

FUN-E-MOUSE

Team 50: Chuangy Zhang, Yingyu Zhang, John Casey

ECE 445 Final Report - Spring 2022

TA: Jamie Xu

May 4th, 2022

Abstract

The Fun-E-Mouse is aimed at providing a fun interactive playtime experience for both the owner and the cat. Smartphone remote-controlled driving gives the device greater accessibility to more users who want to play with their cats directly. Automatic mode uses sensors to drive the toy in a randomized motion to keep the cat entertained for hours without harming it or surroundings.

Table of Contents

Table of Contents	3
1. Introduction	5
1.1 Problem and Solution Overview	5
1.2 High-Level Requirements	5
2. Design	6
2.1 Block Diagram	6
2.2 Physical Design (See Appendix C)	7
2.3 Block Description	7
2.3.1 Power Management Subsystem	7
2.3.2 Control Subsystem	7
2.3.3 Sensory Subsystem	8
2.3.4 Drive Subsystem	8
3. Design Verification	9
3.1 Power Management Subsystem	9
3.2 Control Subsystem	11
3.3 Sensory Subsystem	13
3.4 Drive Subsystem	14
3.5 Network Subsystem	15
4. Costs	17
4.1 Parts	17
4.2 Labor	18
4.3 Manufacturing	18
4.4 Total	18
5. Conclusion	19
5.1 Summary	19
5.2 Ethicals /Safety	19
5.3 Future Work	20
References	21
Appendix A: Requirement and Verification Table	22
Appendix B: Schematics of Overall System	26

1. Introduction

1.1 Problem and Solution Overview

The most beneficial toys for cats' health are those that are chasable and satisfy the cat's natural hunting instinct. While there are currently many chasable toys on the market, including those that are autonomous or controlled by smartphones, they all encounter the same persistent issues. Currently, there exists no cat toy devices on the market that are both autonomous and have smartphone remote control. Cat owners in the modern era are looking for a device that can entertain their pets when they are seen as distractions while working remotely at home or partaking in other activities. With the addition of self-driving capabilities, these toys can safely navigate one's home while exercising their feline friend. The inclusion of obstacle-avoiding self-driving is useful when the owner is not around or to avoid breaking nearby furniture. This option also gives access to those who enjoy watching their pet play but do not choose to or have the ability to control the smartphone app. The smartphone RC mode also grants accessibilities to those with disabilities or injuries by allowing them to interact with their cats without physically playing with them.

In the current market, the majority of products are lacking in sufficient battery life which can be solved with higher capacity rechargeable batteries instead of the common replaceable sets. In addition, the ability of current chasable mouse toys to transition between different floorings and function properly on those different flooring materials is subpar due to motors with low torque or low voltage batteries. With a high voltage, higher capacity lithium ion battery the extra power would allow motorized cat toys to work with no issues on any flooring material.

There is a need for a smart-phone controlled cat toy that tackles all the issues listed previously that are involved in the current market's options. The need for this type of device has been expressed to us by an ECE business office faculty member, Erin Kristovich. Since originally pitching the idea she has been very interested in our progress and has taken our completed product home to test with her own cats.

1.2 High-Level Requirements

1. **Remote Control via Smartphones:** With the REMOTE mode on, the owner can remotely control the direction of the mouse via a web app from at most 15 feet away.
2. **Rechargeable Battery:** This device is rechargeable through the built-in micro-USB port and able to supply at least 30 minutes in one charge.
3. **Self Obstacle Avoiding:** The mouse can automatically sense and make a detour of obstacles on its path without running straight into it when in AUTO mode.

2. Design

2.1 Block Diagram

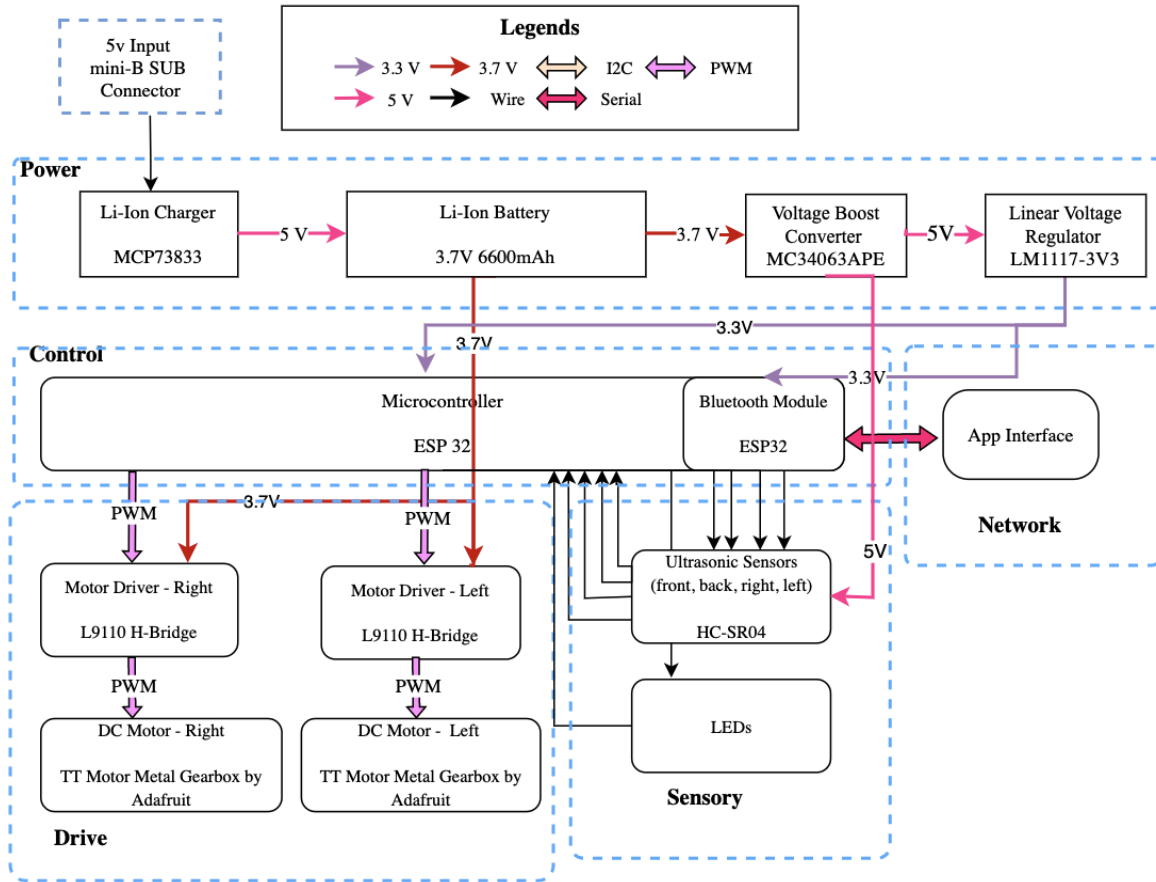


Figure 2.1 Block Diagram

The Fun-E-Mouse project has five major subsystems: power management system, control unit system, drive system, sensory system, and network system. The power management system has a battery charger, a lithium-ion rechargeable battery, and a voltage regulator. The power management system allows us to provide a stable 3.3 V supply to all other four subsystems, it also allows us to recharge the lithium-ion battery. The control unit system has an ESP32 MCU designed by the Espressif System, it is responsible to process all the data coming from the sensory system and network system. The control unit system controls the drive system, and that allows the Fun-E-Mouse to drive smoothly and avoid any obstacles. The sensory subsystem consists of four ultrasonic sensors and two LEDs. The four ultrasonic sensors are located at the front, right, back, and left respectively. This allows the Fun-E-Mouse to detect any obstacles nearby and send the data to the control unit system. The network

system has an App Interface(IOS/Android/Web) that allows us to control the Fun-E-Mouse remotely from a long distance, and we can also configure the different play modes remotely. This network system uses the Bluetooth communication protocol to send the serial command to the ESP32, then ESP32 will process the serial data to control the LEDs, direction and playing mode of the Fun-E-Mouse.

2.2 Physical Design (See Appendix C)

We 3D designed and printed the shape of the mouse in two parts, the top shell and bottom baseplate. We will have the sensors interlock onto the shell of the mouse; the PCB, motors, and battery mount on the baseboard of the mouse inside the “stomach”. This shape encloses all the components of the electronic mouse and has reserved openings for the sensors and LEDs, as well as a micro-USB charging port. An extensive amount of work has gone into the 3D design based on the assumption of dimensions found on datasheets or websites for the parts we have ordered. From scratch to the final design, due to fluctuations in part size, measurements and fitments were changed several times to ensure proper sizing of the 3D printed shell. A strings was attached to the tail of the electronic mouse to make it attractive for cats.

2.3 Block Description

2.3.1 Power Management Subsystem

The power management subsystem consists of a 3.7 V Polymer Li-ion Battery with 6600 mAh by Adafruit, a Lithium Polymer battery charger based on the MCP73833, and a 3.3 V linear voltage regulator that can provide up to 250 mA, and a voltage boost converter that can output 5 V. The power management subsystem allows us to provide a stable 3.3 V output that powers up all other four subsystems. The lithium-polymer battery charger uses a USB mini-B interface to charge the battery. That can come from any computer or any USB wall adapter. The fast-charge current is 500 mA by default, but we can easily adjust the charge current from 100 mA to 1000 mA. The lithium-polymer battery is made of three balanced 2200 mAh cells. Each cell can provide 1.1 A, therefore, the peak current that we can draw is 3.3 A which allows us to power the Fun-E-Mouse for more than 30 minutes. The voltage regulator that we chose to use is the LV1117-33 because this linear voltage regulator can provide a nice clean 3.3 V with 2% regulation. It is perfect to power up the microcontroller, and the 5 V output will power up the other subsystems.

2.3.2 Control Subsystem

The control unit subsystem has an ESP32 MCU designed by the Espressif System. ESP32 has been widely used in many IoT devices, and it is well documented and easy to develop and program. The ESP32 is a low-cost, low-power system on chip with Wi-Fi and Bluetooth capabilities. It also supports

PWM, I2C, and SPI communication protocols. The control unit subsystem sends commands to the drive subsystem through PWM communication protocols. This allows us to drive Fun-E-Mouse left, right, backward, or forward. We also can control the speed of the Fun-E-Mouse as well. The control unit subsystem is also responsible for receiving data from the networking subsystem, and that allows us to control the Fun-E-Mouse within 15 feet distance. In addition, the controller is powered by 3.3 V power from the voltage regulator of the power management subsystem.

2.3.3 Sensory Subsystem

The sensory unit consists of four Ultrasonic Sensors and two LEDs. Ultrasonic Ranging Module HC-SR04 was chosen due to its feature of having a 2 cm - 400 cm non-contact sensing range. They require low power which is suitable for the mouse since it is a battery-powered device. Each side (front, back, left, and right) of the mouse will have one ultrasonic sensor. When one of the sensors detects an obstacle, the mouse will move away from it. To be further specific, the Echo pins of the ultrasonic sensors will go low when the pulses sent out by the sensors are reflected back. The output of the Echo pins will go into the microcontroller as inputs.

2.3.4 Drive Subsystem

The drive subsystem is to drive the Fun-E-Mouse to move forward, backward, rightward, or leftward. This subsystem has two L9110 H-bridge motor drivers for DC motors and 2 metal gear TT DC motors. The L9110 H-Bridge motor driver can direct a single DC motor bi-directionally, and it can provide a peak current of 800 mA. It is good to drive any motor power voltages from 2.5 V up to 12V. There's a PWM input that we can control the speed of the direction of the motor. The all-metal gear TT motors are an easy and low-cost way to get the Fun-E-Mouse to move on different surfaces. It has higher torque and slower rotational speed, and they provide more strength and less speed. It can operate from 3V to 6 V, and it can go faster at higher voltages. At 3 V, it can operate 60 RPM with no load, and it only draws about 0.5 mA when it is stalled.

2.3.5 Network Subsystem

The network subsystem consists of a bluetooth module and an application interface that is developed in IOS, Android, and Web App. The subsystem allows us to control the Fun-E-Mouse remotely. The application interface should enable the user to turn on or off the LED of the Fun-E-Mouse. It also has mode control for Remote and Auto which allows users to switch the playing modes of the Fun-E-Mouse. When the Fun-E-Mouse is in the Remote Mode, then the joystick will be enabled otherwise, the joystick will be disabled. The application interface should be stable and reliable. The latency of the response time must be under 1 second to create a friendly real-time control experience.

The joystick allows the user to control the Fun-E-Mouse to move rightward, leftward, backward, and forward when the Fun-E-Mouse is in Remote Mode.

3. Design Verification

3.1 Power Management Subsystem

The power management subsystem is one the most important subsystems of our design. There are three important requirements that we have to meet to power up the rest of the subsystems. First, we have to make sure that it can provide a 5 V output voltage from our Boost Converter. This is very critical for the sensory subsystem to power up all four ultrasonic sensors. Thus, we soldered all the Boost Converter’s electronic components on the PCB and we connected 3.7 V input voltage from the power supply to the Boost Converter, then we used the oscilloscope to measure the output voltage of the boost converter.

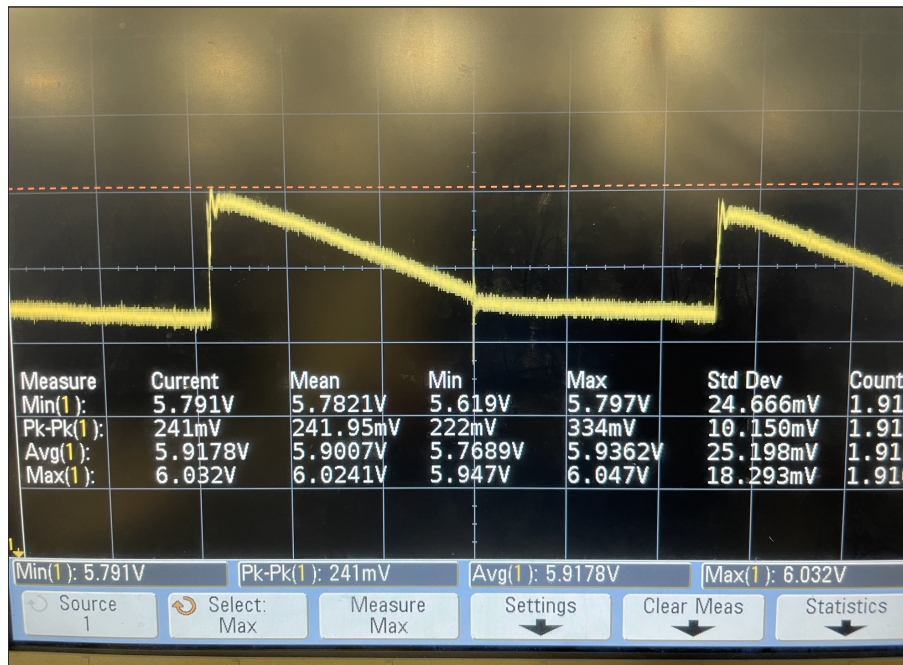


Figure 3.1a Output Voltage of the Boost Converter

Figure 3.1a shows the output voltage of the boost converter and the average output voltage is about 5.9178 V which didn’t meet our requirement. But we were able to power the ultrasonic sensors in the sensory subsystem.

Second, we have to verify that the power management subsystem outputs a clean 3.3 V voltage to power up the control subsystem. We soldered the linear voltage regulator on the PCB and we used 3.7

V input voltage, then we used the oscilloscope to measure the output voltage of the linear voltage regulator.

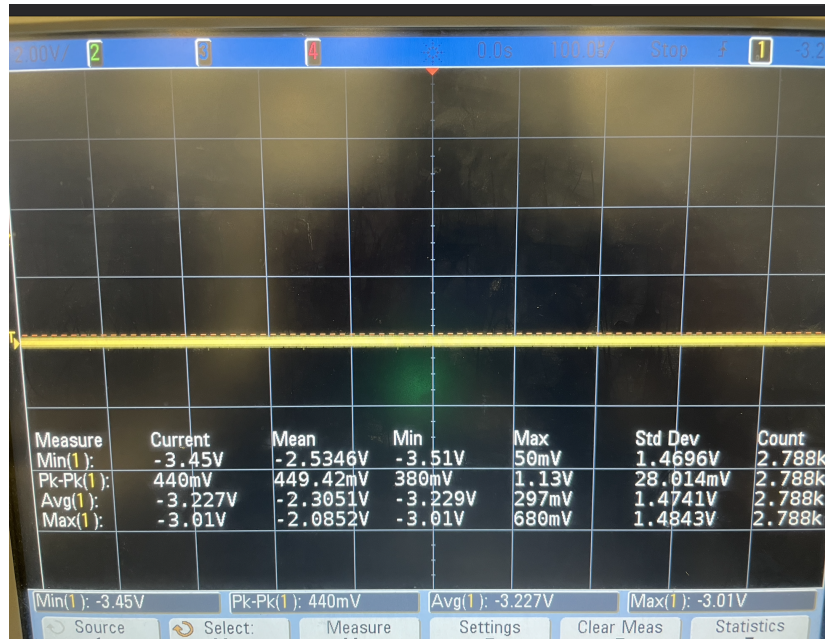


Figure 3.1b Linear Voltage Regulator Output

Figure 3.1b shows the output voltage of the linear voltage regulator. As we can see the average of the voltage is 3.227 V and our requirement was met and verified. By using the voltage regulation formula, we can solve for the regulation percentage of the 3.3 V output as below[1].

$$3.3 \text{ V Regulation Calculation} = \frac{3.30 \text{ V} - 3.227 \text{ V}}{3.227 \text{ V}} \times (100\%) = 2.26 \%$$

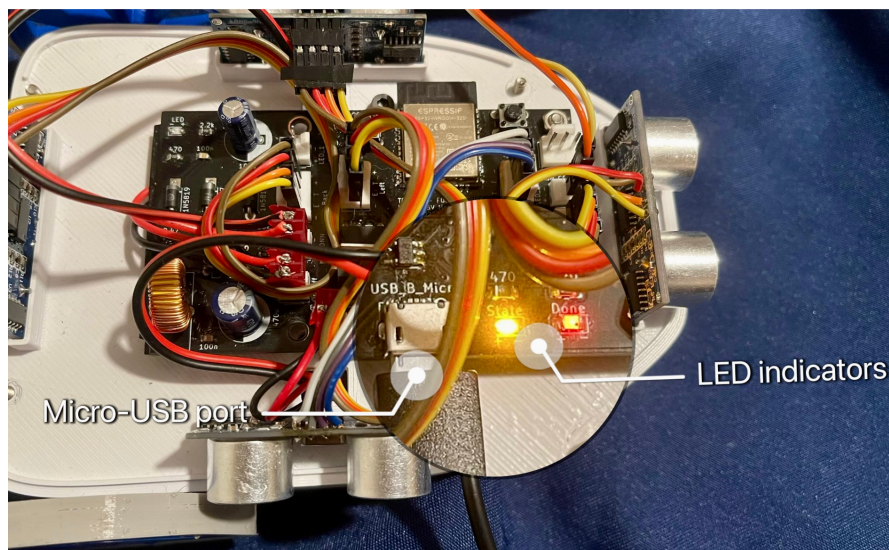


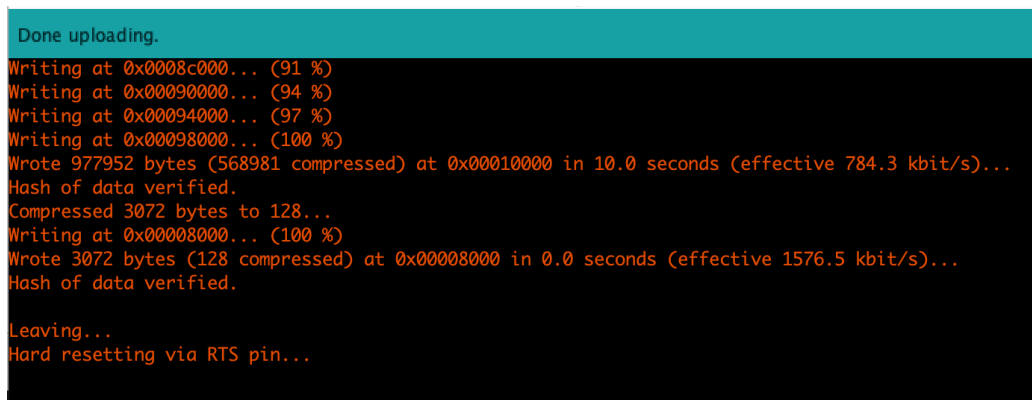
Figure 3.1c Charging Module

We also did some verification to check the performance of the battery. As shown in Figure 3.1c, there is a Micro-USB port mounted on the PCB, with two battery status indicators right next to the micro-USB port. The orange LED indicates that the battery is charging. When the battery is fully charged, both the orange and the red LEDs are on. We started from a completely drained battery, turned on a stopwatch, and plugged the mouse into a power outlet through the micro-USB port on the PCB. It took about 8 hours for both the red and the orange battery status indicators to light up at the same time. In order to find out how long can the battery supply the mouse in one charge, we set the mouse on Automatic mode starting with a fully charged battery and turning on a stopwatch. It took about 12 hours and 14 minutes for the battery to die out (the speed of the mouse decreased significantly). As a result, the battery needs about 8 hours to be fully charged with a charging current of 100 mA, and it is able to supply for more than 12 hours on one charge.

3.2 Control Subsystem

The control subsystem contains a ESP32 microcontroller and it is integrated with Wi-Fi and Bluetooth connectivity for a wide range of applications. ESP32 is the best choice for any real-time control device.

Since we are building our own PCB board, ESP32 will be mounted on the PCB, thus, we wanted to build a circuit that allows us to program the ESP32 through any USB bootloader. In order to meet this requirement, we need to build a reset and flash push buttons to upload the Arduino Code to ESP32.



```
Done uploading.
Writing at 0x0008c000... (91 %)
Writing at 0x00090000... (94 %)
Writing at 0x00094000... (97 %)
Writing at 0x00098000... (100 %)
Wrote 977952 bytes (568981 compressed) at 0x00010000 in 10.0 seconds (effective 784.3 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1576.5 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Figure 3.2a Arduino console output when done uploading

Figure 3.2a shows when Arduino successfully uploaded the binary code to ESP32. This verifies that ESP32 can be programmed anytime when we change our code.

The other requirement of the control subsystem is to be able to receive data at a baud rate of 115200 baud. The requirement allows us to debug our program easier by looking at the Serial Monitor. The serial monitor allows us to print text messages from ESP32 and it comes very handy for us to debug our program.

```

/dev/cu.usbserial-A10L1AG4
Send
21:25:26.515 -> Setting Up Drive SubSystem
21:25:26.515 -> Setting Up Sensory Subsystem
21:25:36.403 -> *****
21:25:36.403 -> New value: true,false,false,0.0,0.0
21:25:36.403 -> *****
21:25:37.300 -> *****
21:25:37.300 -> New value: false,false,false,0.0,0.0
21:25:37.300 -> *****
21:25:38.470 -> *****
21:25:38.470 -> New value: false,true,false,0.0,0.0
21:25:38.470 -> *****
21:25:39.637 -> *****
21:25:39.637 -> New value: false,true,false,0.2,4.3
21:25:39.637 -> *****
21:25:39.841 -> *****
21:25:39.841 -> New value: false,true,false,0.8,7.8
21:25:39.841 -> *****
21:25:40.052 -> *****
21:25:40.052 -> New value: false,true,false,5.9,4.4
21:25:40.052 -> *****
21:25:40.333 -> *****
21:25:40.333 -> New value: false,true,false,1.9,-4.8
21:25:40.333 -> *****
21:25:40.475 -> *****
21:25:40.475 -> New value: false,true,false,-0.8,4.8
21:25:40.475 -> *****
21:25:40.543 -> *****
21:25:40.543 -> New value: false,true,false,0.0,0.0
21:25:40.543 -> *****

 Autoscroll  Show timestamp
Newline 115200 baud Clear output

```

Figure 3.2b Serial monitor console at 115200 baud

Figure 3.2b shows the serial monitor console of ESP32. It outputs control subsystem information for us to debug. When there's a bluetooth transaction, it outputs the value which it received. This also verified that we met the requirement.

Lastly, one of the requirements is to maintain a stable bluetooth connection up to 15 ft. Typically Classic Bluetooth and Low Energy Bluetooth can maintain up to 24 ft of connection. But poor connection still can exist when ESP32 doesn't have enough voltage or PCB was designed improperly. We took these into consideration, and we were able to maintain a stable connection up to 36 fts. Figure 3.2c verifies that we were able to control the Fun-E-Mouse about 36 ft apart.

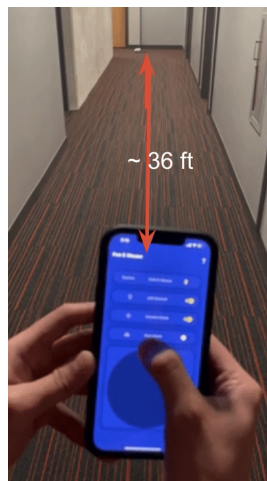


Figure 3.2c Distance between Fun-E-Mouse and App

3.3 Sensory Subsystem

The sensory subsystem contains four ultrasonic sensors. They work together to achieve obstacle detection. When we were unit testing each ultrasonic sensor, we connected each ultrasonic sensor to an Arduino board: the Vcc pin of the ultrasonic sensor to the Vin pin of Arduino Nano, the Trig pin to the D2 pin, the Echo to the D4 pin, and GND to GND. And run the code as shown in Figure 3.3 that we found online[2]. The ultrasonic sensors were able to detect objects that are as small as 2 cm x 5 cm x 4 cm and objects that are as large as 20 cm x 6 cm x 6 cm within a detected distance of 6 inches to 26 inches. The sensors were able to send the correct detected distance to the control subsystem.

```
void setup() {
  pinMode(2, OUTPUT); //define arduino pin
  pinMode(4, INPUT); //define arduino pin
  Serial.begin(9600); //enable serial monitor
}
void loop() {
  //pulse output
  digitalWrite(2, LOW);
  delayMicroseconds(4);
  digitalWrite(2, HIGH);
  delayMicroseconds(10);
  digitalWrite(2, LOW);

  long t = pulseIn(4, HIGH); //input pulse and save it variable

  long inches = t / 74 / 2; //time convert distance
  long cm = t / 29 / 2; //time convert distance
  String inch = " inches t";
  String CM = " cm";

  Serial.print(inches + inch); //print serial monitor inches
  Serial.println(cm + CM); //print serial monitor cm
  Serial.println(); //print space
  delay(100); //delay
}
```

Figure 3.3 Sensor Performance Test

3.4 Drive Subsystem



Figure 3.4 Motor Speed Test

The Drive subsystem contains two DC motors and H-bridges. The primary requirement of the Drive subsystem is to have a forward and backward speed of 1m/s. We did some verification by having the mouse running in a straight line, and using distance over time to find its actual speed. As shown in Figure 3.4, it took about 6 seconds for the mouse to travel 2 meters, which means the mouse has a speed of 0.333 meter per second.

$$\frac{2 \text{ m}}{6 \text{ sec}} = \frac{1 \text{ m}}{3 \text{ sec}} < \frac{1 \text{ m}}{1 \text{ sec}}$$

According to the datasheet of the TT motor that we use, it can draw 160 mA @ 250 RPM at 6 V DC [3]. Speed is directly proportional to the input voltage. As shown in the equation below, with an ideal input voltage, the theoretical top speed that the mouse can reach is 0.85 m/seconds with wheels that have a diameter of 0.65 cm.

$$(0.65 \text{ cm} \times \pi) \times \frac{250 \text{ revolutions}}{1 \text{ min}} \times \frac{1 \text{ min}}{60 \text{ sec}} = \frac{0.85 \text{ m}}{1 \text{ sec}}$$

Hence, in order to have a speed of 1m/sec, we need to redesign our Drive subsystem to change our motors to those that have a higher revolutions and use a battery that has a higher voltage.

3.5 Network Subsystem

The network subsystem is a software application that allows us to control the Fun-E-Mouse in real time. Any real time software application needs to take latency into consideration. The higher latency creates a bad user experience in any real time control environment. Thus, one of the three requirements is to have a low latency between the software application and the ESP32. Our initial design was to use the Wi-Fi to control the Fun-E-Mouse. Our initial network subsystem consisted of RESTful APIs that were hosted on a server. The software application constantly sends out a POST request to update the command data to the server, and then ESP32 constantly sends out a GET request to retrieve the command data from the server. With this framework, control commands are bounced from software application to server and server to ESP32, creating 2 layers of network latency. Figure 3.5a shows our initial design for the network subsystem. Thus, the Wi-Fi framework is not an ideal communication solution for a real time control device.

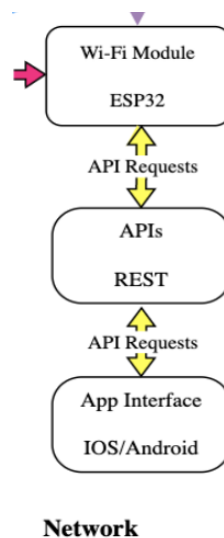


Figure 3.5a Initial Design for Network Subsystem

After changing our communication protocol to Bluetooth, the software application can directly communicate with ESP32, and its latency is around 68 milliseconds. Figure 3.5b shows the calculation of the latency of communication when Fun-E-Mouse in Remote control mode.

```
21:25:39.637 -> New value: false,true,false,0.2,4.3
21:25:39.637 -> *****
21:25:39.841 -> *****
21:25:39.841 -> New value: false,true,false,0.8,7.8
21:25:39.841 -> *****
21:25:40.052 -> *****
21:25:40.052 -> New value: false,true,false,5.9,4.4
21:25:40.052 -> *****
21:25:40.333 -> *****
21:25:40.333 -> New value: false,true,false,1.9,-4.8
```

Figure 3.5b Four Communication Transactions between App and ESP32

The delay between first and second transactions is about 204 milliseconds, the delay between second and third transactions is about 205 milliseconds, the delay between third and four transactions is about 281 milliseconds. The calculation below shows that we met our requirements.

$$\text{Average of Delay} = \frac{204 \text{ ms} + 205 \text{ ms} + 281 \text{ ms}}{3} = 230 \text{ ms}$$

The software application allows us to control the FUN-E-Mouse move forward, backward, rightward, and leftward. It also allows us to configure the playing modes (Auto and Remote), The joystick panel is enable where the Fun-E-Mouse in the Remote Mode, otherwise the joystick panel will be disabled. It has the LED control which allows us to turn on and off the LEDs that located at the eyes of the Fun-E-Mouse. Figure 3.5c shows the user interface of the software application and that met our requirement for the network subsystem.

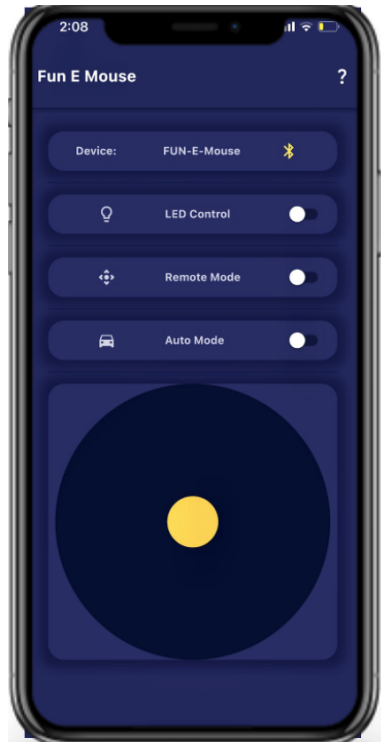


Figure 3.5c Software Application to control the Fun-E-Mouse

4. Costs

4.1 Parts

Part	Part Number	Manufacturer	Quantity	Unit Price
Mircocontroller	ESP32_WROOM_32D	Digikey	1	\$ 4.20
Metal Ball Caster	1738-1328-ND	Digikey	1	\$ 2.50
micro B USB Receptacle Connector	WM1399TR-ND	Digikey	1	\$ 0.96
Boost Switching Regulator IC	296-36112-5-ND	Digikey	1	\$ 0.92
Charger IC Li-Ion Battery	MCP73831T-2ACI/OTTR-ND	Digikey	1	\$ 0.76
On/Off Rocker Switch	EG5619-ND	Digikey	1	\$ 0.67
Tactile Switch	CKN9092-ND	Digikey	2	\$ 0.17
Ultrasonic Sensor	1528-2711-ND	Digikey	4	\$ 3.95
Diode Standard 1000 V 1A	1N4007GOS-ND	Digikey	1	\$ 0.30
Diode Schottky 40 V 1A	1N5819GOS-ND	Digikey	1	\$ 0.51
Voltage regulator	LD1117v33	Amazon	1	\$ 0.78
H-Bridge Motor Driver for DC Motors	L9110H	Adafruit	2	\$ 1.50
Lithium Ion Polymer Battery - 3.7V 10050mAh	5035	Adafruit	1	\$ 29.95
DC Gearbox Motor- 200RPM - 3 to 6VDC	3777	Adafruit	2	\$ 2.95

Headers & Connectors	—	—	—	\$ 4.00
Resistors & Capacitors	—	—	—	\$ 4.00
Total				\$ 74.59

Table 4.1 Bill of Material of One Product

4.2 Labor

According to Salary.com, the average hourly wage for Engineer I in the United States is \$34, and assumed with 250% overhead multiplier. Also, Senior design class is a 4 credit hour class, that means each student must spend at least 12 hours per week. We spent about 7 weeks on this Fun-E-Mouse project, therefore the total labor cost is calculated below:

$$\$34 / hr \times 2.5 (\text{overhead}) \times 12 hr / week \times 3 people \times 7 weeks = \$21,420$$

4.3 Manufacturing

Manufacturing Method	Hours	Rate per Hour	Material Cost	Overall Cost
3D Printing	30	\$ 1.50	\$25.00	\$70
PCB	N/A	N/A	\$4.99	\$4.99
Total				\$74.99

Table 4.3 Manufacturing Cost

4.4 Total

$$\$74.59 (\text{Parts}) + \$21,420 (\text{Labor}) + \$74.99 (\text{Manufacturing}) = \$ 21,569.58$$

Keeping the design of what we have and doing mass production is one potential strategy to reduce the cost of raw materials. Making bulk purchases can reduce the cost per unit. With bulk purchases, we can reduce the cost of raw materials or parts and manufacturing of one finished product by at least 10 %.

5. Conclusion

5.1 Summary

Overall, we were successful in creating a working prototype that we were able to test and modify into our final product that was demonstrated. Our final product meets most of the high level requirements that we set forth to achieve from the beginning; with self obstacle avoidance being the only requirement that needs improvement. Our device was able to achieve the obstacle avoiding self-driving mode, however, it had issues detecting objects of smaller sizes and narrower furniture. The sensors we used were not optimal and their detection of excess noise caused issues that we had trouble addressing without a complete redesign of our sensory subsystem. The app developed for the Fun-E-Mouse is intuitive for all users in the joystick remote-controlled driving mode as well as using the interface to connect to the device from a smartphone or change between modes. While we did come across problems along the prototype stage, we have attempted to address all issues that occurred in our final demonstrated product along with solutions that will be implemented in future iterations.

5.2 Ethicals /Safety

We conducted ourselves according to the Code of Ethics published by IEEE[6] and ACM[7] as following:

1. Be sure “to not engage in harassment or discrimination, and to avoid injuring others”[6].
2. Not make use of user data without consent while users are using our web app that controls the electrical mouse.
3. Be sure to adapt the principle of justice, ensuring all users are equal.
4. Be sure to avoid harm and ensure “to minimize the possibility of indirectly or unintentionally harming others”[7], including pets.

We also adhered to the Safety Guidelines set forth on the ECE 445 course website[5], as well as the document of General Battery Safety[4] provided on the ECE445 course website. Further, we need to protect ourselves and others in the lab when working with batteries. Batteries have many hazards, such as electrical shock, flammable gasses that could fire or explode, and battery acid that can burn skin or eyes. We need to consult the battery’s manuals for safety precautions and proper handling, as well as hazard identification. We need to make sure that the battery is properly secured in the equipment for the finished product to prevent injuries in end users (ex: cats) while improving our battery’s performance. The outer shell of the electronic mouse might contain some electrical (sensors) and mechanical (wheels) parts. We need to make sure that they are safe and not harmful to pets or humans since this is designed to be a toy for cats. We made sure the outer shell of the mouse is firmly mounted,

with no external parts that can easily be ripped off, or swallowed, causing life-threatening damage to cats.

5.3 Future Work

With more time to develop the network subsystem, the Fun-E-Mouse would be capable of using its ESP32 to connect to both Wifi and Bluetooth. This would allow users to turn on the device and play with their cats in both modes while away from home using the wifi connection. With the integration of proper servers and high-speed wifi connection the latency would be minimal compared to our previous wifi testing allowing users to have no issues controlling the device from a long distance. Our current design uses a high capacity battery for long battery life, however, the trade-off is reduced voltage that drives our wheels at a slower speed with less torque. Future versions of the device will include a higher voltage battery of 7.4 V to allow the motor to have a greater variance of speed and higher max speed. Another issue that occurred was brownouts due to spikes in current to the microcontroller. This will be addressed in the next iteration by our change in power source along with the addition of capacitive devices or brownout detection software to avoid issues currently occurring. The auto-driving feature is also in need of more accurate sensing units, such as those with limit modules to better detect the surroundings. In addition, the sensing subsystem's data is coming in one at a time from each sensor currently; multi-sensor fusion would allow the mouse to navigate more efficiently when it senses from all directions continuously. The design of the Fun-E-Mouse has many routes to improve its current functionality, without mass production the inclusion of the components needed to address the issues seen will increase the cost of the product greatly.

References

- [1] “Voltage Regulation | Transformers | Electronics Textbook.” All About Circuits, <https://www.allaboutcircuits.com/textbook/alternating-current/chpt-9/voltage-regulation/> . Accessed May 2022.
- [2] “Ultrasonic sensor with Arduino Nano - How does work Ultrasonic sensor.” SriTu Hobby, <https://srituhobby.com/ultrasonic-sensor-with-arduino-nano-how-does-work-ultrasonic-sensor/>. Accessed March 2022.
- [3] Adafruit, web page. Available at: <https://www.adafruit.com/product/3777#technical-details>. Accessed May 2022.
- [4] “General Battery Safety.” Safe Practice for Lead Acid and Lithium Batteries, [Online]13 April 2016, <https://courses.physics.illinois.edu/ece445/documents/GeneralBatterySafety.pdf>. Accessed February 2022.
- [5] “Safety ECE 445 - Senior Design Laboratory.” ECE 445, <https://courses.engr.illinois.edu/ece445/guidelines/safety.asp>. Accessed February 2022.
- [6] “IEEE Code of Ethics.” IEEE, IEEE Policies, Section 7 - Professional Activities (Part A - IEEE Policies), <https://www.ieee.org/about/corporate/governance/p7-8.html>. Accessed February 2022.
- [7] “ACM Code of Ethics and Professional Conduct.” Association for Computing Machinery, ACM, <https://www.acm.org/code-of-ethics>. Accessed February 2022.

Appendix A: Requirement and Verification Table

Power Management Subsystem		
Requirement(s)	Verification	Result
1. Provide a nice clean 3.3V and 5V output voltages with 5% regulation	1A. Connect a 10K resistor as load at the output of 3.3V and 5V 1B. Use a multimeter to check if the voltage and current meet the requirement 1C. Use an oscilloscope to check if the voltage signal is clean and within 5% regulation	1. Not satisfied
2. Able to recharge the battery from Computer, Wall USB adaptor, or power bank	2A. Plug the charging module via any micro USB cable into a computer, wall USB adaptors, or power bank 2B. Check if the Charging LED on the PCB board is on 2C. Leave the battery charging overnight, and check if the Done LED on the PCB board is on	2. Satisfied
3. Able to power the Fun-E-Mouse at least 30 minutes of continuously running	3A. Once the Fun-E-Mouse Project is done, we will leave the mouse run continuously and record the time 3B. Ensure the the Fun-E-Mouse meet the time requirement	3. Satisfied

Control Subsystem		
Requirement(s)	Verification	Result
1. Able to maintain a stable Wi-Fi connection of at least 15 feet	1A. Set the microcontroller 15 feet away from the Wi-Fi router 1B. Send a control command through RESTful API request 1C. Make sure the microcontroller is performing the right command	1. Able to satisfy it with Bluetooth instead of WiFi
2. The rate of RESTful Request is at least 10-30 per second	2A. Set counter variable in the program that keeps track of how many numbers of the requests 2B. Continuously to send requests in 1 second 2C. Ensure the total number of requests is within the range	2. Satisfied

Control Subsystem		
Requirement(s)	Verification	Result
	of 10 - 30	
3. After mounted to PCB, the microcontroller should be program, and it should able to transmit data at baud rate of 115200	3A. Connect the USB2UART program breakout board to the Tx, Rx, and GND pin 3B. Run the Wi-Fi Unit test code via Arduino IDE, and burn bootloader, The program should able to read the data output at a rate of 115200	3. Satisfied

Sensory Subsystem		
Requirements	Verification	Result
1. Able to detect a chair leg or water bottle within the range of 2cm - 50cm in front of the ultrasonic sensor	1A. Place a chair or a water bottle in front of the ultrasonic ranging module in the range of 2cm - 50cm 1B. Write an unit testing program that output if the sensor detects an object 1C. Check the serial output, then see if the sensor detected the object	1. Satisfied
2. Able to detect a 4x4x10 cm cube within the range of 2cm -50cm in front of the ultrasonic sensor	2A. 3D printed a 4cm x 4cm x 10cm cube, and place the cube in front of the ultrasonic ranging module in the range of 2cm - 50cm 2B. Write an unit testing program that output if the sensor detects an object 2C. Check the serial output, then see if the sensor detected the object	2. Satisfied
3. Able to turn on or turn off the LED remotely at most 15 feet away	3A. Write an API request to turn on/off the LED 3B. Send a control command from the user interface at most 5 feet aways 3C. Check if the LED meet the corrected status	3. Satisfied

Drive Subsystem

Requirements	Verification	Result
1. Able to move forward and backward at speed of 1m/s	1A. Find an empty hallway and make a 10m path 1B. Write a program that control the motors to drive forward or backward at speed of 1m/s 1C. Check if the drive subsystem able to do 90 degrees right-turn and 90 degrees left-turn	1. Not Satisfied, 0.333 m/sec maximum.
2. Able to turn a 90 degree right turn or a 90 degree left turn	2A. Find an empty hallway and make a 90 degree left-turn path and a 90 degree right-turn path 2B. Write a program that can control the motor to turn left or turn right 2C. Check if the drive subsystem meet the requirement	2. Satisfied
4. Able to stop at forward speed of 1m/s	3A. Find an empty hallway and make a 10m path 3B. Write a program that control the motors to drive forward at a speed of 1m/s, and then after 5 second, control the motors to a complete stop 3C. Check if the motors come to a complete stop	3. No satisfied

Network Subsystem		
Requirements	Verification	Result
1. Able to detect and connect with any 2.4GHz Wi-Fi	1A. write a program that will scan all the nearby Wi-Fi 1B. Inspect the output of the Wi-Fi scan program and determine if it detects all the 2.4GHz Wi-Fi 1C. Provide a known Wi-Fi's SSID and password, and ping google.com to check if it is connected with network	1. Able to satisfy it with Bluetooth instead of WiFi
2. Able to send/receive data to/from the server in most 1 second	2A. Write a program that will send a POST request and a GET request, and have a time variable to keep track of how long the request took 2B. Check the time variable to determine if the Wi-Fi module has met the time requirement	2. Satisfied
3. Able to process the request in most 1 second	3A. Write a program that will send a POST request or a GET request, and each API has had a time variable to keep track of how long the request took	3. Satisfied

	3B. Check the time variable to determine if the API process that data in most 1-second	
4. Able to control the Fun-E-Mouse to move left, right, forward, or backward in 1 second	4A. Set a time variable to record when the control button is pushed, set another time variable to record when the microcontroller received the control command 4B. Check if the time when the controller received the command minus the time when the button was pushed is less than 1 second	4. Satisfied
5. Able to configure the 2 different modes (automatic, remote)	5A. Set the mode to automatic from User 5B. Check if the Fun-E-Mouse controls itself 5C. Switch the mode to remote 5D. Check if the user interfaces are able to control the Fun-E-Mouse	5. Satisfied

Appendix B: Schematics of Overall System

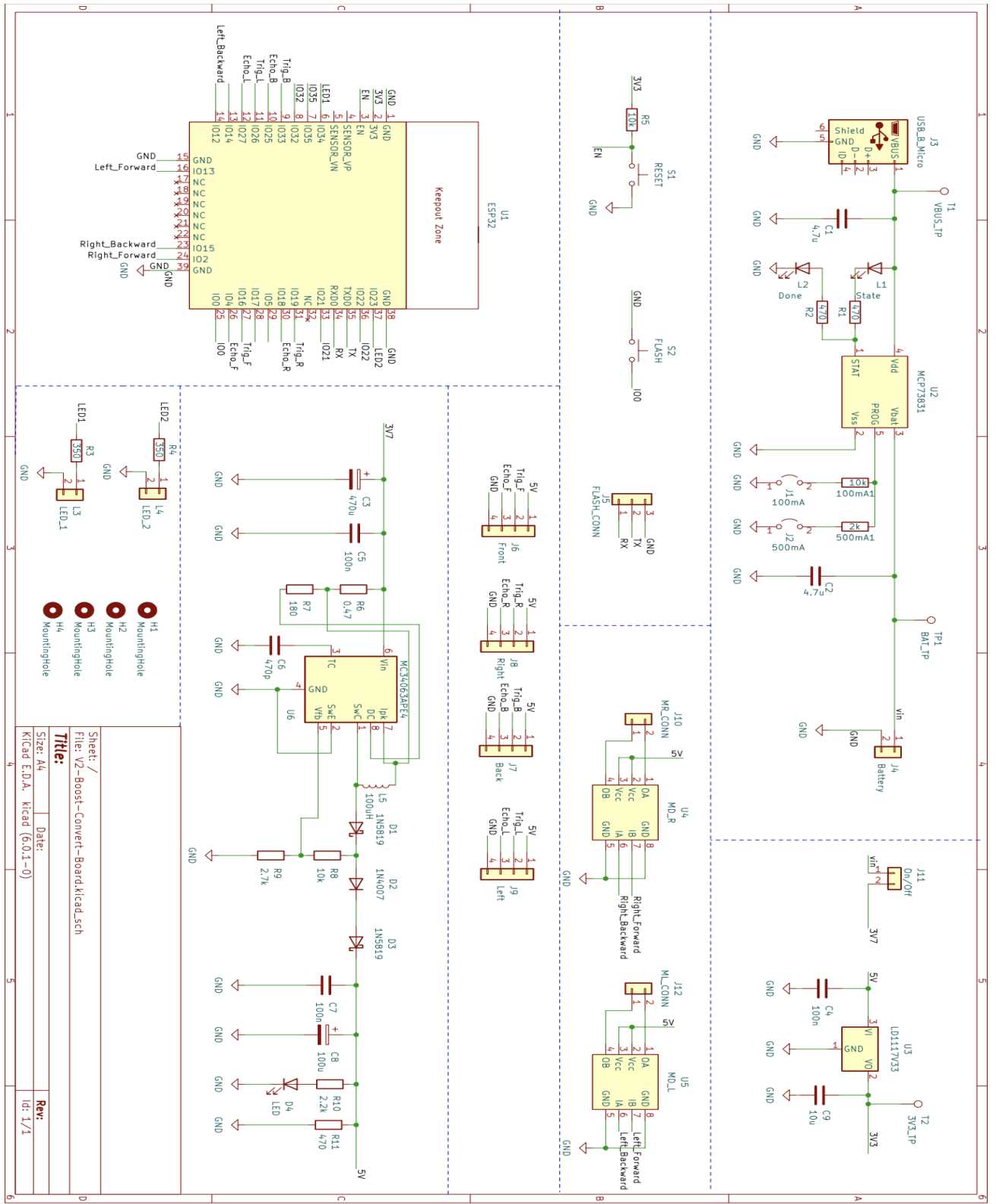




Figure 3.E Prototype View1

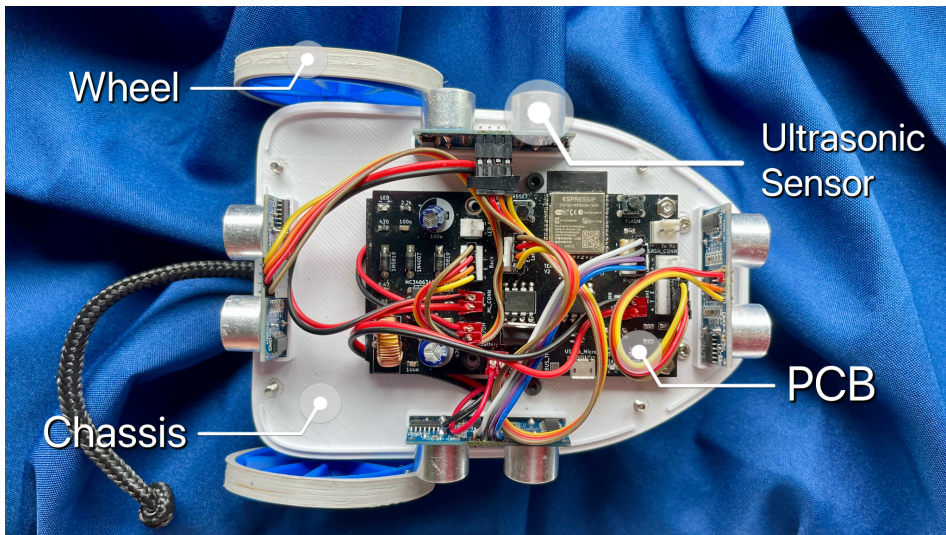


Figure 4.E Prototype View2