Modular Robotics: Molucube

Cole Butler
Ali Helmi
Andrew Hunt
Carlos Torres

CONCEPT OF OPERATIONS

REVISION – 1
3 October 2022

Concept of Operations
for
Modular Robotics: Molucube

Team <10 >

Approved by:

Project Leader Date

Prof. Kalafatis Date

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	09/15/2022	Carlos Torres		Draft Release
1	10/03/2022	Andrew Hunt		Revisions 1

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

1. Executive Summary

Our goal for this project is to create a series of modules that can be combined into a manually configurable, modular robot capable of performing a user defined task. We specifically aim to create a system that is simpler and less expensive than alternative, autonomously reconfigurable designs. Our modular robot system will have modules that are each designed to perform a specific function as a part of the whole robot and can easily be swapped with other modules as well as be reconfigured by the user to perform new tasks. Each module will rely on a central controller module that will gather information from the other modules, as well as tell each module what to do. This will concentrate the most complex tasks in one module, which will allow us to meet our goal of making the other modules simpler and cheaper. A central controller module will also make it easy to configure the robot, as every other module will be set up to accept commands from the central module.

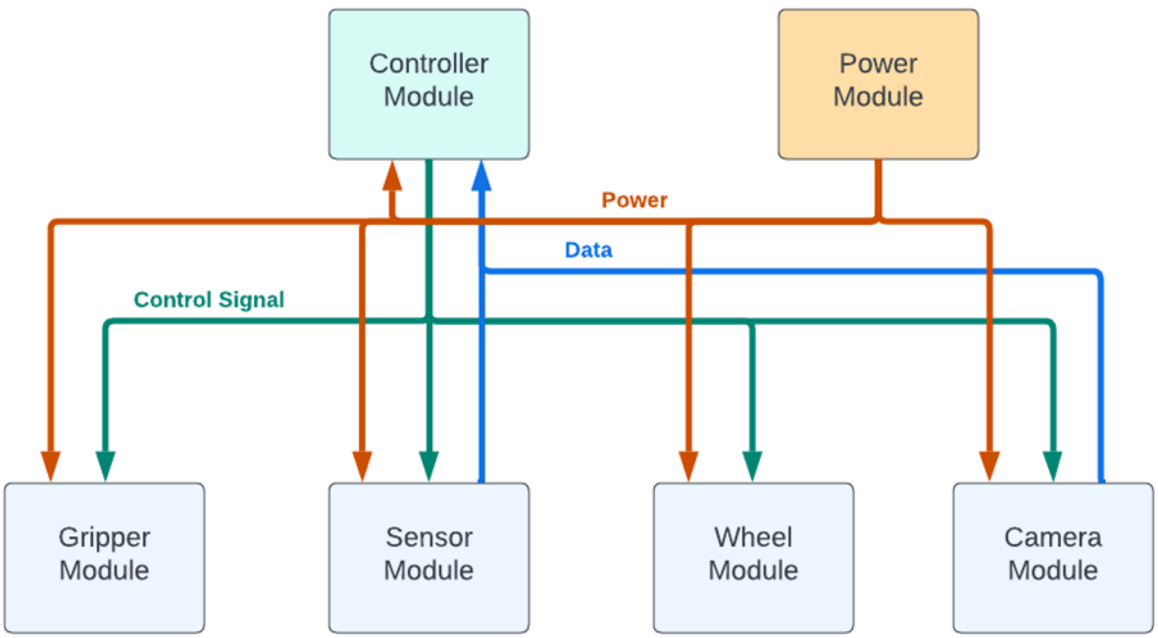

Figure 1 Modular Robotics Block Diagram

2. Introduction

The purpose of this document is to introduce a modularized robot that is designed and constructed in unique ways presented in the following sections. The goal of this project is to be able to build a configurable modular robot that is capable of accomplishing specific tasks programmed to it, while also minimizing the drawbacks of other designs. Our modular robot utilizes different types of modules connected together while each module carries a job in order to achieve the task specified.

2.1. Background

In this report, we would like to introduce robots which are machines made to carry and execute specific tasks with little to no human intervention. Various advancements have been made in the robotics field that includes but are not limited to automation, self-learning, self-reproduction, and many other. Every single type of advancement is backed by extensive research, and the one type we intend to potentially contribute to is modularizing.

Modular robotics is a newly introduced field that is receiving popularity, and its purpose is to modularize components of a robot in a way that makes the robot reconfigurable. Modules utilizing universal connection across the network unlock the ability which allows it to repurpose and achieve lots of different functionalities without the need to be completely redesigned and rebuilt. Many studies have been conducted in the field of modular robotics, and our research discovered many interesting applications and previous works, such as "Molecubes" which is a project done by a team from Cornell University. They produced a successful modular robot that is built from cubic modules which they referred to as "Molecubes" [1]. They presented the hardware and the software of their creation where there is an outline of their roadmap leading up to their final product. Cornell University Team's innovative approach was the diagonal rotation of each module, as explained in the article, which allows three printed circuit boards (PCBs) to be connected at all times. However, this approach adds to the complexity of constructing the modules, addition to having every module capable of all tasks is not only complex but redundant and expensive. Our goal is to minimize those drawbacks and achieve simplicity for the user to be able to build our modules with ease.

Another team from Switzerland proposed the idea of reconfigurable furniture which as implied utilizes the concept of modular robotics making the furniture smart and situationally changeable [2]. A work published in Nature Magazine took modular robotics to a whole different level, where they referenced biological organisms' self-reproductions [3]. Thus, introduced the idea of a modular robot capable of self-reproducing that consists of identical cubic modules and universal connectors. Allowing every module to think for itself and communicate with the other modules to accomplish a certain task cooperatively. A study done by Hossein Ahmadzadeh and his peers, published by J Intell Robot Systems, lists most of the contributions toward modularizing robots ordered by their characteristics [4]. All these projects are innovative, however, they go against the goal we are trying to accomplish with our modular robot.

2.2. Overview

The overview of our project is that we are aiming to construct a modular robot built from different generic types of modules, ideally similar in size but containing identical universal connectors. We plan to achieve a few things that include price reduction and simplification while still having the robot accomplish the intended tasks and behaviors specified by the user. Our project consists of various steps and different subsystems integrated thereby realizing our modular robot. The project's design and intended functionality are explained in detail in the following sections of this document. Briefly, the robot consists of six different types of modules that can be configured and connected in any way of the user's choice. Those modules are the camera module, controller module, wheel module, power module, ultrasonic sensor module, and gripper arm module. We are going for a rover-like robot as an illustration of the capabilities of our design, and our robot is going to be tested on the performance of executing programmed tasks. The flow of data from the camera module and the sensor module which provide the navigation information to the controller module. Using the information provided, the controller module will decide where the robot will go accordingly by sending signals to the wheel modules. And if the intended object was identified then the controller module signals the gripper arm module to pick it up and then drop it in the designated area.

2.3. Referenced Documents and Standards

[1]. Molecubes Extended: Diversifying Capabilities of Open-Source Modular Robotics, Cornell University.
[2]. Roombots—Modular Robots for Adaptive Furniture, Switzerland.
[3]. Self-reproducing machines, Nature Magazine.
[4]. Modular Robotic Systems: Characteristics and Applications, J Intell Robot Syst (2016).
[5]. A low cost, modular robotics tool carrier for precision agriculture research.

3. Operating Concept

3.1. Scope

Modular robots, referred to as “molecubes”, aim to decrease the cost for robot design and maintenance by dividing functions up between separate modules. Conventional robots lack flexibility and are difficult to integrate with new hardware as they are self-contained and designed for a specific purpose. By allotting functions to separate modules, these modules can be combined into various robots while simultaneously making each part easily replaceable.

3.2. Operational Description and Constraints

The proposed molecubes are intended to be used for creating a robot for almost any task simply by connecting the necessary modules. One robot instance could include a controller, camera, gripper, motors, and battery modules.

Requirements are as follows:

- Must operate autonomously, and be programmable for different tasks and configurations
- Modules must be able to connect to other modules and stay together during operation
- Modules must be designed to be relatively inexpensive and easily replaceable

3.3. System Description

- Raspberry Pi/Microcontrollers: The Raspberry Pi will be used to control and communicate with every module connected to a robot via serial communication and will be implemented in the controller module. It will be programmable for different tasks. Microcontrollers will be implemented to help communication between modules and the Raspberry Pi.
- Power Delivery: The power delivery will be implemented into the power module for delivering power to all other modules. It will consist of a battery management system for two LiPo batteries in parallel, and power converters for providing all components with correct voltages.
- PCBs/Housing: Each module performs a different task and will house different parts (motors or a camera). For that reason, each module will require a PCB specific to these parts. The housing will be a box used for containing the parts of each module, as well as providing connections on each face where necessary. These connections are for each module to be fastened together as well as communicate with each other.
- Computer Vision/Robot Arm: The Computer Vision subsystem will primarily make use of camera and sensor modules to allow a robot to make sense of its surroundings. The robot arm will allow a robot to lift and manipulate objects in its vicinity. It will be a simple arm consisting of three joints (shoulder, elbow, wrist) and a gripper.

3.4. Modes of Operations

A modular robot will be capable of operating in several manners defined by the user and the connected modules. It will operate autonomously to perform these defined tasks.

3.5. Users

A modular robot made up of these separate modules has the potential for surveillance environments with unknown conditions due to the low cost and modules being easily replaceable. This will be intended for users with a general knowledge of robotics due to the requirement of the robot being autonomous and programmable.

3.6. Support

Support will come in the form of a detailed instruction manual noting information for each module, use cases, and connections. Short tutorials will be included to give the user step-by-step guidance on creating a simple modular robot from a microcontroller, battery, and sensor module, as well as how to program and control it. These tutorials will grow in complexity as more modules are included.

4. Scenario(s)

4.1. Pick and Place Objects

The main scenario for our robot is to demonstrate the interoperability of our various modules to navigate a marked course to grab specific ping pong balls and transport them to a basket. Our robot will use the camera module and computer vision to autonomously navigate through a marked course as well as identify specific ping pong balls. The motor and wheel modules will propel the robot through the course. The gripper arm module will be used to grab the specified ping pong balls and place them in a basket.

5. Analysis

5.1. Summary of Proposed Improvements

- Modular, multiple parts(cubes) on a robot can be moved around and attached somewhere else and still perform the task at hand. So for example if you have a car, a car has 4 wheels, say the two passenger side wheels come off. The car can no longer drive. So move the rear driver wheel to the passenger side rear and now the car can semi drive. It is not optimal but it can still drive. Our wheel modules can be replaced and swapped around no problem.
- Cost effective, since all parts(cubes) are the same and interchangeable, though less complex than the dedicated robot hence cheaper. Current task dedicated robots are more expensive.
- Interchangeable, multiple parts(cubes) can be moved to another place on the robot and still perform the desired function. As opposed to a dedicated robot where you can not move parts around, it only works where it currently is placed.
- Improved fault tolerance, since parts(cubes) can be moved around and still function just fine, having a fault wouldn't be hard to figure out. Simple disconnect and retest each cube.

5.2. Disadvantages and Limitations

- Camera works best in lit conditions so if robot is in darkness then camera could have trouble detecting objects
- Lack of autonomy, Without autonomy, rigidity and non-back-drivability in coupling, large reconfigurable assemblies are not possible, and modular robots will remain vulnerable to the constraints of a real-terrain, where the ruggedness of an unstructured topology represents a major challenge for reliable mobility and maneuverability in practical applications.
- Maybe won’t be the absolute best at every task as opposed to a dedicated non modular robot that’s specifically designed for that task

5.3. Alternatives

This project has budding research in robotics that tries to optimize the price and scalability. Custom-made robots can be made specific to a task, but that will come at a cost.

Could have army of dedicated robots but this is costly and takes up more space to house these robots

5.4. Impact

Scalability, modularity, fault tolerance, interchangeability, cost-effectiveness, and ease of use.

Through using this modular robot, it would decrease the overall scale of the modules and even increase the overall size of the entire robot depending on how many are connected together so there is no one set size. So to save one's life by decreasing its size to fit in hard to reach human places would prove worth. The fact that the modules/cubes would be interchangeable would make fault tolerance high thus lessening the time to troubleshoot and ease of use. This would first aim to impact rural/unstructured areas as modular robots have not developed that kind of level of rigidity to wide variety or let alone treacherous terrain.

Modular Robotics: Molucube

Cole Butler
Ali Helmi
Andrew Hunt
Carlos Torres

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – Draft
4 December 2022

FUNCTIONAL SYSTEM REQUIREMENTS
FOR
Modular Robotics: Molucube

PREPARED BY:

Author Date

APPROVED BY:

Project Leader Date

John Lusher, P.E. Date

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	10/03/2022	Andrew Hunt		Draft Release
1	12/04/2022			403 Final Report

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Purpose and Scope

This project aims to present a modularized robot design that is reconfigurable and capable of performing specific tasks it was built for, while maintaining simplicity in design and cost effectiveness. This robot represents the possibility of future applications where multitasking robot is needed in limited resources situations, such as: rescue missions, space explorations, and warehousing. In order to achieve a robot capable of multitasking and reconfigurability, we are designing job-specific individual modules which are going to be connected and assembled to realize the bigger robot. Each module's purpose and scope are explained briefly in the following sections.

1.1.1. Camera Module

The purpose of the camera module is to identify and track relevant objects and navigate the path for the robot. Through a programmed python script developed using OpenCV and YOLO libraries, the module knows what to look for in real-time, as the robot and the program are running the camera module should continue to scan all it sees.

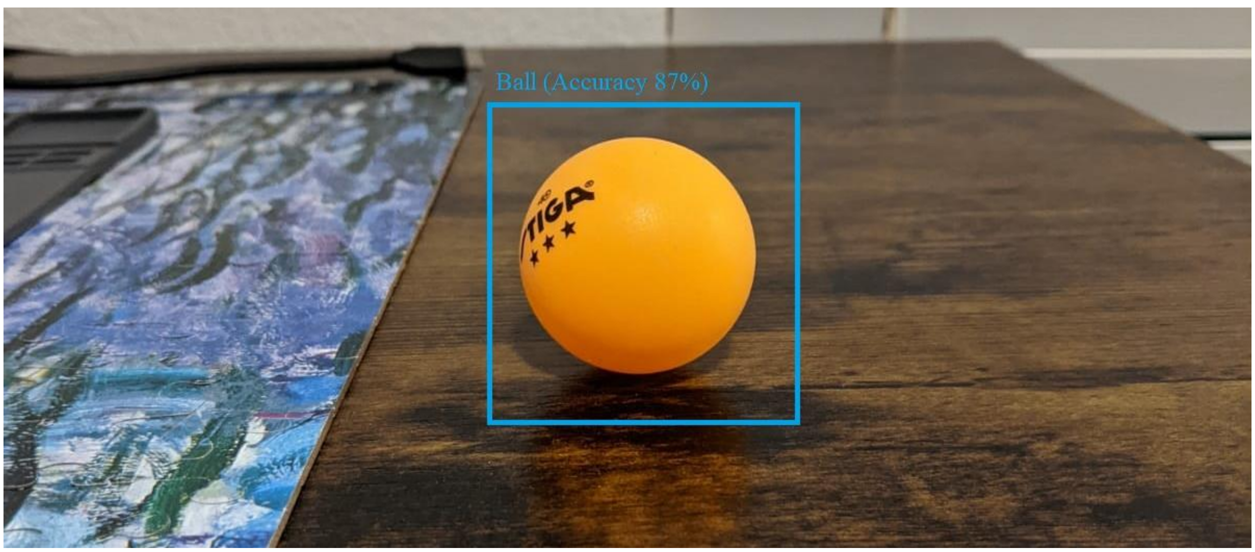

Figure 1 Example of object identification through the camera module lens.

1.1.2. Gripper Arm Module

The purpose of the gripper arm module is that its arm extends over the robot and reaches for the object identified by the camera and signaled by the controller. The gripper arm bends and reaches over the robot and picks up objects using servo motors mounted on the arm and controlled by the controller module.

1.1.3. Controller Module

The purpose of the controller module is to control the entire robot. Equipped with a Raspberry Pi, it sends commands or receives data with the microcontrollers of each module using serial communication.

1.1.4. Sensor Module

The purpose of the sensor module is to provide the robot with additional information about the environment. It uses a ultrasonic sensor to send distance data to the controller module.

1.1.5. Wheel Module

The purpose of the wheel module is to give the robot locomotion. It does this using a DC motor with a wheel attached. This is action is done by the usage of an H-bridge and a microcontroller to provide the logic needed for the mosfets to allow the motor to go backwards and forwards.

1.1.6. Power Module

The purpose of the power module is to supply each module of a robot with power. It contains LiPo batteries, a battery management system and power converters. This is all so that each module is provided with the correct voltage.

1.2. Responsibility and Change Authority

Team members are responsible for the completion of their respective subsystems. Requirement changes can be made only by the project sponsor, Swarnabha Roy.

Table 1 Division of Responsibilities

Subsystem	Responsibility
Power Delivery & Motor Driver	Carlos Torres
Raspberry Pi & Microcontrollers	Andrew Hunt
Computer Vision & Arm	Ali Helmi
Housing & PCBs	Cole Butler

2. Applicable and Reference Documents

2.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
14867857	2014	Multiple object detection using OpenCV on an embedded platform
18357982	2018	Real Time Object Detection and Tracking Using Deep Learning and OpenCV
384	2011	New object detection features in the OpenCV library

2.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
Doxygen HTML 4.6.0	4.6.0-dev Jun 2022	OpenCV Documentation

2.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

3.1. System Definition

This robot serves as a more affordable and accessible solution for users requiring a modular robot. It is manually reconfigurable which allows users to change its functionality to suit various applications. It consists of the following interconnectable modules: Controller, Power, Wheel, Camera, Gripper, and Sensor.

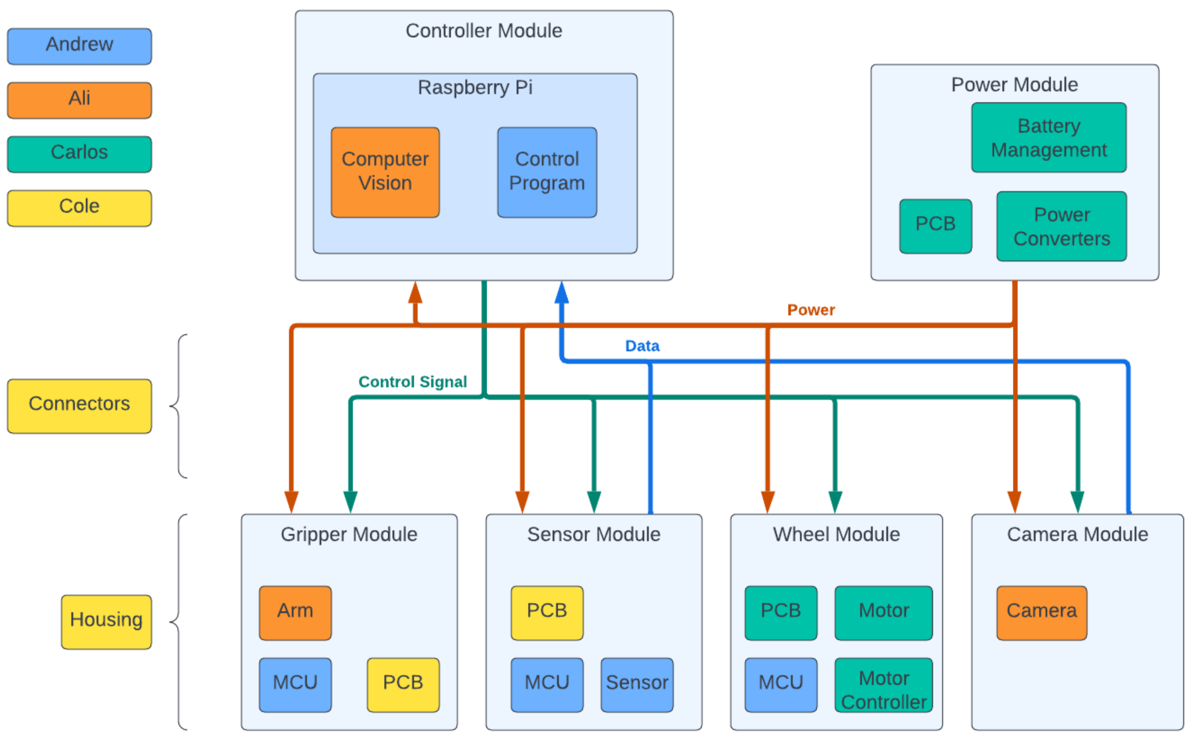

Figure 2. Block Diagram of System

The Controller Module contains a raspberry pi and is used to control the robot's functions including navigation, motion, and environmental detection. It uses serial communication to send commands to the microcontrollers in other modules to control them or receive data. It also uses USB to interact with the camera module.

The Wheel Module consists of a microcontroller, a motor controller, a motor, and a wheel. The microcontroller receives commands from the Controller Module to determine the direction and speed that the motor should run to move the robot where it needs to be to complete its task.

The Sensor Module contains a microcontroller as well as an ultrasonic sensor. The microcontroller collects distance data from the ultrasonic sensor and relays that information back to the Controller Module when requested.

The Gripper Module contains a microcontroller and a gripper arm. The Microcontroller receives commands from the Controller Module to move the gripper arm. The Gripper arm contains an actuator to raise and lower the arm, and a servo motor to move the gripper itself. The gripper arm is used to pick up objects.

The Camera module contains a camera that connects to the controller module via USB. The camera data is used by the controller module to locate and track objects; as well as gather information to aid in navigation.

The Power Module contains a battery management system as well as power converters to create the voltages needed to power the Raspberry Pi, Microcontrollers, and motors used in the robot.

3.2. Characteristics

3.2.1. Functional / Performance Requirements

3.2.1.1. Run Time

The robot needs to have decent run time. Something where it lasts a good amount of time before batteries need to be charged. So, 2 x 22.2V LiPo batteries will be placed in parallel.

Rationale: Placing batteries in parallel would increase the duration of the batteries and it keeps us from having to purchase one big battery and have to fit that. As opposed to 2 smaller sized batteries.

3.2.1.2. Locomotion

The robot shall have the ability to move itself around its environment.

Rationale: The robot needs to be able to move itself around the environment to complete the tasks required by the user

3.2.1.3. Computer Vision and Sensor Based Navigation

The robot shall use a camera and computer vision as well as ultrasonic sensors to determine what is in its environment to successfully navigate.

Rationale: The robot needs to be autonomous, which will require a camera and sensor to navigate and find its tasks.

3.2.1.4. Customizable

The robot shall be capable of varied module combinations to produce different robots.

Rationale: Dividing functions into separate modules allows for different combinations of functions. This increases the versatility, and potential use cases of a modular robot.

3.2.2. Physical Characteristics

3.2.2.1. Mass

The mass of the robot is TBD.

Rationale: We do not have a requirement for the overall robot to be a certain mass.

3.2.2.2. Volume Envelope

The volume of each module can vary depending on its intended purpose, but the starting dimensions will be around 4”x4”x3” (LxWxH).

Rationale: As this is a modular robot, a module could be nearly any size to fit its purpose, but controller, power, and blank modules are intended to be the above-mentioned size. This size was chosen as it fits around the Raspberry Pi and leaves room for wires to go to the side connectors

3.2.2.3. Module Connections

Modules shall be able to connect to each other and stay together during use. They must be able to come apart at the discretion of the user.

Rationale: If modules were to come apart during use, then a robot in use could lose parts and functionality. If the user cannot easily disconnect a module, then maintenance and modularity of a robot would be difficult.

3.2.3. Electrical Characteristics

3.2.3.1. Inputs

a. Input levels for the robot would be 22.2v provided by 2 x 22.2v lipo batteries placed in parallel.

Rationale: By design, should improve battery run time and series would only provide more voltage which we don't need.

3.2.3.1.1 Power Consumption

a. The highest power for the entire system shouldn't go over 200 Watts.

Rationale: The heaviest load on this power delivery module are the motors and those need a max current (stall) of 9A so there should be any more power needed than this

3.2.3.1.2 Input Noise and Ripple

The input noise for the robot produced by the buck converters would be an issue to data collection relayed to the Raspberry Pi but this would be hammered out in the PCB design.

Rationale: This would need to be situated in the PCB design by placing the converters and data traces far from each other to minimize noise

3.2.3.1.3 External Commands

The robot may receive external commands and/or configurations from the user via wireless communication.

Rationale: Wireless configuration through the Raspberry Pi will make it easier to re-program or manually control the robot, rather than having to disassemble it to access the SD card.

3.2.3.2. Outputs

3.2.3.2.1 Diagnostic Output

The robot may have a log file stored on the raspberry pi in order for the user to diagnose configuration and programming issues.

Rationale: The robot is configured and programmed by the end user, so a file logging what the robot is doing will be helpful for troubleshooting issues

3.2.3.2.2 Raw Video Output

The real-time video footage captured by the camera module will be streamed to the controller module running the object identification program.

Rationale: The program should be able to identify objects of interests and signal the robot to take actions.

3.2.3.3. Voltage Outputs

The Power Delivery module will be utilizing the buck converters to drop the 22.2v input voltage to the desired following output voltages: Raspberry Pi = 5v, Wheel Motors = 12v, Gripper arm linear actuator = 12v and Gripper servo = 5v.

Rationale: These are the operating voltage requirements provided on the data sheets.

3.2.4. Environmental Requirements

The path we want the robot to travel through should be marked by tape or any kind of distinctive markers that can be recognized by the camera.

Rationale: The robot is going to be able to avoid obstacles, but in order for it to not wonder aimlessly we need to guide its path and see how it navigates through.

3.2.5. Failure Propagation

The modular robot shall not allow propagation of errors outside the scope of the presented system interface.

3.2.5.1. Failure Detection, Isolation, and Recovery (FDIR)

3.2.5.1.1 Camera Failure Detection

The written program is made to throw errors in case of any failure occurrence, such as failing to connect with the webcam.

3.2.5.1.2 Built In Test (BIT)

The controller module shall have scripts programmed on it which include failsafe programs that ensures the correctness of the robot's performance.

3.2.5.1.3 Communication error detection

The controller module and microcontrollers will have code to detect when a communication error may have occurred and will request confirmation of the last data frame.

4. Support Requirements

To program the robot, the user will need to use a computer with a terminal that can access the Raspberry Pi through either Bluetooth or Wifi. The user may also remove the SD card from the Raspberry Pi to add programs directly. All robot designs will require at least one controller module and one power module.

Appendix A: Acronyms and Abbreviations

BIT	Built in Test
LiPo	Lithium Polymer
PCB	Printed Circuit Board
TBD	To Be Determined
USB	Universal Serial Bus

Appendix B: Definition of Terms

Computer Vision Computer’s ability to understand and interpret reality through image or videos.

Modular Robotics: Molucube

Cole Butler
Ali Helmi
Andrew Hunt
Carlos Torres

INTERFACE CONTROL DOCUMENT

REVISION – Draft
4 December 2022

INTERFACE CONTROL DOCUMENT
FOR
Modular Robotics Molucube

PREPARED BY:

Author Date

APPROVED BY:

Project Leader Date

John Lusher II, P.E. Date

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	10/03/2022	Andrew Hunt		Draft Release
1	12/04/2022			403 Final report

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

1. Overview

This document will provide information on how each module will interface with other modules, as well as how the user will interface with the robot itself. It also contains information on the expected size of modules as well as how the robot will be powered.

2. References and Definitions

2.1. References

Multiple object detection using OpenCV on an embedded platform (14867857)
22 Oct 2014

Real Time Object Detection and Tracking Using Deep Learning and OpenCV (18357982)
12 Jul 2018

New object detection features in the OpenCV library (384)
10 Sep 2011

2.2. Definitions

LiPo	Lithium Polymer
OpenCV	Open Source Library for Computer Vision
SSH	Secure Shell
TBD	To Be Determined
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
YOLO	You Only Look Once
PCB	Printed Circuit Board

3. Physical Interface

3.1. Weight

The weight of the robot is still TBD, as no weight requirement has been set.

3.2. Dimensions

The overall dimensions of the robot may vary depending on the application and the configuration for the task we want to accomplish. However, generally each module's dimensions should be 4”x4”x3” (LxWxH).

3.3. Mounting Locations

One of the main features and characteristics of the modular robot is that it can be reconfigured and mounted differently based on the task at hand.

4. Thermal Interface

To prevent the Raspberry Pi from thermal throttling or potentially being damaged from overheating, it will be equipped with a heat sink.

5. Electrical Interface

5.1. Primary Input Power

Primary input power will be 22.2v DC provided by 2 x 22.2v lipo batteries placed in parallel with each other. This will be connected to the battery management system to ensure battery safety and proper charging and discharging. Then voltage will be dropped by using buck converters to obtain 12v, 5v and 3.3v outputs that would then be distributed to the rest of the robot.

5.2. Current Levels and Voltages

The voltages required by the Wheel motors, Gripper Linear Actuator, Gripper Servo and Raspberry Pi are 12v, 12v, 5v and 5v respectively. The current draw from the motors at max (stall) will be 9A but we will run them safely at 5A as well as the linear actuator. The others don't draw as much current since they have low resistances. The Buck converters would be able to produce a max output of 2A on the 3.3v and 5v outputs and 5A on the 12v output rail.

5.3. Signal Interfaces

The Raspberry Pi will communicate with other modules via the UART serial communication standard. Communication with the camera will be done via USB.

5.4. User Control Interface

There will be no specific user interface with the robot, as it is intended to be autonomous. Users can interact with the robot either through directly editing the SD card to add user-created programs, or through an SSH connection via either WIFI or Bluetooth to transfer or write new programs to control the robot.

6. Communications / Device Interface Protocols

6.1. Wireless Communications

The Raspberry pi will have a WiFi module using IEEE 802.11 standards, and/or a Bluetooth module using Bluetooth 5.0 for remote configuration of the controller module.

6.2. Video Interface

The camera module will be responsible for providing the video footage real-time to the controller module, through a USB connection, where the video will be processed and objects will be identified.

6.3. Device Peripheral Interface

Intermodule communication will be handled by a UART serial connection, and the camera will be connected via USB.

Modular Robotics: Molucube

Cole Butler
Ali Helmi
Andrew Hunt
Carlos Torres

SCHEDULE AND VALIDATION PLAN

REVISION – 1
4 December 2022

Table of Contents

1. Schedule

Task	4-Oct	11-Oct	18-Oct	25-Oct	1-Nov	8-Nov	15-Nov	22-Nov	29-Nov
Motor Driver and Power Delivery Designs Complete									
Rpi MCU circuits designed									
Computer Vision design completed									
Housing Design completed									
Power Delivery Motor Driver Parts Ordered									
Rpi & MCU prototype boards created									
Computer vision parts ordered									
Housing/Connector Parts Ordered									
Power Delivery Motor Driver Start PCB design									
RPi & MCU PWM Signal generation									
Computer Vision Python script									
Housing/Connectors Assembled & Tested									
Power Delivery Motor Driver PCB design complete									
Rpi & MCU Motor Controller testing and UART signal generation									
Training and Testing Model on a saved video									
PCB Design Completed/Parts Ordered									
Power Delivery Motor Driver PCB Components ordered									
MCU Network Test									
Making program able to detect real time on live feed									
Boards Assembled									
Power Delivery Motor Driver order and test on perfboard									
Sensor Testing									
Adjusting arm's design to match robot's modules sizes									

Testing/Toublshooting Boards									
Power Delivery Motor Driver Solder PCBs									
Printing and Assembling Arm									
Connection Between Boards									
Power Delivery Motor Driver buffer week/testing/troubleshoot									
Testing functionality of both systems and troubleshooting									
Buffer Week for Any Additional Troubleshooting									
Demo and work on final Report									

2. Validation Plan

Test Name	FSR Section	Success Criteria	Methodology	Status	Assignee
Output Voltage Test	3.2.3.3	Correct voltage being outputted to motors, RPI, servo and actuator	An oscilloscope will be used to measure the voltage reading and will be compared to the expected value	Partial Success	Carlos Torres
Battery Management System Test	3.2.3	Battery can charge and discharge properly	An oscilloscope will be used to view voltage of battery while a test motor is run off it.	Untested	Carlos Torres
Wheel Motor Test	1.1.2	Wheel motors can go forwards and backwards	Provide motor with power and send logic to MCU	Partial Success	Carlos Torres
Microcontroller Initial test	1.1	Microcontroller runs basic program	A debugger and LED output will be used to determine correct functionality	Success	Andrew Hunt
PWM Signal	3.2.1.2	Microcontroller generates PWM signal	A logic analyzer and Pulseview will be used to determine correct signal	Success	Andrew Hunt
Motor Control	3.2.1.1	MCU can control motors	MCU I/O will be connected to motor controller	Partial Success	Andrew Hunt
UART Communication	1.1	Raspberry Pi and Microcontroller can communicate	Raspberry Pi will instruct MCU to turn on LED	Success	Andrew Hunt
MCU Network Communication	1.1	Raspberry Pi communicates with multiple MCUs	Raspberry Pi will instruct each MCU to turn on/off LEDs	Success	Andrew Hunt
Sensor Test	3.2.1.3	MCU able to use sensor to detect object	An object will be placed in front of the sensor and the MCU will send a ping and wait for a reflection	Success	Andrew Hunt
Object Detection	3.2.1.3	Successfully identifies ping pong ball and keeps tracking it	Python script uploaded and executed through webcam	Success	Ali Helmi
Path Navigation	3.2.4	Successfully identifies duct tape and maps path in between	Python script uploaded and executed through webcam	Untested	Ali Helmi

Gripper Arm Functionality	3.2.1.2	Assembled and compatible with mounted motors and capable of simple movement	Picks up and drops ping pong ball	Partial Success	Ali Helmi
Housing/Connectors	3.2.2.3	Module Housing easily connects, and resists being pulled apart	Apply force to connected housing and see if it stays together	Partial Success	Cole Butler
Board Functionality	3.2.2.3	No incorrect readings for boards of all modules	Probe board while providing it with power and logic	Untested	Cole Butler
Connection Between Boards	3.2.2.3	Boards successfully connect and can communicate	Probe between both boards for continuity	Untested	Cole Butler

3. Performance on Execution Plan

Most of the execution plan was completed. Tasks not currently complete are due to issues that arose during testing. The project is mostly completed and anything not currently complete will be completed over the break or very early on next semester.

4. Performance on Validation Plan

Most of the validation plan was completed. There are still some issues with the power delivery and motor subsystems, as well as with the gripper arm, but these issues will be sorted out either before next semester or very early in the semester so the subsystems can all be successfully integrated into the entire system.

For further information on Power Delivery Subsystem and Motor Driver Subsystem issues please follow. The Battery Management System (BMS) part of the power delivery subsystem had a copy and pasted issue for all 6 cells where one of the MOSFETs was flipped where the drain and source were flipped and thus causing the external battery charger to not charge the batteries. The Overcurrent and voltage IC on cell 6 ended up exploding the trace so the PCB wasn’t salvageable for other testing.

The buck converter part of the power delivery subsystem was semi working up until I attempted to fix the shorted 12v power line and in the process blew D2 which is the diode for the 12v line along with some other traces connecting to 5v and 3.3v lines so further testing other than validating the 3.3v output voltage was not able to be completed. It was found that the diode was “wired” wrong in the PCB design, and this will be sorted out over Christmas break in time for ecen 404.

The H-Bridge/Motor Driver was able to be validated only in the forward direction but not reverse direction. It turns out that the gate threshold voltage to turn on the two NMOSs were not being met thus they weren't being turned on. Again, this will be calculated and fixed in the PCB design2.0 over Christmas break.

Modular Robotics: Molucube

Cole Butler
Ali Helmi
Andrew Hunt
Carlos Torres

SUBSYSTEM REPORTS

REVISION – Draft
4 December 2022

SUBSYSTEM REPORTS
FOR
Modular Robotics: Molucube

TEAM 10

APPROVED BY:

____
Project Leader Date

____
Prof. Kalafatis Date

____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	12/04/2022			Draft Release

Table of Contents

List of Tables

List of Figures

1. Introduction

The modular robot will be able to autonomously navigate a course to perform a desired task, in this case, collect ping pong balls and deliver them to a basket. The robot is broken down into four subsystems: Power Delivery and Motor Driver, Raspberry Pi and Microcontrollers, Housing and PCBs, and Object Detection and Gripper Arm.

2. Power Delivery and Motor Driver

2.1. Subsystem Introduction

Power Delivery Subsystem

- This subsystem is split into two further systems, the Battery Management System (BMS) and the Buck Converters. To ensure safe battery external charging and regular use discharging the BMS was needed. It would connect to an external Lipo battery charger and then the balancing cables from the Lipo battery packs would connect to the BMS as well. This ensures the battery cells all charge and discharge at the same rate. The BMS was tested and during testing a mosfet on the PCB blew. So, it was not able to be tested for battery charging and discharging validation as this was a PCB design flaw and couldn’t be fixed in time. This would be further discussed in the failure analysis portion section below.

- As for the buck converters I was only able to test the 3.3v and 5v power outputs as the 12v power output rail was shorted and was narrowed down to a design flaw in the PCB where a diode was shorted. However, I was only able to get voltage outputs from 3.3v and 5v lines via multimeter since it was tested at my house during thanksgiving break. Once I got back to lab, I did some adjustments to get the 12v line working and ended up frying the board and it's useless now. This too will be further discussed in the failure analysis section.

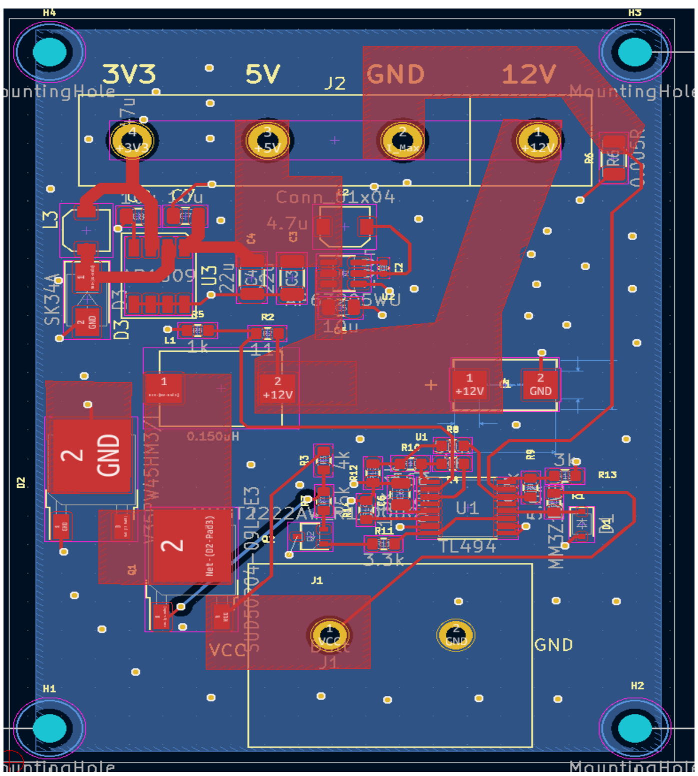

Figure 1 Buck Converters PCB Design

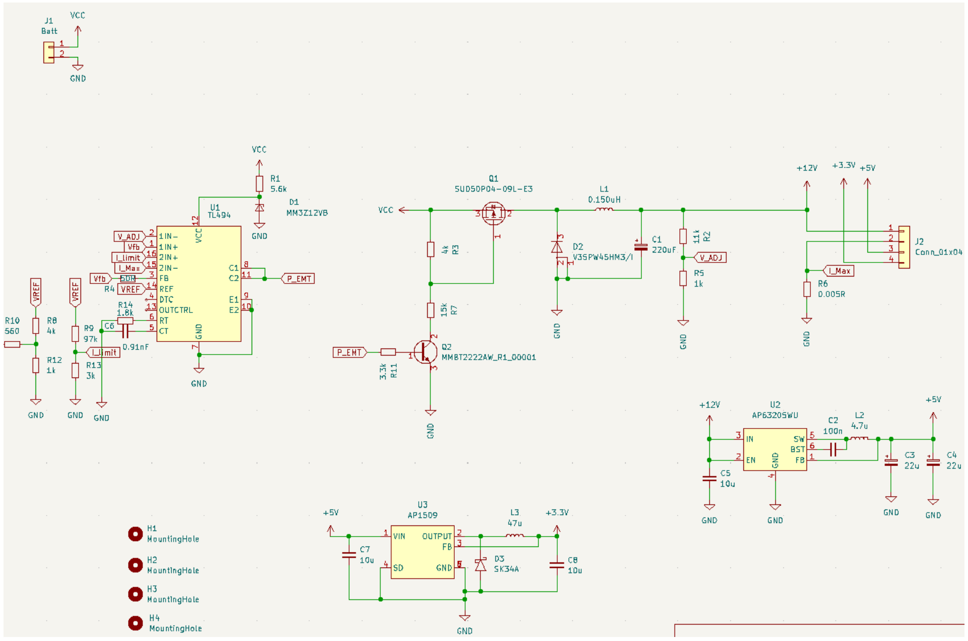

Figure 2 Buck Converters Circuit

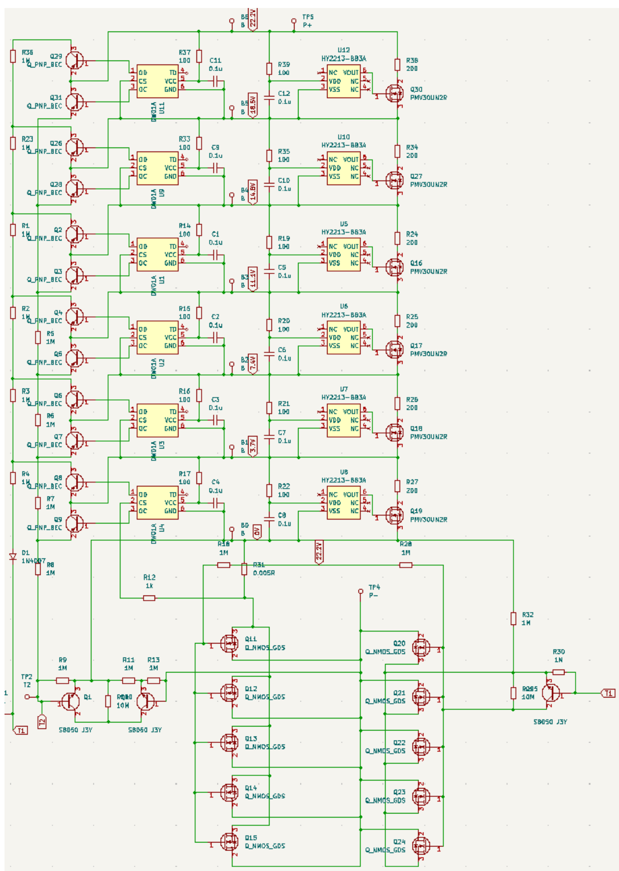

Figure 3 BMS Circuit

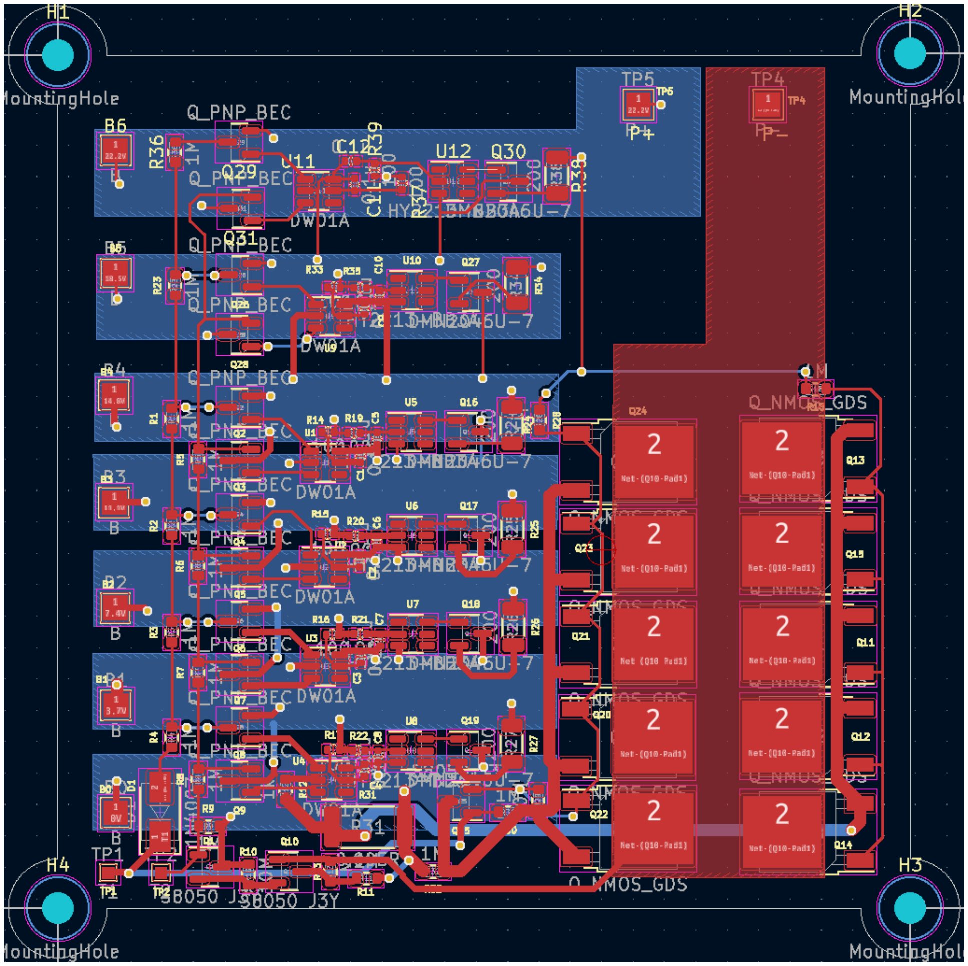

Figure 4 BMS PCB Design

Motor Driver Subsystem

- The Motor Driver subsystem would be taking in the power from the 12v buck converter as well as the 3.3v for the microcontroller and then using the logic from the microcontroller would allow for the H-bridge to control the direction of spin of the motor. This was tested and only forward direction of the motor was achieved, and it was later found that the NMOSs were not being turned on due to gate voltage not being high enough. This will be discussed further in the failure analysis section.

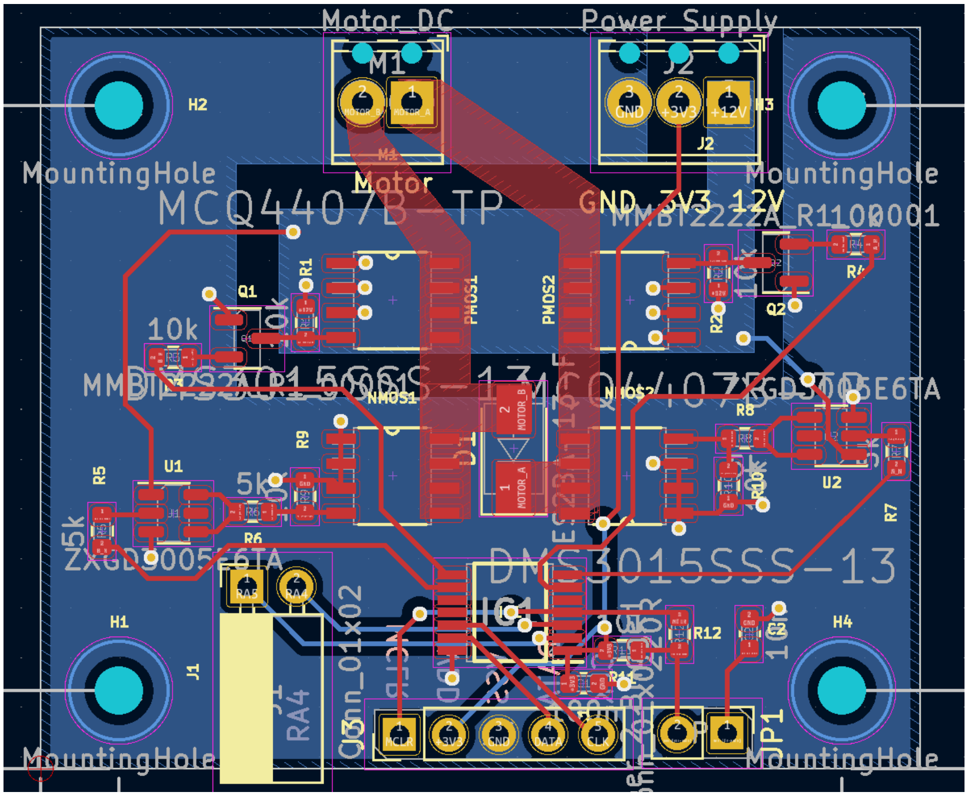

Figure 5 H-Bridge PCB Design

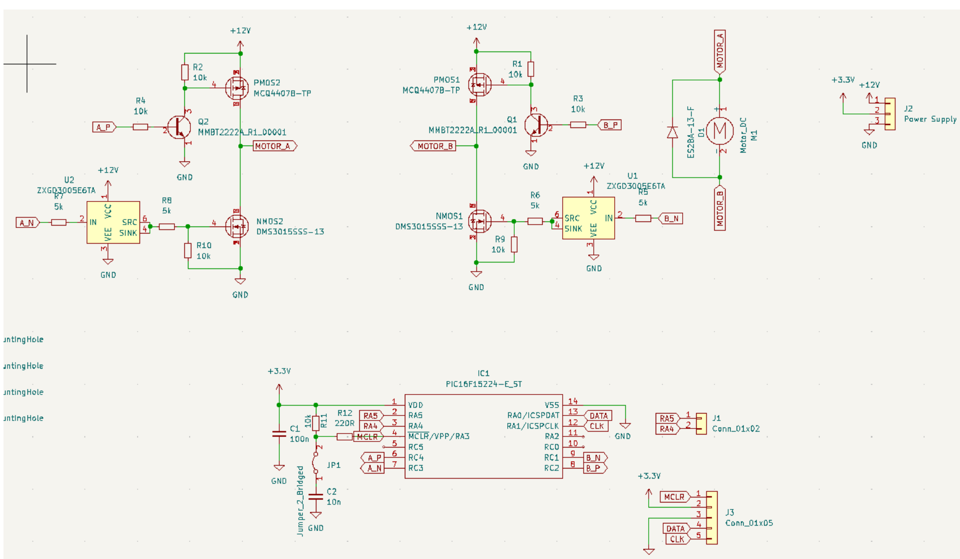
Figure 6 H-Bridge Circuit

2.2. Subsystem Details

Power Delivery Subsystem

- Two 22.2V Lipo batteries placed in parallel would be fed into the buck converters where then the voltage would be dropped or “bucked” down to 12V with a full load current of 30A then 5v and then 3.3V both with full load currents of 2A each. The DC wheel motors and gripper arm linear actuator both need 12v to operate, the raspberry pi and gripper arm servo need the 5v and the various microcontrollers need the 3.3v to operate. A major challenge was being completely down on the 12v line. As the 5v and 3.3v outputs relied on the 12v line to work the entire system was down. I was forced to bypass the 12v line and force 12v on to the 5v line that would be passed to the 3.3v line to work.

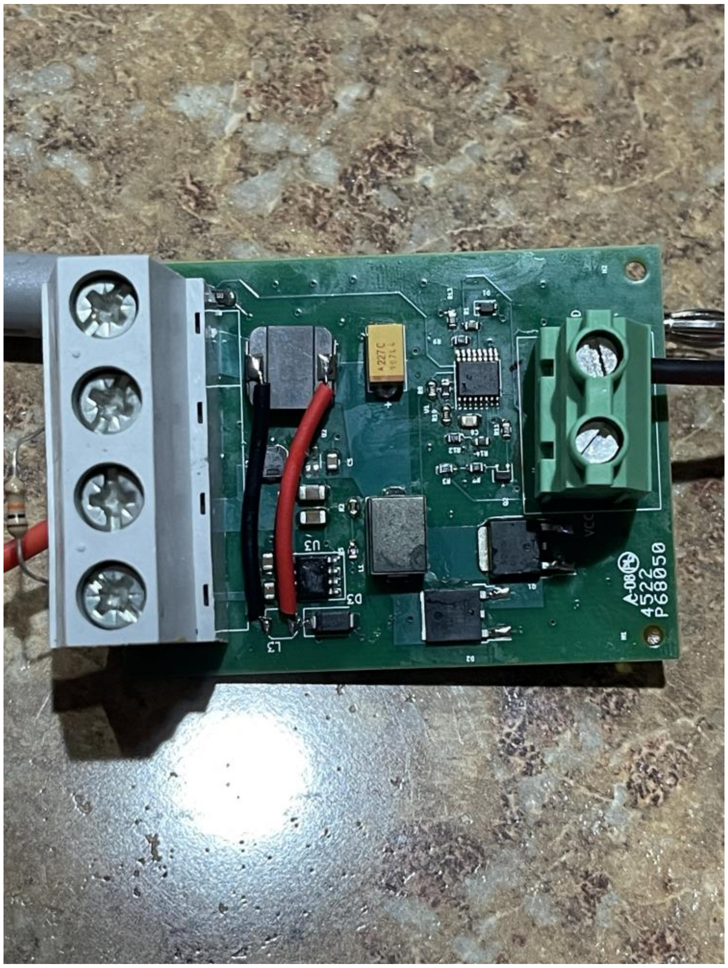

Figure 7 Physical Buck Converters PCB

- The BMS would take in the range of .1A-10A and 150W charging from the external charger and then the balance wires from the Lipo batteries themselves would be connected to the BMS PCB where the BMS would make sure all cells charge and discharge at the same rate ensuring battery health and safety. Each cell contains 3.7V which totals to 22.2v for 6 cells.

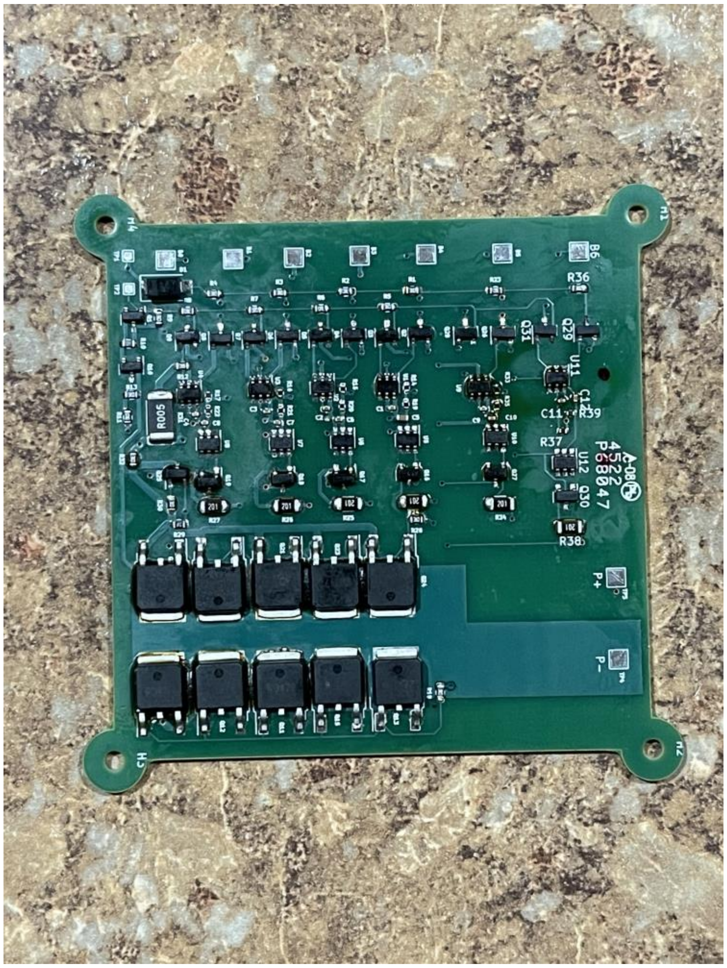

Figure 8 Physical BMS PCB

Motor Driver Subsystem

- The Motor Driver subsystem receives power for the wheel motors from the 12v output of the buck converter and then uses the 3.3v output line for the microcontroller. Logic from the microcontroller is used to control the “gates” of the H-bridge to then control which direction the DC motor spins. The motors are rated at a stall current of 9A but since our robot has four motors and the robot itself isn't heavy, we wouldn’t see the motor reach that 9A, so I designed the motor driver to operate at a safe and reasonable 5A for the motor.

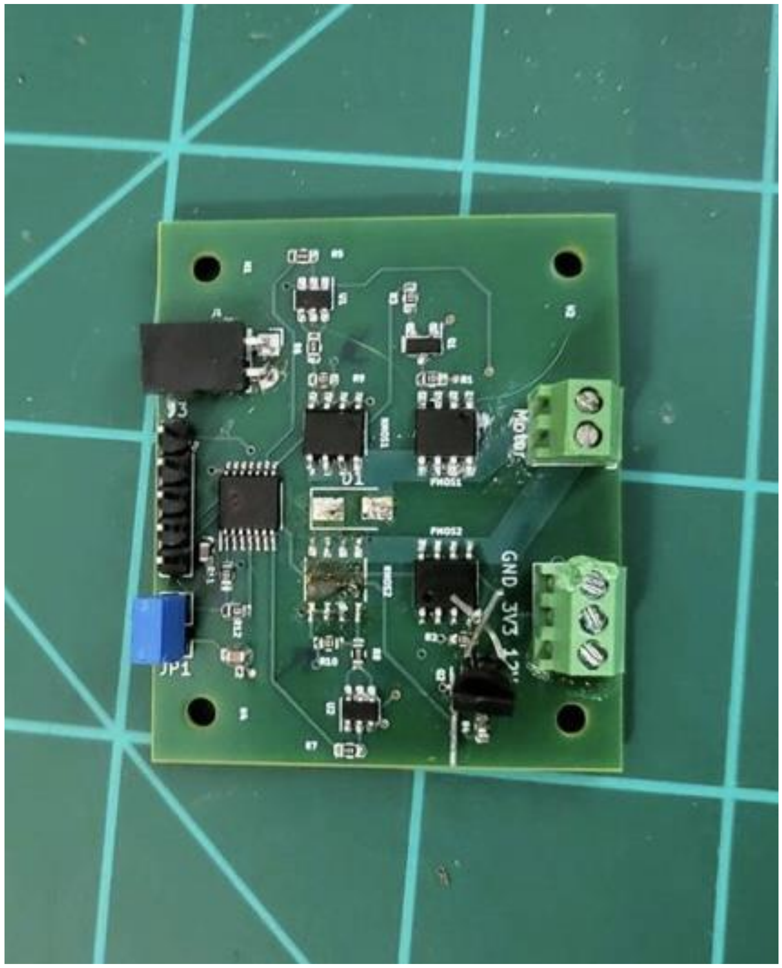

Figure 9 Physical H-Bridge PCB

2.3. Subsystem Validation

Power Delivery Subsystem

- In the introduction of my subsystems, I mentioned that I was able to get the 3.3v and 5v outputs of the buck converters to work but only was able to measure the output voltage using my handheld multimeter back at home during thanksgiving break I was unable to test for actual output current, voltage noise and efficiency until I was back in lab. Once I was back in lab, I attempted to fix the 12v line and in the process blew the board, so I was unable to salvage what was left. A proper buck converter should be able to produce the same voltage across a range of load hence why Dr.Lusher wanted us to use the eLoad in lab to test this. As well as operating up to a certain current. Since the 3.3V buck converter runs off of the output of the 5v line I couldn’t really vary the input voltage as it would always be 5v give or take a few mV.

Table 1 3.3V buck converter voltage data

INPUT VOLTAGE	OUTPUT VOLTAGE	INPUT CURRENT
4.98V	3.31V	.03A

Figure 10 3.3V Buck Converter Output Voltage Validation

- The BMS would

Motor Driver Subsystem

- The Motor Driver subsystem was able to be semi validated. I was able to get the motor to turn but only in one direction. It was found that both NMOSs weren't being turned on so bypassing those and solely using the two PMOSs to turn the motor for demo was done.

2.4. Failure Analysis of Subsystems

To start, the buck converter PCB was semi functional at one point but out of desperation to get the 12v line fixed in time for demo I fried the board. After further and extensive analysis of the PCB design it was found that D2, which is a diode shown below, was "wired" wrong. The output/large pad of the diode was supposed to be connected to the 12v polygon line but instead I connected that large pad to ground and the 12v polygon line was connected to pin 1 which is connected to pin2 which was connected to ground which is why I was getting conductivity between the 12v output and the ground input thus a short. This will require me to redesign that small portion of the PCB design and reorder the board over Christmas break along with putting the correct footprint of the inductor for the 3.3v output as I ordered an inductor that was 3 times the size of the pad and so I had to solder wires from the pads to the inductor to make it work for demo.

For the BMS some small minor adjustments need to be made. When trying to connect my BMS to the external battery charger the charger told me “Reverse Polarity” at first, I didn’t know what this meant exactly then when I disconnected my circuit from the charger to diagnose, my U11(battery management IC) component for cell 6 sparked then smoked. It had burnt the traces inside the board and I couldn’t test/validate. It was found that I had a mosfet flipped and the drain was connected to BATT+ when it was supposed to be to the source. U11 oversees overcurrent protection and voltage. And I was missing a 1k ohm resistor on pin 2 of U11. Both problems were minor but I made this mistake on all 6 cells so I must go back and implement these fixes on the BMS PCB design for all 6 cells. Which would be done over Christmas break and then validated as soon as possible for 404.

Lastly, for the motor driver like I mentioned before, only the forward direction was able to work but not reverse direction. This is because the NMOSs of the H-bridge weren't being enabled so I modified the circuit to only run off of the PMOSs and got the motor going in one direction. After trouble shooting the issue, I found that the gate threshold voltage of the NMOSs was 2.5v and after viewing the voltage being measured at the gate, I found that it was about 1.7v. So im going to recalculate my values over Christmas break and have a working final PCB completed by 404.

3. Microcontrollers and Raspberry Pi

3.1. Subsystem Introduction

This subsystem consists of a Raspberry Pi, which is the main controller of the robot, as well as several microcontrollers, which control and serve as an interface between the Raspberry Pi and the motor, sensor, and gripper arm modules.

3.2. Subsystem Details

3.2.1. Raspberry Pi

A Raspberry Pi 4 serves as the main controller for the entire robot. While the computing power of the Raspberry Pi 4 is not necessarily needed for this subsystem, it is necessary in order to have enough performance for the Object detection subsystem. It does though have several features needed for this subsystem. It supports Wi-Fi as well as Bluetooth which will be necessary for communicating with the robot remotely. When the robot is able to connect to a known Wi-Fi network, it is possible to transfer files and update the software for the robot. If a known Wi-Fi network is unavailable, the Raspberry Pi is set up to support serial over Bluetooth which allows terminal access to run or edit programs as necessary.

The Raspberry Pi will eventually serve a larger purpose of running all of the control logic for the robot, which will allow it to navigate, locate objects, pick them up, and place the objects in a desired location. For 403, the Raspberry Pi mostly serves as an interface between the microcontrollers and a laptop in order to test the desired functionality before integration. It currently runs python scripts which talk to the microcontrollers via UART to get them to perform the desired functions.

For now, the Raspberry Pi consists of several test scripts that use a communication class that allows for easy communication with the microcontrollers. The class takes the desired address of the microcontroller as input, as well as an array of bytes intended to be sent as data. The class takes these inputs and constructs a data frame to be sent via UART. The class is also able to receive data from the buffer and return the data it receives.

3.2.2. Microcontrollers

The microcontrollers used are the PIC16F15224. These microcontrollers have the features needed for the various modules, including UART and PWM while also being a low cost to meet the goal of making a more affordable robot. The firmware of each microcontroller is mostly the same. They all run the same main program, and the type of module and communication address is chosen at compile time, which determines which other functions will be included by the compiler.

3.2.2.1. Communication

The microcontrollers communicate with the Raspberry Pi via UART. It was originally planned to use the RS-485 protocol for communication, but this requires the ability to send 9 bit words, which the Raspberry Pi is unable to do. Instead, 8 bit words are used with the most significant bit being reserved to indicate an address byte. Data is sent in frames consisting of an address byte, various numbers of data bytes, a check byte, and then an end byte, consisting of the number 126 in decimal as shown in the table below.

Table 2 Microcontroller data frame

Address Byte	Data Bytes	Check Byte	Final Byte
Value > 128, MSB + MCU Address	Data Values < 128	XOR of data bytes	126

While UART is normally point to point communication, the robot will require a bus with multiple microcontrollers. This is accomplished by having the microcontrollers send their transmit pins to a tristate mode so they will not fight with each other when one of them is transmitting back to the Raspberry Pi. Each Microcontroller will listen to each data frame, but only take action and respond if the frame is being sent to the microcontroller's address.

The Raspberry Pi and Microcontrollers communicate at a baud rate of 115200 bps to ensure fast transmission of data frames so there will not be a large gap between commands sent to motor modules.

3.2.2.2. Motor Drivers

Pulse Width Modulation is required to control the speed of the wheel motors, as well as select the angle for the servo motors in the gripper arm module. The PIC16F15224 has two PWM channels which allow for setting various pulse frequencies as well as the duty cycle. This allows for setting the correct frequency to match the required switching speed for the transistors of the motor driver.

The motor driver controller waits for a data frame from the Raspberry Pi which will tell it which direction to move and what speed to use. When this information is received, the microcontroller sets the desired values and sends a message back to the Raspberry Pi to verify it received the command. The different modes for the motor driver are: Forward, Reverse, Coast, and Brake. Forward and reverse also require a speed from 1-100 which is a percentage of how much power to use. Coast allows the motors to spin freely. Brake will cause the motors to resist motion.

When changing speeds, the motor driver is able to instantly change to the desired speed. When changing direction, the motor driver firmware delays several microseconds between stopping one direction and starting the other in order to allow all of the transistors to switch to the new configuration to prevent shorting directly to ground.

3.2.2.3. Sensors

The sensor module requires the use of a timer as well as an interrupt. The microcontroller must send a 10μs pulse to the ultrasonic sensor. The sensor measures the distance to a nearby object and then returns a pulse to the Microcontroller, the length of which indicates how far away an object has been detected. The length of this pulse in microseconds can be converted to inches or centimeters by dividing by a constant based on the speed of sound. The rising and falling edges of the pulse is detected by the microcontroller with a pin interrupt, and the length of the pulse is measured by a timer.

The sensor firmware currently constantly polls the ultrasonic sensor every 10ms and sends the distance data back to the Raspberry Pi. For integration with the entire robot, where there will be multiple sensors, this will likely have to be switched to the Raspberry pi requesting the sensor do a measurement and return the distance, in order to avoid any potential interference between different sensors.

3.3. Subsystem Validation

3.3.1. UART Communication

UART Communication was validated with a logic analyzer to determine the correct data was being sent and received. As shown in the figure below, the Raspberry Pi was able to send a data frame telling the microcontroller to set the motor mode to forward, and to set the speed to 55%. The Microcontroller received this frame, changed the percentage of the PWM, and sent back the original check byte to verify to the Raspberry Pi that it received the correct command. The entire process of sending the command, changing state and sending the reply takes around 850 μs.

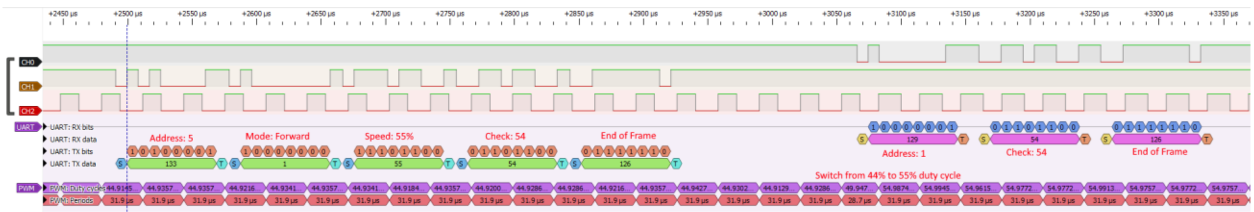

Figure 11 Data frame send and receive test

The validation for communicating with multiple microcontrollers on a bus was also validated with a logic analyzer as shown below. The Raspberry Pi sent out a data frame to each microcontroller's address, telling each to toggle an LED connected to one of the I/O pins. Each microcontroller only responded when its address was sent. This entire process took around 2.2 ms. Extrapolating out to multiple motor drivers, the process of updating each wheel motor could be expected to take around 3.5 to 4.5 ms, which should not cause a noticeable gap between the motors changing modes.

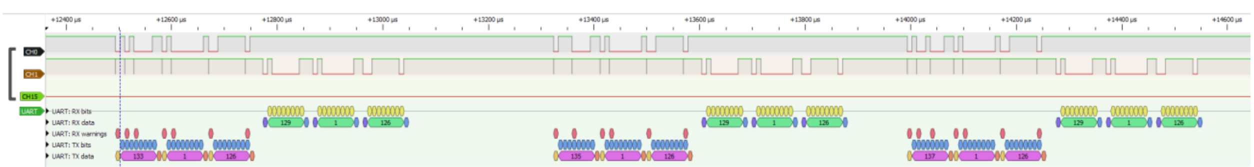

Figure 12 MCU network test

Validation for individual addressing of each microcontroller was done visually with a python script running on the Raspberry Pi. The script asks for an address as input. It then takes this address and sends the command to the desired microcontroller with the communication class.

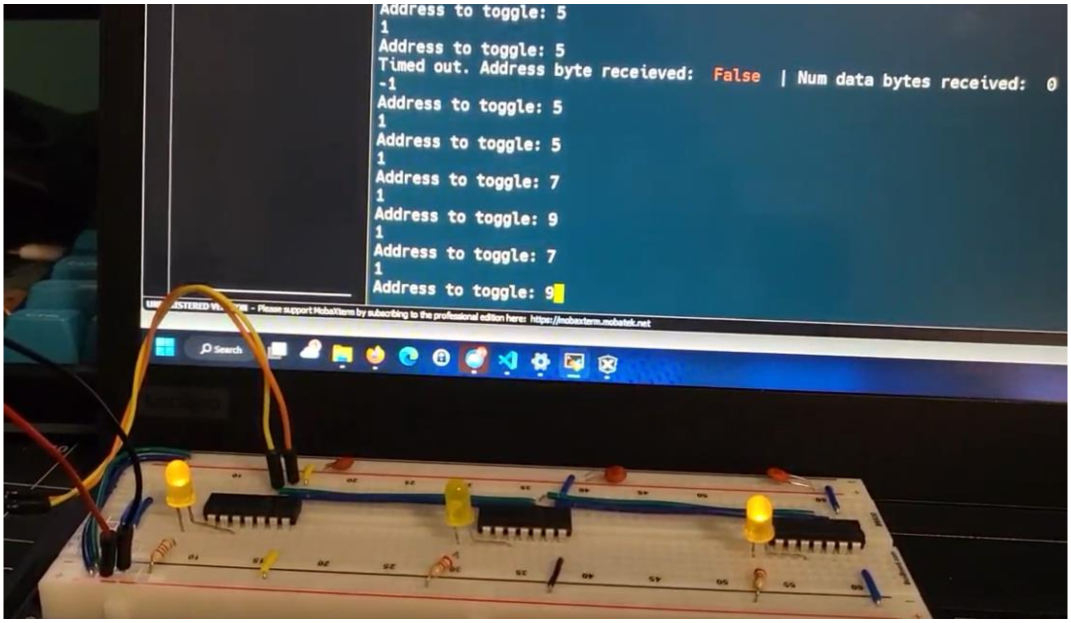

Figure 13 Microcontroller network test with LEDs

The only issues remaining with the UART communication is an occasional error. Sometimes bytes are lost or malformed, so error handling will need to be added in 404 to gracefully handle communication issues.

3.3.2. Sensor

The sensor firmware was validated via logic analyzer and visually with a python script to determine that the microcontroller was receiving correct data and that it matched up with real world measurements. The logic analyzer confirmed that a 10 μs pulse was being sent as required by the data sheet, and that realistic pulse was sent back. The sensor module prototype was then validated visually by placing an object a known distance from the sensor and running a python script on the Raspberry Pi to constantly poll the microcontroller to measure the distance.

The measurement was not extremely accurate, as the floating point operations take too much memory on the microcontroller, forcing the use of integer math. This was compounded by needing to use inches, as that was the only unit available for larger measurements. Measurements can be made more accurate by switching to centimeters or millimeters which will give more detail with integers.

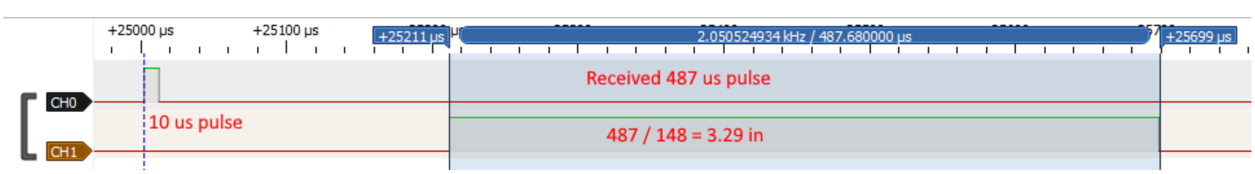

Figure 14 Analysis of ultrasonic sensor signal

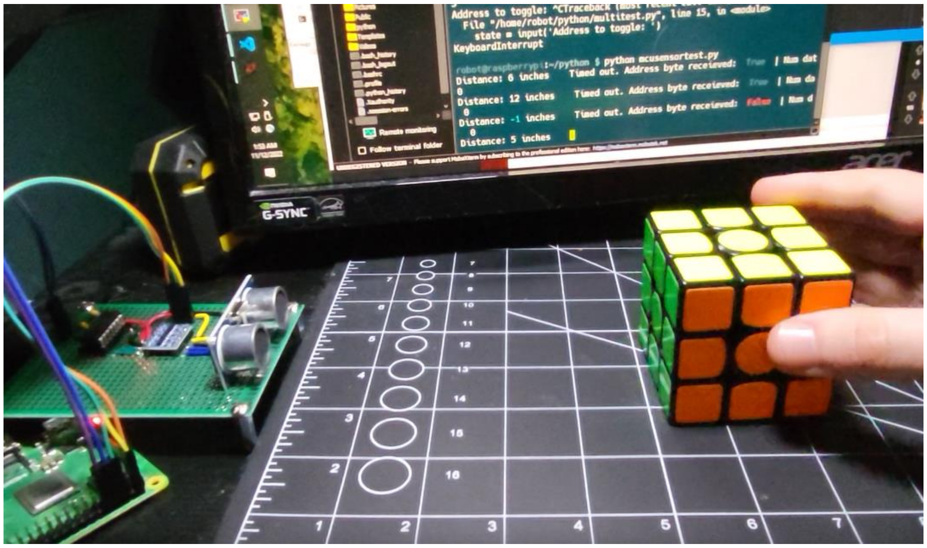

Figure 15 Visual confirmation of sensor module

3.3.3. Motor Driver

The motor driver was validated via a logic analyzer as well as visually with the motor driver PCB. It was not fully able to be tested, as there were some errors with the PCB that prevented full functionality. The PCB was able to be modified in a way to allow the microcontroller to turn on the motor and control the speed. The motor controller firmware was initially tested without the motor driver to ensure the outputs were correct, to prevent possible errors that would lead to a short. The figure below shows a change in direction where the PWM has a 45% duty cycle, the firmware turns off all outputs to allow any transistors to fully turn off, and then turns on the other side at 65% duty cycle.

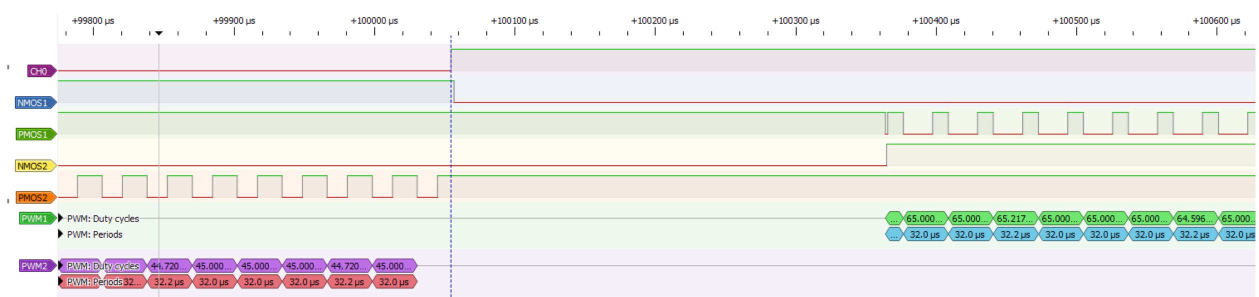

Figure 16 Motor driver initial tests

Due to the prototype motor driver PCB not functioning, not every aspect of the motor driver was able to be validated. It was possible to use a multimeter to validate that the correct outputs were being turned on for each direction in a similar way to the previous figure. The circuit was unfortunately not configured correctly to allow these control signals to do anything. After modifying the circuit to allow one direction, it was possible to validate the ability to turn the motor on and off, as well as modify the speed. Originally, the PWM signal was being sent at 31Khz, but this proved way too fast to allow the BJT transistor that enabled the PMOS side of the H-bridge to fully switch. The frequency was dropped to around 500hz which allowed for full speed control of the motor. Once the motor driver design errors are corrected, it will need to be retested with the firmware to confirm functionality.

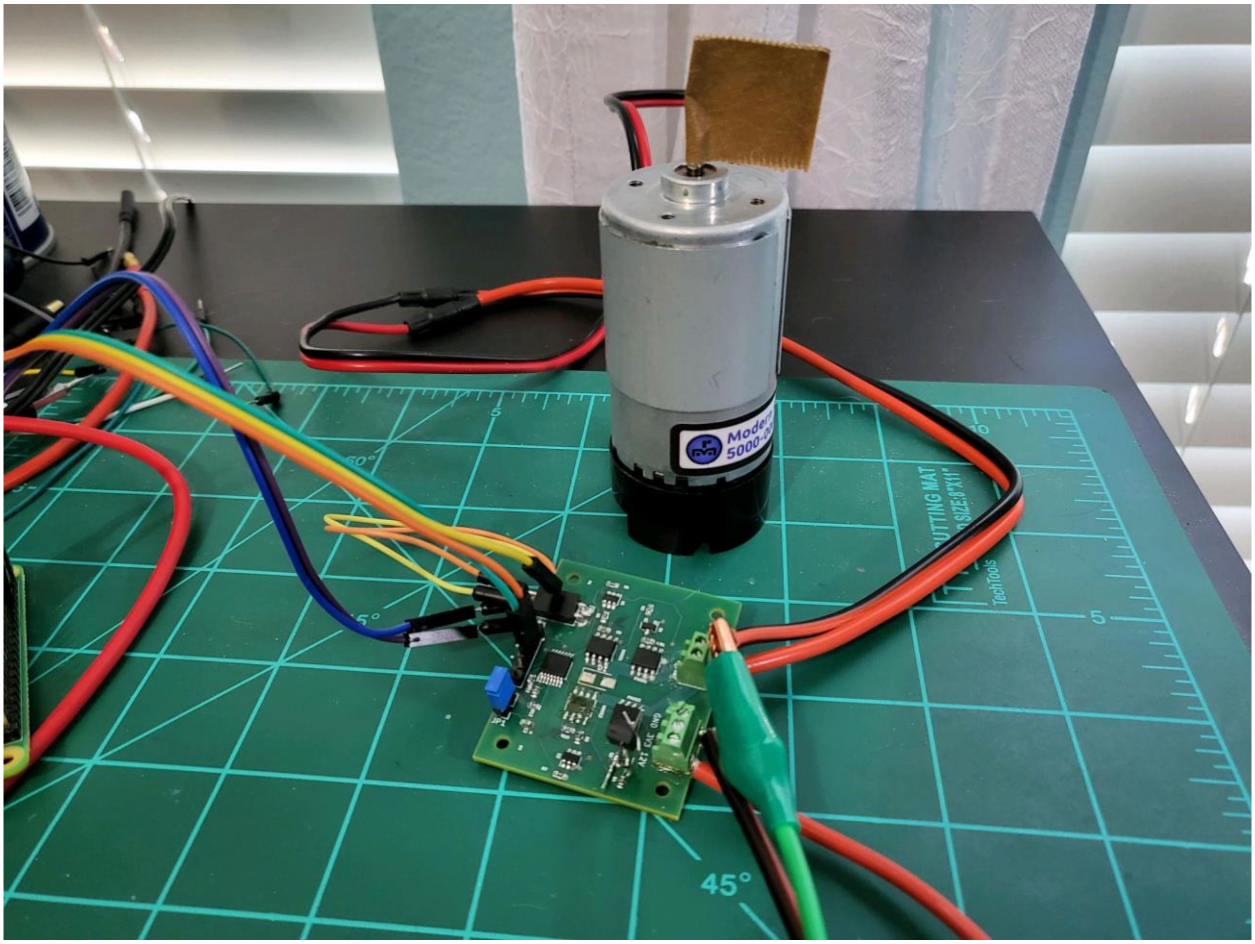

Figure 17 Motor driver test setup

3.4. Subsystem Conclusion

The Raspberry Pi and Microcontroller subsystem was able to successfully complete nearly all required tasks. UART communication between multiple microcontrollers at once was successfully tested, as well as the ability to send and receive commands and data, which will be vital for the final integration. The sensor module is able to successfully measure distance to an object and then send this data to the Raspberry Pi when requested. The motor driver firmware is able send the correct control signals to control the H-bridge of the motor driver, unfortunately, the motor driver design does not currently work, so full functionality can not be confirmed.

4. Housing and PCBs

4.1. Subsystem Introduction

4.1.1. Housing Subsystem

The purpose of the housing is to contain and provide mounting for all hardware of each module. It also provides a means of connecting modules together and preventing separation when in use. There is a housing design for each module.

4.1.2. Housing Subsystem

This subsystem includes PCBs for the sensor module, arm module, and connectors.

4.2. Subsystem Details

4.2.1. Housing Subsystem:

The basic design for all housing is a cube with an empty cube-shaped interior for storing and mounting any hardware. All housing designs are equipped with at least on face for connecting to another module. The connecting face is an octagonal pattern of small stubs protruding out and holes. When connecting two modules, these stubs slot into the holes of the opposing face to prevent any lateral movement. Magnets are used on the interior side of a connecting face to keep the modules together when in use. One side is screwed on separately to the housing, so that the interior is accessible.

Table 3 Housing General Dimensions

Exterior Side Length	Interior Side Length	Wall Thickness	Exterior Edge Fillet Radius
4.5 inches	4 inches	0.25 inches	0.25 inches

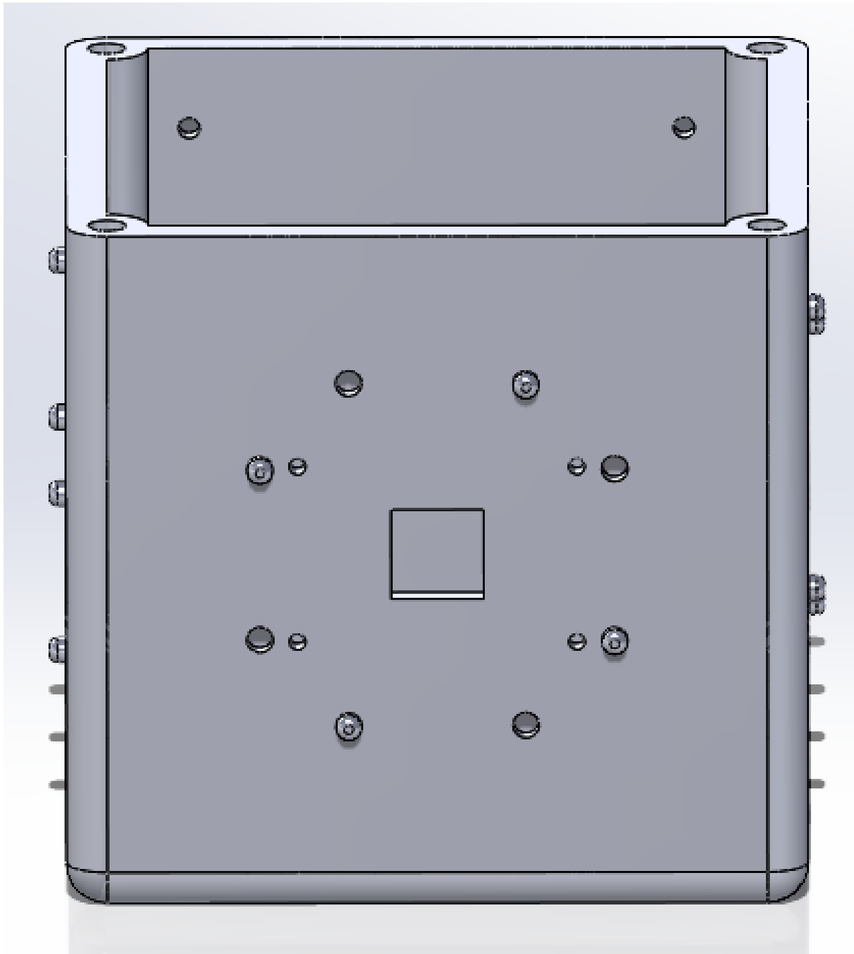

Figure 18 Control Module

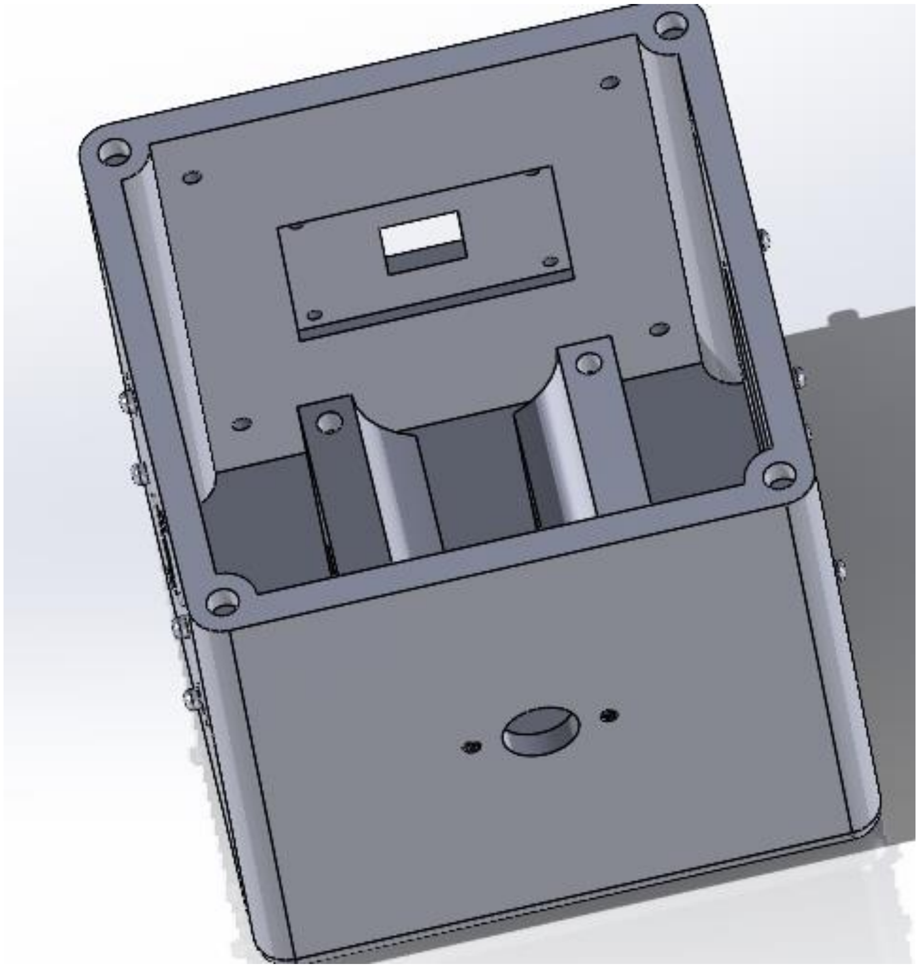

Figure 19 Motor Module

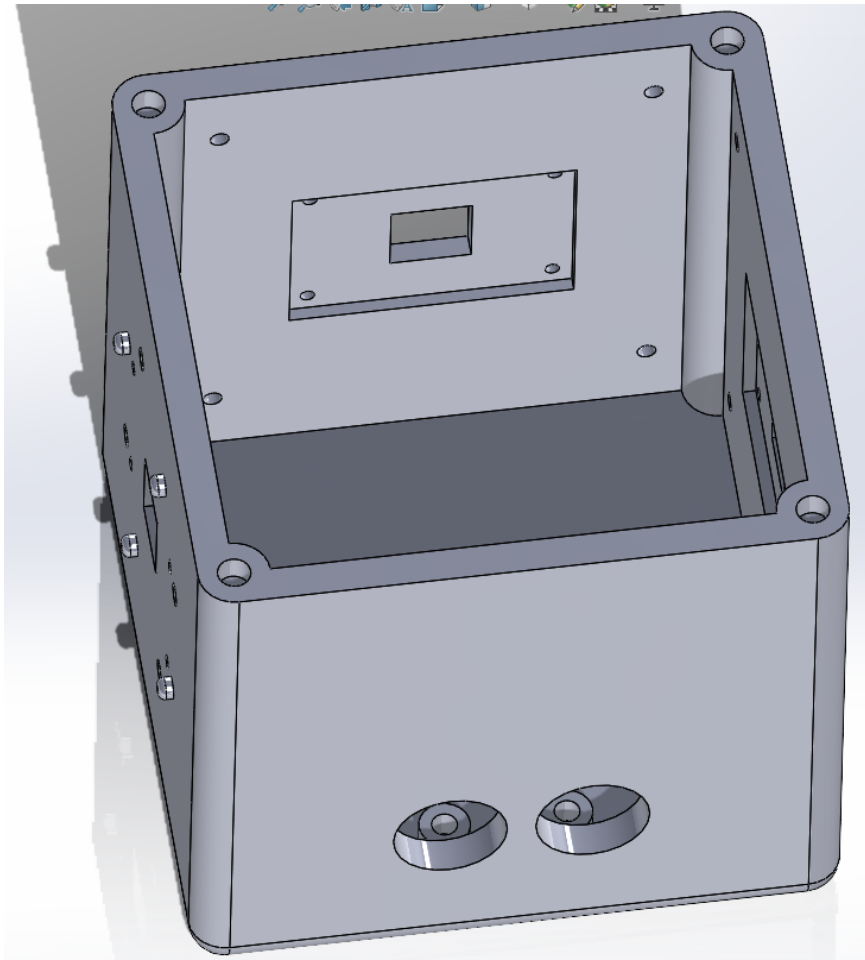

Figure 20 Sensor Module

Figure 21 Printed Housing 1

4.2.2. PCB Subsystem:

4.2.2.1. Sensor

The sensor PCB primarily consists of a microcontroller, 2 level shifters, and a 4-pin connector, J4, for mounting an ultrasonic sensor. The microcontroller, PIC16F15224-E/ST, requires 3.3V power, and the sensor, HC-SR04, requires 5V power. The level shifter is used for shifting logic levels from 3.3V to 5V for good communication between the microcontroller and the sensor.

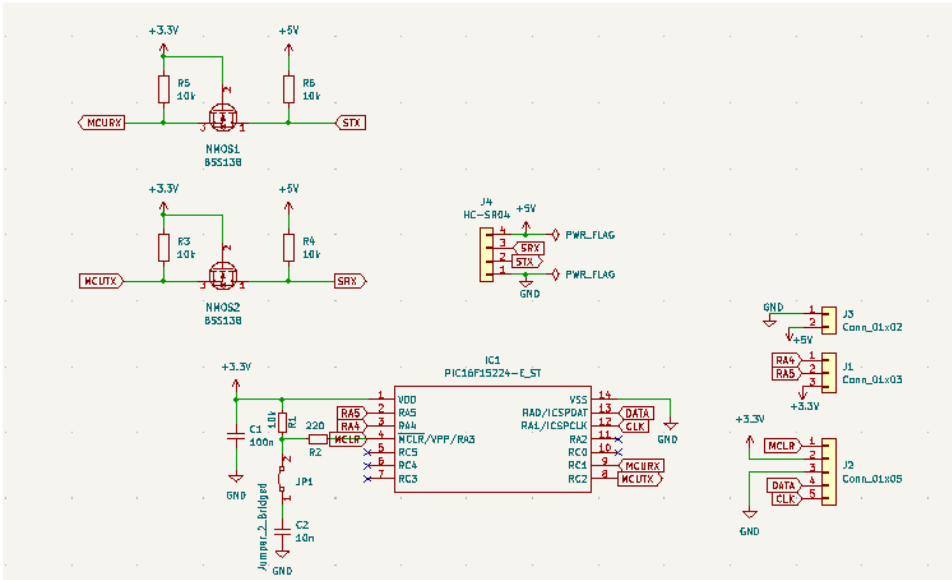

Figure 22 Sensor Schematic

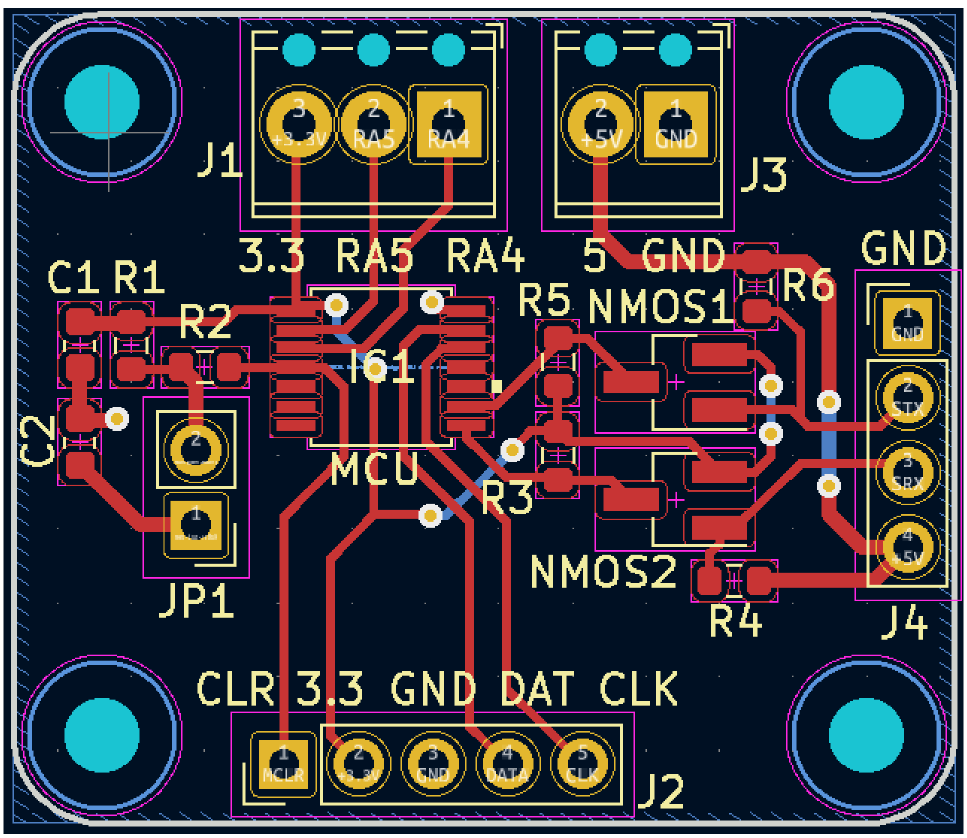

Figure 23 Sensor PCB Design

The 5-pin connector, J2, is used only for programming the microcontroller when not in use. When programming, jumper JP1 must first be disconnected. When in use, J1, J3, and JP1 will be connected.

Table 4 Servo Circuit Components

Component	Value
IC1	
NMOS1, NMOS2	
R1, R3, R4, R5, R6	10 kΩ
R2	220 Ω
C1	100 nF
C2	10 nF

Arm

After the demonstration, it was determined that a PCB design would be necessary for the arm subsystem. This design I very similar to that of the sensor and consists of a microcontroller, 3 level shifters, and 3 3-pin connectors, for connecting servomotors. The microcontroller, PIC16F15224-E/ST, requires 3.3V power, and the servos require 5V power. The level shifter is used for shifting logic levels from 3.3V to 5V for accurate control of the servos by the microcontroller.

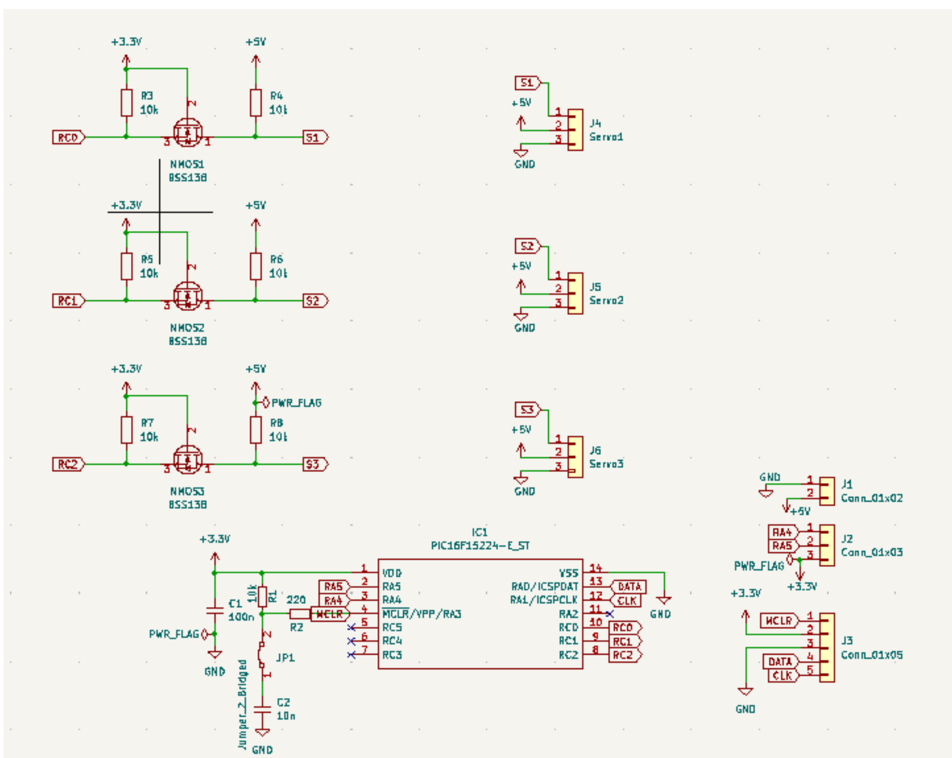

Figure 24 Arm Schematic

Table 5 Arm Circuit Components

Component	Value
IC1	
NMOS1, NMOS2	
R1, R3, R4, R5, R6, R7, R8	10 kΩ
R2	220 Ω
C1	100 nF
C2	10 nF

The 5-pin connector, J3, is used only for programming the microcontroller when not in use. When programming, jumper JP1 must first be disconnected. When in use, J1, J2, and JP1 will be connected

Connector

The connector is a very simple design with the sole purpose of providing electrical connections between modules, and for sending signals to the correct components. These signals include 3.3V, 5V, 12V, GND, UART RX & TX, and USB 2.0 D+ & D-.

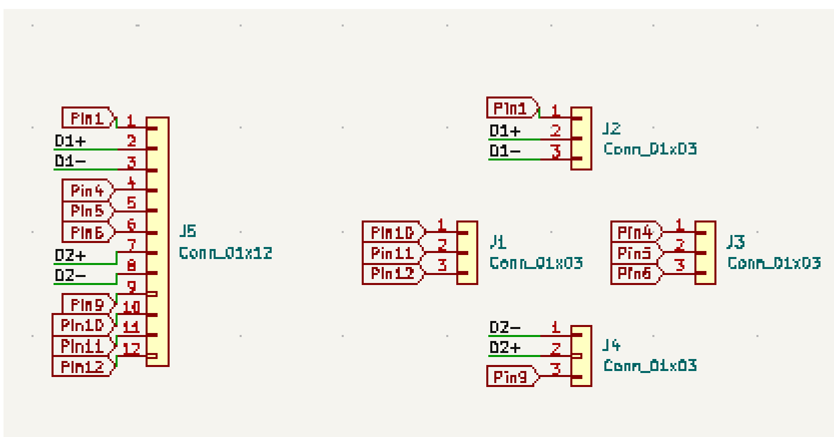

Figure 25 Connector Schematic

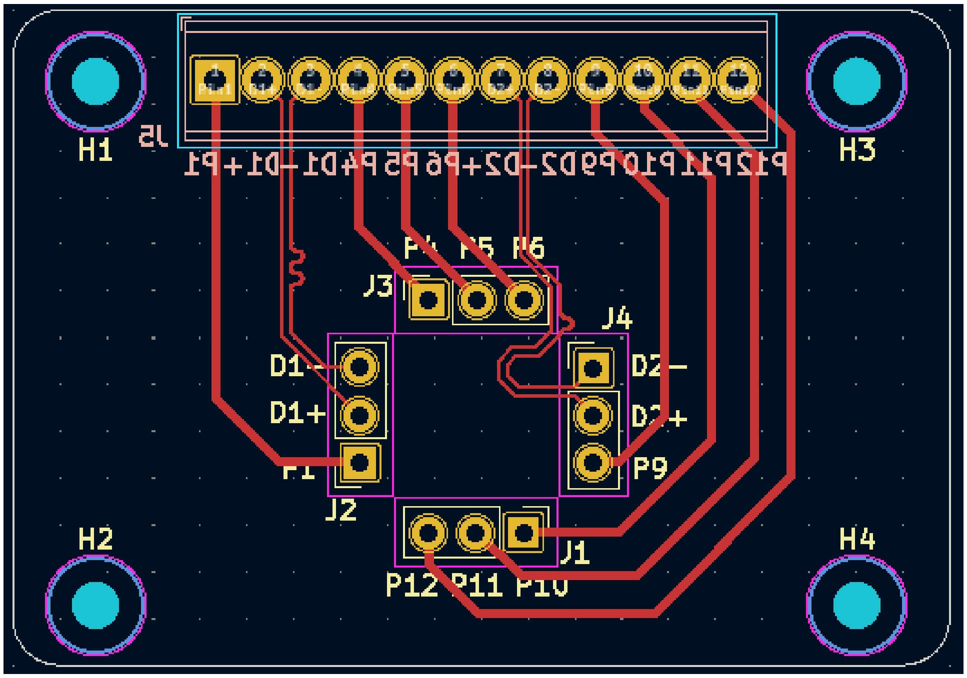

Figure 26 Connector PCB Design

The current layout of this design is 4 3-pin connectors arrayed in a square with alternating male and female pattern. The intent of this is to allow for modules to be connected in different orientations. These pins connect to a 12-pin connector on the opposite side of the PCB, to then be routed to parts inside a module.

4.3. Subsystem Validation

4.3.1. Housing/Connector

The housing subsystem could only be partially validated. The initial print did not turn out very well and there were several issues. Most holes were not large enough for screws or heat-set threaded inserts, and the stubs on the connecting faces were deformed.

I put 2 of the connector designs on perfboard and attached them to connecting faces on the housing. I was able to successfully connect them together in that way, however it was not flush due to the issue with the deformed stubs.

Figure 27 Housing/Connector 1

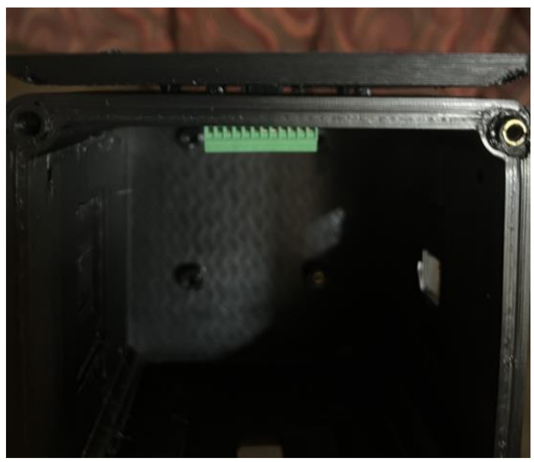

Figure 28 Housing/Connector 2

4.3.2. Sensor

The sensor PCB design was initially tested on perfboard by probing for continuity and checking correct voltage levels. Once the microcontroller was programmed by Andrew, and equipped with the HC-SR04 sensor, it was tested and shown to work correctly and accurately during the demonstration. This was done by placing your hand in front of the sensor next to a measuring tape and reading the distance readout on the computer. Unfortunately, I do not have an image from the demo, and Andrew has the sensor. One thing that Dr. Lusher pointed out was that I was not using a crystal oscillator in the design and was instead using the one built into the microcontroller. This could be a potential issue in the future and may be an improvement to the design.

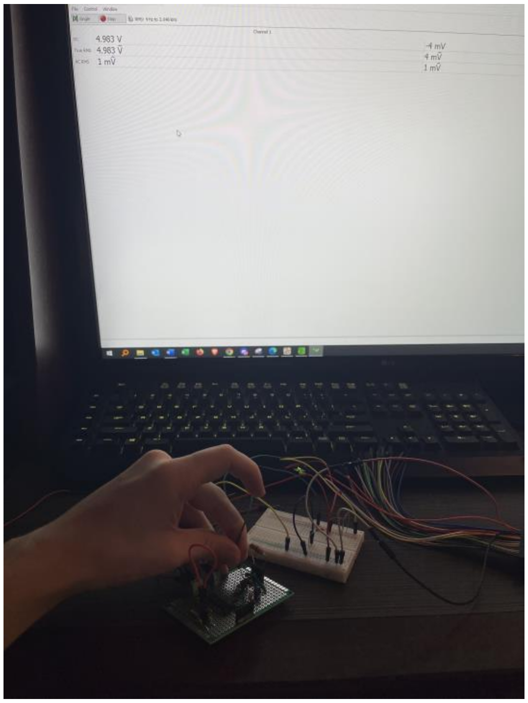

Figure 29 Sensor Design Probing

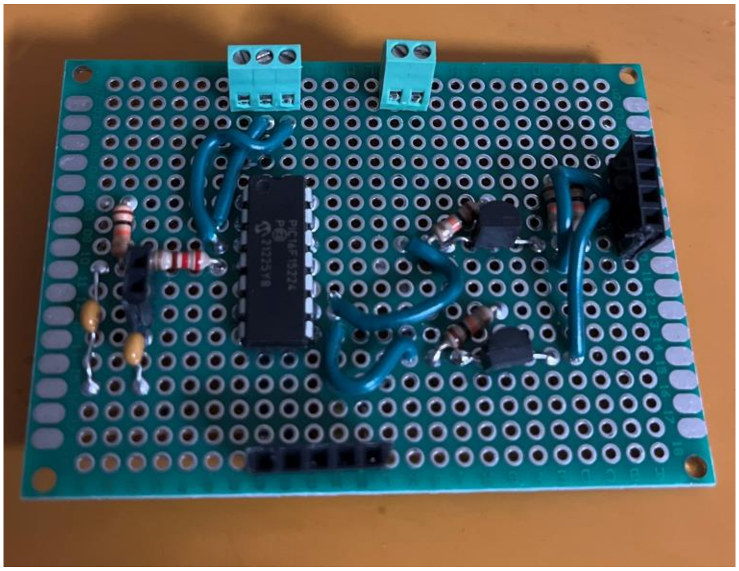

Figure 30 Sensor Design on Perfboard

4.4. Failure Analysis of Subsystem

4.4.1. Housing/Connector

As was stated in the validation, the print of the housing had some deformation issues preventing flush connections. Another part was printed, and it came out much more accurate, however it still had the issue with hole sizes being too small. The stubs held shape more consistently, however about 2 came out slightly warped. With this 2nd print the connecting faces were almost flush but not quite. This could be corrected by replacing the stubs with holes for metal rods to be inserted and heat set, however given that there was only slight deformation of the stubs on the second print it may be better to simply increase the hole size for the stubs altogether. The insert holes for the magnets were too small as well, so they were increased by heating the plastic. The magnets I had were quite small and ended up being too weak, so I will be using larger and possibly more magnets.

Figure 31 Printed Housing 2

5. Object Detection and Gripper Arm

5.1. Subsystem Introductions

Here lies a detailed report and analysis of the subsystems of which this section consists. Those subsystems are the Object Detection subsystem and Gripper Arm subsystem which are explained as follows and all files relating to the following subsystems are included in the Github repository inside their respective folders.

5.1.1. Object Detection Subsystem Introduction

The purpose of this subsystem is to utilize the webcam shown in the figure below to input video data of what is essentially in front of the robot and process it. The video is going to feed the raspberry pi through a USB connection. The object detection subsystem then identifies all objects inside the frame of the webcam's line of sight and sends those data to determine the next action.

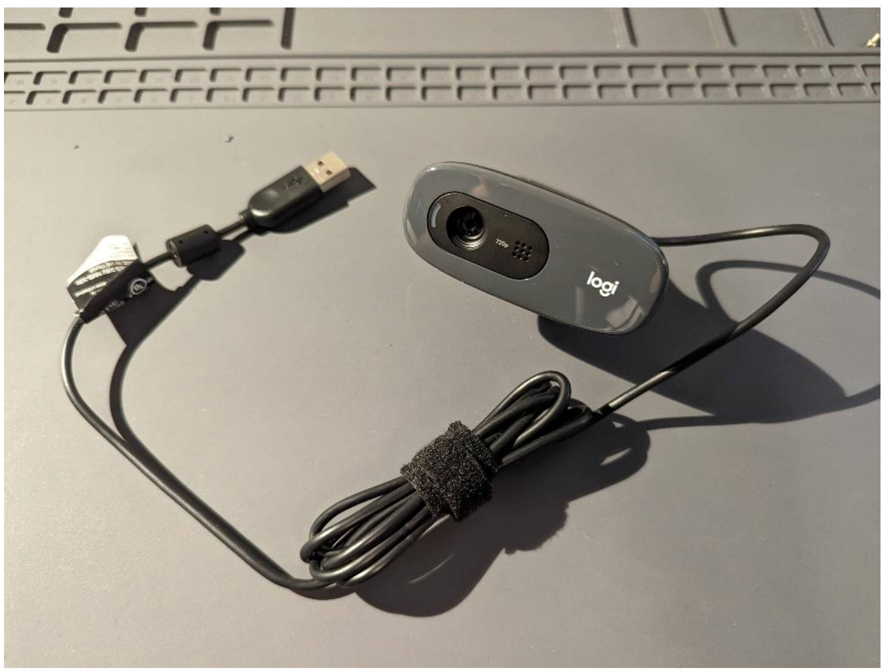

Figure 32: Webcam Interface Used.

5.1.2. Gripper Arm Subsystem Introduction

The purpose of this subsystem is to reach over the robot and pick up the desired object once identified by the object detection subsystem. The arm contains three main servo motors that control its movements. Two servo motors at the base of the arm for multiple reasons are discussed in the later sections, and one servo at the gripper head.

5.2. Subsystem Details

5.2.1. Object Detection Subsystem Details

The program which processes the live feed from the webcam and identifies objects in frame real-time is written using python and utilizes the libraries of OpenCV and YOLO (You Only Look Once). Initially, the program at early stages used to only able to process saved videos and output the results in another video with overlays. Then through multiple updates and modifications now the program is capable of processing while the video data is being transmitted.

The program's object detection accuracy is tied to its processing speed, and the ratio is inversely proportional meaning that if the accuracy is increased the program ends up being slower and vice versa. Thus, two versions of this subsystem were created each specialized in one aspect, one had superior accuracy, and the other outputted result as fast as the webcam could capture and stream.

During presentations and demonstrations, the program was made to present its results and show what it can identify for visual purposes, as shown in the figure below, which also proved detrimental to its processing speed. Since the robot should be able to coordinate and move accordingly on its own, the robot does not need a video stream of what it sees. Rather the robot requires only information and said coordinates to act on. Having said that, the script was modified once again at this point to be able to process everything without needing to show, and the script was tested heedlessly. The program also is capable of outputting its result in a fully customizable manner that suits the main controller of the robot and determining which form is a task for 404. As of now, the program can output to the console all the data it can extract including but not limited to the name of the object, the center coordinate of the object, and the accuracy of the object's identification.

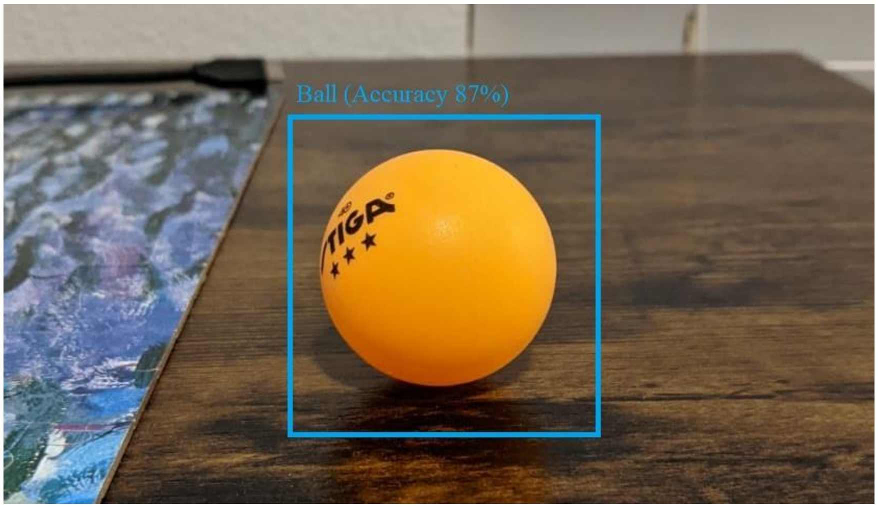

Figure 33: Representation of How Object Detection Looks Like.

5.2.2. Gripper Arm Subsystem Details

The arm is made to be lightweight to minimize the tension on the base the arm extends over the robot while having a sturdy base to support such extension. The first reason for having two heavy servos at the base is to add weight and support the base so it can withstand the full extension of the arm during task completion. The second reason for having two servos is to be able to fully control the arm's movement and customize where exactly should the arm go. One servo controls the vertical movements and points the gripper toward the desired altitude, and the second servo pushes the gripper forward and backward controlling the horizontal movements. On the other hand, the gripper head's servo is specifically chosen to be a small light servo to reduce the tension and the weight on the arm. Since no heavy tasks like lifting the entire arm are required from this servo, choosing a smaller one works perfectly.

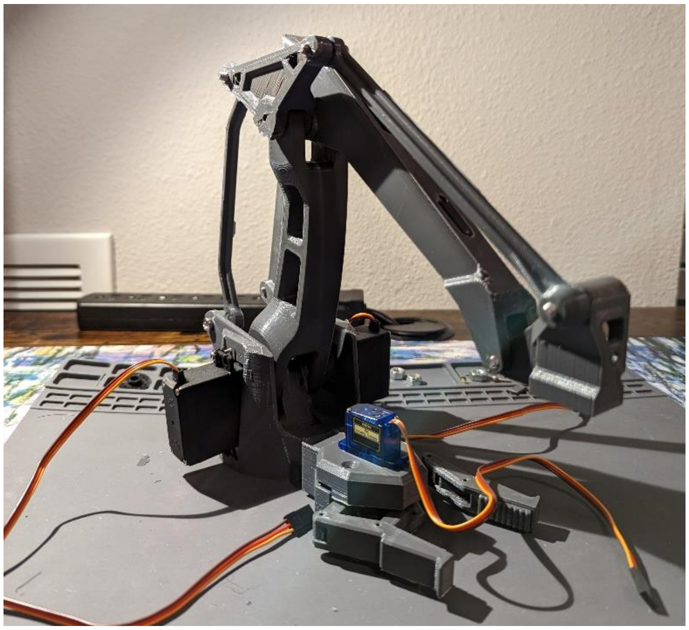

Figure 34: Gripper Arm Demonstration.

As shown in the figure above there are two supporting links in the back of the main arm and each has a purpose. The straight backlink is used to control the vertical position of the second arm, in addition, to support. The curved backlink however is, in contrast to the first backlink, connected to the base itself instead of the moving parts of the arm. This link's purpose is to limit and control the orientation of the triangular joint which is responsible for stabilizing the gripper head and ensuring that it always points to a certain direction regardless of the arm's positioning. The gripper head is connected to the main arm to control its position and is connected to the triangular joint through a straight link to determine where it points. Technically there are two parallel arms, however only one carries the weight and the other only consists of thin links whose sole purpose is to stabilize and support the main arm.

5.3. Subsystem Validation

5.3.1. Object Detection Subsystem Validation

The goal of this subsystem is to be able to provide the main controller with sufficient information to determine where the robot should go and what objects it should pursue, and for that, we do not need the object detection program to show anything. Once the program was not needed to present and eliminated the show feature, its processing speed was upped by almost double, and it did not use as much memory which is to be expected. In addition, the program proved to be capable of outputting all sorts of data in various forms, some of which are visual and console print as shown in the figure below. The figure also demonstrates the capabilities of the program where it can focus on a few objects and ignore all others, which is a feature that will be used in 404 as well.

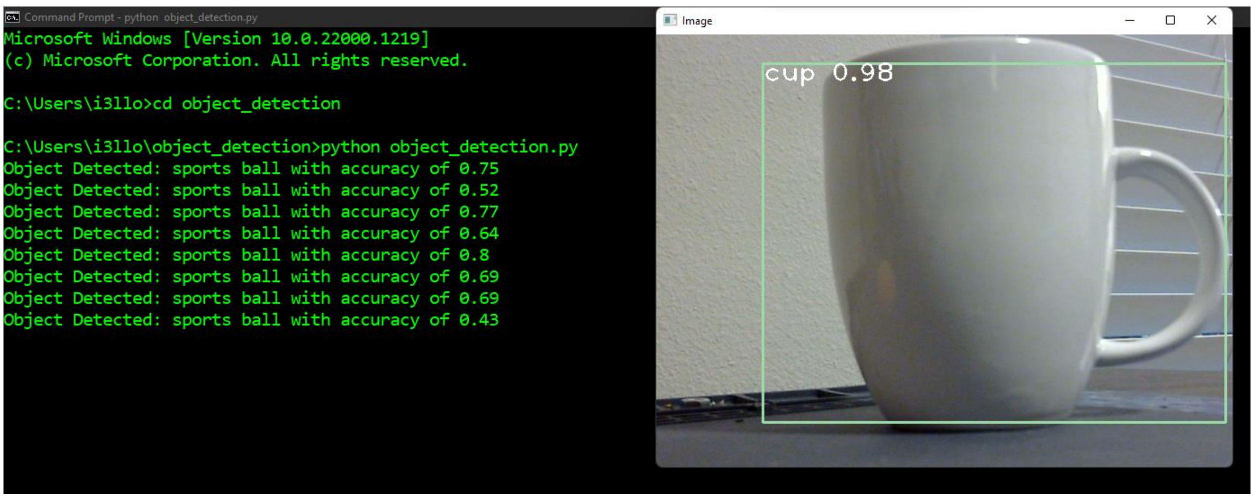

Figure 35: Both Forms of Results Output.

5.3.2. Gripper Arm Subsystem Validation

For this subsystem to be able to move as intended, it needs to go to 404 and be integrated with the other subsystems for power, control signals, and housing. However, as of now the gripper arm, fortunately, has a PCB tailored for its specific usages made by the PCBs subsystem. The arm's PCB will allow for the power the arm needs to flow and provide the connection so that the main controller can send signals and control its movements. Nonetheless, for validation purposes, the arm proved that it could move through its full range of motion, in addition, simple movements controlled by an Arduino were shown to the professor during demonstrations. Two example positions of the arm were captured and presented in the figures below for clarification. One note regarding the base servo motors is that one of them is 1-2mm too far out. The servo still connects to the gear and moves properly, however, just to be safe and more secure the servo should go deeper and fully locks in place.

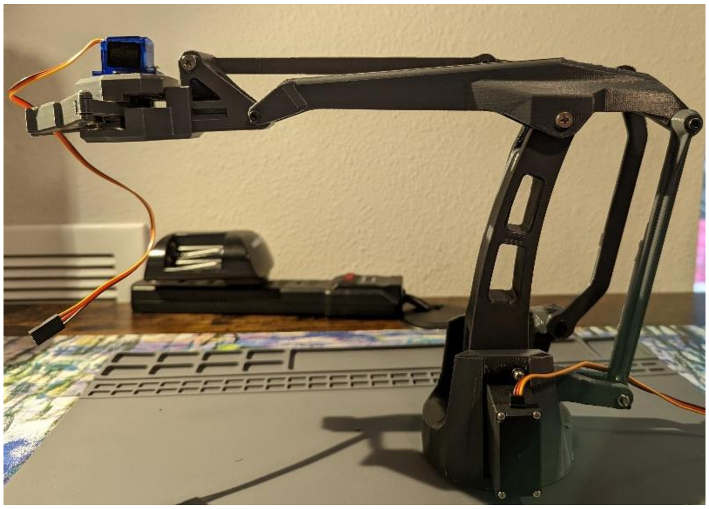

Figure 36: Gripper Arm Example Position 1.

Figure 37: Gripper Arm Example Position 2.

5.4. Subsystem Conclusions

Both parts of this subsystems have been completed and validated accordingly, both subsystems produced appropriate results. As of this stage and the state of both subsystems, both of them proved readiness to go into 404 with minor expected modifications. Modifications such as for the object detection subsystem would be tailoring the output results to however fits the main controller. Additionally, the base of the gripper arm subsystem will be integrated with the housing subsystem and is expected to be adjusted to fit with the module, on top of the servo position adjustments.