calculations.

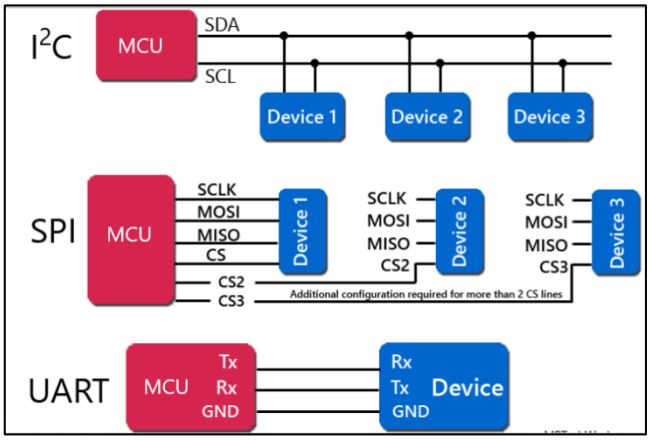


Figure 5.18 Different Communication Protocols

Source: Please refer to [6]

The part that would require more work and time to complete would be modifying the ODrive firmware so that it uses SPI as its default communication protocol; and using its built in FREERTOS middleware, in order to assign tasks and implement a PID controller. By default, the ODrive utilizes UART communication to communicate with external microcontrollers. However, the ODrives have a STM32F4 chip that is used to program each of the pins on its board.

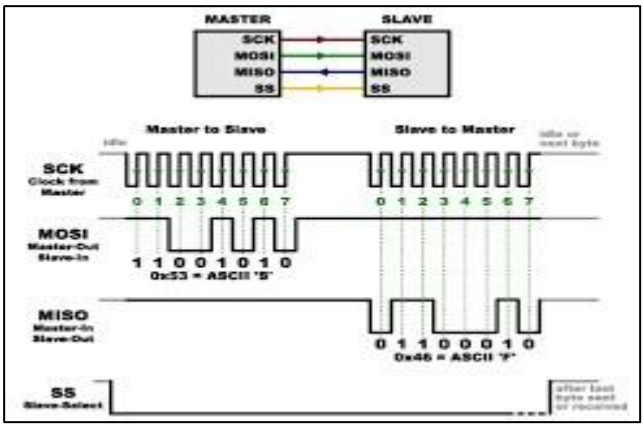


Figure 5.19 SPI Communication Protocol

Source: Please refer to [28]

G. *PID Controller Implementation*

A PID system was implemented into the software of the TOF sensors. It worked by taking the user input 'desired' distance for the TOF sensor to read. That value was compared to a filtered version of what the TOF sensor was reporting to the Jetson Nano.

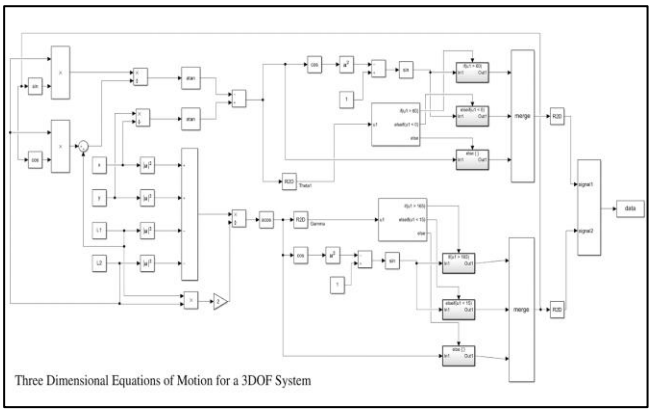


Figure 5.20 MATLAB 3DOF System

Source: Please refer to [28]

The comparison result would trigger a PID function that would make the motor spin in either direction until it met its desired position given by the user. It did so through a series of functions that adjusted the speed and direction of the motors at a decreasing rate as the motor closed in on its desired position. More details about this will be discussed in other sections.

H. 3D Point Cloud Generation

At the beginning of Fall semester, we were set on having the robotic dog operate semi-autonomously while navigating in a 3D point cloud generated map. The 3D point cloud map would be generated by performing Simultaneous Localization and Mapping, otherwise known as SLAM. Due to the unforeseen circumstances of the Coronavirus pandemic we were unable to obtain actual images of our point cloud, but the figure below shows an example of a generated map using the ORB-SLAM2 program.

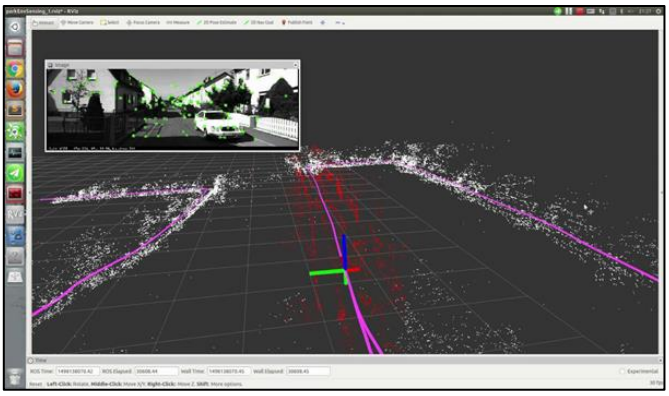


Figure 5.21 ORB-SLAM2 Point Cloud Map Example

Source: Please refer to [18]

Being able to write a program that can perform SLAM is math intensive and could be considered a senior project on its own. To circumvent the difficulties of developing a SLAM solution, we decided to use an existing one in the form of a Robot Operating System (ROS) node. ROS allows robotic developers to use open-source programs to perform robotic specific actions. We initially decided to use a XBOX One Kinect camera because it has a built-in stereo camera, infrared sensor, 50 degree field of view, and can be found for an extremely cheap price.

Unfortunately, after roughly two weeks of implementation and troubleshooting of the SLAM using the Kinect with the Jetson Nano, we found that the ROS distribution installed (Kinetic) on the Nano is not compatible with the Kinect and the Operating System is not compatible with any other ROS distribution. Our solution to this problem was to use a simpler much less powerful Logitech monocular webcam that was compatible with the ROS Kinetic distribution. Using ROS, we were able to combine ORB-SLAM with RVIZ, which is the ROS visualizer and displays the generated map and create a map at a rate of about 19 FPS. By using a monocular camera to create the map, it was not possible to extract the correct distance from the camera to the points in the map.

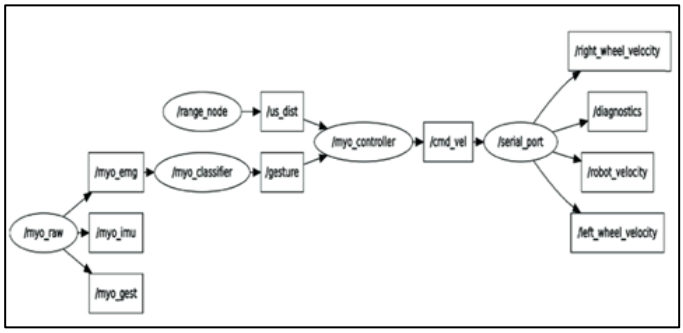


Figure 5.22 ROS Node Graph Example

Source: Please refer to [13]

I. Implementation of Feedback Sensors

1) Inertial Measurement Unit

The IMU (Inertial Measurement Unit) module was a large portion of our robot's feedback system that provided information back to our main computing device, the Jetson Nano. The IMU measures certain data such as linear and angular acceleration in the XYZ axis; some also come equipped with a temperature sensor and magnetometer that can provide other useful data. In our case, we used the MPU 6050 IMU which was able to give us linear and angular acceleration; and the temperature readings.

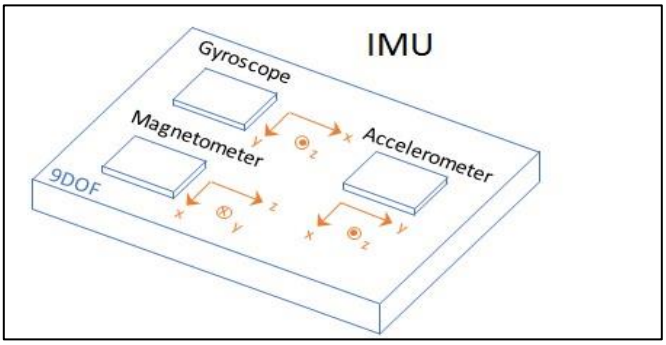


Figure 5.23 9DOF IMU

Source: Please refer to [22]

The development platform that we utilized for the first semester of Senior Design was STMicroelectronics' proprietary and free to use software: STM32CubeMX. The software allowed us to efficiently configure initialization settings for the microcontrollers we were going to use and allowed code generation to be much faster.

The IMU is a very sensitive device and is quite easily susceptible to noise from the environment. In addition, there hadn't been any previously developed library in order to interface the IMU with our specific board. Thus, this was the very first feature that the team worked on the beginning of the semester. Designing a library for hardware interfacing is a very tedious and complicated task. It required quite a bit of time to debug and test the functions of the library were performing the expected operations; nonetheless, once every aspect was tested, we were able to effectively interface the MPU6050 with the STM32F303K8 NUCLEO board. Our designed utilized the DMA, NVIC, and I2C features of the board in order to efficiently capture data from the IMU.

Moreover, once we retrieved the data, another problem arose and that was data calibration. This was another feature that we worked on in the beginning of the semester, and that was to use filtering techniques to clean up our sensor data. By simply applying a complementary filter when writing the interfacing software, we can filter out a great portion of the noise and have reliable readings. The testing procedure for the IMU was exactly as outlined in our Device Test Plan. The team used a plotting library in Python in order to plot the uncalibrated readings of the IMU with that of the calibrated readings; When both data plots were compared, we expected to see a considerable difference in the data plotted as the noise variable has a profound effect on the accelerometer and gyroscope data. The consistency of the calibrated IMU was another requirement in our device test plan. To do this, we utilized two calibrated IMUs and perform some specific-controlled movements on them to collect the data. The movements should be nearly identical so that no factors from this variable should be considered as much as others.

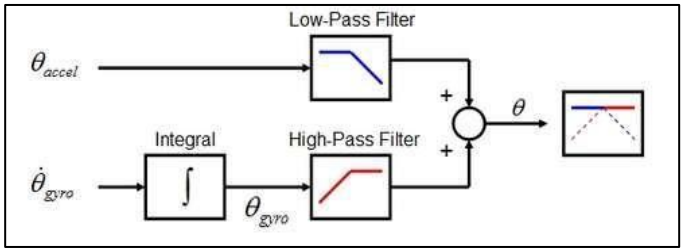


Figure 5.24 IMU Angular Velocity Calculation Process

Source: Please refer to [11]

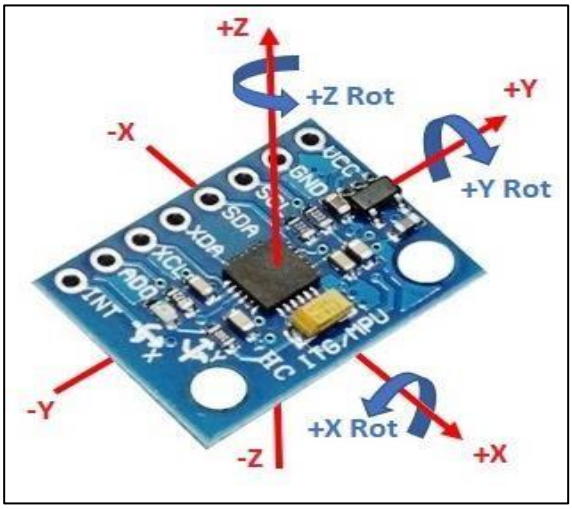


Figure 5.25 A look into the MPU6050 and the different readings it takes at different axes.

Source: Please refer to [4]

When we compared the plotted calibrated data with the uncalibrated plotted data, we saw that there were "spikes" in our acceleration plot when we jolted the IMU back and forth. For the angular position plot there should be a steady drift of the readings over time for the uncalibrated IMU. For the calibrated IMU, the readings for both the position and angular position should fairly remain stable even with sudden

movements. There should be no drift on the angular position plot, the plot should remain relatively stable when the IMU is stationary.

2) Time-of-flight Sensors

ODrive Robotics is still in the process of updating the board's firmware to support absolute encoders which allows us to determine the absolute position of the BLDC. The current firmware, however, only supports rotary incremental encoders which can only provide the angular velocity and index signal which is like the way a bookmark function. Incremental encoders can be used to find the absolute position but was deemed impractical to have our robot perform a homing sequence on every startup. The other reason it was deemed impractical was because the BLDC motors would be driving a pulley which in turn rotates the rod moving the ball screw carriage along the road. The way the actuation works would be prone to slippage, thus making the homing sequence pointless as slipping occurs.

Our solution of obtaining the absolute position of the ball screw carriage was to use a Time-of-Flight sensor that would read off a flat surface attached to the carriage. The figure below shows a diagram of how it would work for our application.

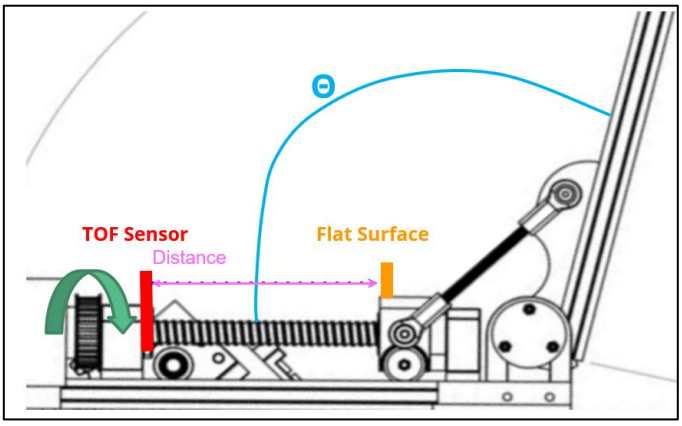


Figure 5.26 TOF Sensor Application

Source: Please refer to [6]

We were using the TOF sensor as feedback for motor control using the ODrive boards for commands. Each motor would need a TOF sensor as feedback so we would need 12 TOF sensors for the 12 BLDC motors, we were only able to get a single leg working with 2 degrees-of-freedom, so we only implemented 2 sensors. The TOF sensor we used to be the VL6180X, capable of reading the distance between 5mm to 200mm, developed by STMicroelectronics and the breakout board was supplied by Adafruit. The sensor interfaced via the I2C communication protocol and did have a fixed address of 0x29 with no possibility of changing it in Python without it losing its new address when turned off; in other words, the sensor did not have any non-volatile memory.

For us to read from multiple TOF sensors we needed an I2C multiplexor. We used the TCA9548A 8-to-1 I2C multiplexor which had a Python library supplied by Adafruit decreasing the development time. After wiring up the Jetson Nano I2C bus to the I2C Mux, where the Mux was wired up to the appropriate pins on the TOF sensor, we were able to read from the sensors at a rate of about 400kHz.

3) Rotary Encoders

The incremental rotary encoders are used to determine the angular velocity of the BLDC motors and ODrive Robotics built-in a safety feature in their boards preventing the motors from reaching a certain speed without encoders mounted and connected. Implementing the rotary encoders was plug-and-play due to the simplicity of the ODrive tool. We just needed to set up the encoder specifications into the boards such as the CPR (counts per revolution), encoder type, and index signal. After setting up the encoders we were able to have full control of the BLDC motors.

4) Long Range Communication/Camera

In order to make the robotic quadruped operate semi-autonomously, the team also proposed adding long range functionality to the project. The idea was to have a first-person view (FPV) camera on the front of the robotic quadruped in order to have an image transmitted back to us to aid in search and rescue operations. This is normally a wide-angle camera to give you a full view of the data that is in front and near the robotic quadruped while under operation. The camera will then need to be connected to a transmitter to send the data images to a computer or screen fast enough to get accurate representation of the surrounding area while under operation.

To properly test this, first we needed to setup and connect the FPV camera to a board in order to make

sure we can get images and data from the camera at real time. Once we have the camera working and transmitting data to our computer, we then needed to setup the transmitter and receiver in order to be able to move the camera freely and still be able to transmit data to our computer. After we get these both setup and working, the next part would be to test them. The things we tested were the resolution of the camera to make sure it meets the specifications, the response time of the camera and how fast the data is sent back to the computer, and the distance of how far the signal we will be able to get before the signal is cutout or the response time is unusable. In order to test the resolution of the camera, we can see the resolution on our computer as well as if the images sent from the camera are pixelated. The next thing to test is the response time in which we can see in code how fast the data is sent back to us as well as moving objects in front of the camera and seeing if there is a delay in the image we see on our screen. The last thing to test is the distance of the signal of the transmitter/receiver, and we would test this by going outside and measuring the distance of how far we can move before the images become unusable due to the signal being weak.

J. Full-Scale Model Leg Prototype

To reach the point of leg control, the TOF sensors and encoders needed to be implemented beforehand. After we set up the ODrive boards and confirmed that we were able to control the leg without feedback, we wrote a Python module that utilized all of the necessary libraries such as the ODrive tool, sensor implementation, and other libraries we wrote to compute the kinematics.

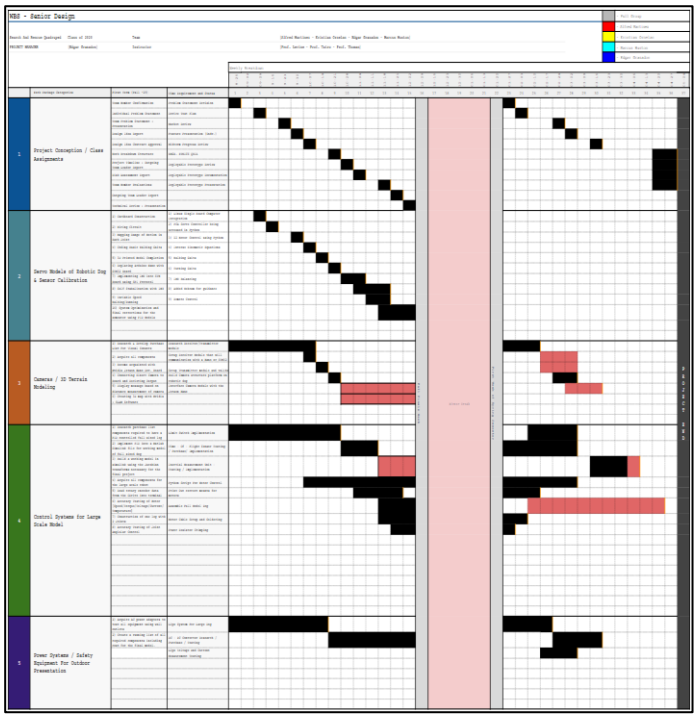
VI. Work Breakdown Structure

A. Brief Description

The project was coordinated with a work breakdown structure, WBS, as specified by the guidelines of the course. This was the team's first time using such tool as a time management device and it proved to be greatly beneficial on multiple occasions. For one, the WBS served to help us decide what bits could be fractioned up and placed on a timeline before beginning research and development on our project. Since the WBS diagram, was required to be developed early in the course, it forced each of us to take a realistic look at what was achievable, how long each core component of the project should take, and how to partition the workload into sub-categories. By doing such partitioning, we were able to assign tasks to members and track their progress to what we had originally estimated.

Some of the original estimations were off, however. Timing on a project like this can be tricky to calculate accurately. But having these charts allowed us to adjust timing and stay on track without developing 'tunnel vision'.

B. Work Breakdown Structure Diagram



Source: Please refer to [6]

C. List of Tasks & Hours Invested

Feature	Team Member	Hours Invested
Full-scale Model Leg Electronics	Edgar Granados	30 Hours
	Marcus Huston	30 Hours
	Kristian Ornelas	30 Hours
	Alfred Martinez	30 Hours
Full-scale Model Kinematics	Edgar Granados	30 Hours
	Marcus Huston	30 Hours
	Kristian Ornelas	60 Hours
	Alfred Martinez	30 Hours
	Edgar Granados	30 Hours
	Marcus Huston	30 Hours

Small-scale Model Electronics	Kristian Ornelas	20 Hours
	Alfred Martinez	30 Hours
Small-scale Model Kinematics	Edgar Granados	30 Hours
	Marcus Huston	30 Hours
	Kristian Ornelas	45 Hours
	Alfred Martinez	30 Hours
Camera System / Software	Edgar Granados	30 Hours
	Marcus Huston	30 Hours
	Kristian Ornelas	80 Hours
	Alfred Martinez	30 Hours
Power System	Alfred Martinez	20 Hours

Source: Please refer to [6]

VII. Risk Assessment

Every engineering project will have risks involved but a good engineer will try to seek out those risks beforehand and take the necessary precautions to avoid said risks. This section will examine the risks we identified and what steps were taken to mitigate them.

A. Lack of Technical Experience

1) Risk Description

The technical skill and experience required will vary project to project, but this robotic quadruped did need a fair amount of both skill and experience. The risk related to our lack of technical skill and experience is that we would not be able to move the project forward and plateau our development.

2) Mitigation Techniques

The steps we took to help mitigate this risk included the following: consulting with our lab instructors, research, and time. With our limited knowledge, a great resource we took advantage was consulting with Professor Levine and Professor Thomas for advice on how to proceed. Their knowledge and experience were invaluable to our overall project progress.

We were extremely out of our comfort zone for this project and needed to do the proper research on robotics, various types of actuators, batteries, etc. By doing extensive research we were able to arm ourselves with knowledge to better prepare ourselves when we encountered unforeseen problems during development.

The last way we fought against this risk was by putting the time in to learn what we needed and spend the necessary amount of man hours to complete the tasks with the level of difficulty taken into consideration.

B. Embedded System Development Time

1) Risk Description

The amount of time to develop an embedded system can be time intensive and difficult when compared to a high programming/scripting language. The plan was to program the ODrive boards in C but with the combination of our other courses, the workload in senior design, and the time necessary to implement robotic applications in an embedded system.

2) Mitigation Techniques

After we realized just how long the development time of the robotic dog in embedded C would be, we decided to control the large-scale leg using a combination of the ODrive tool library and Python. Due to the large number of mathematical libraries for performing matrix operations, ODrive Robotics also converted the ODrive tool into a Python library capable of issuing commands to the motors using a Python script.

C. Scheduling Conflicts with Mechanical Engineering Team & Schoolwork

1) Risk Description

In the demanding major of any concentration in engineering making sure everyone on the team can meet to discuss project topics or to work on the project tasks, this becomes even more difficult as our team consists of four EEE/CPE members and four ME members. The specific risk involved is that if the team members cannot meetup based on their availability, then the project cannot move forward.

2) Mitigation Techniques

The primary way we mitigate this risk was to maintain constant communication for updates and use a user-friendly phone application capable of communicating over most devices. Luckily everyone

on our team had a compatible device so contact was able to remain.

D. COVID-19 / Campus Closure

1) Risk Description

This particular risk is one that most of the world was not prepared for. The risk in this case was that the CSUS campus would need to be shut down in order to minimize the spread of the virus and advocate social distancing. If the campus would be closed then we would no longer have access to the labs in Riverside Hall despite our FOB keys, thus, continuing our work would be a significant challenge with no workspace.

2) Mitigation Techniques

The mitigation we took for this risk was to continuously work on our written assignments from home and communicate over a virtual meeting application. Fortunately, the project was expected to be finished just a few days before the campus was officially closed and did not require us to be on campus to work on the project.

VIII. Design Philosophy

The Societal Problem that we wanted to address is that disasters happen every year and are uncontrollable which increases the need for search and rescue workers which means the danger they put themselves into rises as well. We aim to create a design to help lower the chances of injury and keep more people safe by creating a semi-autonomous robotic quadruped. This section goes through and will cover how we approached solving our societal problem and how we decided a robotic quadruped would be a solution.

A. Brainstorming Philosophy

In order to come up with a solution to our societal problem, we had to understand the societal problem in a greater depth as well as what products are out there that can be a solution to the problem. The first thing we had to look at was the main type of disasters that happen every year and which type of disasters that we wanted to focus on. Since disasters are such a complex topic, we decided to choose a narrower topic which is to aid the disaster workers with a robot designed to help with search and rescue. We had to analyze the market and decide between a wheeled robot or a legged robot. A wheeled robot is easier to design and operate which makes it more stable, but the downfall of a wheeled device is that it cannot go over complex material. This made us choose a legged robot since we could program it to be able to walk over obstacles as well as stay balanced when moving. After we decided we wanted to make a robotic quadruped, we talked to the mechanical engineering team into designing what materials would be best used as well as how the legs would move.

B. Prototyping Philosophy

Once we decided on a robotic quadruped, we wanted to start prototyping in order to start working on the code since we knew it would be a long and tedious process. This gave the mechanical engineering time to come up with an idea of how the final robotic quadruped will look like as well as how it will move. Our first idea was to make a quick prototype that would be cheap, easy to make, and fast to build. The first prototype we created was a cardboard model with servos and an Arduino microcontroller. This made it cheap to make as well as fast and easily programmable. Creating the first prototype gave us time to start learning about the walking gaits as well as the kinematics it would need in order to walk without falling over. Once we got some walking gates programmed and tested, we ran into issues once we started to speed up the walking gaits. The issue we ran into was the grip at the end of the cardboard robotic quadruped legs were hard to get grip since most tables were slippery as well as being made out of cardboard. We tested out gluing rubber bands to the feet to produce more grip against the slippery surface but only helped so much. After we came across this issue and attempted to fix it, we decided it would be best to make a bigger more complex robotic quadruped that we could design to walk better. The second prototype we designed was a little bigger 3d printed robotic quadruped with high torque servos as well as an STM32 board to control the robotic quadruped. This made it so our robotic quadruped had even legs, a perfect body, as well as stronger servos that could support the body while under pressure walking. With the new prototype built, we were able to program and test different walking gaits as well as faster moving gaits and even bouncing in place. Once we reached this point in the process, the mechanical engineering team came up

with a complete design for our main robotic quadruped. The only issue was that we had to wait for parts to come in as well as machining for some of those parts that a professional would have to take care of. Instead of waiting for a long time and doing nothing, we came up with ideas that we could work on as features that we could add to the main robotic quadruped. We also worked on building one leg that we can start coding on to give us an idea of how big and how to control the brushless motors. The leg consists of O-drives, brushless DC motors, encoders, and metal rods. In order to control the legs, the TOF sensors and encoders needed to be implemented beforehand. After we set up the ODrive boards and confirmed that we were able to control the leg without feedback, we wrote a Python module that utilized all of the necessary libraries such as the ODrive tool, sensor implementation, and other libraries we wrote to compute the kinematics. This gave us a fully built leg with control of it as well as a small prototype with walking gaits and kinematic equations to step distances. This gave us a big idea of how to program the full built robotic quadruped as well as basic control of the legs. We would just need to combine everything we had into one big robotic quadruped. This leads to our final deployable prototype.

IX. Deployable Prototype Status

A. Brief Description

In the end of the Spring semester, our final deployable prototype included a small-scale model system capable of maintaining its walking gait and stability by using an IMU module as the feedback device. In addition, the team was also able to design the kinematic and control program for a large-scale model leg. The large-scale model leg could move to a chosen position given by the user to a very accurate degree. It also utilized feedback modules such as TOF sensors, Rotary encoders, and a motor driver to appropriately control the speed and required torque of each individual motor. The small-scale robotic dog had the potential of being able to accurately maintain its stability and walking algorithm even if changes in its environment were introduced. Although the team was not able to fit the IMU sensor module onto the small dog; in theory, the control algorithm that was written should have gotten an initial reading of the dog, set the initial reading as the origin or "balanced" state, and as the dog transitioned into its walking gait, the algorithm would begin to drive the appropriate controls to the servos; in order to, maintain the system's stability. For the larger scale model dog, because the team was able to calculate and produce the kinematic equations for this system as well, the same control algorithm could also have been used. We had proposed using the Jetson Nano Development Board as the main "brain" of the system in which it would communicate with the sensor modules (IMU and TOF Sensors), and with the ODrive Motor Controllers to control the motors. The only main difference between the small-scale model and larger scale model would have been the TOF sensors, but nonetheless, we could have incorporated this difference within the ODrive Feedback System and modeled this as a servo in the small-scale model. Unfortunately, because of the unexpected events that had occurred due to the COVID-19 pandemic, several of our resources and facility equipment were restricted to us. And with the lockdown order and social distancing regulations put in place by our local government, our development time was tremendously impacted. Even though the team was unable to assemble the full-scale model dog, all the components needed to have a working robotic system were designed and tested. Thus, in theory, our full-sized model would have been able to stand and maintain its stability if all these components had been integrated together and tested. The next sections will discuss the testing results and parameters that were designed by our team; and how they related to our measurable metrics and final feature set.

B. Prototype Performance

This section will present a detailed overview and outline of the performance of our deployable prototype and give specific information of each of the components used. It will give a general explanation and background information of the component module, the manufacturer's specification, testing procedure, and testing results. Lastly, the overall performance of the final prototype will be examined and how applicable it would be in real world situations.

1) Inertial Measurement Unit Performance

This section highlights the listed test procedures for the Inertial Measurement Unit. Covering the sensor's purpose, listed qualities from the manufacturer, and, of course, the tests conducted on the device to determine their accuracy.

The purpose of the device testing procedure for the Inertial Measurement Unit is to determine whether the modules can accurately take readings from each of the sensors. Hardware defections and noise sensitivity are some of the most common errors that arise from sensor data. In general, the most common way to correct the data is by utilizing software filtering algorithms such as low pass, complementary, the Kalman filter. Once the data has been corrected, the measurements will then be used in the control algorithm that will adjust the positions of the servos, so that the entire robot system maintains its stability. To do this, during the initialization of the robot, it will be placed in a static starting position. Then the system will capture the readings from the IMU and set this orientation as the origin point. Now, as the robot begins to transition to its walking gait; anytime there is a fair enough deviation from the origin point, the algorithm will adjust the position of the servos so that the system tries to maintain stability based on initial readings of the IMU.

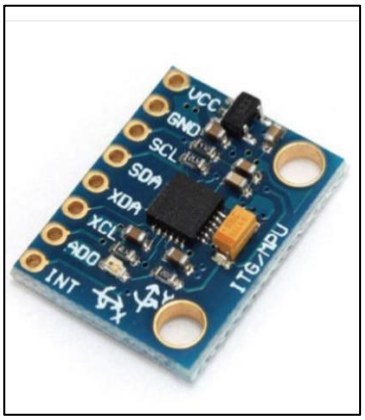
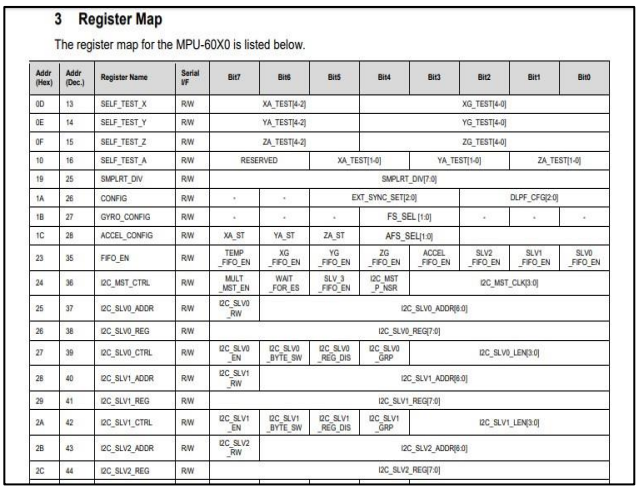


Figure 9.1 MPU6050 IMU

Source: Please refer to [4]

The Inertial Measurement Unit used in this design is an MPU6050 IMU module in which it comes equipped with a 3-axis accelerometer, temperature sensor, and 3-axis gyroscope. The accelerometer and gyroscope can take measurements in the x, y, and z axis of an object. Using these readings, we can then calculate the roll, pitch and yaw of an object. According to the manufacture specifications, the modules support I2C communication protocol with a maximum clock frequency of 400 KHz. The accelerometer can output readings at 1,000 Hz with a maximum output value of ±16 g. For the gyroscope, it can represent an angular velocity value of up to 2,000 °/s at a maximum data rate of 8000 Hz. The power requirements for the breakout board that contains these sensors can operate at voltage levels from 2.375-3.46V respectively. Lastly, both modules contain a series of 16-bit ADCs that are used to convert the analog signals from the sensors and write it to the FIFO Buffer which can later be read through the IMUs registers. Further feature customization can be performed by writing to specific registers in the IMU module.



3 Register Map

The register map for the MPU-60X0 is listed below.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
23	35	FIFO_EN	R/W	TEMP _FIFO_EN	XG _FIFO_EN	YG _FIFO_EN	ZG _FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0 _RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0 _EN	I2C_SLV0 _BYTE_SW	I2C_SLV0 _REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LEN[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1 _RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1 _EN	I2C_SLV1 _BYTE_SW	I2C_SLV1 _REG_DIS	I2C_SLV1 _GRP	I2C_SLV1_LEN[3:0]			
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2 _RW	I2C_SLV2_ADDR[6:0]						
2C	44	I2C_SLV2_REG	R/W	I2C_SLV2_REG[7:0]							

Figure 9.2 MPU6050 Register Map

Source: Please refer to [4]

Two steps were done to demonstrate the noise sensitivity correction as well as the hardware functionality. As previously mentioned, one of the most common issues with IMU readings is susceptibility to noise. Luckily by applying simple filtering algorithms we can compensate for this issue. In our design we utilized a complementary filter to correct the data that was being received from the IMU. The characteristics of the accelerometer is that it tends to suffer from sudden jitters and spikes in its readings; nonetheless overtime the readings are fairly accurate. For the gyroscope the readings are very precise, however, the gyroscope suffers from drift due to integrating over time. When we apply a complementary filter, we effectively take the

advantages of both accelerometer and gyroscope, fuse the data, and get a very accurate estimation of what the reading should be. The filtered data does not suffer from these random spikes in readings nor does the data drift as time passes. The following plotted data demonstrates the calculated Roll Angle of the IMU using the accelerometer, gyroscope, and complementary filter. The test utilized two IMUs in order to determine whether there were any hardware issues with the IMU. The data demonstrates that both IMUs had very similar readings when plotting the roll angle; thus, we can conclude that the data given off the IMUs was accurate and there were no hardware problems.

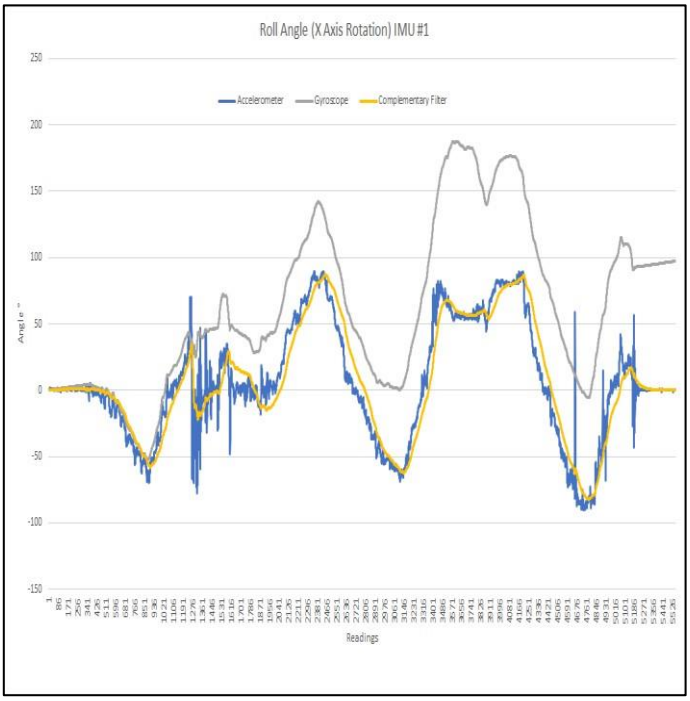


Figure 9.3 IMU #1 X-Axis Rotation Filtered versus Unfiltered

Source: Please refer to [6]

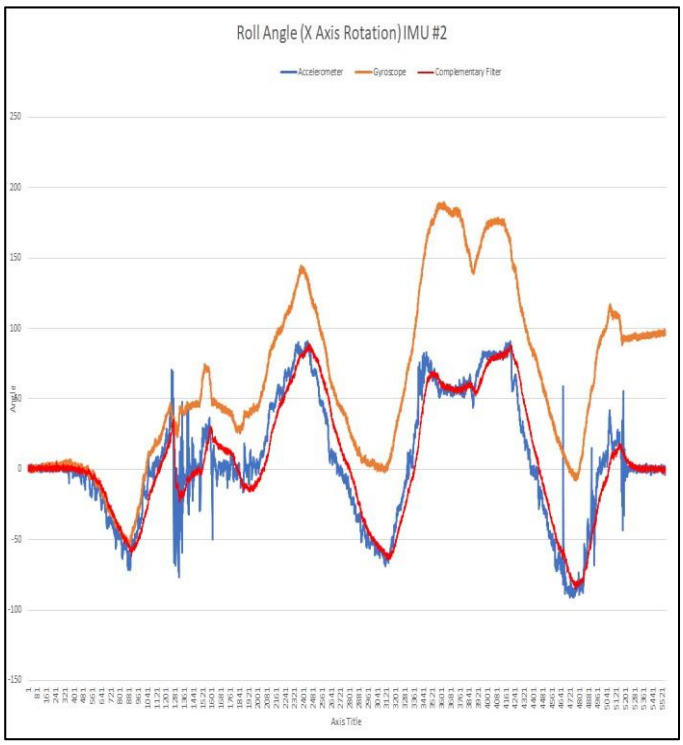


Figure 9.4 . IMU #2 X-Axis Rotation Filtered versus Unfiltered

Source: Please refer to [6]

2) Brushless DC Motor Performance

The BLDC motors were a primary component of the device. They were to be attached directly to ball screws to create custom linear actuators. This design went through several iterations, but the most current iteration includes the motor being attached via metal spring coupler to the rod directly.

Since the design was immature and the device specifications are not entirely known, motors were selected that would overcompensate for any weight requirements thrown at it. This meant that the test motors came at a higher cost, higher current consumption, and higher weight, but this allowed us to find which speeds and torques worked best for us situation. The motors were selected after researching hobby RC maker projects and other engineering projects. By ordering these types of motors, we were able to see them in use in many examples online to see expected results. Industrial level motors were out of the budget supplied to us as well as the typically too bulky. BLDC Motors have several benefits over regular Brushed DC Motors. They are typically 85-90% efficient, whereas brushed motors are typically 60-70% efficient. They are lighter for equivalent torque, have a longer lifespan, and are quieter due to

less contact between parts. The specifications of BLDC Motors can be hard to comprehend at first.

The most significant specification for our purposes was the KV Rating – the measure of revolutions per minute given 1 volt and no load. The RPM can therefore be estimated by multiplying the voltage given to the motor by its KV rating. In our case, 24 volts is being constantly supplied to the motors. Our specific motor has a kV rating of 192.

Max Speed = (192kV) * (24 volts) = 4608 RPM

The motors were tested for their ability to hold specific RPM specified by the user through software. This was considered an integral component to the project because we needed to have accurate control of the motor's speeds and rotary positions along the ball screw. Without the motor tests conducted accurately; we could not proceed to controlling the angles of the leg joints on the robot accurately. This would then prevent us from properly being able to control the walking motion and balancing of the dog. The specific test conducted on the motors was to control their RPM by using encoders mounted to the spinning shaft in their center axis. The specific sensors used were the CUI AMT102 Capacitive Encoders. They have a rotational resolution of 8192 points per rotation [CPR], and a maximum reading speed of 7500 RPM. This meant that all there was left to do was write the script to control the encoder reading and motors then test them to an external metric. The external metric used was a standard Tachometer which measured the side of the motor as it rotated. This allowed for a quick and accurate comparison between the RPM external reading and the input value from the software.

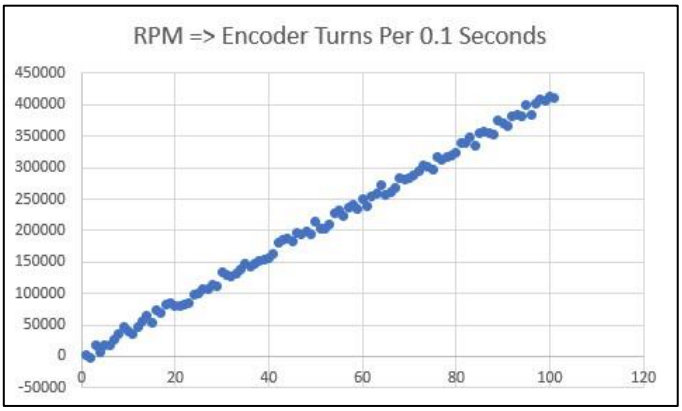


Figure 9.5 CPR/10th of Second & RPM Comparison

Source: Please refer to [6]

In the python scripts used to control the motors then legs, the RPM was controlled by adjusting the number of encoders clicks every 100 milliseconds in a constant while loop. We were able to control the motor to a very accurate degree after overcoming the current limiting issues and attaching the encoder to the motor mount with more stability.

3) Time of Flight Sensor Performance

The purpose of the time-of-flight sensors are to provide the position of the carriage along the rod where the carriage corresponds to a specific angle of the limbs which in turn, allows the robot to perform various gaits. The time-of-flight we decided to use is the VL6180X which is manufactured by STMicroelectronics. The sensor does not come installed on a breakout board but Adafruit does manufacture a breakout board capable of operating at any voltage between 3V-5V. The sensors used were chosen because their features aligned with what our project required. The sensors are primarily designed to read on the range of 0mm to 100mm with a potential to read higher ranges depending on the surface reflectance of the object and ambient conditions. The sensor can only be interfaced using the I2C bus. So, the sensor is limited to reading at a rate of 3.2Mbit/s but the Adafruit library that allows our single board computer to interface with the sensor was limited to 4Kbit/s.

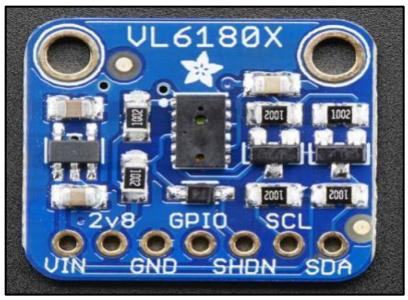


Figure 9.6 VL6180X

Source: Please refer to [6]

In order to verify that the VL6180X sensor was working properly and providing accurate readings for feedback. The figure below shows a top-level diagram of the test procedure setup.

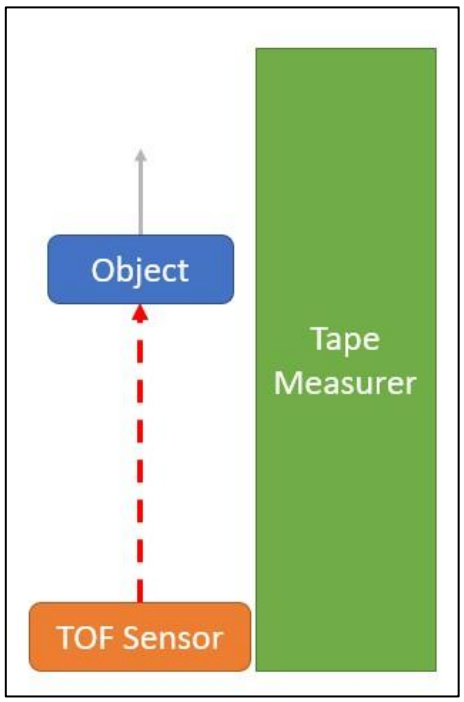


Figure 9.7 TOF Sensor Testing Procedure (Top-View)

Source: Please refer to [6]

The first step in the test procedure was to setup the TOF sensor directly aligned with an object. Then a tape measurer was placed along the side of both the object and sensor. A Python script was written to read the data from the sensor and calculate the average of ten samples. The average was compared to the actual range from the tape measurer. This sequence was done for every inch starting at a range of 1 inch to 8 inches. The percent error was then calculated comparing the measured range and the actual range. See Table 9.1. for the test results of this test procedure.

Tab. 9.1 TOF Test Results

Actual Distance (mm)	Measured Distance (mm)	Percent Error (%)
25.4	23.2	8.66
50.8	45.8	9.84
76.2	70.4	7.61
101.6	97.8	3.74
127.0	119.0	6.30
152.4	142.8	6.30
177.8	162.4	8.66
203.2	184.8	9.06

Source: Please refer to [6]

The percent error was consistently under 10% which is small enough for the sensors to fulfill their purpose.

X. Marketability Forecast

A. Increase Marketability Discussion

Robotics is an interdisciplinary branch of engineering and is normally a programmable, mechanical device used in place or to help a person to perform tasks with a high degree of accuracy. This opens a huge world of possibilities, from large, intricate systems to small common daily used items. Robots are seen everywhere and have integrated into our society and daily lives. These inventions are changing the way we live and work. Robots can be used in many situations and for lots of purposes.

When it comes to the potential consumers that our quadruped robot can be aimed toward, the most obvious buyer could be government agencies, such as the military. The global military robot market is projected to grow to 30.83 billion by the year 2022. This expansion in the market is due to developing countries such as India, China, and Russia adopting these technologies and increasing their research budget. One of the main drivers that seems to project success in this consumer market is the military's focus on using robots as an alternative for human soldiers for specific tasks. For instance, our quadruped robot has the capabilities of being deployed in situations which would be too dangerous for actual human personnel. Such situations could include providing medical assistance to wounded soldiers, or even traverse through a hostile environment. One of the advantages that our robot could potentially provide; and at the same time take advantage of, is the emphasis in legged locomotion. The use of legged robots over their wheeled counterparts is that they provide the advantages of traversing through uneven terrain and maintain dynamic stability.

Moreover, because the design of our robot has the attribute of being highly portable toward other applications, another potential consumer market could be in the logistics robotic market. The global logistics robot market is expected to reach around 11.8 billion by 2022. Logistics robots are used to

automate the process of storing and moving products as they are packaged through the supply chain. As companies, such as Amazon continue to experience rapid growth and dominate the ecommerce domain; the demand for faster and more efficient logistical strategies will need to be considered for the ever-expanding group of customers. According to Tractica Research Group, "worldwide shipments of warehousing and logistics robots will grow rapidly over the next 5 years." In fact, they predict that the number of units will increase from 194,000 in 2018 to 938,000 by 2022. As previously mentioned, because our robot can be ported into other applications, the use of it as a logistical robot could be a viable option to meet the market demands of the logistic robotic industry.

In order to fully understand where a project will fit within an existing market or industry. One must examine what products their project will be competing against. Robotics is not an entirely new area of study but the marketing and retailing of robots is relatively new. With the robotics industry still in its infancy, a semi-autonomous quadruped will be able to find a place amongst its competition. The SPOT, in our case, would be the primary competitor of our robotic quadruped. The SPOT features include speed of 1.6 m/s, 90-minute runtime, swappable battery, 360 degrees of vision for autonomy, and customization options. The applications that Boston Dynamics claim to be suitable for the SPOT are construction, oil/gas, entertainment, and public safety. Our robotic quadruped would be primarily competing against the SPOT in public safety. In order to compete with something such as the SPOT, we would need to make some changes to our robotic quadruped.

B. Necessary Changes for Manufacturing

With the cost of creating the robotic quadruped being approximately $4000, in order to sell it as a product and ready for manufacturing, as well as compete with other products, then we would need to make some changes to the hardware, software, and construction of the robotic quadruped.

1) Hardware

In order to get the robotic quadruped ready for manufacturing then we would need to create a PCB board or microcontroller that can handle everything that the Nvidia Jetson Nano does as well as the O-drives. The idea would be to simplify the boards into one complete board with only the correct number of inputs and outputs to control everything that is needed on the robotic quadruped. This will simplify the wire management throughout the robot as well as draw less power throughout the entire system so then the robotic quadruped could run longer and more efficiently. With combining all the boards into one, it will clean up the robotic quadruped, give less room for hardware errors, as well as lower the cost in creating the robotic quadruped. With changing some of the hardware and lowering the cost, that would make the robotic quadruped more marketable.

2) Software

The next area that would need improvement before the robotic quadruped is ready for manufacturing is the software that is being ran for the robotic quadruped. Currently the robotic quadrupeds' movements are mostly controlled through the O-drives which take a lot of lines of code which makes it less efficient and more prone to errors. Also, in order to make the robotic quadruped more marketable, we would need to add more software to make the robotic quadruped more functional as well as stable. With updated software and more testing, the code, the robotic quadruped would become more autonomous as well as dynamically stable. This means that the robotic quadruped would be smarter, faster, and more stable. With more testing and more features, it would make the robotic quadruped better for more applications as well as more marketable for what it can do.

3) Construction

The last area that would need improvement to make the robotic become more marketable and readier for manufacturing is the construction of the robotic quadruped. To improve on cost, it would be beneficial to have a company produce the construction of the robotic quadruped in bulk. It would also be better to make the robotic quadruped not as big and heavy. Using lighter materials and making it smaller would make the robotic quadruped more efficient as well as cost effective. Another thing that the robotic quadruped will need is protective encasement for the legs as well as the body to make

sure the wires and boards to not get harmed as well as for user's safety.

With changing the hardware, software and construction of the robotic quadruped, it will make the robot, lighter, smaller, smarter and most importantly at lower costs. The idea is to change the hardware into one microcontroller/ board, use lighter constructing materials, as well as modifying the code/ software to make the robotic quadruped smarter and more capable of completing its tasks. All and all, it would not be very hard to do, just smaller adjustments and testing to make the robotic quadruped ready for manufacturing and ready for the market.

XI. Conclusion

With the increasing number of natural and non-natural disasters there is a demand for solutions to aid or assist in saving as many lives as possible as well as minimizing the cost of said disasters. This demand along with the advancements in technology, even more so in the fields of robotics and automation, is enough reason for the development of a semi-autonomous robotic quadruped. Although this specific project will take multiple years to complete and will need to be optimized better both electronically and mechanically, the benefits of aiding in search & rescue operations that could potentially save lives is insurmountable. When this very complex project is finally completed by several teams of undergraduates, what could possibly be as exciting is what projects in advanced robotics does industry have in store for us.

References

[1] "*Gait* - Physical Therapist Assistant 180 with P. Hill at Washtenaw Community College," *STUDYBLUE*. [Online]. Available: https://www.studyblue.com/notes/note/n/gait/deck/16020893.

[2] "54V, 1.5kW, >99% Efficient 3-Phase BLDC Drive Reference Design," *PowerPulse.net*, 26-Aug-2019. [Online]. Available: https://powerpulse.net/54v-1-5kw-99-efficient-3-phase-bldc-drive-reference-design/.

[3] "Facts Statistics: U.S. catastrophes," *III*. [Online]. Available: https://www.iii.org/fact-statistic/facts-statistics-us-catastrophes.

[4] "MPU-6050 GY-521 3-Axis Accel & Gryo Sensor Module," *ProtoSupplies*. [Online]. Available: https://protosupplies.com/product/mpu-6050-gy-521-3-axis-accel-gryo-sensor-module/.

[5] "Number of recorded natural disaster events," *Our World in Data*. [Online]. Available: https://ourworldindata.org/grapher/number-of-natural-disaster-events.

[6] Alfred Martinez III, Edgar Granados, Marcus Huston and Kristian Ornelas, "Search & Rescue Robotic Quadruped," Dept. Elect. Eng., CSUS., Sacramento, CA

[7] B. Yirka, "Robot snake automatically wraps around an object when thrown (w/ Video)," *Phys.org*, 22-Mar-2013. [Online]. Available: https://phys.org/news/2013-03-robot-snake-automatically-thrown-video.html.

[8] *BBC World*. National Interagency Fire Center (NFIC), 2017.

[9] Boston Dynamics [Online]. Available: bostondynamics.com

[10] D. Kumar and V. Divakar, "SPY ROBOT USING SELF-BALANCING ALGORITHM WITH PAN AND TILT ..." [Online]. Available: https://www.researchgate.net/profile/Vishnu_Divakar/publication/281746636_SPY_ROBOT_USING_SELF-BALANCING_ALGORITHM_WITH_PAN_AND_TILT_CONTROL_OF_CAMERA/links/55f6ee1108aeafc8abf51ed6/SPY-ROBOT-USING-SELF-BALANCING-ALGORITHM-WITH-PAN-AND-TILT-CONTROL-OF-CAMERA.pdf.

[11] David Wooden, Matthew Malchano, Kevin Blankespoor, Andrew Howard, Alfred Rizzi, Marc Raibert, "Autonomous Navigation for BigDog", IEEE International Conference on Robotics and Automation.

[12] Department of Conservation.

[13] Dr. Asad Yousuf, Mr. William Lehman, Dr. Mohamad A. Mustafa, Dr. Mir M Hayder, "Introducing Kinematics with Robot Operating System (ROS)", 122nd ASEE Annual Conference & Exposition.

[14] F. Tedim and D. Roye, Imprensa da Universidade de Coimbra, 2020.

[15] G. Kenneally, A. De, D. Koditschek. "Design Principles for a Family of Direct-Drive Legged Robots." IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 1, NO. 2, JULY 2016.

[16] H. P. Moravec, "Robot," *Encyclopædia Britannica*, 27-Nov-2019. [Online]. Available: https://www.britannica.com/technology/robot-technology. .

[17] Huang, W., Kim, J., and Atkeson, C. (2013). Energy-based optimal step planning for humanoids. In 2013 IEEE International Conference on Robotics and Automation (ICRA), pages 3124–3129, Karlsruhe, Germany
IEEE Entity Web Hosting - Home. [Online]. Available: http://ewh.ieee.org/soc/ras/conf/fullysponsored/ssrs/2011/demo.html.

[18] Ilya Afanasyev, Artur Sagitov, Evgeni Magid, "ROS-Based SLAM for a Gazebo-Simulated Mobile Robot in Image-Based 3D Model of Indoor Environment", International Conference on Advanced Concepts for Intelligent Vision Systems, 2015.

[19] Introducing SpotMini. Boston Dynamics. https://www.youtube.com/watch?v=tf7IEVTDjng
Joel Chestnutt, Navigation Planning for Legged Robots.

[20] K. Reid, "2011 Japan earthquake and tsunami: Facts, FAQs, and how to help," *World Vision*, 29-Feb-2020. [Online]. Available: https://www.worldvision.org/disaster-relief-news-stories/2011-japan-earthquake-and-tsunami-facts.

[21] Kalouche, Simon. "Design for 3D Agility and Virtual Compliance Using Proprioceptive Force Control in Dynamic Legged Robots." The Robotics Institute Carnegie Mellon University, Carnegie Mellon University, Aug. 2016, https://www.ri.cmu.edu/publications/design-for-3d-agility-and-virtual-compliance-using-proprioceptive-force-control-in-dynamic-legged-robots/.

[22] Kenta Takaya, Toshinori Asai, Valeri Kroumov, Florentin Smarandache, "Simulation Environment for Mobile Robots Testing Using ROS and Gazebo", 20th International Conference on System Theory Control and Computing (ICSTCC), 2016.

[23] Kuindersma, S., Permenter, F., and Tedrake, R. (2014). An efficiently solvable quadratic program for stabilizing dynamic locomotion. In IEEE Intl. Conf. on Robotics and Automation (ICRA), Hong Kong, China.

[24] M. Shuman, "Fire risk leaves Tuolumne County residents scrambling to find affordable insurance," *modbee*, 16-Jul-2019. [Online]. Available: https://www.modbee.com/news/local/article232609147.html.

[25] M. Simon, "Watch Boston Dynamics' Robot Dog Strut Through a Construction Site," *Wired*, 12-Oct-2018. [Online]. Available: https://www.wired.com/story/boston-dynamics-is-prepping-its-robot-dog-to-get-a-job/.

[26] Marc Sons, Christoph Stiller, "Efficient Multi-Drive Map Optimization towards Life-long Localization using Surround View", Intelligent Transportation Systems (ITSC) 2018 21st International Conference on, pp. 2671-2677, 2018.

[27] *Number of Fires throughout the United States*. 2018.

[28] O. Takehiko, K. Hajime, O. Hock, J. Šedo, E. Raju, L. S. R. Krishna, N. Rao, M. A. Rusafi, M. Billal, F. Yasmin, S. Iqbal, and A. Talli, "Inverse kinematics problem of 3-DOF robot arm in 2D plane. (a) Three...," *ResearchGate*, 31-Jul-2018. [Online]. Available: https://www.researchgate.net/figure/Inverse-kinematics-problem-of-3-DOF-robot-arm-in-2D-plane-a-Three-joints-and-an_fig3_46401997.

[29] P. Sayta, "Thermite Firefighting Robot," *#InnovationVibe*. [Online]. Available: https://innovationvibe.com/2020/thermite-firefighting-robot/.

[30] S. A. Archfield, R. M. Hirsch, A. Viglione, and G. Blöschl, "Fragmented patterns of flood change across the United States," *AGU Journals*, 09-Oct-2016. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/full/10.1002/2016GL070590.

[31] S. Seok, A. Wang, D. Otten, and S. Kim, Actuator design for high force proprioceptive control in fast legged locomotion, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012.

[32] Snapolitano, "The Future of Disaster Relief Includes Drones and AI," *U.S. Chamber of Commerce Foundation*, 07-Jun-2018. [Online]. Available: https://www.uschamberfoundation.org/blog/post/future-disaster-relief-includes-drones-and-ai.

GLOSSARY

Robot: a programmable machine capable of carrying out complex actions

Quadruped: an animal which has four feet

Inertial Measurement Unit (IMU): an electronic device that measures and reports a body's force, angular rate, and orientation

Time-of-Flight Sensor: an electronic device that measures and reports distance using an infrared light transmitter and infrared light receiver.

Autonomous: capacity to make an informed decision

Incremental Encoder: an electronic device that provides rotational position changes

Brushless DC Motor (BLDC): a motor that converts supplied electrical energy into mechanical rotational energy

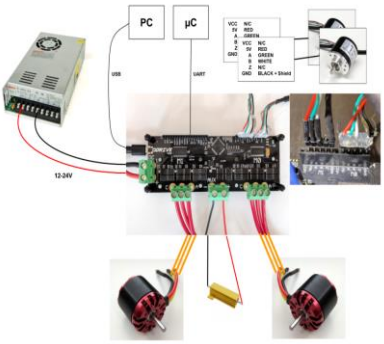
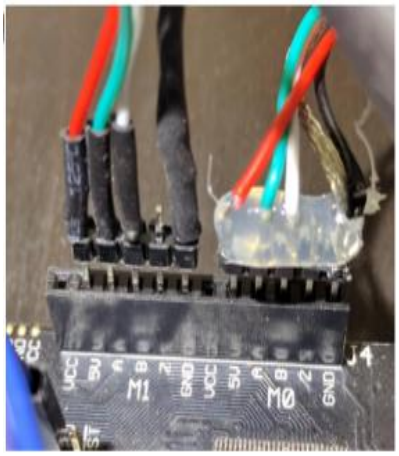
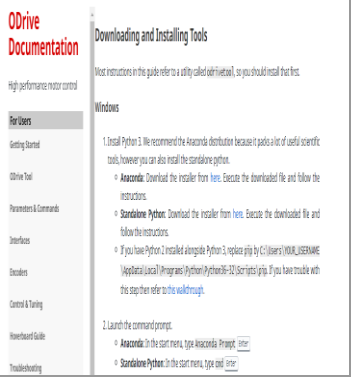
Multiplexer: a device that selects between analog or digital input signals and forwards it to a single output

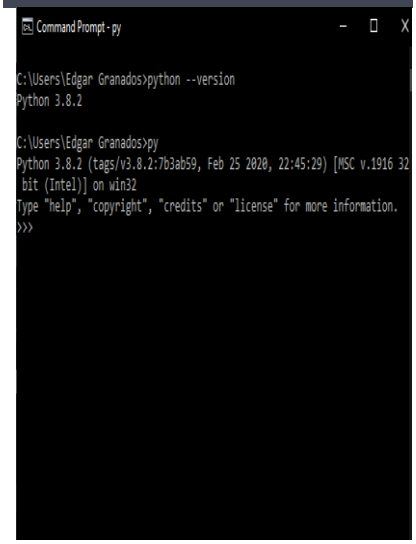
APPENDIX A. USER MANUAL

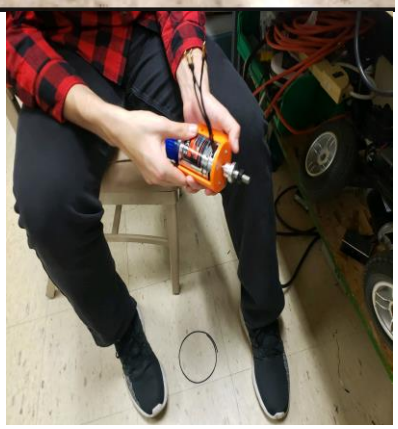
INTRODUCTION:

This user manual is intended as a reference resource for future teams who are interested in continuing the design and development of the robotic quadruped. The user manual outlines several key components and how they are to be set up in order to have them fully operational. In addition to the step by step process of setting up, the user manual also includes visual aids to help in the demonstration of each component set up. It is assumed that the readers of this manual will be familiar with some of the software tools needed to follow the step by step process; if in any case they are not, please refer to the online official documentation for each unfamiliar software tool. If there are any questions or concerns regarding the steps in the user manual, please contact us.

Table A1.1. Large-Scale Leg Startup

Step Number/Name:	**Description:**	**Process:**	**Visual Aids:**
Step 1: ODrive	This step is to ensure the ODrive motor controller board is correctly connected to the 3-Phase BLDC motors as well as the power supply and computer interface via USB.	[1] Connect the A, B and C phase terminals on the board to the motors ensuring that the middle wire of the motor is connected to the B phase terminal [2] Connect the 5 wires from the encoder to the proper terminals on the ODrive. (Make sure to match encoders to the corresponding axis of the motor. Marked as “M0” or “M1” [3] Connect the Jetson Nano you will be using to control the motors via USB to the micro-USB port on the ODrive board. (Please note that the board cannot turn on via USB) [4] To power up the board, connect a 12-24V power supply to the DC positive and negative ports on the board. [5] Open up an internet browser and proceed to the ODrive Documentation Website to set up the appropriate software.	  

		[6] Install either Python3 or Anaconda (a distribution of Python). It is extremely recommended that you install Anaconda as several essential libraries are installed with it that would otherwise be needed to install separately with Python Standalone [7] Open up a terminal window and run the command "python", "py" or "python3". Assure that Python3 is the version currently running by using the command Python –version before entering the previous command. [8] Install the ODrive tool onto the python environment by using the following command: **pip install odrive** [9] Assuming you are using Windows, you will need to install a final driver for the ODrive tool to detect the ODrive module. Install the Zadig utility driver as shown in the visual aid. [10] Check 'List All Devices' from the options menu, and select 'ODrive 3.x Native Interface (Interface 2)'. With that selected in the device list choose 'libusb-win32' from the target driver list and then press the large 'install driver' button. [11] To launch the main interactive ODrive tool, type odrivetool Enter. Connect your ODrive and wait for the tool to find it. Now you can,	  

		for instance type odrv0.vbus_voltage Enter to inspect the boards main supply voltage. It should look something like the diagram shown in the visual aid. [12] Now you will be able to type commands into the odrivetool shell and will be able to control the motor. One thing to keep in mind is in order to have full control of the motor, and encoder must be mounted to it. Otherwise the motor will not calibrate and motor control may not function.	  ODrive control utility v0.4.0 Please connect your ODrive. Type help() for help. Connected to ODrive 306A396A3235 as odrv0 In [1]: odrv0.vbus_voltage Out[1]: 11.97055721282959
Step 2: Jetson Nano	This step provides the process of ensuring the I2C MUX, TOF sensors, and power are connected properly.	[1] Connect the appropriate I2C bus pins from the Jetson Nano to Vin, GND, SDA, and SCL on the I2C Mux (TCA9548A) [2] Connect any of the SDA# and its corresponding SCL# to the SDA/SCL pins to the TOF sensor (VL6180x). Also connect the Vin and GND pins on the TOF sensors to the same power supply as the I2C MUX [3] Ensure the proper peripherals are connected to the Nano such as the keyboard, mouse, and HDMI cable [4] Power up the Nano via its micro-usb cable by a 5V power supply capable of providing at least 1.5A	

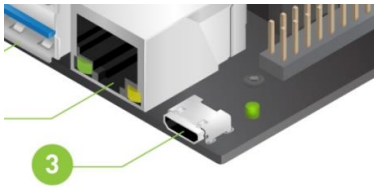
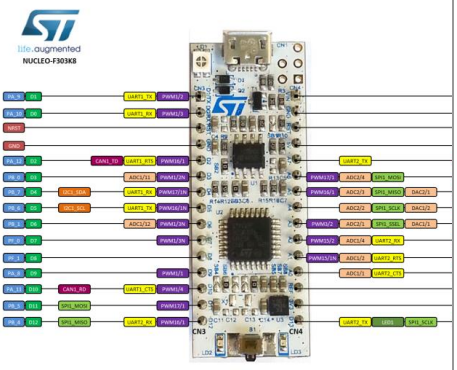
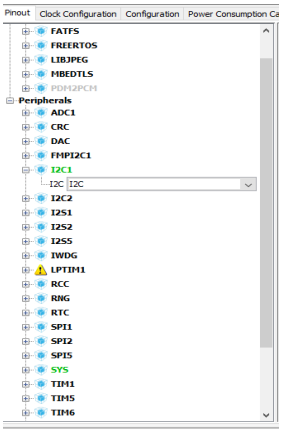
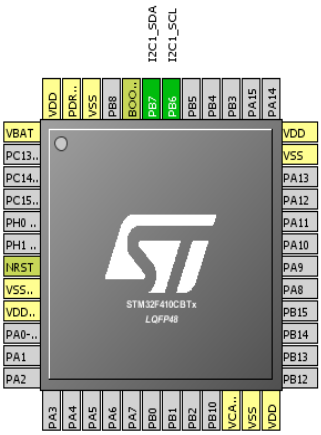
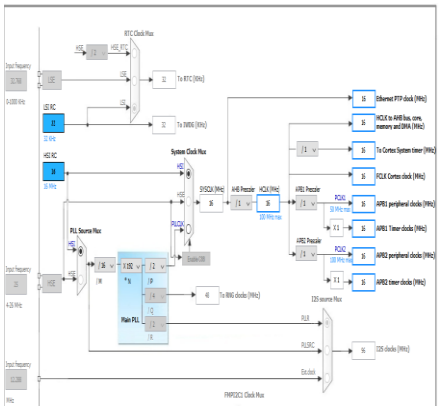
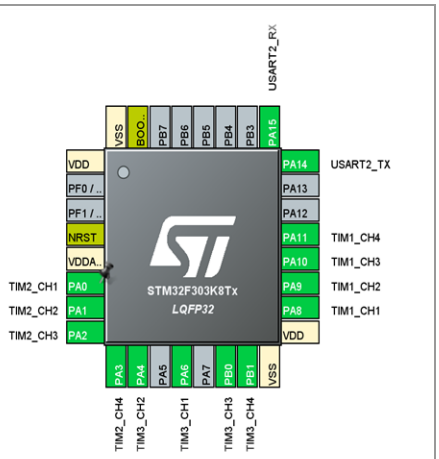
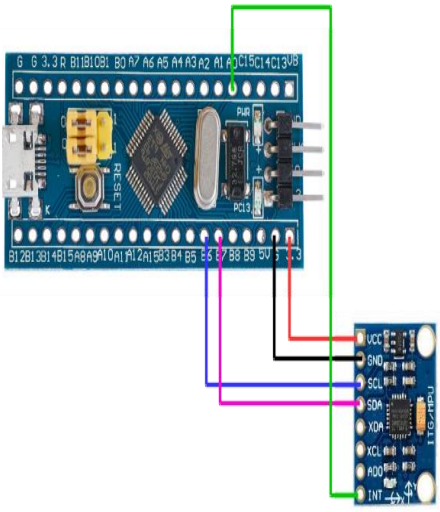
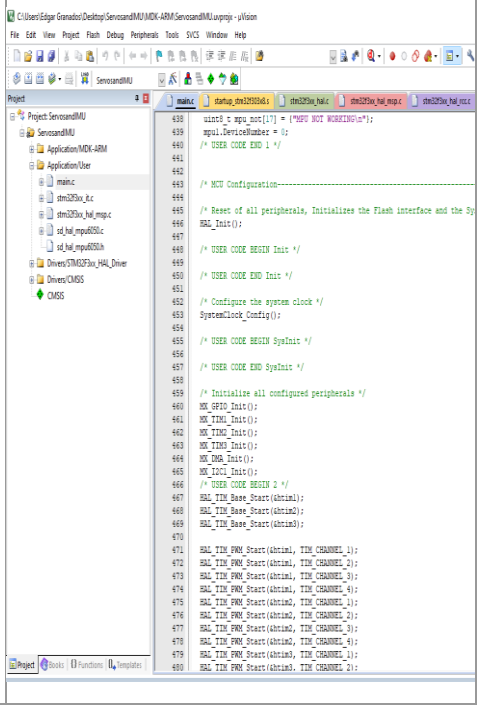
			 
Step 3: Jetson Nano Access/VSCode	This step provides the process of accessing the VSCode that contains the Python source code that can be built on to create more demanding actions from the large leg.	[1] Go to the terminal and use the “sudo i2c detect” command to ensure that the I2C address of the MUX is detected. (Ensure that Step 2 is completed) [2] To enter the VSCode IDE enter in the command “code-oss” and the relevant code can be found in the “Megabyte” folder [3] The Python scripts can now be executed. (Note: The scripts that require the use of the I2C bus must be run as a super user in order to access the bus [4] The Large-Scale Leg can now be controlled	sudo i2cdetect -r -y 1 0 1 2 3 4 5 6 7 8 9 a b c d e f 00: – -- – – – – – – – – – – – – 10: – -- – – – – – – – – – – – – -- 20: – -- – – – – – – – – – – – – -- 30: – -- – – – – – – – – – – – – -- 40: – -- – – – – – – – – – – – – -- 50: – -- – – – – – – – – – – – – -- 60: – -- – – – – – – – – – – – – -- 70: – -- – – – – – –

Table A1.2. I2C Set Up

Step Number / Name:	Description:	Process:	Visual Aids:
Step 1: Setting up the communication protocol(s)	This step provides information on how to setup the communication protocol on an STM32 board using the STM32CubeMX software.	[1] Download the STM32 Software from their official website like the visual aid shows. [2] Load the software up and you will need to make sure that you select the correct development board you are using, otherwise the pin configurations will not work. [3] Now we will be able to properly set up any supported communication protocol needed to have the MCU interfaced with another module. The following visual aids demonstrate the supported communication protocols on this MCU; as well as the pin diagram that becomes automatically assigned by the software when enabling a feature. [4] The next step will be to set up the proper clock frequency configuration in the “Clock Configuration” tab. To do this we simply input our desired frequency	  

		within the allowed boxes. The desired input frequency can be acquired for the module specifications you are trying to interface. And the specific bus can also be determined through the MCU's datasheet. [5] With the appropriate pin and clock configurations set by the software, you will now be able to appropriately interface any desired module sensor that supports the configuration set by you.	 

Step 2: IMU interfacing and measurements	This step is to give information on how to interface the IMU with the ARM cortex development board or Jetson Nano board.	[1] Connect the appropriate I2C bus pins from assigned pins on the STM32 Board to Vin, GND, SDA, and SCL on the MPU 6050. [2] Also connect the Vin and GND pins on the TOF sensors to the same power supply as the I2C MUX [3] Ensure the proper peripherals are connected to the MCU such as the keyboard, mouse, and HDMI cable [4] Download the software project from the appropriate source code folder, so that you are able to use the IMU library that was written for the MCU. [5] Run the program using either Keil uVision or Attollic Truestudio in order to test if the IMU is sending data to the MCU. [6] To run on the Jetson Nano, simply install the adafruit library using the following command: **sudo pip3 install adafruit-circuitpython-mpu6050**	  

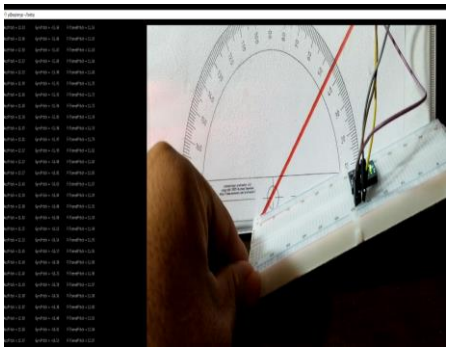
		[7] Now that the proper support library installed, the next step is to run the provided python script: IMUtest.py in order to have the IMU module properly interfaced with the Jetson Nano. [8]: Run the script by typing the following command: **python IMUTest.py** [9] You should see angles being printed onto the terminal shell after clicking enter on the keyboard.	 

Table A1.3. Small Dog Set Up:

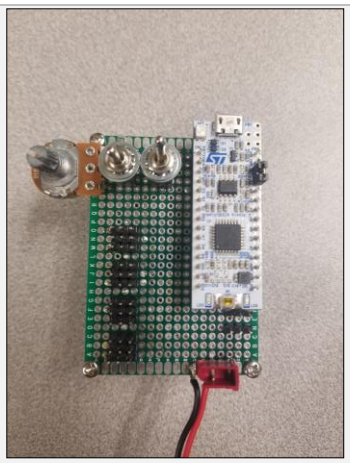
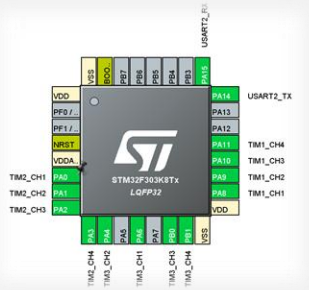
Step Number / Name:	Description:	Process:	Visual Aids:
Step 1: Checking all wire connections	Making sure all Servo Motors are connected correctly. *This step assumes that all the Software has been updates and the user will not be concerned with maintenance or upgrading.	[1] Simply check the pin female to male connectors to ensure that all Servo Motor wires (12 motors x 3 wires each) [2] If any are loose or disconnected, reconnect in the correct orientation. Each set of wires should be labeled 1 through 12.	 
Step 2: Powering and Testing	This step requires the user to connect 2S Li-Po Batteries (7.5 V) to the onboard connector. Alternatively, a DC barrel-jack connector is also available on the PCB board to connect an AC-DC, 5V-10Amp Power Supply. *Note: The Barrel Jack Connector will add some instability into the system.	[1] Simply attach either power source to the connectors available. [2] A switch at the top of the PCB board will allow the user to keep the power supply attached without constantly running. A red dot will indicate the ‘ON’ setting.	

Figure B1.1. Large-Scale Dog Hardware Block Diagram

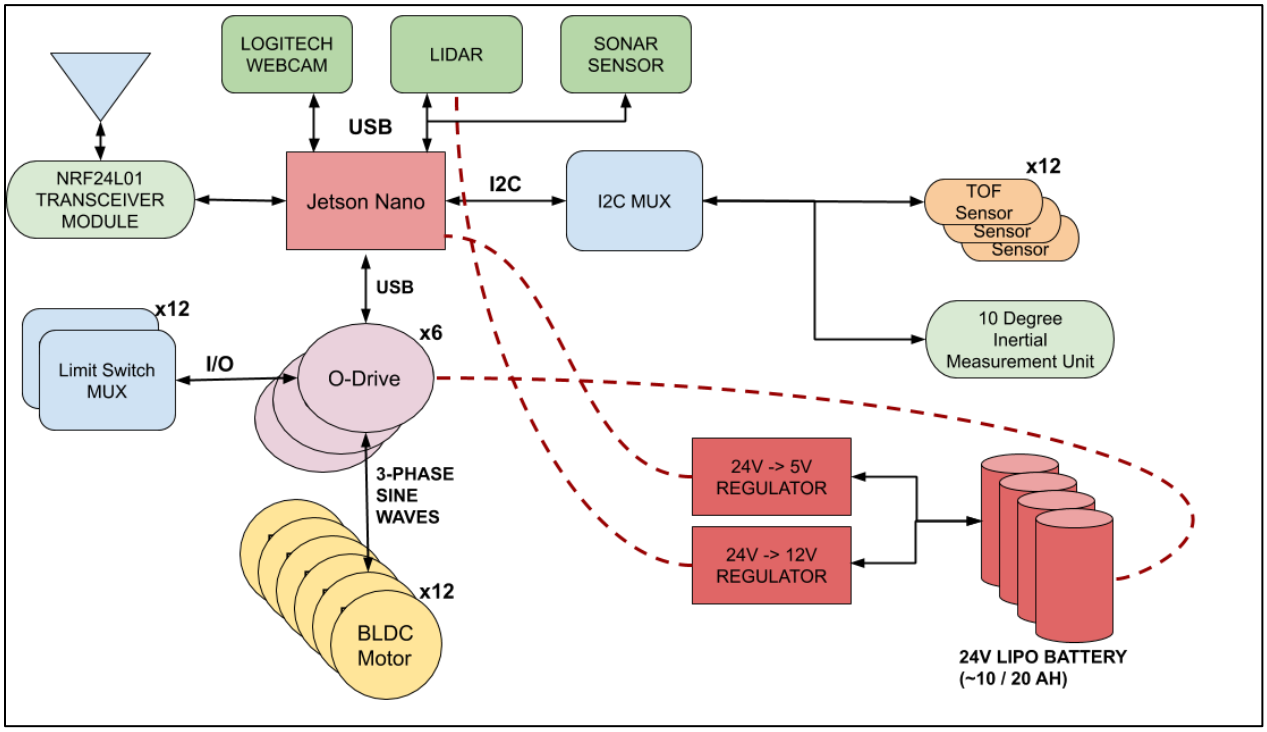


Figure B1.2. I2C MUX Block Diagram

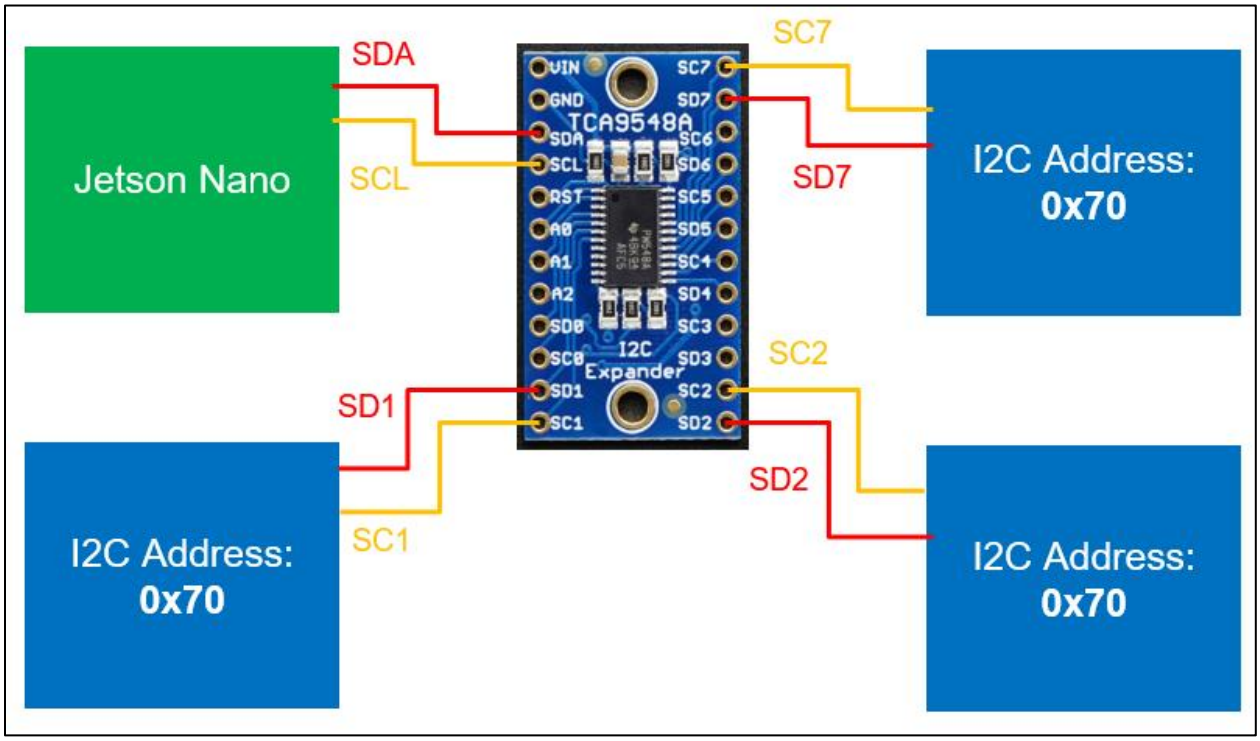


Figure B1.3. Time-of-Flight Testing Procedure (Top-View)

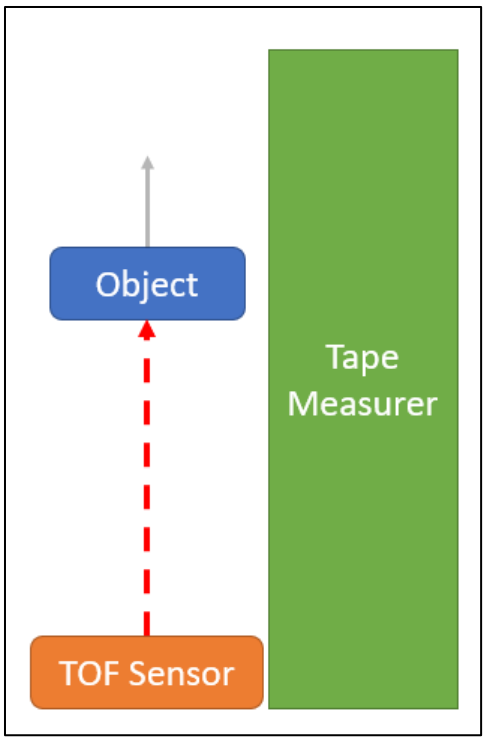
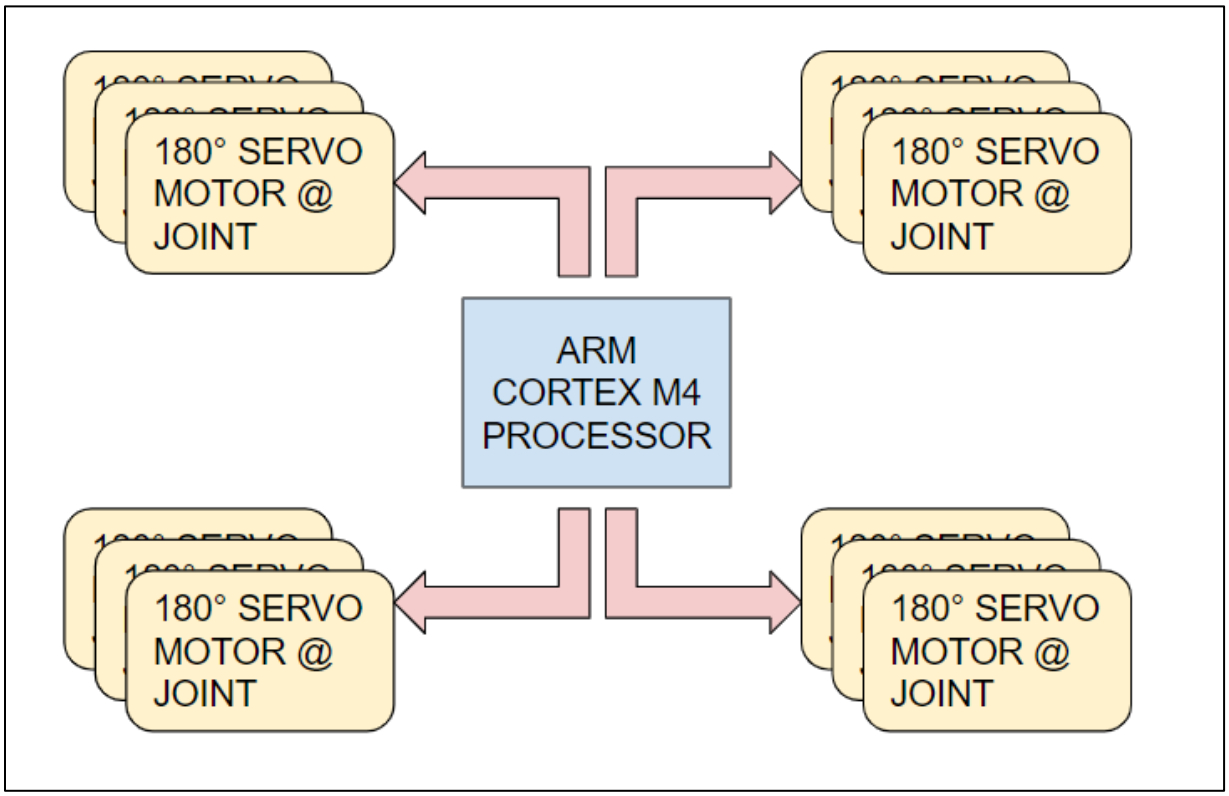


Figure B1.4. Small-Scale Dog Hardware Block Diagram



Note: Barrel-jack DC Power Supply not shown (5V – 10Amp).

Figure C1.1. TOF Sensor Feedback Flowchart

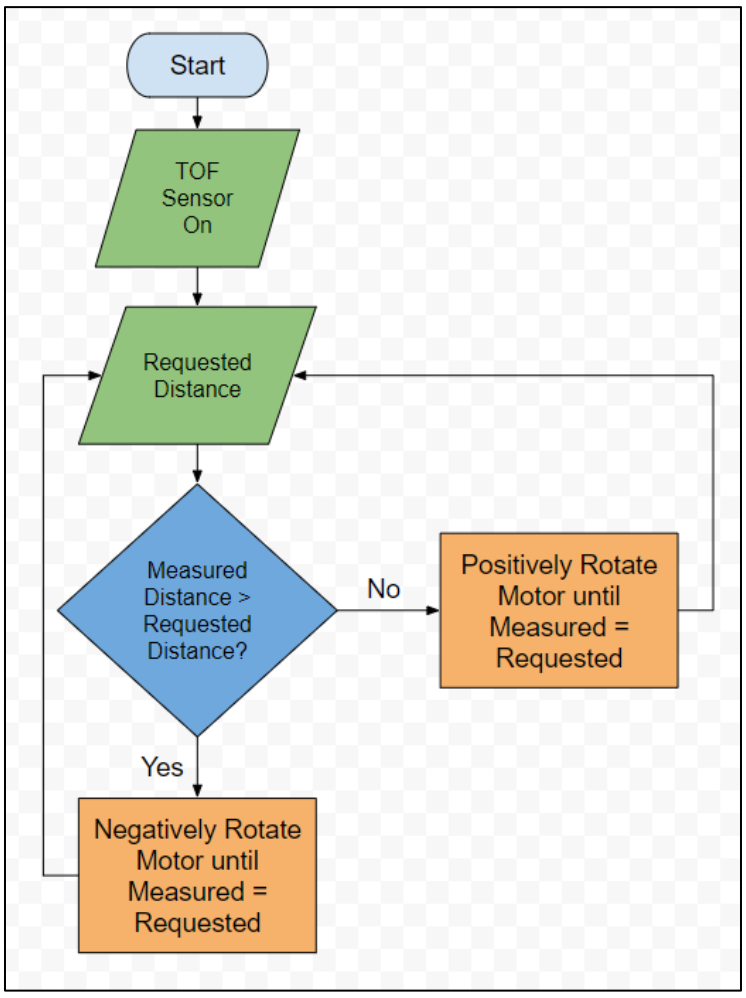


Figure C1.2. ORB-SLAM2 3D Point Cloud Flowchart

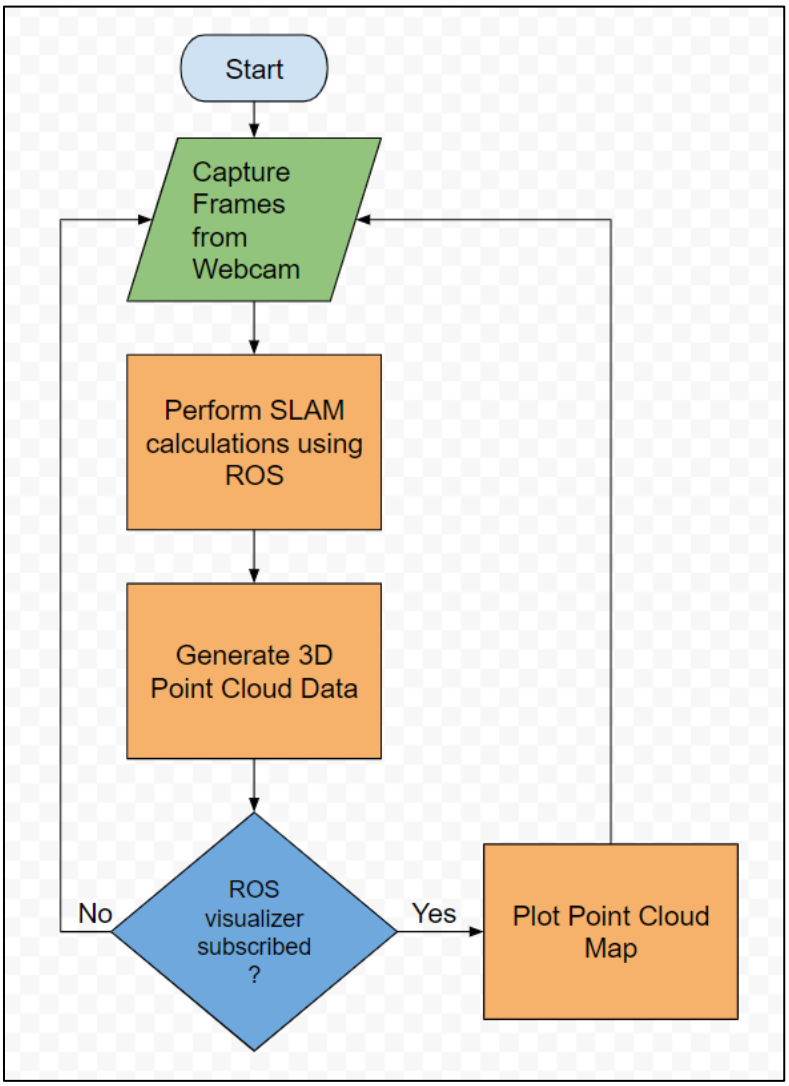


Figure C1.3. Small-Scale Dog Software Block Diagram

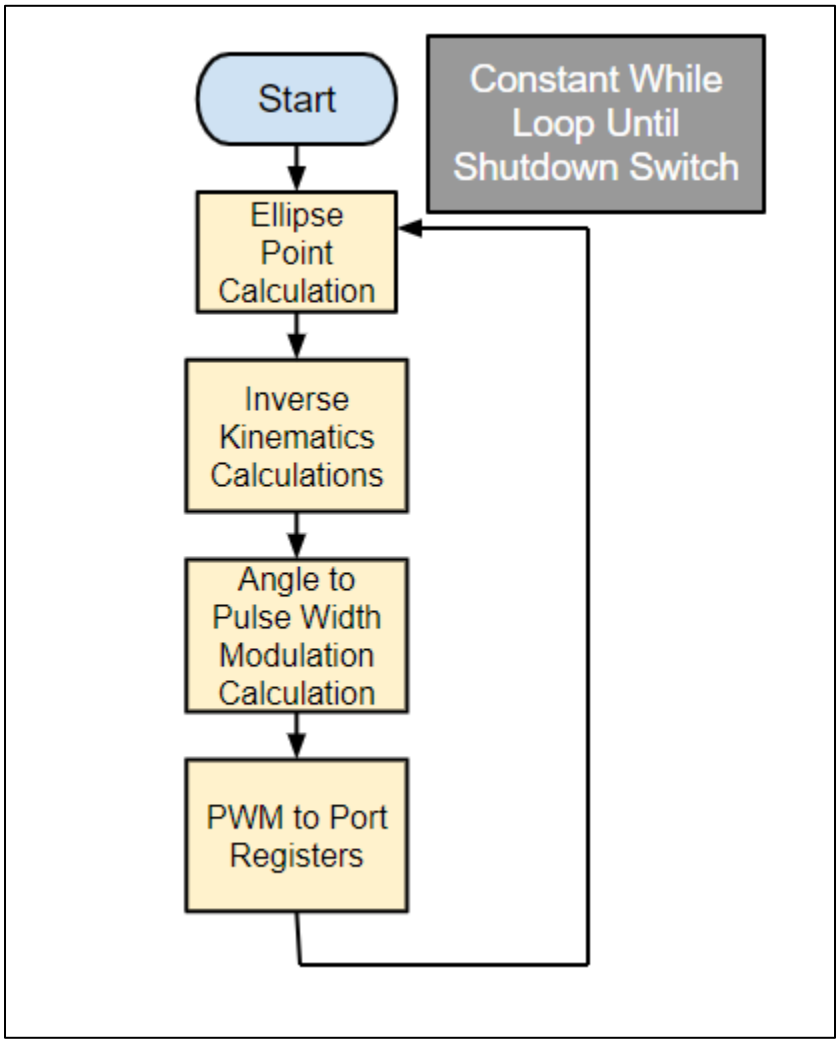


Figure D1.1. Large-Scale Dog CAD Model

Figure D1.2. Large-Scale Dog CAD Model (Side-View)

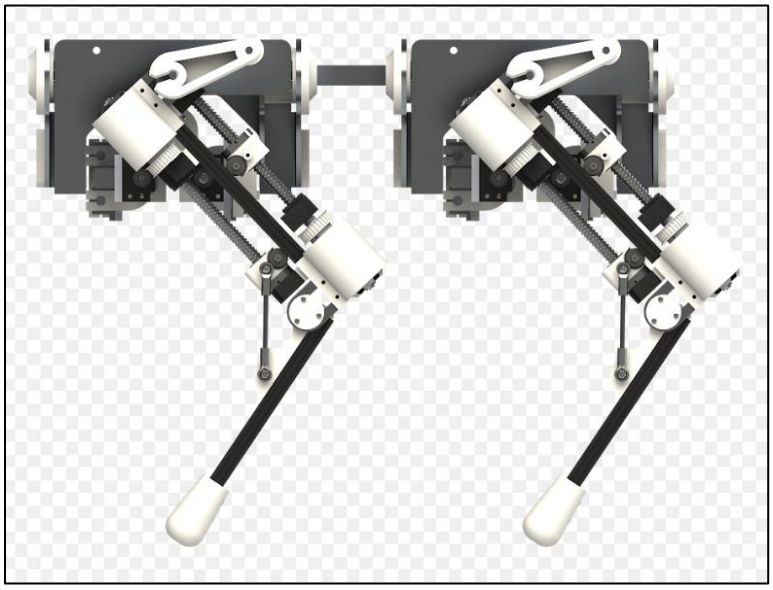
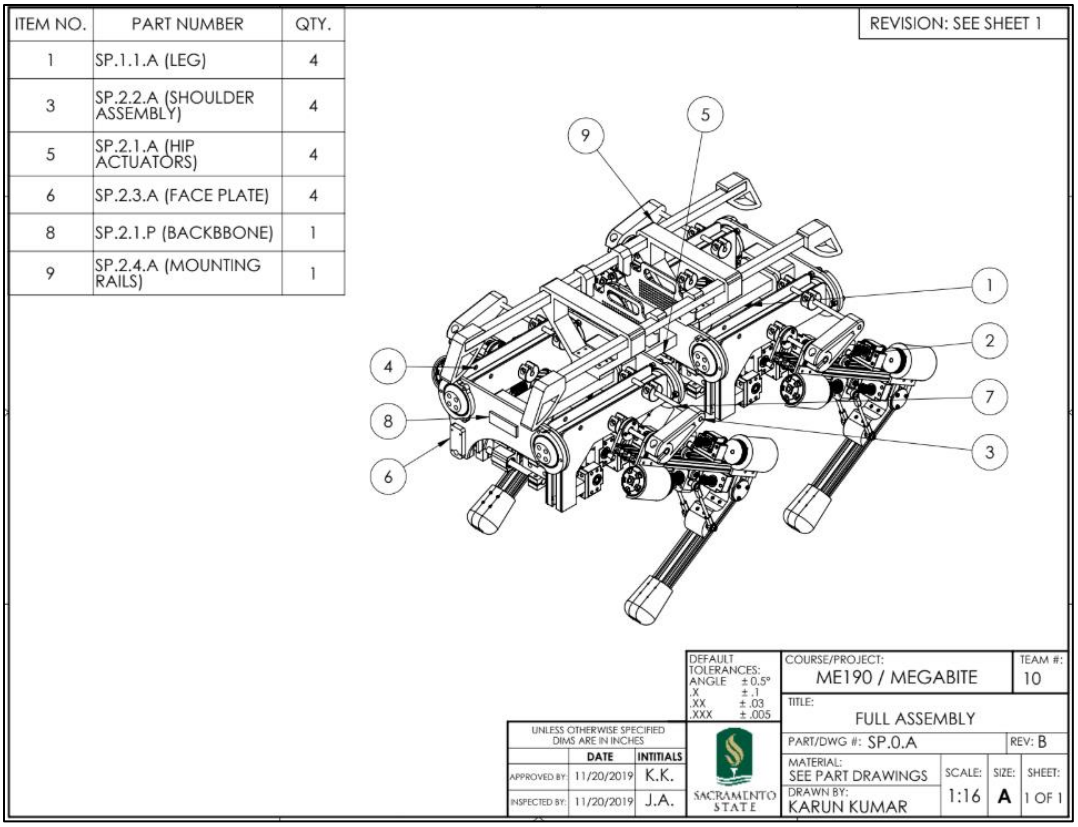
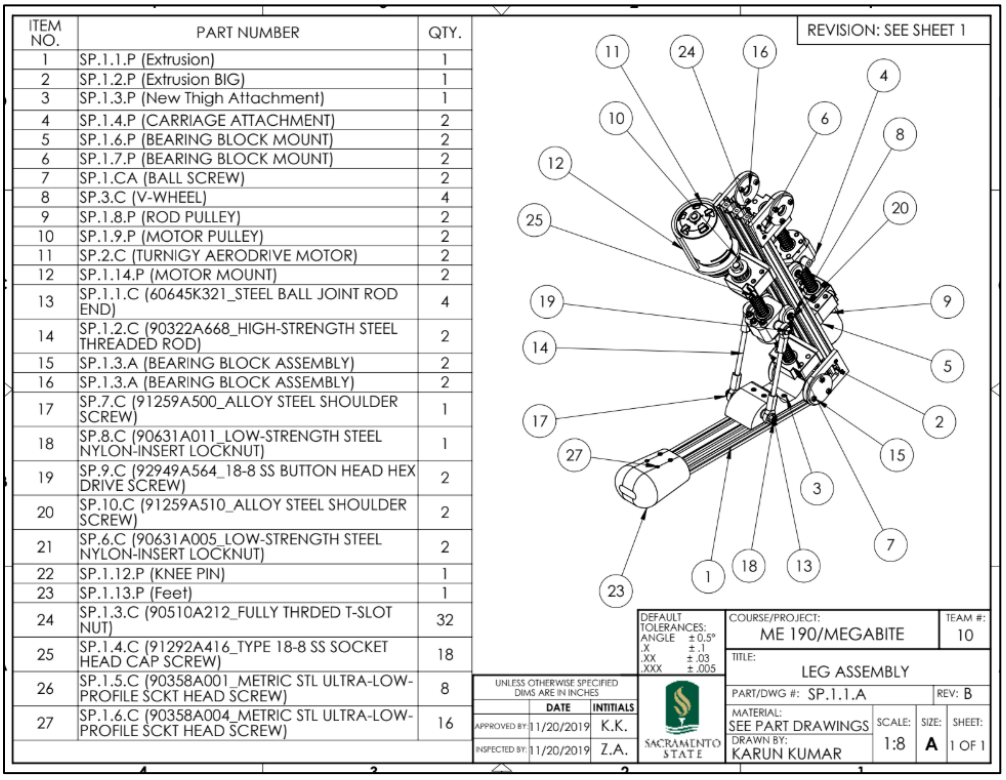


Figure D1.3. ME Full Assembly



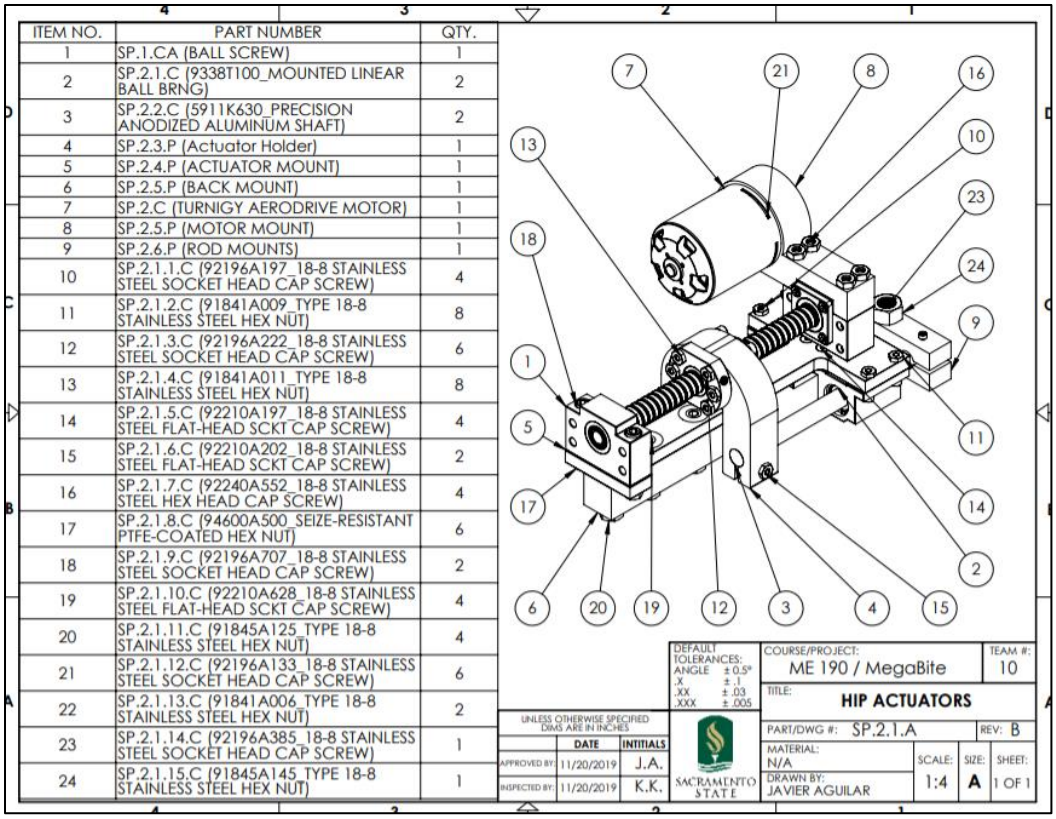
ITEM NO.	PART NUMBER	QTY.
1	SP.1.1.A (LEG)	4
3	SP.2.2.A (SHOULDER ASSEMBLY)	4
5	SP.2.1.A (HIP ACTUATORS)	4
6	SP.2.3.A (FACE PLATE)	4
8	SP.2.1.P (BACKBBONE)	1
9	SP.2.4.A (MOUNTING RAILS)	1

Figure D1.4. ME Leg Assembly



ITEM NO.	PART NUMBER	QTY.
1	SP.1.1.P (Extrusion)	1
2	SP.1.2.P (Extrusion BIG)	1
3	SP.1.3.P (New Thigh Attachment)	1
4	SP.1.4.P (CARRIAGE ATTACHMENT)	2
5	SP.1.6.P (BEARING BLOCK MOUNT)	2
6	SP.1.7.P (BEARING BLOCK MOUNT)	2
7	SP.1.CA (BALL SCREW)	2
8	SP.3.C (V-WHEEL)	4
9	SP.1.8.P (ROD PULLEY)	2
10	SP.1.9.P (MOTOR PULLEY)	2
11	SP.2.C (TURNIGY AERODRIVE MOTOR)	2
12	SP.1.14.P (MOTOR MOUNT)	2
13	SP.1.1.C (60645K321_STEEL BALL JOINT ROD END)	4
14	SP.1.2.C (90322A668_HIGH-STRENGTH STEEL THREADED ROD)	2
15	SP.1.3.A (BEARING BLOCK ASSEMBLY)	2
16	SP.1.3.A (BEARING BLOCK ASSEMBLY)	2
17	SP.7.C (91259A500_ALLOY STEEL SHOULDER SCREW)	1
18	SP.8.C (90631A011_LOW-STRENGTH STEEL NYLON-INSERT LOCKNUT)	1
19	SP.9.C (92949A564_18-8 SS BUTTON HEAD HEX DRIVE SCREW)	2
20	SP.10.C (91259A510_ALLOY STEEL SHOULDER SCREW)	2
21	SP.6.C (90631A005_LOW-STRENGTH STEEL NYLON-INSERT LOCKNUT)	2
22	SP.1.12.P (KNEE PIN)	1
23	SP.1.13.P (Feet)	1
24	SP.1.3.C (90510A212_FULLY THRDED T-SLOT NUT)	32
25	SP.1.4.C (91292A416_TYPE 18-8 SS SOCKET HEAD CAP SCREW)	18
26	SP.1.5.C (90358A001_METRIC STL ULTRA-LOW-PROFILE SCKT HEAD SCREW)	8
27	SP.1.6.C (90358A004_METRIC STL ULTRA-LOW-PROFILE SCKT HEAD SCREW)	16

Figure D1.5. ME Hip Assembly



ITEM NO.	PART NUMBER	QTY.
1	SP.1.CA (BALL SCREW)	1
2	SP.2.1.C (9338T100_MOUNTED LINEAR BALL BRNG)	2
3	SP.2.2.C (5911K630_PRECISION ANODIZED ALUMINUM SHAFT)	2
4	SP.2.3.P (Actuator Holder)	1
5	SP.2.4.P (ACTUATOR MOUNT)	1
6	SP.2.5.P (BACK MOUNT)	1
7	SP.2.C (TURNIGY AERODRIVE MOTOR)	1
8	SP.2.5.P (MOTOR MOUNT)	1
9	SP.2.6.P (ROD MOUNTS)	1
10	SP.2.1.1.C (92196A197_18-8 STAINLESS STEEL SOCKET HEAD CAP SCREW)	4
11	SP.2.1.2.C (91841A009_TYPE 18-8 STAINLESS STEEL HEX NUT)	8
12	SP.2.1.3.C (92196A222_18-8 STAINLESS STEEL SOCKET HEAD CAP SCREW)	6
13	SP.2.1.4.C (91841A011_TYPE 18-8 STAINLESS STEEL HEX NUT)	8
14	SP.2.1.5.C (92210A197_18-8 STAINLESS STEEL FLAT-HEAD SCKT CAP SCREW)	4
15	SP.2.1.6.C (92210A202_18-8 STAINLESS STEEL FLAT-HEAD SCKT CAP SCREW)	2
16	SP.2.1.7.C (92240A552_18-8 STAINLESS STEEL HEX HEAD CAP SCREW)	4
17	SP.2.1.8.C (94600A500_SEIZE-RESISTANT PTFE-COATED HEX NUT)	6
18	SP.2.1.9.C (92196A707_18-8 STAINLESS STEEL SOCKET HEAD CAP SCREW)	2
19	SP.2.1.10.C (92210A628_18-8 STAINLESS STEEL FLAT-HEAD SCKT CAP SCREW)	4
20	SP.2.1.11.C (91845A125_TYPE 18-8 STAINLESS STEEL HEX NUT)	4
21	SP.2.1.12.C (92196A133_18-8 STAINLESS STEEL SOCKET HEAD CAP SCREW)	6
22	SP.2.1.13.C (91841A006_TYPE 18-8 STAINLESS STEEL HEX NUT)	2
23	SP.2.1.14.C (92196A385_18-8 STAINLESS STEEL SOCKET HEAD CAP SCREW)	1
24	SP.2.1.15.C (91845A145_TYPE 18-8 STAINLESS STEEL HEX NUT)	1

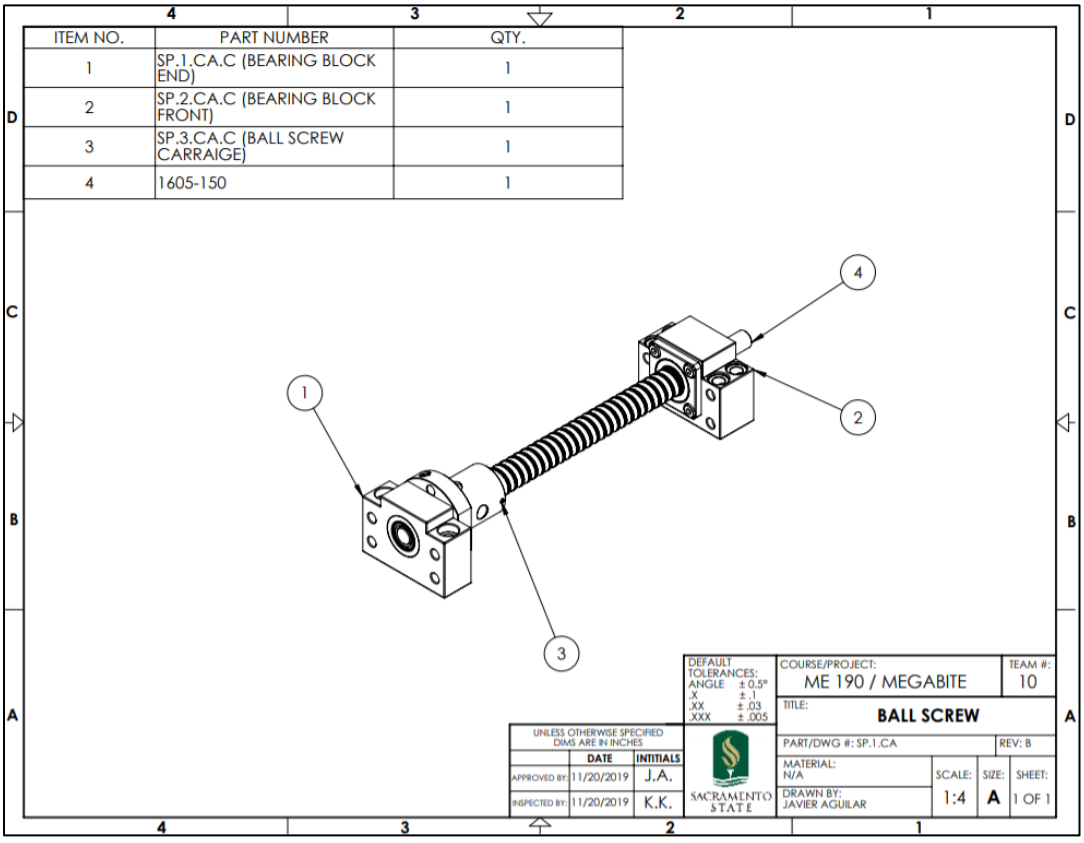
DEFAULT TOLERANCES: ANGLE ± 0.5° .X ± .1 .XX ± .03 .XXX ± .005
UNLESS OTHERWISE SPECIFIED DIMS ARE IN INCHES

	DATE	INITIALS
APPROVED BY:	11/20/2019	J.A.
INSPECTED BY:	11/20/2019	K.K.

SACRAMENTO STATE

COURSE/PROJECT: ME 190 / MegaBite
TEAM #: 10
TITLE: HIP ACTUATORS
PART/DWG #: SP.2.1.A
REV: B
MATERIAL: N/A
SCALE: 1:4
SIZE: A
SHEET: 1 OF 1
DRAWN BY: JAVIER AGUILAR

Figure D1.6. ME Ball Screw



ITEM NO.	PART NUMBER	QTY.
1	SP.1.CA.C (BEARING BLOCK END)	1
2	SP.2.CA.C (BEARING BLOCK FRONT)	1
3	SP.3.CA.C (BALL SCREW CARRAIGE)	1
4	1605-150	1

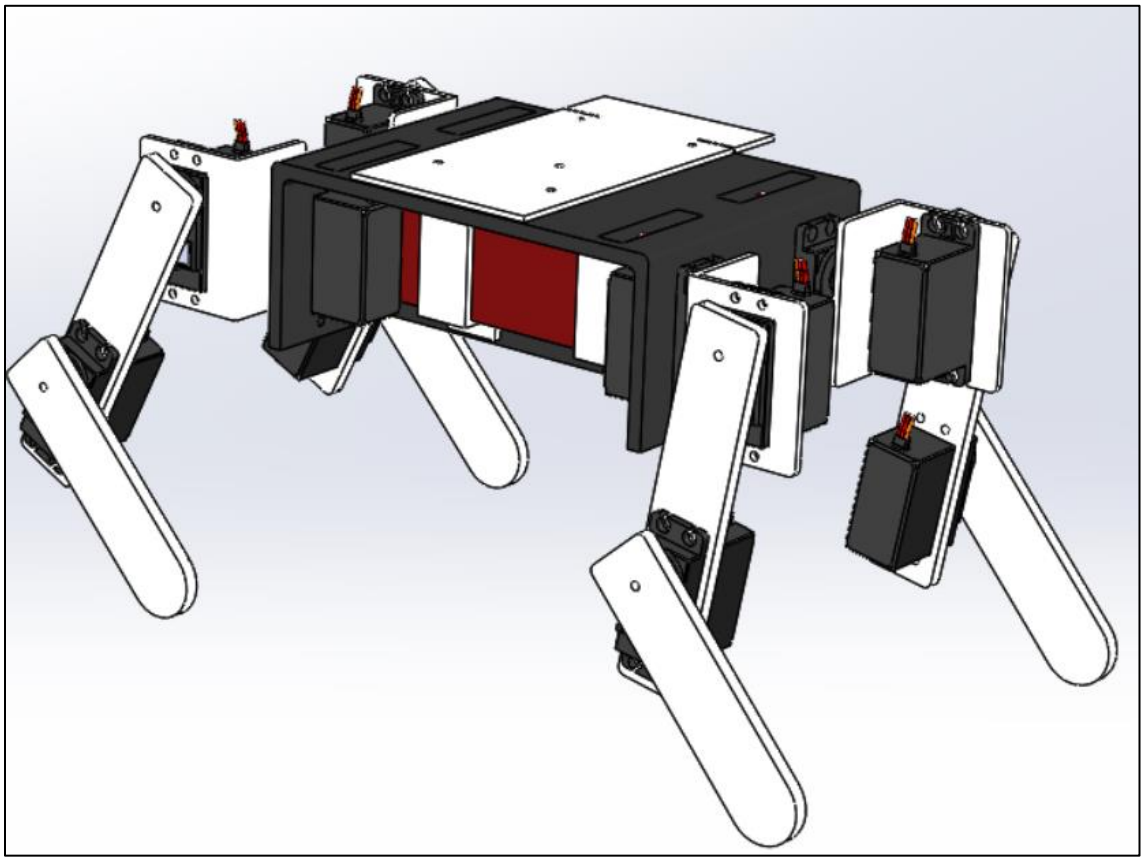
DEFAULT TOLERANCES: ANGLE ± 0.5° .X ± .1 .XX ± .03 .XXX ± .005
UNLESS OTHERWISE SPECIFIED DIMS ARE IN INCHES

	DATE	INITIALS
APPROVED BY:	11/20/2019	J.A.
INSPECTED BY:	11/20/2019	K.K.

SACRAMENTO STATE

COURSE/PROJECT: ME 190 / MEGABITE
TEAM #: 10
TITLE: BALL SCREW
PART/DWG #: SP.1.CA
REV: B
MATERIAL: N/A
SCALE: 1:4
SIZE: A
SHEET: 1 OF 1
DRAWN BY: JAVIER AGUILAR

Figure D1.7. Small-Scale Dog CAD Model



APPENDIX E. VENDOR CONTACTS

Name:	Phone Number:	E-mail:	Website:
ODrive Robotics, INC	N/A	info@odriverobotics.com	https://odriverobotics.com/
ST Microelectronics	N/A	N/A	https://www.st.com/content/st_com/en.html
Nvidia	N/A	N/A	https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/
Fethi Belkhouche	(916) 278-7346	belkhouf@ecs.csus.edu	http://athena.ecs.csus.edu/~belkhouf/
Thomas Douglas	(916) 278-6366	douglas.thomas@csus.edu	N/A
Russ Tatro	(916) 278-4878	rtatro@csus.edu	https://www.csus.edu/indiv/t/tatror/

KRISTIAN JOSUE ORNELAS

(707) 386-0338 ● krisornelas85@gmail.com ● 1929 Oliveglen Ct. Fairfield, CA

OBJECTIVE:

Actively seeking a full-time or internship position in Hardware, Firmware, or Software Engineering

EDUCATION:

Bachelor of Science, Electrical & Electronic Engineering **Concentration:** Analog/Digital & Controls
California State University, Sacramento, CA **Expected:** May 2020
***Overall GPA*:** 3.24 ***Major GPA*:** 3.43

WORK EXPERIENCE:

Electronics Hardware Intern S&C Electric Company May 2019 – August 2019

- Assisted in the design and implementation of a Linked List DMA within a FPGA between a digital signal processor and microprocessor used to increase data transfer efficiency
- Successfully wrote four state machines in Verilog capable of interfacing with external memory via the AHB-Lite bus and verified expected results with ModelSim
- Documented Linked List DMA project with state machine flowcharts and block diagrams
- Presented summer project details with PowerPoint to 20 full-time engineers and managers

Dangerous Goods Specialist Lead Fedex Express July 2015 – November 2017

- Effectively trained and managed six high performing team members in auditing and data entry
- Decreased company spending by developing an efficient auditing process for maximum productivity and efficiency
- Inspected 20 – 70 pieces of hazardous material per shift with four team members before the daily deadline

SKILLS-LANGUAGES, TOOLS, PLATFORMS:

C, C++, Verilog, Python, MATLAB, Quartus Prime, OrCAD PSpice, Attollic TrueStudio, STM32CUBEMX, Libero, Linux (Ubuntu, Debian), VMWare, Visual Studio, ModelSim, Oscilloscope, Logic Analyzer, Signal Generator, FPGA, x86 Assembly, PCB Design, Analog Circuits, Robot Operating System (ROS), RTL Design, GitHub, Embedded Systems

PROJECT EXPERIENCE:

Senior Design Project (in progress):

- *Semi-Autonomous Quadrupedal Robot*: Currently involved in designing and building a Semi-Autonomous Quadrupedal Robot with 7 other team members. Assisting in the design of the Control and Power Systems by implementing kinematic models in code, simultaneous localization and mapping through a monocular camera with ROS and I2C/UART communication protocols for TOF/IMU sensor readings as feedback in Python.

Digital/RTL Design Projects:

- *4-bit Adder*: Designed a full 4-bit adder in Verilog, verified using ModelSim simulator, individually assembled test-bench, and a FPGA
- *Sequence Detector:* Individually, developed a state machine in Quartus using Verilog, built to detect a pre-determined sequence of bits

Computer Interfacing Projects:

- *Wheeled Robot*: In a team of two, assembled a wheeled robot capable of line-following using IR sensors and performing maneuvers specified by the user in C programming language
- *Raspberry Pi Camera*: Successfully wrote a Python script in a Raspberry Pi that would take a picture and request the user to name the file and automatically save it in a pre-determined folder
- *NVIDIA Jetson Nano GUI*: Created a graphical user interface with Python and the OpenCV library that displays the livestream via webcam and the IMU data being read by the NANO using the I2C protocol

AFFILIATIONS/AWARDS:

Dean's Honor List **Fall 2018 – Fall 2019**
IEEE, *Member* **Fall 2018 – Present**
SHPE, *Member* **Spring 2018 – Present**

EDGAR GRANADOS ONATE

• (916) 793-5496 • edgargranados153@gmail.com • github.com/Edgar153

OBJECTIVE:

Actively seeking a full time/internships opportunity in the areas of Hardware, Firmware, or Software Engineering.

EDUCATION:

Bachelor of Science, Computer Engineering ***Expected:*** December 2020

California State University, Sacramento

Overall GPA: 3.32 ***Major GPA:*** 3.41

WORK EXPERIENCE:

Software Development ***Freelance*** ***March 2019 - May 2019***

- Worked with a client to implement a software program that monitored user activity on a web forum and discord text channel. The program would automatically tweet user content from these forums onto the client's business twitter account. Twitter's Java and Python APIs were used to design and create the software.

Client Services Assistant ***Granados Gardening Service*** ***January 2015 – Present***

- Communicate with clients regarding services, quotes, and customer support for my father's business. Other responsibilities include planning out and designing schematics for landscaping projects requested by clients.

SKILLS-LANGUAGES, TOOLS, PLATFORMS:

C, Java, Verilog, Python, JavaScript, Kotlin, Android Studio, Selenium Web Driver, Jenkins, Keil uVision, ARM Assembly, x86 Assembly, Git Version Control, TestNG, Apache Maven, HTML/CSS, Xilinx Vivado Design Suite, Quartus Prime, DOS, Windows (XP, Vista, 8.1, 10), MS-DOS, UNIX, Linux (Ubuntu, Debian), VMWare, OpenCV, Machine Vision, Oracle SQL, PHP, Eclipse IDE, Embedded Systems, Robotics, Logic Analyzer.

CURRENT AND RELEVANT PROJECTS:

Senior Design Project:

- *Semi-Autonomous Quadrupedal Robot*: Designed the Control and Feedback System of a semiautonomous quadrupedal robot. The team consisted of 4 Mechanical Engineer (ME), 2 Electrical Engineer (EE), and 2 Computer Engineer (CpE) students. Interfaced several sensor modules and assisted in writing the walking algorithm for the robot. Embedded C and Python was used to write the control algorithm of the robotic system as well as several communication protocols to interface modules together.

Java Projects:

- *Shopify Automation Tool*: Designed a web scraping application that would allow users to automatically search and checkout items out of various Shopify seller websites. Selenium WebDriver, TestNG, Apache Maven, Git Version Control, and Jenkins was used to deploy this application.
- *Client Billing System:* Created and coded a GUI program, using the java.swing package, for my father's Gardening business. The program allowed users to add/delete client profiles, input client information, and provided the user with an automated invoice-making service that printed out invoices.
- *Real-Time Cryptocurrency Monitoring System:* Implemented a cryptocurrency monitoring program using Java Sockets and the java.net package. The program would retrieve information about cryptocurrency coins from several finance websites and display them onto a console window. The program was utilized by 16+ users

Computer Hardware and Machine Vision Projects:

- *Direct Mapped Cache Design*: In Verilog, designed and simulated a cache controller module that utilized the direct mapping scheme to store data onto cache blocks. The controller would be able to interface between a CPU and Main Memory to perform read or write operations.
- *PCI Bus Arbiter:* In Verilog, designed and simulated a PCI Bus Arbiter that performed bus arbitration among multiple master devices on a PCI Bus. The bus arbiter utilized the Round-Robin Priority Scheme to designate the PCI Bus to the appropriate master device.
- *Object Detection and Tracking Robot:* Designed a robot that would be able to detect and follow a spherical object. Python and OpenCV was used for image processing and controlling the robotic system.

Android App Development:

- *Remote Video Surveillance Application*: Currently developing a mobile surveillance application for the Android platform. The app will allow users to monitor their home surveillance systems from their phone using the system's IP Address and network protocols such as DDNS and Port Forwarding.

AWARDS/CLUBS:

Deans Honor List	***Spring 2017 – Spring 2019***
Engineering and Computer Science Scholarship	***Fall 2019***
Web Development Club, *Member*	***Fall 2018 – Spring 2020***
ACM, *Member*	***Spring 2019 - Present***
SHPE, *Member*	***Fall 2019 – Present***

Alfred W. Martinez III

Electrical & Electronics Engineer

7248 Sunwood Way
Sacramento, CA
(916) 571 - 4991
alfredmartinez555@gmail.com
www.linkedin.com/in/AlfredMartinezIIIEEE

Objective:

Actively seeking a full time/internships opportunity in the areas of Hardware, Firmware, or Software Engineering.

EDUCATION

Bachelor of Science, Electrical & Electronic Engineering California State University, Sacramento, CA
Expected Graduation: *May 2020*
Concentration: *Controls / Embedded Systems*
Major GPA: *3.461*
Overall GPA: *3.227*

SKILLS/TOOLS/PLATFORMS

Embedded Systems, C / C++, Python, Java, ARM Assembly, x86 Assembly, Linux, MATLAB, Quartus Prime, Verilog & VHDL, PSpice, ModelSim, Advanced Design System, Visual Studios, VMWare, PCB Design, Atollic TrueStudio, Oscilloscope, Logic Analyzer, Signal Generator, STM32CubeMX, PyCharm, Arduino, Robotics, Machine Vision, OpenCV, NumPy, Windows, Robot Operating System (ROS).

COURSEWORK

Circuit & Network Analysis, Controls & Feedback Theory, Transmission Line Theory, PCB Design, Circuit Design, Simulating, and Testing, Electromechanical Conversion, Robotics, Machine Vision, Communication Systems, Microprocessor Architecture and Programming, Product Design Management

WORK EXPERIENCE

Delivery Driver, Domino's Pizza, Sacramento, CA
February 2017 - Now
Officer Candidate, United States Marine Corps, USA
December 2015 - Now
Math & Science Tutor (Grades 9-12), Freelance / Private
December 2015 - May 2018
Line Cook / Oven Section Leader, Pete's Restaurant & Brew, Sacramento, CA
January 2016 - July 2018

PROJECTS

Robotic Dog (Mini & Large), Senior Design

AUGUST 2019 - MAY 2020

ARM Cortex M4: Developed a bare-metal robot dog using an STM32 Arm Microcontroller. At the base, clocks, timers, and pulse width modulation registers were all assigned to control 12 robotic joints in real time.

PCB Prototyping & Wire Management: Developed a prototyping board to house all components and connections to facilitate simplistic debugging and prevent damage to components. Learned proper soldering and PCB prototyping techniques.

Robotic Kinematics: Used trigonometry and geometry to control the position of the 4 legs of the robot using angular control of the joints.

Remote Monitored Fish Tank

JANUARY 2019 - JUNE 2019

GUI & Embedded System Connection: Worked on a user interface using an online Apache Web Server that could display data from embedded sensors in a fish tank remotely. This included temperature, humidity, water level, and security sensors as well as a motor to control the feeding rate of fish food.

Firefighter Camera Assistant

JANUARY 2020 - MAY 2020

Machine Vision / OpenCV / Python: By using footage from a firefighter's helmet camera, a system was developed to detect edges of furniture/rooms/hallways in a smoky or dark room and relay that data to a small screen placed inside the helmet.

Verilog/VHDL Designs & Simulations

AUGUST 2019 - MAY 2020

Developed and Simulated four-bit adders, multipliers, and multiplexers. Also, developed several finite state machines, flip-flops, and a 4-bit microcontroller in Verilog.

Marcus Huston

8146 Big Sky Drive, Antelope, CA 95843 | (916) 616-7927 | marcus.huston96@gmail.com | Computer Engineering Student

Education

Bachelor of Science in Computer Engineering — Spring 2020(est.)
Minors: Applied Mathematics & Studio Art — Overall GPA: 3.20 / 4.00
California State University, Sacramento

Member of Tau Beta Pi – Engineering Honors Society - achieved the top 12.5% of your Junior class or 20% of your Senior class. Tau Beta Pi is the only engineering honor society representing the entire engineering profession.

Professional Profile

I am an innovative individual looking to collaborate in diverse environments in order to immerse / enhance my knowledge, problem-solving skills, and project management experience to challenge me to grow and learn new languages / emerging technologies.

Skills Summary

- **Java, C, Python, x86 Assembly, Verilog, Microsoft Office and Adobe Suite.**
- **Experience working with Oscilloscopes, Microcontrollers, and FPGA.**
- **Good analytical, communication and technical writing skills**

Relevant Courses

CMOS and VLSI	**Database Management Systems**	**Operating System Principles**	**Advanced Math Science & Engineering**
Computer Networks & Internets	Advanced Computer Organization		
Probability & Random Signals	Computer Hardware Design	Electronics	
Data Structures & Algorithm Analysis	Advanced Logic Design	Computer Interfacing	
Discrete Structures	System Programming Unix	Network Analysis Signals & Systems	

Work Experience

Law Office of Marcus, Regalado, & Marcus, LLP
August 2014 – Present

- Internal and external communication
- Data entry / Calendar statutes
- Client intake process coordinator
- Utilization review manager
- Assisting the attorneys/paralegals
- Assists in the development of project scope and project management documentation using MS Office.
- Trained incoming employees in preparation for their job
- Reports on analytics and project progress
- Installed software and hardware on employee machines

Project / Leadership Experience

Hornet Hyperloop
International Student competition organized by SpaceX and the Boring Company.
Controls Team

- Designed, analyzed, and debugged various software and systems
- Optimized design using simulations and analysis
- Use of CAN bus communications for main pod controls as well as data acquisition
- Ability to work with other engineers, collaborate, and test ideas
- Ability to solve technical problems
- Code: C Programming Language

American Sign Language Glove
Student project to create a bridge between ASL and English by converting signed data and displaying it as letters.
October 2018 – December 2018

- Organized and directed weekly team meetings and optimized group effectiveness.
- Created schematics to optimize design and transition into simulation and analysis
- Integrated embedded systems to communicate and increase speed
- Ability to quickly adapt, debug, and solve technical problems
- Code: C Programming Language
- Hardware: Raspberry Pi, Analog to Digital Converter, Accelerometer, Flex Sensors, etc.

Magnetic Levitation Quadcopter Drone
Student research project to design, create, and build a drone that levitates based on rpm against the magnetic fields
March 2019 – Present

- Troubleshooting skills
- Performed background research and study for the solidification processing of materials in magnetic fields.
- Researched and analyzed to decrease friction
- Designed and analyzed a structure capable of holding neodymium N52 grade magnets for solidification process.
- Code: C Programming Language