Junior Design Final Documentation & Schematics

The Waymo-ers

Chris Foraste, Miles Houston, Devon Kumar, Jacky Zhao

General Overview:

The goal of this project was to design a robotic car with the ability to navigate a track containing lines of different colors while avoiding collisions using custom-made color and distance sensors. Furthermore, the robotic car was designed to enable communication between itself and another robotic car through the usage of Web Sockets to perform joint track navigation. Finally, an additional feature was implemented to allow users to remotely control the robotic car using hand movements.

Mechanical Documentation:

Design Strategy:

To improve aesthetics, cable management, and adaptability, we designed a modular robot chassis with flexible electronics mounting points. The polycarbonate chassis plate, on which all components are mounted, features a periodic array of mounting holes. These holes were incorporated early in the design process to allow for future flexibility in attaching electronics and breadboards that we might not have been able to anticipate at the time. In addition, we included several larger-diameter holes near the motors and at other strategic locations to route wire bundles through the chassis. These openings significantly improved cable management and helped keep wiring organized and unobtrusive. As the electronics were assembled, we also made an effort to consolidate related components onto dedicated breadboards, often grouping them with their associated sensors. This modular organization further contributed to a cleaner and more maintainable robot.

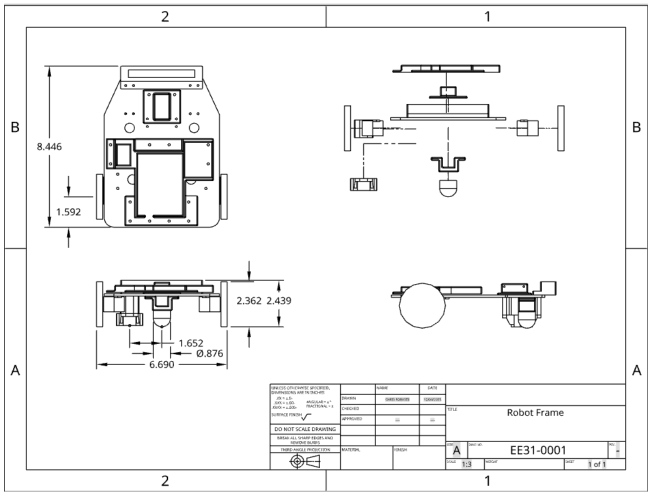


Figure 1. Key Dimensions of Autonomous Vehicle

For the Go-beyond, one of our primary goals was to ensure that the attachment was secure, comfortable, and enabled full access to all the M5's ports. As such, we created a sewable mount, designed to be small enough to fit on the back of the average hand, thereby preventing any mobility issues. The ten separate mounting holes provide a strong and secure attachment that allows the microcontroller to easily move with the glove. Due to the low weight of the microcontroller and the smooth surface of the mount, a simple piece of masking tape is sufficient to mount the microcontroller, even when the controller is held upside down or shaken. Finally, the case was made to be open so the LCD can be seen and all of the buttons on the controller can be accessed normally.

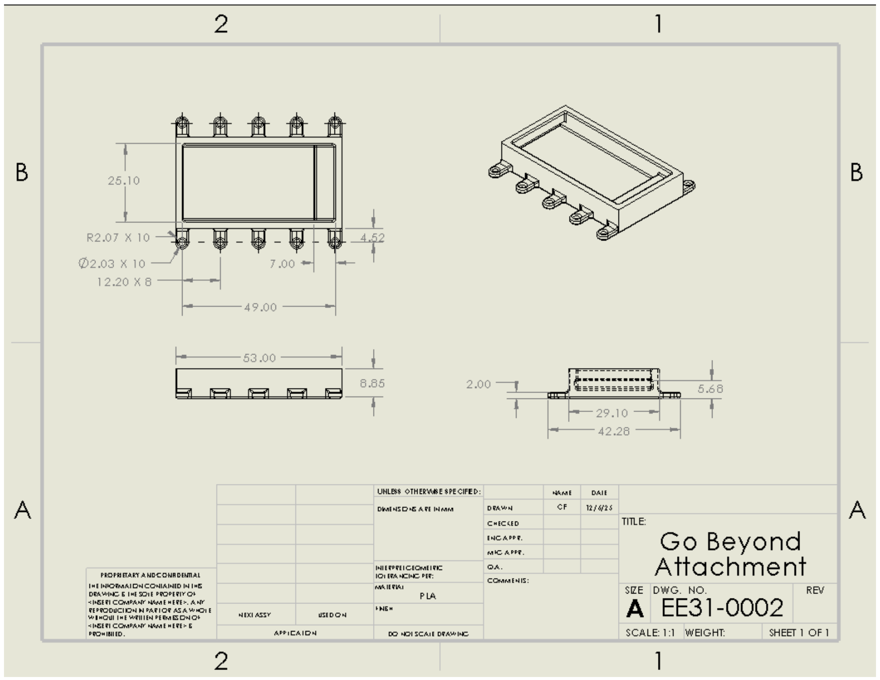


Figure 2. Go Beyond Attachment Schematic

Electrical Documentation:

Distance Sensor:

The distance sensor consists of two separate modules, each consisting of two IR LEDs and a phototransistor. 470 Ohm resistors were used to limit the current through the IR LEDs and 1M resistors were used to bias the phototransistors appropriately. This produces a voltage between 0 and 5 Volts at Vb and Vd, which are fed into pins A1 and A3 on the Arduino board respectively. The only difference between the two modules is their physical distance, with the Vd module being further from the robot than the Vb module. This allows for background light subtraction, with the Vd module detecting the wall and the Vb module providing a stable reference point (see Figure 4).

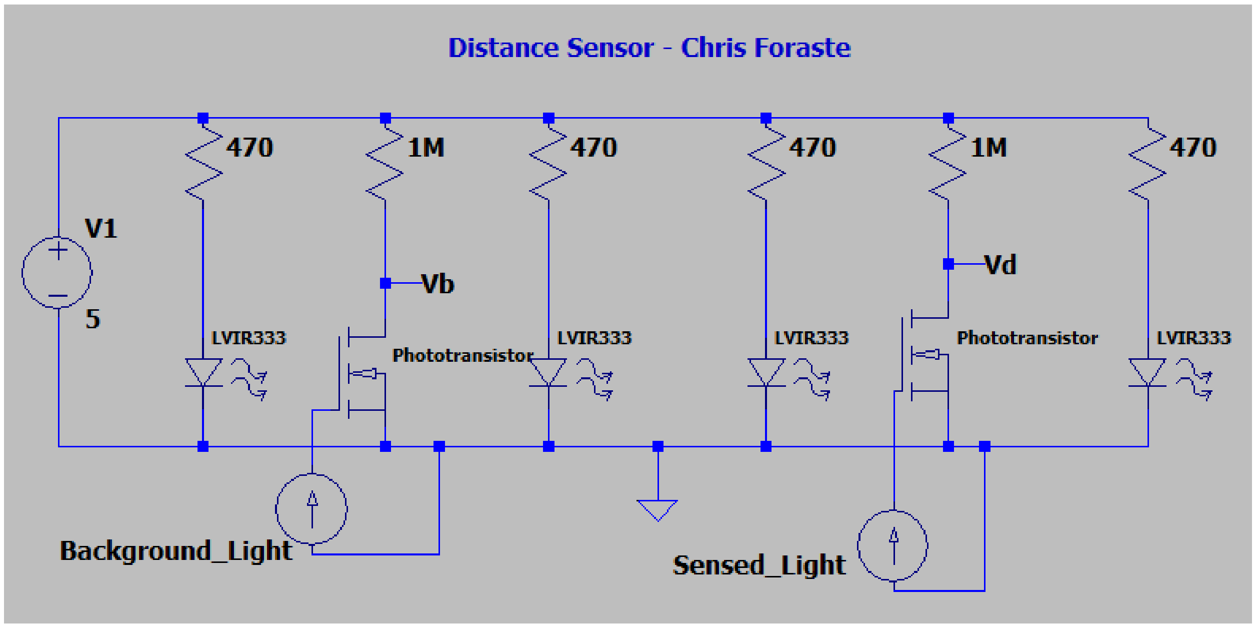


Figure 3. Schematic of the circuit used to design the distance sensor.

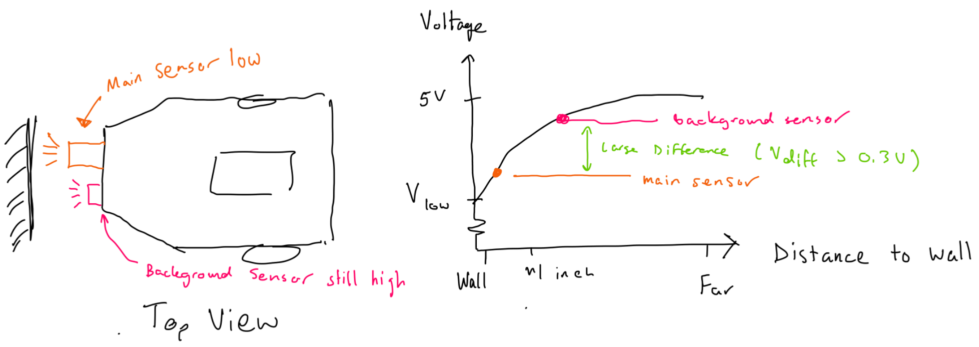


Figure 4. Theory behind the distance sensor.

Color Sensor:

The color sensor consists of two banks of three LEDs, a Jameco photocell, and two NPN transistors used to control power delivery to the LED banks. Each transistor is driven by a digital output pin on the Arduino, allowing the corresponding LED bank to be powered when the digital signal is high. Both LED banks share a common 100 ohm current-limiting resistor. The Jameco photocell is used to measure reflected light intensity and is wired in series with a 240 ohm resistor to form a voltage divider. The resulting voltage is connected to an analog-to-digital converter input on the Arduino.

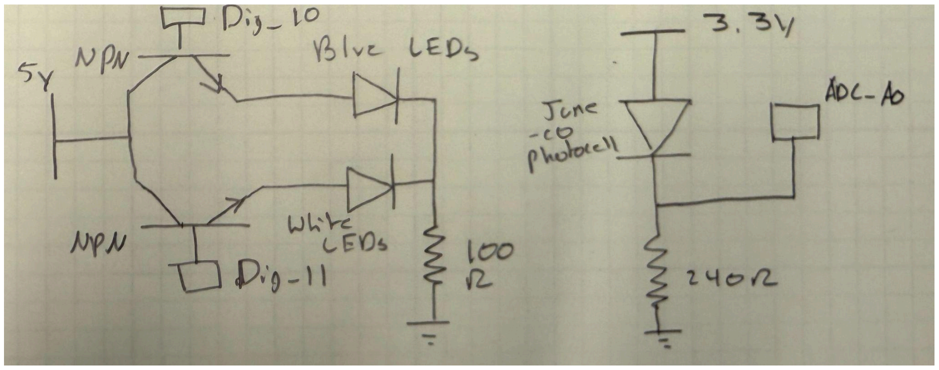


Figure 5. Schematic of the circuit used to design the color sensor.

H-Bridge and Motors:

The H-Bridge and Motors circuit was made using an L293D dual H-bridge motor driver chip. It uses pulse width modulation input into the 1A to 4A input ports to control the motors through the 1Y to 4Y output ports. The speed of the motors are controlled by the duty cycle of the pulse width modulation. The battery and L293D chip share a ground with the Arduino to ensure proper voltage difference readings.

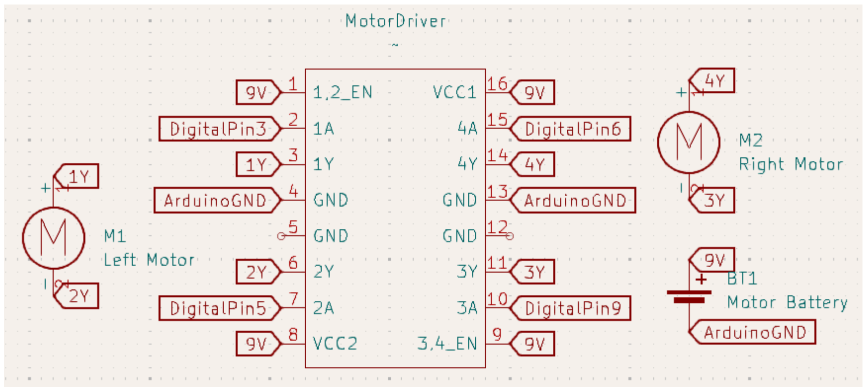


Figure 6. Schematic of the H-Bridge circuit.

Voltage Divider for Voltage Reader:

Due to the 5V limitation of the analog-to-digital converter of the Arduino's pins, it was necessary to step-down the 9V battery to a voltage level that was readable and safe for the Arduino's analog pins. A voltage divider circuit was implemented consisting of an R1 value of 10k Ohm and an equivalent resistance R2 value of around 12.5k Ohm made from 3 resistors wired in series. The voltage of the node between the 10k Ohm resistor and the other resistors was connected to pin A4 of Arduino to convert the voltage using the analog-to-digital converter. This digital voltage was then used in the software to determine the state of the battery. After determining the state of the battery, an LED in the circuit shown in Figure 8 would light up to indicate to the user if the battery needed to be charged. The circuit consists of 3 LEDs wired to different digital pins on the Arduino connected in series with 330 Ohm resistors.

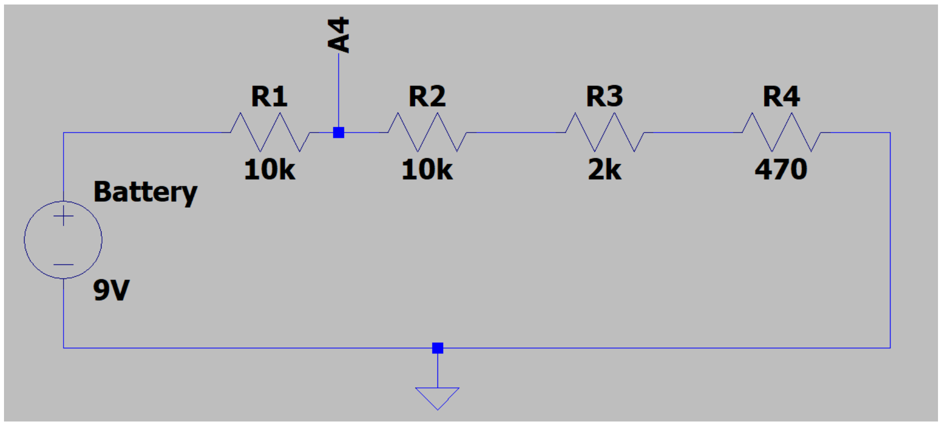


Figure 7. Schematic of the voltage divider circuit.

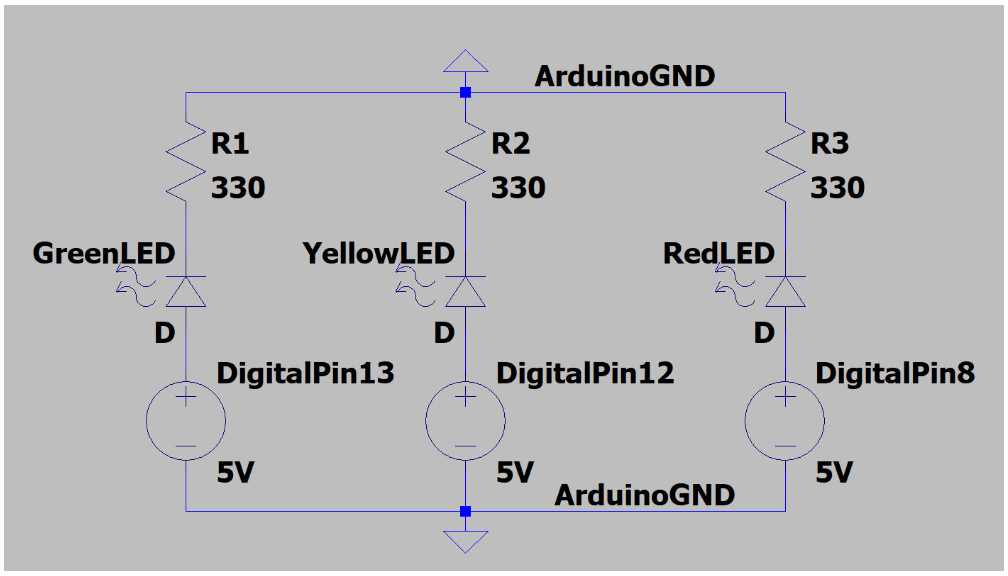


Figure 8. Schematic of the voltage checker indicator circuit.

Schematic of the Entire Robot:

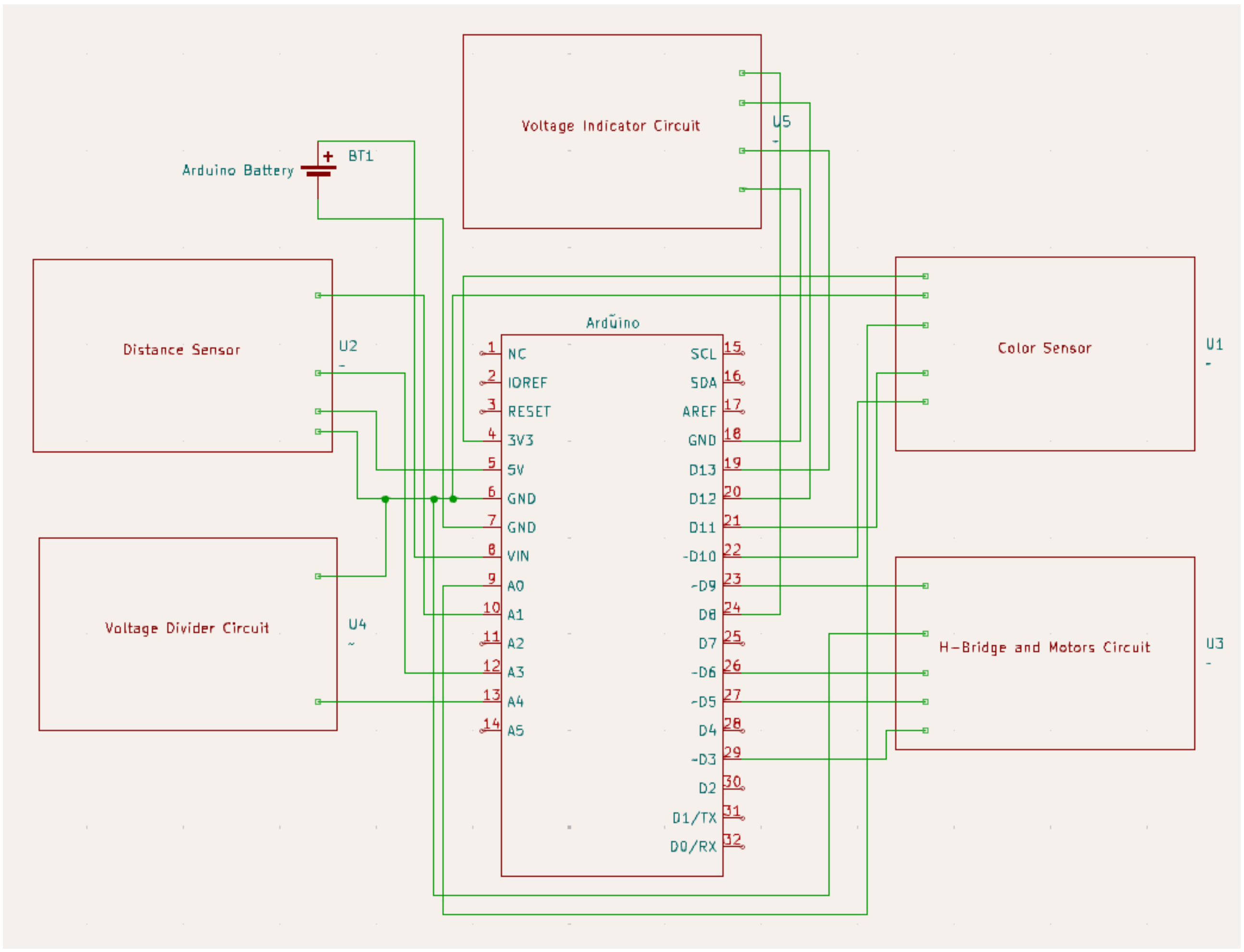


Figure 9. High-level schematic overview of the entire robot showing how all the circuits and components are connected to the Arduino.

Software Documentation:

Main Code Repository Link**:** **https://github.com/devonk05/EE31-Project**

Go-Beyond Code Repository Link**:** **https://github.com/chris-foraste/EE31_M5StickC**

Color Sensor:

The goal of the color-sensing algorithm is to use intensity measurements from a photocell to determine whether the sensor is viewing a track line or the darker background of the board. The algorithm operates in three stages First, the system reads normalized intensity measurements from the photocell. Because the sensor uses two differently colored LEDs, each color measurement consists of two readings– one for each LED. Second, the resulting pair of intensity values is compared against four pre-calibrated measurement pairs, corresponding to the four colors on the track. The algorithm determines the closest match by computing the L2 norm of the difference between the measured values and each pre-calibrated pair. Finally, the identified closest match is stored in a rolling list of the five most recent measurements. A majority vote over these measurements is used to determine the final color output of the algorithm.

Distance Sensor:

The distance sensor code takes a differential measurement between the phototransistor placed further from the robot (the active sensor) and the background light sensor closer to the robot. This is then passed through a five point moving average filter, which dSensor.poll() returns. If the difference is greater than an adjustable threshold in main (0.65 Volts), then the robot knows it has detected a wall.

Drivetrain:

The drivetrain class is built off of the wheel class. The wheel class acts as a software interface between the motor and the arduino. The wheel class contains only pin definitions. The pins are digitally-written so they output PWM to the H-bridge. The drivetrain inherits two wheel objects, which are used in the bot's movement commands such as go, reverse, turn, pivot, etc. The drivetrain manipulates the duty cycles written to the H-bridge in order to produce the desired bot motion. The drivetrain also features distance and time parametrized overloads of the standard movement commands by running for a specific duration met by the argument passed to the function. The drivetrain is used in several classes within the overall bot and serves as a foundational piece of building the other components.

Voltage Reader:

The Voltage Reader class contained two functions that enable the user to monitor the voltage level of the batteries used on the robot. The first function is called read() and it uses the built-in analogRead() function on Arduino to read from pin A4. The voltage is then multiplied by 5 and then divided by 1023. This is because the analogRead() function maps the input voltage to 10 bits, which can represent numbers from 0 to 1023. 0 corresponds to 0 volts and 1023 corresponds to 5 volts. Thus, the function reads the input voltage and converts the 10 bit voltage representation back into a float from 0 V to 5 V. The second function in the class is called voltCheck(). It calls the read() function and uses if-statements to determine whether to turn on or off pin 8, pin 12, and pin 13. Pin 13 turns on the green LED, which signals that the battery level is within acceptable operating range (battery voltage greater than 3.5 V). Pin 12 turns on the yellow LED, which signals the battery level needs to be charged soon (battery voltage between 3.5 V and 3 V). Pin 8 turns on the red LED, which signals the battery is not usable and needs to be charged immediately (battery voltage is below 3 V). The battery voltages used in the if-statements are the voltage readings after the voltage of the battery is stepped-down using the voltage divider circuit. Finally, only one pin is turned on at a time, so the other two pins are turned off.

main.cpp / Main:

The main file of the src folder contains the function definitions for the Web Socket in addition to several of the Arduino state machine functions. The user is able to define which track navigation mode they are in by setting the respective global variable as true and the other 2 false. To clarify, there are three modes: Solo Track Navigation, Partner Navigation as Bot 1, and Partner Navigation as Bot 2.

Main begins by instantiating system objects such as the drivetrain, the line-following, distance sensor, and color sensor. Within setup(), the arduino configures PWM pins, performs the startup protocols for the sensors, and establishes a connection between the arduino and the Web Socket server. Between setup() and loop(), some static state variables are declared to aid in keeping state transitions predictable by preventing repeat state-action on successive loop iterations.

Within loop(), the bot is initialized to an idle or beginning state (contingent upon the navigation mode). When the necessary trigger is asserted, state transitions begin. At the beginning of each loop iteration, messages are read in from the server in order to check for relevant messages sent via the Web Socket from the partner team (if not solo navigating). A large ‘if’ statement then checks the state (an integer), driven by the color sensor, line-following algorithm, and the Web Socket the distance sensor, and dictates drivetrain logic. If the bot is in partner navigation mode, it may also send a message via the Web Socket depending on the specific state.

Go Beyond Main:

After adapting Devon’s Web Sockets code to connect the M5StickC to the Web Sockets server, the controller takes a differential IMU measurement between its orientation and a user set base orientation. Depending on the measurement, it sends a command over Web Sockets to the main robot, which has a receiver that parses the commands into driving functions. Pressing the central button sets the base orientation to the current position (thus also serving as an emergency stop button). Pressing the right side button switches whether the robot is listening to the glove or its track navigation by sending a Web Sockets message. An internal state also tracks the current method of control, and the glove will not send Web Sockets messages in track mode unless it is to turn glove mode on.

Go Beyond IMU:

Because the M5StickC only has an accelerometer and gyrometer, a Kalman filter was implemented to accurately predict the angle of the glove. Each loop cycle, the IMU's orientations are updated through the Kalman Filter and it outputs its current orientation, accounting for taring. Taring is controlled by pressing the central button, which sets the current orientation as the zero point.

Materials and Cost:

Link: Materials and Cost Spreadsheet

The total final cost of the project is around $265 dollars. The main expense of the project was the usage of 3 Arduino Uno MK2 WiFi boards for testing purposes and the parallelization of work, which contributed a cost of around $162 to the project. However, only one Arduino board was used in the final product. Thus, the cost could be reduced to around $158 dollars if only one Arduino was used throughout the research and development part of the project. The tradeoff of this reduction in cost would be longer production time.

Component	Quantity	Cost Per Part	Total Cost
Acrylic Board	1	$7	$7
Arduino Uno MK2 WiFi	3	$53.80	$161.40
Ball Extension	1	$0.10	$0.10
Color Sensor Attachers	4	$0.10	$0.40
Half-Breadboard	2	$1.20	$2.40
IR LEDs	10	$0.15	$1.50
880nm phototransistor NPN	8	$0.68	$5.44
Mini Breadboard	1	$1.00	$1.00
2.5mm Screw	10	$0.20	$2.00
JameCo Photocell	4	$1.49	$5.96
100 Ohm resistor	6	$0.10	$0.60
220 Ohm resistor	3	$0.10	$0.30
1k Ohm resistor	1	$0.10	$0.10
10 Ohm resistor	2	$0.10	$0.20

100k Ohm resistor	1	$0.10	$0.10
10k Ohm resistor	2	$0.10	$0.20
330 Ohm Resistor	4	$0.10	$0.40
3.3k Ohm Resistor	1	$0.10	$0.10
2k Ohm Resistor	1	$0.10	$0.10
470 Ohm Resistor	6	$0.10	$0.60
1 MegaOhm Resistor	4	$0.10	$0.40
10 MegaOhm Resistor	4	$0.10	$0.40
BS170 MOSFET	6	$0.25	$1.50
0.1 uF Capacitor	1	$0.40	$0.40
0.047 uF Capacitor	1	$0.40	$0.40
LEDs	14	$0.20	$2.80
Rechargeable 9V Batteries (2)	1	$2.39	$2.39
Battery Clips (20)	0.05	$8.95	$0.45
Pololu Wheel 60x8mm (pair)	2	$5.71	$11.42
E Switch	1	$0.79	$0.79
Pololu Mini Plastic Gearmotor Bracket Pair - Tall (pair)	2	$5.95	$11.90
Ball Caster with 3/4" plastic ball	1	$5.49	$5.49
L298DNE Motor Driver	3	$4.61	$13.83
Green Protoboard	2	$0.40	$0.80
TC4584BP Hex Inverter	1	$0.41	$0.41
Pushbutton Switches	3	$0.10	$0.30
Distance Sensor Holder	1	$0.15	$0.15
3D Printed Arduino Attachment	1	$0.30	$0.30
M5StickCPlus	1	$19.99	$19.99
Total Cost			$264.0175

Table 1. This table contains all the components, the quantities used and the individual costs per part, as well as the total calculated cost of the project.

Picture of Robot:

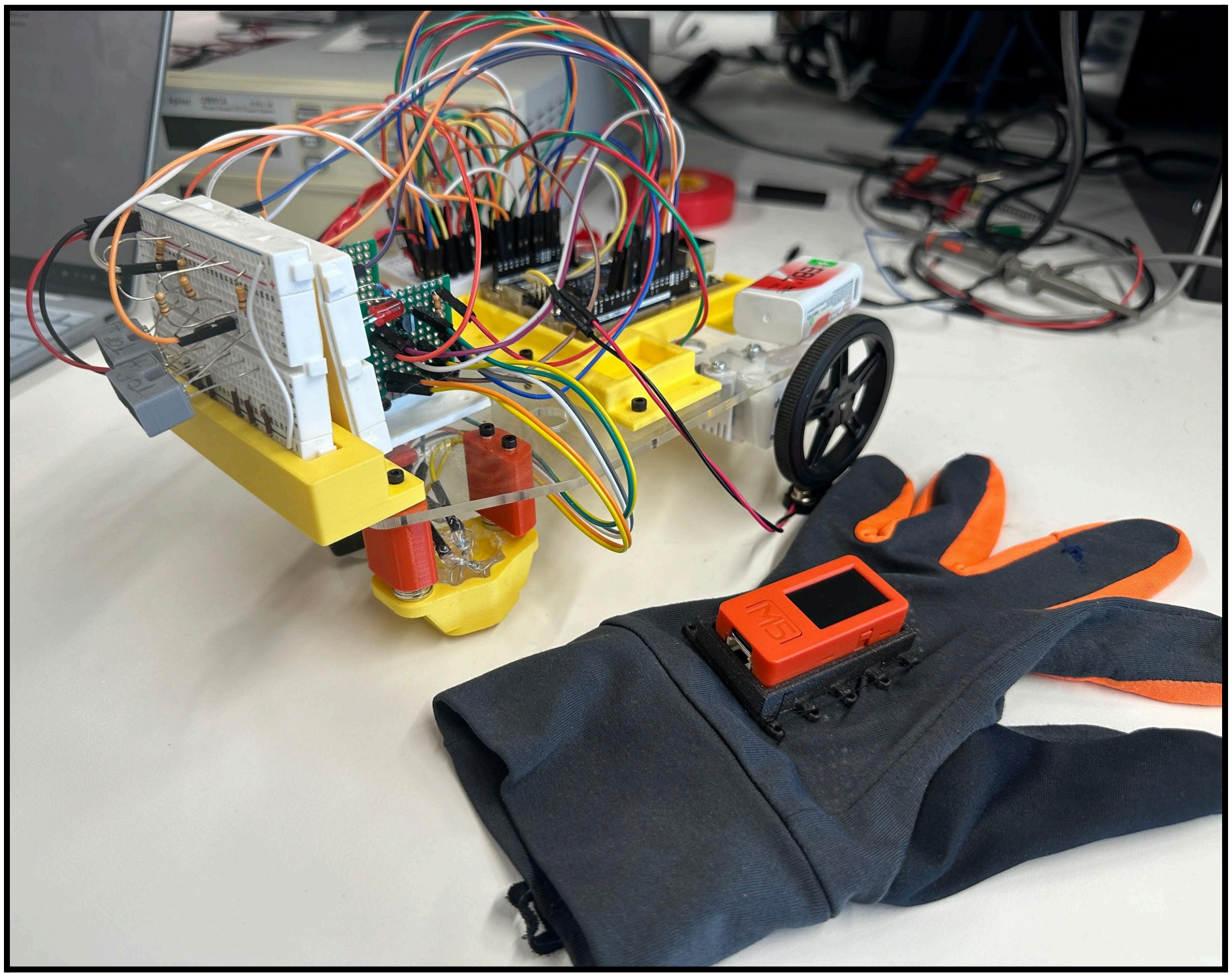


Figure 10. Picture of the Completed System

Addressing Customer Desires:

There were several ways in which customer desires were addressed in the development of the robot. Customers would want an indicator to check the status of the batteries and whether or not they needed to be charged. Thus, an LED voltage indicator circuit was implemented to show the user the current status of the battery that provides power to the motors. Monitoring the motor battery indicates whether both batteries need to be charged because the motors require more power, so when the motor battery level drops below the threshold, it would be reasonable to charge both batteries. Many of the wires were also braided and 3D printed component holders were produced, addressing the customer desire for neatness.

<u>Addressing Software Security Concerns:</u>

In order to prevent the bot from being controlled by an unauthorized machine over the Web Socket, we implemented a very basic encoding/decoding scheme in collaboration with our partner team. This encoding/decoding scheme's purpose is two-fold: with a noisy/heavily trafficked server, selecting which message to use can be difficult, and the security measure dramatically reduces the load that the Arduino must process. Additionally, the uniqueness in the bot commands sent via the Web Socket makes it so that a potentially malicious machine must know exactly what the encoding/decoding schema is, and then must send the message within a very short window in order to hijack the bot. The procedure is described in the next paragraph.

In addition to the ID/tag sent to the server with each message, the partner team must send, as one string "Waymo[Cmd][Time]". Any deviations from this will result in a command not being processed. This is all a secondary defense, since the stripping of the partner ID is performed at each WSreceive, verifying the identity of the sender.

<u>Quick Start Instructions - Track Navigation:</u>

1. Charge batteries to ~8.5V. Lower voltages may hinder bot performance.
2. Load up the desired program. For solo-track-nav, there is a boolean at the top of main.cpp that corresponds to the side of the track you wish to run. Look for "BLUESIDE" and upload the code to the bot.
3. Power on the robot by FIRST plugging in the arduino (look for battery leads connecting directly to the arduino). Then, SECOND, quickly plug in the battery attached to the h-bridge breadboard.
    a. Usually, there will be a few second pause as the bot attempts to connect to wifi and then it will drive straight. Sometimes, if you do not plug in the batteries quickly, or in the wrong order, you will start in a turning state. If this happens, power cycle the bot and restart (3).
4. Set down the bot on the corresponding side of the track and cross your fingers!

<u>Quick Start Instructions - Go Beyond:</u>

1. Charge the batteries to approximately 8.5V.
2. Load the main.cpp program from the EE31-Project repository onto the robot
3. Plug the battery directly connected into the Arduino. DO NOT plug in the motors yet.
4. Load the main.cpp program from the EE31_M5StickC library onto the M5 Stick if it has not been already.
5. Disconnect the glove from the computer. If the LCD turns off, press and hold the leftmost button until power returns to the controller.

6. Press the button on the upper right side of the casing until you see Mode: Glove on the screen.
7. Put your hand straight out in front of you, with your palm face down. Then, with your other hand press the large button that says M5. You should then see “Command: S” on the screen.
8. Now that your robot is safely stopped, you can plug the motor battery in. Try and keep your glove hand flat while you do this.
9. The robot should now be able to drive. Rotate your hand left and right to turn. Raise your arm to drive forwards and lower it to reverse. Return your arm to the neutral position to make the robot stop. Refer to the posing diagram on the next page for a more precise depiction if needed.
10. If at any time you feel the controls have become unresponsive, press the central button to reset the internal angle measurements.

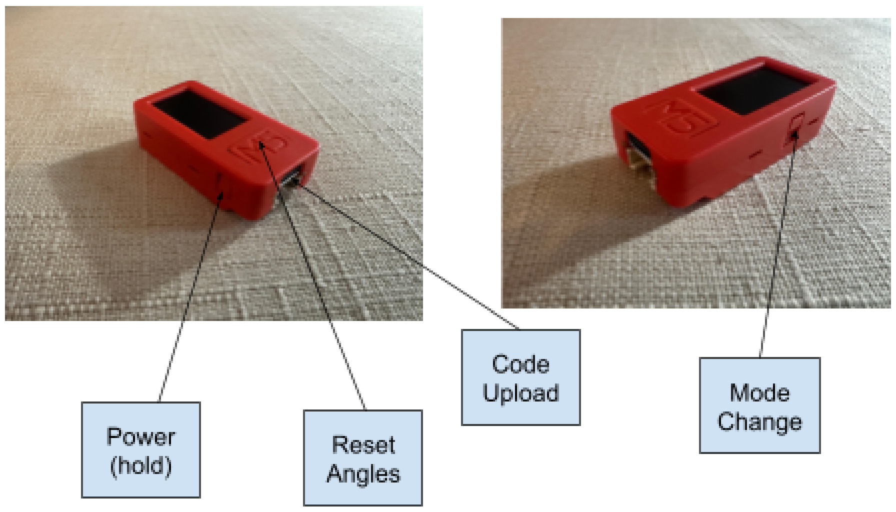


Figure 11. Controls for the Go Beyond Controller

<u>Posing Diagram:</u>

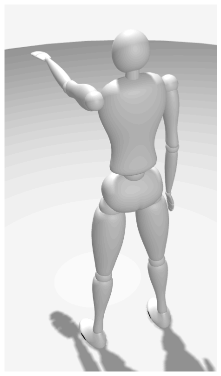
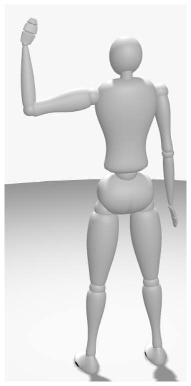
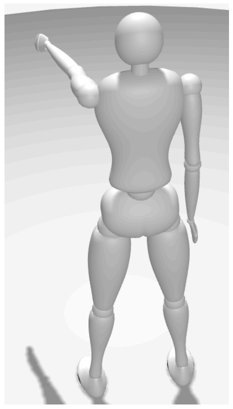
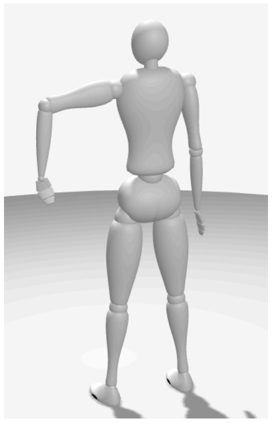
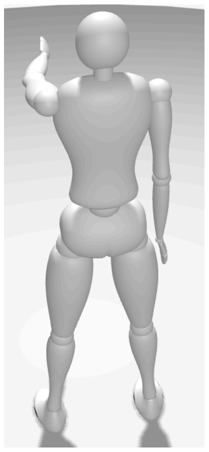
Action	Image	Action	Image
Stop (S)		Go (G)	
Turn Left (L)		Reverse (B)	
Turn Right (R)			

Table 2. Available poses for the Go-Beyond Glove Controller