Imperial College London

June 19th 2022

# ELEC40006 - Electronics Design Project

# Technical Report

Group: PDMB

Lecturers: Dr Edward Scott and Mrs Esther Perea Borobio

Word count: 6825

Names and CID:

Amor Zhao - 02019680

Ben Marconi - 02028527

Bhavya Sharma - 02022359

Pengyuan Shao - 02039969

Rory Brooks - 02020772

Zeyd Chamsi-Pasha - 01885955

# Table of contents

## Abstract

The purpose of the Engineering Group Design Project was to build a lunar rover which can identify minerals by being remotely controlled across a surface resembling the moon. Our group approached this assignment by splitting up the tasks required to design such a vehicle into four sub-categories: Sensors, Mechanics, Software and Connections and IO interfaces. Acknowledging precisely what tasks had to be completed within these sub-categories and consulting the PDS (product design specification) helped us fairly assign work to each member of our group and plan out internal deadlines. After completing the required tasks from each sub-category, a chassis was designed accordingly and the rover was assembled. Once we had our first prototype, its movement and manoeuvrability was tested around a surface with obstacles. Similarly, it was tested whether the rover correctly identified each of the six minerals it was supposed to detect. This testing phase allowed us to correct any minor issues that were present at the final stages of the design. The result was that by the time we were to demonstrate the project, we had a fully functional, reliable and up to specifications rover to deliver.

## Introduction

For the Engineering Group Design Project, we were tasked to build a lunar rover, whose primary objective is to identify unusual rocks. The rover is required to be remotely controlled through an arena with obstacles, sequentially identifying each rock present in the arena. It does so by analysing the acoustic, magnetic and electromagnetic characteristics of the rocks, interpreting the data according to Figure 1. The detected mineral is then displayed in the same interface used to remotely control the rover. Finally, the rover must be designed while minimising costs and mass, consistent with its purpose of being sent to the moon. This report outlines the research and decision making alongside technical and non technical work that went into building a fully functional final product.

| 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 |

Figure 1: Properties of the Minerals.

## Sensors

### Measuring Infrared

There are four stages in the IR (infrared) detecting circuit, which are receiving the signal, amplification, high pass filtering and converting to digital form using a Schmitt trigger. To receive the signal, an infrared diode was connected in series with a resistor. When infrared light was emitted towards the diode, its resistance would decrease, causing an increase in the voltage across the resistor. In the amplification stage, an inverting amplifier with a gain value of 1000 was used to amplify the signal. However, the actual gain would not exceed 40 as the output voltage had to stay in the range 0-5V. A capacitor was added between the first and second stages to block the input voltage from the 2.5 DC bias. The signal had frequencies 353 Hz and 571 Hz, so a chip LT1366 (from autumn term) was used, providing a gain bandwidth product of 400 kHz.

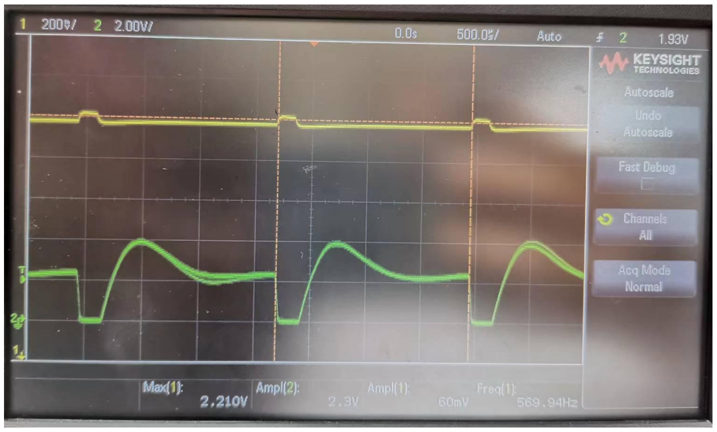


Figure 2: Signal before amplification(yellow) and after amplification(green)

However, the amplified signal was still unstable, making it difficult to set the bias value for the Schmitt trigger. In order to block any noise, a high pass filter was added. After that, the signal could be converted to a square wave using a Schmitt trigger, which was biased at 2 volts. Figure 3 shows the sketch of circuit while Figure 4 screenshot of the signal input for the microcontroller:

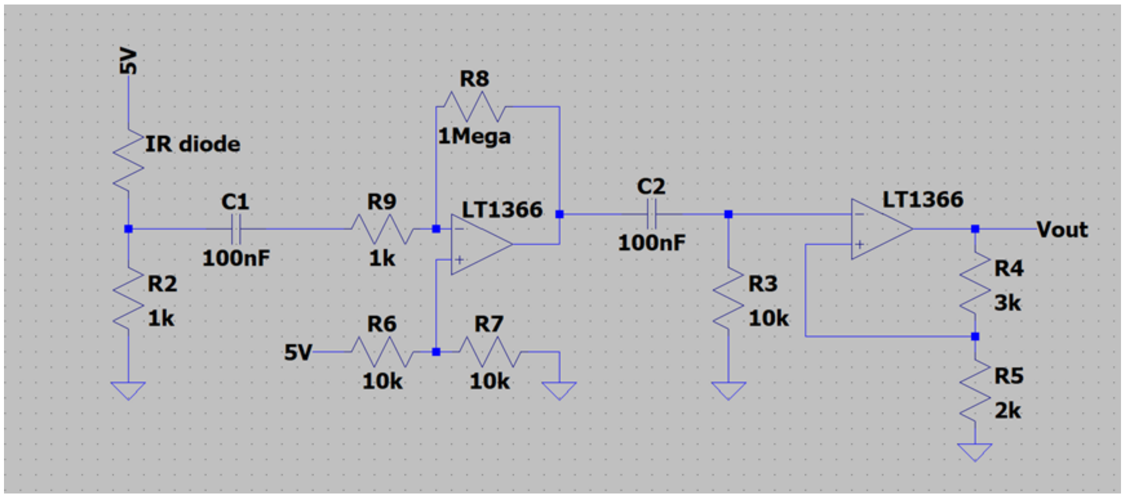


Figure 3: Circuit diagram of the IR sensor

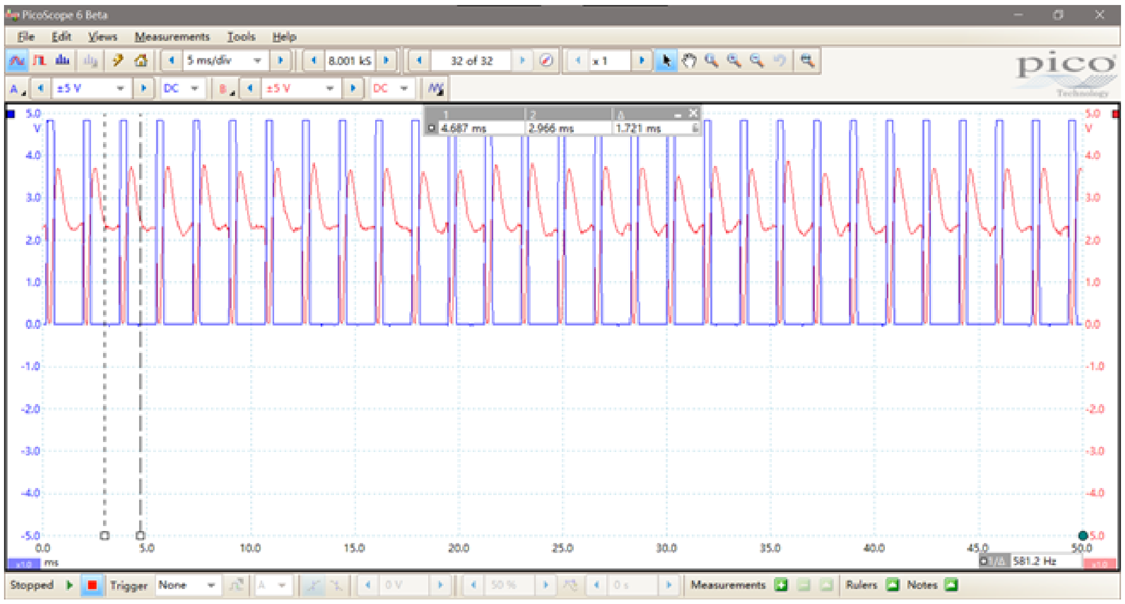


Figure 4: Screenshot of picoscope reading before and after the schmitt trigger

(red: signal output from filter; blue: signal output from schmitt trigger)

The output signal was inputted to the microcontroller that calculated its frequency.

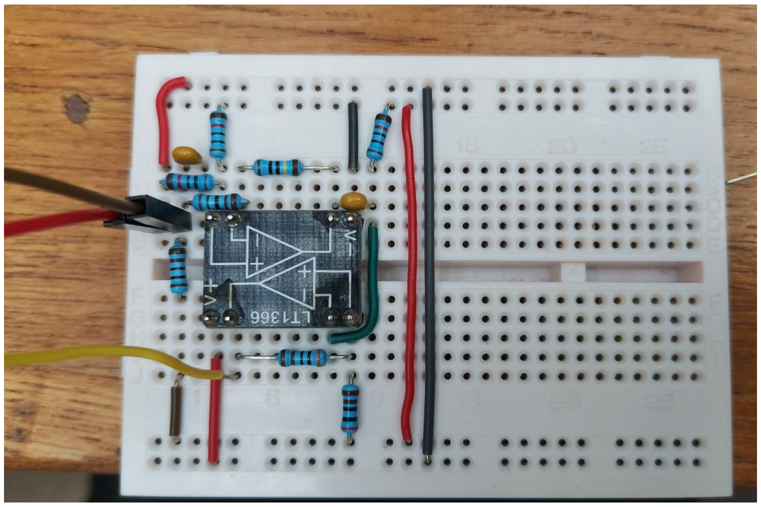


Figure 5: Photograph of physical IR sensor circuit

When testing the sensor with Metro M0 boards, it was noticed that the input pins on Adafruit Metro M0 have built-in diodes which would only allow voltage under 3.3V to go through. In order to make sure that the input signal would have a proper magnitude, an additional 10k resistor was added to the output side, as shown in Figure 6.

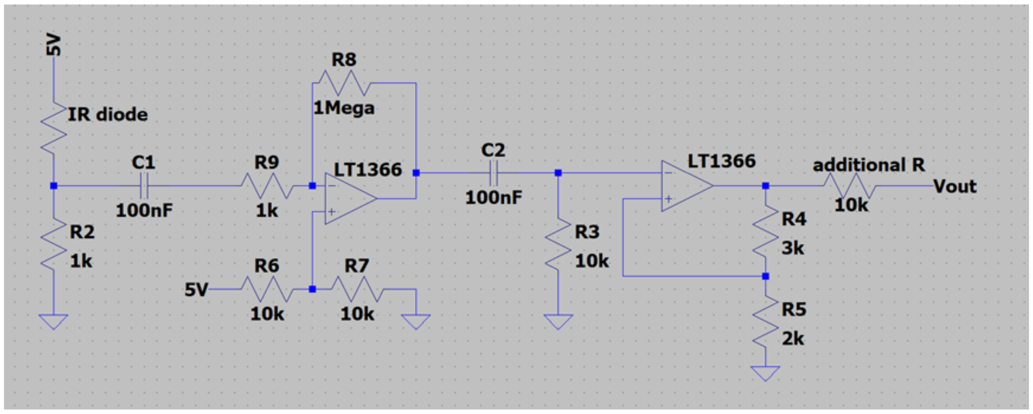


Figure 6: Circuit diagram with the additional resistor

## Measuring Magnetic field

To distinguish between *Adamantine* and *Xirang* the rover must be able to detect the direction of the magnetic field generated by the minerals. A simple way to identify the direction of the magnetic field is using the Hall effect, where a transverse electric field is produced in a conductor when an electric current perpendicular to a magnetic field is flowing through it.[1] Depending on the direction of the electric field generated, we can deduce the direction of the magnetic field.

Rather than building our own circuit, our group decided to use a Hall Effect sensor module, an option we considered more reliable, less complicated and that did not require further amplification. The sensor has an analogue output equivalent to the magnitude of the electric field generated which, when read by an Arduino analogue input, is centred at 766, as shown in Figure 7. Thus the magnetic field direction can be determined with an analogue read, where values greater than and less than 766 represent the different directions of the field respectively.

Once again, software is used to process the output data of the sensor. While the Hall Sensor also provides data on field strength, this is not necessary to differentiate between rocks so we did not make use of this feature.

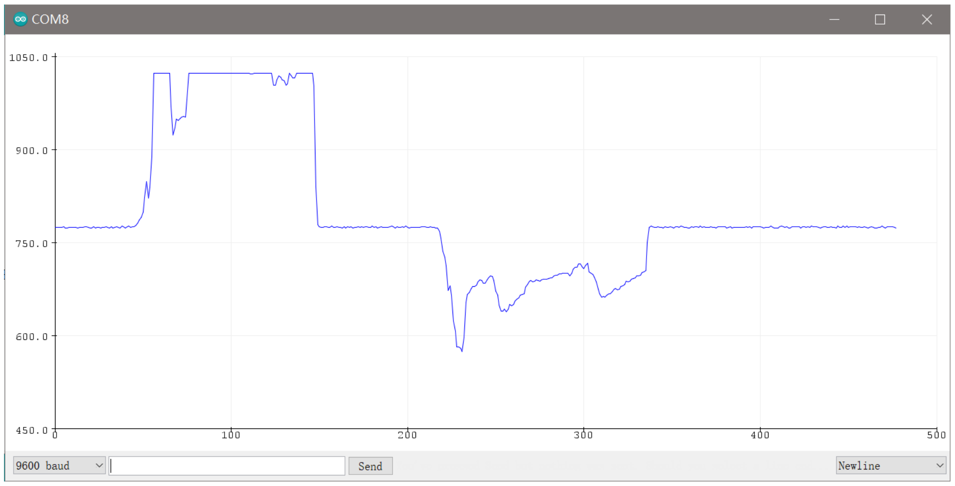


Figure 7: Screenshot of the Arduino Serial Plotter illustrating a changing magnetic field

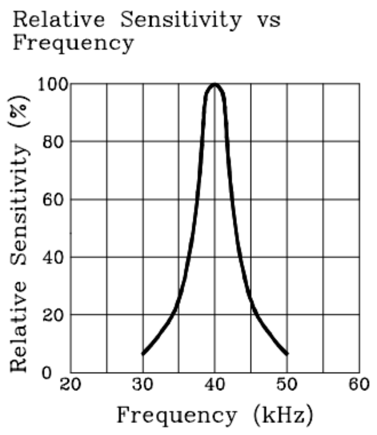
## Measuring Acoustic signals

We are asked to detect acoustic properties in the projects for distinguishing between Gaborium, Netherite, and other rocks. The task of measuring acoustic properties is simplified by the fact that in each case the frequency of the acoustic signal emitted is 40kHz.

[1] Hall, E., 1879. On a New Action of the Magnet on Electric Currents. *American Journal of Mathematics*, 2(3), p.287.

An ultrasonic transducer was used as a sensor to detect the waveform. The ultrasonic transducer acts like a varying current source that converts sound energy to an electrical signal or vice versa. In an ultrasonic receiver, there is a piezoelectric layer[2]. The layer converts a certain amount of pressure/sound vibration into the corresponding streams of electron flow, from which we receive the signal. Each type of transducer has its own built-in resonant frequency. For the one we used (tct-40r), the resonant frequency is 40 kHz, which is perfectly suited for the detection of a 40 kHz ultrasonic wave. The built-in resonant frequency blocks out acoustic signals at other frequencies due to low sensitivity and only picks up the 40 kHz useful signal.

[3]



[4]

Figure 8: Photo of sensor and its Relative Sensitivity vs Frequency

The acoustic circuit we designed includes three stages: receiving the signal, amplification, and peak detection. In the first input stage, ultrasonic signals are received through an ultrasonic transducer.  By connecting one port to ground and another port to output, the transducer outputs an electrical signal at 40 kHz according to the waveform

[2] WatElectronics.com. 2022. *Piezoelectric Sensor, Working, Operation Using Arduino and Applications*. [online] Available at: <https://www.watelectronics.com/piezoelectric-sensor-and-its-operation/> [Accessed 15 June 2022].

[3] Tangle Of Tings. 2022. *TCT40-16R/T Proximity 40kHz Ultrasonic Sensor TX / RX Pair*. [online] Available at: <https://tangleoftings.com/TCT40-16RT-Proximity-40kHz-Ultrasonic-Sensor-TX-RX-Pair_p_122.html> [Accessed 10 June 2022].

[4] TCT40-16R/T datasheet

of the sound. According to what we found, the output waveform directly from the sensor has an amplitude of 20mV, which is too small to be distinguished from noise, so we added an amplification stage.

In the second stage, a negative feedback loop is built for inverting amplification. The feedback resistance ratio is 75, so the entire circuit has a negative gain of 75. V+ is biased at 2.5V for a complete amplification. An output capacitor is added to eliminate DC-biased voltage. After the amplification stage, the output waveform has an amplitude around 160mV - 250mV, larger than 40mV noise, allowing us to distinguish the signal from noise. This is shown in Figure 9.

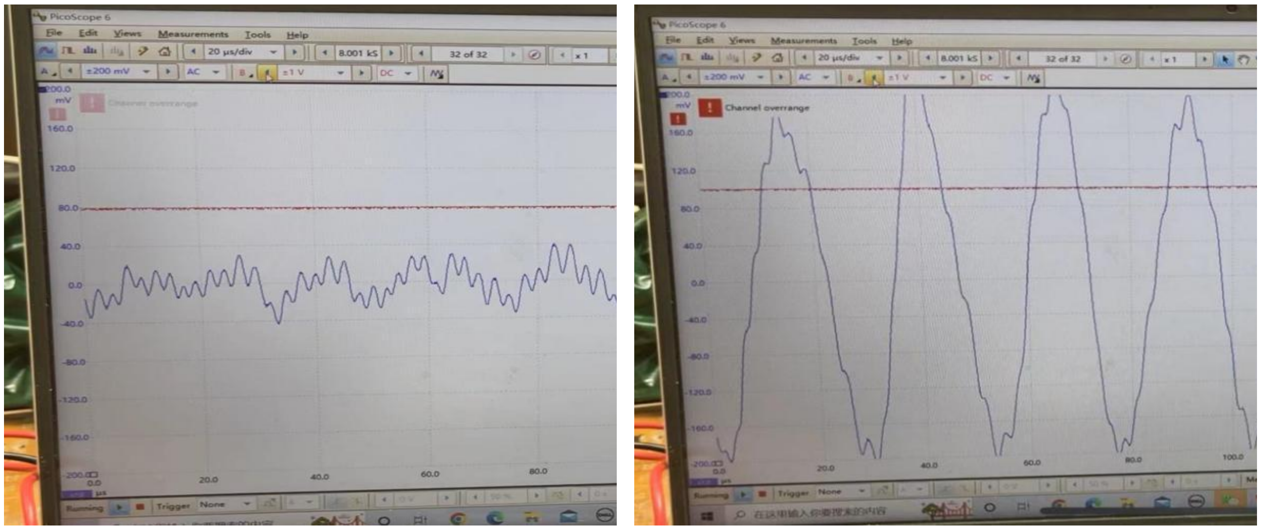


Figure 9: Signal before and after amplification (blue line)

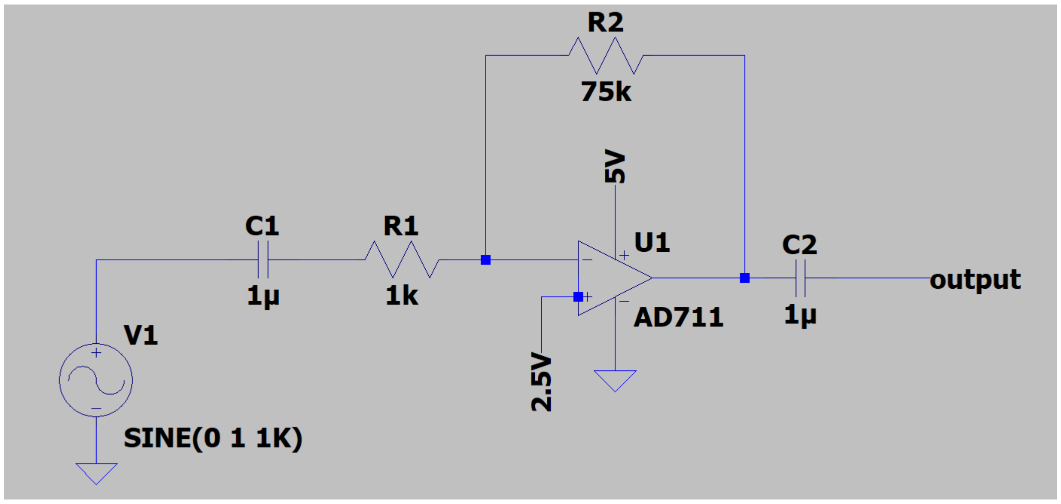


Figure 10: Input and amplification stages of the acoustic sensor circuit

Although after the second stage signal and noise could already be distinguished through the naked eye, further processing must be done for the microcontroller to detect the peaks. Thus, we add a peak detection stage to convert an AC varying sine signal into a DC constant signal.

In the third stage, a half-wave rectifier, made of a diode, a resistor, and a capacitor, is used to perform peak detection. To counteract the 0.6V voltage drop across the diode, we biased the output signal at around 0.7 V. In practice, the voltage drop across the diode is about 0.3V, so the output waveform is biased around 0.4V. To implement a nearly constant output waveform, a large time constant is chosen (1s), larger than one period (25 μs), with a capacitor value of 1μF and a resistor value of 1MΩ. As a result, the capacitor has a slow discharging rate and barely discharges until the next cycle. The output signal is a constant DC voltage of the peak, shown in Figure 11. If there is a 40 kHz signal, the DC voltage from the acoustic signal is around 200mV (0.6V together with the biased voltage) , if there is no signal, the DC voltage from the acoustic signal is around 40mV or smaller, depending on the peak of the noise (0.45 together with the biased voltage), shown in Figure 12.

Initially, we tried to implement a full-wave rectifier. However, this required the use of four diodes and a voltage drop of 1.4 V. This proved to be inefficient, so we opted to use a half-wave rectifier as it accomplished the same task while reducing the complexity of the circuit and cost.

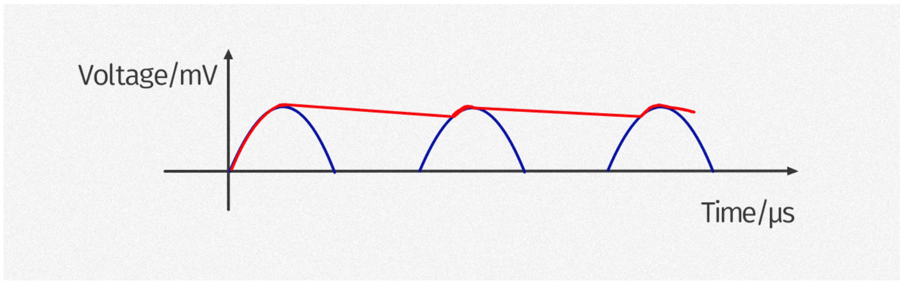


Figure 11: Half-wave rectifier

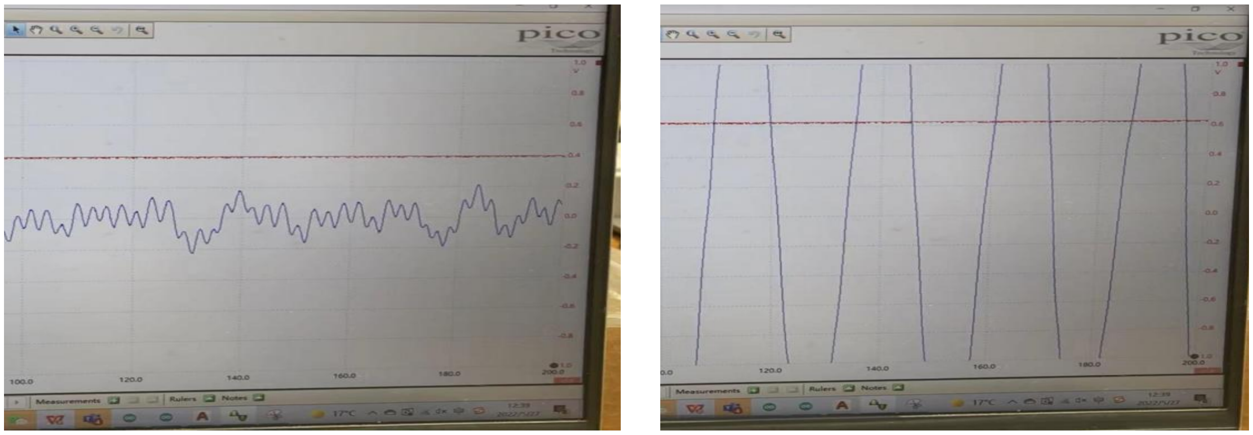


Figure 12: Signal after peak detection (red line)

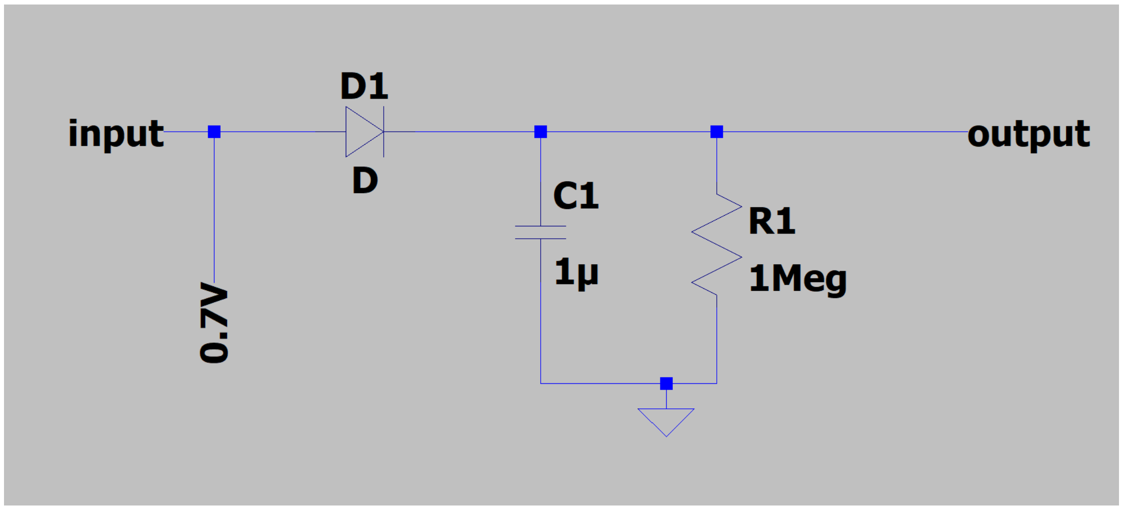
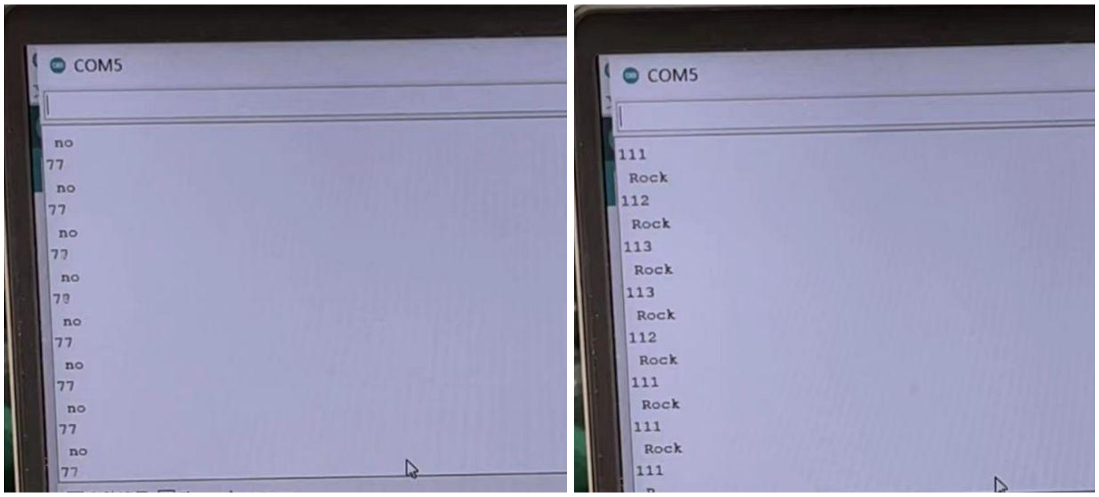


Figure 13: Third stage of the acoustic sensor circuit

For the software to detect the signal, the output signal from the third stage was connected to an analog pin (A0) on the metro board. Using analog read, the arduino would convert the amplitude of the signal between 0-5V into a number between 0-1023. A threshold value (80) was set to determine whether there is an acoustic signal or not, shown in Figure 14. If the value was above 80, a rock would be considered as detected. If not, there is no rock detected.



```
COM5

no
77
no
77
no
77
no
78
no
77
no
77
no
77
no
77
```

```
COM5

111
Rock
112
Rock
113
Rock
113
Rock
112
Rock
111
Rock
111
Rock
111
```

No signal detected Signal detected

Figure 14: Signal captured by arduino

The acoustic signal detected by the circuit is strong. The signal could be detected from a distance of 20 cm away pointing to the rock.

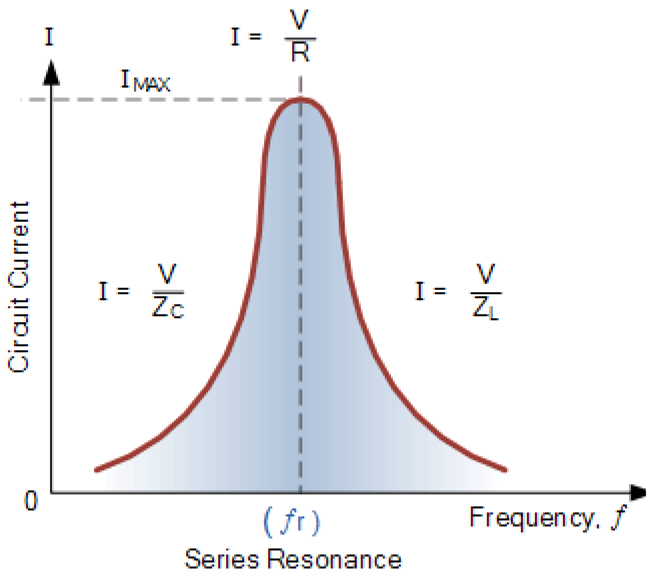
## Measuring Radio Waves

Out of all the mineral properties that we had to measure, radio waves proved to be the most challenging. The following steps were undertaken in order to detect the radio wave frequencies and identify the rocks with the corresponding properties.

1. Capture the radio signal for the specific carrier frequencies.
2. Amplify the signal so that it can be processed by the Arduino.
3. Demodulate the radio wave.

To receive radio signals an antenna was created using a coil of wire. An electromagnetic wave with a changing field passing over a coil of wire will induce a small voltage across it, as per Faraday's law of electromagnetic induction. This introduces a signal into the circuit.

A resonant circuit was created using appropriate capacitors and the inductor formed by the coil (inductance: 120 micro-Farads) with corner frequencies of 61kHz and 89kHz which are the carrier frequencies of the radio waves. As per the theory, at resonance, the impedance is infinite (peak of the resonance graph) therefore the voltage in the circuit will also be higher. Therefore, to achieve resonance for the 61kHz frequency another capacitor was placed in parallel with the 27nF capacitor and turned on using the switch. [5]



$$LC = \frac{1}{(2\pi)^2 \times (frequency)^2}$$

$$120\ microFarads \times Capacitance = \frac{1}{(2\pi)^2 \times (61000)^2}$$

$$Capacitance = 58nF$$

$$LC = \frac{1}{(2\pi)^2 \times (frequency)^2}$$

$$120\ microFarads \times Capacitance = \frac{1}{(2\pi)^2 \times (89000)^2}$$

$$Capacitance = 27nF$$

Figure 15: Graph and calculations indicating resonance peak with very high impedance.

In order to be efficient with the space on the breadboard, only one circuit was built with capacitors for 89kHz and a second set of capacitors was included in parallel for 61kHz. Switching between the capacitors for the two frequencies was done by using a bipolar junction transistor as a switch which requires the transistor to be in saturation mode. As per the data sheet for BC547B, maximum saturation base current was no more than 5mA, hence a base resistance of 560Ω was used with 3.3V supplied from the microcontroller and a large collector resistance of 3kΩ to prevent the current affecting the resonance circuit.

[5] Electronic Tutorials. 2014. *Series Resonance Circuit*. [online] Available at: <https://www.electronics-tutorials.ws/accircuits/series-resonance.html> [Accessed 13 June 2022].

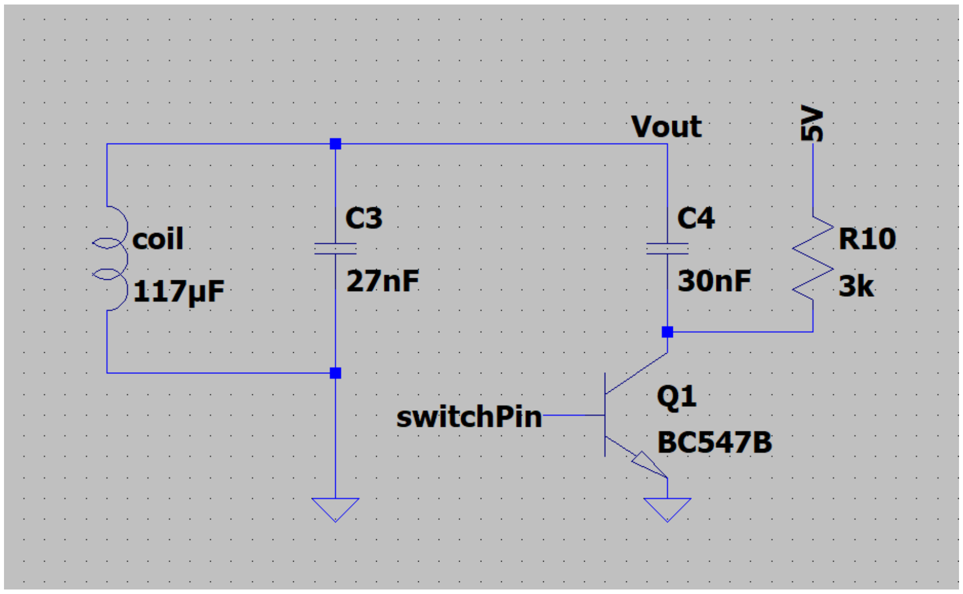


Figure 16: Input stage of the radio wave detection circuit

This voltage however is not high enough for the Arduino, therefore needs to be amplified. Hence, a simple inverting amplifier circuit was included as the next stage with a gain of approximately 20, shown in Figure 17.

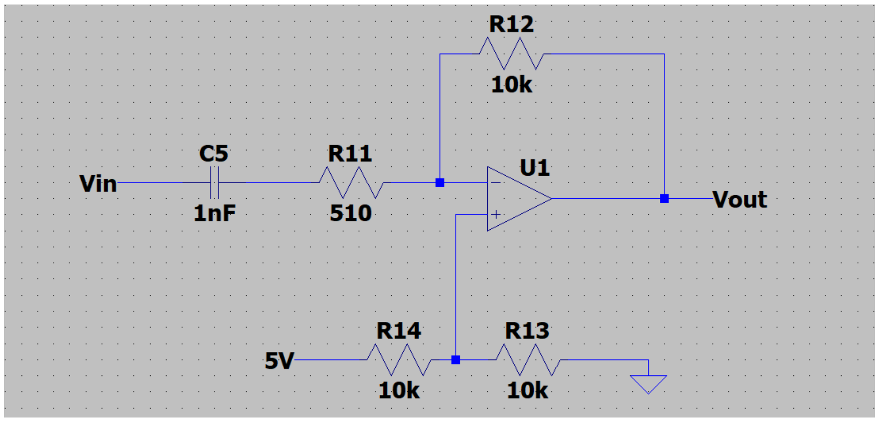


Figure 17: Amplification stage of the radio wave detection circuit

An envelope demodulator was built using a diode which acts as a rectifier (i.e. only half of the signal is detected). Initially an LED was used however this didn't work since it has a very high turn on voltage and was replaced with another non-light-emitting silicon diode with a lower turn on voltage. The capacitor (1nF) filters out high frequencies and the resistor (13k) is chosen such that the RC factor is between 1/wc and 1/(2piB). Since we needed RC to be much greater than 1/wc we set it equal to 5/wc.

The output signal voltage from demodulation was 2-3V with a frequency of 151 Hz or 240Hz. Since the arduino works best with a 0V or 5V input, a Schmitt trigger circuit was used to adjust the range and filter out noise since it will produce a digital signal. If the input rises above 2.5V, then the output rises to the maximum value of 5V. If the input decreases below 2.5V, then the output falls to the minimum value of 0V. The Arduino was programmed to detect the frequency of the demodulated signal giving the value of the modulating signal.

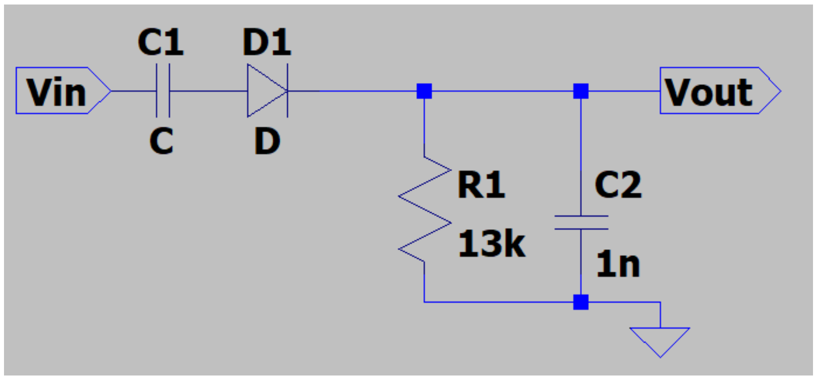


Figure 18: Diode demodulation circuit schematic

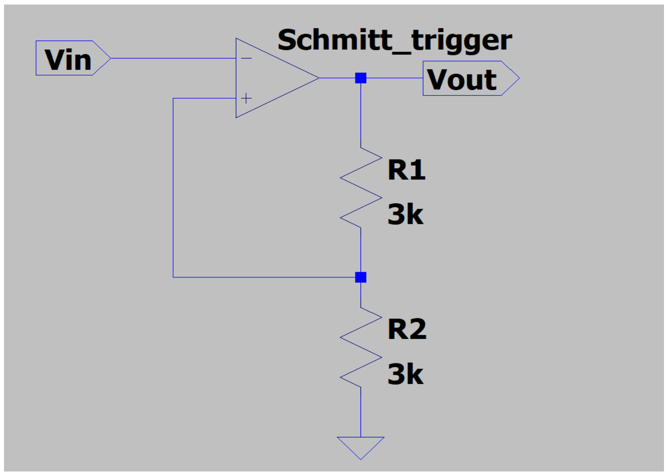


Figure 19: Diagram for Schmitt Trigger circuit

In summary, the radio wave carrier frequency is known through whether the signal appears when the switch is off (89kHz) or on (61kHz) and the modulating signal frequency is calculated in a program, giving the combination of carrier and modulating frequencies to be used for analysis.

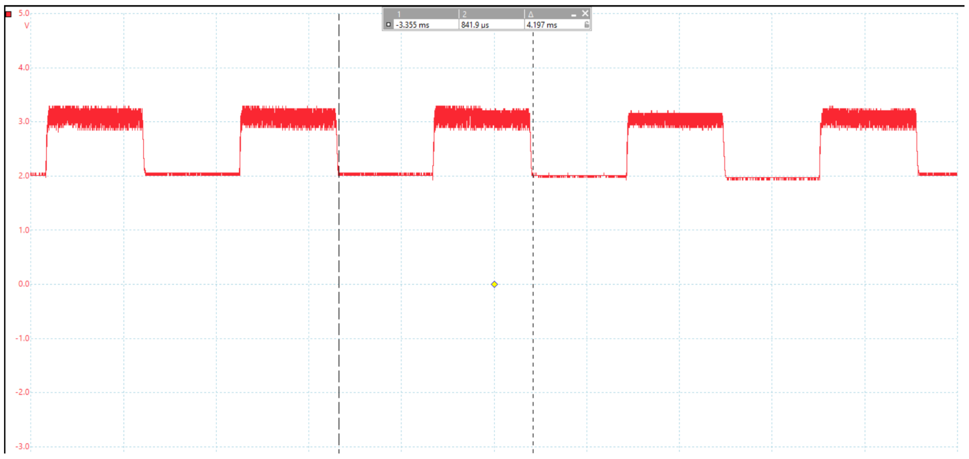


Figure 20: Output before Schmitt trigger (from demodulation circuit) of frequency 238.27Hz

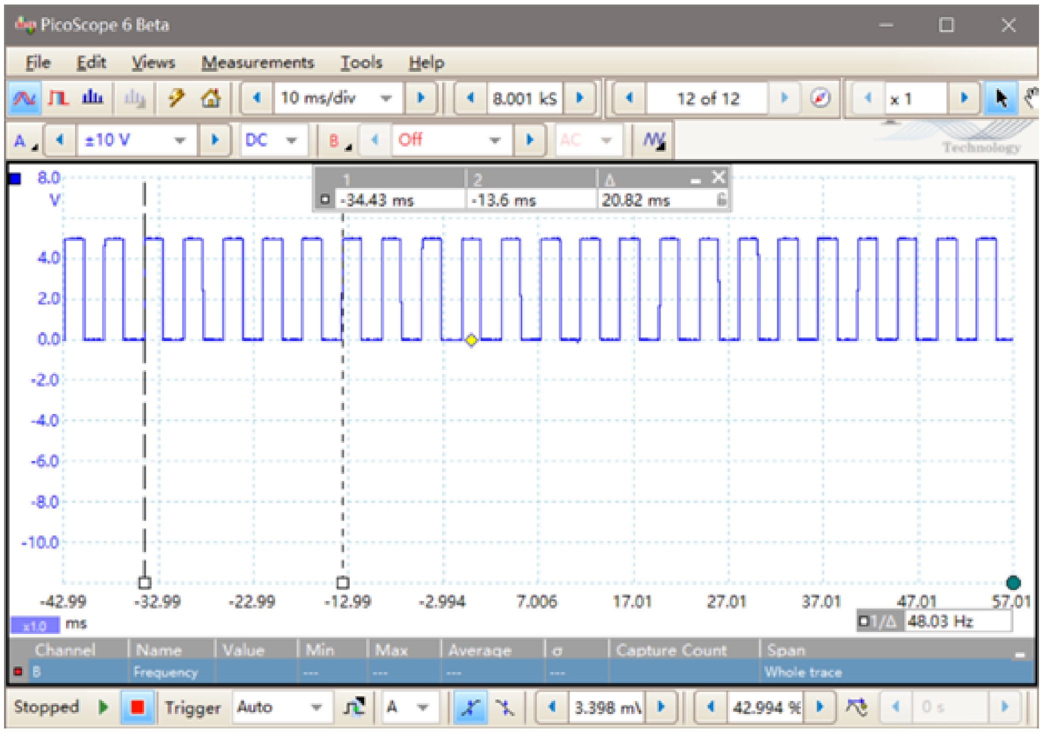


Figure 21: Output after Schmitt trigger: voltage at either 0V or 5V - without noise

Frequency: 5/(20.82m) = 240.15Hz

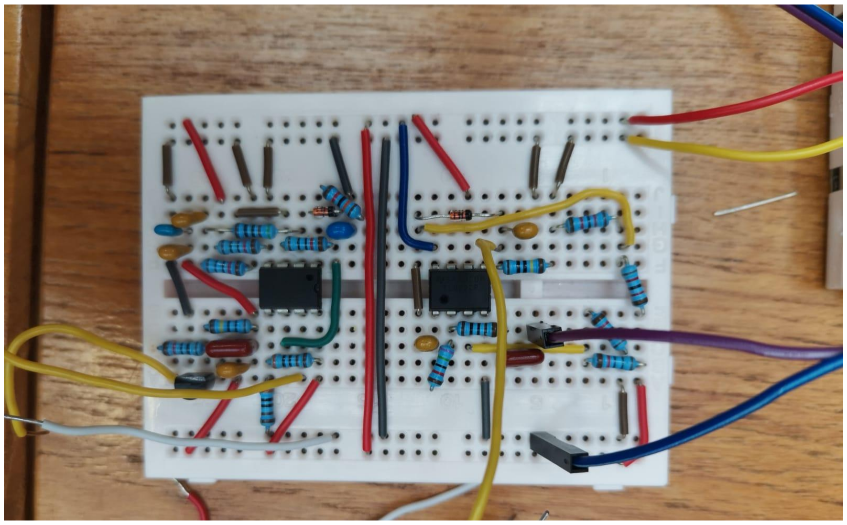


Figure 22: Photograph of physical real circuit (left: radio sensor; right: acoustic sensor)

## Connections and IO interfaces

### Connections

An Adafruit WI-FI shield was connected to the Adafruit Metro M0 to allow the microcontroller to act as web page server after connecting to a WI-FI network. The setup function of the Arduino sketch prints the IP address in which the HTTP web server is started, allowing us to access the web page using a browser. The device that accesses the web page must be connected to the same WI-FI network as the microcontroller. The user sends HTTP requests of type *Get* to the microcontroller to control its movement. An example of a request is shown for ‘backwards’:

```
function backwards(){xhttp.open("GET", "/backwards"); xhttp.send();}
```

When the Rover detects a mineral, it communicates with the web page using the *send* method to display the mineral’s name. For example, when Adamantine is detected, the conditional in Code 2 gets executed, and the plain text “Adamantine" is sent to the server.:

```
if ( radio_151 > 7 )
    {
      server.send(200, F("text/plain"), F("Adamantine"));
    }
```

To display this in the web page, a *span* element is nested within a *div* element. When text is sent to the server by the microcontroller, the state changes, updating the text displayed in the textbox:

```
<div  class="textbox">
      <span id="state">Awaiting scan request</span>
</div>
```

Javascript is used to configure this:

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {if (this.readyState == 4 &&
this.status == 200) {document.getElementById("state").innerHTML =
this.responseText;}};
```

## Input/Output Interface

The HTML webpage mentioned above acted as both the input and output interface for the Rover. Figure 23, in the following page, shows what the website looks like and what each button does. The four arrows are used to control the direction of the movement of the rover. The HTML *buttons* send HTTP get requests through the *id* and *onclick* parameters, as shown below for the backwards button. The arrow shown on the button is included as a *div* within the button of a predefined class.

```
<button id="back" class="button" onclick="backwards()">
      <div class="button__arrow button__arrow--down"></div>
</button>
```

They can be used either by clicking them using a mouse or pressing W, A, S and D on a keyboard. No stop button is required, as the movement of the rover when a button is clicked lasts 100 milliseconds. To achieve continuous motion, the user must press and hold the keyboard commands. The scan button is pressed once the Rover is close enough to a mineral, and this will result in the textbox displaying the name of the detected mineral. When the Rover is not near a rock, the default display message is “Awaiting scan request”.

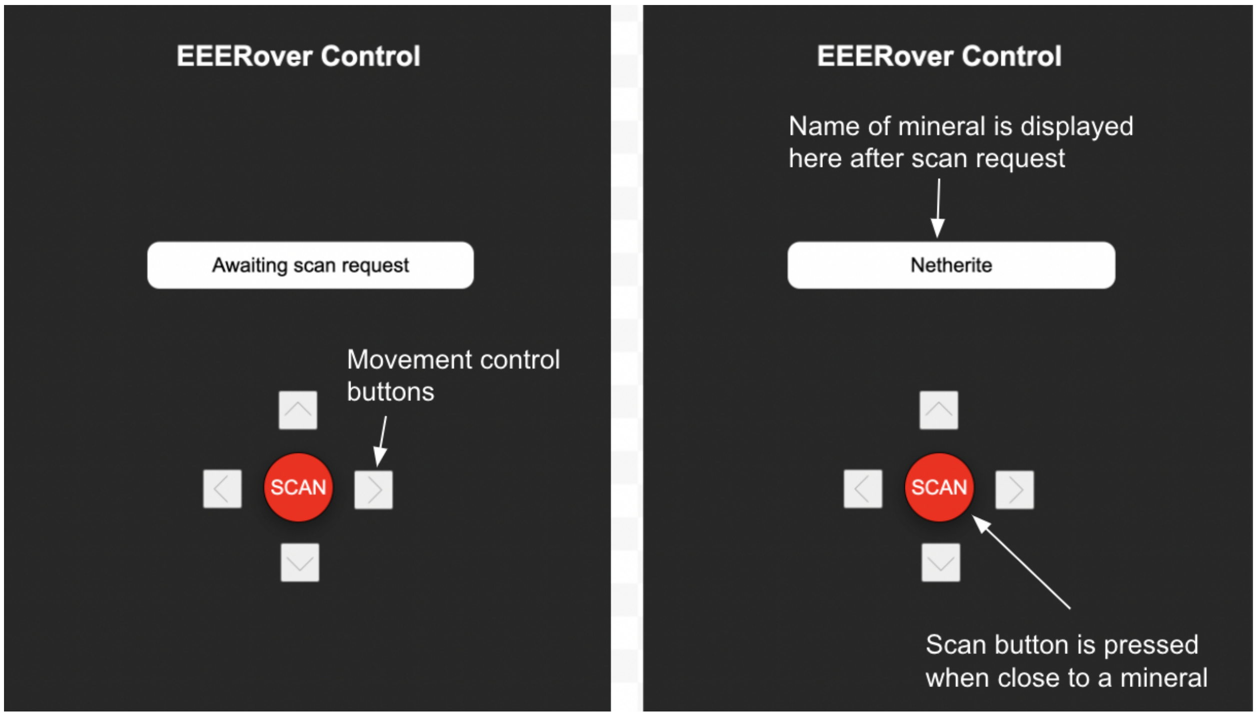

EEERover Control
Awaiting scan request
Movement control buttons
SCAN
EEERover Control
Name of mineral is displayed here after scan request
Netherite
SCAN
Scan button is pressed when close to a mineral

Figure 23: Input/output interface of the rover

## Design and Structure

### High Level Design

To come up with a high level design, we broke down the problem into the sub-sections and came up with ideas to address them. We had 6 different minerals to be identified and this could be done by detecting the different combinations of properties shown in Figure 1 using the rover.

The first thing we would need are sensors or suitable sensing circuits for the four different signals and radiations to achieve the identification of the minerals. The data from the sensors needs to be analysed which would be done digitally through an Arduino program. In order to get the data from the sensors to the Arduino, we needed circuits to amplify and convert analogue data to digital.

This data then needs to be sent to a suitable interface so that a result can be seen. Since a WiFi shield was provided, a web interface could be used to display the output of the Arduino analysis program.

A key functionality of the rover is to be able to detect minerals spread across a large moon surface, therefore it must be able to remotely travel across the surface whilst avoiding obstacles in the way. Motored drive would be needed for this and we could use the skills learnt in the lab work from the EEEBUG project for this. To be remote controlled, the WiFi shield can be used again to control it from a website.

Once all of these parts are completed, they can be assembled onto a chassis, once again following the basic design from the EEBUG.

Thus, our high level design took into consideration the main components of the rover and how they would interact with each other.

## Detailed design

In the repository, we were given the design of EEEbug (.dxf file). The file contains the measurements of the breadboard and metro board. In our design, we copied the measurements and designed our own shape.

Our chassis design included several requirements:

1. A place to fit two breadboards, one Adafruit Metro M0 board (with WiFi shield connected on top of it), one EEEbug PCB, and one battery pack.
2. A round area where the coil can be fixed on.
3. A place to fix other sensors
4. A place to install four wheels
5. Several holes for wires to go through

By combining all the requirements, we designed the following chassis and printed it through a laser cutter, shown in Figure 24.

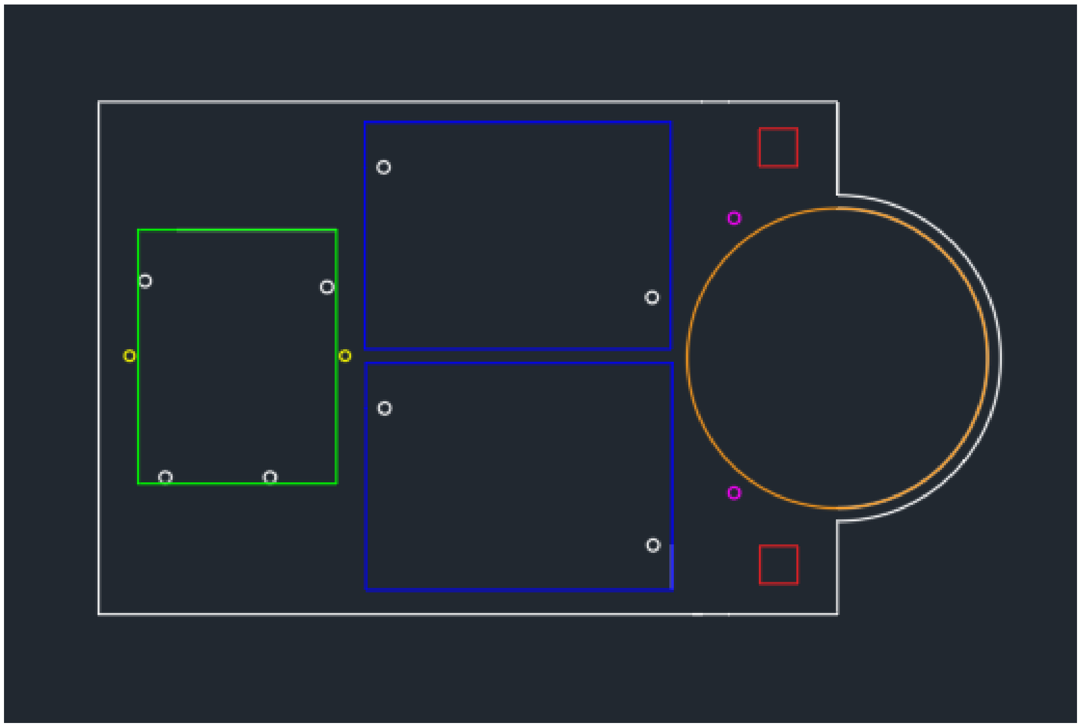


Figure 24: Chassis Design

The big orange circle on the right is for the coil, and it is at the front of the vehicle. The two blue squares are spaces for breadboards. The green square at the left is for metro boards. The two yellow circles are holes for the battery pack installed on the back of the chassis. The two red squares are planned for sensors' wires to go through. The two pink holes are planned to use wires to fix the coils on the board, although they are also used for other purposes in the end.

In our initial plan, the motors were stuck on the chassis by glue, so this design did not include holes for the motors. However, after consulting with the TAs, we found that sticking motors on the chassis is not good engineering work, and is not strong enough for launching to the moon as well. To fix the problem, eight new holes were drilled into the chassis for holding the motors. We used several M3 screws, nuts, and right-angle brackets to fix the motors.

Another problem was that we didn’t leave space for the EEEBug PCB in the designed chassis. The problem was fixed by installing the PCB on top of the breadboard using small screws and nuts through the pink holes. The coils are fixed through tapes instead.

Besides that, our initial plan was to fix acoustic and infrared sensors under the chassis facing toward the rock. Nevertheless, this arrangement received weak signals. Thus, we changed the design by fixing the sensors all in the middle of the coil, above the rock. The new arrangement receives all the signals perfectly, shown in Figure 25.

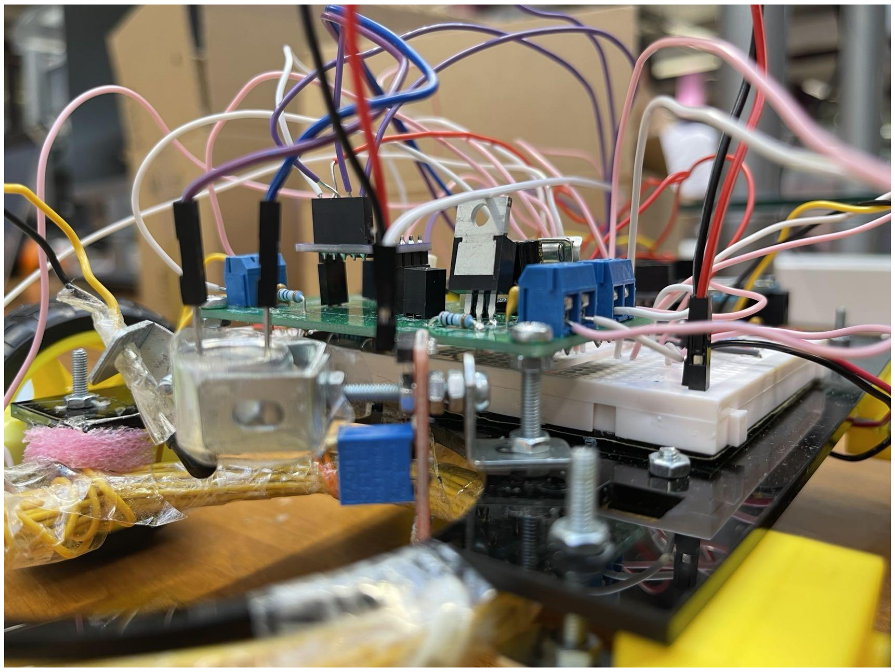


Figure 25: Sensor Arrangement

## Motorisation and Power

An Adafruit 4-channel motor shield was initially chosen to drive the motors. This shield would have been attached directly into the Adafruit Metro M0 microcontroller which would have freed up space on the rover by not having to use the EEEBug printed circuit board. However, the two were not compatible, and this meant that the PCB had to be used alongside the H-Bridge Motor Driver module provided in the kit. Figure 26 shows the required connections.

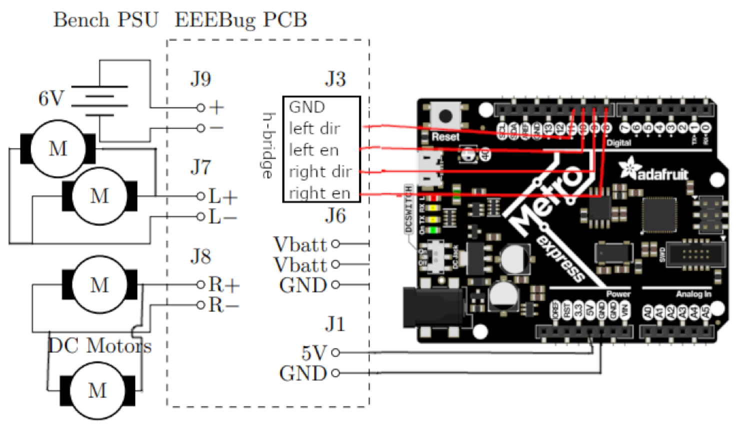


Figure 26: Motors and H-Bridge Connection Schematics

The following table shows how each command was represented with HIGH and LOW combinations of the 4 digital outputs. The details of Arduino code used to control the motors can be found in the software section.

| Command | left_en | right_en | left_dir | right_dir |
|---|---|---|---|---|
| Forward | HIGH | HIGH | HIGH | HIGH |
| Backward | HIGH | HIGH | LOW | LOW |
| Left | HIGH | HIGH | LOW | HIGH |
| Right | HIGH | HIGH | HIGH | LOW |

Table 1: Digital signal outputs to represent commands

Both the Adafruit Metro M0 and the EEEBug PCB were powered by four 1.5V batteries connected in series, which were placed under the rover. The two DC motors on each side of the rover were connected in parallel, making the rover an all-wheel drive.

## Software

### Web page

The webpage that is returned is written in HTML, with CSS styling and Javascript elements embedded within it. Before including the web page code in the arduino sketch, its formatting is slightly adapted so that it is of type string. It is also split into eight smaller web pages so that the content takes less time to load in the browser compared to having one large web page. These 8 smaller webpages which together form the full HTML code are sent via the sendContent method. Javascript is used to define functions that send HTTP requests to the server when the HTML buttons are pressed. These requests are interpreted by the server using the *on* method. This method links each request sent by the web client to a predefined function in the Arduino sketch. These functions are either to control the vehicle or to make a Scan request, and are explained in the following sections.

### Vehicle Control

As shown in Figure 24, the rover's movement is controlled by buttons on a webpage. The forwards and backwards and buttons call a function that sets the required direction on the engines, turns them on and brakes after 100 milliseconds. The left and right buttons set the direction of the left and right engines so that they are opposite and turn them on for 100 milliseconds. This causes the rover to spin on itself in the direction requested by the user. Javascript is also used to connect certain keyboard keys to the HTML buttons. This allows for a smoother control of the Rover, as the user does not have to continuously drag the mouse across the buttons, but instead uses the keyboard keys. Table 2 clarifies how the user can use keyboards to operate the Rover.

| Keyboard Key | Equivalent Button |
|---|---|
| W | Forwards |
| A | Left |
| S | Backwards |
| D | Right |
| Enter | Scan |

Table 2: Keyboard commands to operate rover

## Processing Data from Sensors

There were 4 sensors implemented on the rover, each of which had a void subfunction to be executed when detecting the corresponding signal.

The first subfunction detected the magnetic field. It read value directly from analogue input A4, and would increase if the rover was approaching an upward magnetic field, and vice versa.

Both modulation radio signals and infrared signals required counting for frequencies of their input: digital square signals, and they were realised in a similar way. The function  for example, would allow the program to measure the time of a continuous HIGH input from the pin called . And the last coefficient, 20000, was aiming to limit the waiting time to 20 milliseconds, so that the program would not waste too much time waiting for the signal, as in some cases there were none. After measuring the HIGH input time, the program would also measure for the LOW time, and add the 2 together so as to have the total period. Then it would divide the period by 1 second to get the frequency in Hz. The detected frequency would be nearly 0 when far away from a signal source, so the program considered the radio signal as 151 Hz when the calculated frequency was in the range of 100-200 Hz, and 240 Hz when the input was 200-300 Hz. The infrared sensor would give a result of 353 Hz with a range of 200-500 Hz, and 571

Hz with range 500-700 Hz. The sensors themselves worked accurately enough, but the program was made in a way to avoid as much noise effect as possible.

The last sub function worked for the acoustic signal. Because the ultrasonic wave was only 40 kHz or nothing, this sensor's input frequency did not have to be counted. So, it was connected to an analogue input port A0 and would only provide an increase in flag number if the input voltage went above 0.4 V.

In the Arduino program, all 4 void sub functions were called one after another for 20 times, in order to get rid of any effect of anomalies, which might be caused by the environmental noise. As mentioned earlier, in order to change the coil's tuning frequency, another set of capacitors would be connected to the circuit when a switch was turned on. The switch had a resistance which allowed a large enough base current to saturate the BJT when a high voltage output, 3.3V was applied to the base. So, in the program, the loop was processing the sub function of detecting for the radio frequency 10 times with the switch opened and then 10 times with the switch closed.

Every time the sensors detect a signal, the corresponding sensor flag would increase by 1. At the end of 20 times of detection, the program would go through the conditions for all of the rocks one by one, with every signal considered as received if it had a flag number of more than 10, and for radio the threshold was 7 as each carrier frequency was detected 10 times only.

## Testing

The process followed was as below, after each stage was successful, we moved to the next step of testing:

1. Testing each key component individually e.g. IR sensor circuit, radio waves circuit, and viewing the results on an oscilloscope.
2. Testing each sensor circuit with its respective Arduino code to see if Arduino was able to detect and process the signals correctly or if they needed to be amplified.
3. Test the entire Arduino code which analyses all the signals and displays the rock name on the Serial Monitor with the rock.

4. Test the integrated Arduino code with the website to ensure the correct rock name is displayed on the website whilst operating the rover using the website.

During the final stage, the website functionality was improved to allow the rover to move using the keyboard. A change was also implemented to scan only when a command is sent using the Enter key as opposed to continuously scanning while moving because that takes a lot of time and processing power.

We also realised that the mineral with the 61kHz radio carrier frequency was not being detected and when the rock was moved half a centimetre closer to the coil, it was being detected. This meant that the signal attenuation was taking place; however, the point of interest was why a similar phenomenon did not take place for the 89kHz frequency. There could be two possible causes: the rock could be outputting a weaker field for the lower frequency, or there was something in our circuit that produced a weaker signal for the 61kHz frequency, or both.

It was found that the bipolar junction transistor which was being used as a switch was adding a resistance to the LC circuit, causing it to become an RLC circuit, thereby reducing the resonance peak (making it wider) and thus attenuating the signal reaching the Arduino. Possible options to rectify this included using a MOSFET or diode circuit as a switch as shown below in Figure 27 instead of a BJT, since it doesn't need any base or collector resistors, or implementing a mechanical design to move the coil closer to the rock. However, moving the coil closer to the rock had the disadvantage of the coil potentially being stuck on top of the rock by encapsulating it, and thereby making it harder for the rover to drive away after the mineral had been identified.

Photos and videos were taken of the rover demonstrating successful tests; these have been uploaded to a Google Drive and can be accessed here: https://drive.google.com/drive/folders/1cEfdNgB4zesglfpg_vjwSQ-biuoMHu6C?usp=sharing

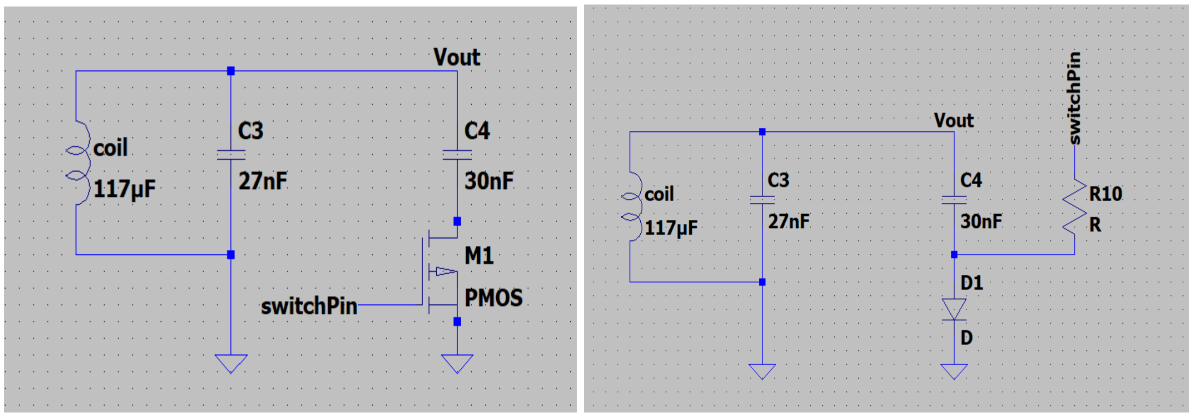


Figure 27: Circuit of a switch using MOSFET(left) and using a diode(right)

The latter, mechanical option was chosen since it had a lower cost and was much simpler to implement and didn't involve having to remove all the components from the rover to edit the circuit. One end of the coil was lowered to enable the signal to reach the coil at the distance required and the other end was kept at the height it was previously so the rover could easily drive away from the rock without having the coil stuck on it.

## Post-testing Improvements:

1. The coil was changed to a bigger diameter so that it is more sensitive to the switch in the radio sensor during the initial individual radio circuit testing stage.
2. A decoupling capacitor was added to the IR circuit to avoid interference with Arduino.
3. Output resistors were added to limit the signal voltage going into the Arduino board (since it has a diode at the inputs and operates at 3.3V so it won't allow the current to go through if it is too high).

# Project Management

One of the first tasks the team undertook was creating a detailed plan for the progression of the project. At the earliest stages this required communication within the team to establish each member's strengths based on any previous experience they had working in similar projects. This information would then inform the allocation of tasks within the group, from research and brainstorming to final design and manufacture.

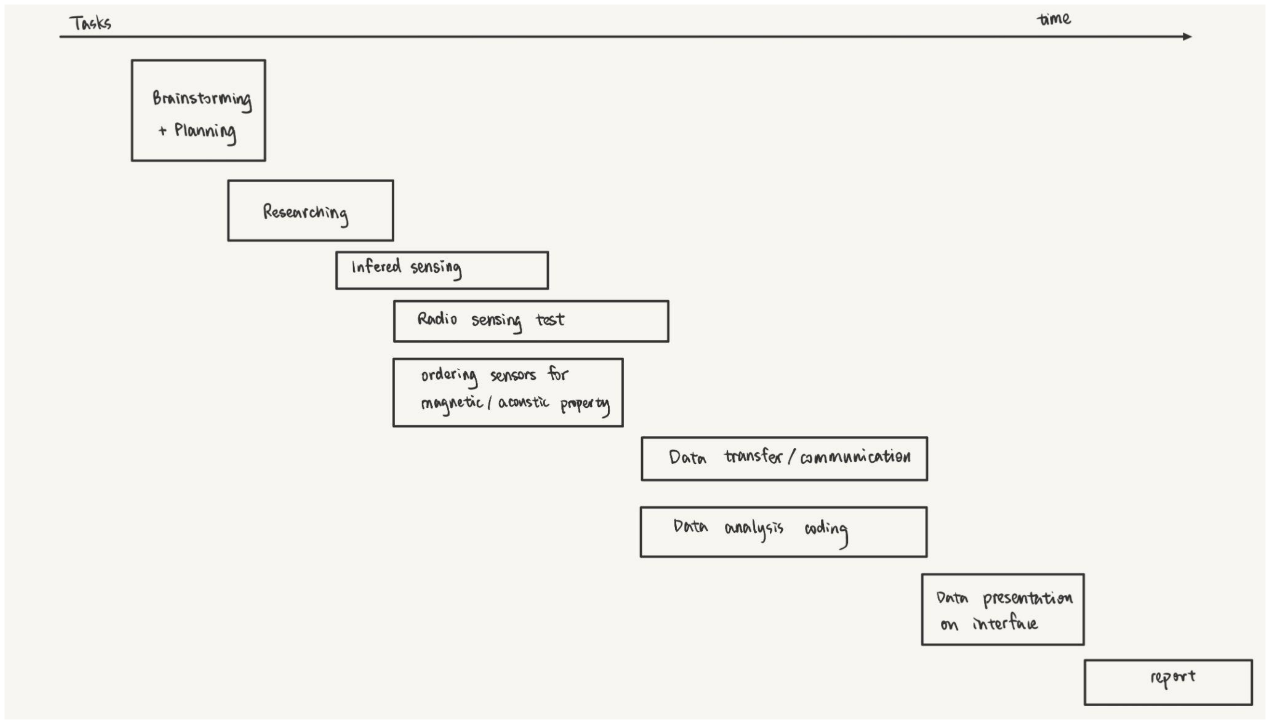


Figure 28: Early Gantt Chart showing conceptual project progression.

Initially, we separated the research stage into 4 areas; these being (1) the sensors, (2) the wireless connectivity, (3) the software, and (4) the mechanical aspect of the project. Estimating the amount of research needed for each, we assigned 2 group members to investigate the sensors, and 1 group member each to connectivity, software and mechanics. The final group member was tasked to begin drafting the report according to the research of the other members. This is a role they largely maintained throughout the duration of the project. This approach enabled the team to utilise the full depth of knowledge and experience of each member in their respective specialty and allowed for easier collaboration between those in the team with similar specialties, as well as more evenly distributing the workload.

Moving past the initial research and brainstorming stage, a more thorough plan was devised to best carry the knowledge and ideas we had researched forward into the design stage of the project. This plan defined the schedule we intended to follow from the earliest prototypes to the final manufacture of the rover, and is shown in the Gantt Chart in Figure 29.

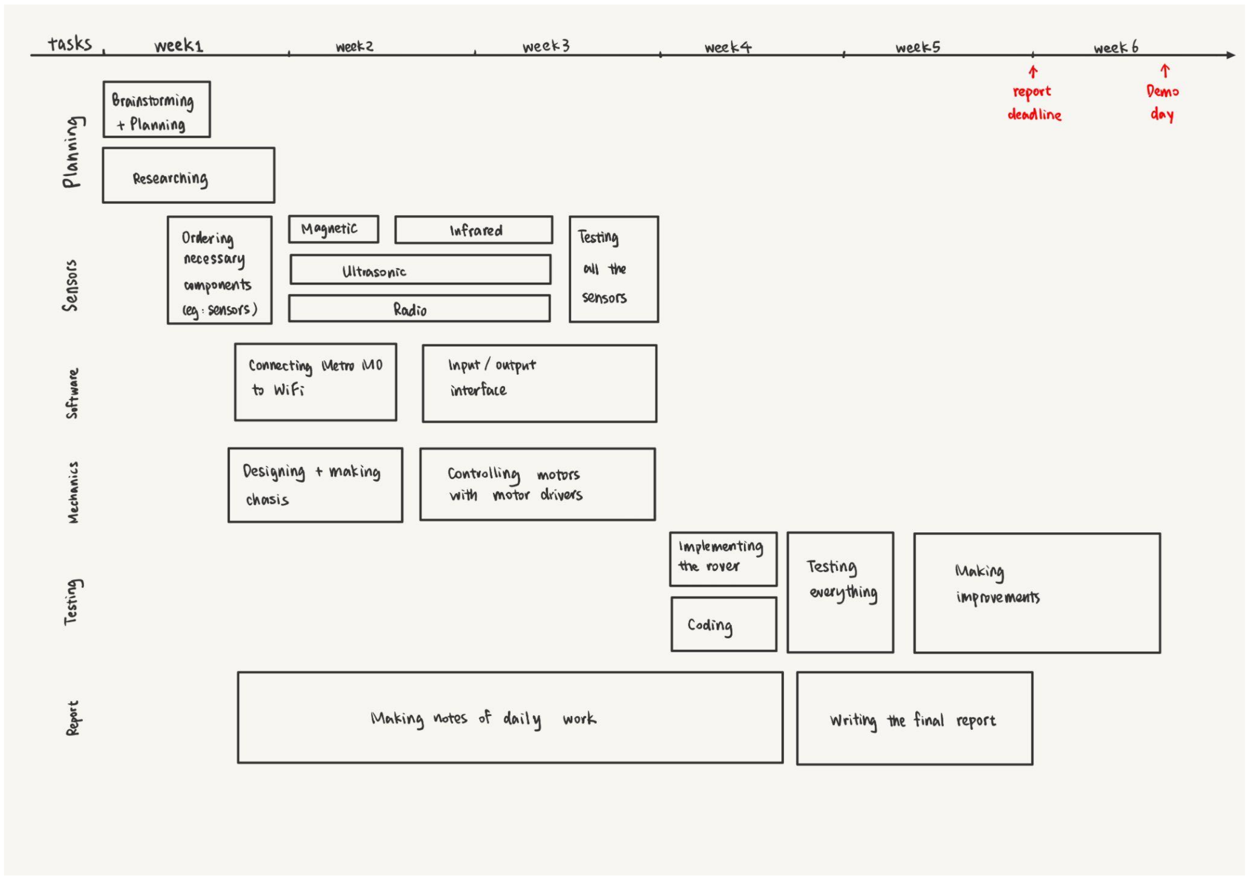


Figure 29: Final Gantt Chart

As with any effective plan, elements of redundancy and flexibility were built in, to allow for group members to shift their priorities as different aspects of the project developed. This enabled a more efficient workflow throughout, as each member could address their focus to where it was most needed, and reduced the stress placed on any one individual.

In retrospective evaluation, the project progression was largely accurate to the Gantt Chart. While certain aspects took a lot less (e.g. magnetic sensing) or more time (e.g. wireless connectivity) than had been anticipated, for the most part the plan remained unchanged. The benefit of having multiple weeks towards the end reserved exclusively for testing and further development, as laid out in the Gantt Chart, enabled optimisation and fine tuning of the rover, thus allowing for the greatest possible degree of reliability and achieving a high level of performance from our design concept.

## Product Design Specifications

Out of the 32 elements in a regular PDS(Product Design Specification), we discuss here the ones that seemed most relevant to the EEERover. These are weight, target product cost, performance and time-scale.

In line with its purpose of being sent to the moon, the rover's weight had to be minimised. Sensor circuits were implemented together neatly on two pieces of breadboards in order to avoid using more boards, which would make the rover become heavier. Using flat wires rather than regular jumper wires meant that the space on the breadboards was much more optimised. Additionally, the chassis was designed in the final stages of the project to ensure that it was as small as possible while providing space for all the necessary components.

To meet the product target cost of £60, we made use of the already available components as much as possible. For example, preloved op-amp chips and one LT1366 op-amp chip from autumn term were used to satisfy different amplification requirements; one EEEBug PCB from spring term was used with the new motor drivers; a coil which was tuned by capacitors was used to detect radio signals instead of purchasing a specific built antenna. The only components that were bought were infrared sensors, an acoustic receiver, a magnetic hall effect sensor, wheels and flat wires. The total cost of these was £27.

As for the performance of our rover, it was designed to be as reliable as possible. In order to get the most accurate results, every sensor was tested separately and evaluated on its sensitivity. The more sensitive and accurate sensors were considered to be more trust-worthy during the programming, and would be relied on more heavily in the software section. Another aspect of the performance was the rover's speed. This was achieved not only in a mechanical way (4-wheel drive), but also with programming methods. During the testing of the rover, it was found that the time of detecting rocks could be largely reduced. The waiting times when no signal was received could be decreased by 98% for the IR sensor, and 96% for the radio sensor.

Finally, we were given a very strict deadline regarding the submission date. In order to respect the set deadlines, the whole group was heavily involved in the project management stage from the beginning. Developing a Gantt chart, setting internal deadlines and splitting tasks as equally as possible to the different group members were only some of the ways that we ensured we met the time-scale requirements. This process is described in more detail in the section above.

## Future Work

With the time we had available we were able to deliver a rover that entirely satisfied the specifications of the minimum viable product. Nevertheless, throughout the design and construction of the vehicle, there were several additional implementations that we would have liked to include but that would have used up too much time. Hence, we include this section to describe how we would have further improved the rover if we had more time.

Firstly, we would have taken the circuits on the breadboard and soldered them onto a circuit board. This would mean the connections are more reliable, as we do not run the risk of a wire accidentally losing contact and thus inhibiting the full functionality of the rover. This would also have reduced the overall mass of the rover, as the boards are significantly lighter than the breadboards. We would also have spent more time tidying the other wires in the rover.

Another improvement would have been to include a servo motor to adjust the position of the coil of wire that acts as an inductor. This is because when our rover approaches a rock, the coil slightly touches the rock before reaching its final scanning position. This could be avoided by raising the coil and putting it back down once the rock is in the right position to be analysed. However, it is also true that this would increase the mass of the rover.

A final improvement that we would have liked to work on is increasing the speed of the motors. The only way we could have done this without changing the actual motors provided would have been by increasing the current flowing through the motors.

Thus we would need a different motor driver configuration, potentially using two drivers and PCBs rather than one.

## Conclusion

To conclude, we used four different parameters to distinguish between all six rock types.  To identify rocks emitting infra-red (Thiotimoline and Netherite) at different frequencies, an infra-red sensor was used. Infra-red sensor placed in a potential divider circuit outputting high when signal detected. The signal is then amplified using an inverting amplifier. To reduce noise a high pass filter was implemented and finally a Schmitt trigger to convert the electrical signal from analogue to digital.

To detect the magnetic field, a hall effect sensor was used to determine the direction of current travelling in the circuit demonstrating whether the field direction was up or down. This allowed us to differentiate between Adamantine and Xirang.

To identify Gaborium and Netherite, we used an ultrasonic receiver to detect the acoustic signal. We then used an inverting amplifier circuit to distinguish the acoustic signal emitted from the rock from external noise. Finally, we used a half-wave rectifier circuit to keep the output wave near constant for Arduino to interpret the difference between noise and signal.

As there was no pre-built sensor for the detection of radiowaves, we used a coil of wire to detect varying electromagnetic waves which induced a voltage across it. We used a resonant circuit to see single out frequencies that matched with the rock. As there are two different radio wave frequencies to detect, we used a BJT to act as a switch to identify both. Signal was then amplified using an inverting amplifier. To demodulate the signals we used a half wave rectifier to implement an envelope detection and carry out the modulated frequencies. A Schmitt trigger is then used to convert the electrical signal from analogue to digital.

A Wi-Fi-shield was used to allow the microcontroller to host a webpage programmed in HTML, from which a user controlled the motors and the type of rock detected was displayed. The webpage made it so that the keys W, A, S, D could be used to govern the motion of the rover.

Softwares are programmed according to how sensors work and are combined with motor drivers in the end. The chassis was built using the software AutoCAD, this was the last to be implemented as the shape and size needed to account for the spacing of all sensors, breadboards and motors. We decided to make the rover an All-wheel drive and powered it using a 6V battery pack. A photograph of the final product is shown in Figure 30 on the next page.

Overall, our project successfully fulfils the fundamental objective outlined at the start of the report: “to build a lunar rover which can identify minerals by being remotely controlled across a surface resembling the moon.” Furthermore, all of the criteria in the Product Design Specification were met. The cost was less than half of the provided budget, the rover is fairly lightweight as well as stable, the performance fulfils the objective, i.e. all of the minerals are correctly identified, and the project was completed as per the deadlines, meeting the time-scale.

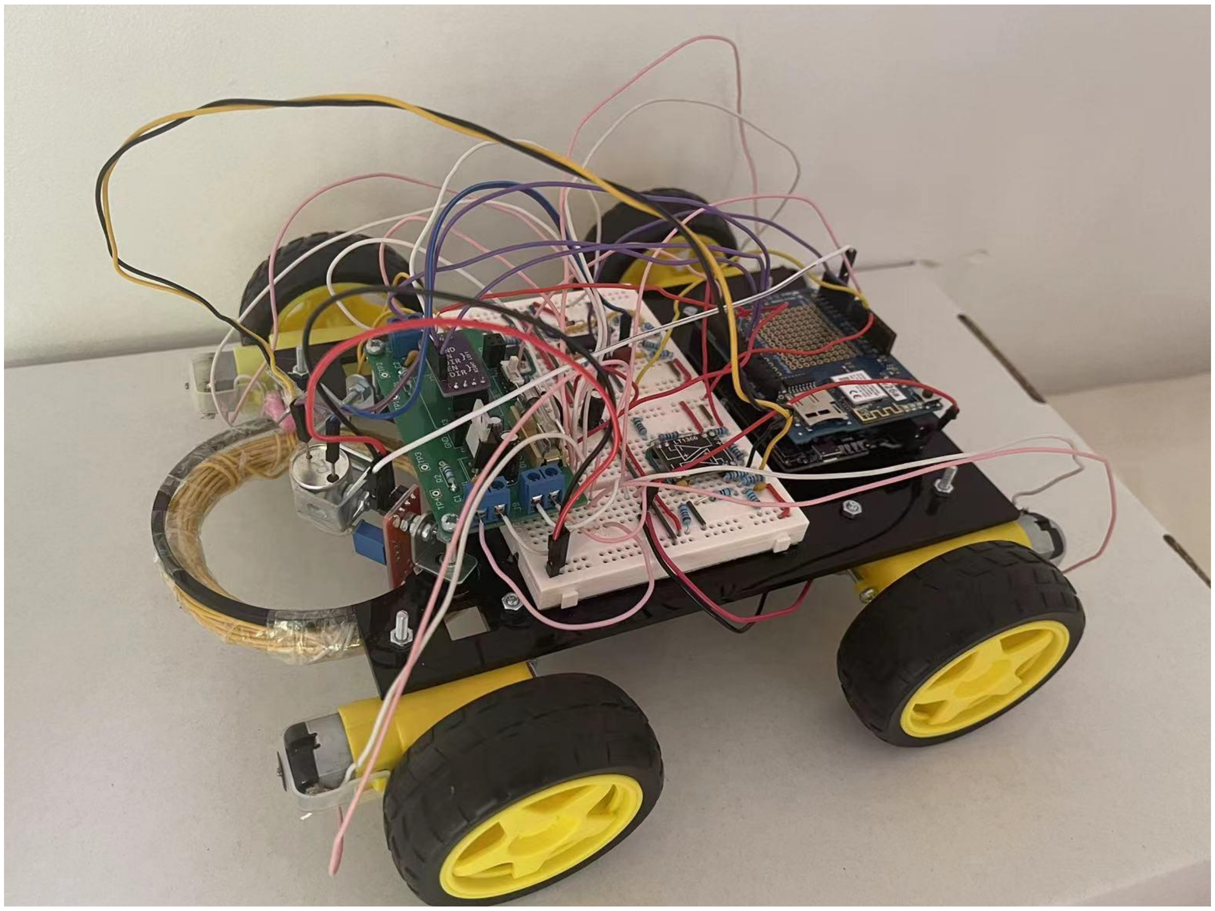


Figure 30: Photograph of EEErover in its final version