EE175AB Final Report

Kitchen Helper

EE175AB Final Report
Department of Electrical Engineering, UC Riverside

Project Team Member(s)	Sunil Alexander, Richard "Dylan" McGee, Grant Beatty
Date Submitted	March 14, 2022
Section Professor	Dr. Roman Chomko
Revision	Revision 1.1
URL of Project YouTube Videos	https://www.youtube.com/watch?v=ewehUymT7uM&ab_channel=GrantBeatty https://www.youtube.com/watch?v=yvtiDKHStDQ&t=12s https://www.youtube.com/watch?v=M6Imj_xpfE4 https://www.youtube.com/watch?v=NrHi9aHbBfA&t=10s
Permanent Emails of all team members	gbeat002@ucr.edu rmcge002@ucr.edu salex009@ucr.edu

Summary

The contents of this report outline the process of design and development for a simple kitchen helper robot. This document contains details regarding the product's subsystems, their construction, and their integration.

Revisions

Version	Description of Version	Author(s)	Date Completed	Approval
0.1	First Draft — Many Errors	Sunil Alexander, Richard "Dylan" McGee, Grant Beatty	3/9/2022	
0.2	Final Touches and Format Changes	Sunil Alexander, Richard "Dylan" McGee, Grant Beatty	3/12/2022	
1.0	Final Report	Sunil Alexander, Richard "Dylan" McGee, Grant Beatty	3/13/2022	
1.1	181W Revised Report	Sunil Alexander, Grant Beatty	6/9/2022	

Table of Contents

1. Executive Summary

This project is called the Kitchen Helper Robot whose purpose is to set and clear a table of dishes and utensils. For a lot of homeowners, the function of setting up and taking down kitchen utensils and dishes could sometimes be a hassle and time consuming. This robot can find and pick up specific objects on a table and put them in designated positions, this can be used to distribute dishes and silverware to each person on a table or to gather them to a specified area to be put away.

The robot is designed with ultrasonic sensors to gather information about its surrounding area. The microprocessors take the information and use an algorithm to navigate a table. The camera allows the robot to decipher what objects to pick up and react accordingly. The computer vision algorithm in the current design is able to identify specific colored cups. The computer vision system is accurate enough to pick up a 1.5 ounce cup using color and shape recognition. Upon acquiring a cup with the robot arm, the Kitchen Helper moves it to a designated location.

2.*Introduction

The impetus for our robot design was inspired by robots that can move autonomously that are used in industries today such as the iRobot Roomba autonomous vacuum and the Kiva sorting robot. In most modern kitchens there are little to no tasks that are autonomous. Every task requires humans to do menial work that could be seen as a waste of time. The robot we created addresses the need for automation in our modern kitchens. Our robot incorporates several aspects of engineering such as robotics, power systems, and embedded systems.

2.1 * Design Objectives and System Overview

The objective of the project was to design a robot capable of helping with tasks on the kitchen table. Realistically, the robotic system would function as an assistant capable of identifying kitchenware and moving around to set or clear the kitchen table. The robot was designed to move between square cells in the directions up, down, left, or right. The sonic sensors on the system are intended to detect obstacles in the cells beside or in front of the system. The base of the robot needed to be small enough to traverse a table which is why we decided to have the robot fit inside a 1 foot by 1 foot area as our size constraint. One of the critical objectives for the robot was keeping it on the table without falling, therefore the accuracy specifications were very important. The robot turned with an accuracy of 2.3 degrees and could move straight. The robot arm needed to pick up the cup accurately and therefore could only be off by 3cm. The object detection done by the computer vision system had to be as fast as 10 frames per second and be able to send the accurate centroid data to the main system.

High Level Design

The sensors of the robot function as the primary inputs. The sonic sensors detect obstacles and sense proximity to the target object. The IMU module detects the turning direction of the system. The encoder readings show the movement of the wheels. The camera senses the target object. All of these components allow the kitchen helper robot to accurately traverse through various obstacles in its surroundings and acquire vision of a target object.

The motors are the primary outputs of the system. DC motors propel the wheels of the system and they are connected directly to the custom printed circuit board. The robot arm used to grab objects is propelled by servo motors which are connected to an elevated perfboard. These motors allow the robot to move around and manipulate its environment.

The hardware of the system allows the robot to use the data from the sensors to perform its functions in an organized manner. The microcontroller units that are used are two Teensy 4.0 chips and a Jetson Nano. The finite state machine, the controller of the system is contained in the Teensy which is soldered to a custom printed circuit board. The other Teensy controls the robot arm and is soldered to a perfboard. The Jetson Nano is connected to a Raspberry Pi camera module and is used to perform image processing on camera data which allows the robot to identify objects.

The software of the system was C++ for the Teensys and Python for the Jetson Nano. One Teensy primarily performs all the main functions. It communicates with the sensors and other microcontroller

units to make decisions about how to move around. The industry standard communication protocols used are serial communication through USB and I2C.

2.2 * Backgrounds and Prior Art

One of the robots that inspired our design was the Kiva, an Amazon warehouse robot capable of moving 450 kg shelves and moving at a pace of 5km/h while navigating through a dynamic environment. The robot is only 2.5 by 2 feet and 1 foot high. The other robot that influenced our design was the Butter Robot, created through the collaboration of Adult Swim and Digital Dream Labs, and based on a fictional robot in the cartoon television series *Rick and Morty*. This robot was designed to retrieve a stick of butter to the person who requested it.

Figure 2.1: Kiva Robot — a warehouse shelf mover

Figure 2.2: Butter Robot

2.3 * Development Environment and Tools

Hardware Development

For the PCB design we made multiple PCB libraries since we couldn't find parts for them on Eagle so we made multiple custom ones such as for the motor encoder connections, the voltage regulator, and the motor brackets. The PCB has a top and bottom layer to have a common 5v and GND layer. We used Eagle to make our finalized PCB.
To supply power to the system we used an INIU power bank which supplies 5V and has 10,000mAh. It powers the Jetson Nano through a USB cable and connects to the main Teensy through another USB cable which supplies power and serial communication. A 7.4V LiPo battery, downregulated to 5V, supplies power to the robot arm and the Teensy that controls it.

Software Development

The robot arm Teensy 4.0 and the main chassis Teensy 4.0 are programmed with the Arduino IDE using an add-on called Teensyduino. The language the teensy was coded in was similar to C++ the arduino IDE uses. The CV Python scripts were first developed and implemented in an IDE called Google Colaboratory on pictures taken from a smartphone camera. The program was then moved onto the Jetson Nano which uses the Linux4Tegra operating system. The script compiles using Python 3.4 in the command terminal.

2.4 * Related Documents and Supporting Materials

USB specifications can be found on this website:

> "Document Library | USB-IF". Oct-2021. [Online]. Available: https://www.usb.org/documents. [Accessed: March-2022].

I2C specifications can be found on this website:

> "I2C-bus specification and User Manual - NXP". Oct-2021. [Online]. Available: https://nxp.com/docs/en/user-guide/UM10204.pdf. [Accessed: March-2022].

2.5 Definitions and Acronyms

Arduino — open source electronics platform
CPU — Central Processing Unit
GPU — Graphics Processing Unit
I2C — Inter-Integrated Circuit
IEE — Institute of Electrical and Electronics Engineers
IDE — Integrated Development Environment

IMU — Inertial Measurement Unit Module
Jetson Nano — mini-computer with GPU and ARM CPU
MCU — Microcontroller Unit
OpenCV — Open Computer Vision Library
PCB — Printed Circuit Board
Pololu — electronics and robotics store
PWM — Pulse Wave Modulation
SSH — Secure Shell
SUDO — SuperUser DO
Teensy 4.0 — ARM Cortex-M7 microcontroller
Teensyduino — add-on for the Arduino IDE that allows for the programming of the Teensy
UCR — University of California, Riverside
USB — Universal Serial Bus

3. Design Considerations

3.1 Realistic Constraints

The main constraints that had the most significant impact on this system were weight, size, spatial, time, and budget constraints.

Weight and Size Constraints

The Robot chassis needs to be small enough to fit within a 1 foot by 1 foot cell and the weight can not be so much that the DC motors cannot move the load.

Spatial Constraints

The Robot must be able to move straight and turn with an accuracy of 2.3 degrees. It should be maintained within the 1 foot by 1 foot cells at all times. The robot arm must be able to grab the demo cup with an accuracy of 3 cm.

Project Time and Budget Constraints

The project had a budget of 600$ and a time constraint of 10 weeks.

3.2 System Environments and External Interfaces

The two main components that were utilized in this project were the Teensys and the Jetson Nano.
The Teensys use the Arduino IDE with the Teensyduino add-on and additional libraries. The Jetson runs with the Linux4Tegra operating system and runs Python 3.4 scripts with additional libraries.

3.3 Industry Standards

This project uses I2C and USB industry standards
I2C allows the main MCU to communicate with the IMU module to acquire directional data.
Serial communication via USB cable enables the Jetson to communicate object centroid data when pinged by the main MCU.

3.4 Knowledge and Skills

The designing and construction of the robot required several branches of engineering such as robotics, control systems, embedded systems, computer vision, electronic circuits, and power management.

The following skills are required: Eagle for PCB design, soldering, wire management, multimeter use for wire testing, programming skills in Arduino, C/C++, Python, and Linux.

Courses and Experience of Group Members:

Grant Beatty

Prior Knowledge:
Engineering Circuit Analysis (EE 1A/B), Intro to Computer Science for Engineering Majors (CS 13), Logic Design and Intro to Embedded Systems (EE 120A/B), Sensing and Actuation for Embedded Systems (EE 128), Computer Vision (EE 146), Arduino Programing.

New Knowledge:
Python on Jetson Nano, OpenCV on Jetson Nano, Serial Communication, Additional Linux Commands.

Sunil Alexander

Prior Knowledge:
Engineering Circuit Analysis (EE 1A/B), Intro to Computer Science for Engineering Majors (CS 13), Logic Design and Intro to Embedded Systems (EE 120A/B), Automatic Control (EE 132), Project Management, Soldering, Oral Presentation.

New Knowledge:
C++, PCB Design, Project Management, Subsystem Integration and Additional Linux Command.

Richard Dylan McGee

Engineering Circuit Analysis (EE 1A/B), Logic Design and Intro to Embedded Systems (EE 120A/B), Introduction to Robotics (EE 144), Computer Vision (EE 146), Arduino, FSM coding, Eagle PCB design.

New Knowledge:
Serial Communication and Additional Linux Commands.

3.5 Budget and Cost Analysis

Overall the Cost of the robot was slightly above 500 dollars. Various mistakes were made on the initial PCB boards that were preventable, therefore increasing the cost of the project. The Jetson Nano was an unnecessary purchase considering that the Raspberry Pi could perform all functions of the final design at only a fraction of the cost. Initially, the Jetson Nano was purchased for its ability to process convolutional neural networks, which ultimately was not used in the project.

Parts	Quantity	Cost
Jetson Nano	1	$106
sn754410 motor driver	1	$2.95
sonar sensor(HCsr04)	4	$9
prototype breadboard	3	$34.56
wires package	3	$7.98
LSM6DS33 imu	1	$15.95
l7805cv voltage regulator	1	$0.63
servos to make robot arm	1	$27
32 gb Micro SD card	1	$16
INIU power bank	1	$20
Raspberry Pi Camera	1	$25
motor Brackets Pair for 99:1 25D dc pololu motors	1	$7.95
aluminum mounting hub pair	1	$6.95
Wheels Pair	2	$7.95
DIY Cardboard Robot Arm	1	$49.00
KeyeStudio 4 dof arm	1	$56
pololu arm kit	1	79.95
bigger dc motors 25D 99:1	2	34.95
TOTAL COST:		$663

Figure 3.1: Budget Table

3.6 Safety

Basic personal protection equipment was used when handling potentially hazardous tools. Masks were worn when using the soldering iron to reduce the risk of inhaling the soldering vapor and gloves were used when drilling the motor brackets into the PCB. To prevent motors from running prematurely and driving the system off of the edge of a table or counter top, a sonar sensor check was made to stop any robot movement until it was ready for use.

3.7 Documentation

During the construction, coding, and testing the robot after any major construction, or coding test we recorded a video of whatever was just finished. Then, we shared the video in a Google drive. Also for any substantial milestone in the code or PCB design we put it into github in order to revert back to previous commits if need be.

3.8 Risks and Volatile Areas

The main volatile area of the robot was the power supply. Through trial and error, it was discovered that if the polarity of the battery for the PCB was reversed (as in power to GND and vice versa) the PCB would short circuit and become useless. This risk was mitigated with the addition of a JST battery connector which was soldered to ensure that the battery could only be plugged in the correct way. The pathfinding had two possible risks: obstacle reading and direction tracking. If the sonar sensor did not read an obstacle in a cell properly then it could potentially path incorrectly or run into the obstacle. A similar error would occur if the direction tracking was faulty. To mitigate the occurrence of the obstacle detection errors, the sonar sensor was set to sample 10 readings when looking for an obstacle. The direction tracking was ensured through various tests that confirmed the orientation of the robot as well as the positions of the neighboring cells.

4. Experiment Design and Feasibility Study

There were over 10 experiments and feasibility tests that were paramount in the success of this project. Every subsystem of the kitchen helper robot needed to be tested in order to function as an integrated system. These subsystems include the robot arm, motors, and computer vision. In this section, specific experiments of each subsystem are elucidated.

4.1 Experiment Design

4.1.1 Sonar Sensor

Dylan McGee designed and performed this experiment.

Objective: To ensure sonar sensors can give consistent readings when detecting objects within a radius of 18”.

Setup: The get_sonar_dist function was isolated and outputs were printed to the Serial Monitor.

Procedure: Have the handmade block obstacles moved in increments of 4 inches and look at the Serial monitor to see if distance readings changed accordingly.

Expected Results: The displayed distance values should be proportional to the actual distance between the sonar sensor and the obstacle.

4.1.2 Basic Motor Control

Dylan McGee designed and performed this experiment.

Objective: To verify the dc motor pair can perform the basic traversal functions such as forward movement, halting, and turning.

Setup: Each function was isolated and run consecutively with a stop gap between each of them.

Procedure: Have the commands running and observe that the robot does the expected motor functions at the correct order as in the code.

Expected Results: The robot should turn left and right then move forward and stop.

4.1.3 IMU PID Controller

Dylan McGee designed and performed this experiment.

Objective: To determine if the IMU based PID controller could allow the robot to move straight within 2.3 degrees of accuracy.

Setup: Set up the PID controller class to use the imu_angle as the current value and assign it relevant setpoints,kp,kd etc.

Procedure: Make the robot perform the PID forward function and then see if it goes straight.

Expected Results: The robot should be accurate enough to drive forward without deviation to the left or the right.

4.1.4 Differential Velocity PID Controller

Dylan McGee designed and performed this experiment.

Objective: To determine if the differential velocity based PID controller could allow the robot to move straight within 2.3 degrees of accuracy.

Setup: Set up the PID controller class to use the difference in velocities between the motors as the current value and assign it relevant setpoints,kp,kd etc.

Procedure: Make the robot perform the PID forward function and then see if it goes straight.

Expected Results: The robot should be accurate enough to drive forward without deviation to the left or the right.

4.1.5 Concurrent Separate Velocity PID Controller

Dylan McGee designed and performed this experiment.

Objective: To determine if two velocity based PID controllers could allow the robot to move straight within 2.3 degrees of accuracy.

Setup: Set up multiple PID controller classes to use the velocity for the respective motor(left or right) as the current value and assign it relevant setpoints,kp,kd etc. (as per the biases of the specific motor), Then add the error to a global left_pwm and right_pwm values to adjust the motor speeds

Procedure: Make the robot perform the PID forward function and then see if it goes straight.

Expected Results: The robot should be accurate enough to drive forward without deviation to the left or the right.

4.1.6 Pathfinding Simulation

Dylan McGee and Grant Beatty designed and performed this experiment.

Objective: To make sure the pathfinding algorithm to be used (Treamux) would solve the 5x5 maze with any given obstacle placement.

Setup: Have the Treamux Pathfinding algorithm print these values onto the computer screen: the maze, the current states of the cells in the maze, and where the robot currently is.

Procedure: Input different obstacle configurations and make sure the Tremaux algorithm outputs a path from the starting cell to the maze and also make sure the algorithm correctly changes the visited number for each coord (the amount of times the coord has been visited) whenever it occupies that coordinate.

Expected Results: To have the Tremaux algorithm simulation working for any viable obstacle configuration.

4.1.7 Computer Vision Methods

Grant Beatty designed and performed this experiment.

Objective: To construct a proper object detection algorithm for the Pi camera and Jetson Nano, several methods of object isolation with OpenCV were tested.

Setup: Use Google Colaboratory to run various built in CV algorithms: SIFT, SURF, ORB, Contour Matching, and Color Filtering.

Procedure: Take several picture samples and test to see if they are able to utilize the algorithms. Display matched keypoints and/or isolated pixels to determine if the algorithm is working properly.

Expected Results: The CV algorithms should be able to isolate the target object and calculate its centroid data.

4.1.8 Camera Test

Grant Beatty designed and performed this experiment.

Objective: The Jetson Nano requires a compatible camera and so a suitable one must have confirmed functionality.

Setup: Gather the Jetson Nano and the USB and Pi cameras for testing.

Procedure: Test the cameras to verify connection and functionionality by using the built-in streaming app then run using a python script.

Expected Results: The camera should be able to stream and the frame data must be accessible through a python script.

4.1.9 Subsystem Communication Teensy to Jetson Nano

Dylan McGee and Grant Beatty designed and performed this experiment.

Objective: Make sure the serial communication works properly and that the Jeston Nano can send a char array of 8 chars starting with ‘a’ and ending with ‘f’ (to denote the start and end of a transmission) and that the teensy can convert the middle 6 chars to two integers.

Setup: First attach the USB serial cable to the teensy and Jetson Nano.
Have the Jetson Nano continuously send the char array of ‘a’,’1’,’0’,’0’,’3’,’3’,’1’,’f’ and then set up the code on the teensy so that if the received integers are 100 and 331 the robot will move forward.

Procedure: Setup the connection and run the python script of the Jetson Nano and watch to see if the base robot moves forward starting from a halt position.

Expected Results: The Teensy will properly receive and convert the character array which will flag the motors to forward.

4.1.10 Arm Selection

Sunil Alexander designed and performed this experiment.

Objective: To find an arm that is small enough to fit on the chassis yet big enough to grab a cup.

Setup: Build various arms to see if they fit our standards.

Procedure: Connect the arms to the circuit Sunil made with the arduino attached that had arm code to test to see if each servo was working.

Expected Results: An arm that could properly fit onto the chassis and be able to pick up a cup.

4.1.11 Arm Functionality

Sunil Alexander designed and performed this experiment.

Objective: To find angle positions for the robot arm that allow it to pick up a cup that is one foot away.

Setup: Sunil built a breadboard setup that had 2 joysticks attached to it and an arduino with code that Sunil wrote.

Procedure: The code functioned so that when moving the joystick, the servos would move the arm and the angle values would print on the screen. So Sunil attached the arm to the chassis and Grant read the values from the computer and wrote down the values.

Expected Results: When the robot is in the right position to pick up the cup, the arm should be able to pick it up with no issues.

4.1.12 Battery Selection

Grant Beatty and Sunil Alexander designed and performed this experiment.

Objective: To find a battery that could adequately power the Jetson Nano as well as the main Teensy and motors.

Setup: Gathering the Jetson Nano and attaching it to the chassis along with the batteries.

Procedure: Plug in the various power sources and see which ones work

Expected Results: Optimally the smallest battery tested is able to power the systems.

4.1.13 Chassis Construction/Mounting Methods

Sunil Alexander designed and performed this experiment.

Objective: Design and construct the chassis to create equilibrium and provide versatility.

Setup: Velcro and wooded platforms were utilized to help attach various parts to the PCB.

Procedure: Experiments were performed with different wrapping and placement techniques for the arm, Jetson Nano, and perf boards. Longer spokes were attached to the robot arm. Then, they were wrapped in velcro and attached to the front of the board. A wooden platform was created to house the Jetson Nano and was mounted on top of the motors in the rear of the robot. The spacing that was provided by the wooden platform allowed for the camera from the Jetson Nano to be threaded through the bottom of the arm and be attached to the front of the arm with velcro.

Expected Results: A chassis that could be taken apart without damaging any components as well as being stable enough to have optimal PID control testing.

4.2 Experiment Design , Data Analysis, and Feasibility

4.2.1 Sonar Sensor

Results:The sonar sensors worked splendidly and gave lower distance values when the obstacle was close and higher values when the object was far away. The sonar gave readings in the 100-950 range depending on the distance an obstacle was away. If it was too far away the readings were in the 10,000's.

Feasibility: Yes

4.2.2 Basic Motor Control

Results: After some initial debugging the motors did all of the basic motor traversal functions in the correct order.

Feasibility: Yes

4.2.3 IMU PID Controller

Results: The IMU PID controller would go straight for 1 12" cell. After that the accumulated error drift was too much and it wouldn't go straight for more than 4 seconds consistently.

Feasibility: No

4.2.4 Differential Velocity PID Controller

Results: The robot did not go straight.

Feasibility: No

4.2.5 Concurrent Separate Velocity PID Controller

Results: The robot went straight.

Feasibility: Yes

4.2.6 Pathfinding Simulation

Results: The Simulation Code had the robot found a path to the goal from the start for a variety of hardcode obstacle coordinates.

Feasibility: Yes

4.2.7 Computer Vision Methods

Results: All of the methods were unsuccessful except for the color filter which was marginally successful.

Feasibility: Yes (for color filter)

4.2.8 Camera Test

Results: The USB camera worked but there was no easy method to access the camera data from a Python script so this option was ruled out. The Pi camera that was a success was the Pi camera V2.

Feasibility: Yes (for Pi camera)

4.2.9 Subsystem Communication Teensy to Jetson Nano

Results: The robot went forward after successful serial communication.

Feasibility: Yes

4.2.10 Arm Selection

Results: An arm was purchased and modified with a custom gripper extension that enabled it to pick up a small cup.

Feasibility: Yes

4.2.11 Arm Functionality

Results: The values that were acquired from this didn’t have to be changed and picked the cup up multiple times, so we obtained successful arm functionality.

Feasibility: Yes

4.2.12 Battery Selection

Results: The regular phone charger power bank worked for the Jetson Nano but it did not adequately power the other systems. However, the INIU power bank was successful at powering them.

Feasibility: Yes

4.2.13 Chassis Construction/Mounting Methods

Results: There were many times that the PCB needed inspection and having the ability to take different parts off with ease was very important. The chassis performed very well considering it was able to accurately navigate through objects, locate, and pick up the cup.

Feasibility: Yes

5. Architecture and High Level Design

5.1 System Architecture and Design

Figure 5.1 is the full block diagram of this project. It illustrates the tasks each MCU is designed to do and how each of the MCU's communicate to each other. MCU1 processes the Computer Vision tasks which include the centroid data and the script startups. MCU2 controls all communication between the MCU's and is the central finite state machine control. Lastly, MCU3 controls the PWM signals sent to the robot arm.

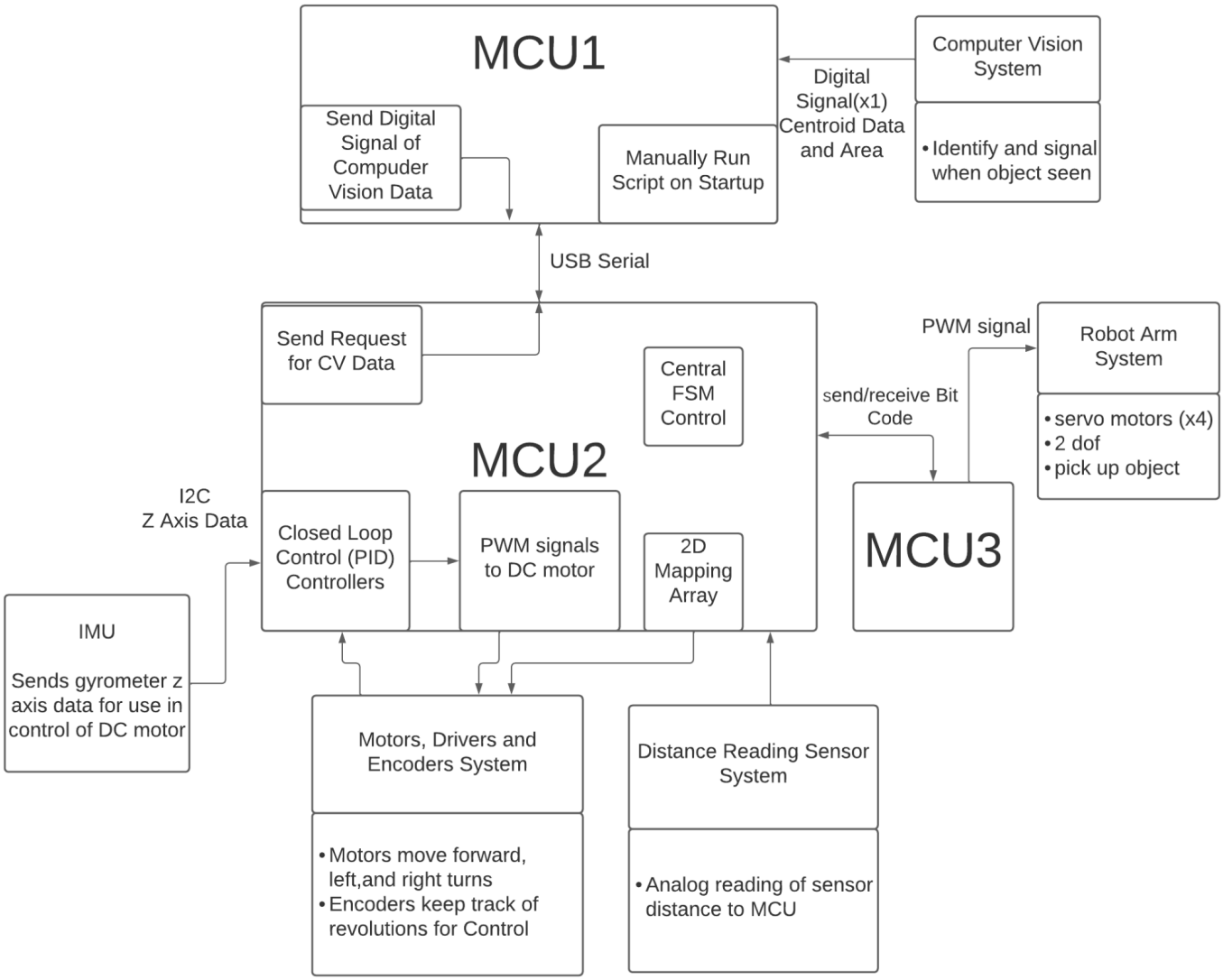

Figure 5.1: System Block Diagram

5.2 Hardware Architecture

Figure 5.2 illustrates a block diagram of the hardware architecture of this project. The diagram depicts how power is moving from the sources throughout the various devices of the robot.

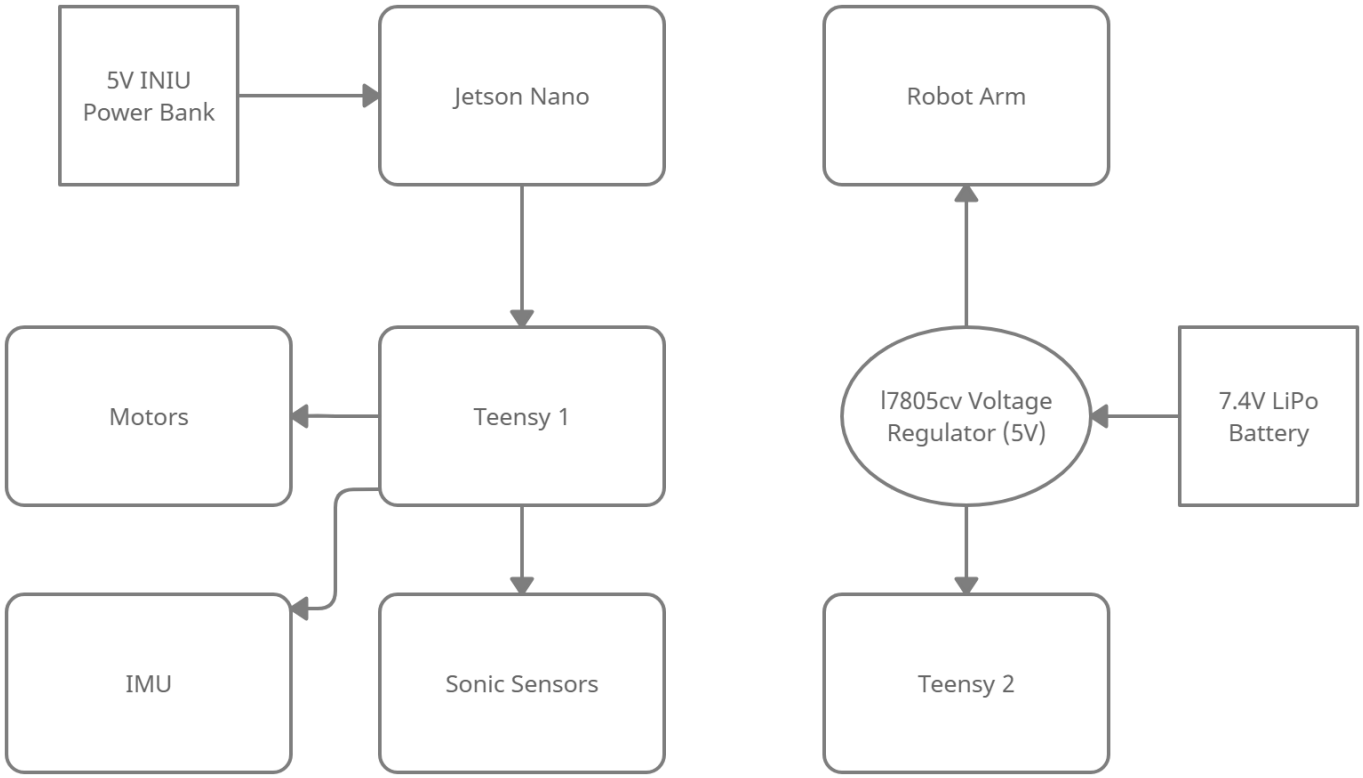

Figure 5.2: Hardware Block Diagram

5.3 Software Architecture

Figure 5.3 illustrates the finite state machine of the software architecture of this project. The finite state machine shows all of the primary commands and operations of the system. The robot uses the Treamux search algorithm to move around a set space. On each cell in the space, obstacles are scanned using the sonic sensors. This information is utilized by the search algorithm to determine the orientation of the robot before it moves forward to the next cell. In the FSM diagram this is represented by Turn Right, Turn Left, Turn Reverse, and Go One Cell states. After reaching the goal cell, a set sequence of states is executed starting with the Find Cup state. The camera scans for the cup and enters the remaining states based on its findings.

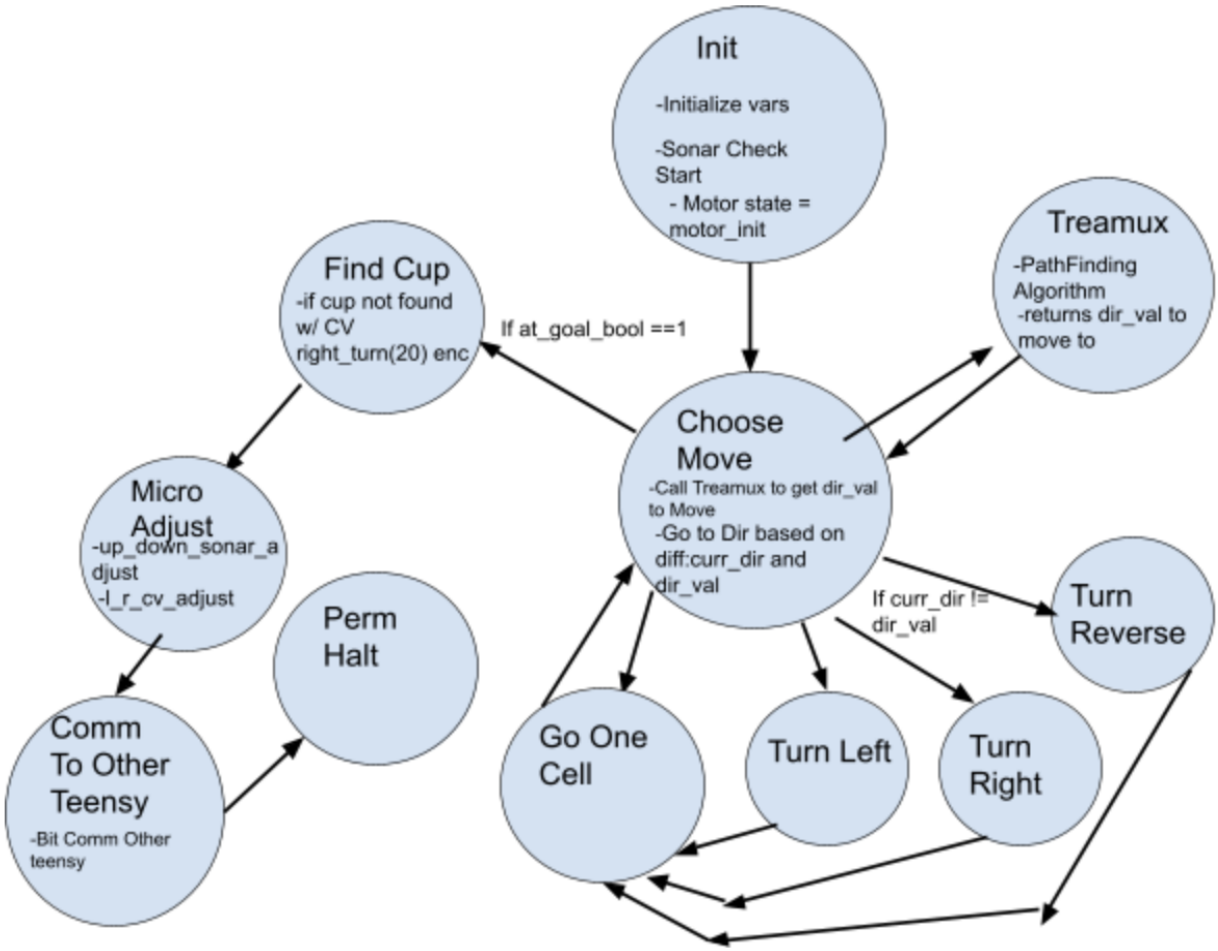

Figure 5.3: Finite State Machine

5.4 Rationale and Alternatives

Originally, the project was tested with the cell phone power bank which was small in size and fit on top of the robot with minimal obstruction. It was then discovered that this small power bank did not have enough voltage supply to power the robot. We then opted for a larger power bank that would fit underneath the chassis of the robot.

	INIU Power Bank (10000mAh)	Cell Phone Power Bank (2200 mAh)
Cost	20$	6$
Weight	0.32 kg	0.08 kg
Dimensions	13.2 x 6.9 x 1.3 cm	9.5 x 2.2 x 2.2 cm
Power	(3A)(5V)	(1A)(5V)

Figure 5.4: Battery Analysis

Initially, the Jetson Nano was set to be the primary controller of the system in the original design. However, serial communication between the Teensy and Jetson Nano was a difficult task. This attribute was developed towards the end of the project so the Teensy was used as the main computer because all the functions that integrated the IMU module, motors, and sonic sensors were already on this microcontroller early on in the development of the kitchen helper.

The Raspberry Pi may have been the superior choice of MCU to run the computer vision. It is cheaper, smaller, and consumes less power (Figure 5.5). The primary difference between these devices is that the Jetson Nano has a GPU which gives it better image processing power. However, this aspect goes unused. The initial designs for the image processing used machine learning methods but a more simple and fundamental approach was developed using OpenCV instead. The CV algorithm that was used likely would have functioned the same on a Raspberry Pi.

	Jetson Nano (2 Gb Model)	Raspberry Pi 4 (2 Gb Model)
Cost	200$	130$
Processing Power	1.43 Ghz 64 bit quad core	1.5 Ghz 64 bit quad core
RAM	4GB with 1600 MHz	4GB with 1600 MHz
Power	5W-10W	2.56W-7.30W
Wi-Fi	not built in	built in
GPU	128 core GPU	no GPU

Figure 5.5: CV MCU Analysis

6. Data Structures

Base Robot Teensy

Dylan programmed, assembled and debugged this code.

senior_design_proto_5.ino

- The main file for the main Teensy code, this imports all of the other required header files that the robot uses to run, also includes the setup which sets up all the pins and sets up the starting state for the main FSM. Then all that's left in the main file is to call the motor_tick() function to call the FSM.

Dylan programmed, assembled and debugged this code.

global_values.h

- The header file here includes all of the variables used in the main robot program. It's in a separate highest priority header file to make sure the variables here can be used in any file of the robot.

Dylan programmed, assembled and debugged this code.

gyro_func.h

- The header file here includes all of the gyro functions such as the update_gyro function and the complementary filter used with it as well. All of this is to get the angle of the robot during turning functions

Dylan programmed, assembled and debugged this code.

misc_functions.h

- The header file here includes misc functions that don't really fit anywhere else in the code. This mostly includes PID functions and PID controller classes that were not used in the final build.

Dylan programmed, assembled and debugged this code.

motor_func.h

- The header file described is an integral part of the entire code. Not only does it contain the traversal functions such as forward, halt, reverse etc., it also includes the working PID controller class of the separate concurrent velocity for the left and right motors, the encoder PID function that calculates the velocity and then calls the PID controller class and then adds the P_term and D_term to the left and right pwm values, it also includes the go_one_cell function (which is the movement function that makes the robot go approximately 1 12" cell length,also it has both of the micro adjust functions for the sonar adjust and the cv left and right adjust ,and also includes the entire FSM which handles what the entire robot does and when.

Dylan programmed, assembled and debugged this code.

sonar_func.h

- The header file includes all of the functions used to get the distances values from the sonar, it also includes a separate get_sonar_dist_test function that has a delay and prints out the values to the

serial monitor for use if we need to test if the sonars work real quick instead of commenting and uncommenting the delay and prints in the main get_sonar_dist function.

Dylan programmed, assembled and debugged this code.
treamux_func.h

- The header file handles the entire pathfinding algorithm, so it includes the custom struct class for the coordinates (coord in the code) that has the y and x coordinates , the visited_num int (the number of times a coord is visistied. It also includes the functions of neighbors_func which when given a coord returns the 4 neighbors to it the up, right , left and down neighbors in that order. In the main treamaux_func first looks for obstacles with the sonars to set the coord struct as inaccessible. Then it does the pathfinding algorithm with all available neighbors and then returns the optimal direction as per the algorithm. The entire main function is called once in the CHOOSE_MOVE state of the main FSM.

CV Jetson Nano Code

Dylan programmed, assembled and debugged this code.
ReduceSizeAndPad.py

- The function described takes the frame array and pads the outside of the frame as well as reduces the size of the frame using built-in functions.

Dylan programmed, assembled and debugged this code.
simplified.py

- The function applies a color filter on the frame data so that the only pixels highlighted on the frame are specific shades of red, green, and blue.

Dylan programmed, assembled and debugged this code.
IsolateObj.py

- The function uses connected component sequential labeling to identify all of the objects in the frame. Objects that do not meet the size threshold are filtered out as well as objects that do not fit the shape criteria. The centroid of the largest object is calculated and returned.

Dylan programmed, assembled and debugged this code.
six_chars.py

- Splits the centroid data into a six char array.

Dylan programmed, assembled and debugged this code.
serial_test.py

- Calls six_chars to format the centroid data then sends each digit 1 byte at a time through serial communication.

Dylan programmed, assembled and debugged this code.
test2.py

- The main file gathers frame data from the camera at 10 frames per second and combines all of the above CV functions. It sends the centroid data of the target object if a ‘z’ character is sent by the Teensy through serial communication.

Robot Arm Teensy

Sunil programmed/assembled and Grant assembled/debugged this code.
grant&sunil's_servo_code.ino

- The code runs the robot arm grab function when it receives a high signal from the other Teensy. It also sends a signal back when it is done.

7. Low Level Design

There were multiple schematics and low level design circuits that were integral to this project. The robot chassis doubled as a PCB that connected most of the devices in the circuit. The PCB was designed based on the specifications of the microcontrollers. The schematics for this project were created as a blueprint before soldering. The connections were checked with a multimeter to ensure that current flow was expected and then the finite state machines were created to illustrate how the commands functioned in our code.

7.1 Base Robot Circuit

- This is the base robot circuit for the kitchen helper robot

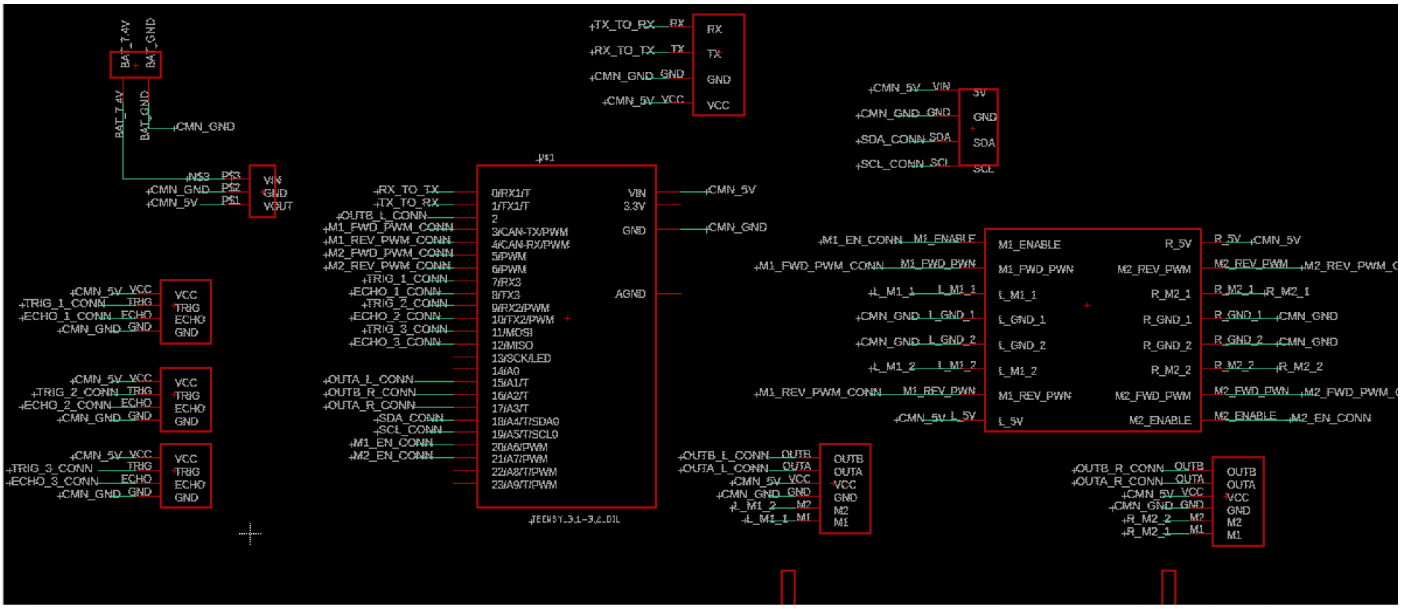

Figure 7.1: PCB Circuit Schematic

- The figure above shows the main circuit involving the sonars, power regulator to battery connection, the Teensy 3.2 used as the main MCU, the HC-05 Bluetooth module, the IMU using I2C connection , the motor driver , and both motor encoder pins headers. To make the schematic easier to debug, label connections were used.

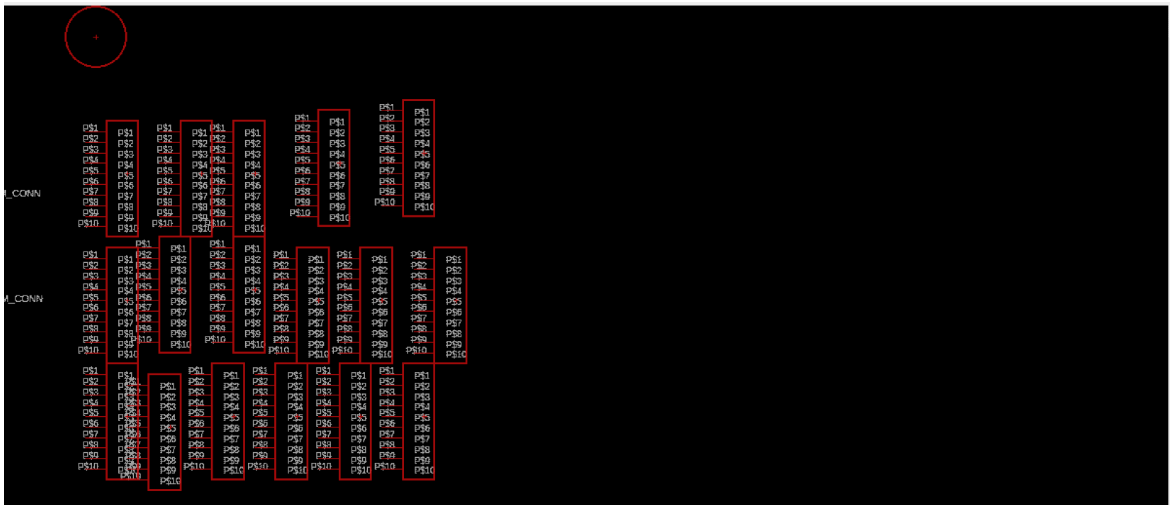

Figure 7.2: Main MCU Circuit

- Above are the test rows that were used for hardware testing different components without the use of the main PCB.

7.2 Base Robot PCB

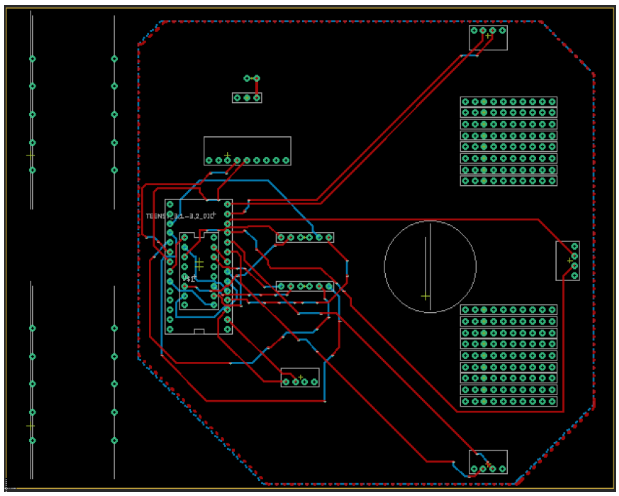

Figure 7.3: PCB Circuit Board

- The image of the PCB above includes all of the trace connections between all of the components. It also has 2 layers of connections. The top red layer was the one used the most. Also 2 polygon layers were used for Common 5V and Common ground. These layers were used to cut down on the amount of traces needed. The SN7 motor driver was placed beside the teensy to save space since it could fit in heighwise beneath the teensy.

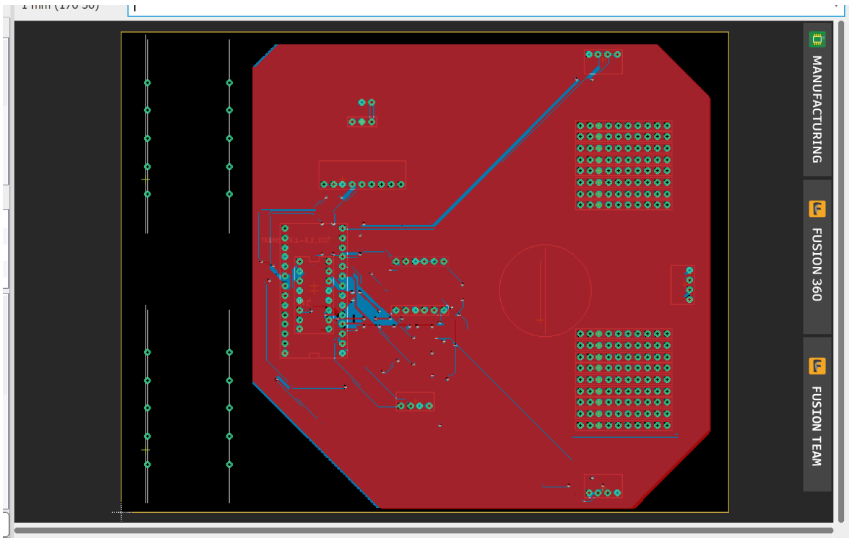

Figure 7.4: PCB Ground and Power Layers

- The image above is of the ground and power layers when the airwires are optimized making it clear to see where the 5V and Gnd layers are (they mostly cover the same area so the blue is covered by the top red layer)

7.3 Arm Connection Schematic

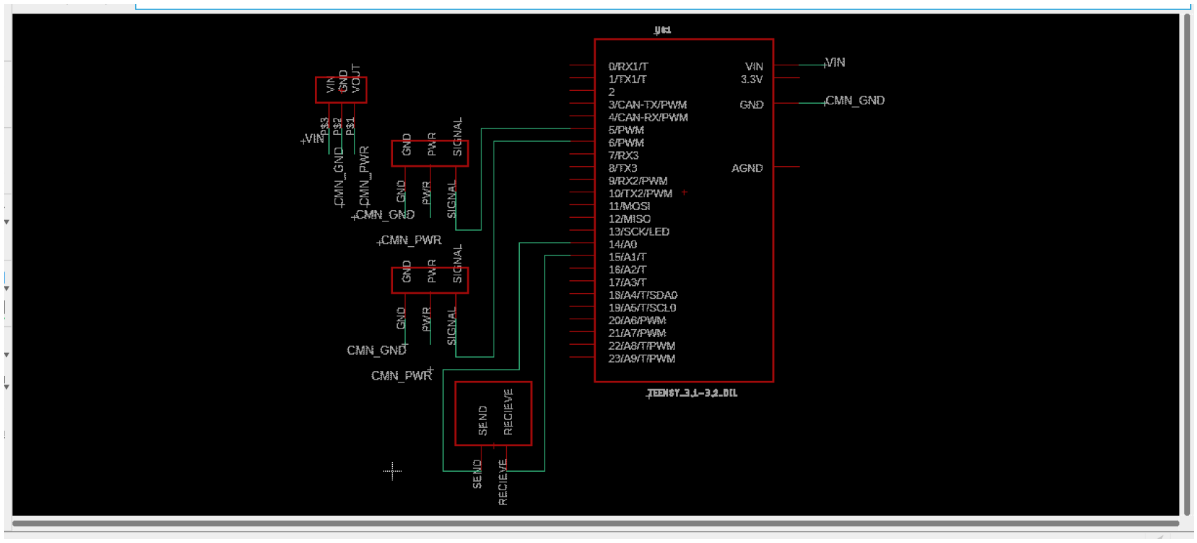

Figure 7.5: 2nd Teensy Circuit Schematic

- The arm had a 5 V and ground line for each of the servo motors as well as a PWM signal line connected to a Teensy. Both microcontrollers communicate through send/receive pins that were custom made.

7.4. Total FSM Flowchart

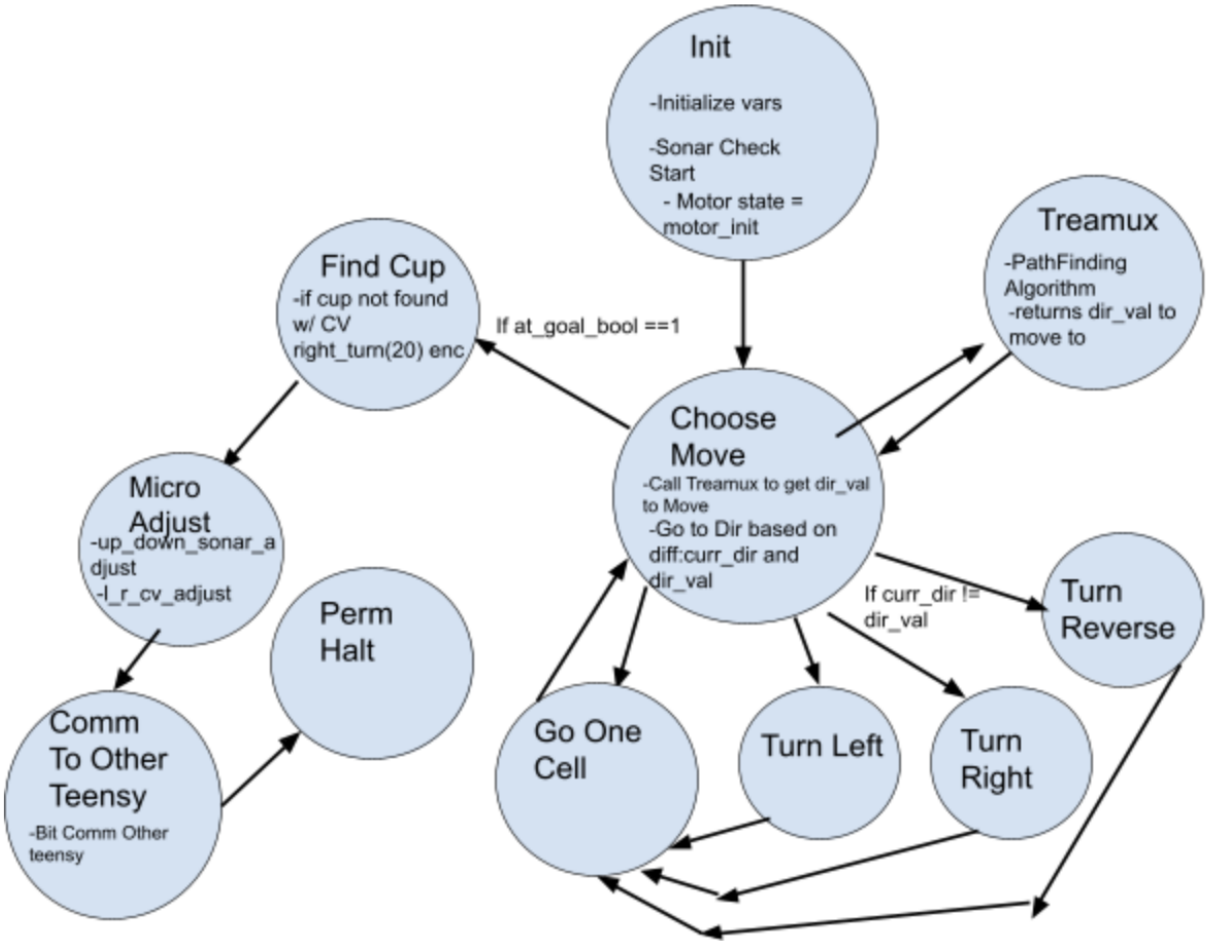

Figure 7.5: Total FSM Flowchart

7.5 Go_One_Cell Flowchart

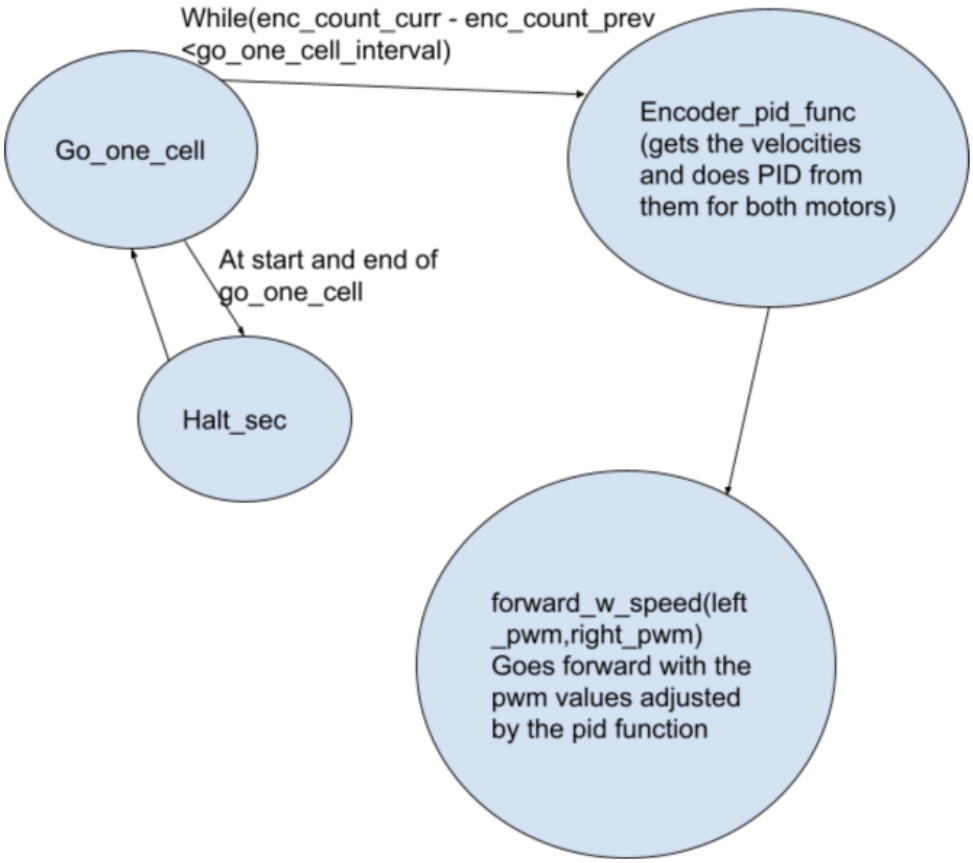

Figure 7.6: Go_One_Cell Flowchart

7.6 CV Micro Adjust Flowchart

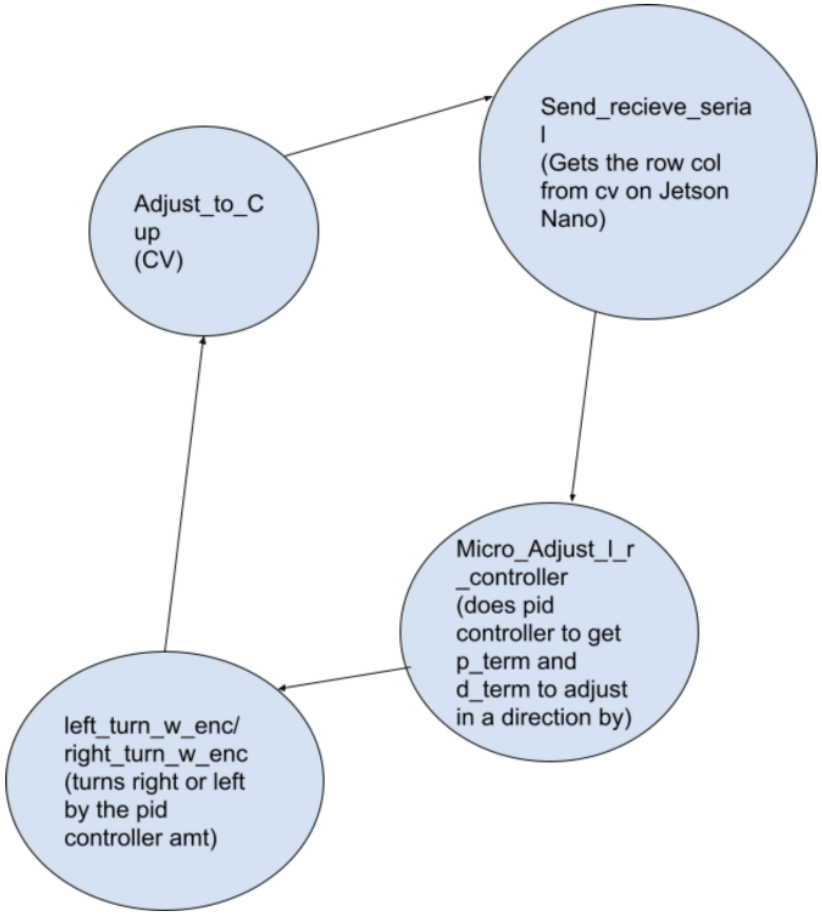

Figure 7.7: CV Micro Adjust Flowchart

7.7 Sonar Micro Adjust Flowchart

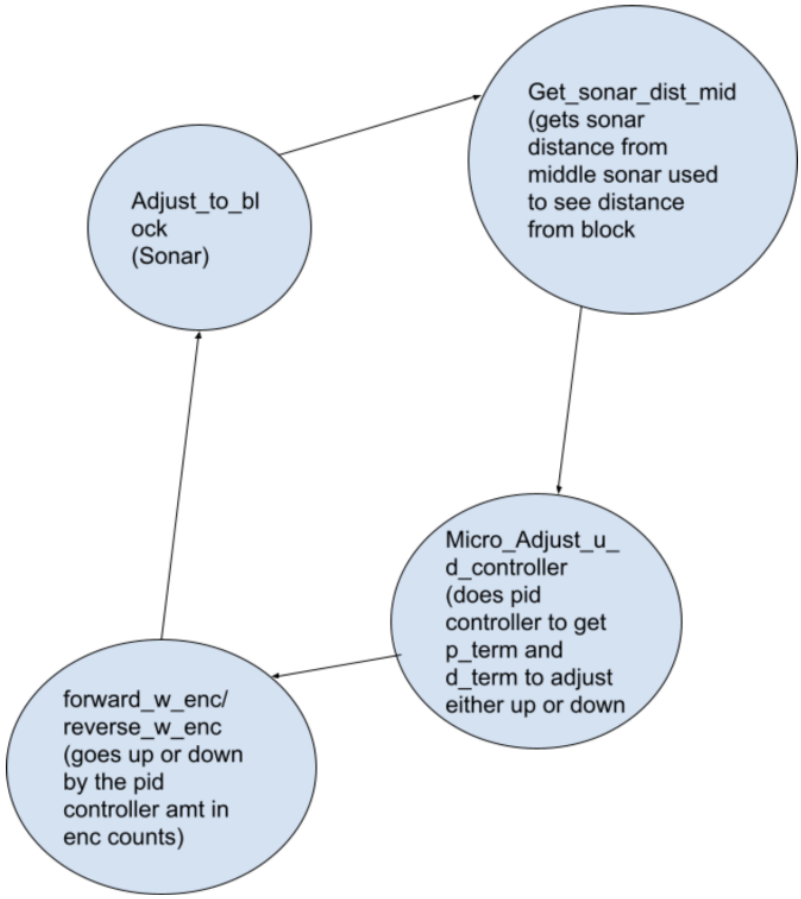

Figure 7.8: Sonar Micro Adjust Flowchart

7.8 Tremaux Flowchart

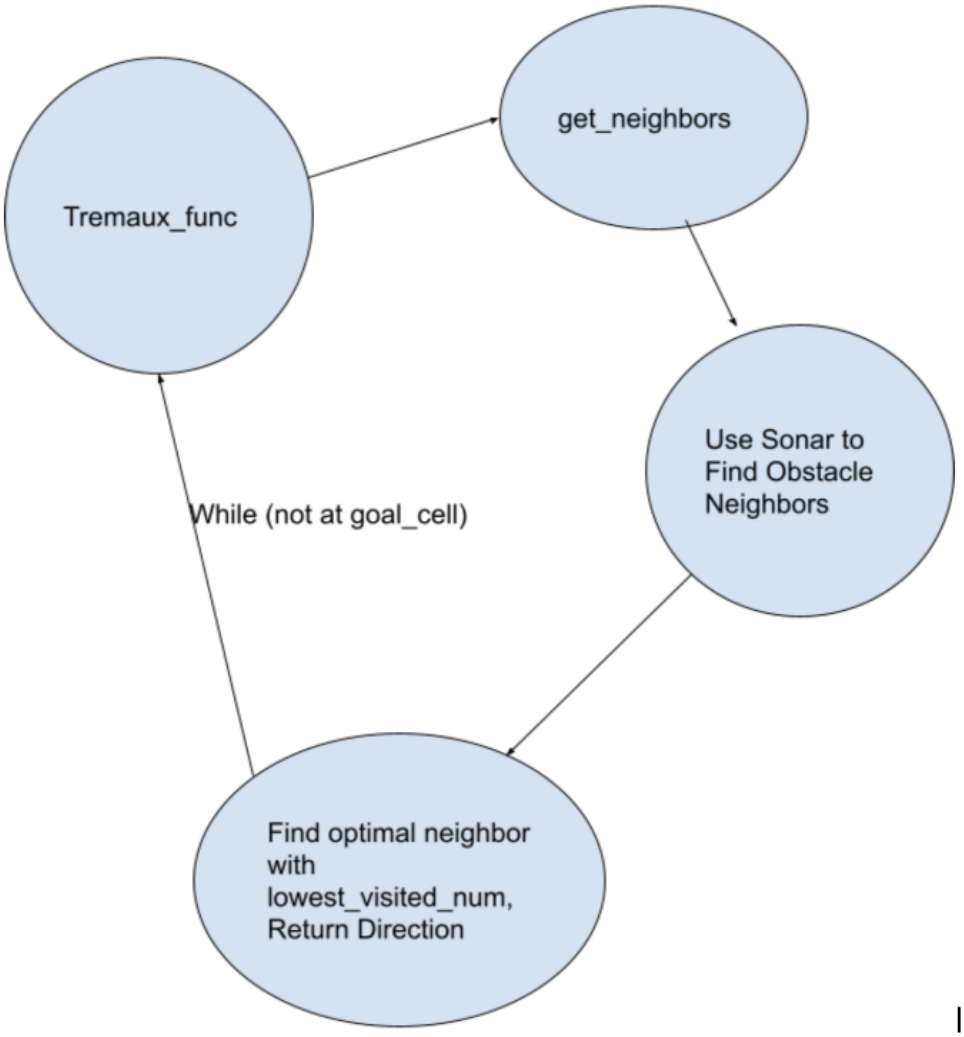

Figure 7.9: Tremaux Flowchart

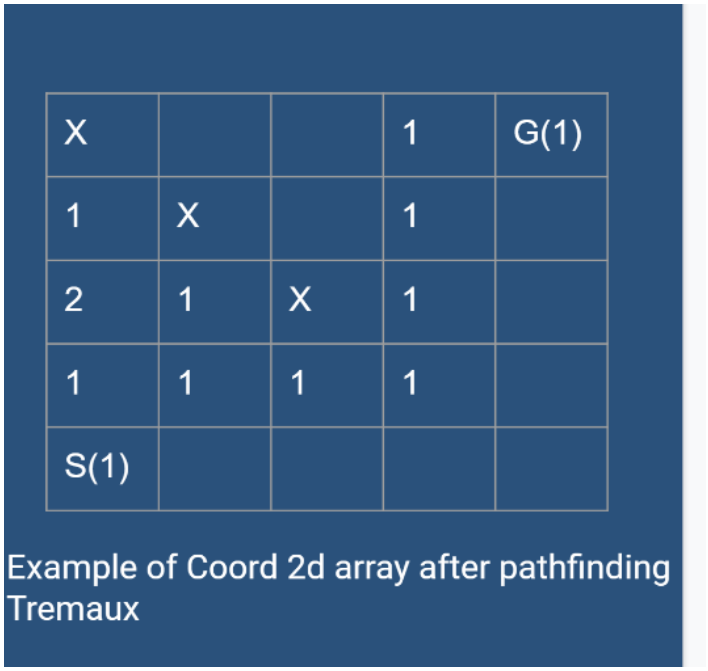

Figure 7.10: Tremaux Algorithm Example

7.9 Extraneous Explanation

- The PCB was a custom part built on a platform called Eagle and the library was also custom made.

8. Technical Problem Solving

8.1 Problems

While designing and constructing the Kitchen Helper Robot there were nine problems that were prominent. Most of these problems were hardware issues involving single systems; only a couple of them were multisystem issues. The functionality of our robot would have been greatly reduced if the identified issues were not resolved.

8.1.1 Sonar Distance Issues

Dylan McGee identified and solved this problem.
The Sonar sensors would sometimes produce high values even in instances where the reading before and the reading after are the same.

8.1.2 Going Straight PID Issues

Dylan McGee identified and solved this problem.
Finding the right PID controller was an issue since multiple PID controllers did not allow the robot to go straight.

8.1.3 Pathfinding Issues

Dylan McGee identified and solved this problem.
Choosing an appropriate pathfinding algorithm was a challenge for implementations not just for Treamux but for A* search and flood-fill as well. There were issues with the algorithms due to coding in C++ as well as the embedded system environment which produced recursion for floodfill due to the limited stack space on the microcontroller.

8.1.4 Accumulated Traversal Error

Dylan McGee identified and solved this problem.
Since the robot does not acquire sensor or camera data to assist with moving, accumulated error happens. Small differences in terms of going straight or the correct encoder count result in large displacements from the intended position. The robot needed to be able to traverse the 5 by 5 cell space without readjustment.

8.1.5 CV Color Problem

Grant Beaty identified and solved this problem.
A color filter was used to isolate the object. Determining the thresholds for the colors was a challenging problem to solve. The RGB color values could not simply be guessed. There were 3 kinds of cups that needed isolation: 1 Red, 1 Green, and 1 Blue.

8.1.6 CV Object Noise

Grant Beaty identified and solved this problem.
After the color filter was working there was still some additional noise left in the processed images as well as background objects. This caused the centroid data to not accurately represent the center of the objects that needed to be isolated.

8.1.7 First Iteration Arm Issue

Sunil Alexander identified and solved this problem.
The initial arm this project was going to use was called the meArm which included 4 servos that had 4 different functions. A gripper function, a turn function, an extension and retraction function, and an up and down function. Two variations of this arm were bought, one made out of wood and one made out of aluminum. This arm was good but it appeared to have too many servos and the flat bottom would have made it hard to mount.

8.1.8 Chassis Issue

Sunil Alexander identified and solved this problem
The chassis had multiple mounting issues that needed to be fixed for the robot to function properly. The Jetson Nano, robot arm, arm teensy circuit board, and battery didn't have proper mounting ideas in order to keep it balanced and PCB components easily accessible.

8.1.9 New Arm Gripper Issue

Sunil Alexander identified and solved this problem
The new arm that was acquired did not have a gripper wide enough to fit the cup. The demonstration would communicate the proof of concept more if the robot could pick up an actual cup.

8.2 Solutions to the Problem

8.2.1 Sonar Distance Issues

To fix this the average of several readings for the final distance was used, a low pass filter was made in code in order to filter out high sonar values.

8.2.2 Going Straight PID

This was fixed by using the finalized 2 separate concurrent velocity controllers as specified in section 4. Also, the best way to get the velocity was to use the double data type and also divide the difference in encoder counts of current - prev by a constant time so a millis timer was used.

8.2.3 Pathfinding Issues

This was fixed by pivoting to the Tremaux algorithm since the way it was formatted lended itself to it just returning a value for each direction the robot decides to go through which lends itself well to the state machine well since the robot could just call it and get the next direction versus the path return of A* search. Also treamux was easier to integrate the obstacle detecting function as well, since the neighbor cells updated and use sonar to detect obstacles while traversing it compared to A* star search knew where the obstacles were beforehand. Overall though, using the Tremaux algorithm was the right call in terms of making the pathfinding work with the rest of the project and being functional on its own.

8.2.4 Accumulated Traversal Error

This is the only issue that was not resolved with an elegant solution since adjustiting for the small minimal error that happens while going straight or the small minimal amount of travel distance error required extreme precision.

8.2.5 CV Color Problem

To create the color filter pictures were taken of each of the 3 cups, which required isolation, in various lighting conditions. The mean, median, minimum and maximum of the RGB values of each cup were calculated. This data informed the thresholds for the color values used to isolate the colored cups.

8.2.6 CV Object Noise

To reduce noise an attempt was made by first eroding and then dilating the image. This method was good for small pixels in the frame but was not good for larger error objects. This issue was fixed by using connected component sequential labeling. This way the objects could be identified in the frame and sorted. The objects were filtered out by length-width ratio and size. This reduced the objects down to only the cups. Then the algorithm selects the largest of the cups in the frame (see figure 8.1).

(Original) (Color filter) (Size-Shape Filter)

Figure 8.1: CV Filters

8.2.7 First Iteration Arm Issue

This issue was fixed by constructing a new arm where we would only use 2 servos.

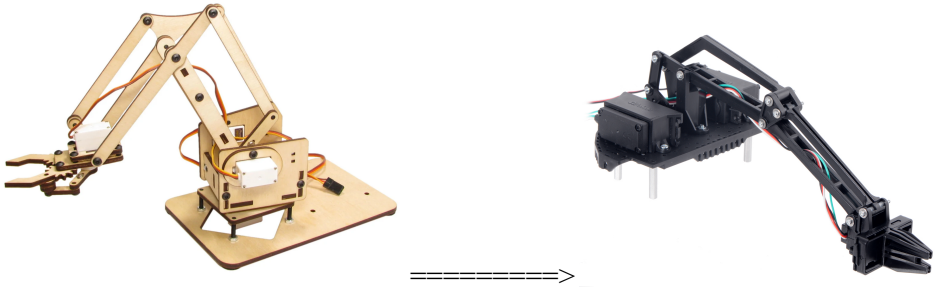

Figure 8.2: First Arm and Final Arm

8.2.8 Chassis Issue

This issue was solved with different types of mounting techniques. The arm was mounted with velcro so that we could take it off and make repairs. A platform was built so that the Jetson Nano can be easily accessed and we wouldn't have to use putty. The 2nd teensy perf board was attached to the Jetson Nano and the arm. The battery pack was velcroed underneath the PCB.

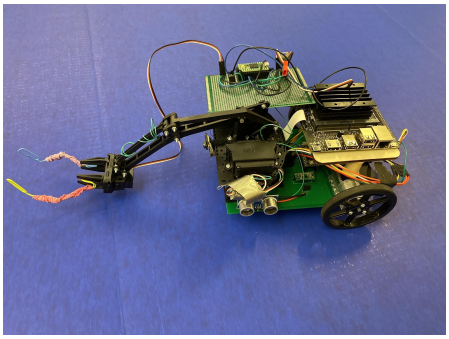

Figure 8.3: Full Robot Chassis

8.2.9 New Arm Gripper Issue

This issue was solved by creating various versions of an arm extension, and with a large amount of testing, An efficient arm gripper was constructed that can be formed to meet whatever shape we wanted to pick up. The extension was wrapped in rubber in order to increase grip.

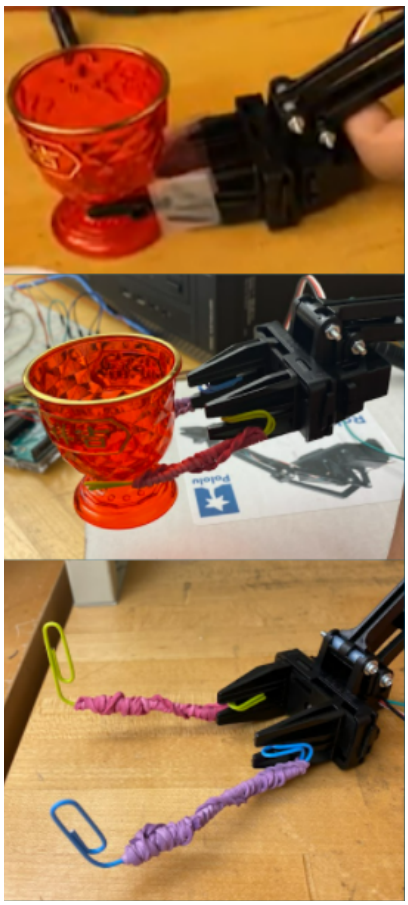

Figure 8.4: Gripper Iterations

9. Test Plan

9.1 Test Design

9.1.1 Test 1: Turning

Dylan McGee designed and performed this test.

Objective: The purpose of this test is to see if the IMU module can be used to accurately turn the chassis of the robot 90 degrees in an accurate manner.

Setup: The code for accessing the IMU data must be created and the motors must be properly set to turn until reaching the desired 90 degree angle.

Procedure: Place the robot on the ground and run the code to observe how far it turns.

Expected Results: Optimally the robot turns around 90 degrees with an error range of 2.3 degrees.

9.1.2 Test 2: Computer Vision

Grant designed and performed this test.

Objective: Run the CV code that was developed on the Google Colaboratory on the Jetson Nano.

Setup: Port the code files onto the Jetson Nano and use them to perform CV algorithms on each frame of the camera stream.

Procedure: Run the camera, place a green cup in view of the camera, and print the bounding box and centroid values of the cup onto the screen.

Expected Results: The camera should be able to detect the cup and display centroid values consistently without any error.

9.1.3 Test 3: Pathfinding

Dylan and Grant designed and performed this test.

Objective: To traverse its surroundings properly the robot must sense its surroundings and place any obstacles that it finds onto the mapping array. It must also avoid all obstacles and follow the Trémaux search algorithm which should land the robot at its final destination.

Setup: By this point the turning and movement of the base motors PID controllers should be fine tuned and the sonic sensors should be properly calibrated.

Procedure: Set up various test obstacles in the 5 foot by 5 foot environment and run the code on the robot to see what happens.

Expected Results: The Trémaux algorithm should work and the surrounding objects should be detected as obstacles and the robot will always find a path to the goal cell.

9.1.4 Test 4: Micro Adjust Function

Dylan and Grant designed and performed this test.

Objective: The robot needs to be in a particular position and direction so that the grab command on the arm can properly pick up the cup. We used the CV and sonar data to accomplish this.

Setup: PID controllers will be needed for the centroid data and sonic sensor data in order to properly control movement. The communication between the Jetson Nano and Teensy should be set up allowing the Teensy to utilize the centroid data and the sonic sensor should be calibrated.

Procedure: Attach the camera onto the base of the robot arm so that it is in the front and center of the device. Place a cup on a platform in front of the robot. Run the files on the Jetson Nano and Teensy.

Expected Results: Using the CV data and the sonic sensor data the robot should be at the proper proximity and angle for the robot arm to pick up the cup.

9.1.5 Test 5: Teensy Communication

Sunil Alexander designed and performed this test.

Objective: To make sure that the main Teensy can communicate with the arm Teensy to perform the grab commands.

Setup: Connect the input and output pins of the Teensys so that they can communicate with high-low signals. Connect the arm teensy to the robot arm.

Procedure: Run the code on both Teensys and make sure that the grab commands run when they are supposed to.

Expected Results: The arm should do nothing for a delay time then perform the grab command.

9.1.6 Test 6: Grab Function

Dylan and Grant designed and performed this test.

Objective: To have the robot identify a cup that is placed in front of its camera and have it move into position and pick it up.

Setup: Attach all parts of the robot together and make sure that the other subsystems work.

Procedure: Place a cup on a platform in front of the robot and run the experiment.

Expected Results: The robot should align and turn itself properly and then perform the grab command to pick up the cup.

9.1.7 Test 7: Final

All were involved in designing and performing this test.

Objective: At this point the robot should be mostly working. The robot should traverse through the environment, reach the goal cell and pick up the cup.

Setup: Calibrate all of the subsystems and make sure that they work. Make sure the camera is aligned properly.

Procedure: Run the experiment with various obstacles and see which of them work.

Expected Results: The robot should be able to perform all functions properly.

9.2 Bug Tracking

In terms of bug tracking, when testing subsystems we kept track of which aspects were malfunctioning at any one time. For example when testing the traversal functions, we kept in mind what part could cause which problem such as when the robot was not seeing an obstacle the issue was either sonar or mapping related or if the robot was not going straight that was the PID. If the micro adjust stage was not going well there likely needed to be tweaks to the desired centroid value or a PID fix in the micro adjust states of the FSM. Bug tracking was done mostly as a combined effort among the group members.

9.3 Quality Control

Quality control procedures were a priority in order to minimize design errors. The lab room and workstations were organized and cleaned. Multiple parts were bought in case anything breaks. The circuits were tested with a multimeter so the connections were stable.

9.4 Identification of Critical Components

9.4.1 Pi Camera V2

Since the Raspberry Pi camera was a critical component of the project, it should be placed properly in front of the robot to read the data properly. If it is not placed properly then the target centroid value in the main Teensy must be recalibrated.

9.4.2 Jetson Nano

The Jetson Nano was a critical component because it is the most expensive part and so replacing it would be costly.

9.4.3 Sonic Sensors

The sonic sensors were critical components because the soldering joints on them were fragile and they would break easily.

9.4.4 SN754410 Motor Driver

Placing the motor driver correctly was integral to the functionality of the project. There were issues when attaching it to the PCB because its pins would become bent which would then harm its connection to the motors. Because of this, preliminary testing was required to establish that the motor driver was properly connected.

9.5 Items Not Tested by the Experiments

9.5.1 l7805cv Voltage Regulator

The voltage regulator was not tested by itself because it was assumed that it would work.

9.5.2 Bluetooth Module

The Bluetooth module would have allowed the user to control the kitchen helper robot remotely. The bluetooth module was already attached to the PCB. However, it was never tested or implemented because of time constraints.

10. Test Report

10.1 Test 1: Turning

Iteration 1:

1. The result of the test was overturning when going right and underturning when going left.
2. The chassis was significantly off from turning 90 degrees in both directions.
3. We had expected the IMU to be sufficient by itself for turning. We had not expected that the drift of the IMU module would have a significant effect.
4. For the next test we recalibrated the target turning angle based on this information.

Iteration 2:

1. The result of the test was that it showed that the robot was able to turn properly.
2. The turning was within the 2.3 degree margin of error which was expected.
3. The tweaking of the values worked to improve the accuracy of the turning in an efficient manner.
4. No further tests were performed because the results were sufficient.

10.2 Test 2: Computer Vision

Iteration 1:

1. The result of this test was that it showed that CV code on the Jetson Nano is capable of recognizing red and green cups.
2. The actual results were less promising than what was expected.
3. The cups could be recognized but only from specific lighting conditions and angles. Rotating them slightly would cause the cups to be not recognized by the camera.
4. For the next test we removed the Hu Moments requirement from the shape filter and added more lighting to the viewing environment.

Iteration 2:

1. The results of this test showed more consistent object tracking.
2. The results met expectations.
3. The cups were tracked consistently with very little error values.
4. No further tests were performed because the results were sufficient.

10.3 Test 3: Pathfinding

Iteration 1:

1. For the first iteration the PID tuning was off and so the robot was not moving straight.
2. The cell traversal algorithm appeared correct but the forward motion of the robot had a tilt and it was way off center in terms of being at the goal cell.
3. The obstacles were avoided but this test only used pre-coded obstacles for the pathfinding algorithm.
4. In the next test the PID controller will be fixed.

Iteration 2:

1. With the fixes to the PID controller the robot was traversing much more accurately.
2. The goal cell was reached and the pathfinding algorithm appeared correct.
3. Tuning for the PID controller allowed for accurate movement of the robot and showed that the movement algorithm appeared correct.
4. In the next test the algorithm will not use precoded obstacles. Instead, obstacles will be detected by the sonar sensors.

Iteration 3:

1. Issues with the sonar readings caused the robot to run into obstacles or read obstacles that were not there.
2. The results were not expected since the robot was not detecting obstacles correctly.
3. Sonar sensor noise appeared to cause many problems.
4. For the next iteration we added a filter for the sonar sensor and we sampled multiple readings to make sure the selected reading was correct

Iteration 4:

1. After the sonar sensor was fixed the robot was no longer running into obstacles.
2. The results were expected.
3. The filtering and averaging of inputs allowed us to utilize the sonar sensors properly
4. No further tests were performed because the results were sufficient.

10.4 Test 4: MicroAdjust Function

Iteration 1:

1. The computer vision controlled left and right movements are supposed to align the robot so it directly faces the cup. However, the robot was moving away from the cup.
2. The camera was placed upside down so the movements were inverted.
3. Moving the camera or changes to the code accounting for the inverted inputs should fix this issue.
4. The CV code was altered and fixed so that the upside down camera would not be an issue.

Iteration 2:

1. The robot was having issues aligning with the cup properly. It was not in the proper position to pick up the cup consistently.
2. The robot was not quite in position to pick up the cup.
3. The robot has significant errors in its movement and turning which needs to be reduced.
4. In the next test the robot will make multiple sets of adjustments to its proximity and direction relative to the cup.

Iteration 3:

1. The robot made multiple adjustments based on the sonar sensor and camera data which enabled it to move into the proper position.
2. The results were satisfactory because the robot was able to pick up the cup.
3. The error in the movement and turning was reduced with the use of several readings.
4. No further tests were performed because the results were sufficient.

10.5 Test 5: Teensy Communication

Iteration 1:

1. The communication between the Teensys was semi-functional.
2. The arm would keep performing the grabbing motion repeatedly. This was not ideal.
3. There appeared to be a repeated signal sent to the Teensy controlling the arm.
4. For the next test we added code to make sure the main Teensy only sends a high signal once and then sends low until it receives a signal.

Iteration 2:

1. The Teensys appeared to be communicating without issue.
2. The results were expected; the grab function ran once
3. The code fixed the issues with the repeated signal.
4. No further tests were performed because the results were sufficient.

10.6 Test 6: Grab Function

Iteration 1:

1. The gripper on the robot arm was not consistently picking up the cup.
2. The expected outcome was for the arm to fit properly around the cup and pick it up.
3. The gripper was too small to fit around the cup without modification.
4. Paper clips were added to extend the gripper and widen it and rubber bands were added to create better grip.

Iteration 2:

1. The gripper consistently picked up the cup with the new modification.
2. The results were consistent with the expected outcome.
3. The new gripper modification fixed the previous gripping issues.
4. No further tests were performed because the new gripper provided consistent successful results.

10.7 Test 7: Final

Iteration 1:

1. The tests would mostly always find a path to the final goal cell but sometimes there would be issues in picking up the cup at the end.
2. The consistency of the robot finding the cup at the goal cell was not ideal.
3. The accuracy of turning was slightly off so the robot was not always facing the goal obstacle.
4. Fixes were made to the PID controller to make the turning and movement even more accurate. Also, a new state was added to the robot that had it rotate and scan for the cup.

Iteration 2:

1. The small edge cases in which the robot did not detect the cup were minimized through the new find cup state in the FSM. Also, after more tuning of the PID controller the robot's movement had increased accuracy.

2. The resulting robot was able to consistently reach the goal cell with 90% accuracy
3. The changes to the PID and the addition of another state allowed the robot to function properly.
4. No further tests were performed because the results were sufficient.

11. Conclusion and Future Work

11.1 Conclusion

The Kitchen Helper Robot ultimately completed its tasks and proved that it is a viable proof of concept. If timed allowed this project would contain more features that provide its user with more functionality which is what we hope to include in future iterations of the design.

The technical objectives of the design required a combination of several elements of electrical engineering. We utilized state machines, robotics, computer vision, PID controllers, and several communication protocols that combined the subsystems. We extensively learned multiple concepts of electrical engineering to create a functional prototype that was successful in completing its objective.

Despite getting most of the primary functions to work successfully we were unable to get the robot to perform several functions. The robot initially was supposed to check each cell for the target object but we had to simplify our design so it only checks for objects adjacent to the goal cell. We also wanted to have the robot take the target object and move it to a bin near the start cell. This required extensive use of movement and pathing and that would have added significant complexity so this idea was never fully realized. Bluetooth control of the robot was also not implemented because of this reason.

Ultimately the robot is able to perform a variety of tasks involving the integration of various subsystems. Using PID controllers, the chassis is able to go straight and turn very accurately with DC motors, which is a tedious task in of itself. It successfully navigates through cells on a table top sized space using sonic sensors and mapping to detect and keep track of obstacles. Upon reaching the destination it uses computer vision and a robotic arm to identify and grab a target object.

The tasks that Grant was responsible for were computer vision, MCU integration, and testing design. Sunil was responsible for the robot arm, chassis design, and project management. Dylan was responsible for the base robot code, PCB design, and control systems.

Grant:

Throughout the project I learned a significant amount about solving physical problems using elements of engineering. I further developed my programming skills in several ways. I had never coded in python before or used the OpenCV library, both of which I used for this project. Most of my experience with CV was with basic algorithms in MATLAB. Initially there was a bit of a learning curve but Python is an easy language to figure out so I mastered it relatively quickly. I learned to never use for-loops in Python because they are very slow and that built-in CV functions are extremely fast. I got the opportunity to use the tools from my EE 146 CV class to solve technical problems which was a very engaging process. My knowledge of embedded systems and MCUs were improved by working with the Jetson

Nano and the Teensys. Figuring out how to perform serial communication on these devices was an interesting task. Working with a team was a useful experience.

Communicating deadlines, planning, developing, and meeting up for testing are professional skills that I will take away from this project. Figuring out how to research a problem also aided me significantly in the design process especially with learning Python. The importance of contingency plans was an aspect of designing that I initially underestimated at the beginning of the project. I will make sure to focus on this area significantly more in future projects. Overall the project was a success because of extensive cooperation and communication between myself and my group members.

Sunil:

Over the course of this design project I learned so many skills that would aid me in my engineering pursuits for years to come. I programmed the arm in C++ and therefore strengthened and retained my programming skills. I also created a test board and created a circuit that had joysticks and an arduino attached to it so I could conduct very thorough testing of the servos and the arm as a whole. I learned how to problem solve in a very efficient way while dealing with the arm extension process and chassis modification/constructing.

I was in charge of overall project management for this design project and it made me realize that I would like to possibly pursue a career in a field related to professional development. I organized our google docs in such a way that we had discussion notes from every lab section we attended so we could go over what needed to be done for the future. I created two comprehensive Gantt charts that I learned how to make on google spreadsheets, one for Fall quarter and one for Winter quarter.

Throughout this project I learned that the most ethical responsibility an engineer can have while making a product is to make sure that this product would make someone's life better. If that mentality isn't at the heart of our project then we would be doing our customers a disservice. I believe our Kitchen Helper Robot was designed with these ideas in mind and I am thankful for all of the knowledge I have gained from helping this project come to fruition.

Dylan:

I learned a lot doing this project since I did all of the base robot functions so all the traversal code, the pathfinding, the PID controllers, the FSM and the pcb design and construction of the base robot, I also handled all of the circuitry issues of the project through first testing with a multimeter and then soldering any bad connections together.

For specific things I learned I learned how to properly implement a velocity PID controller which is something I didn't know how to do before. I learned a lot about I also learned a lot more about PCB design then I did before in order to make schematics better and also easier to read.Also I learned more on how best to make a PCB board in I also got more practice with implementing FSM into code because I made the whole program run off of a FSM. Also I learned more about how to get subsystems to

communicate through custom digital connections using custom spinlock code and also communicating through a Serial Connection between a Jetson Nano and a Teensy.

Overall I'm happy with how the project turned out since it could traverse a 5x5 maze (while going straight with PID) and go and pick up a cup at the goal cell.

11.2 Future Work

The potential for future improvements for this project are abundant. The CV system could be upgraded to have the camera identify a variety of kitchenware objects instead of limiting it to cups. Computer vision using neural networks would be implemented to identify more specific items. There could also be a modification of the arm grabbing system as well. The arm would have to be able to come up with a variety of different ways to pick up a majority of kitchen/table setting related objects such as plates, candles, and silverware. Additional sensors or another camera could be added to calculate the optimal way to grab each object. More efficient mapping and more complex movement protocols could also be added for faster traversal.

On our end, as the developers of this product, we need to make sure that the materials and parts of our robot are ethically sourced. This means not cutting corners on buying products that are cheaper, but could very well be made unethically by a company that doesn't pay its workers properly. We should also be conscious about how our robot will affect the environment. We should keep in mind what types of materials are sustainable and also what kind of impact our electronic waste could have on the environment.

Our Kitchen Helper Robot can help in larger kitchens like in restaurants and catering services that need multiple cups and utensils set in a specific order everytime. If restaurants find a need for our robot then hotels would soon follow and that would help people out all over the world. Then after this, we could develop a smaller and user-friendly version that could uniquely fit into the everyday life of a working class family and ease their burdens instead of causing them.

11.3 Acknowledgement

- Dr. Roman Chomko - Professor of Senior Design Class
- Merrick Campbell - TA for Senior Design Section 021
- Dr. Tofigh Heidarzadeh - Professor of Technical Communications and Documentation

12. Appendices

12.1 Appendix A: Parts List

Parts	Quantity	Link
Jetson Nano	1	https://amzn.to/3Fl8jiC
sn754410 motor driver	1	https://www.pololu.com/product/24
sonar sensor(HCsr04)	4	https://www.sparkfun.com/products/15569
prototype breadboard	3	https://jlcpcb.com/
wires package	6	https://www.amazon.com/Jumper-Female-Wire-Arduino-raspberry/dp/B01MT530B8
LSM6DS33 imu	1	https://www.pololu.com/product/2736
l7805cv voltage regulator	1	https://www.mouser.com/ProductDetail/STMicroelectronics/L7805CV?qs=9NrABl3fj%2FqplZAHiYUxWg%3D%3D
servos to make robot arm	1	shorturl.at/dAGM5
32 gb Micro SD card	1	https://www.amazon.com/32gb-Micro-Sd-Card/s?k=32gb+Micro+Sd+Card
INIU power bank	1	shorturl.at/cqCEH
Raspberry Pi Camera	1	https://www.raspberrypi.com/products/camera-module-v2/
motor Brackets Pair for 99:1 25D dc pololu motors	1	https://www.pololu.com/product/2676
aluminum mounting hub pair	1	https://www.pololu.com/product/1081
Wheels Pair	2	https://www.pololu.com/product/1420
DIY Cardboard Robot Arm	1	shorturl.at/hjyC1
KeyeStudio 4 dof arm	1	shorturl.at/dvIN0
pololu arm kit	1	https://www.pololu.com/product/3550
bigger dc motors 25D 99:1	2	https://www.pololu.com/product/3207

12.2 Appendix B: Equipment List

Glue_gun
Multimeter
Wire Solder
Soldering Iron Tip Cleaner
Solder Sucker Pump
Stranded 22Ga Wire (Assorted Colors)

12.3 Appendix C: Software

Software list

PCB Modeling: Eagle
Software Development: Arduino IDE, Google Colaboratory, Python IDE

Program Files

Grant Beatty Code:

https://github.com/gbeat002/Senior-Design
CV -ReduceSizeAndPad.py -simplified.py -IsolateObj.py -six_chars.py -serial_test.py -test2.py

Richard “Dylan” McGee Code:

https://github.com/anorakalot/Senior_Design-/tree/main/senior_design_proto_5
Base Robot Teensy (in Senior_Design_Proto_5 Folder)

senior_design_proto_5.ino
global_values.h
gyro_func.h
misc_functions.h
motor_func.h
sonar_func.h
Treamux_func.h
(tremaux folder)

PCB Design Custom Libraries

Sonar_Sensor.lbr
Test_row.lbr
Motor_bracket_25D.lbr
Motor_encoder_25D.lbr
Arm_Connection.lbr
Custom_bit_comm.lbr

https://github.com/anorakalot/PCB_DESIGN
IMU.lbr
Battery_input.lbr
Voltage_regulator.lbr

Sunil Alexander Code:

https://github.com/Sunil98/Senior-Design

Teensy Arm

-grant&sunil's_servo_code.ino
-meArm_Servotest.ino

Link to Google Drive w/ all other project videos:

https://drive.google.com/drive/folders/1-0Ty6ppuRTMtv3moIdNHyPeYwxIwMtAl?usp=sharing
https://drive.google.com/drive/folders/1HYhf_CaxBAEnVH4NM4gNPAFn-sjZ5ZCw?usp=sharing

13. References

[1] "PJRC". Oct-2021. [Online]. Available: https://www.pjrc.com/teensy/td_download.html. [Accessed: 07-January-2022].

[2] "Jetson Nano Developer Kit," NVIDIA Developer, 14-Apr-2021. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Accessed: 06-Jan-2022].

[3] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," Nature News, 16-Sep-2020. [Online]. Available: https://www.nature.com/articles/s41586-020-2649-2 [Accessed: 06-January-2022].

[4] "Jetson Hacks," 7-Jun-2020. [Online]. Available: https://github.com/JetsonHacksNano/CSI-Camera/blob/master/simple_camera.py. [Accessed: 06-Jan-2022].

[5] "pyserial," 14-May-2019. [Online]. Available: https://github.com/pyserial/pyserial/releases. [Accessed: 06-Jan-2022].

[6] "Sparkfun," 18-Jan-2020. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf. [Accessed: 06-Jan-2022].

[7] "SN754410," 18-March-2020. [Online]. Available: https://www.ti.com/lit/ds/symlink/sn754410.pdf. [Accessed: 06-Jan-2022].

[8] "L78 VR," 21-April-2021. [Online]. Available: https://www.mouser.com/datasheet/2/389/cd00000444-1795274.pdf. [Accessed: 06-Jan-2022].

[9] W. Burger and M. J. Burge, Digital Image Processing: An Algorithmic Introduction, 2nd ed. London: Springer-Verlag, 2016.