**To:** Jie Yang

**From:** Team 5, EE-286 Section 1Mason Gerace, Felecia Hildebran, Joshua Pollock, Taylor Yee  
**Date:**  December 4, 2018 **RE:** Project 3 - Final Report

This report will showcase team 5’s entire project to build a device that has at least two motors, three sensors, and a minimum of five LEDs. This report will start out by giving an introduction to the project, as well as a problem statement and backstory. Next, we will go into detail on the customer needs, wants, requirements, and constraints of the project. Then we will have the matrices we used to help determine which idea to use and a summary of our background research. We next will go into the process of building and testing our design, then proceed to summarize our demonstration. Lastly, we will discuss our design process in great detail.

Attachments: none

**Memo cover sheet – the memo should introduce the report and list all attachments to the report. The report should begin on a new page.**

**Introduction and Problem Statement**:

**Introduction:**

We started out with 4 designs that were introduced to us and we had to create a fifth and final idea ourselves. The designs introduced to us were: a launcher, a digital metronome/8-bit music player, drone simulator, and a clock. The idea we came up with was making an autonomous rover. The customer wanted this project tested and finished by December 4th. This report will go into detail about the requirements and constraints, our matrices, our background research, how we built and tested our project, our demo results, and finally our design process we went through.

**Problem Statement:**

The team must create a project using the Arduino, two motors, and three sensors. Additionally, the project must make noise and include flashing lights.

**Backstory and Formal Problem Statement:**

**Backstory:**

Elon Musk has been looking for concepts for a autonomous rover to roam Mars. However, every rover he has looked at thus far has ran into objects and not met his expectations. Elon Musk would like a unique rover that is easy to build and program. He would also like a rover that is creative, cost effective, and simple.

**Formal Problem Statement:**

The team must create a rover-like toy using the Arduino, two motors, and three sensors. Additionally, the toy must make noise and include flashing lights.

**Needs, Requirements, and Constraints:**

**Needs:**

* Use four motors
* Be able to tell how far the rover is from an object
* Must make noise
* Must include flashing lights

**Requirements:**

* Use at least 2 motors
* Use at least 3 sensors of which only 1 can be mechanical
* Use an arduino

**Constraints:**

* Amount of time to build project
* Cost - while our team can spend money on this, we are trying to avoid having to do so as much as possible

**Designs, Pairwise and Decision Matrices:**

The first design, a launcher, would be able to launch an object. The user would be able to adjust the launch angle and direction with two motors that would be controlled by a joystick. LEDs would light in response to the projected distance (farther = more LEDs lit), a buzzer could serve as a countdown timer, and an LCD screen could provide additional information.

The second design, a digital metronome, would cater more to musicians. Two motors would run simultaneously and generate rhythms that could be adjusted by the user through use of a peripheral: a joystick, IR remote, or potentiometer. An LCD could be used to write additional details, and a piezoelectric buzzer could provide an additional sound for drummers, to alert them of what beats to hit.

The third design, a drone simulator, would simulate the flight of an RC drone through the user of 4 motors and one or two joysticks. The LED lights would be used to simulate the direction(s) of the flight path, and an LCD would be used to write additional details, such as flight angles and trajectories, speed (in RPM), direction and tilt.

The fourth design, a clock, would have two motors that would change according to the time. Additionally, at certain times different LEDs would light up and the time would be displayed on an LCD screen as well as a seven-segment display. The device would be able to be turned on/off with the use of a button.

The fifth design, an autonomous rover, would require the use of many different sensors to ensure it functions properly. For example a sonar sensor would be used up front to keep the rover from slamming into objects. LEDs, the LCD, or the 7-segment display could be used to provide feedback to the user from the rover. Side sensors would be required to ensure the rover does not bump into items while turning as well.

Below is the pairwise comparison matrix. We used a 1,3,5 scale to rank the criteria based on their importance to the project. Buildability and programmability ranked the highest, because we felt that these were the most important to focus on. Once we could build and program our device, then we could turn to other factors, such as creativity. Cost and simplicity had the smallest weights because the team believed that these were not as crucial to the success of the project. Creativity fell in the middle of all the criteria.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Buildability** | **Programmability** | **Creativity** | **Cost** | **Simplicity** | **Geometric Mean** | **Weight** |
| **Buildability** | 1 | 1 | 3 | 3 | 5 | 2.14113 | 0.34417 |
| **Programmability** | 1 | 1 | 3 | 3 | 5 | 2.14113 | 0.34417 |
| **Creativity** | 1/3 | 1/3 | 1 | 3 | 3 | 1.00000 | 0.16074 |
| **Cost** | 1/3 | 1/3 | 1/3 | 1 | 1 | 0.51728 | 0.08315 |
| **Simplicity** | 1/5 | 1/5 | 1/3 | 1 | 1 | 0.42168 | 0.06778 |

**Figure 1: Pairwise Matrix**

Below is the final decision matrix with the weight of each criteria taken into consideration. We can see from this matrix that the clock and autonomous rover were very close to one another, as both designs appeared to be easy to build and program. However once we took the weight of each criteria into consideration, our group came to the conclusion of using the autonomous rover.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Weight** | **Launcher** | **Digital Metronome/8-bit Music Player** | **Drone Simulator** | **Clock** | **Autonomous Rover** |
| **Buildability** | 0.34 | 1.38 | 0.34 | 0.69 | 1.72 | **1.03** |
| **Programmability** | 0.34 | 0.34 | 0.69 | 1.38 | 1.03 | **1.72** |
| **Creativity** | 0.16 | 0.16 | 0.80 | 0.48 | 0.32 | **0.64** |
| **Cost** | 0.08 | 0.42 | 0.08 | 0.17 | 0.33 | **0.25** |
| **Simplicity** | 0.07 | 0.27 | 0.07 | 0.14 | 0.34 | **0.20** |
| **Total** | 1.00 | 2.57 | 1.99 | 2.85 | 3.75 | **3.85** |

**FIgure 2: Decision Matrix**

**Background Research:**

Our team looked into several types of motors, including DC, stepper, and servo motors. These motors differ in their functionality, and therefore we had to do research on which would best fit our design’s needs. DC motors are built for faster rotations, such as in commercial fans and RC cars, and are continuous rotation models that are controlled by PWM (pulse width modulation).

To detect the presence of objects, our team knew we needed a motion detector or sensor of some sort. One that we researched was an ultrasonic sensor, such as the HC-SR04. This sensor works in the same principle as bats do with echolocation - sending out a wave and detecting based on the waves that are refracted and ‘bounced back’ to the receiver module of the sensor.

In order to generate noise for our rover, one buzzer type we researched was the piezo buzzer, which comes in most Arduino starter kits. There are two main types: active and passive buzzers, and they differ in how they are programmed and operated. In general, however, these require two pins: a power/input source, and GND.

To operate our rover, we needed wheels of some sort. To that end, we looked into using the MakerLab at Cline Library at NAU to 3D-print some wheels. In order to 3D-print using the MakerLab, several types of software can be used for the design - such as Tinkercad, SolidWorks, and AutoDesk. One thing we had to keep in mind is that the MakerLab will only accept designs with millimeters as the unit.

**Build and Test:**

*Describe the building and testing of the device. Document any changes made from the initial plans, explain the procedures including the mistakes or changes made along the way. At least two photographs are required. Additional photographs and sketches are encouraged and may be included in the matrix.*

*A one paragraph summary at the end of this section is required.*

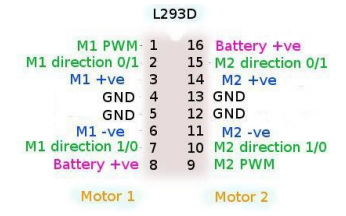
*(This is possibly the most important part of the report, a minimum of two pages is expected.)*

To begin the building phase, our team started by experimenting with each device we were going to use separately. This allowed us to figure out how each separate device worked. To begin experimenting, we started with the ultrasonic sensor. This sensor is the HC-SR04 ultrasonic ranging module. There are 4 pins on this sensor: VCC, TRIG, ECHO, and GND. The VCC and GND connections take input from the +5V and GND rails of the Arduino UNO. The TRIG and ECHO pins hook into the digital pins of the Arduino. Using the SR04.h library from Elegoo, we can use a helper method “.distance()” to read the distance in centimeters. This sensor has an accuracy of up to 400 centimeters. One thing our team noticed that if the distance was greater than 400 cm, the helper method would return 0. We took note of this as we would later need to take this into account when reading the distances on the rover.

After the ultrasonic sensor, our team tested out the active buzzer. The reason we chose the active buzzer instead of a passive buzzer was due to the fact that passive buzzers require an AC signal to generate a sound. Using the libraries given to us by Elegoo, the buzzer has a function called “tone(pin, integer)”. This allows us to generate tones using the active buzzer by calling this function and inputting our desired frequency in hertz.

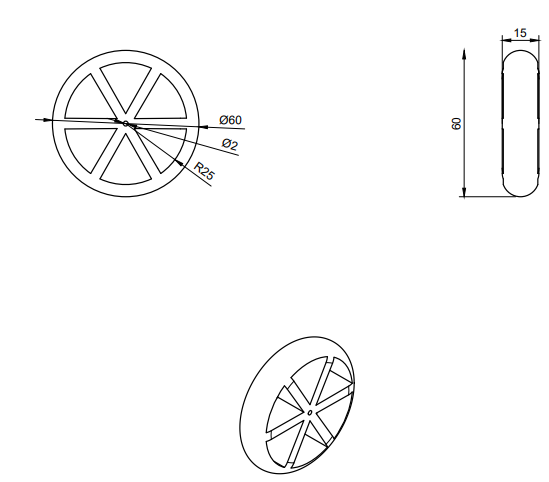
Our team decided to not experiment with the LEDs, as we all knew how they worked and were confident in using them. The next thing our team experimented with was the switches on the Arduino’s shield. The button has the name of SW0 and can be read using this.

The last thing our team experimented with was the L293D chip, DC motors, and an external power supply. In our case we started with a 9V battery powering the Elegoo power supply. The L293D motor driver allows us to hook up 2 motors per chip and move them forwards or backwards. On each side of the chip, there is an enable pin, direction pins, motor pins, ground pins, and a VCC pin. These chips also protect the Arduino from negative voltage as the DC motors are switched off.



**Figure 3: L293D Pinout Diagram**

From here our group was comfortable to start building the rover. Our group used a WB-108 breadboard from ‘CircuitSpecialists’ as the base of the rover. This breadboard contained 4 breadboard and several power rails and was plenty of space for our wiring. We then hot glued the DC motors to the bottom of the board. The DC motors we tried to use were the small 5V motors from the Elegoo kit. Later we would find that these motors would not be able to move the rover without being geared. We then started to design the wheels using AutoDesk Fusion. One of the group members has experience using 3D CAD and 3D printing with the MakerLab. We measured the diameter of the motor shaft using digital calipers and then produced the design.



**Figure 4: Wheel Schematic**

After assembling the rover each device was woking, however the motors we had were not getting enough current to push the rover forward. Our team believed this was due to the weight of the breadboard and swapped it for a much smaller unit. It was suggested to us that we ditch the 9 volt power supply and try out a 6 volt supply using AA batteries. With these two things combined, the rover was finally able to move with a gentle nudge, however the rover would eventually stop. We swapped the small DC motors in the Elegoo kit for ones with a bit more power behind them. With this swap we also needed to provide more voltage to the motors. We added a 9V battery in series with the 6 volt AA batteries to give us a combined 15 volts. From here the rover started to function properly and just needed a push to get it started. While testing the rover, the 9V battery died and was replaced with another 4 AA batteries. This gave us a total of 12V instead but increased the current for the motors. The rover now needed no external push to operate.

From this point on the rover functioned as intended but we still needed to calibrate the delays for when the rover detects an object. We had the rover back up, turn to the right, and then read the distance again. We ended up finding that the delay needed varied based on the surface the wheels were on and needed to take this into account. After testing different surfaces, we found backing up for 1.5 seconds, turning for 3 seconds, and waiting to read the distance for another 2 seconds yielded best results. Due to the sensitivity and pickiness of the ultrasonic sensor, the waiting time of 2 seconds was crucial to getting an accurate reading. With this all figured out, the rover was ready for demonstration.

**Demo Results:**

Our demo went as planned, however, we misunderstood what would be classified as a sensor. We used an ultrasonic sensor, a buzzer, and a button for our sensors and later found out that a buzzer does not count as a sensor. We also had slight issues with the traction of the wheels causing the device to cease movement and need a little push to start moving again. Given more time, we would revise our model to include the third sensor we were missing. This sensor would more than likely either be a photoresistor to detect the amount of light in the area, or a tilt switch that would turn off the rover if it tilted too much. To better improve our rover we could have redesigned the wheels so that they had better traction. We also could have added a another ultrasonic sensor to the back of our rover to ensure it would not hit objects while going backwards.

**Illustration of the Design Process:**

In this project, our team followed both the 4 and 10-step design processes. When we were presented with the initial project requirements and problem statement (problem statement = problem identification, research, and problem specifications), we first compiled research on basic parts we would need to use, such as different types of motors, ultrasonic sensors, and buzzers. Once we had done so, we entered the next phase (formulate solution/concept generation and design), and created more specific parameters for our design so that we had a clearer idea of what to build. We developed several concepts that we believed fit all the parameters, and weighed our options through the use of our pairwise and decision matrices. After choosing a concept, we drew out several rough schematics, and, when the team felt satisfied, began prototyping and programming the Arduino to work in conjunction with the physical circuit. As emphasized in the build and test section of this report, our team cycled through several designs and encountered difficulties, especially with the movement aspects of our rover. These problems resulted in steps of the overall process being pushed back and dates changed to reflect our progression, which can be seen in Appendix B. As such, we were forced to follow the cyclic nature of the design process and continue refactoring our initial design and program, and even conducting more research, until we reached the final product, which can be seen in Appendices D and E. A general explanation of both the schematic and Arduino program can be found in the flowcharts in Appendix C.

**Appendixes:**

1. *(A) Team development: Discuss the team’s evolution through the four stages team development – forming, storming, norming and performing.*
2. *(B) Gantt chart: Update and revise the team’s Gantt chart to accurately reflect the teams progress up through the report writing. Compare this to the original Gantt chart submitted on November 13. Fully explain any and all changes made to the Gantt chart. The Gantt chart may be displayed in “landscape” orientation if this will facilitate reading it.*
3. *(C) Flow Charts: a flow chart that is focused on the mechanical aspect of the project, a second flow chart for the Arduino program.*
4. *(D) Schematic – prepared electronically with correct circuit symbols, functionally correct. Use a box or a circle for the motors.*
5. *(E) Arduino code properly documented.*

**Appendix A: Team development**

*Team development: Discuss the team’s evolution through the four stages team development – forming, storming, norming and performing.*

When we all first got together during the forming stage, we came in groups of twos. Josh and Mason have worked together and Felecia and Taylor have worked together. So before we even started we already had underlying team chemistry. From the start, we were all comfortable in sharing ideas, delegating work, as well as getting work done. We meshed very well from the beginning with little to no problems.

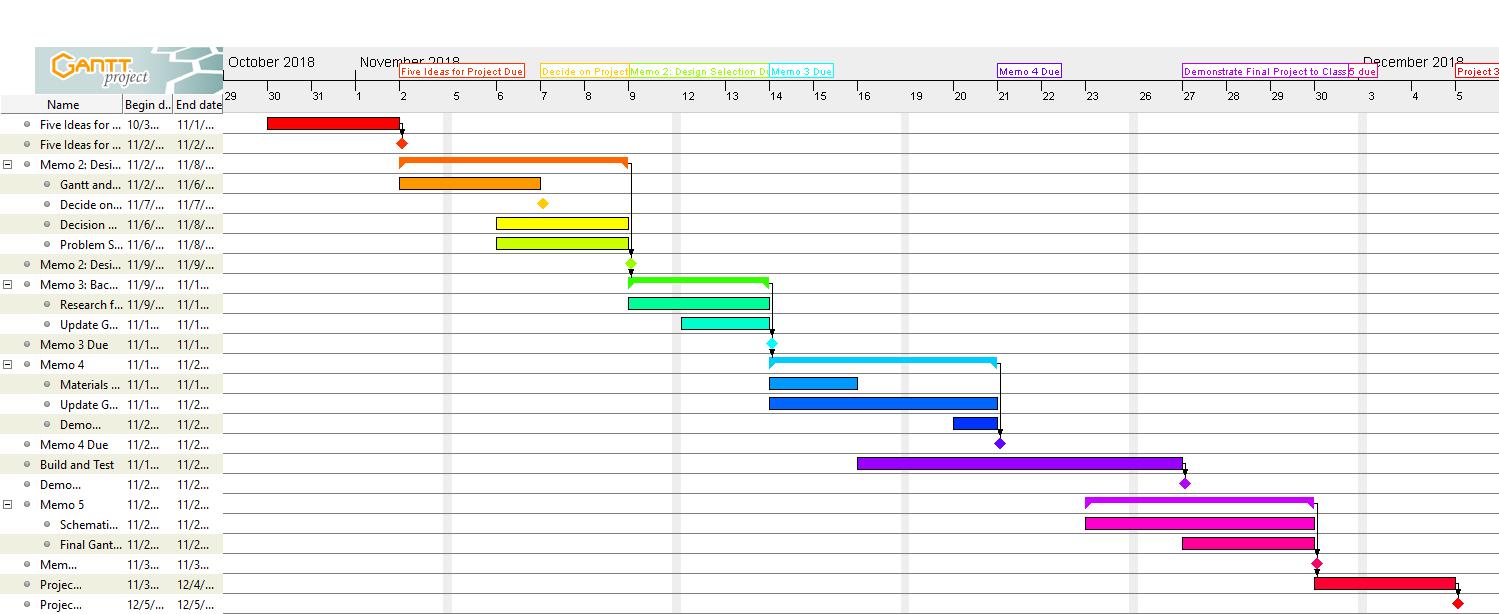
During the storming stage, this is where we truly got to see who had what kind of ideas for this project. We used the four ideas that were given to us in the beginning of the project and came up with the fifth one to be an autonomous rover. Both groups got more familiar with each other, then it slowly became easier and easier to work because we all had a common goal, do good on the project.

By the norming stage, we all knew what parts we played in this group. Josh, with his long experience with arduinos did most of the coding and building with help here and there from us. Then the rest of us made sure we did every deliverable since Josh was doing the circuitry. Josh was almost made into the team leader without us selecting him as such and we worked very well with the way the team has formed.

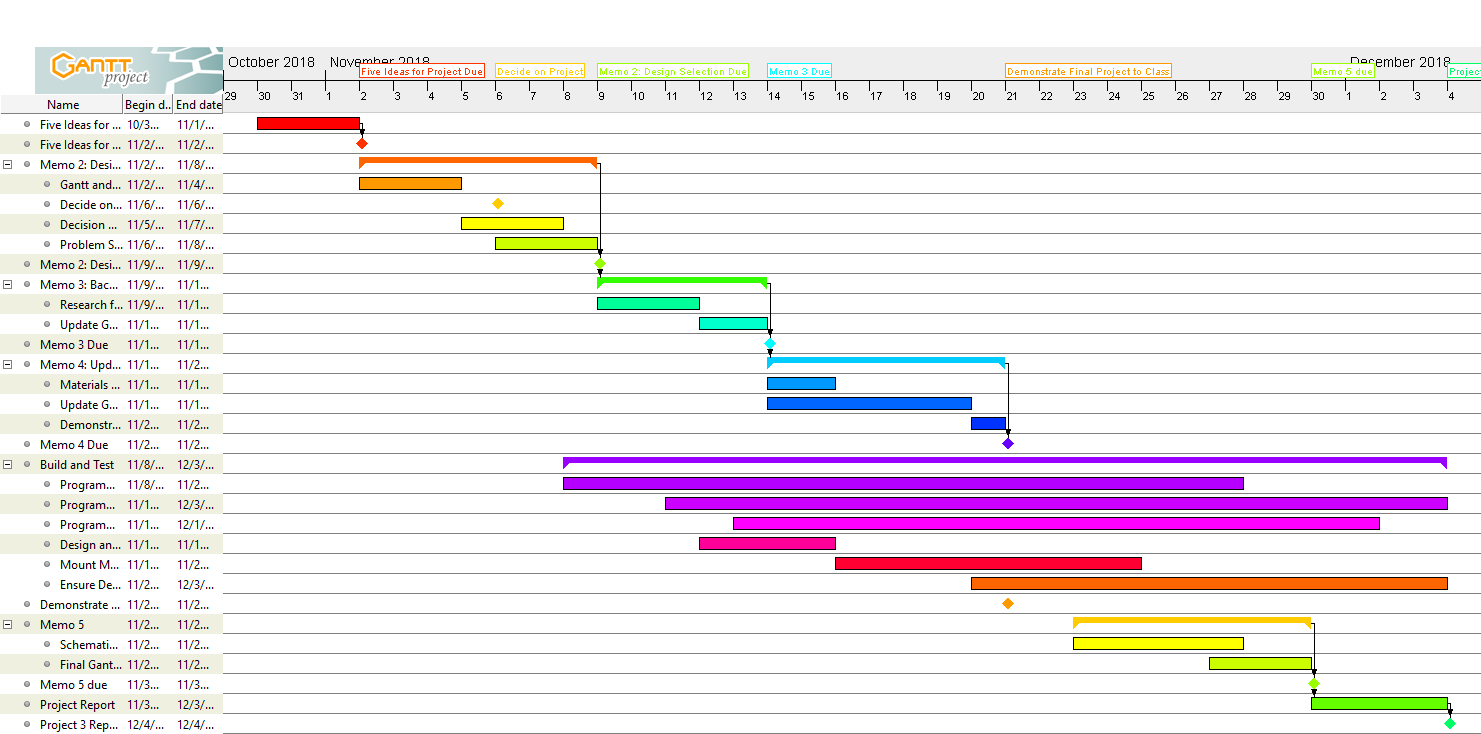
Lastly, the performing stage. With four brilliant minds coming together to get our autonomous rover to work, we ran into a handful of hiccups during this stage. We finalized and demonstrated our project to the class, in which we did very well. We missed one sensor because of a misunderstanding about the project, but we were easily able to overcome that. Overall, this team came together very well from the beginning and has been able to do work, not just well but also efficiently.

**Appendix B: Gantt Chart**

Several due dates throughout the project which caused our gantt chart to change. These due date changes allowed us more time to work on certain memos as well as building our device. We changed our decision of which project we would do to the 6th of November instead of the 7th of November so as to give ourselves more time to work on starting the project. Additionally, we changed our first gantt/WBS chart and decision matrix deadlines to earlier dates since we had completed them before we had expected. We moved our update gantt/WBS charts first update to start after the research section was finished so as to focus on one thing at a time. In addition to that, we had our second update for the gantt/WBS charts end a day earlier to allow more time to focus on making sure we were ready for our first demo. We extended our due date for the build and test section and various sections were added underneath it so as to have a more specific timeline for our building process. The sections underneath the build and test section each show the timeline of building and programming specific aspects our device. For example, one of these sections is building and programming the motors. We also changed our schematic deadline to an earlier date because we finished it earlier than anticipated. Lastly, we changed our project report to be due a day earlier than originally anticipated.

**

**FIgure 5: Original Gantt Chart**

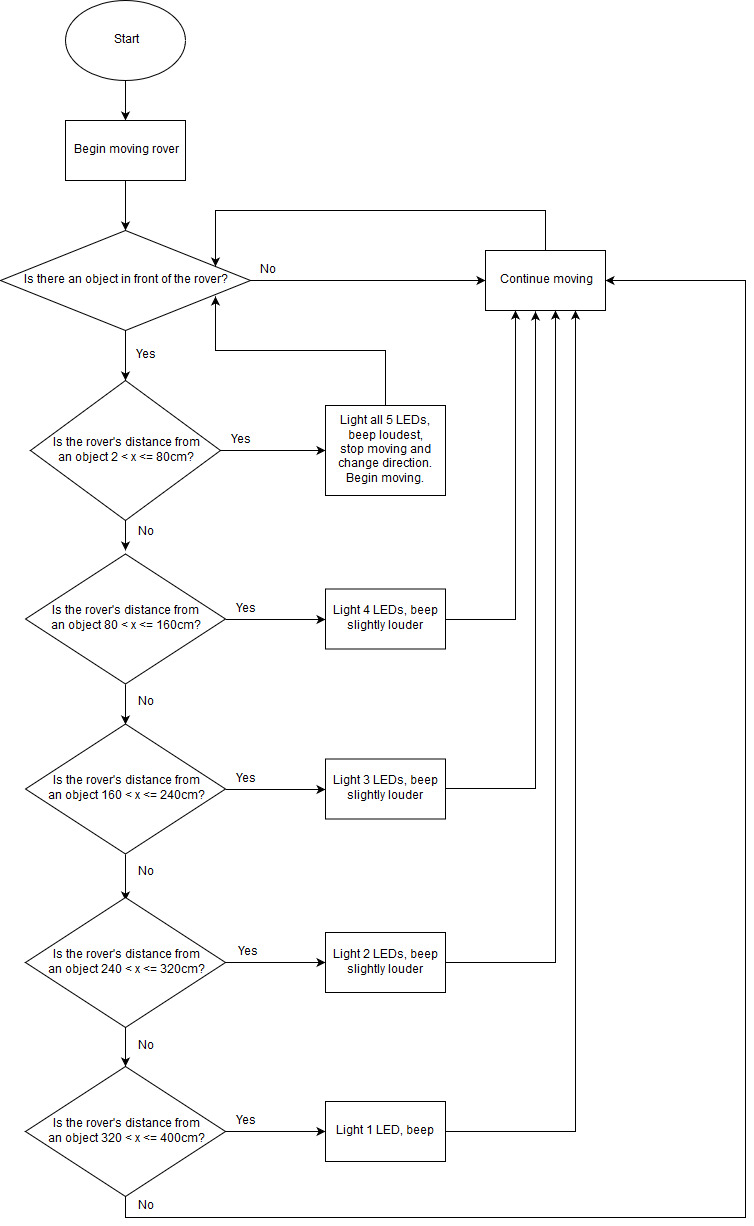


**Figure 6: Final Gantt Chart**

**Appendix C: Flowchart**

*Flow Charts: a flow chart that is focused on the mechanical aspect of the project, a second flow chart for the Arduino program.*

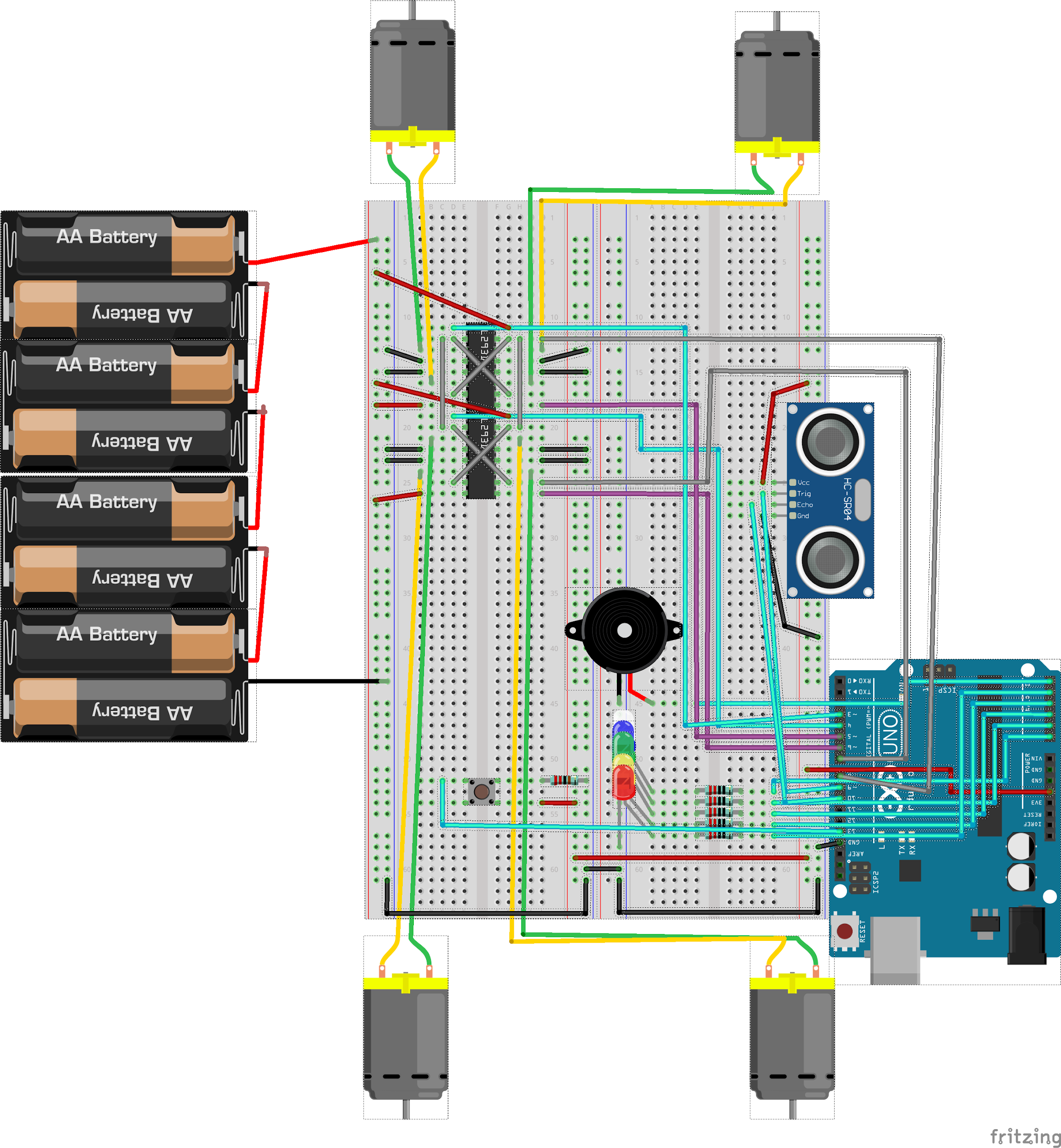
This appendix includes a flowchart that explains the mechanical functionality of our team’s automated rover. It explains how the rover physically functions, while also including some pseudocode to explain how the rover’s algorithm operates. Essentially, the rover will begin moving when the button is pushed, and continue moving forward until an object has been sensed, at which point the rover will back up and turn slightly, then begin moving forward again.



**Figure 7: Mechanical flowchart**

**Appendix D: Schematic**

*Schematic – prepared electronically with correct circuit symbols, functionally correct. Use a box or a circle for the motors.*

**

**Figure 8: Schematic**

**Appendix E: Arduino Code**

#include <SR04.h>

#define TRIG\_PIN 9

#define ECHO\_PIN 10

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Digital Pins

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

const int firstLED = 14, secondLED = 15, thirdLED = 16, fourthLED = 17, fifthLED = 18;

const int buzzer = 19, button = 13;

const int backLeft = 3, backRight = 4, frontLeft = 5, frontRight = 6;

const int forwards = 7, backwards = 8;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Defining Sensors

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SR04 ultraSonic = SR04(ECHO\_PIN, TRIG\_PIN);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Defining Variables

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double distance;

boolean pressed = false;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Defining Methods

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void objectDetected();

void startWheels();

void stopWheels();

void rangeHandle();

void rangeHelper(int ledCase);

void turnOffLED();

void buttonWait();

void setup()

{

pinMode(TRIG\_PIN, OUTPUT); // Sets the trigPin as an Output

pinMode(ECHO\_PIN, INPUT); // Sets the echoPin as an Input

for (int index = 3; index < 9; index++)

{

pinMode(index, OUTPUT);

}

for (int index = 14; index < 20; index++)

{

pinMode(index, OUTPUT);

}

pinMode(button, INPUT);

digitalWrite(forwards, HIGH);

digitalWrite(backwards, LOW);

}

void loop()

{

if (!pressed)

{

buttonWait();

}

while (pressed)

{

startWheels();

distance = ultraSonic.Distance(); // Distance in cm

rangeHandle();

if (distance <= 45 && distance > 2)

{

objectDetected();

}

}

}

void objectDetected()

{

stopWheels();

digitalWrite(forwards, LOW);

digitalWrite(backwards, HIGH);

startWheels();

delay(1500);

stopWheels();

while (distance <= 45 && distance > 2)

{

digitalWrite(backRight, HIGH);

digitalWrite(frontRight, HIGH);

delay(3000);

stopWheels();

delay(2000);

distance = ultraSonic.Distance();

}

digitalWrite(forwards, HIGH);

digitalWrite(backwards, LOW);

}

void startWheels()

{

digitalWrite(frontRight, HIGH);

digitalWrite(frontLeft, HIGH);

digitalWrite(backRight, HIGH);

digitalWrite(backLeft, HIGH);

}

void stopWheels()

{

digitalWrite(frontRight, LOW);

digitalWrite(frontLeft, LOW);

digitalWrite(backRight, LOW);

digitalWrite(backLeft, LOW);

}

void rangeHandle()

{

// 2cm to 400cm

if (distance < 2 || distance > 350)

{

rangeHelper(0);

}

else if (distance > 320 && distance <= 400)

{

rangeHelper(1);

}

else if (distance > 240 && distance <= 320)

{

rangeHelper(2);

}

else if (distance > 160 && distance <= 240)

{

rangeHelper(3);

}

else if (distance > 80 && distance <= 160)

{

rangeHelper(4);

}

else if (distance <= 80 && distance > 2)

{

rangeHelper(5);

}

}

void rangeHelper(int ledCase)

{

turnOffLED();

switch (ledCase)

{

case 0:

turnOffLED();

noTone(buzzer);

case 1:

digitalWrite(firstLED, HIGH);

tone(buzzer, 100);

break;

case 2:

digitalWrite(firstLED, HIGH);

digitalWrite(secondLED, HIGH);

tone(buzzer, 200);

break;

case 3:

digitalWrite(firstLED, HIGH);

digitalWrite(secondLED, HIGH);

digitalWrite(thirdLED, HIGH);

tone(buzzer, 400);

break;

case 4:

digitalWrite(firstLED, HIGH);

digitalWrite(secondLED, HIGH);

digitalWrite(thirdLED, HIGH);

digitalWrite(fourthLED, HIGH);

tone(buzzer, 600);

break;

case 5:

digitalWrite(firstLED, HIGH);

digitalWrite(secondLED, HIGH);

digitalWrite(thirdLED, HIGH);

digitalWrite(fourthLED, HIGH);

digitalWrite(fifthLED, HIGH);

tone(buzzer, 800);

break;

}

}

void turnOffLED()

{

digitalWrite(firstLED, 0);

digitalWrite(secondLED, 0);

digitalWrite(thirdLED, 0);

digitalWrite(fourthLED, 0);

digitalWrite(fifthLED, 0);

}

void buttonWait()

{

int buttonState = 0;

while (true)

{

buttonState = digitalRead(button);

if (buttonState == HIGH)

{

delay(500);

buttonState = digitalRead(button);

if (buttonState == HIGH)

{

pressed = true;

return;

}

}

}

}