ELECTRICAL AND
COMPUTER ENGINEERING
COLORADO STATE UNIVERSITY

Rambots

Final Report
Fall 2021

Senior Design Team Members:

Electrical Engineering:
Jared Davis, Ritwik Vadapally, Craig Olson, Oscar Coronado Rosales

Computer Engineering:
Kristopher Alquist, Devin Pohl

Other Team Members:

Junior VIP Members:
Alex Kolodzik, Gwyndolyn Tari

Advisors:

Faculty Advisor and Supervisor:
Olivera Notaros

Industry Advisors:
David Farrell
Ian Bernstein

Engineer In Residence:
Eli Scott

Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado 80523

Approved by: ____Olivera Notaros_____
Date of approval: ___12/10/21_________

Abstract

The RamBots Senior Design team is tackling the problem of making advanced quadrupedal robotics available at an affordable price. In industry, available and sufficiently advanced solutions are inaccessible due to cost constraints, while affordable alternatives are significantly lacking in software-based capabilities. Seeking to build off of moderately developed open source projects, the team is aiming to be efficient with duplication of effort, extending an existing platform to be more useful. The end-goal of this project is to develop a platform as advanced as what is seen in industry, but at a price which is accessible to a much wider audience.

The team has made strides towards this goal by extending an existing open source quadrupedal robotics platform. While this platform, as it was previously published, lacked notable software development, the team has prepared a development environment to solve this problem. With work on machine learning, advanced embedded system integration, and electronics management, a foundation has been created to allow for significant improvement.

Non-integrated work from this semester indicates promising future results. Successes have been made in object detection on embedded systems, machine learning based balance and fluid motion, and tooling for microcontroller programming. Future work will include integration of these topics onto a single platform.

TABLE OF CONTENTS

Introduction

Project Overview

The ECE Outreach team is in the process of designing and building a ram-themed quadrupedal robot. Controlled by multiple microcontrollers, the robotic ram will include actuators, cameras, and a nearly entirely 3D printed chassis. The robotic ram will be based off of an existing open source robotics project, with effort being focused on improving electronics and software. The purpose of the robotic ram is to create a low cost, fast set-up time, and open source robot and will be used to attract middle and high school students into the field of Electrical and Computer engineering as well as robotics. The physical robot can be used to express electrical engineering concepts while the software and code can be used to express and show off the computer engineering field. Additionally, as this robot is open source, emphasis will be placed on its use as a platform for future development and education by auxiliary parties.

Why This Project Is Important

With the proliferation of robotic interest being seen today, there are a lot of problems for those who seek to participate. The biggest being affordable quality platforms available for a wide variety of projects and disciplines. What this project aims to achieve is to make a high quality, affordable, and open-source robotics platform with the goal to spark interest in STEMM and promote the use of robotics in any discipline.

Involvement with Outreach

The Electrical and Computer Engineering (ECE) Outreach team from Colorado State University (CSU) is a dedicated group of students who work to raise awareness, understanding, and interest in ECE topics through various demonstrations, lessons, and workshops. Working with middle and high schools in northern Colorado, ECE Outreach uses informative and entertaining projects to introduce students to the STEMM field, hoping to foster a greater enthusiasm for engineering.

Our Senior Design team has been and will continue to work closely with the ECE Outreach team. As one of our main project goals is to make the final design expressly affordable, and another is to extend existing open source platforms in an easily accessible and extensible manner, our project fits well within Outreach. While involvement has been limited so far due to startup development, future plans involve showcasing RamBots at Outreach events to inspire future electrical and computer engineers.

Early involvement with Outreach has yielded significant advantages as our project has begun. In addition to additional up-front funding, we have received important networking with industry members. These industry members have given great advice on avoiding time-sinks early in the design process which we were unlikely to receive if not working with Outreach. Additionally, other networking from Outreach has allowed us to source parts, services, and consulting at free or reduced rates.

State of Industry

As mentioned previously in this report, there has been a large amount of prior development in the field. Boston Dynamics, in particular, is a very popular example of work highly related to our project. But, of course, Boston Dynamics is not alone in this regard. Several industrial robotics companies, such as Wolf Robotics, are experimenting in fluid motion and higher level decision making. The problem with this work, however, is that it is all closed source. Therefore, specifics from industry are unable to help us in this regard, and sadly can be ignored. General concepts, luckily, can be implemented -- our EIR and faculty advisors working in industry have been invaluable in this regard.

Open Source Alternatives

With industry implementations off the table for future development, and a lack of mechanical engineering students able to help us create a design from scratch, focus has shifted to improving existing, open source platforms. Preliminary project research focused heavily on this topic, and finding examples of existing open source projects. The first existing design, interestingly, came from a CSU ECE Open Option Project, pioneered by one of the VIP students on our team. While we ultimately decided not to expand on this design in favor of more developed alternatives, this was a useful point of research. Aside from this, several other projects published on GitHub were considered. One such project was a bipedal child-sized robot. This was discounted due to complexity; while we are confident in our ability to get the base model working, we know that without mechanical engineering students, any modifications or repairs would take such a long time that we would be wasting our time and efforts. Choosing a more simple model was a step in the right direction in this regard. With this in mind, we reviewed several open source robots which have very simple designs. The problem we found with overly simple designs is the lack of fun factor; as an end goal of this project is to use it for Outreach demos and workshops, it is useful to have something that will get future engineers of all levels excited. A good balance was struck with a quadruped robot; the extract project we decided to extend is explained in the next section.

OpenDog Platform

With the goal of a quadrupedal robot decided, the team began research on existing open source implementations. Eventually, the openDog project by James Bruton was decided on. Several factors went into this decision, including the extensive written and video documentation on this project. It was clear that significant development has gone into the hardware of this robot, but to date the software is lacking. This, of course, is the perfect target for our project; we aim to improve software without much work in hardware. At the beginning of the semester, the openDog V2 project was available. However, we knew that a V3 was on the horizon. For this reason, we strategically delayed specific portions of development throughout the semester, printing only parts of the V2 robot for small testing. Due to drastically increased lead time on critical parts thanks to the global parts shortage, this ended up working quite well for our development -- the schematics for openDog V3 have just recently been released just before our critical parts are arriving. This will give us time for printing the entire chassis as critical parts finish arriving and non-critical parts are ordered.

Remainder of this Document

The rest of this report will include aspects important to prior work on this project. Beginning with high-level project objectives, the project will be explained in more detail. Then, design considerations will be visited, with analysis of project risk and time management. Following this, specifics of this semester's completed work will be detailed. Standards for contributing to open source projects will then be discussed. Conclusions on work completed will then be presented, along with considerations for future work. The back matter of this writeup includes notes on budget and fundraising, evolution of the project plan, and acknowledgements.

Objectives and Constraints

Background Research

When it comes to robotics, many groups and individuals have designed and produced quadruped and even bipedal robots. Boston Dynamics is a great example of a group that has designed and developed amazing mechanics and motion of bipedal and quadrupedal robots. Other individuals have designed their own robots, like James Bruton who created the openDog V3 quadruped robot. ABB and FANUC produce industrial robots that have many applications from painting, welding, moving objects, etc.. These robots come in all shapes and sizes and can have different applications.

While industrial robots and the robots that Boston Dynamics have made are pricey and large, individuals can make smaller economic versions of robots that use smaller components like an Arduino, Raspberry Pi 4B, SG90 servo motors, AA batteries, etc.

Most robots, whether they are quadrupedal or bipedal, will have the following components: a body/frame, a controller, sensors, actuators, a power system, and sometimes a user interface. Some robots can be remote controlled like those made by ABB and FANUC, others will use controllers like an Arduino Uno, Raspberry Pi, even an FPGA board. Robot sensors include ultrasonic, sound, temperature, contact, proximity, distance, or even tilt sensors. Actuators include servo motors, stepper motors, DC motors, etc.. Robot power systems can range from AA batteries to using DC-DC power converters to even using wall plug power supplies. ABB and FANUC have custom controllers with user interfaces that can allow you to manually control the robot.

With all this in mind it's important to state some problems when it comes to the robotics community. While it may be simple enough to go out and buy an Arduino Uno/Mega, 12 SG90 servo motors, a servo shield and AA batteries, programming the robot is what can prevent most people from stepping away from their project. Modern robots can be programmed in numerous languages from Arduino C++, Python, or RAPID, configuring and understanding how hardware works can be difficult for even the most experienced programmers.

High Level Objectives

This project has many objectives ranging in various categories and importance.

Appearance

Although the exact appearance has yet to be solidified, it was determined to be the size of a large canine: approximately 1 meter long by 0.5 meters high and weighing no more than 27.5 kilograms. The outward appearance will use curved edges with a chassis hiding all or most electrical components and wires. In addition, at the front of the robot there will be a ram head inspired by Colorado State University's Cam the Ram.

Functionality

The main objective is to have full quadrupedal motion. This entails, at the minimum, walking at the same pace as a canine of its size. Higher degrees of articulation such as running, climbing, and traversing various terrain are objectives that build off of each other as progress is made.

The other functionalities will be regarding its computer vision capabilities. This will primarily be to detect faces of its owners, and simple object recognition using machine learning. From there, more complicated abilities will be introduced starting with identifying how many people are in a crowd, and then onto more sophisticated object detection.

Philosophy

The goal behind the creation of the RAMbot is to provide a high-quality, open-source alternative to leading computing platforms in order to spark more interest in engineering and STEMM as a whole. The objective is to make every aspect of this project as transparent as possible. This includes a full parts list, all files regarding 3D printed materials, well documented code, and, if necessary, all custom PCBs. This may also include video or written tutorials in the construction of the RAMbot.

Low Level Objectives

Below is a list of the requirements and specifications decided on by the team:

- Safe and Robust 3d printed materials(PETG)
- Able to walk and autonomous movement
- Uses the data from the sensors to detect humans, objects, hurdles etc;
- Open source for future generations of the project
- Able to take it to Football/ Volleyball games and High Schools for Outreach
- Power
    - 22.2V 6000mAh Battery for motors
    - 11.1V 2200mAh battery to support the brain functions

Implementation Details

Below is a list of the implementation details for the current iteration of the project:

- Hardware:
    - Size(tentative): length - 39”, Height - 19”, Width - 15”
    - Weight - 20 Kilograms
    - Build Material: PETG (high heat resistant due to temperature of motors)
    - 12 90V rated Motors
    - 6 Odrive 3.6 Motor Controllers
    - Sensors
        - Camera - Facial Detection
        - MPU 6050 Accelerometer - To make sure the Bot balances itself

    - Safety
        - Curved edges
        - Controller - For manual override and to turn on/off
        - Electric sealing - To prevent getting shocked
    - ESP32 - To act as a body controller and I/O board
    - RaspberryPi - General connector and Machine Learning base
    - Intel Neural Compute Stick II: Machine Learning Accelerator
- Software:
    - Raspberry Pi: TensorFlow, Keras, OpenCV, OpenVINO
    - ESP32: Stock RTOS, Rust for general programming, LLVM C for dropdown

Design Considerations

FMEA Table

In project design a FMEA table is used to assess potential product risks and it helps to identify potential points of failure in the project that you're working on. The FEMA table for our project can be found in Table 1. FMEA stands for Failure mode and effects analysis. Creating a FMEA table enables us to better understand what can go wrong in our project and how long it can delay the project. Whether it's a part breaking or something not arriving on time. Correctly and accurately making a FMEA table is an instrumental part of the project design. Without a properly made FMEA table we wouldn't know how much time to budget if we encounter any issues. For example, if a sensor malfunctions it could take a long time for us to be able to diagnose the issue. Since depending on the sensor or part we will have to disassemble the robot just to get to it and remove it safely. That's just to get to the sense, after that we will have to determine if it's a broken sensor or something else is broken. Due to the current shipping issues, it could be weeks or months before we could even get a replacement part. That is why it's important to work out everything you can possibly think of going wrong during the project and incorporate that into the FMEA table. Without the table you wouldn't be able to properly assess how many delays you can encounter or be able to plan ahead so that we reduce the time we are affected by having a broken part or delay. Making the FMEA table helped us realize how beneficial it is to order spare parts for our project in case something does go wrong since we can't afford to wait a couple of months to get a new part since that will essentially put our project at a standstill. Making the FMEA table also got our team to think of issues that we wouldn't have thought about if we didn't have to make the table. When we first started making the schedule, we didn't think too much about how one little thing going wrong could set our project back a couple of weeks. For example, we accidently ordered the wrong sized 3D printer filament. This set us back around a week since we had to use the idea 2 product lab to print some of the parts.

The most difficult thing about the FMEA table was to come up with failures since we are a first-year project and hadn't built the robot yet. We first came up with some basic examples of issues that we could have and how it would impact the project. As we've worked on the project, we have made slight changes to the FMEA table. We've mostly adjusted the scoring system once we got more experience with how some delays can affect us. The severity, frequency the issue can occur, and detectability scores were all judged on a 1-10 scale. One being the least severe or most detectable and ten being the most severe or least detectable. The final score is called the risk priority number. The RPN is calculated by multiplying the score for severity, frequency of the issue, and detectability together. The lower the final RPN is the better. If any of the potential failures has a high RPN value it needs to be assessed and we would need to come up with a solution to make sure that it won't happen or to make the issue more detectable.

Process Step	Potential Failure Mode	Potential Failure Effect	SEV1	Potential Causes	OCC2	Current Process Controls	DET3	RPN4	Action Recommended
What is the step?	In what ways can it go wrong?	What is the impact on the customer if the failure mode is not prevented or corrected?	How severe is the effect on the customer?	What causes the step to go wrong (i.e. how could the failure mode occur)?	How frequently is the cause likely to occur?	What are the existing controls that either prevent the failure mode from occurring or detect it should it occur?	How probable is detection of the failure mode or its cause?	Risk priority number calculated as SEV x OCC x DET	What are the actions for reducing the occurrence of the causes or for improving its detection? Provide actions on high RPNs and on severity ratings of 9 or 10
Powering Up	It may not power on	Non working robot	8	• The robot was not properly shut down or charged. • Power system failure.	2		1	11	We will have to write the proper shutdown, start up and charging procedures in the manual.
	Slow startup	It costs the company time	3	• Inefficient code • A slow computer	3		2	8	We will make sure that the computer is as quick as it can be while still being financially viable. We will also try our best to make the code efficient.
Movement	Servo Didn't move	The robot can't complete the desired movements	6	• Wiring failure • Programing failure	2		1	9	We will double check all of the code and the wire connections before the robot is complete and research more durable components.
	Robot won't balance	The robot can't move successfully	7	• Programing failure	3		1	11	We will be able to push updated code to the robot to fix errors in the old code.
	Sensor malfunctioning	The robot can't sense its environment	6	• Broken sensor	1		2	9	We will test every sensor before the robot is complete and program an alert or alarm to go off if this happens.

Table 1: FMEA Table

Project Risk

In large projects like this, significant risk is present. While this may include risk of harm to the user, it may also include risk of harm to the project, or even risk of delay in deployment. In order to better handle risk in our project, we have identified several potential sources of risk in deployment. These sources are presented below in Table 2.

#	Risk Event	Probability /100	Impact, weeks	Score, hours	Effect	Risk Mitigation Plan	Person responsible for implementing control
1	Parts arrive late	60	1 week per part	50	Delay the project	We will have extra parts to cover any issues we have with shipping.	Jared Davis
2	Wrong color 3d printing filament	5	1	5	Delay the project or The customer gets a different color robot	Make sure we have a sample of the desired filament and the shipper delivers the correct color.	Ritwik Vadapally
3	Short circuit	10	Varies	10	It could delay the project by a few minutes or cause multiple devices to break and cost a lot of money.	We will make sure that all the wires are properly terminated.	Craig Olson
4	Personal injury	10	Varies	60	It can injure one of our teammates.	Make sure everyone involved when testing or working on the robot knows what their task is and how to react incase of an emergency.	Devin Pohl
5	Incorrectly calculated power requirements	15	1	55	The robot won't function correctly.	Have another teammate double check the power calculations.	Craig Olson
				Total risk :	180		

Table 2: Risk Analysis

Work Completed

Hardware

One of the largest challenges faced by the team during planning was the unanticipated lack of mechanical engineering students. This was circumvented by outsourcing expertise on mechanical engineering to existing open source designs. Significant research was performed early in the project; the final decision was to use James Bruton's openDog platform. At the beginning of the semester, version two of this platform was available, shown in Figure 1. In early December, version three was released, shown in Figure 2. With critical parts shared between versions, the team decided to pivot towards this version. Both of these implementations, however, lack on the electronics and hardware side; the bulk of the project work was dedicated to these two areas.

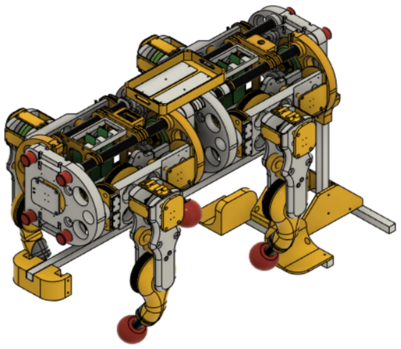

Figure 1: openDog V2 [1]

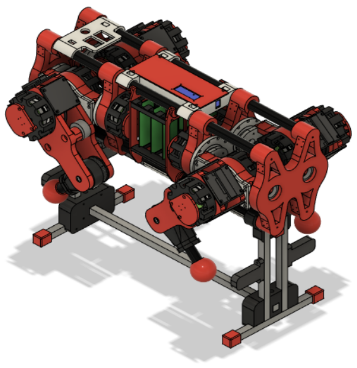

Figure 2: openDog V3 [2]

Electronics

While we spent most of this semester tackling the challenge of getting parts ordered, we acquired the motor drives for the motors, specifically Motor Drive 5.6V 3.6. When they arrived they weren't soldered or assembled so one of the tasks we overcame was soldering all 8 Motor Drives and testing them for future objectives. We also decided on eagle power l8308 90V brushless motors. Which are clones of the original motor designed around this project, but due to shortages this was the only option available. Each Odrive motor drive[4] connects to two motors and two encoders that sends information about position back to the teensy brain. The exact schematic can be seen in Figure 3 for reference of a single motor drive connected to the serial port.

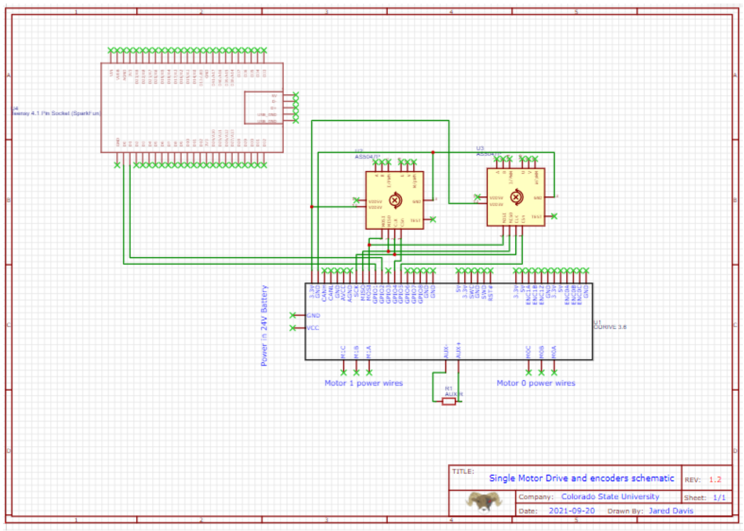

Figure 3: Wiring Diagram. Pinout taken from [7]

3D Printing

Since the model for openDogV3 is almost completely 3D printable we spent the majority of this semester getting all the parts organized and ready for printing. While we tried to 3D print the robot during the semester we ran into the problem of where to 3D print. We did most of our 3d printing testing in the senior design lab but those printers occasionally had problems whether it's issues connecting to them or bed leveling issues. The other option that we used this semester was the I2P lab in the basement of the engineering building. While the lab was accessible, it was unreliable and hard to use as well since there are 14 printers being used by students in the ENGR 300 Course, Non-Engineering students, and other engineering students who paid the lab fee. The I2P lab had time slots being filled up constantly and with the limited printers. We were only able to successfully 3D print the stand for the robot after attempting to get the needed time slots for over three weeks. At this rate we would never be able to finish 3d printing the 233 parts needed for the robot so we looked for other options.

We met with Ty Thourot, a junior who owns a 3D printing and design company called 3Drogue. He has agreed to use his company to 3D print the entire robot in 40% infil quality using PETG filament. He will be 3D printing the model of openDogV3 in parts as per how much we can pay with through a P-Card. He will create small invoice's and we will use a P-Card to pay those invoices since the total estimated price to 3D print the entire model of openDogV3 is around $2,500-3,000. We have been in contact with him since October of this year 2021 and will continue to be in contact with him this Winter break and next semester in order to finish 3D printing and advance to other objectives.

Software

Firmware

In regards to interfacing with sensors and motors, a large amount of development has occurred. While the default idea was to reuse the Raspberry Pi already employed on the project for peripherals, issues with latency and bandwidth were quickly discovered. Therefore, it was decided to use a dedicated microcontroller for latency-sensitive I/O operations. Research indicated two options in this regard: ESP32 and Teensy. Both boards were ordered, and preliminary experiments conducted. Due to the ESP32's lower cost, better runtime compatibility, and superior community support, it was chosen to move forward in the project.

As previously mentioned, the ESP32 has extensive community support. So much so that the Rust programming language has it as an unofficial STD target. This is important because compatibility is seamless with the ESP32's RTOS. This allows for operations such as dynamic memory allocation, file system access, wifi and bluetooth operation, multithreading, and a myriad of other features. These all are trivially easy to use when employing Rust on the ESP32 – a programmer can treat the board exactly like a standard desktop computer. This is in contrast to using the Arduino IDE to program, as is the standard method, where every nontrivial operation requires an explicit and poorly explained dependency. Even on other boards which can handle Rust, workflow isn't as seamless; more work is required by the end programmer in regards to dependencies.

All that said, employing Rust on the ESP32 is not a task free from problems. The first problem is in the build system. Because the Xtensa architecture, used by all Espressif chips including the ESP32, is not an official LLVM target, community builds of Clang LLVM and the Rust compiler are required. These can be set up automatically, but take some additional up-front work. The second problem is in validation: because running Rust on the ESP32 is not officially supported, validation of functionality needs to be performed in order to ensure that everything is running well within spec. This second point was a source of significant effort; all functionality of the board was meticulously validated according to manufacturer specifications. GPIO, UART, SPI, I2C, I2S, WiFi, Bluetooth, Ethernet, and other miscellaneous technologies were examined using tools such as oscilloscopes and logic analyzers to ensure feature parity with a solution based on the Arduino IDE approach.

Machine Learning

Object Detection

The primary focus of machine learning on this project is to employ computer vision. Thus far, object detection on the Raspberry Pi has been achieved. It can recognize people and most common objects with fair certainty, and what it sees can be reported to the user. A demonstration of the current capabilities of the ML algorithm can be seen in Figure 4. The user can see what the ML algorithm is inferencing on the screen, where a bounding box is drawn around an object it detects, and the percentage it believes that object to be. However, the current state of the object detection system is slower than what can be considered tolerable. Right now, it is inference at about 4-5 frames per second, and this creates a bad user experience. The model being used, YOLOv5s[6], is trained with the MS COCO[2] dataset, and is currently quantized into a TensorFlow Lite version model. This means the model is smaller in size, and less cumbersome on the Raspberry Pi's CPU. This is very important, because before when using the unquantized model, it was inferring at about 1-2 frames per second. In addition, the first model ever tested for object detection was YOLOv5, which is the full sized version, and this simply would not run on the Raspberry Pi. Currently, all the software dependencies for object detection are resolved, meaning testing can be done more efficiently. The most obvious solution to resolve the problem of low frame rate would be to use hardware acceleration. The Raspberry Pi has a feeble onboard GPU, which would not help inference time. Therefore, other hardware accelerators are needed, most notably USB devices that are essentially TPU's. So far, the Intel Neural Compute Stick II is what we attempted to implement. This implementation has not been successful, and it has to do with outdated API support and the depreciation of many software packages that the compute stick relies on. There are many options we are currently in the process of researching in order to overcome these difficulties. Finally, the camera used is a Logitech HD Webcam C525, and this was chosen because the Raspberry Pi cameras may cause problems with how they need to be attached to the main board via a ribbon cable. This webcam supplies ample cable length and video quality.

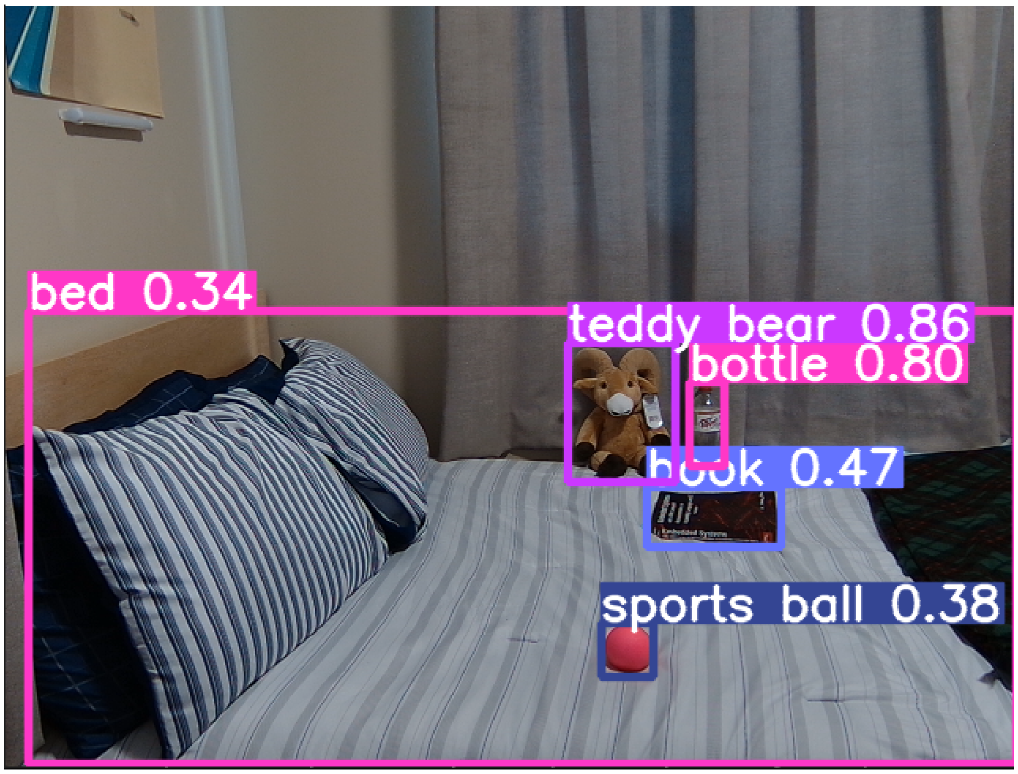

Figure 4: Object Detection Demonstration

Fluid Motion

Another aspect of machine learning we are looking into is the motion of the dog. Due to the lack of a physical robot this semester it was extremely difficult to work on this aspect of the project. We did a lot of research on self balancing robots and robots that could pick themselves up or stop themselves from falling down. To work on a machine learning model and test out different models on different surfaces we decided to order Waveshare Pippy.

Figure 5: Waveshare Pippy [8]

Waveshare Pippy, depicted in Figure 5, is a small robot dog kit that runs on a Raspberry Pi and utilizes an ultrasonic sensor, a camera and motors with in-built gyroscopes to walk. The ultrasonic sensor acts as a radar to detect the distance of different objects from the dog, we use a Raspberry Pi camera which can be used for object/ motion detection. Since the motors have inbuilt gyroscopes, and each leg has 2 motors the dog is able to balance itself and is able to comfortably walk on different types of surface. The implementation of this can be seen in Figure 6. Due to its design and capabilities, Pippy is a perfect test base for our project as it is much harder to make the big dog walk right away. This model is a perfect fit for all of our software tests as it includes a camera which can be used for computer vision as stated earlier. We have the body of the dog built and just have to make sure the models created by our team are able to work perfectly with this model and once we are sure the software works we can deploy the software on our bigger project.

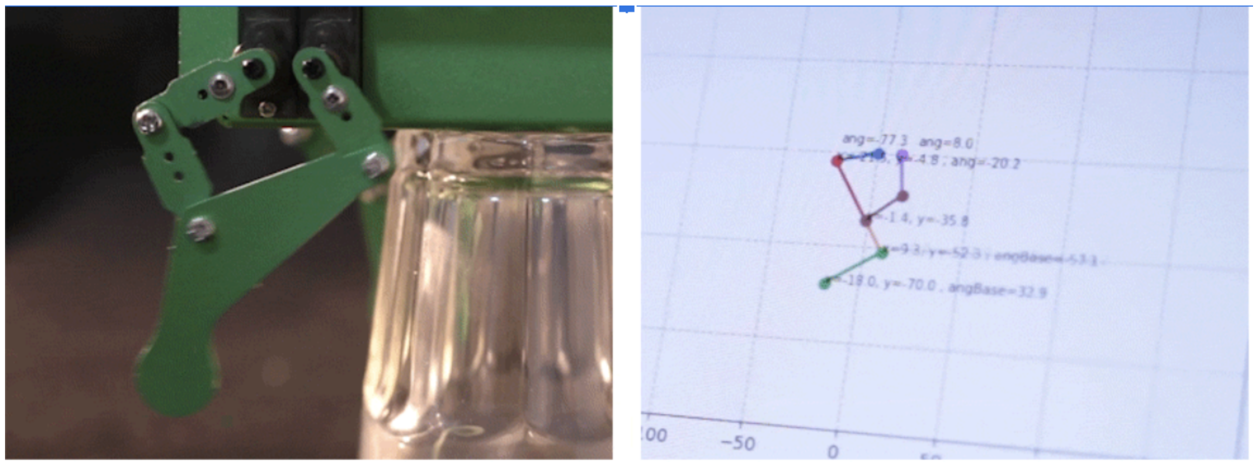

Figure 6: Motion simulation of the legs [8]

Standards for Open Source

Although this project is fairly open ended, there are several standards by which the team needs to follow. This is because work contributed will be open source, or available to the general public. Considering that a future goal of this project is to be a platform for further development, we need to be careful not to duplicate work. In order to achieve this, emphasis should be placed on documentation, reproducibility, and notation of findings.

The first point to consider, documentation, is one which has been extensively researched in the context of open source. Not only will it help in the goal of using this project as a platform for additional development, but it will also aid future senior design students in picking up this project. Luckily, several tools are available which make this job a lot easier. For the microcontroller side of this project, most of the code will be written in Rust, which has extensive tooling for automatically creating documentation from source code. This tool is called Rustdoc[5], and will be an invaluable resource. Other programming languages used on this project have similar tools to aid in this task. In regards to electronics design, the tools used here have a similar story. EasyEDA, the PCB design software used by the team, is built in such a way that notating and exporting schematics for use in documentation is extremely easy. Furthermore, parts of the project which would usually receive little attention in regards to documentation, such as budget, are being meticulously recorded. The intent behind this decision is that users who wish to reproduce our work should need to do as little setup as possible.

The second point to consider, reproducibility, is an important step in software development. Many technologies require extensive setup before being able to run. This falls counter to our design philosophy; the more time a user needs to spend setting up a development environment, the less they can spend tinkering with the project, and the less likely they are to be interested. For this reason, many steps have been taken towards automatic setup, and notation of system requirements. When working in open source, this is seen as a mark of quality; we intend to do well in this regard so our work is adopted by others.

The final point to consider, notation of findings, will be important for those who need to dive deep into development. This is primarily focused for future senior design students, as they are likely to run into many of the same mistakes we have made. Notating what has gone wrong in addition to what has gone right with our project will go a long way in removing duplication of effort. Furthermore, notating findings will be exceedingly useful to an auxiliary party wishing to make large modifications to the project. For example, if a future Outreach member wishes to be able to program the robot using the Scratch programming language for a Middle School workshop, it will save a large chunk of time if anecdotal troubleshooting steps are available.

Conclusions

This project will be successful moving forward into next semester, because of the strong groundwork carried out. This lays a strong foundation for all aspects of the project, not only in our plans for assembly once we have acquired the parts, but in the logistical aspects of this project. We learned how to effectively use the funding available to us, and how to navigate shipping issues. In addition, there are plans to acquire more funding moving forward, which provides this project with a very comfortable budget. We have developed a pragmatic method for getting into contact with the appropriate people for help with purchasing, funding, and technical advice. This moving forward will help guide the rest of the project by identifying the critical aspects that need more attention. Next semester, there will be a large amount of effort in constructing the Rambot, and getting it to balance and walk. This is to ensure we have some demonstration of the fundamental aspects of a quadruped robot. From there, other aspects of the design such as fluid motion, and computer vision will be further elaborated. The only slight uncertainty that remains is how every part we need will be 3D printed. We are currently printing some parts, and have a connection with someone who can print 3D parts in mass. If this fails, we have discussed alternatives to downsize the scope of this project and explore other premade designs of quadrupedal robots that can be extrapolated and improved upon. Overall, we are confident in our ability to deliver a working open source quadrupedal robot that can spark interest in STEMM and be a suitable platform for any related projects.

Future Work

Next Semester

High Level Objectives

As this project has been intended to be multi-year from the start, a significant amount of thought has gone into how to continue work. In regards to next semester, a rigid plan has been prepared. With the remainder of our hard-to-get parts shipping in over winter break, the task of building the robot is front and center. This will include preparing the parts from the recently released openDog V3 design for printing, ordering auxiliary parts such as bearings and hardware, and assembling the base frame. Once that is completed, integrations with other areas of the project can begin. This will start with electrical work; the designs of connecting boards and power supplies need to be finished and fabricated. Then, firmware can be developed on microcontrollers, providing a foundation for movement of the robot, as well as communication with peripheral processors. Finally, integration with the Raspberry Pi can begin, placing machine learning and proper user control onto the robot. This is the goal for next semester: to integrate all of the work done so far onto one platform.

Hardware

Although development this semester has focused mainly on the openDog V2, the schematics for openDog V3 have recently been released. This includes the 3D cad which is what we will be using for our skeleton. The version 2 of openDog was a great starting point and we used it for practice on 3d printing and construction before the version 3 was released. The main issue with version two is the legs had 5 to 1 belt reductions which caused belts to skip under certain circumstances if moving too much. The version 3 is upgraded with 10 to 1 gear ratios which allow much stabler support for the already heavy, 20 kilogram, robot. Along with the better stability its design allows it to be folded up similar to the boston dynamics spot for easier transportation.

3D printing something as heavy as this made us consider what material and infill we should be using for the robot. We had to think of the budget yet had to ensure the parts won't break under too much pressure or melt if sitting in a car on a warm day or the motors overheated. This led us to deciding to use PETG for our 3D printing which is robust and has a much higher heat resistance being a much overall better choice than standard ABS.

The 3D model contains around 230 parts which all needed to be individually separated and downloaded to a separate stl file for 3d printing. A simple script can separate the parts into individual components through autodesk fusion, but the issue is there is no easy way to download all of these parts as stl files as a macro function. This means that each part must be individually downloaded as an stl file and ensure no double or lacking parts.

Software

In regards to software, there are two main areas of work left to complete. The first is in regards to ESP32 work. The intent for next semester, now ESP32 features have been validated, is to implement firmware for the robot. This will include writing code to allow motors to function, auxiliary sensors to be read, and an overall interaction platform to be built for future use. The firmware of this bot is intended to be built once, need little modification from here on out, and be used as a robust layer of abstraction for future development. Specifically, this will involve a few steps. The first step is getting servos to work properly with the ESP32. Next, auxiliary sensor data needs to be read. After this, asynchronous programming can begin; the ESP32 needs to be able to make high speed, low latency calculations for movement. And because the Raspberry Pi is a high-latency device making high-level decisions, the ESP32 should be able to make decisions asynchronously. This will require the implementation of a mid-level API, which must be well documented as a proper layer of abstraction. Goals for this API are allowing the programmer to simply ask the bot to move to a specific location, summarize sensor data, or query hardware statistics.

The bulk of firmware development will be in communication. As the Raspberry Pi on this bot will be the main processor, making the high level decisions, it needs to be able to communicate with the real-time ESP32 microprocessor which asynchronously handles movement. The chosen communication technology is ethernet. This has been chosen over on-board WIFI and Bluetooth for latency reasons. It has also been chosen over CAN bus for bandwidth; very large amounts of data will need to be sent to and from the Raspberry Pi to allow for interfacing with machine learning. Furthermore, having the high bandwidth of ethernet will remove a common bottleneck for another desired feature: logging. If the team can configure the robot to dump all gathered information, tagged with time of acquisition, it will be possible to virtually replay events. This wealth of data can then be used to analyze, step by step, performance of the robot. Doing so will be very helpful for debugging, as well as simulating how the machine learning side of the robot will react in certain situations.

The second main area of work left to complete is the machine learning side. While large successes have been made so far with running neural networks on the Raspberry Pi, work is not finished in this regard. The first thing to do is finish research on object detection, and reach a stable implementation. The next target for machine learning is obtaining hardware acceleration. First, experiments will be made with the Intel Neural Compute Stick. Preliminary research into this device suggests that this will not be an easy task; the development environment for achieving acceleration of machine learning on embedded systems using this device is expressly convoluted at best. Therefore, it is anticipated that a significant amount of time will be required to get this working. However, it will certainly be very useful; early performance results show that accelerating object detection will increase performance by more than 15x. If this holds true, then object detection can become real time. Real time object detection will also introduce new problems with the communication step between the ESP32 and the Raspberry Pi previously mentioned. We anticipate the need to work increasingly closely with firmware as time goes on to optimize the speed of decision making as much as possible.

Another area in machine learning that the team wants to explore is in balance. Preliminary experiments from work already completed this semester suggests the possibility of greatly improving balance and fluidity of motion in the robot using machine learning. Due to the extremely low latency required to get fast, fluid motion working, it is entirely possible that this region of machine learning will need to be executed directly on the ESP32, instead of the Raspberry Pi. We anticipate significant issues to occur here, and in either case, a large amount of research into embedded machine learning will be required. However, we intend to begin by segmenting the tasks at hand; running high-level machine learning for fluid motion on the Raspberry Pi, and building an interface into the firmware for fast execution. We hope that the ethernet communication previously mentioned will facilitate this properly, but in the event that the previously mentioned ESP32 method is required for latency reasons, this approach will allow work to be shared and little time to be lost.

Outreach Work

As more work is completed, there will be an increasing ability to show our work. As we are working closely with the ECE Outreach Senior Design Project, this will fortunately be easy for us to do. A large portion of the work in setting up meetings will be taken care of, but our team will still need to prepare demos. These demos can take place at on-campus events, or if things initially go well, at visits for local middle and high school students. If progress is particularly fast and demos go very well during next semester, there is the possibility of further involvement in Outreach. This would come in the form of workshops, an activity that would likely employ our VIP students to take concepts learned in RamBots and apply them to smaller demos.

Future Years

While future years of senior design are quite far away, the team has already given this topic a great deal of thought. As mentioned above in the Standards for Open Source section, the team intends to make this project as easy as possible to continue work on. Seeing so many other senior design teams fail spectacularly in this regard, we have already employed several strategies to mitigate this, and hope to prepare as well as possible for the hand-off to next year's senior design students. Specific goals, obviously, have yet to be developed but we are in the process of developing a framework to help future students in this regard. One such way we have done this is scouting for future senior design members. Our current VIP students express interest in joining as full-effort Senior Design students next year; we are working closely with them to ensure that our documentation and processes are easy to pick up as someone with surface-level knowledge of the specifics of this project. Additionally, we are (informally) interviewing other juniors that may join this project in the spring or next fall, pushing them towards personal and academic projects closely related to potential future work on RamBots.

References

[1] J. Bruton, “opendogv2,” https://github.com/XRobots/openDogV2/, 2020.
[2] ——, “opendogv3,” https://github.com/XRobots/openDogV3, 2021.
[3] Microsoft, “Coco - common objects in context,” https://cocodataset.org/home, 2021.
[4] ODrive Organization, “Odrive,” https://github.com/odriverobotics/ODrive, 2021.
[5] Rustdoc Organization, “The rustdoc book,” https://doc.rust-lang.org/rustdoc/index.html, 2021.
[6] Ultralytics Organization, “Yolov5,” https://github.com/ultralytics/yolov5, 2021.
[7] PJRC, “Teensy and Teensy++ pinouts,” https://www.pjrc.com/teensy/pinout.html, 2021.
[8] Waveshare, “Pippy Product Listing,” https://www.waveshare.com/pippy.htm, 2021.

Appendix A: List of Abbreviations

- FMEA: Failure Modes and Effects Analysis
- STEMM: Science, Technology, Engineering, Mathematics, Medicine
- PETG: Polyethylene Terephthalate Glycol
- CAD: Computer Aided Design
- PCB: Printed Circuit Board
- FPGA: Field-Programmable Gate Array
- ABB: Automation Company that Operates mainly in Robotics, Power, heavy electrical equipment and automation technology areas.
- FANUC: Japanese Robotics Company that provides automation products and services like robotics and computer numerical control wireless systems.
- RTOS: Real Time Operating System. Incredibly lightweight operating system which runs on microcontrollers to provide basic functionality.
- TPU: Tensor Processing Unit. Processors that are better at tensor calculations than conventional CPUs.

Appendix B: Budget

Presented in this section is a cost outline for the project. Although updated several times throughout the semester, it reflects the current state of the project including anticipated upcoming costs. All items have been purchased indirectly through Outreach. This table only notes costs.

See Appendix D for details on funding.

Item	Quantity	Price	Parts Total	Purchased
As5047 encoder	8	16.76	134.08	Yes
ESP32 board	3	11	33	Yes
teensy 4.1	2	29.38	58.76	Yes
Pi NoIR Camera - 8 Megapixel (V2	2	25	50	Yes
Network Switch 8 port	1	13.49	13.49	Yes
Network Switch 5 port	1	17.99	17.99	Yes
Intel Neural Compute Stick 2	2	70	140	Yes
lcd voltage display	1	10	10	Yes
5v converter	2	9.1	18.2	Yes
Raspberry Pi battery	2	12.95	25.9	Yes
battery charger	1	40.99	40.99	Yes
Display	1	74.89	74.89	Yes
Updated 10/5:				
Dremel	1	96.99	96.99	Yes
Large motor battery	1	170	170	
Updated 10/12:				
Raspberry Pi 4 Model B	1	149.99	149.99	Yes
Camera Module 2	2	47.99	95.98	Yes
PLA Plus Filament Green	1	24.99	24.99	Yes
PLA Plus Filament White	1	26.88	26.88	Yes
dvi to hdmi cable	3	4.69	14.07	Yes
Updated 11/10				
Google Coral	4	149	596	No
18560 battery	10	10.83	108.3	No
mpu 6050	1(3)	9.99	9.99	No
Raspberry Pi 4 Model B	1	149.99	149.99	No
PLA Plus Filament Green	1	24.99	24.99	No
PLA Plus Filament White	1	26.88	26.88	No
Motor Drive 56V 3.6	8	185	1480	Yes
Motor KV90	16	85	1360	Yes
3D Printing Estimates				
Full Bot using PETG	2	2500	5000	No
			Sum Total	$9952.35

Appendix C: Project Plan Evolution

Overview of Design Changes

This project went through several major revisions. Planning first began at the end of the Spring 2021 semester, as senior design was beginning. As this is a first-semester and very open ended project, a large number of decisions needed to be made before work could begin. The very first iteration of the project plan aimed for a bipedal robot. As the team had plans to incorporate mechanical engineering students into the design process, the plan was to start everything from scratch. This included designing the frame, electronics, and hardware. With an overall objective of creating a robot to be used in search-and-rescue scenarios, we thought that this would be an appropriate challenge for a large and academically diverse team to work on for a few years.

The first major design shift occurred in the first week of semester. It was at this point that the team learned that mechanical engineering students were in short supply, and none were able to join our project. With this in mind, significant restructuring of the project was required. The first objective to change was designing everything from scratch. With the ability to properly design a frame ourselves no longer a possibility, we came to a solution in two parts. The first was to use a design already available and open source as a platform to place our electronics and software on. The second was to simplify the overall design from a bipedal robot to a quadrupedal robot. The intent behind this change was to ensure that our members -- who do not have experience in mechanical engineering -- could properly manufacture and repair the robot if designs arose. Choosing a simpler design allowed us to be more efficient with our time.

The next major design shift occurred when dealing with purchasing parts. Unfortunately, due to the global parts shortage, our first few intended designs included parts which were entirely unobtainable. In the pursuit of finding a design which could be realized with readily available parts, discussion led to James Bruton's openDog V2, an open source quadrupedal robot made of mostly 3D printer parts. This was a great advantage for us, as the only difficult parts to obtain in this design included motors and motor drivers. Seeing that this bot was great in terms of hardware, but significantly lacking in terms of hardware, it was the perfect platform for us.

The most recent design shift was more minor: James Bruton released schematics for openDog V3. With notable improvements in functionality, the team decided to move forward with printing this model over the old one. At the point of its release, all parts ordered remained compatible, so little budget was lost during this switch. Marginal work was lost in printing parts for the openDog V2, but the team decided that the improvements in design were worth the cost of time.

At the current time, this is the current project plan: to manufacture openDog V3, and improve on its electrical and software capabilities.

Timeline

Our timeline for the project, miraculously, has stayed fairly consistent throughout the semester. While minor changes have been required to handle week-to-week problems, we have managed to mostly follow the first draft of our timeline due to liberal allowances for setbacks. The timeline is presented below, with color color coordination provided below it.

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16
Website	Devin	Devin	Devin											Devin	Devin	Devin
Budget	Jared Ritwik Oscar	Jared Ritwik Oscar	Jared Ritwik Oscar													
Project Plan	Jared Ritwik Oscar	Jared Ritwik Oscar	Jared Ritwik Oscar	All Team			Jared Ritwik Oscar	Jared Ritwik Oscar	All Team					Jared Ritwik Oscar	Jared Ritwik Oscar	All Team
FMEA			Craig													
3D Printing								Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig	Jared, Oscar, Craig
Power requirements				Craig	Craig											
Electronics Choices						Jared	All team	All team								
Embedded Systems																
Purchasing			All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team	All Team		
Wiring Schematics			Jared, Oscar	Jared, Oscar	Jared, Oscar	Jared, Oscar	Jared, Oscar	Jared, Oscar	Jared, Oscar	Jared, Oscar	Jared, Oscar	Jared, Oscar	Jared, Oscar	Jared, Oscar		
Robot Build													Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik
Wiring													Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik	Jared, Oscar, Craig, Ritwik
Programming					Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin	Devin	Devin	Devin	Devin
Testing												Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris	Devin, Kris
Machine Learning					Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik	Kris, Ritwik
Presentation Slide												All Team			All Team	
DTVC Document										All Team						

Figure 6: Timeline

Not Started	
In Progress	
Finished	Finished but will need revision
Revision needed	
Revision in Progress	

Figure 7: Timeline Color Meaning

Appendix D: Funding

As part of senior design each student receives $200 dollars as initial funding. Our team consists of 6 senior design students which brought our initial budget to $1,200. As part of the Outreach team, we were granted an initial budget of $3000 to order our parts. This brings our current operating budget to $4,200.

In addition to this, the team is also working towards securing funding from Ball Aerospace, in collaboration with the GoKart team. Our initial requirements for this proposal was $5,500. Due to project setbacks and efforts towards a robust plan, this was increased to $11,000. While this process is still in progress, it is anticipated that the proposal will be accepted and the funds received early next semester.

Acknowledgments

It is very important to acknowledge all of the people who have contributed to this project and helped get us to the point we are at, even with all of the major struggles along the way. We would like to thank the following individuals:

Olivera Notaros for helping put us all together and started up on the project. We know we have had a lot of issues up to now but she has always been there at the most crucial moments.

Our EIR, Eli Scott, who has done so much review and guiding for design, production and choosing parts. He made most of the more difficult seeming parts of this project much easier.

Our two Industry mentors Ian Bernstein and David Farrell who were with us from the very start of this project and helped in the tough moments for deciding what this project should be about, and have always offered themselves to do as much as they can and for a first year team it means a lot.

Our Junior VIP members who are making important contributions despite their very busy schedules.

Ty Thourot has lent his expertise in 3D printing, since many of us have little to no experience in this field.