Robot Project

ECE 4898

Senior Design Final Report

Kyle Smith

Justin Scharer

Donovan Cruz

Mark Bowen

December 13, 2023

GitHub Repository:

<https://github.com/markbowen-uccs/RobotKit_SeniorDesign>

[Executive Summary 3](#_Toc153394342)

[Objective 3](#_Toc153394343)

[Project Requirements 4](#_Toc153394344)

[Robot Kit Requirements 4](#_Toc153394345)

[Sumo Bot Requirements 4](#_Toc153394346)

[Mechanical Requirements 4](#_Toc153394347)

[Software Requirements 4](#_Toc153394348)

[Conceptual Design 5](#_Toc153394349)

[Components 5](#_Toc153394350)

[Chassis Design 7](#_Toc153394351)

[Design 1 7](#_Toc153394352)

[Design 2 8](#_Toc153394353)

[Design 3 8](#_Toc153394354)

[Design 4 9](#_Toc153394355)

[Accessories Design 10](#_Toc153394356)

[Controller Case Design 12](#_Toc153394357)

[Controller PCB 13](#_Toc153394358)

[Controller Design 15](#_Toc153394359)

[Wireless Communication 16](#_Toc153394360)

[Final Design 16](#_Toc153394361)

[Robot Kit 16](#_Toc153394362)

[Components 16](#_Toc153394363)

[Chassis 17](#_Toc153394364)

[Robot 18](#_Toc153394365)

[Controller 18](#_Toc153394366)

[Components 18](#_Toc153394367)

[Design 19](#_Toc153394368)

[Software 20](#_Toc153394369)

[Appendices 23](#_Toc153394370)

# Executive Summary

The UCCS electrical engineering and computer engineering degrees have a variety of classes students must take to graduate, one of which is the Introduction to Robotics class. The Introduction to Robotics class introduces students to primary skills within the respective fields of research such as circuit building, coding and troubleshooting. Famously this class has been hard for all the wrong reasons. The SparkFun Inventor’s Kit was very inconsistent across multiple components of the kit, such as the motors, ultrasonic sensor, and the PWM capabilities.

Leslie Tekamp (referred to as the customer or sponsor) enlisted the help of the Senior Design Team to help develop a more reliable kit for the students to use during class, with the prospect of getting the kit sourced out and mass produced on a larger scale. Tekamp also wanted the opportunity to take advantage of the new kit to be able to take it to STEM events as a representative of the electrical engineering and computer engineering departments. In addition to being a lab kit, the students would also be able to make a “Sumo Bot” program that could serve as the demonstration piece at the STEM events. This gave the team the opportunity to develop a wireless remote control to control the robots.

The Senior Design Team has developed a cost-effective and energy-efficient robot with both projects/goals in mind. One of the customer's requests was to change the main hardware component from an expensive Arduino based board to a cheaper more efficient raspberry pi model. The problem with using a raspberry pi would be the inability to use the Arduino coding environment, which is very beginner friendly. Taking this into consideration, the team used the Raspberry Pi Pico W microcontroller board to develop both the lab kit and the controller, which had just received compatibility with the Arduino environment.

# Objective

The primary objective of this project was to redesign and manufacture a new robotics kit for use in the Introduction to Robotics class. The new kit is intended to fix the shortcomings of the old kit when it came to ease of use and reliability. Within the new kit the customer requested that a Raspberry Pi Pico be used, new DC motors with motor encoders are chosen, and the microcontroller should be Arduino IDE compatible.

The secondary objective of this project was to design the robot with the intention of making a “Sumo Bot” demonstration for STEM outreach events. The plan was to have the robot kit have an onboard wireless connection that would talk with a developed controller. The controller would either be tilt or joystick activated and would drive the robot according to the direction inputted by the user. The body of the robot would also be capable of protecting the microcontroller as another robot crashed into it, attempting to push it out of the ring.

# Project Requirements

The project requirements were derived from previous class structure and discussions with the customer about how to improve upon the old kit, and the new expectations. Major robot kit requirements are ROBOT 1-7, which are what helped shape the outcome of some of the design choices. The “Sumo Bot” requirements were derived the same way, as well as being reliant on some of the robot kit requirements to make possible. From the requirements discussed with the customer we were able to generate requirements for both the hardware and software sides of the project, as both would have their own requirements to make the system work as intended.

## Robot Kit Requirements

ROBOT-1: The Microcontroller must be Arduino/C code programmable.

ROBOT-2: The robot kil shall cost less than $100.

ROBOT-3: The robot kit shall include all components from the previous kit.

ROBOT-4: New motors should be selected with a motor encoded built in.

ROBOT-5: A new chassis shall be designed to house the robot kit components.

ROBOT-6: The new chassis should be durable enough for new lab requirements.

ROBOT-7: A external power source should be able to power the kit components.

## Sumo Bot Requirements

SUMO-1: The robot kit shall contain the necessary parts to complete the “Sumo Bot” assignment.

SUMO-2: There should be a controller that can wirelessly connect to the robot kit.

SUMO-3: The controller should be safe for children.

SUMO-4: The controller should be tilt activated.

SUMO-5: The controller should be durable enough for kids.

SUMO-6: The controller should have an adjustable sensitivity.

SUMO-7: The controller should be ergonomic.

## Mechanical Requirements

MECH-1: The robot kit power supply should be able to provide 5V DC power.

MECH-2: The kit components should be able to be powered from the microcontroller, take 3.3V DC

MECH-3: The wheels should be pliable enough to keep the robot from slipping.

## Software Requirements

SOFT-1: Both the controller and robot shall utilize the same method of communication.

SOFT-2: The controller and robot should be easily identifiable for ease of connections.

SOFT-3: The controller should have “dead-zones” to keep robot from moving accidentally.

SOFT-4: The robot should stop during “Sumo Bot” when a certain color tape is seen.

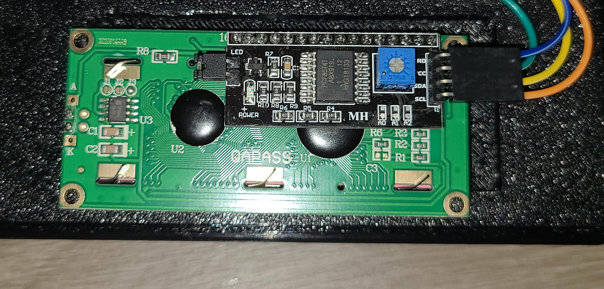
# Conceptual Design

## Components

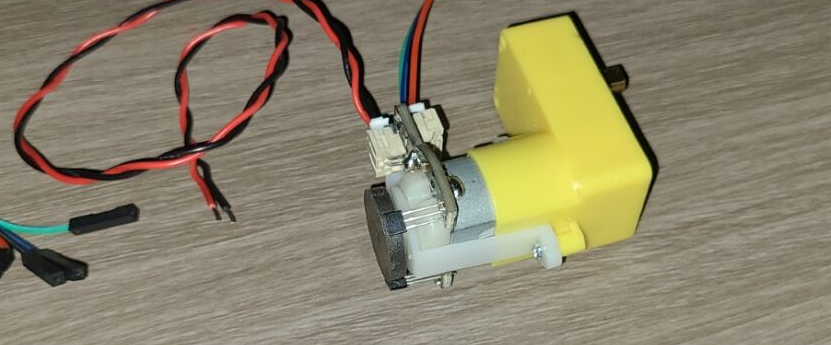
Part of this project was to choose a new microcontroller and implement the previous labs, as the old Arduino board was too expensive to meet our budget requirement. We looked at many microcontrollers such as the Teensy® 4.1 Development Board, SparkFun Pro Micro, and the Raspberry Pi Pico. The Teensy development board has plenty of pins for all the lab components but the board itself runs at 600MHz and costs about 31$ per board. The SparkFun Pro Micro only has about 15 usable pins, which is not enough for the labs to be doable, and still costs about 20$. The Raspberry Pi Pico has about 30 usable pins and only costs about 5$, for this reason we decided to go with the Pico microcontroller. This board takes in anywhere from 2.8-5.5V DC and can output 3.3V DC. So, when deciding on new parts, we had to take the 3.3V output into consideration.

As discussed with the sponsor, there were some upgrades desired within the new kit as well as new parts being needed to fit within the power constraints of the new microcontroller. The Raspberry Pi Pico microcontroller is capable of outputting 3.3V DC so it makes components from the old kit useless. These old components are the ultrasonic sensor, LCD display, motors and the components of the controller would have the same constraints, the joystick and the accelerometer gyroscope chip would all have to operate off 3.3V DC. We also found that using a 16 pin LCD was going to be a problem as it would take up close to half the available pins that the Pico microcontroller provides, so we would also have to go to a four-pin serial LCD to compensate for this problem. One thing that had to be considered is the addressing for the serial LCD as it would run off I2 C addressing circuit built into the Pico. While conducting research for this we found that if we use the default set pins for the I2 C address of the Pico, which is GPIO pins 4 and 5, the LCD will perform as expected.

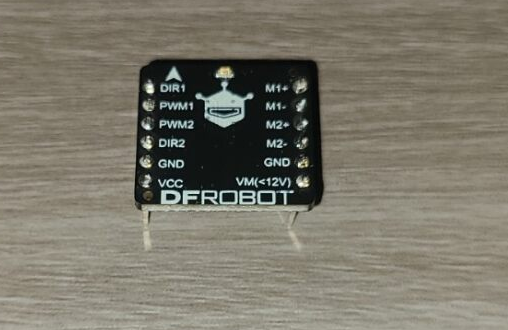




The motors were the other high value component we had to change out on the robot kit, as previously they were running on 5V DC with some current step up from a motor driver. While the old motor driver allowed in 2.3V-5.5V, we would still need to change the motor input voltage due to the change in amperage that would occur. The other design part we needed to consider was a motor encoder. This additional component to the motor would count revolutions per minute and was planned to be used by the consumer at a later date. While finding motors that operated off 3.3V DC was easy, we could not find any motor and motor encoder combo that worked solely from 3.3V. We had to make a comprise and decided on a motor that would operate with 3.3V and an onboard encoder that would operate from 5V DC from our selected power source.



As noted before, the old motor driver worked without input voltage of 3.3V, but there were still inherit problems with the driver. One of the main problems is that the driver would end up shorting out the entire system, potentially harming the microcontroller and everything attached to it. On the old motor driver there was a “Stand By” pin that kept channels open, or powered, even while not in use. The motors would make a whining noise due to this. The shorting problem would come when the motors switched directions too fast, causing the current to flip directions too quickly, which would make the motor controller short, then shorting the microcontroller. The new motor controller we decided to use in this project did not have that “Stand By” pin among many other useless pins, so we theorized that this would solve the shorting problem. Unfortunately, the motor controller got stuck in transit, not allowing us enough time to implement it into the final design.



## Chassis Design

### Design 1

The chassis within the old Sparkfun kit had some flaws that prevented the rest of the kit from working to their full potential. The DC motors were to be connected with double sided tape or Velcro tape, which caused the motors to sag and twist. This caused the robot to drive incorrectly, whether it was turning when trying to drive forward, or the motors not having enough traction to actually drive the robot forward. There was also a problem with the drag of the chassis, as there was no third wheel or bearing to keep the rear end of the robot from dragging on the table, causing the robot's' inconsistencies to become even worse. One of the biggest design motivations was to find a way to secure the motors to the chassis better than previously done, while still being able to house the microcontroller and breadboard. The original goal with this was to avoid using external hardware, so we relied on cleaver CAD work to develop a housing and pinch system to hold the motors in place.

Figure


This was the first draft through of the design, taking in input from the customer. My idea was that the walls would house the motor, keeping it from twisting while the pegs standing would pinch and hold the motor in place, without the need to tape or glue the motors to the chassis. This first draft was a failure as the pegs were improperly place in relation to the motor, the top peg was about ½ and inch off from touching the motor.

### Design 2

Due to the awkward design of the motor, having a 90° angle turn in them for the motor encoder, it took a couple of attempts to get the poles to touch the motor, but to get the pegs to hold the motor was much more challenging. Going through an external source for 3D printing, we quickly ran into the problem of thresholds. The printer utilized a 0.4mm (about 0.02 in) extruder nozzle, which could not achieve the desired 0.2mm (about 0.01 in) threshold that would be required to hold the motor in place without it moving or dropping out of the chassis. There was also the problem of the pegs being very fragile. Because the pegs were free standing with nothing supporting them, they were very easy to break off. Even after removing the critical breaking point at the 90° intersection with the base of the chassis with a bevel, the pegs were still very prone to break off. Making the pegs thicker, or making them have a larger area and perimeter, would have worked, but would have caused problems in manufacturing the chassis. This would have caused warping problems in the print, as the pegs would not have time to cool off between layers and would still be fragile given enough force. Ultimately this idea was scrapped due to too many constraints and problems it would cause for future use.

A grey rectangular object with holes

Description automatically generated

### Design 3

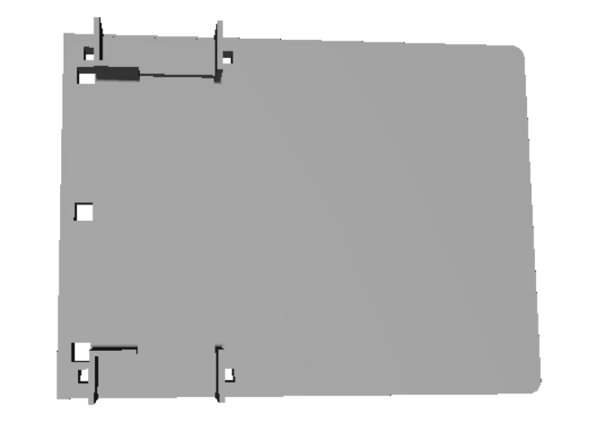
After discussing alternatives with the customer, we eventually decided to allow external hardware to be included in the lab kit. The next step was to decide on what hardware to use since double sided tape and Velcro did not work well in the past. Zip ties allowed for ease of use, easy integration, and enough strength to keep things rigged and tight. There were a few designs considered for implementing the zip ties that anchored the motors in two places, one against the walls and one where the motors bend. This design raised the motors from the chassis even further to be able to run zip ties under the chassis, keeping them hidden from the above view. To keep the bend of the motors secured, we designed nodules that extruded up that would allow zip ties to run through and secure the motor. Unfortunately, there were problems printing such overhangs and housings, as the 3d printer was not interfacing correctly with supports and was not calibrated to print without them, so overhangs became a nightmare, and resulted in multiple failures.

A grey rectangular object with rectangular objects

Description automatically generated

### Design 4

At this stage the design was feeling much more complicated and problematic than anticipated. This is when we took a step back and re-evaluated what was needed to secure the motors to the chassis. While testing the labs for the robot we found that one zip tie around the entire chassis did a very good job holding the motors in place, while the housing walls kept the motor from twisting. In this final iteration of the design, we got rid of the complex over hangs and extrusions and opted for holes that go through the entire chassis that were big enough to run ¼ inch thick zip ties through. These holes were placed next to the housing walls and allowed for a tight fit around the motors.



## Accessories Design

During the lab there are a few modifications to be made in order for some of the labs to function properly. For any of the driving labs, there needs to be some kind of pivot wheel attached to the back of the robot to keep the back of the robot from dragging, allowing the motors to operate properly. In the kit prior, the solution was simply a binder clip clipped onto the end of the robot, so it was sliding on metal in a smaller place rather than plastic over a larger space. Our solution was to glue a caster bearing to the back of the robot, allowing it to smoothly drive without friction dragging the motors down.



Any of the labs including the light sensor needed a way to reduce the amount of external light the photocell was receiving. The use of a milk shake straw worked very well in the old kit, so we decided to not to change that, but we needed a way to extend the light sensor away from the chassis in order to do a line following lab. In the past this was done with old pencils, pens, chopsticks, and even Legos, which all worked fine but all failed at one point or another. The solution was to develop a dedicated “pole arm” that would extend the photocell a fixed length from the chassis and allow the milkshake straw and photosensor to be held a certain distance from the table. We made this a modular piece, as in something that is not glued to the chassis and something that can be placed in various positions depending on where the photocell is needed in that instance.

A grey rectangular object with two sides

Description automatically generated

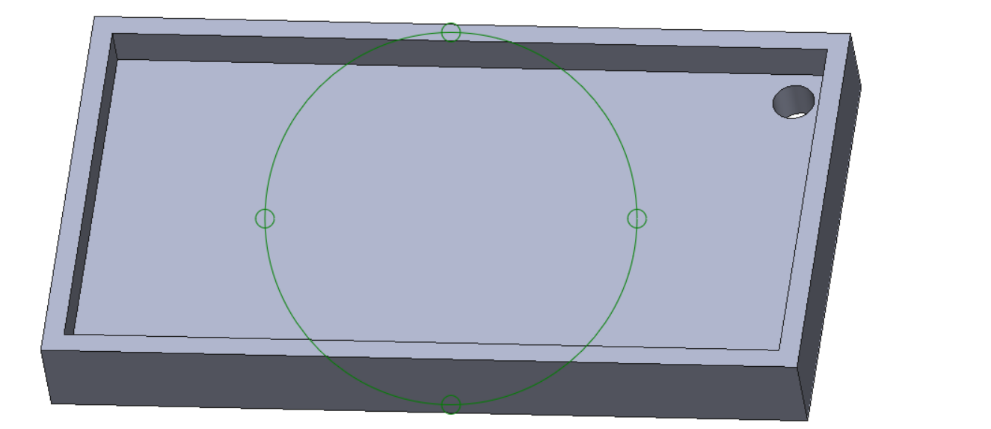
The last accessory designed was for the “Sumo Bot” STEM outreach requirement. To keep the robot and all its components safter, we designed a bumper that would encase the components sticking out from the robot, namely the photocell and the wheels. This was also designed like the pole arm, where it was meant to fit within the modular holes in the front of the robot, leaving a space open for the photocell. There were a few design flaws upon initial printing and fitting. There were a few designs that did not span the entire bumper, and even the final part did not span far enough from the robot to cover the wheels. These problems are simply solved by adding a few extra inches on each side of the bumper to allow it to cover the wheels.

A grey rectangular object with a shadow

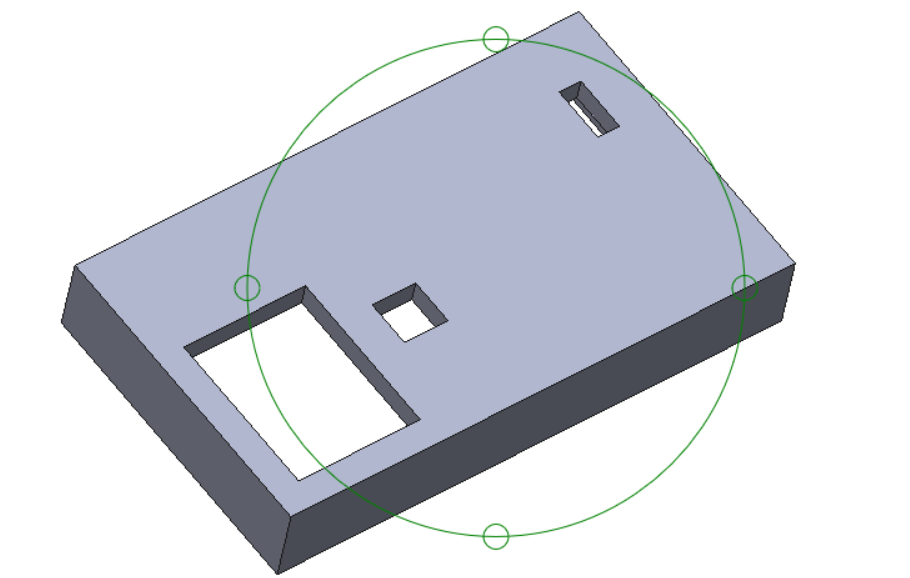
Description automatically generated

## Controller Case Design

One of the requirements for the “Sumo Bot” portion of the project was to design a controller case that would both protect the hardware of the controller and be ergonomic in its feel. As the controller is to be handled by children of all ages, it must be strong enough to withstand any problems that may come alongside that. We were also tasked with making the controller ergonomic, or to make it feel good in the hands of the user. We decided to model the controller off old Nintendo NES controllers, compact and square. This design allows for simple components to extrude from the controller such as the joystick and potentiometer to be accessible. The controller was made up of two different parts, a bottom housing and top lid. The bottom housing was designed to hold the controller PCB in place, allowing a power source to be wired in, and for everything to be easily accessible.



The top lid was meant to cover the PCB while still allowing the components to show through. The main components that needed to show through were the joystick, potentiometer and the switch that changed the control system. The potentiometer and switch were two things that were not able to physically stick though the top lid, instead there was a window designed so that a small tool could interact with the components. The joystick was the most important component that needed to be interacted with, so a window was designed big enough for the joystick to stick through but also allow for full range of motion.



Unfortunately, there wasn’t an opportunity to get the controller case printed out. Troubleshooting the “Sumo Bot” functionality took quite a bit of time, modifications were made to the controller and the PCB it resides on. The case fits the size PCB that was printed out, but the positioning of elements as well as the wiring kept changing. This postponed the printing of the case.

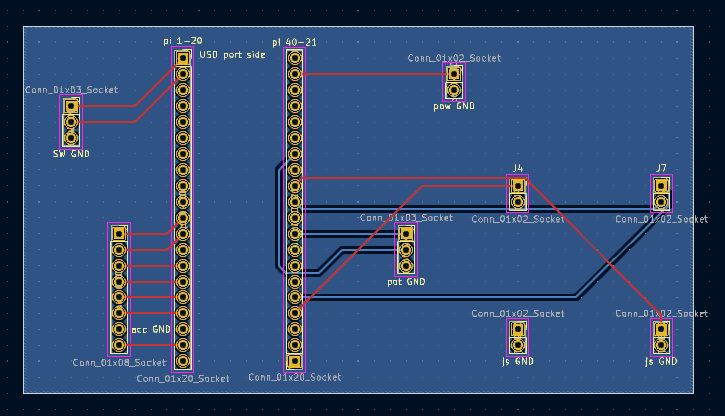
## Controller PCB

The design for the PCB went through a few iterations. In the PCB Figure 2 and PCB Figure 4, the design layout of the manufactured and final is shown. The PCB manufactured has two errors. The potentiometer was not connected. It should be connected to the ground. In the next iteration, this was fixed by reworking the traces to make it easier to be connected to ground. The third design had to include a diode because the battery pack put out too high a voltage for the raspberry pi. The fourth design included a different potentiometer with a different footprint. After reviewing the design, the problem could have been resolved using a three-element battery pack instead of the four-element battery pack that was used in testing. During testing all the power pins had to be connected to pin 36. This was rectified in the seventh iteration of the design. In the eighth design a 3.3V power plane was implemented.

A computer screen shot of a diagram

Description automatically generated

PCB Figure 1– Manufactured PCB circuit diagram



PCB Figure 2– Manufactured PCB layout

A computer screen shot of a computer program

Description automatically generated

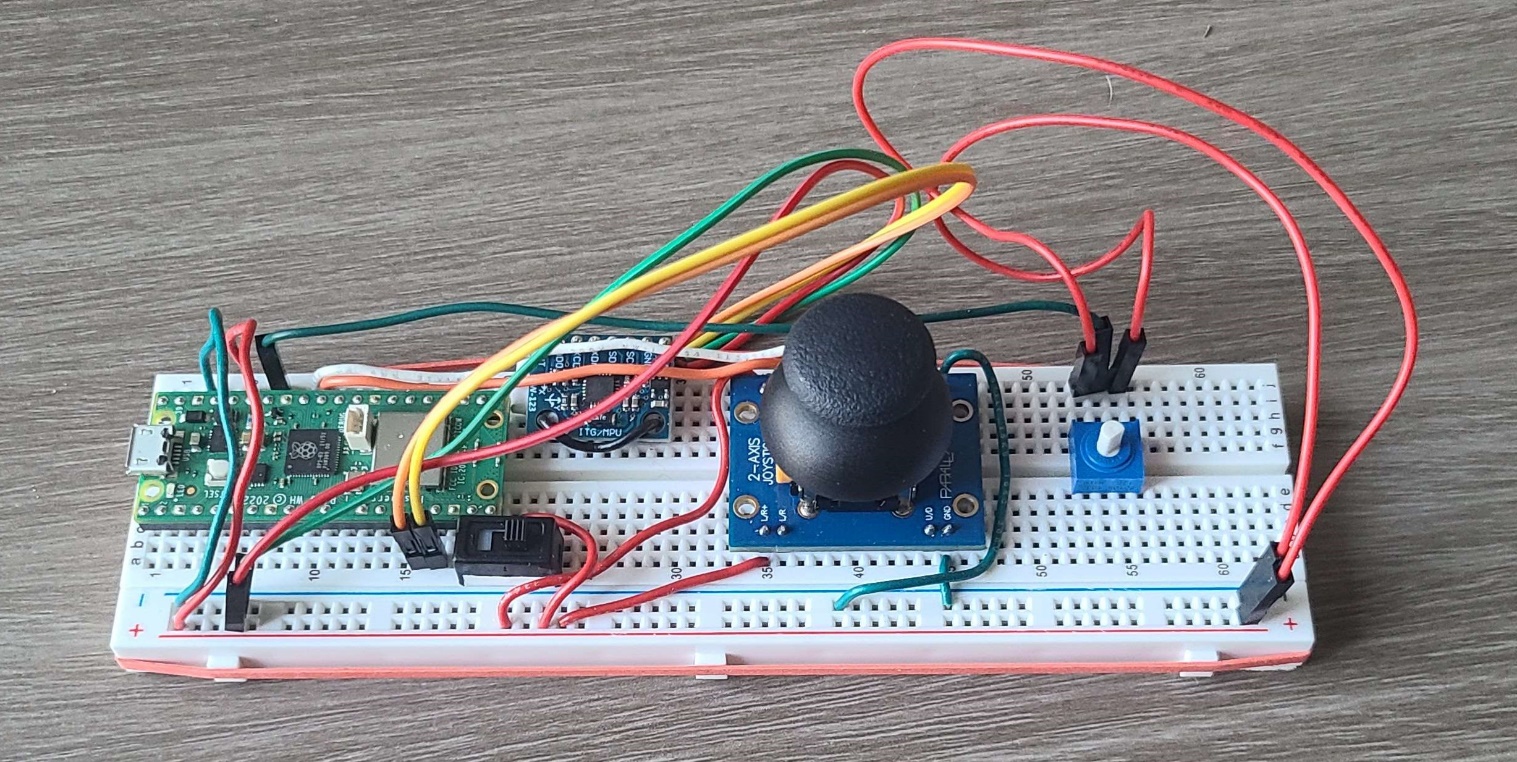
PCB Figure 3– Final PCB circuit diagram

A computer screen shot of a circuit board

Description automatically generated

PCB Figure 4– Final PCB layout

## Controller Design



The controller consists of the chosen microcontroller, a Raspberry Pi Pico W, a two-axis joystick, a potentiometer, a switch, and an MPU-6050 3-axis Accelerometer Gyroscope. Starting with the simple components, the switch is to change input modes between the MPU-6050 and the joystick and the potentiometer is to adjust the sensitivity of the MPU-6050. The two-axis joystick was chosen based on the shape of the stick and compactness. The MPU-6050 was chosen due to how popular it is and how much documentation it has with Arduino. Lastly the Pico W was chosen for the cheap price, features, and wireless capabilities. Shown above is the prototype, however, the final PCB was designed based off this. Once all parts are ordered and printed, the final controller can be assembled.

## Wireless Communication

According to our requirements, it is necessary to get a controller and a robot to communicate wirelessly to demo sumo bot for outreach events. At first, communication using Bluetooth 5.2 was attempted because the Raspberry Pi Pico W has a wireless chip that possesses functionality for Bluetooth and Wi-Fi. The Pico W can connect to other devices with Bluetooth, such as a controller. To test this, a Dual Shock 4 controller was used to connect to a Raspberry Pi Pico W using a GitHub repository. Some code was tested, and the controller gave data on its features. However, it has too many features and we were tasked to design a controller. Since a Pico W is on both the robot and controller, we investigated methods that the two boards could communicate with each other wirelessly. The Bluetooth SDK was released back in July of 2023 but so far mostly has libraries made in Python by users. Some Arduino/C libraries exist but none cover the functions we want, i.e. transmitting values across two Pi’s using Bluetooth. While the libraries exist for this exist, they are not supported by C nor Arduino as of this report.

As shown later in this report, we decided on a server-client model where our bot is the server, and the controller is the client which sends input data which the server receives as values for the motors. This method was chosen because there was enough documentation on it, and it possesses an advantage of range.

# Final Design

## Robot Kit

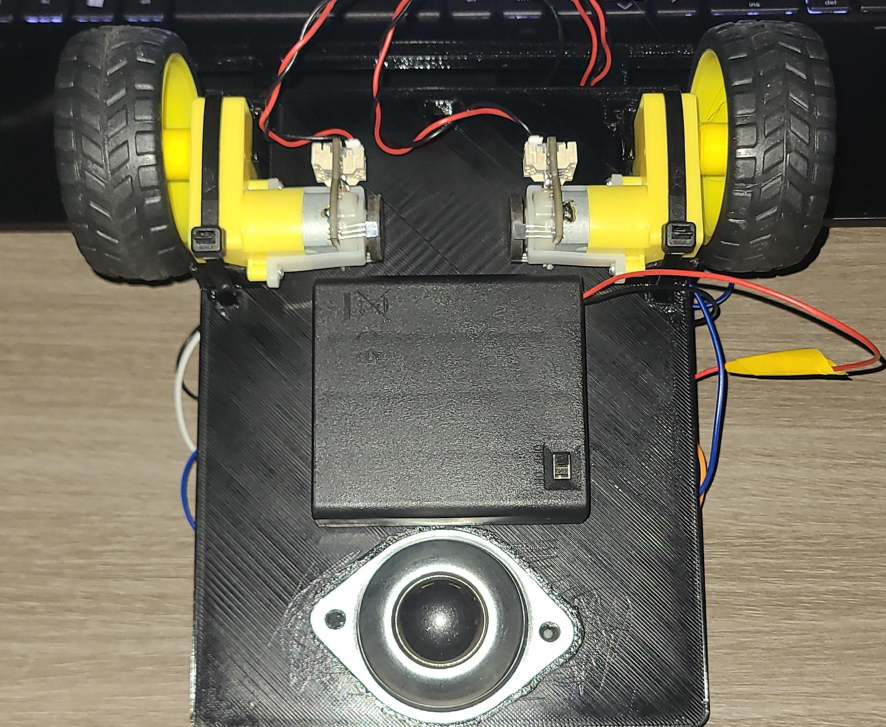
### Components

As per our requirements, ROBOT-2, the cost of the robot kit was to land at 100$ or less. As we were totaling up the robot, we had to add 20$ onto the kit price for distribution and manufacturing costs. Even with this our kit came 15$ under budget, allowing for upgrades or other inclusions later on.

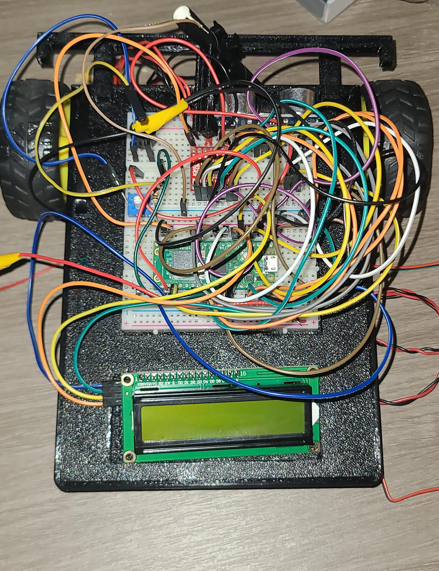
|  |  |  |
| --- | --- | --- |
| part | number needed per kit | cost per kit |
| Raspberry Pi Pico With Wireless & Pre-soldered Headers | 1 | 7 |
| motor | 2 | 14.8 |
| tire | 2 | 1.98 |
| Lcd | 1 | 4.11 |
| Ultrasonic | 1 | 3.95 |
| Breadboard 600 | 1 | 9 |
| Programmable led | 1 | 0.3 |
| 3 pin switch | 1 | 0.56 |
| 10k Potentiometer | 1 | 1.25 |
| Servo | 1 | 3.95 |
| buzzer 3v | 1 | 0.63 |
| tmp 36 | 1 | 2.3 |
| Battery Box 4 AA | 1 | 2.95 |
| 10k Resistors | 10 | 1 |
| 330 Resistors | 10 | 1 |
| Photoresistor | 1 | 0.23 |
| Buttons | 4 | 0.55 |
| led blue | 5 | 0.2 |
| led red | 5 | 0.1 |
| led yellow | 5 | 0.1 |
| led green | 5 | 0.1 |
| Jumper Wires mm | 40 | 2.95 |
| diode | 1 | 0.21 |
| motor driver | 1 | 4.5 |
| Caster Wheel | 1 | 1.88 |
| Zip Ties | 4 | 0.15 |
| Manufacturing |  | 20 |
|  | Total: | 85.75 |

Chassis

As discussed in the conceptual design, the final iteration utilizes zip ties to secure the motors to the chassis, while the walls of the motor housing keep the motors from twisting. We were able to secure the motors with an 11 inch long ¼ thick zip tie. In addition to the zip ties, we mounted a ball bearing caster wheel to the back of the chassis, to keep it from dragging and causing the motors to slip.



Robot



## Controller

### Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Raspberry Pi Pico With Wireless & Pre-soldered Headers | <https://vilros.com/products/raspberry-pi-pico-wh-with-wireless-presoldered-headers-3-pin-jtag-connector?src=raspberrypi> | 1 | 1 | 7 | 7 |
| 2-Axis Joystick | <https://www.parallax.com/product/2-axis-joystick/> | 1 | 1 | 7.95 | 7.95 |
| Accelerometer | <https://www.amazon.com/dp/B00LP25V1A?_encoding=UTF8&psc=1&ref_=cm_sw_r_cp_ud_dp_WXATJCQHHJABKZ6905WV> | 1 | 3 | 9.99 | 3.33 |
| 10k pot | <https://www.digikey.com/en/products/detail/adafruit-industries-llc/356/6827130> | 1 | 1 | 1.25 | 1.25 |
| Battery Box 3 AA | <https://www.mpja.com/Battery-Box-with-Cover-Switch-Holds-3-AA-Cells/productinfo/36025+BH/> | 1 | 1 | 2.25 | 2.25 |
| PCB |  | 1 | 3 | 48.3 | 16.1 |
|  |  |  |  |  | 37.88 |

The controller for the advanced Robot Kit integrates a Raspberry Pi Pico W with wireless connectivity, including a Parallax 2-Axis Joystick, MPU-6050 3-Axis Accelerometer, and a 10k potentiometer for precise control. A 3-pin slide switch facilitates toggling between control types, and power is managed by a Battery Box with 3 AA batteries. This design satisfies the SUMO-1 requirement and was verified at the Demo.

### Design

**Programming and Control Logic:** Programmed in Arduino, the controller seamlessly interfaces with the Robot Kit via wireless connectivity, adapting to various Wi-Fi networks. It features a switch for joystick or accelerometer/gyro control and a potentiometer for real-time sensitivity adjustments.

**Data Stability and User Customization:** Data normalization ensures seamless continuity with the robot regardless of the control type being used. Potentiometer limits prevent voltage fluctuations, contributing to system robustness. Customization options within the Arduino code include an adjustable dead zone and a pre-set max angle parameter for optimal control precision.

**Wireless Communication Setup:** The Arduino program facilitates Wi-Fi network connection between the Robot (server) and the controller (client). The controller initiates communication, continuously transmitting data to the Robot, which interprets and translates user inputs into robotic movements. Satisfying the SUMO-2 requirement as verified during the Demo.

**Safety and Durability Considerations:** SUMO-3 and SUMO-5 requirements were not satisfied at the Demo. However, an included CAD design addresses safety by housing the PCB securely. Although the controller's overall durability requirement for kids is not currently satisfied, the CAD design mitigates potential hazards, ensuring a safer educational experience.

**Tilt Activation and Ergonomics:** Tilt activation, powered by the MPU-6050, adds an interactive element. Satisfying the SUMO-4 requirement as verified during the Demo. Although the ergonomic requirement described in SUMO-7 was not met at the Demo. The CAD design addresses comfort and ergonomic concerns.

**Adjustable Sensitivity and User Customization:** The controller design meets the SUMO-6 requirement with a potentiometer for adjustable sensitivity, allowing users to tailor responsiveness based on thresholds or age groups, as verified during the Demo.

### Software

**Software Design for Robot Vehicle (Acting as Wi-Fi Server)**

* **Introduction:** This section outlines the design and functionality of the PicoW2 robot control program, which serves as a Wi-Fi server for remote control. The program is developed using the Arduino framework and facilitates robot movement based on input from a client. It covers the design aspects, dependencies, configurations, and initialization procedures of the robot control program.
* **Dependencies:** The program relies on the following libraries:
  + WiFi.h: For handling Wi-Fi connectivity.
  + LiquidCrystal\_I2C.h: For interfacing with the LCD screen.
  + Wire.h: For I2C communication.
* **Configuration**
  + ai1, ai2, bi1, bi2: Pins for motor control.
  + pwma, pwmb: PWM control pins for motor speed.
  + lightSense: Pin for a light sensor.
  + STASSID, STAPSK: Wifi network SSID(or IP) and password
  + port: Port number for the server.
* **Server Initialization**
  + Initializes a WiFi server on the specified port.
  + Creates a structure (SentData) to hold data received from the client.
  + Sets a default cooling motor voltage (motorCool).
* **Setup Function**
  + Configures motor control pins, sets analog read resolution, and initializes the LCD.
  + Configures WiFi settings and establishes a connection.
  + Displays connection details on the serial monitor and LCD.
  + Starts the server and displays server IP on the LCD for client use.
* **Main Loop**
  + Resets motor control pins to LOW voltage.
  + Checks for incoming client connections.
  + If a client is connected, enters an infinite loop to receive and process data.
* **Data Reception and Processing**
  + Reads data sent by the client into the currentData structure.
  + Displays received data on the serial monitor and LCD.
  + Controls motor movement based on received data.
* **Movement Control Functions**
  + **Forwards:**
  + Controls forward movement upon receiving a positive Y value.
  + The Y value also controls the overall speed of the motors.
  + Adjusts motor speed to change direction based on the X value using:
    - **Inside Motor Value**
  + **Backwards:**
  + Controls backward movement upon receiving a negative Y value.
  + The Y value also controls the overall speed of the motors.
  + Adjusts motor speed to change direction based on the X value using:
    - **Inside Motor Value**
  + **Spin:**
  + Controls spinning movement upon receiving a zero Y value.
  + Adjusts motor speed and direction based on the sign and value of X.
* **Conclusion** The program successfully establishes a WiFi server, receives input from a client, and controls the robot's movement based on the received data. The movement is determined by the X and Y values, with different functions for forward, backward, and spinning motions. The LCD screen provides status updates, and the serial monitor logs relevant information.
* **Note** The specific functionalities related to the light sensor and an unspecified button are mentioned in the code but commented out. These functionalities can be troubleshooted and integrated into the sumo bot program.

**Software Design for Remote Control (Acts as WiFi Client)**

* **Introduction:** This section provides a detailed overview of the software design for the Arduino program responsible for remote controlling the robot. The program reads sensor data from an MPU6050 sensor, potentiometer, and joystick, establishing a WiFi connection to send the data to the server (Robot Kit). It covers key components, data structures, code structure, and dependencies for the remote-control program.
  + **Dependencies:**
  + WiFi.h: For handling WiFi connectivity.
  + Adafruit\_MPU6050.h: For handling the Accelrometer/gyro
  + Adafruit\_Sensor.h: For handling sensor readings
  + Wire.h: For I2C communication.
* **Key Components**
  + **MPU6050 Sensor** - Reads accelerometer data with custom I2C pins. 2.2.2
  + **Joystick** - Reads analog data from two joystick pins. - Used to control sensor data if activated.
  + **WiFi Connection** - Uses Pico W WiFi module for connecting to a WiFi network.
  + **Potentiometer** - Reads analog data from a potentiometer. - Controls the magnitude of sensor data.
* **Configuration**
  + I2C\_SDA, I2C\_SCL: For MPU-6050 communication
  + STASSID, STAPSK: Wifi network SSID(or IP) and password
  + host, port: server identified by IP address and port. - Sends sensor data to the server.
  + potPin: For potentiometer communication
* **Communicated Data Structure**
  + A structure named SentData stores sensor data, including X and Y values and the potentiometer value to send to the Robot over the wifi network.
* **Setup Function**:
  + **Pin Initialization** - Configures default LED pin and custom I2C pins.
  + **Serial Communication** - Initializes serial communication for debugging.
  + **MPU6050 Initialization** - Initializes the MPU6050 sensor with specific ranges and bandwidth.
  + **WiFi Connection** - Connects to the WiFi network using provided credentials. - Reboots if connection fails.
* **Main Loop:**
  + **Client Initialization** - Initializes a new WiFi client for server communication.
  + **SentData Initialization** - Initializes the currentData struct used for data communication.
  + **Sensor Data Collection** - Reads potentiometer value and checks for joystick activation. - Reads accelerometer data if joystick is not activated. Reads joystick values if joystick is activated.
  + **Data Normalization** - Normalizes pitch and roll values considering dead zones and maximum angles or normalizes joystick values considering just dead zones. - Multiplies normalized values by the potentiometer value and loads the values into the currentData struct.
  + **Data Transmission** - Sends the entire SentData structure named currentData to the server.
* **Conclusion** This section provides an in-depth understanding of the software design for the Arduino program. It covers key components, data structures, code structure, and dependencies. Developers can use this document as a reference for maintenance and future enhancements.

# Appendices