ELECTRICAL AND COMPUTER ENGINEERING
COLORADO STATE UNIVERSITY

RamBOTs

Mid-project Report

Fall Semester 2022

Senior Design Team Members:

Electrical Engineers:

Alex Kolodzik, Michael Bearly, Kyle Biskupski

Computer Engineers:

Gwyndolyn Tari, Evan Hassman, Eric Percin, Thomas Veldhuizen

Mechanical Engineers:

Eric Olson, Kyle Moore

Other Team Members:

Junior Outreach Members:

Anna Biolchini, Joey Reback

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

Project advisor(s): _____Olivera Notaros_____

Approved by: _____Olivera Notaros_____

Date of approval: ____12/10/2022_____

ABSTRACT

The RamBOTs team's goal is to create an open source quadrupedal robot that is openly available, affordable, and intended to be an educational tool. Currently, these types of robots are only available for purchase by companies, not individuals. With that said, the price of these robots still makes them inaccessible even for the companies that are able to purchase them. This project is an extension of ECE outreach and aims to make this form of robot more accessible by building off of openDog V3 created by James Bruton to create an effective, accessible, and affordable version of a quadrupedal robot that is able to serve as an educational tool for students of varying age groups and technical levels.

The basis of this project focuses on the open source project mentioned above which, although openly available, lacks software documentation and leaves room for improvement and additions as a whole. The team has already improved upon the documentation of the open source project and continues to add their own modifications. Such modifications include a more in-depth physical controller and a control method utilizing machine learning to get the robot to follow a desired object. A number of unit tests (particularly for the motors) have been implemented to help ensure a smoother debugging process that is well-documented and can be replicated by another team.

The principal findings of this project so far include the necessity of good documentation and unit testing when it comes to debugging and assembling a robot consisting of many integrated components. It also includes ODrive configurations, power, communication, Arduino and Raspberry Pi code, and the project's git repository.

As this project is still relatively new, there is still plenty of future work needed as the full assembly is yet to be achieved. Some of the key future aspects include full assembly, fine-tuned balance and walking movements, gyroscope integration, machine learning integration, and preset programmed movement sequences.

TABLE OF CONTENTS

Chapter 1. Introduction

1.1 Context/Overview

Quadrupedal robots are a relatively recent development within the robotics field. Due to this, they are not commonly available or open for commercial sale at all. These forms of robots are specifically restricted to be purchased exclusively by larger companies and not individuals. Even if one is able to purchase one, they are incredibly expensive making them entirely inaccessible. With that said, the overall goal of this project is to create larger accessibility for this form of robot.

The base of the project utilizes the open source openDogV3 and intends to improve upon and extend the capabilities of the base model. This project adds to the goal of accessibility by having a majority of the robot consist of 3D printed parts. Along with this 3D printing, the robot contains a total of twelve brushless motors, six ODrives (one for every two motors), a Teensy 4.1 Arduino microcontroller, and a Raspberry Pi 4a+. The Teensy is used to communicate and control the ODrives to produce desired motor movement while the Raspberry pi serves as a way to interface control methods as well as run the machine learning object recognition program.

1.2 Goals/Project Design Requirements

To begin with goals, there are five major goal categories for this project. The first category pertains to stable standing/balancing and movement. That means that the motors are able to consistently and accurately be programmed to move in a desired pattern or sequence. The robot should also be able to walk towards a desired position without falling over and is able to complete common operational movements and an implemented gyroscope is able to accurately determine and return movement data. The robot will have three major forms of movement/control input. The main form of control movement focuses on a physical controller's input utilizing joysticks and buttons to communicate to the robot a desired movement sequence. The secondary form of control utilizes the already existing object recognition capabilities to allow the robot to follow a desired item/sign. The final form of control also depends on machine learning as it focuses on following voice commands as input for certain pre-programmed sequences such as walking forward, sitting, or emitting a sound.

The project design requirements have adapted and changed throughout the course of this semester, but some of the original goals are present. These original goals consist of items such as utilizing two 2.2V batteries as a power source for the entire robot, the final product having a dog-like shape based on the previous 3D printed components, the color of the final product being black (hardware) and white (3D printing filament), the material of the final product is

PetG, PLA, and carbon fiber rods, the microcontrollers in the project consist of a Raspberry Pi (4a+) and an Arduino (Teensy 4.1), and the final product is able to move while carrying 45kg, or roughly 100 pounds.

The new design requirements focus on aspects of the design that are necessary to produce a fully functioning and safe product that were not initially considered. The biggest safety additions include an emergency shutdown button that is large and bright red to be easily recognized and interacted with and two 30A 12-48V breakers such that one can be attached to each battery in the event of sudden and undesired battery discharge. A lot of the new requirements also focused on power and included items such as six 2Ω 50 Watt resistors such that each ODrive has one with the intended purpose to be used for slowing down movement within the brushless motors, 10 AWG 2-core stranded wire to connect the batteries to their respective breakers, 20 AWG 5-core wire to connect the ODrives to the actual motors, and diodes to control current flow such that current exclusively flows out of the batteries to protect the batteries and all components they are attached to.

1.3 ECE Outreach

This project is an extension of ECE outreach whose main goal is to increase both recruitment and retention within the ECE department by offering visits to middle schools and highschools to highlight the major and expose the students to what ECE offers as well as by offering workshops for current CSU students and other academic opportunities to retain more students within the major. With all of that said, the final product of the RamBOTs team is intended to be used as an educational tool by ECE Outreach. Even while in a nonfunctioning state, it has been taken to several outreach-based events to show the possibilities of engineering and interdisciplinary projects. For future use, this project as a whole is capable of demonstrating many aspects of engineering including concepts such as power, multidisciplinary engineering collaboration, mechanics/kinematics, machine learning, CAD, and robotics as a whole.

1.4 Remainder of Report

The majority of the remainder of this report is broken up into 6 other sections. Chapter 2 details a summary of the previous work that was done by the previous (last year's) team and what the current team inherited when they began work in the second year of this project. Chapter 3 focuses on both the hardware and mechanical-based work of the project by providing a general overview and then presenting physical components, power, and communication-based connections. Chapter 4 explains the software design aspects of the project with an overview and then an in-depth look into the Raspberry Pi code, the Arduino code, ODrives and motor interface, and the Git repository. Chapter 5 focuses on the standards used for the project such as open source, education, and accessibility. Chapter 6 is the

conclusion of the report which explains our final findings and results of the project over the course of the semester. Chapter 7 details the expected future work for the project as we transition to the next semester and focuses on items such as continued work and testing with the ODrives and motors and machine learning integration.

Chapter 2. Summary of Previous Work

RamBOTs is a multigenerational robotics project that is in its second year as a senior design option. As such, a great deal of progress had already been accomplished at the beginning of this year. This allowed the team to start working immediately without worrying about several issues that otherwise would have arisen, such as acquiring hardware.

The majority of the time spent by last year's team was acquiring materials. The hardware and electronics required took several months to order due to the extreme supply issues that were present at the time. Having the correct motor, motor controllers, microcontrollers, and other hardware parts saved this year’s project weeks of time despite the fact that there were issues discovered with several parts. Furthermore, acquiring tools and hardware such as the appropriate nuts, bolts, and bearings took a considerable amount of time given the lack of prior experience the previous team members had. Because of their hard work, the current team did not have to worry about any of those issues and dealt mainly with making the previously acquired components work properly and making the robot move.

Another major difficulty was 3D printing, though this was not due to supply chain issues. The chassis is constructed mostly of 3D printed materials. Many of them are quite large and need to be stronger than standard PLA to withhold the strain of the robot. Finding printers large enough to accommodate all the parts needed took time, and printing all the parts took even longer. Once a printing service was found it took approximately 2,500 hours of printer time and cost several thousand dollars. Using stronger plastic, PETG, and the higher infill all increased print time and material cost. However, the 3D printed parts and all the previously mentioned hardware meant this year began with a fully assembled chassis. Lastly, the motor housing and gearboxes were all assembled by the previous team. This included assembling the 3D printed parts but also cutting over one hundred metal rods and fitting them with bearings to ensure smooth operation of the cycloidal gearboxes.

Finally, the previous team managed to develop and implement object detection using machine learning as a way for the completed robot to interact with its environment. The Raspberry Pi, which will be used to control the robot, can currently run this machine learning model. This model utilizes the already existing MobileNet V2 model fine-tuned on the Microsoft COCO dataset; this allows it to recognize the 90 different common objects present within COCO. Unfortunately, the Raspberry Pi is fairly slow to perform inferences on images, so a Google Coral hardware accelerator is used to expedite classification.

The previous team left a solid foundation for the continuation of their project which is a key reason for the significant progress made on the RamBOT this semester.

Chapter 3. Hardware Design

3.1 Hardware Overview

As mentioned in the previous section, the majority of hardware and infrastructure present in the current product was passed on from the previous senior design team. What had not been completed by the previous team was the implementation of movement and the installation of safety features required for a finished product. The hardware team split into groups in order to solve these problems during the first semester of this year's project.

Of the safety issues that were identified, the first to be addressed was the integrity of the components that make up the RamBOT. Improvements were made to the harness and body of the robot to increase stability and usability, power routing was looked at in detail to minimize the risk of using large batteries, and sensors were added to identify irregularities during movement.

This team was also responsible for testing all components for functionality as well as configuring them to work with each other. Through research and repeated tests, a reliable method of setting up the motors, ODrives, and encoders was finalized.

With the improvements discussed in this section, the project will be in an excellent position to proceed with the integration of moving parts in the coming semester.

3.2 Physical Components

There were three major advancements made by the hardware team concerning physical components. The first of these were metal segments added to connect portions of the robot that were previously moving freely on the carbon fiber rods. Metal rulers were used, as they provided adequate strength as well as a consistent width between pieces.

These segments were connected between each motor group housing in order to minimize twitching due to motor movement. The front and back ends of the robot were connected with these segments as well, to negate the possibility of one end sliding off when the bot is tilted or moving.

The second major advancement was adjustments to the harness. The harness is a stand made up of 3D printed parts and metal beams that were

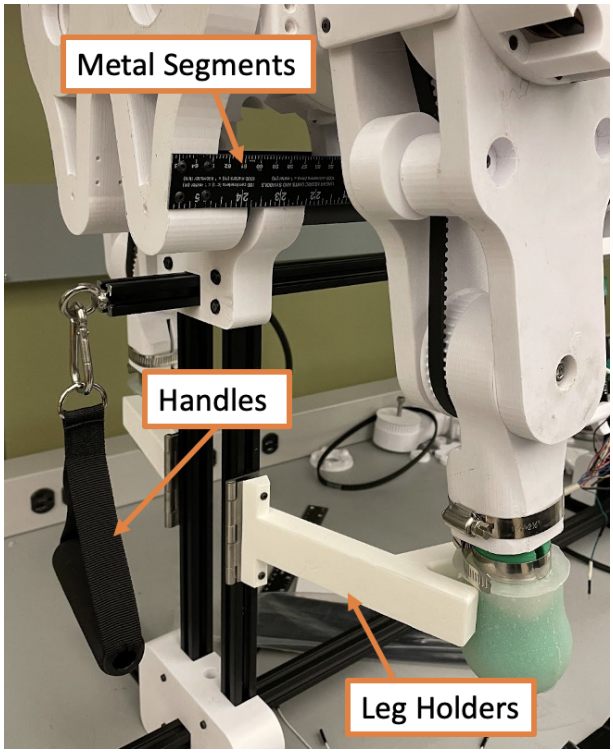


Figure 1: Labeled image of added physical components

previously used only as a workstation, allowing for work to be done on the robot when raised off the ground. Seeing an opportunity to improve the harness, eye hooks were threaded into the axis of the metal beam to allow for flexible handles to be attached for easier transportation of the robot.

Finally, the last advancement mentioned in this section is the addition of rotating leg holders. In attempting to design a new harness, the main concern was the existing leg holders obstructing the path of where a leg could move. Instead of using a majority of the project's remaining budget on a new harness, leg holders that move out of the way during testing were designed. These holders use a y-shaped end for the legs to sit in and are connected to the metal beams that make up the harness with door hinges. When testing begins, and the legs are initialized in a starting position, these holders will be swung out of the way to avoid leg contact.

3.3 Power Wiring

The RamBOT is powered by two 22.2V 6000mAh lithium polymer batteries connected in parallel. Each battery is in series with a 30A breaker which limits the maximum current of the circuit to 60A. A 24V 80A relay is used to connect a power switch as well as an emergency shutoff switch to the circuit. The power switch and emergency shutoff switch only need to be rated for 1 to 2A because the majority of the current that the bot uses will flow through the relay. 10-AWG wire will be used between the batteries, breakers, and relay, and two 10-AWG wires will be run in parallel from the relay to the connection point for the ODrives.

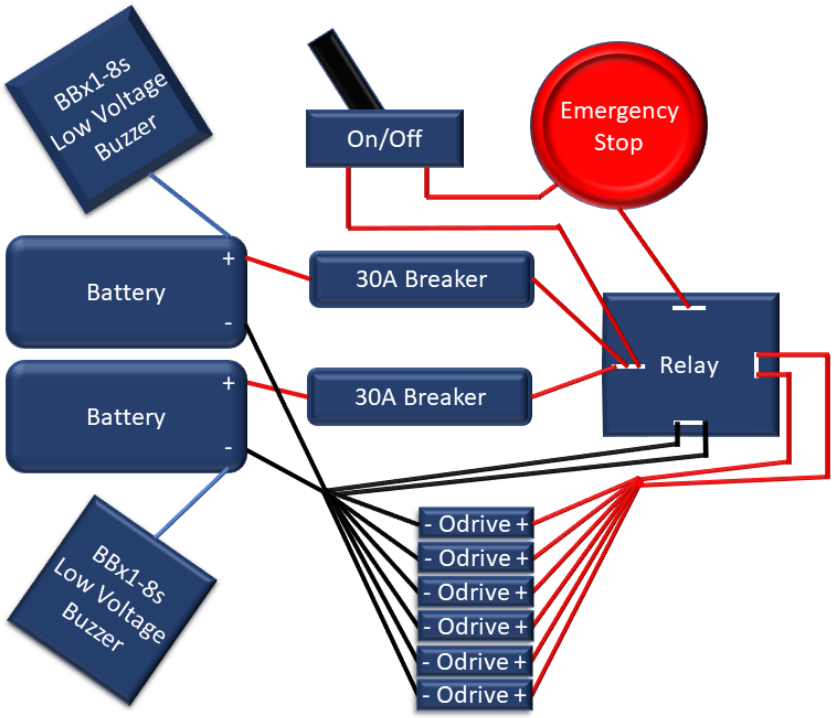


Figure 2: Power Wiring Diagram

Safety was a fundamental concern when it came to designing the high voltage system, and several safety features were added to the design this semester. In addition to the breakers, switches, relays, and low-voltage buzzers were purchased for the batteries. These devices

display the voltage of each cell in the battery and will emit an audible alarm if the voltage of any cell drops below a pre-set value. The voltage detectors are being used for two purposes. The first is to verify that all of the cells in each battery are within the 0.1V tolerance from each other before connecting them in parallel to the bot, and the second is to ensure that the batteries will never deplete below their safe voltage levels.

3.4 Communication Connections

To make sure that the various microcontrollers and subsystems work together, there are smaller wires used to communicate between different parts of the bot. The Pi is thought of as the ‘brains’ of the bot and is delegated to take inputs such as video feed or commands from a remote and convey the direction of movement through a USB connection to the Teensy 4.1. The Teensy can be thought of as the nervous system of the bot; it is responsible for breaking down movement commands into an ODrive-specific message that is sent through the UART transmitter and receiver wires connected to each ODrive. The Teensy will also have a gyroscope/accelerometer connected via I^2C to counteract any unwanted movement.

The Pi will send its movement data over a USB cable and Tensy will receive this information by reading from its first serial port. The wire used for this connection is a USB 2.0 A male to micro USB B male. It will also provide power for the Teensy and any lower voltage components connected to the Teensy. The Teensy has an I^2C connection to the MPU6050 Gyroscope and Accelerometer. I^2C is a standard communication method that allows many devices to communicate with only two wires between them: the SCL and SDA wires. The Teensy works as the master and the gyroscope works as the slave, only responding to requests when the Teensy requests data. The plan is to interpret the rotation of the bot's body and the acceleration to counteract any unwanted movement using automatic compensation.

Another crucial connection is the communication and control of the six ODrives. The way the Teensy will communicate with the ODrives is by a UART connection of two wires to each ODrive. One wire is TX (transmission) and one is RX (receiving). Physically speaking, this means that the final design will need at least 12 wires connected to the Teensy. The team set up a 25-pin connector out of the Teensy housing to be routed to the six ODrives. Each ODrive was assigned a color: for example, ODrive 1 is responsible for the right-front hip and the right-back hip and is connected with solid red wire and red with white stripes wire. There are also multiple ground wires being used to increase signal integrity and also provide each ODrive a connection to ground.

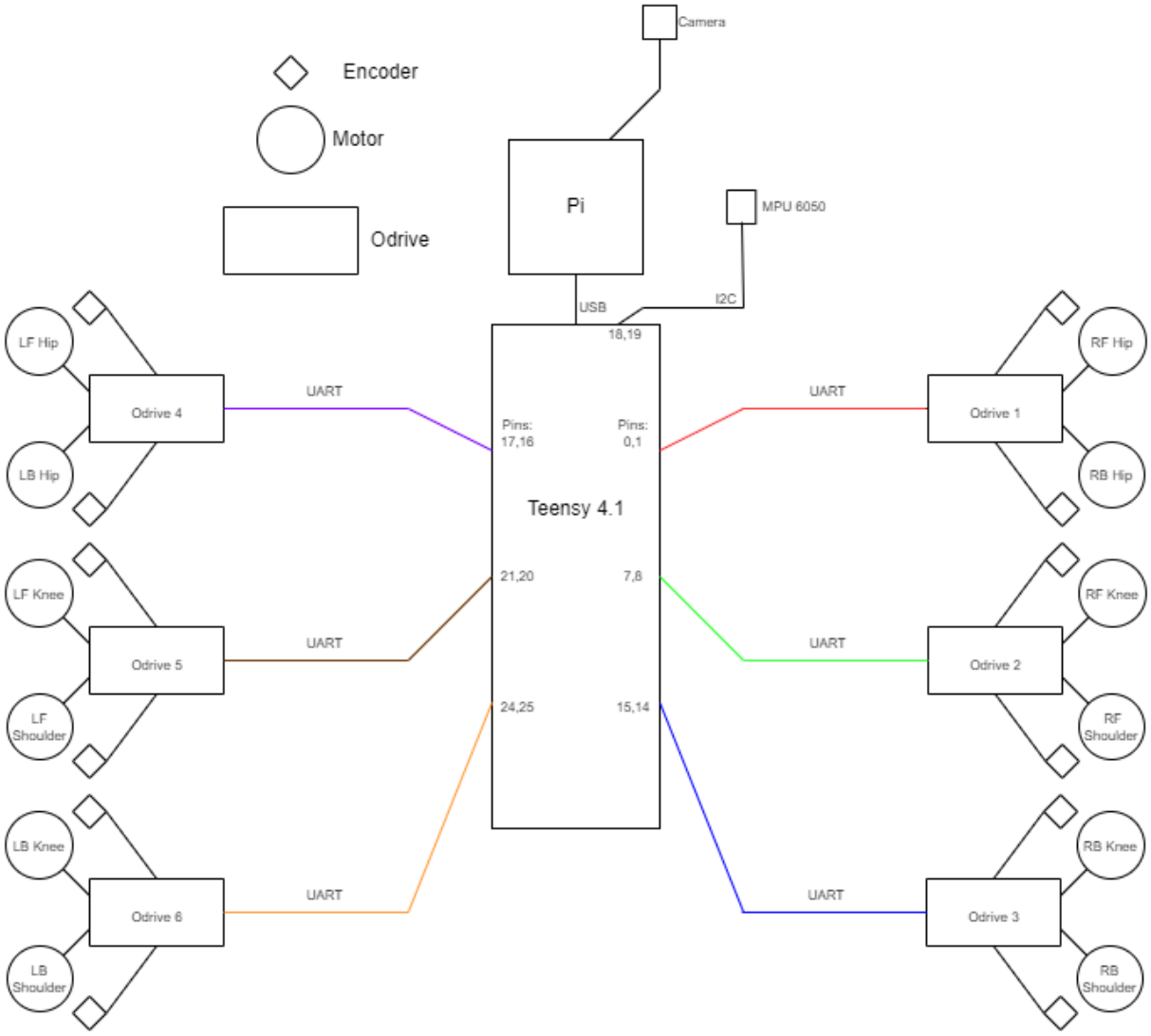


Figure 3: Teensy communication diagram

3.5 Motor, Encoder, and ODrive Functionality Testing

The priority for the Mechanical Engineering students on the hardware team was to ensure all hardware was functional and to make the robot move according to design specifications. The first pieces of hardware to be tested were the motors, as no movement code would be of use if it were sent to faulty motors. To test the motors' functionality, the robot was fully disassembled, and a simple circuit utilizing a potentiometer was assembled. This allowed the hardware team to not only test the functionality of each motor safely but to also observe the responsiveness and speed capabilities of each motor. These unit tests on all 16 motors purchased by last year's team were completely successful, ensuring that the required 12 motors and the 4 in reserve were safe for use.

After ensuring that all motors were functional, the hardware team set up unit tests for the encoders. These tests involved connecting the motors and encoders to the ODrives (shown in wiring diagram figure 3), and the ODrives to a computer to display the outputs of the encoders' readings of the motors' positions. These unit tests were fully successful, confirming that all 8 of the team's owned encoders and the method used to mount them were functioning correctly. After noticing some inconsistencies between different ODrives on startup, it became apparent that much testing and changing of existing configurations on the ODrives was required in order to obtain consistent, reliable outputs from the hardware of the robot.

3.6 Configuration and Implementation

In an attempt to achieve more consistent and predictable results from the ODrives, the hardware team searched for required settings in the openDog V3 documentation and videos. It was found, in a small clip, that the ODrives needed exact settings for the specific motors and encoders used for this project. This was done by plugging the ODrive into a power source and a computer via micro-USB, and then the settings were changed using commands in the "odrivetool" Python library through the command window. From there, the team experimented with changing different configurations in the ODrives as a means to get all of the hardware ready to read movement code and eventually allow for the operation of the robot. The currently known values that must be changed are shown in Table 1 below.

ODrive Configurations	
Parameter	**Value**
Encoder CPR	4000
Motor Pole Pairs	20
Motor Torque Constant	8.27/90 = 0.0919
Controller Position Gain	60
Controller Velocity Gain	0.1
Controller Velocity Integrator Gain	0.2
Controller Velocity Limit	Infinity (less chance of over current failure)
Motor Pre-Calibrated	True (after calibration on set up)
Index Pre-Calibrated	True (after calibration on set up)
Encoder Index Search on Startup	True

Table 1: Set of ODrive configurations for successful motor movement and calibration

After finding a sufficient ODrive configuration, movement was tested using commands in Python to set the positions of individual motors. The maximum bounds that do not result in pieces of the legs contacting were found in this testing. This allowed for the movement focus to be transferred over to the Teensies. While a full leg has not yet been assembled and tested, four motors are currently being tested for continuity. Prior to these trials, the ODrives were set to be put in a closed loop on start-up, allowing instant motor position control. This proved to cause spikes in movement on the index search for the encoder. To combat this, the Teensy code was equipped with a method for reading if the index has been found and putting the motors in a closed loop after it has been found. More information on the software used for this testing and plans for the future is included in the next section.

Chapter 4. Software Design

4.1 Overview and Methodology

Software design of the RamBOT is split into three primary categories: Python programs running on the Raspberry Pi 4 a+, C++ programs running on the Arduino Teensy 4.1, and ODrive endpoint operations. The Raspberry Pi is responsible for receiving various user inputs and communicating them to the Teensy microcontroller over UART in a standardized format. Additionally, as a single-board computer, the Pi is more versatile and powerful than a typical microcontroller. It is capable of running programs with relatively high memory intensity such as machine learning models, and with a USB accelerator such as the Google Coral becomes even more proficient. It also enables the seamless integration of additional peripherals such as Bluetooth controllers, HDMI displays, and USB devices. The Arduino Teensy is used for high-speed kinematics calculations and serial communication with the Pi and six ODrives. Lastly, the ODrives are used for driving twelve motors. All of this software aside from the machine learning model and kinematics calculations are the original work of this year's team. The RamBOT software flow is depicted in the figure below and described in detail in the following sections.

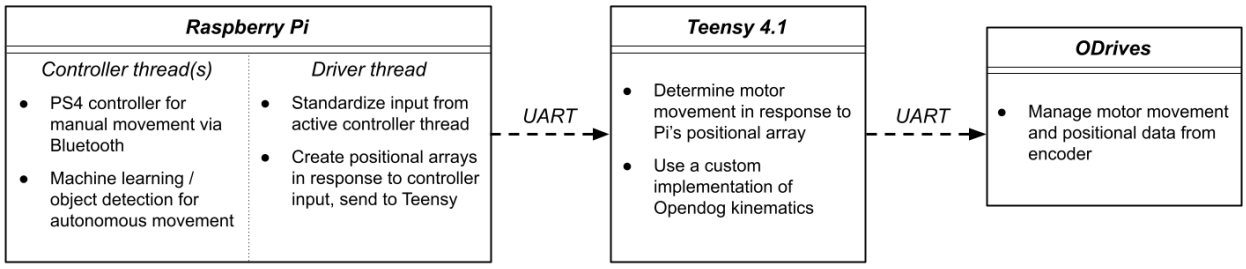


Figure 4: Software flow diagram

Open source / libre development methodologies are essential to the RamBOTs project as an extension of ECE Outreach. The goal of the RamBOT is to make such a robot more accessible and affordable than it is at present; it stands to reason that the software must also be accessible and understandable. Thus, the software team has taken special care to ensure the code they produce is well-documented and legible, and the details of this process are included under section 4.5.

4.2 Raspberry Pi (Python)

James Bruton's original openDogV3 design uses two Teesnsy microcontrollers, one for the processing of controller inputs and one for kinematics calculations and serial communication

with the ODrives. The two microcontrollers communicate over radio frequency to allow for wireless control. The implementation of a Raspberry Pi single-board computer in lieu of the first Teensy is perhaps the most significant difference between the RamBOT and openDogV3. This change allows for a significant increase in processing power and ability, namely the implementation of different controller modes and graphical user interfaces. The Pi's primary program is multithreaded, with up to three threads active at any given time. Controller threads handle distinct methods of controller input, such as a physical controller or autonomous machine learning-based control. The driver thread is responsible for standardizing inputs from the active controller thread and processing input data into a movement array that is sent to the Teensy via serial communication.

The first controller mode / thread utilizes a PlayStation DualShock 4 wireless controller connected to the Raspberry Pi through its native Bluetooth integration and the Pygame python library. The DualShock 4 features two joysticks, a directional pad, two triggers, two bumpers, four buttons, an LED light bar, and a touchpad–all of which are planned to be used for manual control as depicted in the following diagram:

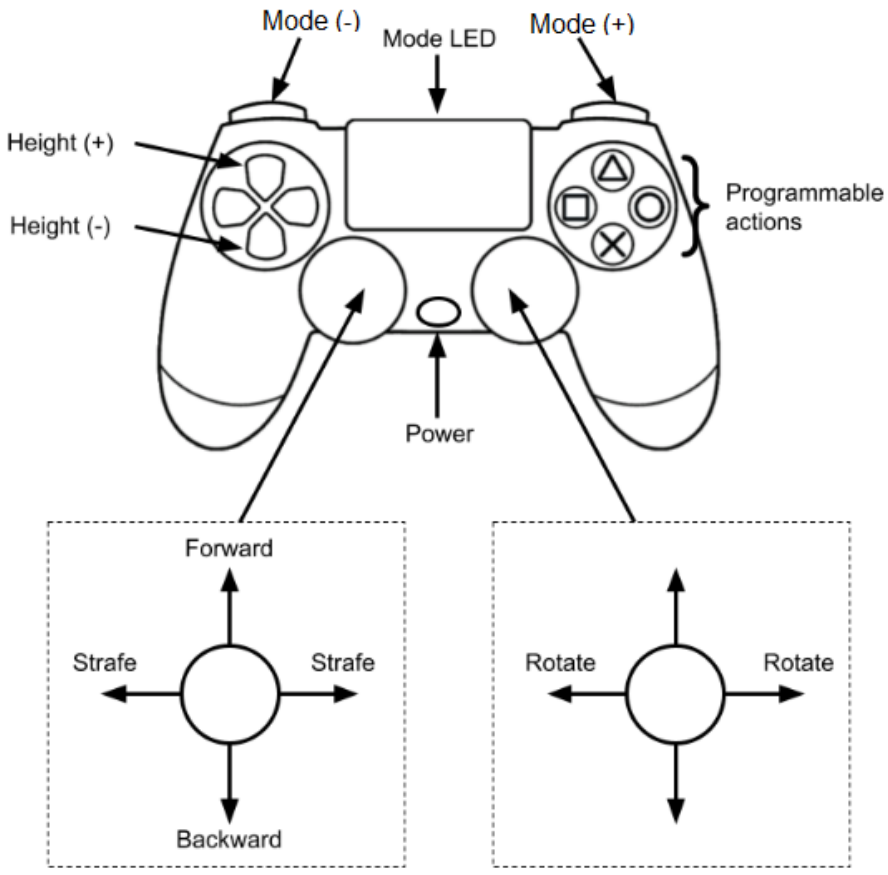


Figure 5: Dualshock 4 controller input map

The second controller mode/thread utilizes a machine learning model for autonomous movement. Using a USB webcam, Tensorflow Lite, OpenCV, and a single-shot multibox detection network trained on the COCO dataset, this model is capable of identifying common household objects and computing their bounding boxes as depicted in the figure below. For instance, it can easily recognize sports balls, humans, cell phones, and backpacks. The intention is to use this model as a basis for autonomous movement by having the RamBOT follow a person or a thrown ball to emulate playing fetch.

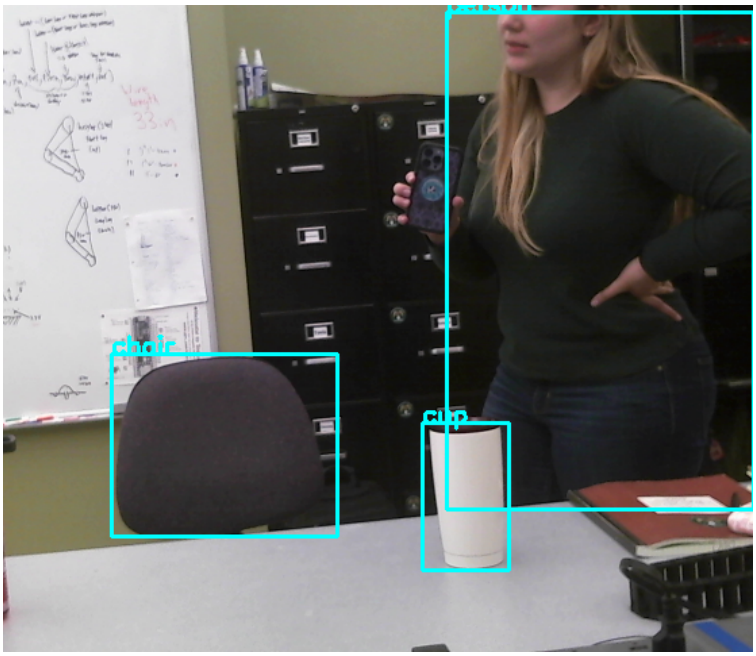


Figure 6: Output of single-shot multibox detection network

Additional controller threads are planned for later implementation–namely additional machine learning modules. For example, a neural network for speech recognition would allow the operator to verbally command the robot, such as to ‘sit’ or ‘speak.’ Different controller threads could very well employ different machine learning models to utilize specific strengths of certain networks, such as one trained especially to identify humans and one trained to follow a particular object.

The driver thread communicates to the active controller thread via a shared queue. The active controller thread only has the ability to place items into the queue, while the driver thread only has the ability to read items from the queue. The queue is formatted in a two letter keyword followed by a colon. An example of this could be the set array value command, keyword ‘AR:’, or the terminate command, keyword ‘TM:’. The driver thread is constantly waiting for items to be placed into the queue to maximize the speed at which the system operates, as it is

very important that once the movement array has been modified, the Teensy is able to update it as soon as possible. The driver thread communicates through a USB serial communication, each message padded to 127 characters to fully fill the serial buffer. This is because the serial communication will wait for a full line before it sends. This allows for a sufficient speed of communication between the devices, eliminating any noticeable lag between the commands. The driver thread is also in charge of creating or killing the other control method threads to ensure no orphan threads are left in memory.

While the controller thread that communicates with the PS4 controller cannot terminate until the robot is ready to be shut down, any additional threads are unneeded and create needless overhead for the Raspberry Pi. These threads will be actuated with the driver thread as well as the PS4 controller thread. The controller thread will set the mode based on the L1 and R1 buttons, then it will send a mode change command (keyword ‘MC’) to the driver thread, whereupon reception, the driver thread will terminate the thread for the current mode, and initialize the thread for the next mode.

4.3 Arduino Teensy (C++)

The Arduino Teensy 4.1 communicates with the Raspberry Pi’s driver thread using a USB serial communication that has been padded to 127 characters to ensure maximum speed and data transfer by bypassing buffer protocol. These strings contain the keywords that control the Teensy itself. Much like the Raspberry Pi’s driver thread, the Teensy takes keywords to adjust the values of the movement array or send special commands to it. This is incorporated into the main cycle of the Teensy.

On startup, the Teensy begins by waiting for the motor controllers to find their indexes. Once found, the Teensy sets the motors into a closed loop before moving into its main loop. The main loop starts by checking the serial buffer for any messages from the driver thread. If any messages are found, the Teensy will execute them based on keywords. The most common keyword is ‘AR:’, the keyword that updates the movement array. These keywords can also include ‘MC’ for mode change, ‘TM' for terminate, or any future keywords, as these methods are very easily expandable. After checking for keywords, the main cycle updates the positions of the legs. This is currently done in an updateLeg() function, but this is subject to change as the actual movement cycle of the legs is still in development. This completes the main cycle of the Teensy, and next the cycle begins again. The Teensy runs at 600MHz and does not require multithreading for sufficient performance, but does possess the capability if the need arises.

4.4 ODrives and Motor Interfacing

All 12 motors on the RamBOT connect to the ODrives for the motor control. ODrives utilize motor encoders to determine the precise position of each motor and allow the motors to be set by index to very high precision. Each ODrive can operate two motors, and as such, six ODrives are utilized on the RamBOT. Each ODrive communicates by hardware serial pins, of which there are eight pairs of serial pins on the Teensy 4.1. This leaves two sets available if needed in the future.

On startup, the Teensy begins its communication with the ODrives by starting the serial_1 to serial_6 objects on the 115200 baud rate. This keeps the 9600 baud rate available for the Pi-Teensy communication. Once these serial objects have been created, ODrive objects from the ODriveArduino library are created, enabling direct control of the ODrives. Next, the ODrives are given velocity and current limits before being restarted and told to find the indexes. The motors then move in a clockwise motion slowly until each has found its index. At this point, the ODrives are placed into a closed-loop state to prevent unwanted movement. Now the ODrives are ready to receive motor movement commands.

The current kinematics function was taken directly from James Bruton's openDogV3 and allows the programmer to enter a leg and an x,y,z position to then calculate the correct index for each of the three motors in a leg to bring that specific leg to the designated position. The integration of this code was fairly seamless and saved a great deal of time for the RamBOTs software team, enabling emphasis on other aspects such as multithreading, machine learning, and Teensy code, as well as the proper movement cycles for the robot legs in order to fully utilize the movement array.

4.5 Git Repository and Documentation

The RamBOTs Github organization was created and maintained throughout the semester for the purposes of revision history, documentation, and the preservation of open source / libre development methodologies. Within this organization are five distinct repositories, each chronicling different aspects of the RamBOT's development:

- **Testing**: the most commonly used repository this semester, with 48 commits from 3 contributors. It contains testing and work-in-progress programs for the Raspberry Pi, Teensy, and ODrives. Among the Raspberry Pi test programs are controller code, serial communication, and graphical user interfaces. The Teensy test programs include ODrive serial communication unit tests, gyroscope programs, and serial communication between the Pi and ODrives with a custom implementation of the

openDog kinematics function. Lastly, the ODrive test programs include initialization protocols and unit tests for the ODrives. The main readme of this repository contains a detailed history of contributions to the code base, challenges that arose, and how they were overcome.

- **Machine Learning**: the repository which contains the object detection model implemented by the previous year's team. This program uses Tensorflow Lite and OpenCV using a single-shot multibox detection network trained on the COCO dataset.
- **Webpage**: the repository containing the source code for the official RamBOTs website. It was developed independently from the previous year's website using JavaScript, the frontend JavaScript framework Vue.js, and the Vue.js material design framework Vuetify.
- **Locomotion**: the repository intended to contain the completed programs with which the RamBOT is operated. Presently it contains the code used by James Bruton in openDogV3 with additional comments and explanations from this year's team.
- **.github**: the repository for display on the public RamBOTs organization profile. It provides an overview of the repositories listed above and includes the copyright information and license of James Bruton's openDogV3 which the RamBOTs project is based on. Also included are links to the official RamBOTs website and James Bruton's openDogV3 Github repository.

Open source / libre development practices have taken a number of different forms throughout the software design process. Firstly, the original openDogV3 software is very useful but sparsely documented. The software team had to work through James Bruton's code line by line to understand it thoroughly enough to take advantage of his work, and while doing so added many comments to the benefit of later readers. Secondly, the team enforced a consistent commenting style for use in their programs. This is a block comment style where, in addition to single-line comments, at the top of a program or function is a block of text detailing the purpose of the code it precedes, who wrote that code, and when it was written or modified. Such homogeneity in comment structure will benefit the understanding of individuals seeking to understand or recreate this project. Lastly, the team has utilized Github readme to a significant extent. Every repository in the Github has a readme in which there is a chart to list important files or directories and their description. The testing readme also includes an extensive history of daily work done to the repository, including what difficulties were faced and how they were resolved. The software team's commitment to documentation reflects the values of the project as a whole and will serve to increase accessibility and understanding of quadrupedal robotics.

Chapter 5. Standards

Despite the general open-endedness of this project, there are still many standards the team needs to uphold in relation to the nature of the project as open source and an educational tool. This means great emphasis must be placed on documentation, reproducibility, and safety.

The origin of the project is open source and it will remain open source throughout its development. This means that the documentation of the project and its construction is absolutely mission-critical. All work that is done needs to be written down and kept somewhere where it can be accessed by members of the public. One main way this has been accomplished is by using Github. This makes it easy to document and track modifications and improvements made to any and all of the software developed. Furthermore, the code written itself must also be made easy to read and understand its functionality. Thus, the team takes careful consideration with naming conventions as well as including abundant, detailed comments in all parts of the software.

The second point, reproducibility, is an equally important step. It is crucial that RamBOTs software and hardware can be reproduced exactly by outsiders attempting the project. This requires detailed documentation but also automates many parts of startup and testing to help others make sure everything is running quickly and intuitively. It would be a poor educational tool if a user loses interest after struggling to get it to work. For this reason, the team has included a myriad of unit tests in their code. Tests isolate specific components and can be run automatically to determine where errors lie to quickly address potential issues.

Lastly, because this is an educational tool, safety is paramount as this robot will operate around children. Special care must be taken that every part of the project has adequate safety measures to avoid injury to team members or any other observers. This has been accomplished by adding both software and hardware emergency stops in all stages of movement. Also, breakers and relays have been included to limit current in places where it could otherwise reach dangerous levels. Finally, all wires and components were selected carefully to make sure that they will never receive more power than what they are rated for which could cause damage to hardware and danger to people nearby.

The goal of our project is to make a quadrupedal robot but also to make it accessible, reproducible, and safe to display. Many steps have been taken to this end which will help make the project more successful.

Chapter 6. Conclusion

Overall, the progress that the RamBOTs team has made this semester was exceptional. When starting the semester, the team essentially had only untested hardware, a 3D printed chassis, and a machine learning model. The motor statuses were unknown and the same was true for the ODrives. The machine learning program was effective, but tangential to the ultimate goal of locomotion.

At the end of this semester, all of the motors and ODrives have been tested and confirmed to be in working order. The robot was disassembled and observed to better understand its structure and functionality. The body was reassembled, and the harness was upgraded to be more reliable and not inhibit movement testing. The team was also able to work through the entire codebase of openDog V3, adding comments and structure to make it easier to read and manage. They established means of communication between the Raspberry Pi, Teensy, and ODrives, and then began adding new features to the code like controller implementation. The motors have undergone rigorous testing and bounds detection to avoid damage to the robot. The controller can communicate directly with these motors and provide them with directions to move. The controller inputs can all be seen on the display and have been programmed to adjust the kinematics of the robot directly.

A number of safety additions were made to the project throughout the semester as well. Breakers were added to both of the batteries to interrupt current flow if any faults occur. There is also an emergency stop button and several other safety switches added in case anything starts performing incorrectly.

The team has overcome many external challenges throughout this semester as well. Some examples include having their budget locked for the beginning of the year, or having to entirely relocate their lab over a period of two weeks. Regardless of these issues, the team was able to divide up work and make significant progress toward finishing the project.

Projecting based on the rate of progress, locomotion is expected by midway through the next semester. This should allow time for refinement and further improvements before the year’s end. The team and project have been extremely successful thus far, and by the end of the year, there will be a working quadrupedal robot to prove it.

Chapter 7. Future Work

Regarding the future plans made for this project, the general structure is depicted within the timeline below. Previous timelines from earlier in the semester are included in Appendix C.

	A	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1	Tasks	1/15-1/21	1/22-1/28	1/29-2/4	2/5-2/11	2/12-2/18	2/19-2/25	2/26-3/4	3/5-3/11	3/12-3/18	3/19-3/25	3/26-4/1	4/2-4/8	4/9-4/15	4/16-4/22	4/23-4/29
2	Update Wedsite				EP				EP				EP			
3	Update Timeline				AK				AK				AK			
4	Documentation	GT	GT		GT	GT		GT	GT		GT	GT		GT	GT	GT
5	FEMA															
6	Budget/ Purchase	EH	EH			EH	EH			EH	EH			EH	EH	
7	Project Plan															All Team Members
8	3D Printing Additional Parts				MB, EH, EO, KM					MB, EH, EO, KM					MB, EH, EO, KM	MB, EH, EO, KM
9	Feet Enhancements															
10	Body Modifications															
11	Motor Functions															
12	ODrive Functions															
13	Leg Functionality															
14	Built Harness	GT, EO, KM, AB, JR	GT, EO, KM, AB, JR	Test Body Harness												
15	Config and Install Motors and Drives	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	Test Body Connections									
16	Encoder Implimentation			AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	Test Encoders								
17	Final Movement Testing								EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	Final Motor Test
18	Power Rec's															
19	Wiring Schematics															
20	Wiring and Batteries	Test Wires														
21	Wire Troubleshooting	AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB	Finish Wires									
22	Gyroscope Research															
23	Gyroscopes Test	KB	KB													
24	Gryoscope Install	KB	KB	KB	KB	KB	Test Gyro									
25	Verify Old Code															
26	Embedded Systems															
27	Controller Build															
28	Multithreading	GT, EH, EP	GT, EH, EP	Communcation Proof												
29	Inegrate New Code	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	Code Test							
30	GUI		GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GUI Test						
31	Machine Learning Enhancing Code	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	ML Test					
32	Additional Modes and Functions									All Team Members	All Team Members	All Team Member	All Team Members	All Team Members	All Team Members	Final Mode Test
33	VIP Projects	AB, JR	AB, JR	AB, JR	AB, JR	AB, JR	AB, JR	Test Extras								

Figure 7: RamBOTs timeline, current version semester 2

In terms of specific tasks and delegations, the current plans are to continue working on the ODrive communications and how they interact with the motors. Now that the harness has been updated, it will allow the legs to be tested more easily; the motors can be placed into the 3D printed parts and tested on the stand itself. After enough testing is done on each of the legs, wires will be finalized and the robot will be tested off of the stand. From there, walk cycles will be implemented using the existing kinematics and testing functions. This task is estimated to be completed around late February to early March.

Regarding software, another area of improvement is the machine learning module for which a camera will be mounted on top of the robot. A 3D printed housing will be created for this camera and the Raspberry Pi's LCD display. Further, the output of the machine learning model (which includes an object's classification and its position on the screen) will be converted to movement protocol. This will allow the RamBOT to track specific objects and follow them, or keep its gaze locked onto a given object. It would be able to follow objects and be controlled

easily by them. The first implementation of walking cycles is expected to be jittery–a problem suffered by openDogV3 and its software. Smoothing out the leg movement and making it more ‘dog-like’ will be another area of software improvement addressed next semester.

The team has exciting ideas for extremities and extra features if time allows. For instance, a robotic ram head with servo motors attached to the horns to allow them to spin and turn. Another idea would be to implement a device that can pick up objects so one could play fetch with the robotic dog using machine learning since it is already trained to identify any kind of “sports ball”. Even the idea of a dancing feature has been brought up or teaching the robotic dog several different tricks that it could do similar to how a dog functions. These tricks or gestures would be implemented with different button combinations on the wireless controller.

Throughout the next semester, the VIPs have also been given their own robotic kits that they will be assembling and using as mini projects to promote RamBOTs as a partner with ECE Outreach. They have each received different kits and will be sharing their experiences with the kits as smaller examples of what robotics can do to excite students.

Another significant part of the RamBOTs project will be visiting classroom environments throughout the semester and having the students in those classrooms 3D print their own robots and program them to run and function over three separate sessions. The team members would all show up to these sessions as well as train other members from ECE Outreach in how to program and work on the different robots.

References

[1]

"Teensy® 4.1." https://www.pjrc.com/store/teensy41.html (accessed Dec. 09, 2022).

[2]

"Robot Dog V3 - 3D Printed & Open Source #1 - YouTube." https://www.youtube.com/watch?v=yXA_KeuYpCY (accessed Dec. 09, 2022).

[3]

"Pygame Front Page — pygame v2.1.4 documentation." https://www.pygame.org/docs/index.html (accessed Dec. 09, 2022).

[4]

J. Bruton, "openDogV3." Dec. 04, 2022. Accessed: Dec. 06, 2022. [Online]. Available: https://github.com/XRobots/openDogV3

[5]

"Getting Started — ODrive Pro Documentation 0.6.4 documentation." https://docs.odriverobotics.com/v/latest/getting-started.html (accessed Dec. 09, 2022).

Appendix A - Abbreviations

- ECE - Electrical and Computer Engineering
- CSU - Colorado State University
- EIR - Engineer In Residence
- VIP - Vertically Integrated Person
- PetG - Polyethylene Terephthalate glycol
- PLA - Polylactic Acid
- COCO - Common Objects in Context
- CAD - Computer Aided Design
- AWG - American Wire Gauge
- UART - Universal Asynchronous Receiver-Transmitter
- USB - Universal Serial Bus
- I²C - Inter-Integrated Circuit

Appendix B - Budget

Purchaser:	Part Name:	Price:	Quantity:	Reimbersed (Y/IP/N):	Date:	Total:
Michael	Yardstick	2.98	5	Y		
	Hose Clams	3.98	2	Y		
	Delivery and Tax	1.73	1	Y	9-10	24.59
Eric (Mech)	Lipo Battery Charger 150W	56.99	1	Y		
	Delivery and Tax	4.57	1	Y	9-22	61.56
Evan	Polytek PlatSil Gel-25 2lb	63	1	Y		
	Fabreze Air Freshener 2pk	5.44	1	Y		
	Expo Eraser	2.97	1	Y		
	Ti Scisors 3pk	7.99	1	Y		
	Box Cutter 4pk	9.99	1	Y		
	Plastic Spoon 50pk	4.99	1	Y		
	Delivery and Tax	14.19	1	Y	9-26	108.57
Evan	50PC Socket Set	44.97	1	Y		
	TI Drill Bit Kit	19.97	1	Y		
	Magnetic Bowl	12.97	1	Y		
	12PC Screwdriver Set	19.97	1	Y		
	Universal Screwdriver	14.97	1	Y		
	Hex Bit Set	8.87	1	Y		
	70PC Drill Driving Set	20.97	1	Y		
	18V Drill	79	1	Y		
	Delivery and Tax	16.74	1	Y	10-10	238.43
Evan	PS4 Controller	59	1	Y		
	Micro USB Cord	10	1	Y		
	Delivery and Tax	5.48	1	Y	10-10	74.48
Evan	Wireless Mouse & Keyboard	23.99	1	Y		
	Delivery and Tax	2.08	1	Y	10-12	26.07
Evan	Tackle Boxes (6)	30	1	Y		
	25/C Cable for DB25	12.99	1	Y		
	DB25 Male	9.58	1	Y		
	DB25 Female	9.58	1	Y		
	Gyroscope Accelerometer (3)	9.99	2	Y		
	Emergency Stop Button	13.99	1	Y		
	Lipo Li-ion Battery Alarm (5)	12.99	1	Y		
	Delivery and Tax	8.5	1	Y	11-11	117.61
Evan	4ft x 6ft White-Board	229.99	1	IP		
	Round Up	0.65	1	IP		
	Delivery and Tax	17.36	1	IP	11-16	248
Evan	Expo Markers	6.45	1	IP		
	Dry Erase Cleaner	9.25	1	IP		
	Delivery and Tax	1.46	1	IP	11-24	17.16
Evan	Robot Dog	243.46	1	IP		
	10-in-1 Robot	299.99	1	IP		
	Delivery and Tax	41.3	1	IP		
	9V DC wire x10	5.99	1	IP		
	Arduino Uno	16.99	10	IP	169.9	
	DC Motor Driver	7.49	10	IP	74.9	
	DC Motors x2	13.49	10	IP	134.9	
	Delivery and Tax	29.42	1	IP		
	9V Batterys x10	8.98	1	IP		
	Delivery and Tax	0.95	1	IP	11-26	1009.79

Total Spent	Actual Funds Remaining:
625.42	2174.58

Total (Including non Reimbersed):	Hypothetical Funds Remaining:
1926.26	873.74

Income:

Name:	Date	Value:
Inital Funds	9-14	2800

Figure 8: RamBOTs Budget

Appendix C - Project Plan Evolution

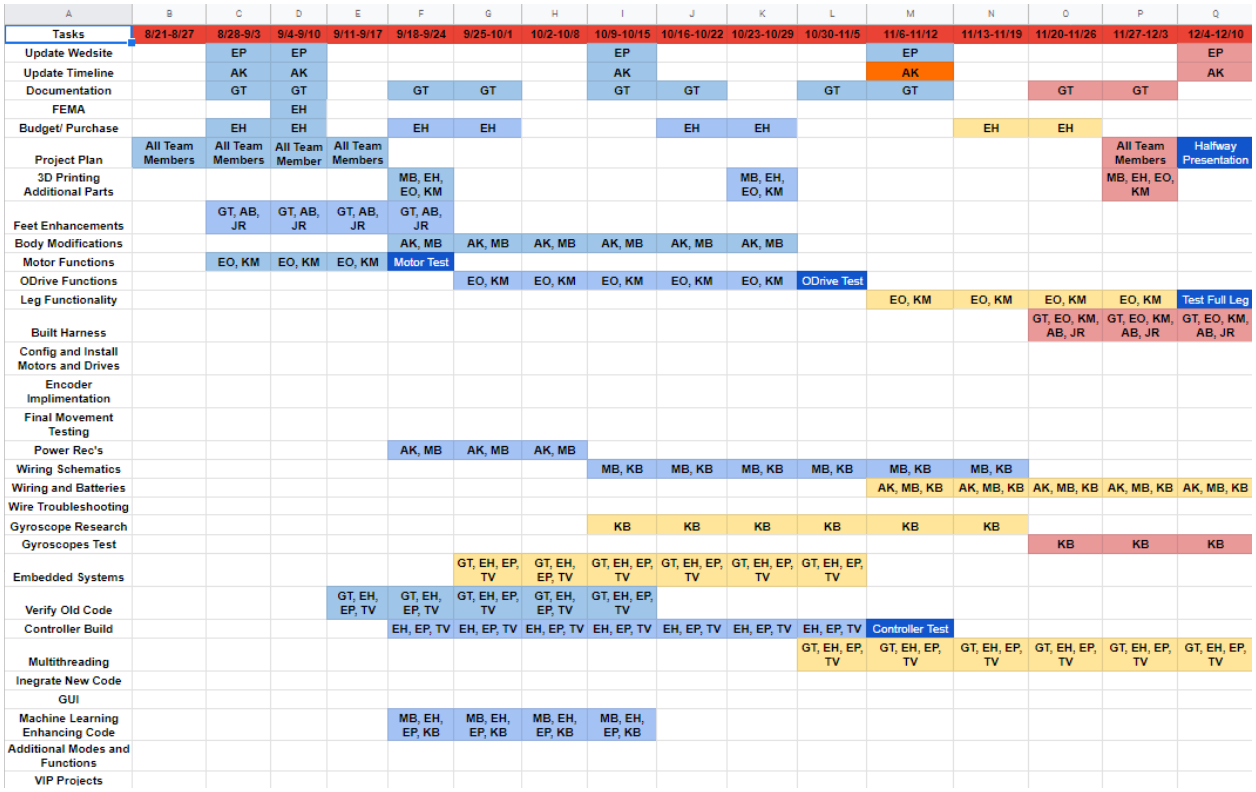
Tasks	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16
Website		Eric P	Eric P													
Update Wedsite								Eric P				Eric P				Eric P
Budget		Thomas, Gwyn	Thomas, Gwyn	Thomas, Gwyn												
Purchasing					Thomas	Thomas	Thomas	Thomas								
Project Plan	All Team Members	All Team Members	All Team Members	All Team Members												
Project Plan Revisions															All Team Members	All Team Members
FEMA			Evan													
3D Printing Additional Parts					Michael, Evan, Eric O, Kyle M					Michael, Evan, Eric O, Kyle M					Michael, Evan, Eric O, Kyle M	
Power requirements					Michael, Alex	Michael, Alex	Michael, Alex									
Feet Enhancements		Gwyn, Joey, Anna	Gwyn, Joey, Anna													
Motor Functionality		Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M												
ODrive Functionality					Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M								
Leg Movement Functional									Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M
Leg Movement Testing																
Motor Troubleshooting																
Embedded Systems						Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas				
Wiring Schematics							Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B					
Wiring												Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B
Wiring Troubleshooting																
Installing Gyrscopes											Kyle B	Kyle B	Kyle B	Kyle B	Kyle B	Kyle B
Testing Gyroscopes																
Programming					Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas	Gwyn, Evan, Eric P, Thomas
Testing Walk Cycle Code																
Machine Learning Relearning Code					Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B								
Machine Learning Enhancing Code									Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B
Machine Learning Finalizing Code																

Color Meaning								
Not Started	In Progress	Finished	Revision needed	Revision in Progress	Revision Complete	Finished but will need revision	Semester 1	Semester 2

Figure 9: RamBOTs timeline, version 1 semester 1

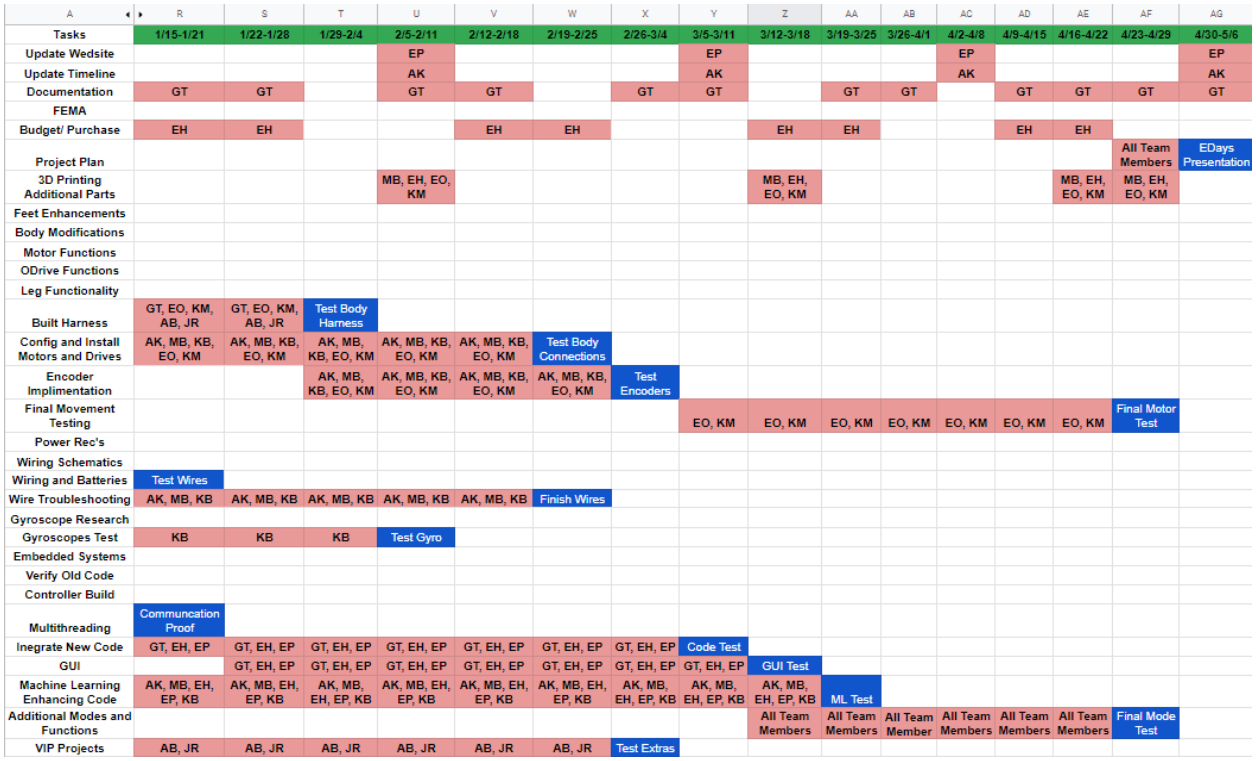
Tasks	Week 17	Week 18	Week 19	Week 20	Week 21	Week 22	Week 23	Week 24	Week 25	Week 26	Week 27	Week 28	Week 29	Week 30	Week 31	Week 32
Website																
Update Wedsite				Eric P				Eric P				Eric P				Eric P
Budget	Alex, Gwyn	Alex, Gwyn	Alex, Gwyn													
Purchasing	Alex, Gwyn	Alex, Gwyn	Alex, Gwyn	Alex, Gwyn												
Project Plan																
Project Plan Revisions	All Team Members	All Team Members														
FEMA																
3D Printing Additional Parts				Michael, Evan, Eric O, Kyle M					Michael, Evan, Eric O, Kyle M					Michael, Evan, Eric O, Kyle M	Michael, Evan, Eric O, Kyle M	
Power requirements																
Feet Enhancements																
Motor Functionality																
ODrive Functionality																
Leg Movement Functional																
Leg Movement Testing	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M									
Motor Troubleshooting								Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M	Eric O, Kyle M
Embedded Systems																
Wiring Schematics																
Wiring																
Wiring Troubleshooting	Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B	Alex, Michael, Kyle B										
Installing Gyrscopes																
Testing Gyroscopes	Kyle B	Kyle B	Kyle B	Kyle B	Kyle B											
Programming	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P
Testing Walk Cycle Code	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P	Gwyn, Evan, Eric P
Machine Learning Relearning Code																
Machine Learning Enhancing Code	Michael, Evan, Eric P, Kyle B															
Machine Learning Finalizing Code		Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B	Michael, Evan, Eric P, Kyle B
	Color Meaning															
	Not Started	In Progress	Finished	Revision needed	Revision in Progress		Revision Complete		Finished but will need revision		Semester 1	Semester 2				

Figure 10: RamBOTs timeline, version 1 semester 2



Tasks	8/21-8/27	8/28-9/3	9/4-9/10	9/11-9/17	9/18-9/24	9/25-10/1	10/2-10/8	10/9-10/15	10/16-10/22	10/23-10/29	10/30-11/5	11/6-11/12	11/13-11/19	11/20-11/26	11/27-12/3	12/4-12/10
Update Wedsite		EP	EP					EP				EP				EP
Update Timeline		AK	AK					AK				AK				AK
Documentation		GT	GT		GT	GT		GT	GT		GT	GT		GT	GT	
FEMA			EH													
Budget/ Purchase		EH	EH		EH	EH			EH	EH			EH	EH		
Project Plan	All Team Members	All Team Members	All Team Member	All Team Members											All Team Members	Halfway Presentation
3D Printing Additional Parts					MB, EH, EO, KM					MB, EH, EO, KM					MB, EH, EO, KM	
Feet Enhancements		GT, AB, JR	GT, AB, JR	GT, AB, JR	GT, AB, JR											
Body Modifications					AK, MB	AK, MB	AK, MB	AK, MB	AK, MB	AK, MB						
Motor Functions		EO, KM	EO, KM	EO, KM	Motor Test											
ODrive Functions						EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	ODrive Test					
Leg Functionality												EO, KM	EO, KM	EO, KM	EO, KM	Test Full Leg
Built Harness														GT, EO, KM, AB, JR	GT, EO, KM, AB, JR	GT, EO, KM, AB, JR
Config and Install Motors and Drives																
Encoder Implimentation																
Final Movement Testing																
Power Rec's					AK, MB	AK, MB	AK, MB									
Wiring Schematics								MB, KB	MB, KB	MB, KB	MB, KB	MB, KB	MB, KB			
Wiring and Batteries												AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB
Wire Troubleshooting																
Gyroscope Research								KB	KB	KB	KB	KB	KB			
Gyroscopes Test														KB	KB	KB
Embedded Systems						GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV					
Verify Old Code				GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV								
Controller Build					EH, EP, TV	EH, EP, TV	EH, EP, TV	EH, EP, TV	EH, EP, TV	EH, EP, TV	EH, EP, TV	Controller Test				
Multithreading											GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV
Inegrate New Code																
GUI																
Machine Learning Enhancing Code					MB, EH, EP, KB	MB, EH, EP, KB	MB, EH, EP, KB	MB, EH, EP, KB								
Additional Modes and Functions																
VIP Projects																

Figure 11: RamBOTs timeline, version 2 semester 1



Tasks	1/15-1/21	1/22-1/28	1/29-2/4	2/5-2/11	2/12-2/18	2/19-2/25	2/26-3/4	3/5-3/11	3/12-3/18	3/19-3/25	3/26-4/1	4/2-4/8	4/9-4/15	4/16-4/22	4/23-4/29	4/30-5/6
Update Wedsite				EP				EP				EP				EP
Update Timeline				AK				AK				AK				AK
Documentation	GT	GT		GT	GT		GT	GT		GT	GT		GT	GT	GT	GT
FEMA																
Budget/ Purchase	EH	EH			EH	EH			EH	EH			EH	EH		
Project Plan															All Team Members	EDays Presentation
3D Printing Additional Parts				MB, EH, EO, KM					MB, EH, EO, KM					MB, EH, EO, KM	MB, EH, EO, KM	
Feet Enhancements																
Body Modifications																
Motor Functions																
ODrive Functions																
Leg Functionality																
Built Harness	GT, EO, KM, AB, JR	GT, EO, KM, AB, JR	Test Body Harness													
Config and Install Motors and Drives	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	Test Body Connections										
Encoder Implimentation			AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	Test Encoders									
Final Movement Testing								EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	Final Motor Test	
Power Rec's																
Wiring Schematics																
Wiring and Batteries	Test Wires															
Wire Troubleshooting	AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB	Finish Wires										
Gyroscope Research																
Gyroscopes Test	KB	KB	KB	Test Gyro												
Embedded Systems																
Verify Old Code																
Controller Build																
Multithreading	Communcation Proof															
Inegrate New Code	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	Code Test								
GUI		GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GUI Test							
Machine Learning Enhancing Code	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	ML Test						
Additional Modes and Functions									All Team Members	All Team Members	All Team Member	All Team Members	All Team Members	All Team Members	Final Mode Test	
VIP Projects	AB, JR	AB, JR	AB, JR	AB, JR	AB, JR	AB, JR	Test Extras									

Figure 12: RamBOTs timeline, version 2 semester 2

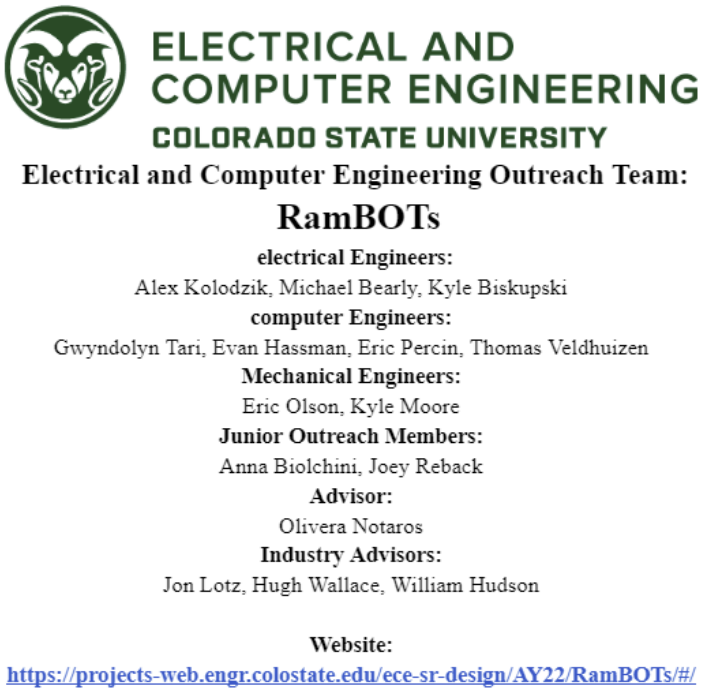
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Tasks	8/21-8/27	8/28-9/3	9/4-9/10	9/11-9/17	9/18-9/24	9/25-10/1	10/2-10/8	10/9-10/15	10/16-10/22	10/23-10/29	10/30-11/5	11/6-11/12	11/13-11/19	11/20-11/26	11/27-12/3	12/4-12/10
2	Update Wedsite		EP	EP					EP				EP				EP
3	Update Timeline		AK	AK					AK				AK				AK
4	Documentation		GT	GT		GT	GT		GT	GT		GT	GT		GT	GT	GT
5	FEMA			EH													
6	Budget/ Purchase		EH	EH		EH	EH			EH	EH			EH	EH		
7	Project Plan	All Team Members	All Team Members	All Team Member	All Team Members											All Team Members	Halfway Presentation
8	3D Printing Additional Parts					MB, EH, EO, KM					MB, EH, EO, KM					MB, EH, EO, KM	
9	Feet Enhancements		GT, AB, JR	GT, AB, JR	GT, AB, JR	GT, AB, JR											
10	Body Modifications					AK, MB	AK, MB	AK, MB	AK, MB	AK, MB	AK, MB						
11	Motor Functions		EO, KM	EO, KM	EO, KM	Motor Test											
12	ODrive Functions						EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	ODrive Test					
13	Leg Functionality												EO, KM	EO, KM	EO, KM	EO, KM	Test Full Leg
14	Built Harness														GT, EO, KM, AB, JR	GT, EO, KM, AB, JR	GT, EO, KM, AB, JR
15	Config and Install Motors and Drives																
16	Encoder Implimentation																
17	Final Movement Testing																
18	Power Rec's					AK, MB	AK, MB	AK, MB									
19	Wiring Schematics								MB, KB	MB, KB	MB, KB	MB, KB	MB, KB	MB, KB			
20	Wiring and Batteries												AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB
21	Wire Troubleshooting																
22	Gyroscope Research								KB	KB	KB	KB	KB	KB			
23	Gyroscopes Test														KB	KB	KB
24	Gryoscope Install																
25	Verify Old Code				GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV								
26	Embedded Systems						GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV					
27	Controller Build					EH, EP, TV	EH, EP, TV	EH, EP, TV	EH, EP, TV	EH, EP, TV	EH, EP, TV	EH, EP, TV	Controller Test				
28	Multithreading											GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV	GT, EH, EP, TV
29	Inegrate New Code																
30	GUI																
31	Machine Learning Enhancing Code					MB, EH, EP, KB	MB, EH, EP, KB	MB, EH, EP, KB	MB, EH, EP, KB								
32	Additional Modes and Functions																
33	VIP Projects																

Figure 13: RamBOTs timeline, current version semester 1

	A	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
1	Tasks	1/15-1/21	1/22-1/28	1/29-2/4	2/5-2/11	2/12-2/18	2/19-2/25	2/26-3/4	3/5-3/11	3/12-3/18	3/19-3/25	3/26-4/1	4/2-4/8	4/9-4/15	4/16-4/22	4/23-4/29
2	Update Wedsite				EP				EP				EP			
3	Update Timeline				AK				AK				AK			
4	Documentation	GT	GT		GT	GT		GT	GT		GT	GT		GT	GT	GT
5	FEMA															
6	Budget/ Purchase	EH	EH			EH	EH			EH	EH			EH	EH	
7	Project Plan															All Team Members
8	3D Printing Additional Parts				MB, EH, EO, KM					MB, EH, EO, KM					MB, EH, EO, KM	MB, EH, EO, KM
9	Feet Enhancements															
10	Body Modifications															
11	Motor Functions															
12	ODrive Functions															
13	Leg Functionality															
14	Built Harness	GT, EO, KM, AB, JR	GT, EO, KM, AB, JR	Test Body Harness												
15	Config and Install Motors and Drives	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	Test Body Connections									
16	Encoder Implimentation			AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	AK, MB, KB, EO, KM	Test Encoders								
17	Final Movement Testing								EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	EO, KM	Final Motor Test
18	Power Rec's															
19	Wiring Schematics															
20	Wiring and Batteries	Test Wires														
21	Wire Troubleshooting	AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB	AK, MB, KB	Finish Wires									
22	Gyroscope Research															
23	Gyroscopes Test	KB	KB													
24	Gryoscope Install	KB	KB	KB	KB	KB	Test Gyro									
25	Verify Old Code															
26	Embedded Systems															
27	Controller Build															
28	Multithreading	GT, EH, EP	GT, EH, EP	Communcation Proof												
29	Inegrate New Code	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	Code Test							
30	GUI		GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GT, EH, EP	GUI Test						
31	Machine Learning Enhancing Code	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	AK, MB, EH, EP, KB	ML Test					
32	Additional Modes and Functions									All Team Members	All Team Members	All Team Member	All Team Members	All Team Members	All Team Members	Final Mode Test
33	VIP Projects	AB, JR	AB, JR	AB, JR	AB, JR	AB, JR	AB, JR	Test Extras								

Figure 14: RamBOTs timeline, current version semester 2

Appendix D - Letters of Interest submitted for grants, or similar



ELECTRICAL AND COMPUTER ENGINEERING
COLORADO STATE UNIVERSITY

Electrical and Computer Engineering Outreach Team:

RamBOTs

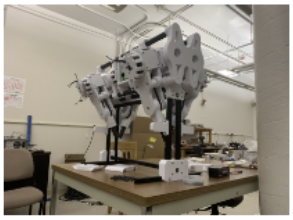
electrical Engineers:
Alex Kolodzik, Michael Bearly, Kyle Biskupski
computer Engineers:
Gwyndolyn Tari, Evan Hassman, Eric Percin, Thomas Veldhuizen
Mechanical Engineers:
Eric Olson, Kyle Moore
Junior Outreach Members:
Anna Biolchini, Joey Reback
Advisor:
Olivera Notaros
Industry Advisors:
Jon Lotz, Hugh Wallace, William Hudson

Website:
https://projects-web.engr.colostate.edu/ece-sr-design/AY22/RamBOTs/#/

Figure 15: Ball Aerospace Proposal Page 1

Introduction:

CSU Outreach is looking to encourage students to become excited about engineering and show them the benefits and unlimited possibilities of the electrical and computer engineering major. One of CSU's Outreach projects is RamBOTs. RamBOTs goal is to design a quadrupedal robot that can walk and demonstrate machine learning. This robot is to be used as an educational tool that will demonstrate the possibilities of engineering to many groups of younger and older students. RamBOTs is looking to add more outreach programs and smaller related design projects than the current budget will be able to support.



The Main Robot/Senior Design Project:

The main goal of RamBOTs is to design a quadrupedal robot. The robot is 31" to 52" (with legs extended) long, 22" tall, and 21" to 31" (with legs outstretched) wide. It will be able to walk based on controller input, image recognition, or voice recognition. As this is a second-year senior design project, the current team inherited the basic assembled chassis and legs that were immobile, which is what the previous year's budget was spent on. The current team's task is to assemble the robot and to enable its movement, as well as to program the controller input and neural networks for image and voice recognition. On top of that, the team will be integrating additional sensors to help create more fluid movement. Currently, the computer engineering students are working on the programming of the ODrives and microcontrollers, the electrical engineering students are wiring the robot as well as drafting circuits for the sensors, and the mechanical engineering students are working on the body and the configurations for the ODrives. Before we are able to advance on this complex project, we require funding to purchase multiple replacement and spare components, as well as funding to test individual parts before overall integration.

Outreach Activities:

While the main goal of RamBOTs is to design the quadrupedal robot, the overall goal of the project is to spark interest in engineering. Because of this, RamBOTs is looking to start workshops in which students of all ages (upper elementary students through underclassmen in college) will be able to design and build robots utilizing tools and equipment used in the real world. This includes 3D printing, soldering, programming, and various other tools for older students. This will be done in collaboration with the ECE Outreach program and will allow proper training to run the workshops and for advertising. We also want to have some finished, smaller projects to be able to show off and spark interest in engineering as well as have something that the students can design and take home.

Team Composition:

RamBOTs is a subsection of ECE Outreach. As such, we would partner with ECE first-year advisors and many ECE instructors to reach out to students and bridge connections within their coursework. There are also many highly motivated and passionate Vertically Integrated Project (VIP) students who will work with us on the projects and help run events. In doing this, VIP students are able to gain valuable hands-on experiences to further their education as undergraduate students. The students on RamBOTs are also incredibly passionate about their projects and their majors as a whole and enjoy sharing their excitement and passion with other students. It is because of this that we will be able to both confidently and competently run these workshops.

Figure 16: Ball Aerospace Proposal Page 2

Simple Project Budget:

Area	Item	Cost
The Main Project (RamBOT)	Peripherals for Machine Learning (Input devices and hardware accelerators)	$950
	Required Replacement ODrives and Motors	$1,300
	Sensors including Gyroscopes, Accelerometers, IR Distance Sensors, etc.	$420
	Power Sources (Batteries, Chargers)	$500
	3D Printing Filament	$460
	Outsourcing 3D printing	$1,600
	Supplies for Testing of Subsystems	$850
	Precise, Long Lasting Tools for Working on the Robot (power tools, drill bits, socket set, etc.)	$750
		Total Cost: $6,830
General Workshops	Various Advanced Independent Project Kits (Team property) (~$304/Pc) x5. Example and Example	$1,520
	Snacks and Refreshments for Workshops	$500
		Total Cost: $2,020
Take Home Robots	3D Printed Robots to be taken home. Highly Interactive Educational Tool. (~$43/Pc) x60 Example	$2,580
	Sets of Engineering Equipment/Tools	$200
	3D Printing Filament	$200
		Total Cost: $2,980
		Overall Cost: $11,830

Thank you again for all the support you have given in the past as well as for your time and consideration. It has been a huge impact in the past allowing us to fulfill our vision in the project properly. We hope you will be able to continue to support us in the future as well. Please feel free to reach out to us with any questions or concerns.

Evan Hassman- EvanH799@gmail.com / Evan.Hassman@colostate.edu / (720-238-5885)
Olivera Notaros- Olivera.Notaros@colostate.edu Gwyndolyn Tari- gtari@colostate.edu

Figure 17: Ball Aerospace Proposal Page 3

Acknowledgements

We are very grateful for the help we have received from several people over the course of this semester and would like to use the end of this report to thank all of those people who have taken time out of their days to contribute to both the team and the project.

Olivera Notaros for guiding us through taking over this project and for contributing a great deal of time to ensure that our lab transition was as smooth as possible.

Our EIR mentors, Jon Lotz, Bill Hudson, and Hugh Wallace, who have spent a lot of time mentoring us to ensure we can navigate some of the more difficult aspects of the project and keep moving the team forward as a whole.

Our VIPs, Anna Biolchini and Joey Reback, who have spent a lot of time in the lab with us doing a variety of tasks even though they are both busy with their own coursework.

Kristopher Alquist for taking the time to help the team with setting up the machine learning aspect of the project that he worked on prior to graduating last year.