Imperial College London

ELEC40006 Electronics Design Project 1

EEEROVER

Submitted 19th June 2022
Word Count: 9656

Pantelis Georgiou Tutor Group
Anish Narain, 02045150
Awais Khawaja, 02022071
Hasnat Chowdhury, 02038079
James Taylor, 02044403
Jamie Turner, 02017907
Oliver Liu, 02038286
Yasser Hassan, 02023619

Contents

Abstract

The aim of this project was to design a remote controlled rover that can explore a lunar terrain and identify rocks based on the signals they emit. This involved splitting up the project into two parts: motion and sensing.

For motion, the rover required motors to drive wheels that enabled forwards, backwards, left, and right movement. Furthermore, the rover's movements were to be controlled over Wi-Fi. This required using a Wi-Fi module with a Metro M0 microcontroller Board and creating the interface to communicate with the board. The chosen user-interface was an Android application built on MIT App Inventor. The buttons on the app enabled the rover to move in the four directions mentioned above. It also included four additional "orbit" buttons, a scan button, and textboxes for the results of the scan.

For sensing, the rover needed to detect infrared, radio, ultrasound, and magnetic signals. For infrared, a pre-built infrared "flame" sensor was incorporated into an amplifier circuit to detect the relevant 353Hz and 571Hz infrared pulses. For radio waves, an inductor in parallel with a capacitor was used to receive the modulated radio waves. An envelope detector was used to extract the modulating signal, and this was fed through a comparator to amplify and clean up the signal. This enabled detection of radio waves and allowed the identification of their modulated frequencies. The ultrasound waves were detected by taking an ultrasound transmitter/receiver, de-soldering the transmitter component and only using the receiver functionality. Finally, two hall effect sensors incorporated into a differential amplifier circuit were used to detect static magnetic fields and determine their polarity.

Once motor controls and the sensor circuitry were functioning, a chassis was built to incorporate all the hardware onto one structure. Finally, the code for the motors and the sensing circuits were combined. All the components were fine-tuned to remove bugs and the project was complete.

Design

Problem Design Specification

Key consideration	Requirements
Reliability	• Need each sensor to work every time it is tested • Need the outcome of each sensor to be accurate every time it is tested • Need the movement controls to work every time they are used through an app • Need chassis to not break/ fall apart at any time • Need app to be able to read information sent from rover reliably • The correct rock should be identified 100% of the time
Safety	• Ensure metro board is not being overloaded and that all its inputs are limited to 3.3V as required • Make sure that all the nuts and bolts are fastened tightly so that they do not drop and cause the chassis to break exposing broken shards • Ensure workspace is tidy and that no components are near the edge of the bench to try and prevent anything falling onto the floor and leading to trips/ falls • When using soldering iron, hot surfaces are not touched and try not to breathe in the toxic fumes • When testing movement make sure surrounding area is clear so that the rover doesn't bump into anyone, causing them to trip and fall • Ensure that no food or drink are consumed near any circuitry so that in the unlikely event of a spill nothing gets broken
Performance	• Be able to control the rover and manoeuvre it around rocks and obstacles • Be able to detect each signal and therefore identify the rock • Be able to communicate with rover through app to tell us what type of rock we are scanning • Maximum scan time should be less than 5s • Movement speed should be greater than $0.1ms^{-1}$
Cost	• Extra products/sensors must not exceed £60 • For more detail see budget section
Disposal and Sustainability	• Acrylic used will be able to be recycled • Will safely disposes of batteries • Sensors/metro board/breadboard/ motors can all be recycled and used in another project • Quick and easy disassembly due to minimal soldering
Testing	• Test range of sensors • Test range of motion on rover • Test speed of transmission to app

	• Test durability • Test portability • Check that the code is robust and efficient • Test overall system at least 20 times
Materials	• Durable • Low cost • Readily available • Parts in contact with heat sources (e.g., Motor contacts) should be able to withstand heat • Shouldn't interfere with sensors • Should be able to withstand the extreme temperatures on the moon • Should be relatively light
Ergonomics	• Controller to move rover must be comfortable for the person using it
Size and weight	• Per the brief it must be lightweight, <750g • Must be able to manoeuvre smoothly and efficiently around rocks/obstacles • Reduce weight to improve movement speed, increasing efficiency of the rover.
Environment	• Resilient to obstacles/terrain • Resilient to external signals/light pollution • Must be stable enough to bump into other rovers or rocks and continue operation
Aesthetics/ appearance/finish	• Not a priority • Can not interfere with any of our sensors • The more unique it is the easier it will be to identify it on the track

Sensor design

Mineral	Property 1	Property 2
Gaborium	61kHz radio modulated at 151Hz	Acoustic signal at 40.0kHz
Lathwaite	61kHz radio modulated at 239Hz	None
Adamantine	89kHz radio modulated at 151Hz	Magnetic field up
Xirang	89kHz radio modulated at 239Hz	Magnetic field down
Thiotimoline	Infrared pulses at 353Hz	None
Netherite	Infrared pulses at 571Hz	Acoustic signal at 40.0kHz

The first part that was designed was the sensor circuits for identifying the different rock types. The table above shows the possible signal types and properties that rocks can emit. The section below documents the iterative development process for each sensor in greater detail.

Magnetic Sensor

To detect the magnetic field, a 49E Hall-Effect sensor was used. It is operated by a magnetic field and the output voltage from the sensor depends on the supply voltage and varies proportionally to the strength of the magnetic field. When there is no outside magnetic field, the quiescent output voltage is half the supply voltage.

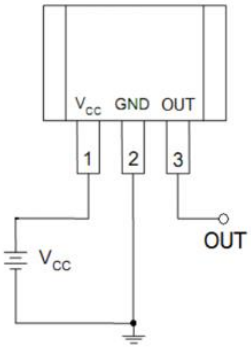


Figure 1 from [3]

When a south magnetic pole approached the flat face of the sensor, the circuit drove the output voltage higher than the quiescent value. On the other hand, a north magnetic pole drove the output lower.

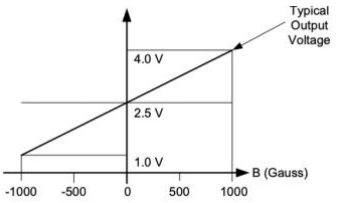
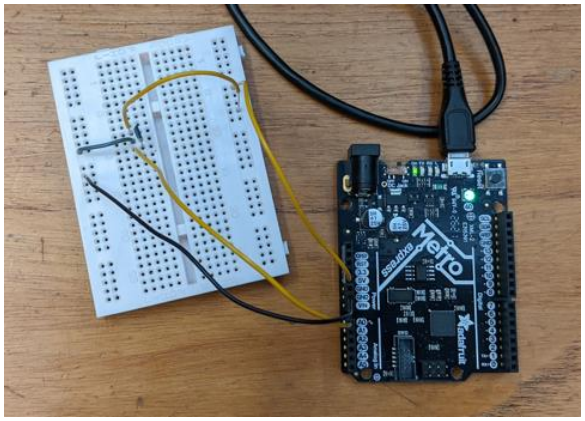
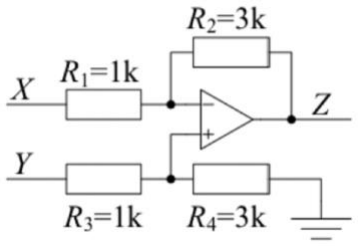


Figure 2 from [4]

The setup in the diagram below was created, and the code below programmed onto the Metro M0 board. The 800 and 600 values in the code comes from voltage input in mV as a ratio of the limit of 3300 mV. This gave a quiescent output value of around 760 when read by the microcontroller. This was later fine-tuned through reading what values corresponded to which pole.



```
#define Hall_Sensor A0          //A0 used with analog output, D2 with digital output

double Val1 = 0;              //Here you can store both values, the Val2 can be boolean

void setup() {

  pinMode(Hall_Sensor,INPUT);
  Serial.begin(9600);

}

void loop() {

   Val1=analogRead(Hall_Sensor);           //We read both values and display them raw on the serial mon
   //Serial.println(Val1);
   //Serial.print("\t");
   if (Val1 > 800){
    Serial.println("north");
   }
   else if (Val1 < 600){
    Serial.println("south");
   }
   else{
    Serial.println("0");
   }
}
```

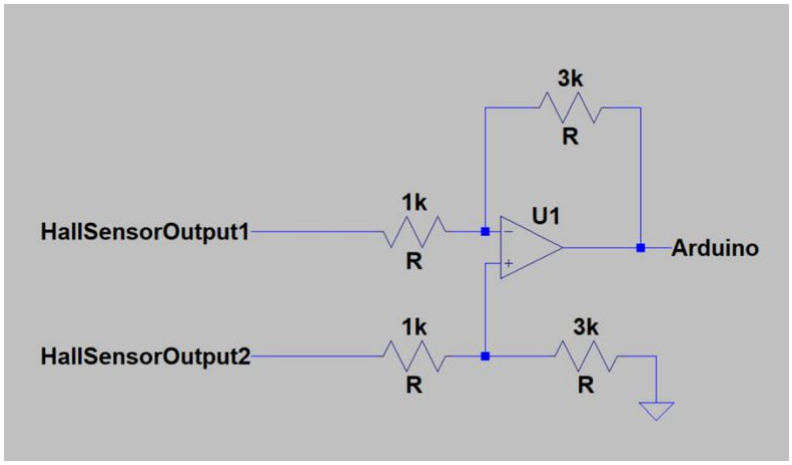


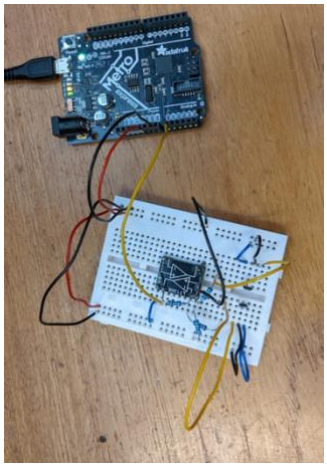
X and Y were outputs of the 2 hall sensors. This circuit gave a gain of 3. Figure from [5]

However, the range of this system was too short. Therefore, the output was amplified so a lower magnetic field change was needed for the same change in voltage output and therefore the magnet would not need to be as close to the sensor. However, the hall sensor was already biased at 2.5V and with the Metro board having an input voltage limit of 3.3V, the possible gain was very low (~1.3). To fix this another hall sensor was added. The difference in the output of the two sensors was then amplified using a differential amplifier set up.

Since the sensors were not perfectly matched, the quiescent difference in outputs was not exactly 2.5V and there was a slight output voltage. However, the overall quiescent difference was much lower than 2.5V making it possible to amplify the overall output with a gain of up to 5.

Thus, with the circuit below, it was possible to achieve the aim of identifying up and down magnetic poles. When the north pole faced the sensor the output voltage dropped, when the south pole faced the sensor the output voltage rose.

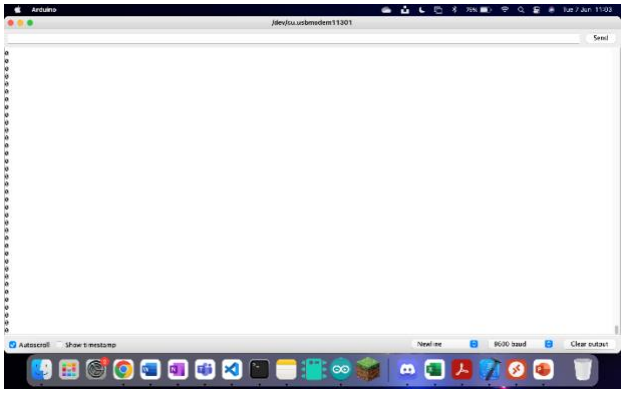


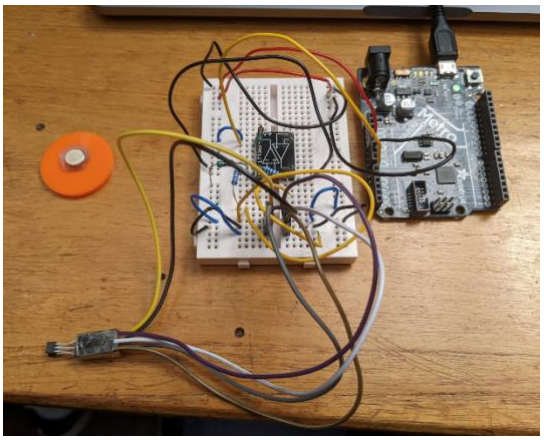


Later, the 3k resistors were changed to 5k resistors to further increase the range.

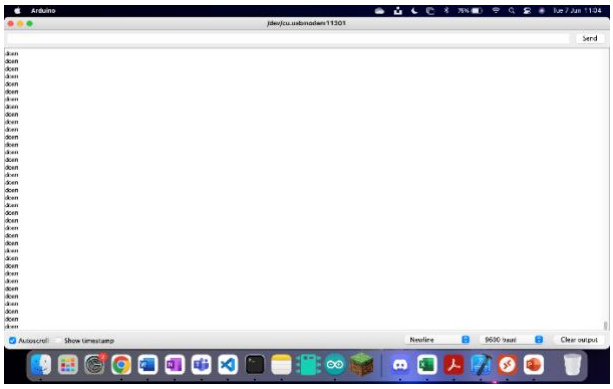
Demonstration

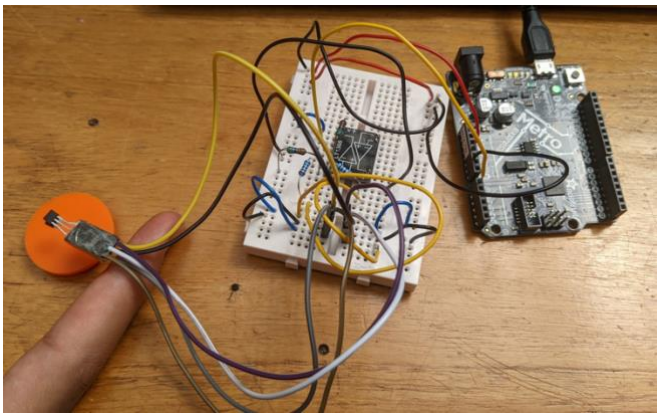
No field



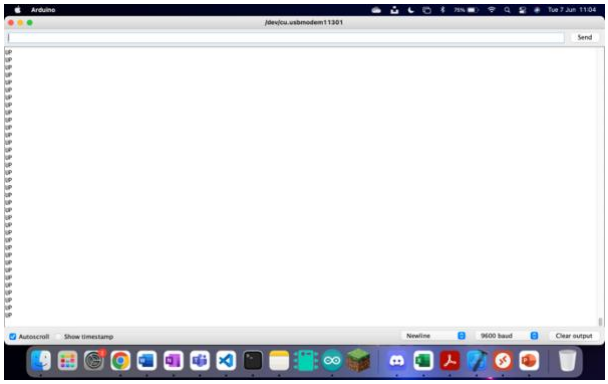


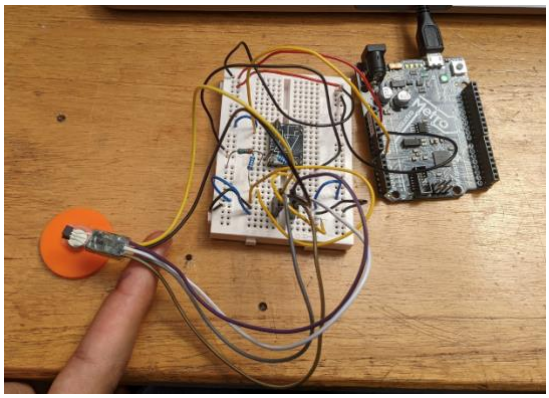
Down Detected





Up Detected





Ultrasonic Sensor

A HC-SR04 ultrasonic range finder was used for the detector.

The HC-SR04 consists of two ultrasonic transducers. One that acts as a transmitter and converts electrical signal into 40kHz ultrasonic sound pulses. The other transducer receives the transmitted pulses. If the receiver obtains a pulse, it produces an output pulse whose width can be used to determine the distance the pulse travelled.

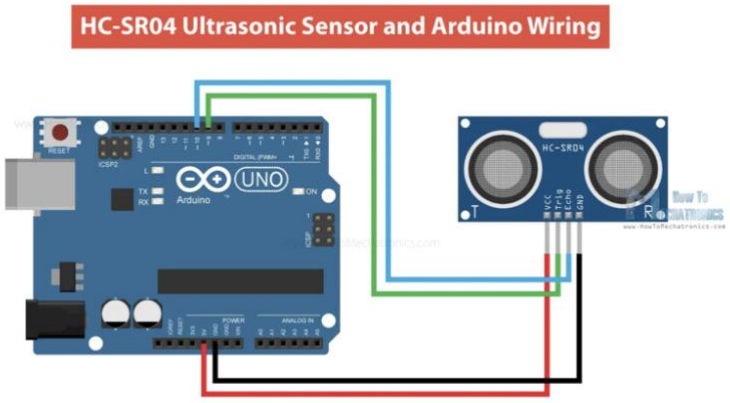


Figure 3 from [6]

Since this sensor operates with acoustic waves, it was utilized in the initial design, even though its distance-finding ability was unnecessary.

The code below, taken from [6] was used to determine the output of the echo pin.

```
sketch_may19a §
/*
  Ultrasonic Sensor HC-SR04 and Arduino Tutorial

  by Dejan Nedelkovski,
  www.HowToMechatronics.com

*/
// defines pins numbers
const int trigPin = 9;
const int echoPin = 10;
// defines variables
//long duration;
//int distance;
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
}
void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  //delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  //digitalWrite(trigPin, HIGH);
  //delayMicroseconds(10);
  //digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  //duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  //distance = duration * 0.034 / 2;
  // Prints the distance on the Serial Monitor
  //Serial.print("Distance: ");
  //Serial.println(distance);
  if (digitalRead (echoPin == 1){
    Serial.print("ultrasound detected");
  }
}
```

The initial plan worked on the presumption that the output of the receiver pin would go high when any ultrasonic 40 kHz wave is detected, however this did not work as expected.

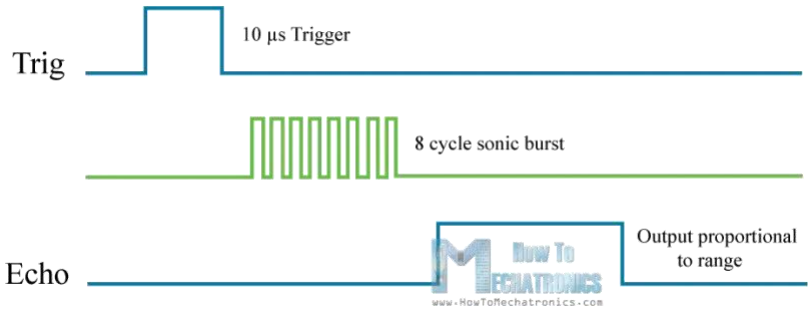


Figure 11 from [6]

The timing diagram above demonstrates the operation of the ultrasonic sensor module. The transmitter produced an ultrasound wave when the 'Trig' pin of the sensor was set to high. This sent out an 8-cycle ultrasonic burst. As soon as the this was transmitted, the 'Echo' pin went high which caused the receiver to start waiting for the wave to be reflected from object. Once the ultrasound was received, the echo pin went low. The duration that the Echo pin was high was used to find the distance from the sensor to the object. Therefore, the circuit would not start "sensing" until it received a high pulse on the trigger pin.[7]

The ultrasonic transmitter was de-soldered. This would "trick" the sensor into thinking it was sending out a signal and that the signal coming back to the echo pin from the rock is the same one emitted from the transmitter. Another approach would be to cover up the transmitter with tape, but this was not used for various reasons, mainly size, weight, and robustness.

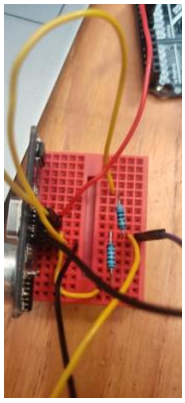
With these modifications, the echo pulse would be the length of the trigger delay if no ultrasonic wave is received, and it would be extremely short if it does receive one.

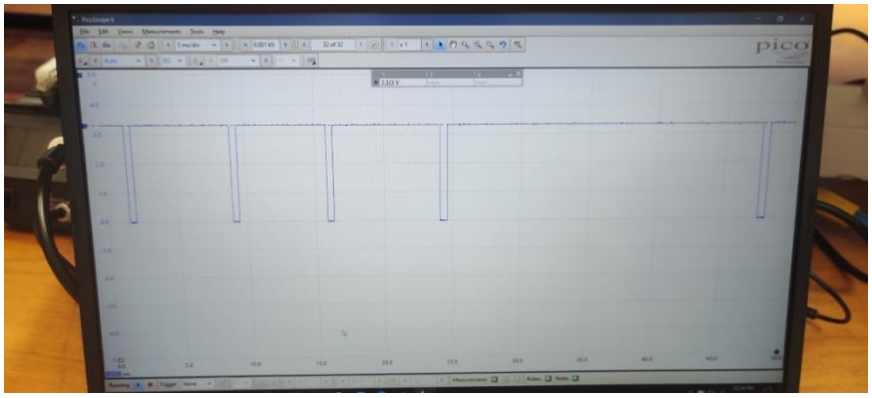
```
duration = pulseIn(echoPin, HIGH);
if(duration<3500){
  Serial.println("detected ultrasonic");
}
else if(duration>7000){
  Serial.println("not detected");
}
```

The code above measured the duration of the echo pulse in milliseconds.

Voltage Divider

One major issue was that the sensor operates at 5V. This means it required 5V to power it and outputs roughly 5V when it received an ultrasonic wave. The Metro M0 board however operated at 3.3V so could not take a 5V input without damaging the board. This was solved using a potential divider. A 1.5k and 3k resistor were used to take the output from the ultrasonic sensor and split the voltage so that only around 3.3V was present at the GPIO pin.

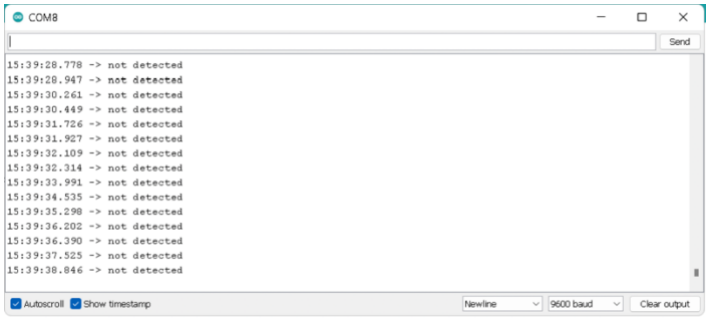


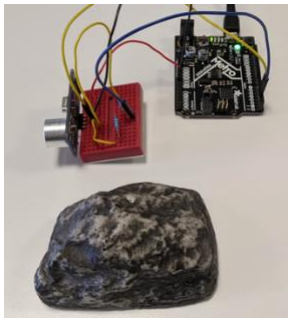


The oscilloscope output shows the value at around 3.3V, sufficient to ensure no damage to the board.

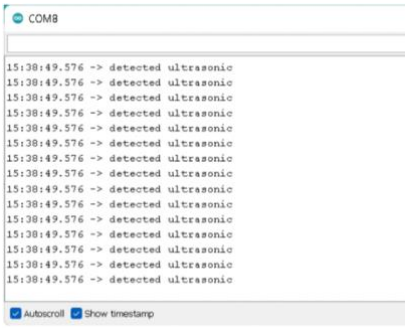
Demonstration

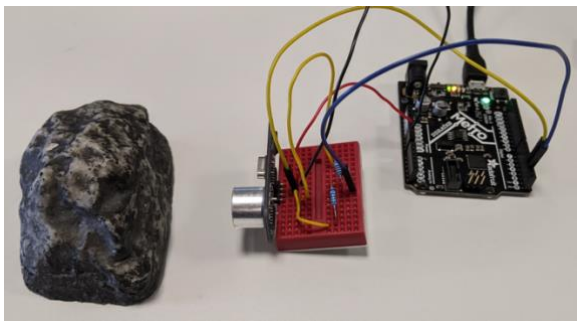
No Ultrasound Detection





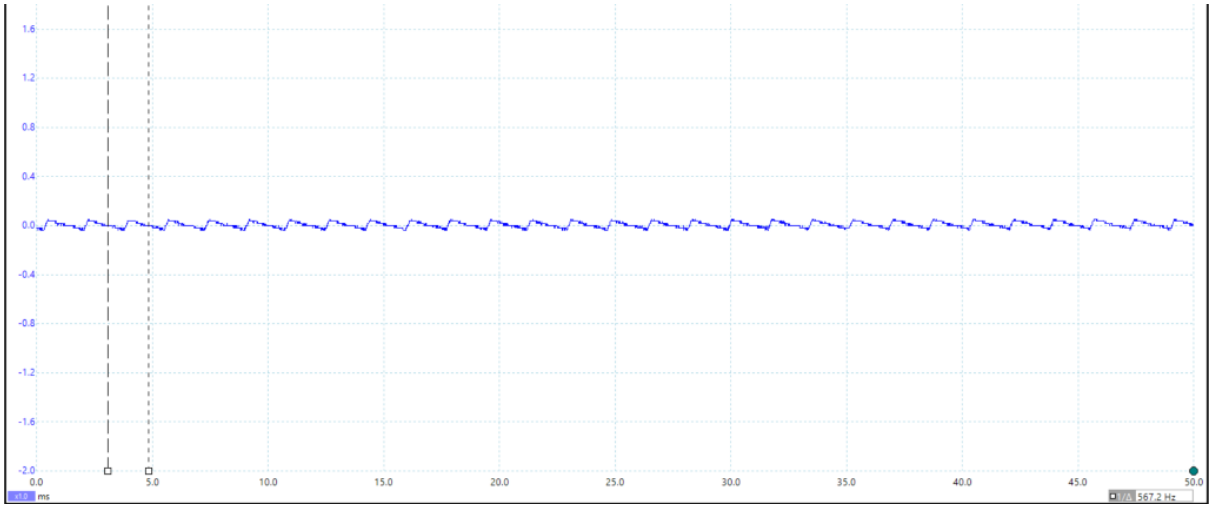
Detecting Ultrasonic





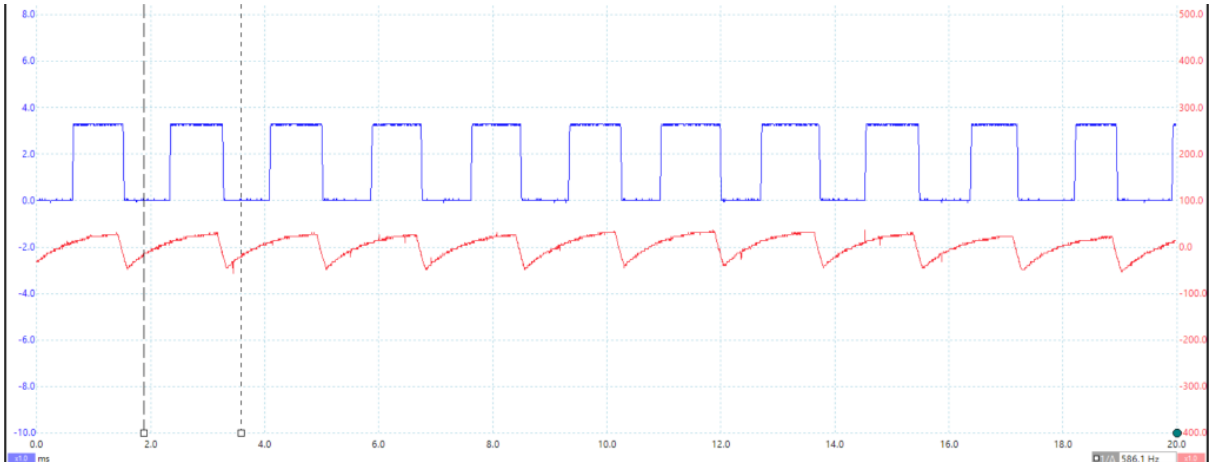
Infrared Sensor

The initial design utilised a SFH300-3/4 phototransistor. To remove the constant DC voltage generated due to background light, a 1 µF capacitor was added between the output of the phototransistor and the output of the circuit. When brought near the rock, the waveform shown below was observed on an oscilloscope.



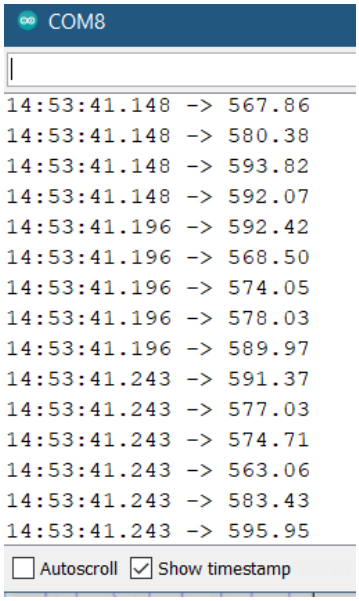
The approximate frequency was measured as 570 Hz, with the expected value being 571 Hz. This waveform was then passed through a 100x amplifier cascaded into a 300x amplifier to reach a voltage level readable by the Arduino. However, it was discovered that the signal was still too weak and noisy to be reliably read so a different approach was used.

Next, a KY-026 IR sensor was tried as the filter over the phototransistor blocks out visible light, resulting in a cleaner signal. A 1 µF capacitor was once again used to remove any DC bias before the signal was passed through a 300x non-inverting amplifier. The output of the IR sensor is shown below in red and the output of the amplifier in blue.



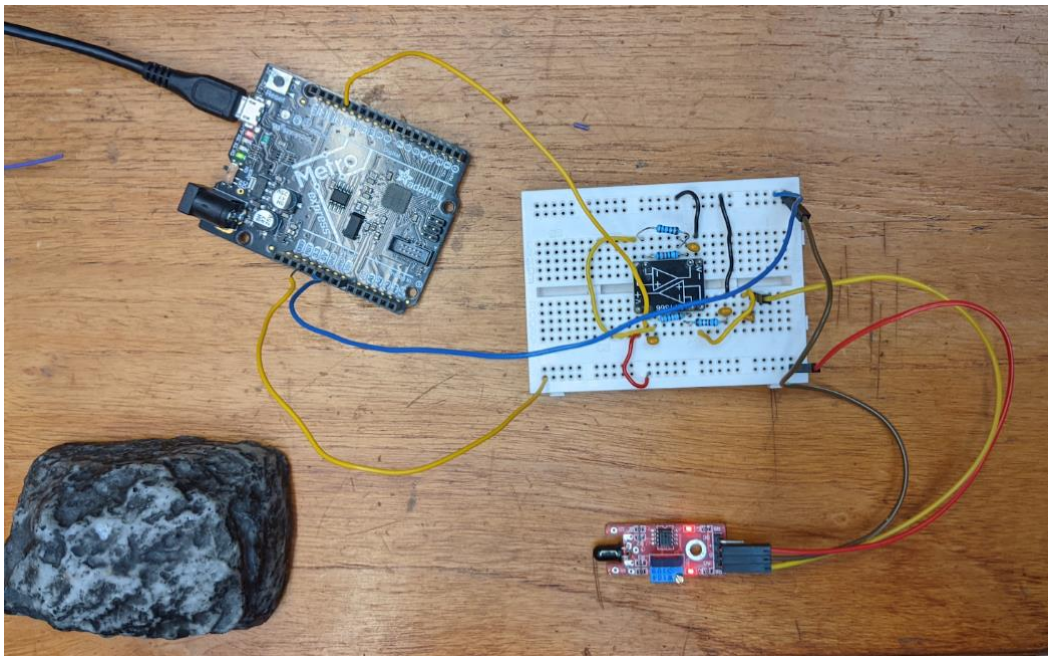
A clear square wave was visible, and ready to be passed to a digital input pin on the microcontroller. The Metro M0 board measured the time that the input is low and then high. The sum of these two values is therefore the time period of the square wave signal. Finding the reciprocal of this value yields the frequency of the wave.

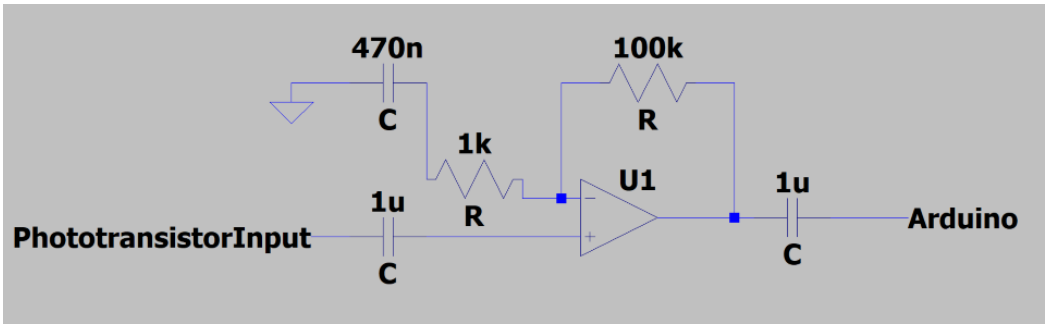
The system was tested with the Exorock emitting infrared pulses at a frequency of 571 Hz. The output of the serial monitor showing the measured frequencies is shown below.



Almost all values were within 30 Hz of the actual frequency. A system was then implemented that prints one of the two possible frequencies (either 571 Hz or 353 Hz) if the measured frequency was within a certain tolerance. Unfortunately, the presence of noise at slightly further distances made the system less reliable. To solve this, a modified program was written that takes ten separate samples of the frequency. It will only output a value if at least half of the samples are within a certain tolerance of the possible frequencies. If less than half of the samples agree with each other, or more than half of the samples are outside of the tolerance, the value -1 will be returned, indicating there is no detectable infrared signals being emitted.

The image and diagram below show the final system used to detect infrared.

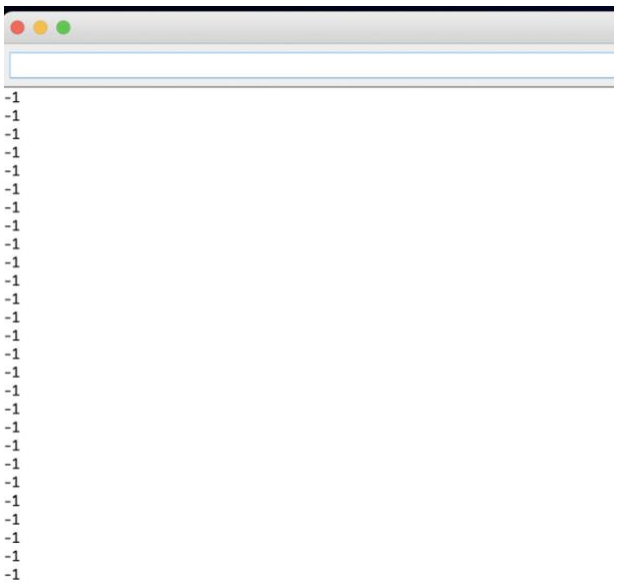


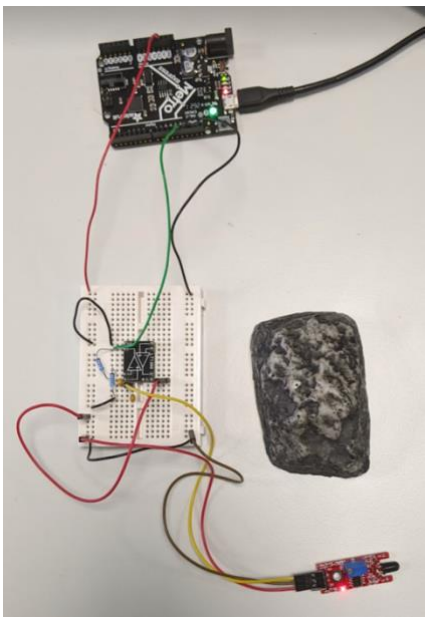


The final system was tested with both possible frequencies. At close range (distances less than 3 cm), the Metro board identified the correct frequency every single time. The further the sensor moved from the rock, the fewer times it would correctly identify the frequency. Any times it didn't, the function would return "Unknown". However, unlike the previous system which only took a single sample, this new system never returned the complete opposite frequency value, meaning it is not susceptible to noise and will not predict the incorrect rock type. It was found that removing the final capacitor improves the reliability of the system as the signal is not shifted down to the point that half of it is negative.

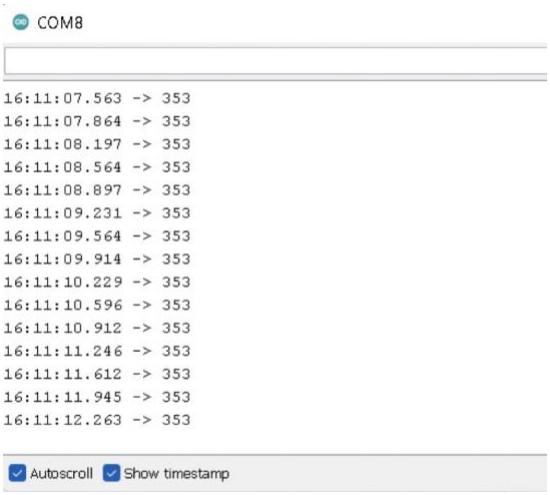
Demonstration

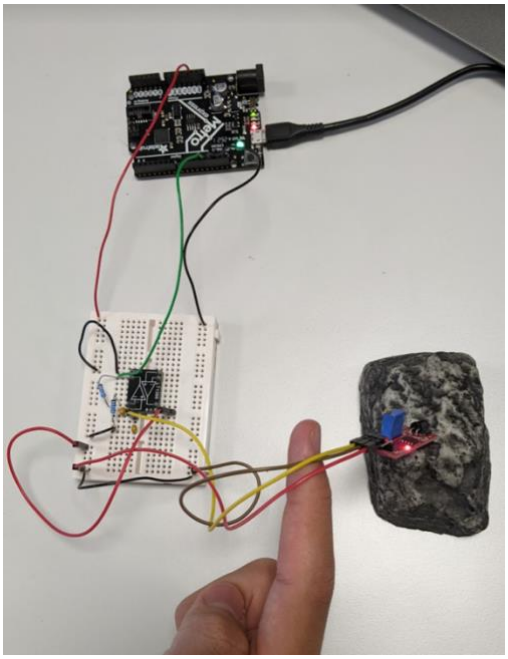
No Infrared



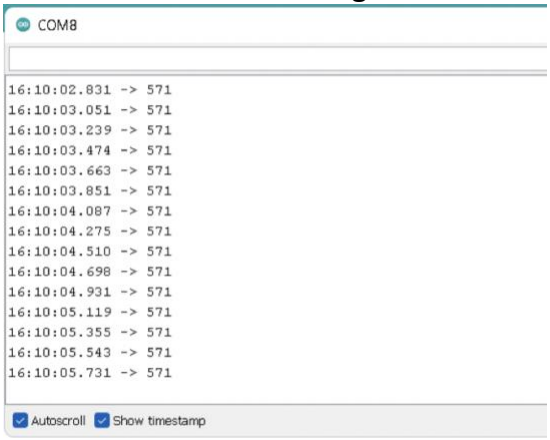


Detection for 353 Hz Configuration



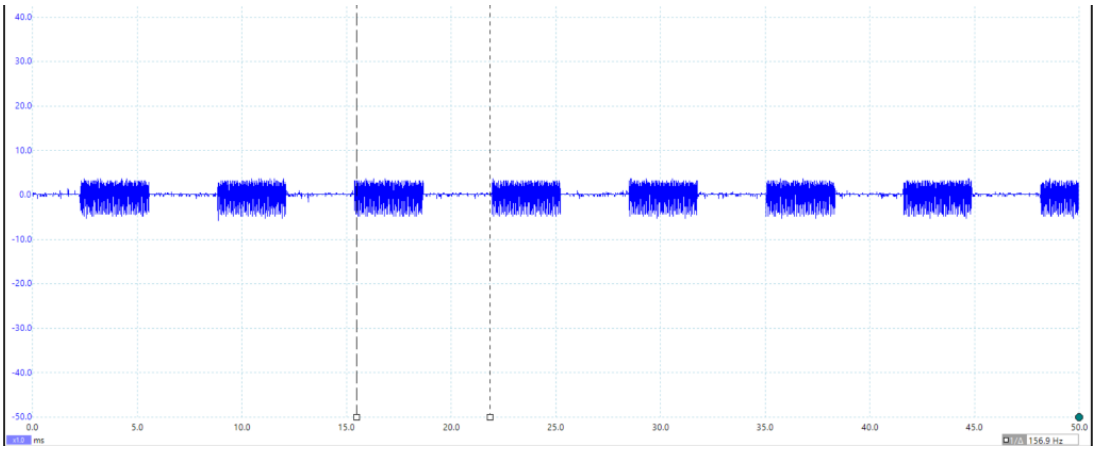


Detection for 571 Hz Configuration

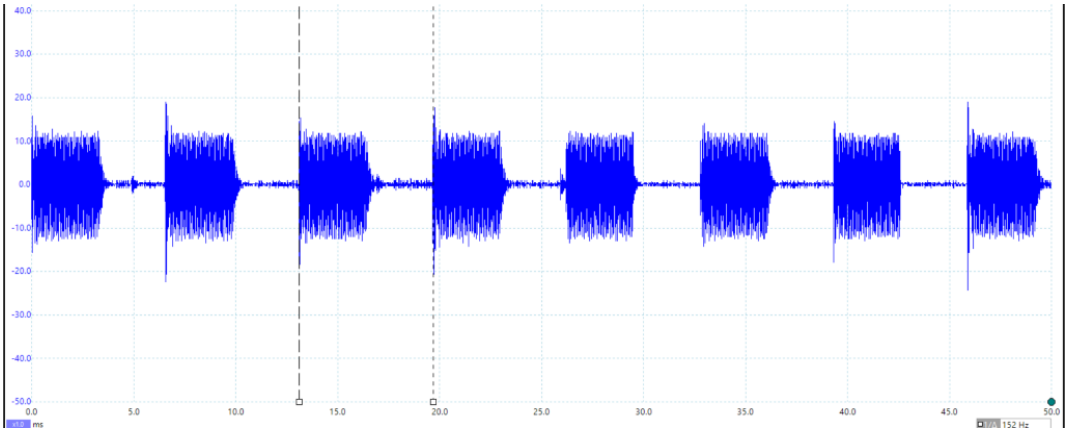


Radio Sensor

A coil antenna was created by winding insulated wire in a circular shape. The antenna, shown below, was connected to the probes of an oscilloscope, and placed on top of an Exorock emitting a radio signal. The output waveform is also shown below.

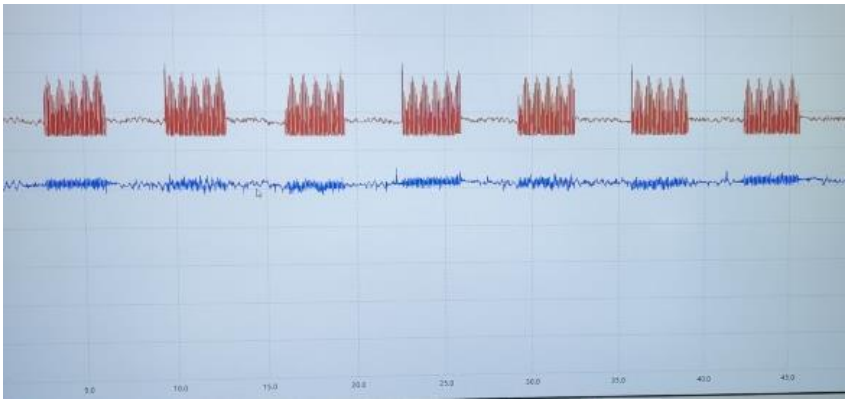


The modulated waveform was clearly visible, and the oscilloscope measured the frequency of the modulating signal to be 157 Hz, with the actual emitted frequency being 151 Hz. A 100 mH inductor was then used in place of the coil of wire, as the higher number of turns in the inductor should result in a stronger signal. The resulting waveform, with the same setup as the coil antenna, is shown below.

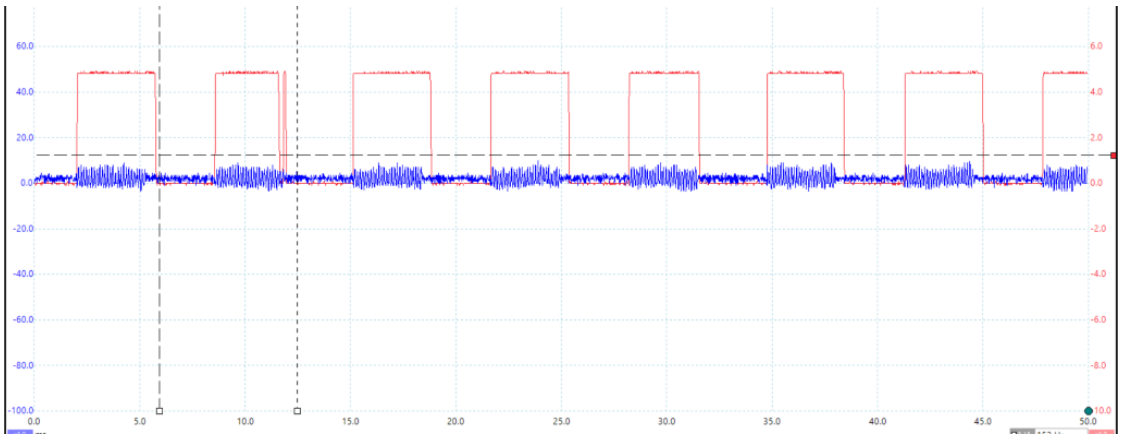


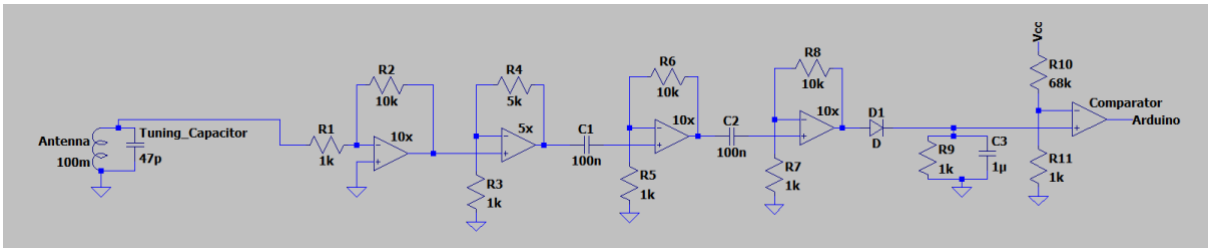
The peaks were at a significantly higher voltage, meaning less amplification was required to be readable by the microcontroller. A resonant LC network was created by adding a 47 pF capacitor in parallel with the inductor. This value was chosen so that the circuit has a resonant frequency of 75 kHz, in between the two possible carrier frequencies. This had the effect of reducing noise since any frequency away from 75 kHz would be attenuated.

The first amplification circuit used four cascaded amplifiers to reach a peak amplitude of greater than 0.7 V. This was necessary as the demodulator used a diode which would only be forward biased if the input signal reached 0.7 V. The stages were capacitively coupled to remove any DC voltage which should not have been amplified. The amplification section provided a total of 5000x amplification. It was necessary to use four separate operational amplifiers as the low gain-bandwidth product [8], combined with the high frequency of the carrier signal, meant that the signal was distorted above a certain individual gain. The output of the amplification stage is shown below with the blue signal as the signal picked up by the antenna and the red signal as the output. The waveforms have different vertical scales, so the output has a significantly greater peak voltage.

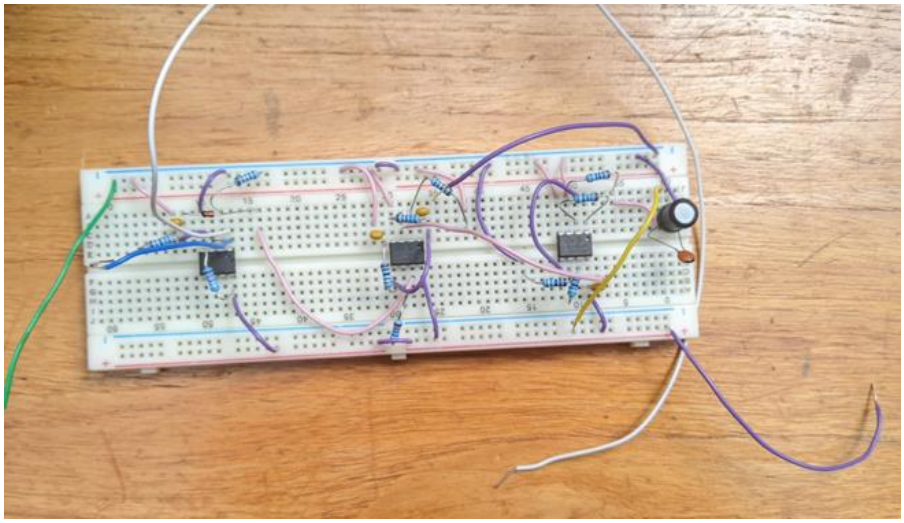


An envelope detector was used to extract the modulating signal. This resulted in a slightly distorted square wave. To clean the signal, the square wave was passed into the non-inverting input of an operational amplifier. The inverting input was connected to a potential divider to hold it at 50 mV. This system acted as a comparator, with the output swinging between either 0V or the power supply (3.3 V), forming a near perfect square wave, shown below with the circuit diagram.

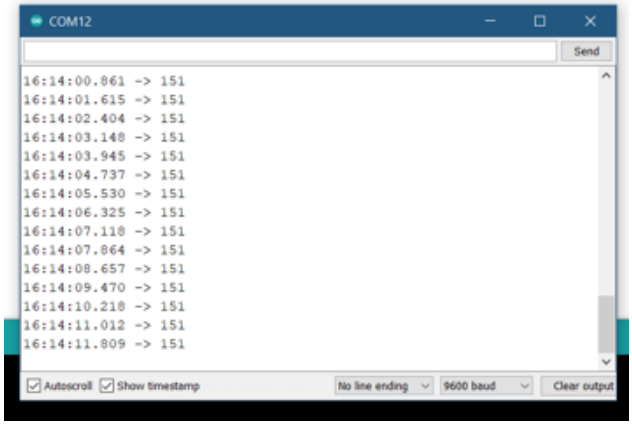




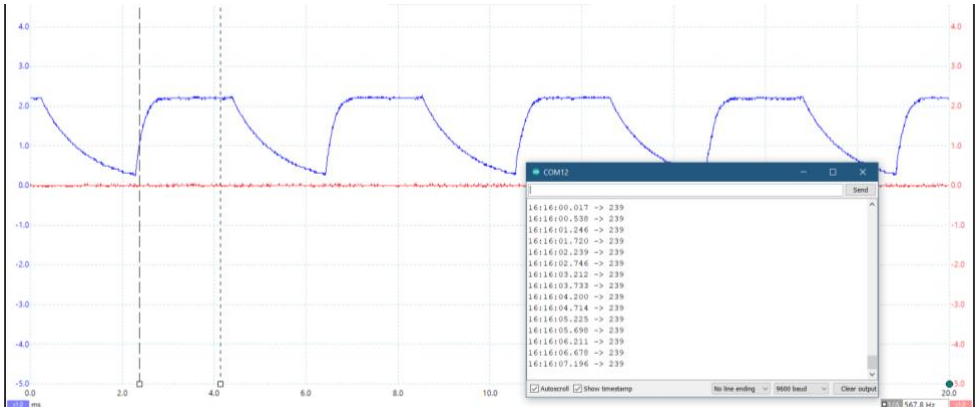
The output of the comparator was passed to a digital input pin on the Metro M0 board. The program measured the length of time that the signal was high, and the time the signal was low was added to this. This value represented the time period of the modulating signal. The reciprocal of this value was taken, producing the estimated frequency of the original modulating signal. As can be seen in the waveform above, the signal was not perfect. The random dips would be interpreted by the program as a very high frequency signal. To solve this, the program was modified to take 40 samples, and only definitively output a value if at least half of the samples were within a certain tolerance of one of the two possible modulating frequencies. If the samples had a wide variety of values, then the program would output a value of -1.



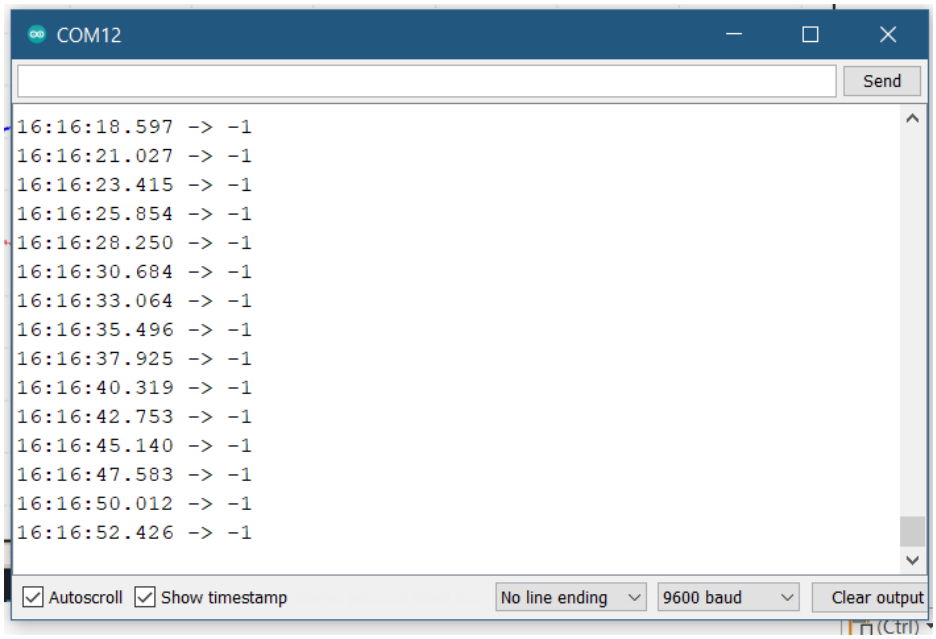
Screenshots below shows the output of the program when the Exorock is set to emit a radio signal with a modulating frequency of 151 Hz.



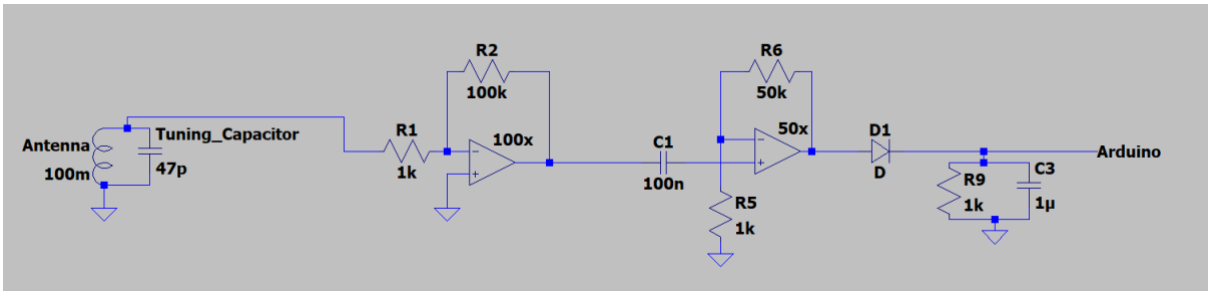
The below graph shows the output of the program when the Exorock is in 239 Hz mode. The oscilloscope waveform of the output of the envelope detector is also shown. Since the Metro M0 board interprets anything above 2 V as a digital high input [9], the signal is essentially viewed as a square wave by the microcontroller.



When the Exorock is not emitting any radio signal, the function returns a value of -1, as shown below.

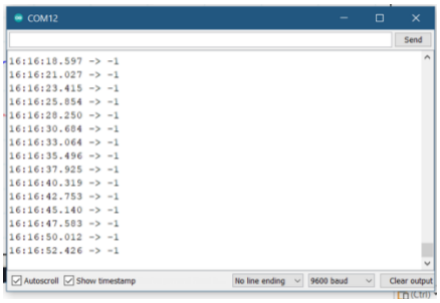


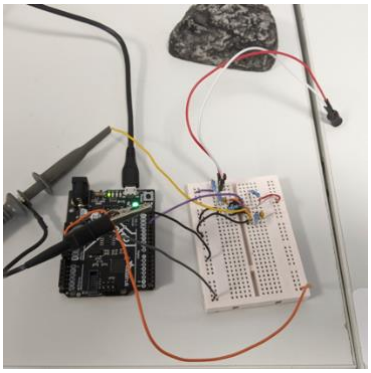
An MCP6292 op-amp was used to compact the circuit into using only a single integrated circuit chip. This was possible as this new op-amp has a gain-bandwidth product of 10 MHz [10], meaning a single op-amp could be used to achieve a gain of up to 1000x. The output of the envelope detector was passed straight into the Metro M0 board as the new system produces minimal noise, so no comparator was needed. The new diagram is shown below.



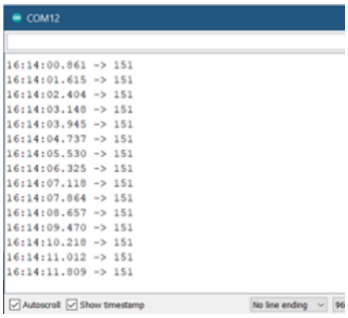
Demonstration

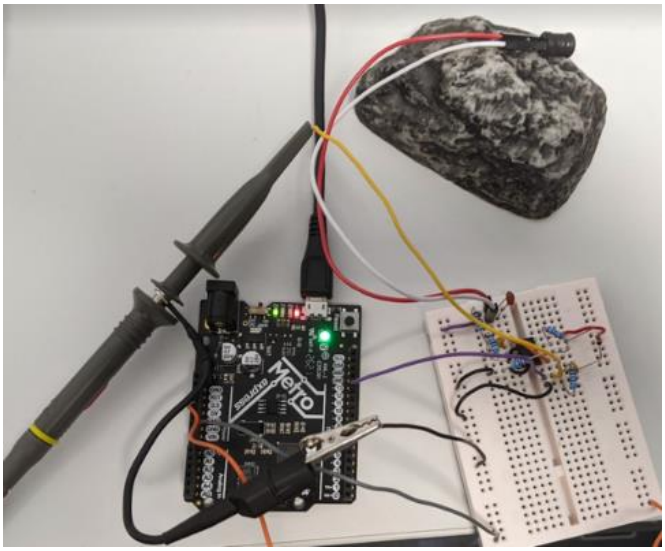
No radio waves



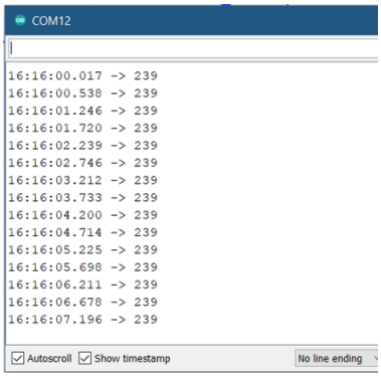


Radio wave modulated at 151 Hz





Radio wave modulated at 239 Hz



Motors and Control

Researching Motor Driver

The H Bridge provided was a TC78H620FNG, capable of driving 2 DC brushed motors at a maximum supply voltage of 18V, well above the 6V supplied by the battery.

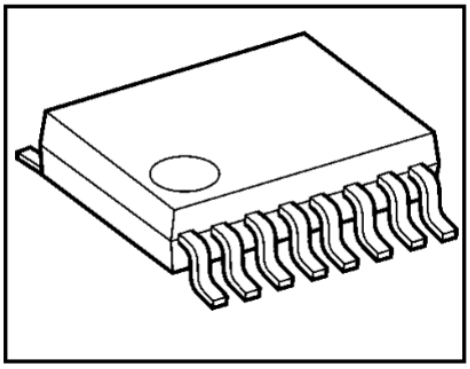


Figure 8 from [11]

Testing the Driver

The first test was just to test if the motor drivers were all initially working. This was achieved using a template found online. PWM on the analogue pins was used to control the speed of the motors. *Details of the connections made can be found in the appendix.* [A] [B] [C]

```
/*
    L298N Motor Demonstration
    L298N-Motor-Demo.ino
    Demonstrates functions of L298N Motor Controller

    DroneBot Workshop 2017
    http://dronebotworkshop.com
*/

// Motor Left: Setting the Pins

int enLeft = 3;
int inLeft = 4;
// int in2 = 7;

// Motor Right: Setting the Pins

int enRight = 9;
int inRight = 8;
// int in4 = 4;

void setup()

{

    // Set all the motor control pins to outputs

    pinMode(enLeft, OUTPUT);
    pinMode(enRight, OUTPUT);
    pinMode(inLeft, OUTPUT);
    // pinMode(in2, OUTPUT);
    pinMode(inRight, OUTPUT);
    //  pinMode(in4, OUTPUT);
}

void demoOne()

{

    // This function will run the motors in both directions at a fixed speed

    // Turn on motor A

    digitalWrite(inLeft, HIGH);
    //  digitalWrite(in2, LOW);

    // Set speed to 200 out of possible range 0~255

    analogWrite(enLeft, 200);

    digitalWrite(inRight, HIGH);
    //  digitalWrite(in4, LOW);

    // Set speed to 200 out of possible range 0~255

    analogWrite(enRight, 200);

    delay(2000);

    // Now change motor directions

    digitalWrite(inLeft, LOW);
    //   digitalWrite(in2, HIGH);
    digitalWrite(inRight, LOW);
    //   digitalWrite(in4, HIGH);

    delay(2000);

    // Now turn off motors

    digitalWrite(inLeft, LOW);
    // digitalWrite(in2, LOW);
    digitalWrite(inRight, LOW);
    //  digitalWrite(in4, LOW);

}

void loop()

{

    demoOne();

    delay(1000);

}
```

Figure 9 is adapted code from [12]

Direction Testing

After now being able to move the motors in a forward direction, the subsequent step would be to think about how to implement the other three directions.

Flipping the direction of one or more motors allowed the rover to move in all of the necessary directions using only two wheels. The functions shown on the right control this logic. The moving_around() function controls the speed of each motor independently.

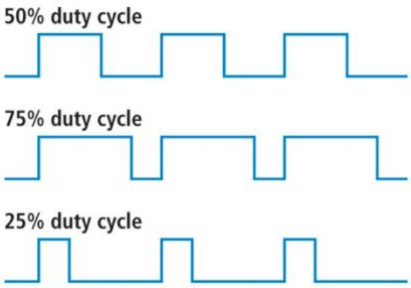
```
void forward()
{
    digitalWrite(inLeft, LOW);
    digitalWrite(inRight, HIGH);
    moving_around();
}

void backward()
{
    digitalWrite(inLeft, HIGH);
    digitalWrite(inRight, LOW);
    moving_around();
}

void right()
{
    digitalWrite(inLeft, LOW);
    digitalWrite(inRight, LOW);
    moving_around();
}

void left()
{
    digitalWrite(inLeft, HIGH);
    digitalWrite(inRight, HIGH);
    moving_around();
}
```

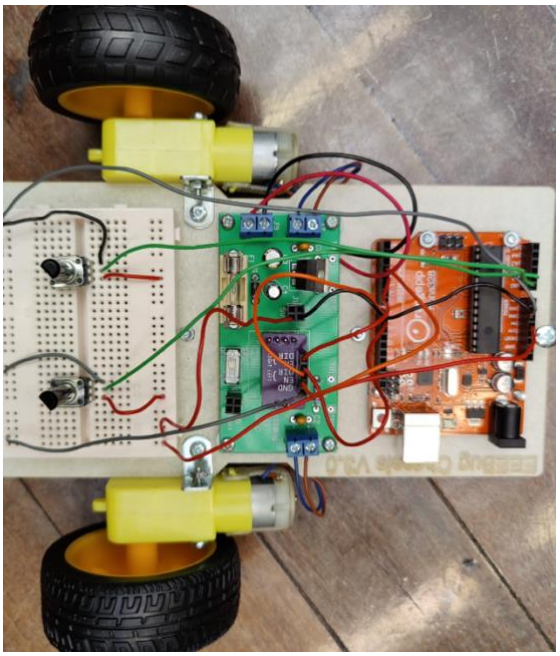
Pulse-Width Modulation

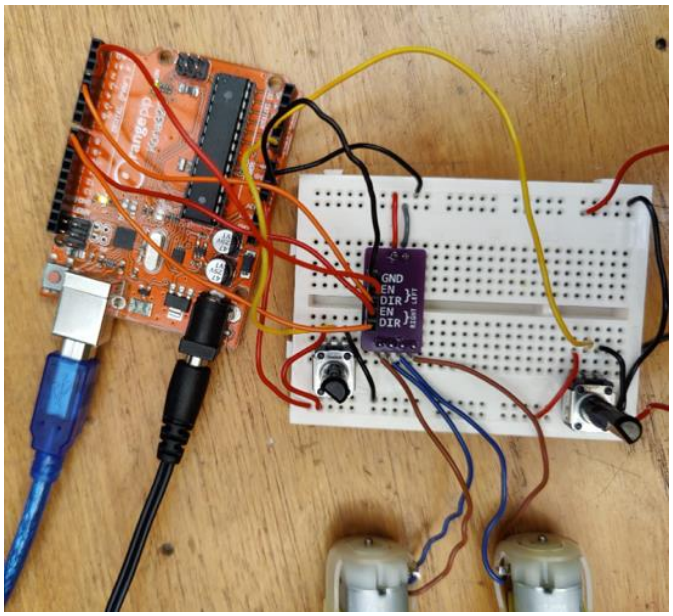
The motor speed can be varied using pulse-width modulation.

Varying the duty cycle allowed the user to define speeds: the higher the duty cycle, the faster the wheels should turn. This was further tested with a potentiometer, as the speed could be varied using the potentiometer using the code below:

Figure 10 is from [13]







```
#include <Servo.h> //includes the servo libraries
// Servo back_left_servo;
// Servo back_right_servo;
// defining characters
#define MOVE_FORWARD 'w'
#define MOVE_BACK 's'
#define MOVE_LEFT 'a'
#define MOVE_RIGHT 'd'
#define FULL_STOP 'e'

// Convert to range of 0-255

SpeedRight = map(SpeedRight, 0, 1023, 0, 255);
SpeedLeft = map(SpeedLeft, 0, 1023, 0, 255);

// Adjust to prevent "buzzing" at very low speed

if (SpeedRight < 20)
    SpeedRight = 0;

if (SpeedLeft < 20)
    SpeedLeft = 0;

// Set the motor speeds

analogWrite(enLeft, SpeedRight);
analogWrite(enRight, SpeedLeft);
}
```

Figure 11 is code from [12]

```
int enLeft = 3;
int inLeft = 4;
// int inRight = 7;

// Motor Right: Setting the Pins

int enRight = 9;
int inRight = 8;
// int in4 = 4;

int SpeedControlRight = A0;
int SpeedControlLeft = A1;

int SpeedRight = 0;
int SpeedLeft = 0;

void setup()

{

    // Set all the motor control pins to outputs

    pinMode(enLeft, OUTPUT);
    pinMode(enRight, OUTPUT);
    pinMode(inLeft, OUTPUT);
    // pinMode(inRight, OUTPUT);
    pinMode(inRight, OUTPUT);
    //  pinMode(in4, OUTPUT);
}

void loop()

{
    // Set Motor A forward

    digitalWrite(inLeft, HIGH);
    digitalWrite(inRight, LOW);

    // Set Motor B forward

    // digitalWrite(in3, HIGH);
    // digitalWrite(in4, LOW);

    // Read the values from the potentiometers

    SpeedRight = analogRead(SpeedControlRight);
    SpeedLeft = analogRead(SpeedControlLeft);
    Serial.println("Right Speed: " + SpeedRight);
    Serial.println("Left Speed: " + SpeedLeft);
```

Figure 12 is adapted code from [12]

Although this allowed control of the rover, it could not be done wirelessly. The next idea was to use a physical joystick on a separate remote, but this would require an entirely separate circuit, integrated with Wi-Fi. Therefore, a more software-based control method was preferred, either an app or a HTML Webpage.

Sending Instructions to the Arduino

After testing that all the directions are working, the next phase was to find a way to send instructions in real time and to command the rover through a wired connection. This then led to researching the serial plotter which would allow the user to send instructions to the Arduino from a keyboard whilst still connected to the Arduino via a wire. Completing and understanding this process would help the transition to wireless communication much easier.

After looking into serial plotters and serial printing to help debug any potential issues that could arise, the code to the right was added.

This enabled the user to control the rover through a wired connection. The only issue that would arise would be small coding mistakes that were later fixed. After each command the user would have to press enter to send the instruction.

```
void loop()
{
  char byte = 0;
  // press q to cancel and exit
  while (byte != 'q')
  {
    Serial.readBytes(&byte, 1);
    // press w to move forward
    if (byte == MOVE_FORWARD)
    {
      forward();
      Serial.print("move forward \n");
      byte = 0;
    }
    // press s to move backward
    if (byte == MOVE_BACK)
    {
      backward();
      Serial.print("move back \n");
      byte = 0;
    }
    // press a to turn left
    if (byte == MOVE_LEFT)
    {
      left();
      Serial.print("move left \n");
      byte = 0;
    }
    // press d to turn right
    if (byte == MOVE_RIGHT)
    {
      right();
      Serial.print("move right \n");
      byte = 0;
    }
  }

  Serial.end();
}
```

Small Improvements

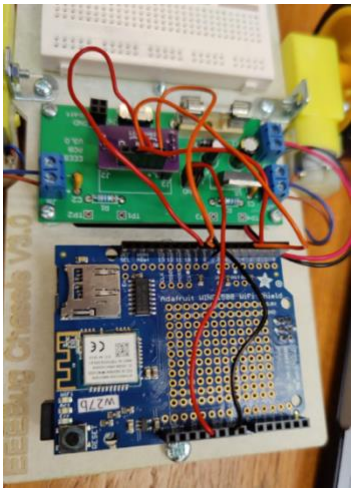
The four directions: left, right forwards and backwards, would be enough, but for the driver, manoeuvrability is key. Therefore, instead of repeatedly pressing multiple buttons to move diagonally, building functions to do this would prove to be useful especially if the rover was stuck in an awkward position. This would improve the overall time it would take to complete the task on the field.

```
void orbit_right_forward()
{
  digitalWrite(inLeft, LOW);
  digitalWrite(inRight, HIGH);
  analogWrite(enLeft, 150);
  analogWrite(enRight, 60);
}

void orbit_right_backward()
{
  digitalWrite(inLeft, HIGH);
  digitalWrite(inRight, LOW);
  analogWrite(enLeft, 150);
  analogWrite(enRight, 60);
}
void orbit_left_forward()
{
  digitalWrite(inLeft, LOW);
  digitalWrite(inRight, HIGH);
  analogWrite(enLeft, 60);
  analogWrite(enRight, 150);
}

void orbit_left_backward()
{
  digitalWrite(inLeft, HIGH);
  digitalWrite(inRight, LOW);
  analogWrite(enLeft, 60);
  analogWrite(enRight, 150);
}
```

The above functions were created. The name "orbit" was assigned to each of these as the speeds were calibrated to be able to rotate continuously around the sample rocks.

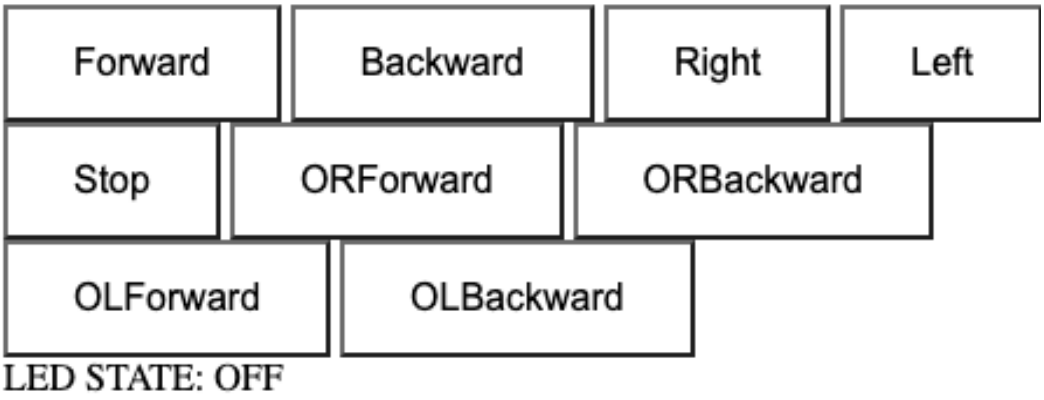
Developing Wireless Controls

To develop wireless controls, the first step was to install and connect the correct libraries to the Arduino and attach the WINC1500 module.

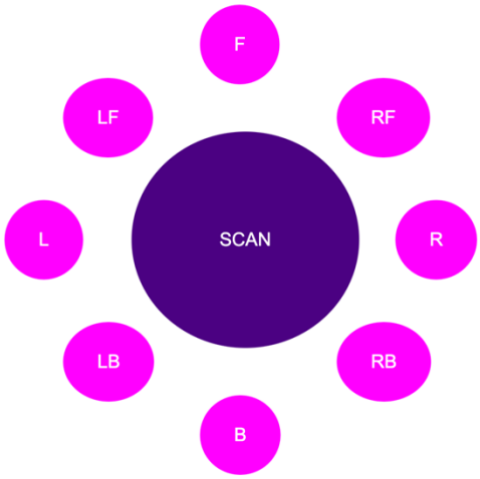


Developing a HTML Page, adding style using CSS, and implementing functionality with JavaScript using HTTP requests seemed to be very promising. This would enable the user to control the rover easily and wirelessly with a click of a button but would also require a stop button to halt the rover after each instruction was sent. Despite this, the method would be ideal because it meant that the rover could be controlled from any device.

Writing code to use for the "proof of concept" stage gave the following HTML page:



With two buttons, one for moving forward, and one for stopping, the commands should execute instantly. However, the main issue that occurred when trying to implement this method was that the HTML page wouldn't load on the web browser, or not execute any of the instructions whenever a button was pressed.



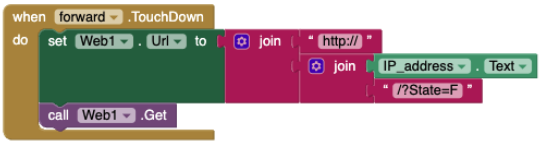

```
<head>
    <style>
    .btn {background-color: #FF00FF;border: none;color: white;padding: 30px 35px;font-size: 19
    .btn:hover {background: #eee;}
    .btnF {position: absolute;left: 300px;top: 100px;}
    .btnB {position: absolute;left: 300px;top: 500px;}
    .btnL {position: absolute;left: 100px;top: 300px;}
    .btnR {position: absolute;left: 500px;top: 300px;}
    .btnLF {position: absolute;left: 160px;top: 175px;}
    .btnRF {position: absolute;left: 440px;top: 175px;}
    .btnLB {position: absolute;left: 160px;top: 425px;}
    .btnRB {position: absolute;left: 440px;top: 425px;}
    .btnScan {background-color: #4B0082;border: none;color: white;padding: 100px 90px;font-siz
    .btnScan:hover {background: #eee;}
    </style>
</head>

<body>
    <form action = "result.html" method = "GET">
        <button class="btn btnF" id = "F">F</button>
        <button class="btn btnB" id = "B">B</button>
        <button class="btn btnL" id = "L">L</button>
        <button class="btn btnR" id = "R">R</button>
        <button class="btn btnLF" id = "LF">LF</button>
        <button class="btn btnRF" id = "RF">RF</button>
        <button class="btn btnLB" id = "LB">LB</button>
        <button class="btn btnRB" id = "RB">RB</button>
        <button class="btnScan" id = "SCAN">SCAN</button>
    </form>
    <br>LED STATE: <span id="state\">OFF</span>
</body>
```

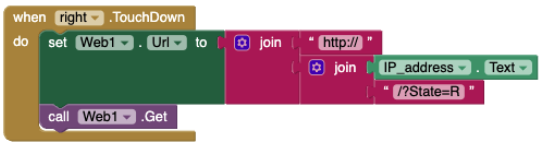
This left two options:

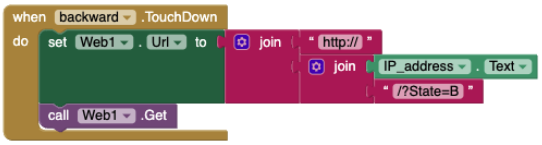
- Continue with the html page and find a way to bypass the problem using another HTML webpage.
- Develop a simple android app to send and receive instruction wirelessly to the Arduino.

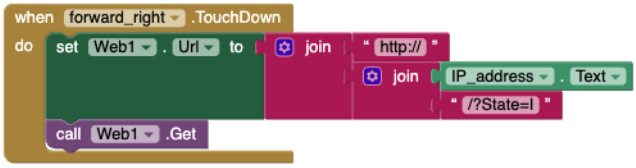
The latter option seemed to be a lot more promising and there were many templates online with source code examples that could be adapted to our specific needs.

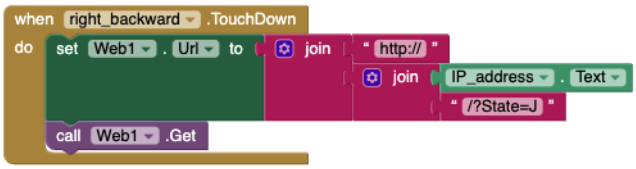
The diagrams below show the code created in the MIT app inventor software to control the back-end functionality of the app.
These original code blocks were taken from [14] and then adapted and then extended for our instruction set.

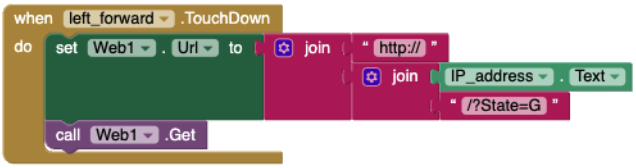


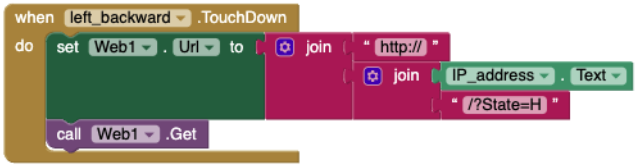


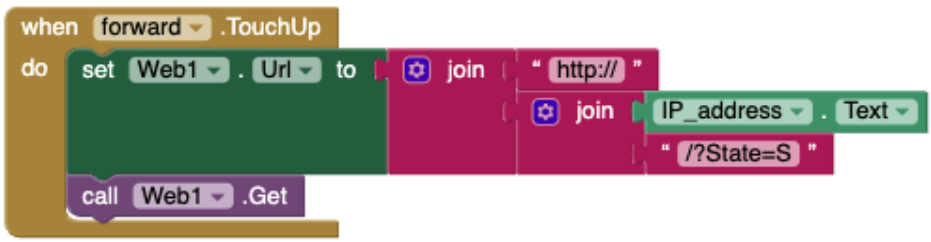










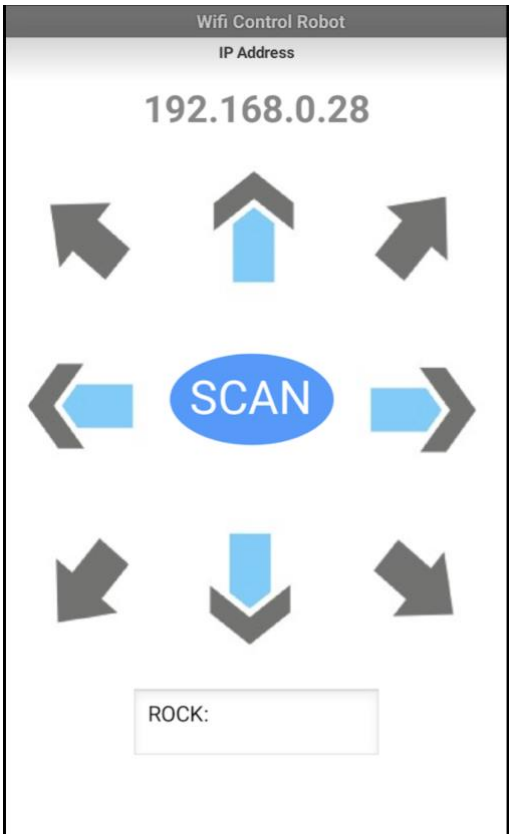


The code below was executed on the Metro M0 to communicate with the app through HTTP requests.

```
void loop()
{
  server.handleClient();
  command = server.arg("State");
  if (command == "F") {forward()};
  else if (command == "B") {backward()};
  else if (command == "L") {left()};
  else if (command == "R") {right()};
  else if (command == "I") {orbit_right_forward()};
  else if (command == "G") {orbit_left_forward()};
  else if (command == "J") {orbit_right_backward()};
  else if (command == "H") {orbit_right_backward()};
}
```

Figure 13 from [14]

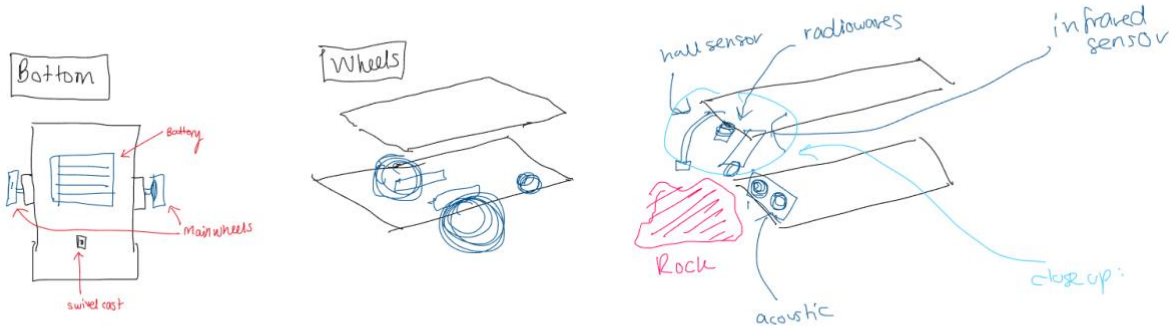
This system worked and there didn't seem to be any issues with the transfer of data. The image below shows the user interface of the app.



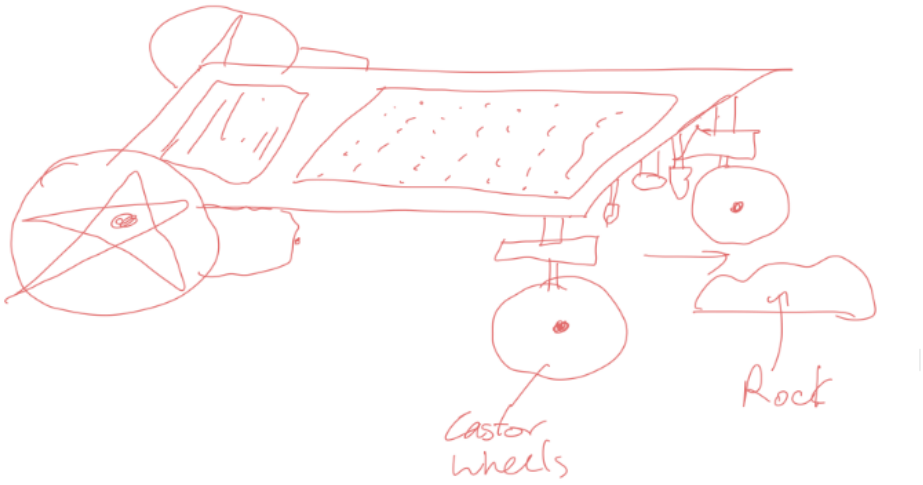
Chassis

Initial ideas were drawn to evaluate the advantages and disadvantages of each, allowing us to come up with a final design that met all our criteria.

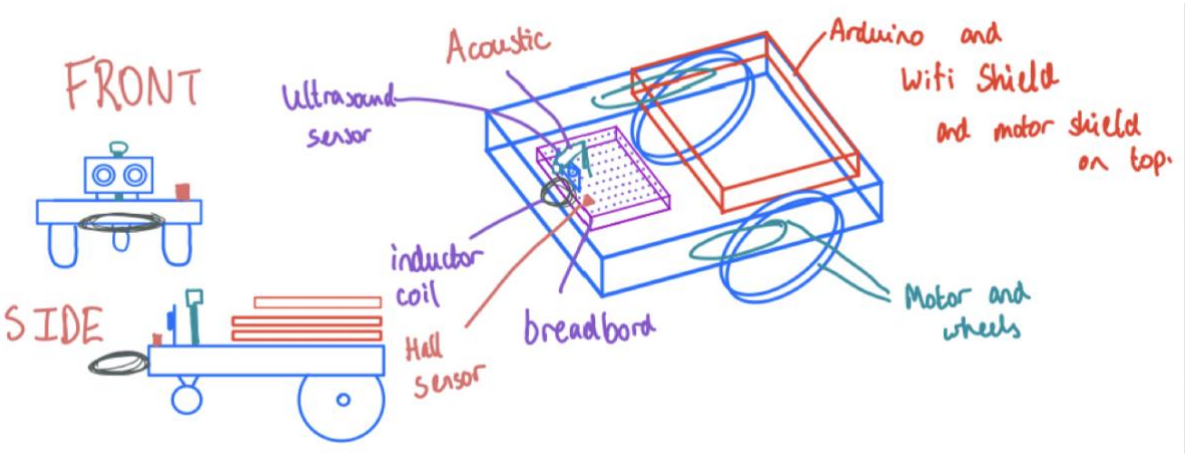
The first idea was a two-layered, laser-cut acrylic body with the acoustic sensor mounted on the bottom layer and all other sensors hanging from the top. Both layers would have space to mount a small breadboard for the rest of the circuitry. The concept drawing is shown below. The advantage of this is that it creates plenty of space for everything needed to be carried. The height of the top layer also allows the sensors to hang above the Exorock, the position where the emitted signals are strongest. The disadvantage is that it adds extra weight, but this could be mitigated by cutting excess material from the top layer where it isn't needed.



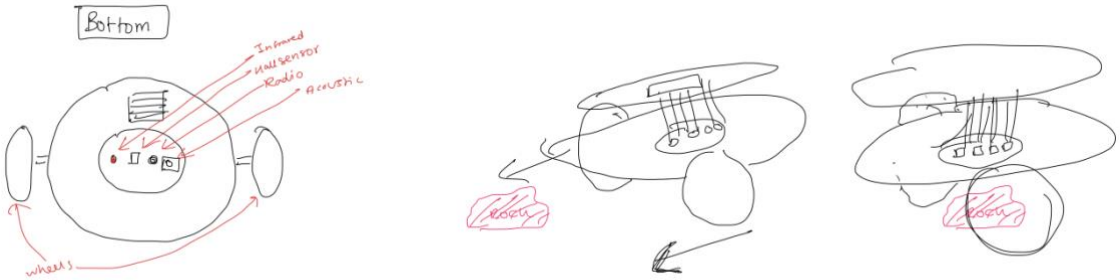
Another potential idea was a single layer that was raised above the height of the Exorock. This would allow it to simply drive over the rock, providing the advantage of positioning the sensors directly above the Exorock. The disadvantage is that it requires an extra two wheels capable of swivelling which adds extra weight and cost to the design. The concept drawing is shown below.



A similar idea was to produce a single layer chassis but with a lower height. This would be the simplest to build since it would only require one swivel wheel and we would not require many changes to the existing chassis. However, it would be challenging to fit all four sensors onto a single breadboard without greatly widening the chassis. One of the aims is to make the build as small and narrow as possible to fit between rocks and navigate the terrain. This would be more difficult with the design proposed below.

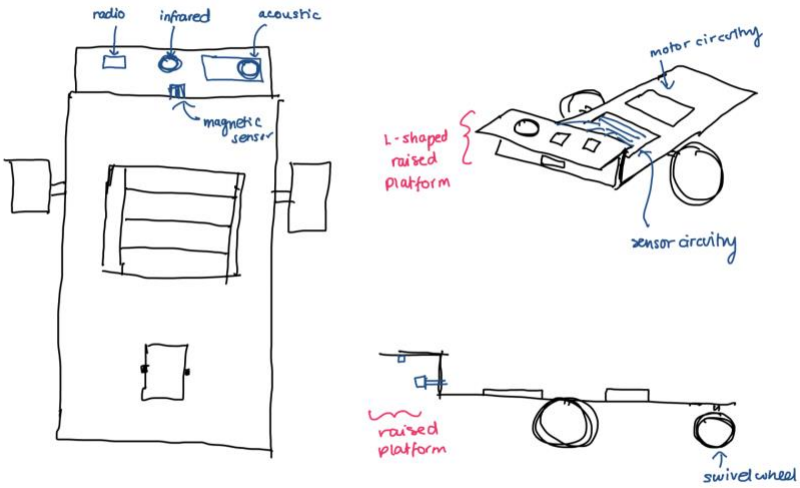


The next design idea consisted of a double layer rounded structure. The bottom layer would be raised enough so that the rover can drive over the rock. The centre of the structure would have the receivers for all four sensors. The rover would be driven in such a way that the centre would be aligned over the top of the rock.



The advantage of this structure is that the sensors can be very close to the top of the rock. This circumvents any issues of range. Furthermore, if the hanging structure is successful, it would ensure that all the complex wiring, besides the motors, could be placed on the second layer. The durability and portability of the hanging sensor's structure is an issue because it could easily become dismantled. Moreover, although in the drawing above two wheels are shown, realistically, to balance the double layer structure, far bigger wheels than the ones available to us would be needed. Alternatively, more wheels could be used to balance the chassis, however this would defeat the ability to drive straight over a rock. Another possibility would be to use complex self-balancing circuitry and algorithms however this would increase the cost, power consumption and is out of the scope of our project.

The final idea was to simply add an extra L-shaped acrylic structure to the original chassis. This would allow us to build on a structure we are already comfortable with. Furthermore, it would require little extra material, and the weight of our overall rover would be low. The rover would also be quite narrow, making it easier to navigate the rough terrain. However, it may be difficult fit the circuitry for all 4 sensors onto one breadboard.



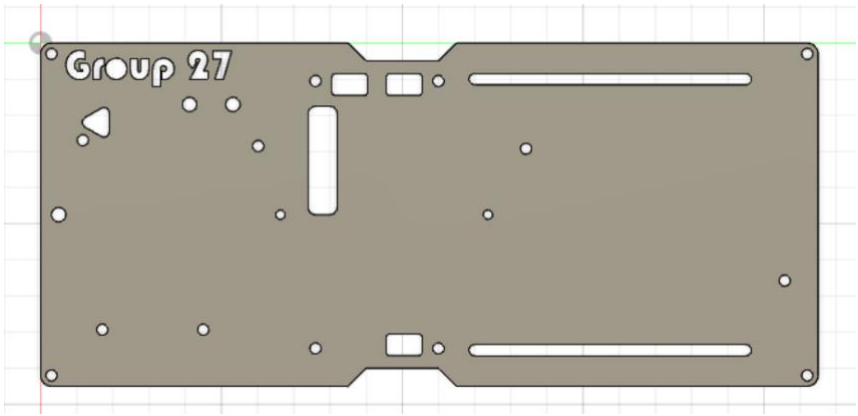
Weighing up the pros and cons, it was decided that this final design was best suited for the brief. Its size gives good manoeuvrability, and the L-shaped structure would be easy to manufacture, requiring little extra material. It also combines many advantages of the other "failed" designs of being able to drive just over the rock and have sensors exactly above it, to improve accuracy of the readings, as well as having the castor wheel at the back for ease of manoeuvrability. Being pragmatic with the wiring would make our design all the more efficient and eliminate the weight of the double layer in a previous design.

Manufacturing

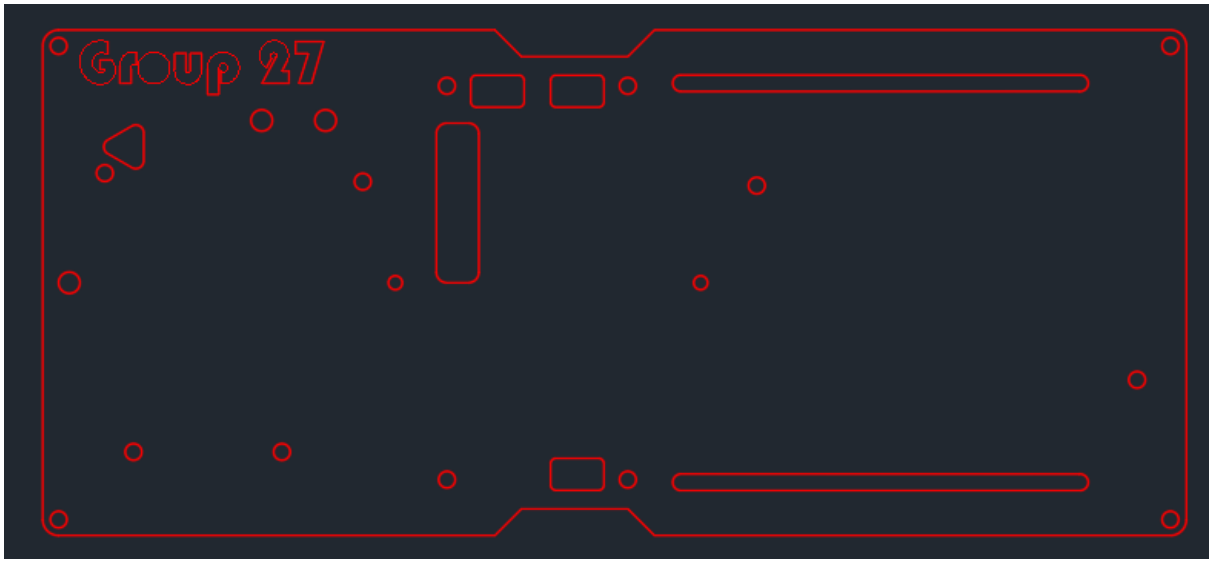
The new chassis was designed in Fusion 360. Based on the original provided design, a few modifications were made to make it more suitable for our needs. The original wheels were mounted horizontally with only a single screw holding the brackets to the chassis (with an additional two screws holding the motor to the brackets). To mount the motor vertically, two holes on the surface of the chassis were needed.

However, to add flexibility in the future, two horizontal tracks were added instead at the front to ensure the brackets could be mounted in any position with any separation. However, adding these tracks reduced the space in the middle, meaning the breadboard no longer fit in the original position.

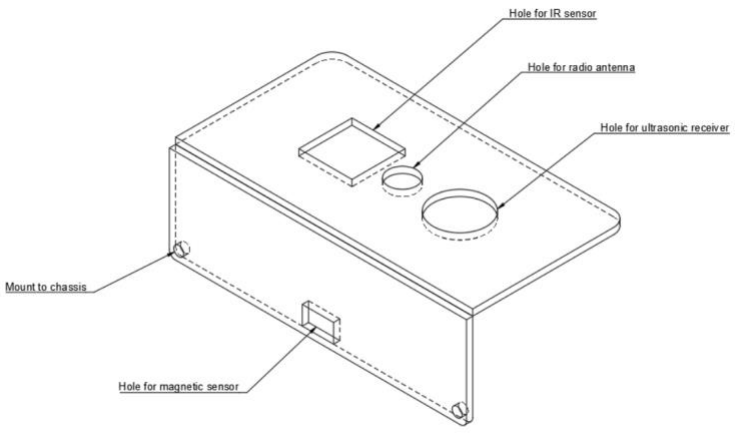
The two screw holes for the breadboard were therefore moved and the length of the chassis was extended to allow the breadboard to be rotated 90 degrees. In addition, four M3 sized holes were added to the corners to facilitate the attachment of any additional structural pieces in the future.



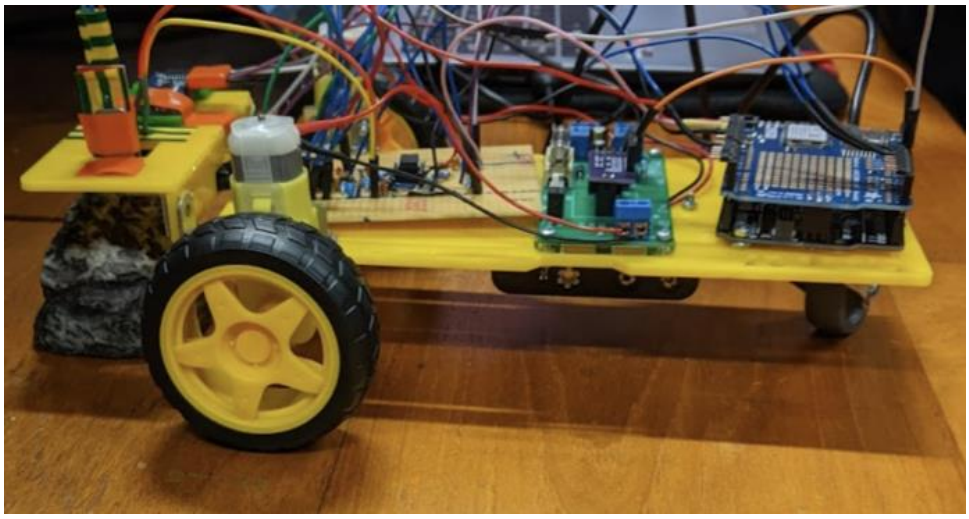
The design was imported into AutoCAD where the line properties could be modified to the standards required by the laser cutter. The laser cutter used required cutting lines to be red with a thickness of less than 0.05 mm. The image below shows the diagram in AutoCAD, ready to be passed to the laser cutter control program.



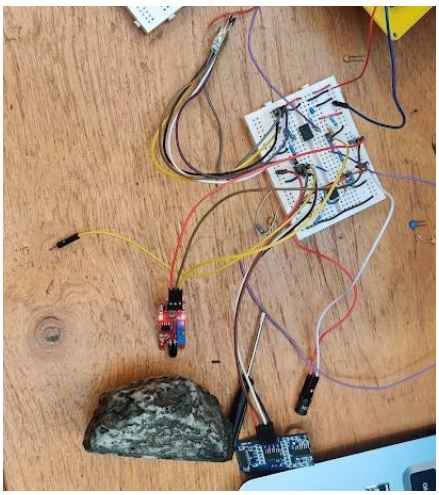
A mount for the sensors was created by laser-cutting two pieces of acrylic and gluing them together. The entire mount was then secured to the chassis through angle brackets and screws. The image below shows the mount structure as well as the cut-outs for each sensor.



The sensors were threaded through the holes and taped into place. The image below shows the final design with all parts mounted.

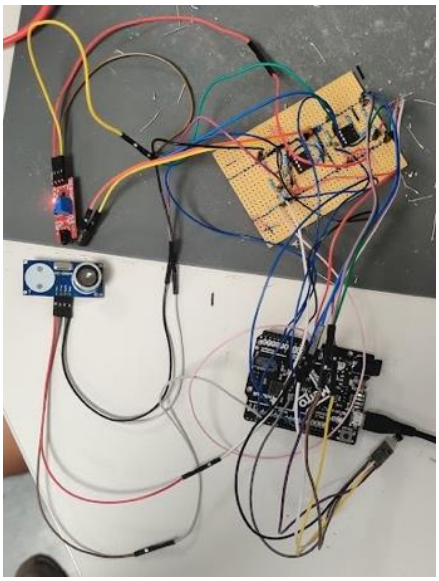


Combining/Testing All Components



Once all sensors had been built and tested individually, they were all combined into a single breadboard. The finished circuit utilised two op-amp chips, meaning a total of four amplifiers were used.

Each sensor was tested in turn to ensure they all still worked. Whilst they did work most of the time, a few of the connections were extremely unreliable and either kept falling out or required someone to hold them in place. The solution to this was to solder all of the components onto a copper stripboard. This would improve reliability as once the components were in place and determined to be functioning, very little would be able to change. Furthermore, the durability of the system would also be improved as the soldered components were more resilient to vibrations and drops than components slotted into a breadboard. The final circuitry is shown, although the image shows sensors which are attached with jumper wires and not actually soldered yet.



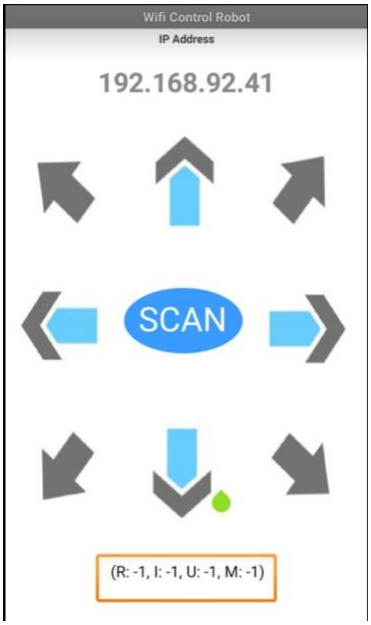
Once the chassis design had been finalised, all the wires connecting the stripboard to the sensors were cut to length and soldered to ensure they would not be able to fall out. The final design was tested once again to ensure all of the sub-circuits worked as intended. At this point, the circuitry was ready to be mounted to the chassis and the sensor code combined with the motor and networking control.

Identifying the Rock

Each of the individual programs used by each sensor were placed into a function, and all the function were placed into a single file. This allowed the Metro M0 board to take measurements from all four sensors to identify the rock type. Within a new scan function each sensor function is called, one after the other, and the returned value stored in a variable. A cascade of if/else if statements are then used to compare the measured values to the table of known rock properties. If the gathered data exactly matches the properties of a certain rock type, the scan function will return a string containing the rock name. If no match is found, the text "Unknown" is returned. The code snippet below shows the comparisons made.

```
if (radioFreq == 151 && ultrasonic == 1 && magnetic == -1 && IRFreq == -1){
  return "Gaborium";
}else if (radioFreq == 239 && ultrasonic == 0 && magnetic == -1 && IRFreq == -1){
  return "Lathwaite";
}else if (radioFreq == 151 && ultrasonic == 0 && magnetic == 1 && IRFreq == -1){
  return "Adamantine";
}else if (radioFreq == 239 && ultrasonic == 0 && magnetic == 0 && IRFreq == -1){
  return "Xirang";
}else if (radioFreq == -1 && ultrasonic == 0 && magnetic == -1 && IRFreq == 353){
  return "Thiotimoline";
}else if (radioFreq == -1 && ultrasonic == 1 && magnetic == -1 && IRFreq == 571){
  return "Netherite";
}

return "Unknown";
```

This scan function can be called in the final program whenever a button on the controller app is pressed. The returned value can then be transmitted back to the app to be displayed on screen. The time taken to scan each rock type, with the final optimised code, is shown below.



Rock Type	Scan Time (ms)
None	2100
Gaborium	533
Lathwaite	1625
Adamantine	1900
Xirang	1627
Thiotimoline	1962
Netherite	538

The absolute worst-case scenario, where the rock is nowhere near any of the sensors, is 2.1s. This will only ever occur if the scan button is pressed whilst not positioned over the rock, where the code for each of the sensors has to wait the entire time out period since it receives absolutely no signal. The range of scan times is 0.5s to 1.9s, fast enough to complete our specification.

Efficiency

Power Consumption

Voltage and current measurements of components in various states were taken to calculate the power consumption.

Component	Voltage (V)	Current (mA)	Power (mW)
Metro M0	5	50	250
Wi-Fi Shield	3.3	80	264
IR Sensor	3.3	4.8	15.84
Ultrasonic Sensor	5	15	75
Magnetic Sensor	3.3	4.55	15
Motor (single)	5	200	1000
Amplifier (single)	3.3	25	82

From testing, it is clear that the motors consume a significant portion of the total power. Therefore, in order to keep the power consumption low, the total number of motors used was minimised. This was the key reason that two motors were used as opposed to four, as the trade-off between speed and power consumption was not worth it.

The circuitry of the sensors was also optimised. Initially, radio circuit required five op-amps because of the limited gain-bandwidth product of the op-amps provided. These were replaced with more powerful op-amps that were ordered online. Other choices in the circuitry also reduced power consumption. The ultrasonic transmitter in the acoustic sensor was removed, decreasing the power required by the sensor. For the magnetic hall sensor, there was a choice to purchase an entire integrated circuit with built-in amplification and other functionality. However instead, two individual hall effect transistors were used along with a differential amplifier. This avoided the unnecessary functionalities in an integrated circuit where power would have been wasted.

Testing the power consumption of the entire rover, when powered by a 6V power supply produced the following data.

State	Current (mA)	Power (mW)
Idle	130	780
Scanning	140	840
Moving	250	1500
Moving (with resistance)	330	1980

Weight

Since the rover was designed for operation on the moon, weight was a pivotal consideration when selecting parts and components to be used.

Some of the components for the rover that were already given were: the PCB to control the motors, the Metro M0 development board, two DC motors, the battery pack, and breadboards. Since these components were required for the rover to function, the main weight considerations were the chassis, wheels, and sensors.

Chassis

The main differences in the weight of the chassis were down to the choice of material. Wood, plastic, and metal were considered. A metal chassis (such as steel) would be too heavy for the brief, due to the cost limitations we would not have access to lightweight yet durable metals. Plastic was seen as a strong choice (especially acrylic) as it is relatively lightweight, but also strong enough for the environment given. Different woods were also considered. MDF and plywood were potential choices due to their light weight and reliability.

Wheels

Although the original EEE Bug wheels were lightweight they were not suitable for handling the rough terrain of the arena. Larger wheels with treads impressed onto the rubber were used to improve traction. This slightly increased weight, however it was a necessary modification.

Sensors

Radio Sensor

Instead of making a large inductor using a coil of wire, a small inductor component was used. This reduced not only weight but space too, as less wire was being used.

IR sensor

The infrared sensor itself was on its own PCB and was already just a sensor. The weight saving aspect of this component was the implementation of a perforated board instead of a breadboard. It is not only more reliable, but due to the material difference it is also far lighter than a breadboard.

Magnetic Sensor

Figure 14 from [17]

As seen in the picture to the left the whole sensor if purchased would include a sensor with extra circuitry. However, this was unnecessary so instead the hall effect sensor itself (leftmost component in the picture) was used only, removing the weight of the PCB.

Ultrasound

The original ultrasound component was a transmitter and receiver. The transmitter was not necessary in the design and was therefore removed to save weight.

Financial Costs

With a budget of £60, a close eye was kept on the rising cost of the project. The pros and cons of any item were carefully evaluated.

Radio

Due to having simple components in the lab, the cost was kept at a minimum when it came to using resistors and capacitors. An inductor was used as an antenna. As this was available from the lab, it was free. The value of this is currently at £1.55.

Additionally, multiple op-amps were used for this, taking up valuable space on the breadboard. To fix this, op-amps with a higher gain-bandwidth-product were bought to efficiently reduce the number integrated circuit chips needed. The minimum quantity of five was purchased, for a total cost of £7.33.

Ultrasonic/Acoustic

The following sensor was just as big as the inductor meaning it was lightweight, low-profile, and therefore perfect for the brief. However, the cost of £11 meant cheaper alternatives were available.

Figure 15 from [15]

The final sensor used was an ultrasonic rangefinder with the transmitter de-soldered. At a cost of £2, it was cheap to purchase.

Figure 14 from [16]

Magnetic/Hall Effect

Hall effect sensors could only be puchased in bulk, meaning the total cost was significantly greater than the price of a single sensor. However, a few sensors had already been acquired by a member of the team, so could be used free of charge.

Infrared

The sensor used in the final design could only be bought in bulk at a total cost of £18. Unlike the op-amps, only a single sensor was necessary so this would have been a massive waste. A member of the team already had one available which could therefore be added for free.

Body

With access to the robotics lab, manufacturing the chassis was also free of charge. The wheels provided to us were very small and slippery and therefore not suitable for purpose.

Figure 17 from [18]

With these costing £8.60, the choice between price and usability had to be decided. These wheels were soon deemed as unnecessary as the only selling point is that you can move in any direction with these wheels, and it will look much cooler. However, the same performance can be expected from normal wheels with perhaps more directional movement functions such as the orbit ones described above.

The last thing item that was needed was a third wheel that could spin around and act as a stabiliser on the unbalanced side. The most logical

choice would be swivel castors. Small and light ones were crucial for the project. The smallest option we could find was 32mm in radius which was perfect for the project. This costs £2.88 but is more than necessary for effective manoeuvrability.

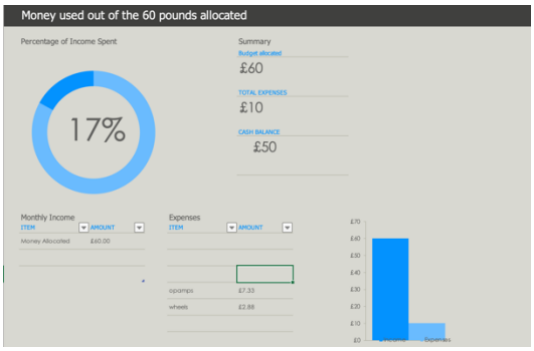
Budget Management

Overall, the total price of the rover is at 79.58 the full breakdown of which can be seen below:

Product:	Price:
Ultrasonic sensor	2.00
IR sensor	1.50
2 hall effect sensors	0.50
wheels	2.88
Metro M0 board	23.70
Adafruit Wi-Fi shield	24.00
Bread boards	5.00
Acrylic sheets	10.00
other	10.00

This table shows the price of everything on the Rover. Most of the components were supplied in the lab boxes. All extra components were bought using the £60 budget. Full breakdown of purchases can be seen below. Some of the components listed had been brought from home and are included in this breakdown however money wasn't technically spent on them

Below is the budget exactly like above but discarding the pieces that were bought in from home and only counting the pieces that were ordered with the £60 allocated.



Taking out one of the sensors to reduce costs and relying on the other three to detect the rock was considered, but ultimately, it was recognised that it would have been less reliable and therefore not worth the £2.00 that would have been saved

The diagram to the right shows the cost, had all of the sensors been bought using the £60 budget, instead of being acquired form other sources.

Driving

To determine who would drive on the day, a driving test took place with a mock course created to test each group member's ability to manoeuvre the rover around the mock course with obstacles in different places as well as speed and reliability when detecting rocks. The results are as follows:

Name	Time (s)		Rock Detected (Y/N)	
	Try 1	Try 2	Try 1	Try 2
Awais	00.41.34	00.50.23	Y	Y
James	00.50.70	00.54.70	Y	Y
Anish	00.52.09	00.54.07	Y	Y
Yasser	01.10.01	00.54.03	N	Y
Hasnat	01.02.10	1.00.10	Y	Y

The best driver, in terms of speed and reliability was chosen to drive on the day of the demonstration.

Project Management

Aims, Milestones, Project Planning

To keep track of the project, the 6 available weeks (with 5 weeks for report submission) was split into two segments: Tues-Thurs and Fri-Mon. Below is a table where, at the beginning of the segment, the aims for that segment would be updated. Then at the end, the milestones crossed and the plan for the following segment were recorded.

Segment	Aim	Milestone	Plan for Later
#1: 17 – 19 May	-Understand project and realise approach to 4 sensors and motors	-Groups have been made: Motors: Hasnat/Anish Radio: Awais/Jamie/Oliver Acoustic/Infrared/Magnetic: Yasser/James -Created LT Spice for radio sensor -Created inductor by coiling wire with a bottle -Brought ultrasonic sensor from home and modified -Started looking into infrared sensors -Understood and tested motor driver	-Still waiting for rock. -Once rock arrives, test and fine tune ultrasonic -Keep working on motors -Start building radio -Keeping working on infrared
#2: 20 – 23 May	-Build and test radio -Finish infrared -Understand Wi-Fi shield -Implement all 4 directions of movement with motors	-Groups changed upon understanding team members interests/strengths -Ultrasonic successfully works with Exorock -Replaced phototransistor with KY-026 infrared sensor -Successfully achieved right, left, and backwards movement using pulse width modulation -Wi-Fi-Shield working, finished writing CSS code for website	-Radio approach not working which is why pivoted to magnetic sensor; will have to return to radio -Create amplification for hall sensor -Clean noise being produced by infrared sensor -Motor code requires rewrite to enable more movement ability (moving diagonally and orbiting)

		-Built hall sensor circuit which successfully detects magnetic fields and identifies polarity	-Website keeps crashing when all CSS added, fix this
#3: 24 – 26 May	-Finish magnetic hall sensor -Finish infrared sensor -Enhance movement, begin looking into controlling movement over Wi-Fi	-Finished differential amplifier with two hall sensors, magnetic sensor complete -Modified sampling of frequencies and got the sensory working for the second frequency, infrared sensor complete -Started tweaking previous radio sensor -Got rover moving over Wi-Fi	-Finish radio sensor -No progress on fixing the website crashing. Not able to change IP address, create a second website and write JS to communicate between second website and Wi-Fi shield webserver
#4: 27 – 30 May	-Finish radio and website	-Radio sensor circuit works but with 5 op amps, we will order stronger op amp -Replaced website with android app which is built on MIT App Inventor -Ordered swivel wheel and op amp for radio sensor	-Begin planning out chassis -Finish radio sensor
#5: 31 May – 2 June	-Chassis -Radio Sensor	-Radio Sensor complete -Different chassis ideas have been documented and we have chosen the idea we want to laser cut	-Design chassis idea on Autodesk and laser cut -Combine all 4 sensors to identify rock -Start writing rest of report
#6: 3 – 6 June	-Write code to identify rock -Chassis -Start report	-Finished code -Successfully laser cut chassis -Copied and pasted all the write up from OneNote and got 3000 words	-Continue writing report -Sensors will occasionally stop working and require tweaks, need to finalise the sensors
#7: 7 – 9 June	-Perfect Sensors -Continue Report	-Built new breadboard which is much cleaner and has been debugged successfully -Sensors, motors, PDS and future work write up complete	-Solder the combined sensor of the circuit onto perf board -Finish project management, chassis, efficiency, abstract, introduction and conclusion
#8: 10 – 12 June	-Put all sensors onto Chassis -Report	-Successively built entire rover -Completed chassis, project management and efficiency write up in report	-Finalise report -Finalise rover

13 – 19 June	-Finish Report -Continue fine tuning rover	-Report complete	-Ensure all sensors and motors are working with no glitches
20 – 23 June	-Complete rover and prep for demo	-Rover complete and demo prepped	Project Complete!

Meetings

Members of the team came to campus at different times during the day and worked in different groups. Occasionally work would be done remotely. To ensure that there was clear communication with all the team members and that everyone was updated with the project's progression weekly Teams meetings were held at a time when everyone was available.
The overall project progress can be found in the appendix. [D]

Future Work

Though the final rover completes the brief, there are a few things that could have been differently now that the project is complete.

Infrared Sensor

Due to random noise in the sensor signal, the current program used to measure the frequency of incoming pulses takes multiple samples to improve reliability. Although this works, the time taken to execute the program is drastically increased. The effective range of the sensor is also reduced as the occasional correct reading is ignored if it is amongst a few invalid values at longer distances. With more time to research and implement a verification algorithm, a better solution could be found.

Given a greater budget, a more suitable infrared sensor could be bought. The current sensor needs to be pointed directly at the Exorock to detect the IR pulses, however there are sensors available that have a wider viewing angle and greater sensitivity. The rover would not need to be as precise with its placement to work.

Radio Sensor

One possible improvement that could have been made to the radio detection and processing system is to use a larger coil antenna as opposed to an inductor. A coil with sufficient turns would likely result in a greater detection range. The inductor was chosen since its small footprint made it easy to mount on the rover. However, given more time, a sturdier support structure for a large coil could have been manufactured, allowing the antenna to be securely mounted on front of the rover, and be driven to hover directly above an Exorock.

A Schmitt-trigger could have been used to clean the final signal that was passed into the microcontroller. This would have removed any noise and ensured that the voltage level was large enough to be interpreted as a digital high input. This could have potentially improved the range as the further the inductor moves from the Exorock, the lower the voltage level becomes. For the system to work as intended, the voltage produced by the envelope detector must exceed 2V. At larger ranges, a clear, period signal is still visible, albeit at a lower voltage than required. The introduction of a Schmitt-trigger could have converted this to a perfect square wave, with the same time period, whilst guaranteeing it reaches the 2V threshold. In the end, it was decided that a Schmitt-trigger was unnecessary as the short range of other sensors meant the rover had to get close to an Exorock anyway, meaning the inductor was already within range. If the range of the other sensors was improved, this improvement could be implemented to improve the overall effective range that the rover can identify rocks at.

To improve the reliability of the rock identification system, the rover could also have measured the carrier frequency of the radio signal. Currently, only the modulating frequency is measured, which, along with the other sensors, is sufficient to uniquely identify all the rock types. Incorrect measurements could be caused by noise, faulty sensors, or damaged circuits. Under the right circumstances, this would result in the program identifying the incorrect rock type, with no indication that any part of the system is defective. If more data were to be collected by the rover, it would be able to recognise that the combination of measurements is not emitted by any of the possible rock types, and therefore could alert the user that part of the system is malfunctioning. For example, a modulating frequency of 239 Hz is used by

both Lathwaite and Xirang rock types. If the magnetic sensor is faulty and assumes there is no magnetic field, the program would incorrectly identify the rock as Lathwaite. However, if the carrier frequency were to be measured, and returned a value of 89 kHz, the program would realize there is no rock with these properties and could then indicate this to the user. It was decided not to do this as it would increase size and power consumption. A future version of the rover could use bandpass filters to detect the two possible carrier frequencies. A signal would only be output by each filter if the specific carrier frequency were being emitted. The output of the filter could then be passed into a circuit like an envelope detector, where a capacitor is used to convert the high frequency AC to a DC signal that can then be read by a microcontroller.

Ultrasound Sensor

Although the ultrasound sensor is arguably the most reliable of the four, the process of repurposing a HC-SR04 ultrasonic distance sensor to suit our needs lead to some code which is quite difficult to understand. This is because the code effectively measures the time an imaginary outputted ultrasonic burst takes to return however we are measuring the time it takes for the ultrasonic from the Exorock to be received which still requires the non-existent ultrasonic transducer to be triggered.

Also, the measured sweep angle of the sensor is 15 degrees, and our solution has a range of ~20cm so if there is an ultrasonic source within said range ahead of the target rock a false positive could be received. If this turns out to be an issue a 3d printed shroud could be fabricated to restrict the measured angle or we could only take measurements when the sensor is fully obscured by the target rock.

To make this system easier and code easier to understand it would be possible to purchase an ultrasonic sensor that is purpose made to just receive an ultrasonic signal and not to calculate distance.

Magnetic Sensor

Originally, a single hall sensor was used because this was enough to detect the magnetic field and find polarity. However, the range was quite low, so it was decided to amplify the output of the sensor. The differential amplifier method of amplification was chosen, and an additional sensor was added. Though this method worked, it is possible to able to detect the magnetic field and amplify the output with a single amplifier. Had an inverting/non-inverting amplification approach been used, an additional sensor would not have been needed. This would have reduced the power required for the circuit and made the overall rover cheaper. It would also simplify the circuitry making the breadboard more efficient and easier to build/debug.

Motor and Control

Instead of using MITAppInventor, taking a deeper look into hosting a local HTML webpage or application and sending instructions to the Arduino using the relevant IP address was another equally valid method. Through this, the team could have learned to develop webpages and code in JavaScript which would prove to be useful in the future. Furthermore, the ability to change the speed of the motors could be added to create a more precise driving experience.

Conclusion

Overall, this project was a success, completing the brief fully. Minor improvements to speed and efficiency could be made as shown in the future work section, but these would require more time and iterations of the design.

Problem Design Specification Results

Key consideration	Requirements pass/fail	Issues/challenges
Reliability	• Need each sensor to work every time it is being tested: **pass** • Need the outcome of that sensor to be accurate every time it is being tested: **pass** • Need the movement controls to work every time they are stimulated to work through app: **occasionally gets overloaded if there are too many rapid inputs but is responsive otherwise.** • Need chassis to not break/ fall apart at any time: **pass** • Need app to be able to read information given to it from rover to reliably: **pass** • The correct rock should be returned 100% of the time: **pass if the rover is arranged so the sensors are positioned above the rock**	• Sometimes, there was interference from other objects/sensors in the room, otherwise all sensors work all the time • Movement works exactly as expected when expected
Safety	• Ensure metro board is not being overloaded and that all its inputs are limited to 3.3v as required: **pass** • Make sure that all the nuts and bolts are fastened tightly so that they do not drop and cause the chassis to break exposing us to broken shards: **pass** • Ensure workspace is tidy and that no components are near the edge of the bench to try and prevent anything falling onto the floor and leading to trips/ falls: **fail**	• Initially had a couple of issues with the metro board input being bigger than 3.3V but that was quickly dealt with through a potential divider • While drilling the Chassis, the right clamps to fasten the board securely on the desk could not be found. An alternative method to drill the board was used, but this was done successfully • Going up to work in the robotics lab meant the sensors and motors had to be moved around and some

	need to improve on this in future as this sometimes led to confusion of components • When using soldering iron, ensure you do not touch the hot part and try not to breathe in the toxic fumes: **pass** • When testing movement make sure area is clear so that the rover doesn't bump into anyone, causing them to trip and fall: **pass** • Ensure that no food or drink are consumed near any circuitry so that in the unlikely event of a spill nothing gets broken: **pass**	nuts and wires got loose/dismantled. However, learning from this, the relevant connections were tightened the and ensured the team was more careful in the transportation process
Performance	• Be able to control the rover and manoeuvre it around rocks and obstacle: **pass** • Be able to detect each signal and therefore rock: **pass** • Be able to communicate with rover through app to tell us what type of rock we are seeing: **pass** • Maximum scan time should be <5s: **pass, worst case 2s if no signals are detected.** • Speed should be >0.1m/s: **pass, ~0.2m/s** • The sensors range should be >10mm: **pass, shortest range is magnetic ~20mm**	• Initially found it difficult to communicate with rover because webserver kept crashing • When multiple buttons are repeatedly pressed quickly, the stop code no longer works, and app becomes unresponsive
Cost	• Extra products/sensors must not exceed £60: **pass expenditure used for final product £31.88** For more detail see budget section	• A few sensors had to be bought which increased the cost of the rover
Disposal and Sustainability	• Acrylic used will be able to be recycled: **pass** • Will safely disposes of batteries: **N/A (yet)** • Sensors /metro board/ breadboard/ motors can all be recycled by use in another project: **pass** • Quick and easy disassembly due to minimal soldering: **fail due to soldering majority of wires on**	• The ultrasonic sensor has the transmitter module de-soldered so it is limited in the type of projects they can be reused in

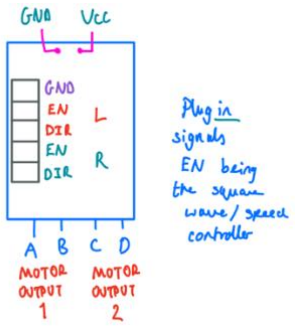
	Veroboard due to greatly increased reliability of circuits.	
Testing	• Test range of sensors: **pass** • Test range of motion on rover: pass • Test responsiveness of app: pass • Test durability: **pass** • Test portability: **pass** • Check that the code is robust and efficient: **pass** • Test overall system at least 20 times: **pass**	• Range caused an issue early on but was quickly resolved
Materials	• Durable: **pass** • Low cost: **pass** • Readily available: **pass, used acrylic and m3 bolts.** • Parts in contact with heat sources (e.g., Motor contacts) should be able to withstand heat: **pass for the fairly low temps of the small dc motors.** • Shouldn't interfere with sensors: **pass, didn't use any conducting materials e.g., carbon fibre, metal s** • Should be able to withstand the extreme temperatures on the moon: **fail microprocessor operating range is ~-40°C to 85°C** • Should be relatively light: **pass could have used lighter materials at greater expense.**	• Had to find perfect trade-off between durability/low cost and everything else • Couldn't find a strip heater to bend the acrylic
Ergonomics	• Controller to move rover must be comfortable for the person using it: **pass**	• Initially had issue with website because making it user friendly made the webserver crash. This was resolved by using an android app instead.
Size and weight	• Per the brief it must be lightweight <750g: **pass, rover is ~500g depending on weight of AA batteries used.** • Must be able to manoeuvre smoothly and efficiently around rocks/obstacles : **pass** • The aim is to be the quickest rover so will need to make it as	• The design wasn't necessarily the most weight efficient because there is an additional structure however, the weight is not excessive and in doing so the size is a lot narrower making it easier for the rover to manoeuvre the terrain

	lightweight as possible as extra weight is equal to extra time: **pass**	
Environment	• Resilient to obstacles/terrain: **pass** • Resilient to external signals/light pollution: **pass** • Must be very stable enough to bump into other rovers or rocks: **pass**	• The terrain is still not fully built so there is no knowledge of what it will look like
Aesthetics/ appearance/finish	• Not really an important part of the design • Not a priority • Can not interfere with any of our sensors: **pass** • The more unique it is the easier it will be to identify it on the track: **pass it is easily identifiable**	• Tried to balance between making it as narrow/light as possible and looking good • Had to sacrifice aesthetics to make sensors function in the right place.

Appendix

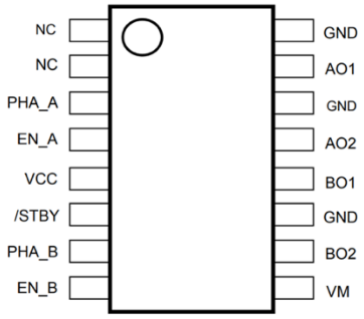
[A]

A simplified diagram of the motor controller was drawn to explain how each pin works.



[B]

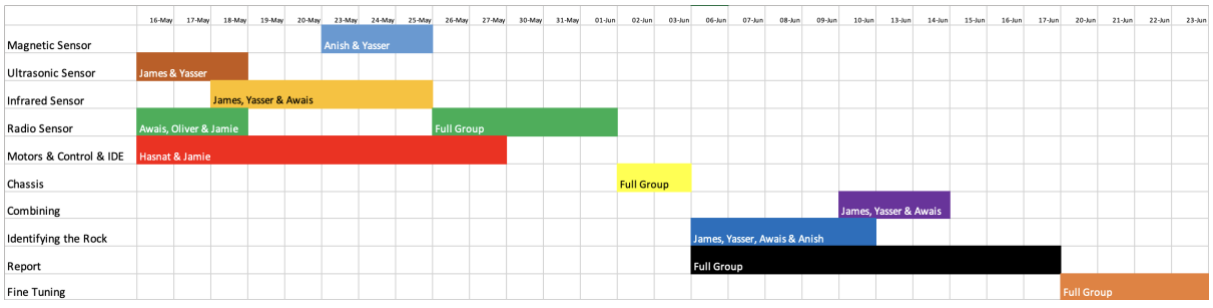
Pin No.	Pin name	Functional description	Remarks
1	NC	Not connected	Please do not connect any pattern
2	NC	Not connected	Please do not connect any pattern
3	PHA_A	Control input pin for Ach (1)	See the table "Input/Output functions".
4	EN_A	Control input pin for Ach (2)	See the table "Input/Output functions".
5	VCC	Power supply pin for logic block	VCC=2.7 to 5.5V
6	/STBY	Standby input	See the table "Input/Output functions".
7	PHA_B	Control input pin for Bch (1)	See the table "Input/Output functions".
8	EN_B	Control input pin for Bch (2)	See the table "Input/Output functions".
9	VM	Power supply pin for output	VM= 2.5 to 15.0 V
10	BO2	Output pin of B phase (2)	Please connect with a motor.
11	GND	Ground pin	
12	BO1	Output pin of B phase (1)	Please connect with a motor.
13	AO2	Output pin of A phase (2)	Please connect with a motor.
14	GND	Ground pin	
15	AO1	Output pin of A phase (1)	Please connect with a motor.
16	GND	Ground pin	



[B] from [8]

[C]

[D]



References

[1]
Wikimedia.org, 2022. https://upload.wikimedia.org/wikipedia/en/thumb/3/32/Logo_for_Imperial_College_London.svg/1280px-Logo_for_Imperial_College_London.svg.png
[2]
Wikimedia.org, 2022. https://upload.wikimedia.org/wikipedia/commons/thumb/c/c5/Shield_of_Imperial_College_London.svg/1200px-Shield_of_Imperial_College_London.svg.png
[3]
"49E Hall-Effect Linear Position Sensor," *http://www.pst888.com/*. https://p.globalsources.com/IMAGES/PDT/SPEC/440/K1139513440.pdf
[4]
https://www.diodes.com/. https://www.diodes.com/assets/Datasheets/AH49E.pdf
[5]
"Operational Amplifiers." https://learn-eu-central-1-prod-fleet01-xythos.content.blackboardcdn.com/
[6]
"Ultrasonic Sensor HC-SR04 and Arduino Tutorial," *HowToMechatronics*, Aug. 04, 2018. https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/
[7]
"How To Mechatronics," *HowToMechatronics*. https://howtomechatronics.com/
[8]
"MCP6002," *Microchip Technology*. https://www.microchip.com/en-us/product/MCP6002
[9]
Learn.adafruit.com, https://learn.adafruit.com/adafruit-metro-m0-express/overview
[10]
"MCP6291/1R/2/3/4/5." https://docs.rs-online.com/2f4a/0900766b813813f6.pdf
[11]
"TC78H620FNG DUAL-BRIDGE DRIVER IC," Toshiba, Nov. 2016.
[12]
DroneBot Workshop, "Controlling DC Motors with the L298N H Bridge and Arduino," *YouTube*. Mar. 11, 2017. Accessed: May 28, 2022. [YouTube Video]. Available: https://www.youtube.com/watch?v=dyjo_ggEtVU
[13]
SparkFun, "Pulse Width Modulation - learn.sparkfun.com," *learn.sparkfun.com*. https://learn.sparkfun.com/tutorials/pulse-width-modulation/duty-cycle
[14]
M. Ansar, "How To Build a Wi-Fi Based Robot with Android Application Control | Android App with MIT App Inventor," *www.youtube.com*, Dec. 24, 2021. https://www.youtube.com/watch?v=S-67_HUxrOg
[15]
"RS PRO Ultrasonic Proximity Sensor - Barrel, 0 → 300 mm Detection | RS Components," *uk.rs-online.com*. https://uk.rs-online.com/web/p/proximity-sensors/2370799 (accessed Jun. 07, 2022).
[16]
Shopify.com, 2022. https://cdn.shopify.com/s/files/1/0176/3274/products/ultrasonic-distance-sensor-hc-sr04-the-pi-hut-100284-1118255996_800x.jpg?v=1646101268
[17]

“Hall Sensor,” *The Pi Hut*. https://thepihut.com/products/hall-sensor?variant=39627167695043¤cy=GBP&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&gclid=Cj0KCQjwwJuVBhCAARIsAOPwGATzPUC8hQkVg_B4fvRZZa6UMm28RbOMf6ZNLDRn6wwhAssYqj2Uyx0aApwLEALw_wcB
[18]
Adafruit.com, 2022. https://cdn-shop.adafruit.com/970x728/4678-01.jpg