Imperial College London

Imperial College London
Department of Electrical and Electronic Engineering

Group ESB: EEE Rover Project Report

Yuanze Xia
CID: 02020493

Timothy Chung
CID: 02015795

Han Jang
CID: 01856804

Haokai Qin
CID: 02034263

Katari Hassan
CID: 02052242

Anthony Bolton
CID: 01913117

June 2022

With our deepest gratitude to Professor Steven Wright and Dr. Edward Stott

Contents

1 Project Management **3**
1.1 Conception and Initiation . . . 3
1.2 Assumptions . . . 3

2 System Specification **3**
2.0.1 Circuit Design and Implementation . . . 4
2.0.2 Sensors Programming . . . 4
2.0.3 Remote Control . . . 4
2.1 Performance and Control . . . 5
2.1.1 Team Communication . . . 5
2.1.2 Gantt Chart . . . 6

3 Rover system Design **6**
3.1 Rover Structure Design . . . 6

4 Circuit Design and Implementation **9**
4.1 Radio . . . 9
4.1.1 Radio sensor . . . 9
4.1.2 Amplification . . . 10
4.1.3 Demodulation . . . 14
4.1.4 Stabilization and Filtering . . . 18
4.1.5 Complete Circuit . . . 19
4.2 Infrared . . . 19
4.2.1 Infrared sensor . . . 20
4.2.2 Amplification . . . 20
4.2.3 Stabilization and Filtering . . . 21
4.3 Magnetic Sensing . . . 23
4.4 Acoustic Sensing . . . 25
4.5 Sensor Programming . . . 26
4.5.1 Frequency Sampling (Radio and Infrared) . . . 26
4.5.2 Hall Effect Sensing (Magnetic) . . . 28
4.6 Remote System Design and Programming . . . 30
4.6.1 User Interface . . . 30
4.7 Testing . . . 33

5 Overall Performance Evaluation **34**

6 Conclusion **34**

7 Appendix **35**
7.1 Product Design Specification 35
7.1.1 Performance 35
7.1.2 Environment 35
7.1.3 Testing Product Cost 35
7.1.4 Size and Weight 35
7.1.5 Material 35
7.1.6 Aesthetics, Appearance and Finish 35
7.1.7 Ergonomics 36
7.1.8 Quality and Reliability 36
7.1.9 Process 36
7.1.10 Disposal 36

1 Project Management

The Project Managements Institute's 5 Phases of Project Management is utilised by this project team as a reference to ensure that all features of project planning and management is captured.

1.1 Conception and Initiation

Project Definition: Design a remote-controlled rover that can explore a lunar landscape and identify artificial lunar rocks that emit unusual electromagnetic and acoustic signals.

Mineral	*Property 1*	*Property 2*
Gaborium	61kHz radio modulated at 151Hz	Acoustic signal at 40kHz
Lathwaite	61kHz radio modulated at 239Hz	None
Adamantine	89kHz radio modulated at 151Hz	Magnetic field up
Xirang	89kHz radio modulated at 239kHz	Magnetic field down
Thiotimoline	Infrared pules at 353Hz	None
Netherite	Infrared pules at 571Hz	Acoustic signal at 40kHz

Table 1: Rock Properties

Table 1 shows the 6 types of lunar rock that the rover is required to identify and the properties they consist of. As seen, each rock emits different signals including Radio, Acoustic, Magnetic and Infrared. By referring to distinct features of each rock in *Property 1* and *Property 2*, users will be able to detect the signals and identify the rock. The project will fulfill the project definition by implementing analogue and digital design to the rover.

1.2 Assumptions

- Artificial moon surface provides enough traction to 4 rover wheels
- Rover weight can be supported by the motors
- Battery voltage does not degrade (Ideally)

2 System Specification

The rover will have the following main functionalities:

- Remotely explore a lunar landscape with help from a smartphone remote control

- Receive, process and analyse signals emitted by the rocks
- Identify and differentiate all 6 types of rock in Table 1

2.0.1 Circuit Design and Implementation

This section explores the hardware schematic and circuit design of the rover. Its main functions include:

- Signal processing
- Amplification of radio and infrared signals
- Demodulation of radio signal
- Signal stabilization

2.0.2 Sensors Programming

- Be able to detect a frequency of a given radio, infrared or acoustic signal
- Transmit sensor readouts to the remote control

2.0.3 Remote Control

- Web app built with HTML, CSS and JavaScript, runs locally on a smartphone
- Connects to the Arduino on the router network to send movement commands via HTTP requests
- Arduino receives commands using the Wifi WebServer Library

2.1 Performance and Control

2.1.1 Team Communication

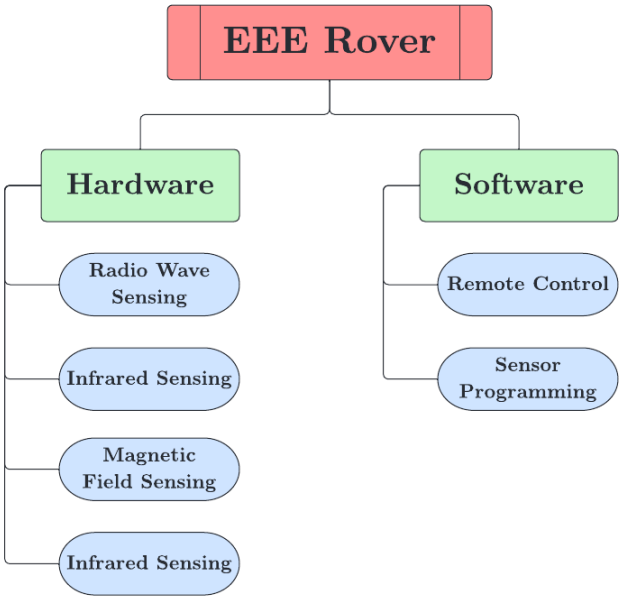


Figure 1: Project Communication Hierarchy

The flowchart in *Figure 1* was implemented by the project team to visualise how each module is linked together in order to work effectively. The hierarchy is spilt into two parts which classify the responsible modules for hardware and software. Meetings are held at least twice a week to discuss problems and ensure progress is on track.

2.1.2 Gantt Chart

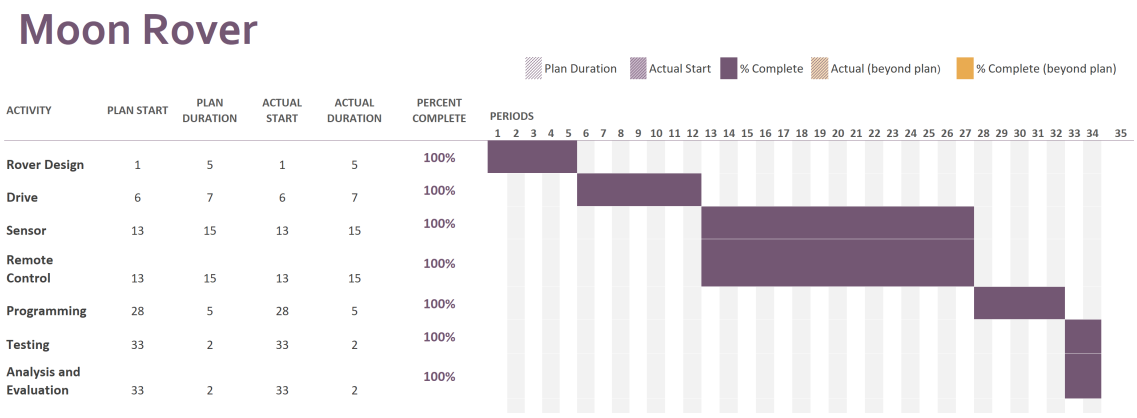


Figure 2: Gantt Chart

A Gantt chart is updated throughout the timeline of the project to keep track of progress. The team followed the timeline as closely as possible to ensure all work is tested and completed before the deadline.

3 Rover system Design

3.1 Rover Structure Design

- The Moon Rover weighs a total of 800 grams and occupies a 30cm-by-16cm footprint. The chassis consists of two EEEBug 3.0 acrylic bodies combined together, with motors and circuits affixed via screws and nuts. The acrylic material is stiff enough to support the weight of the rover while being able to account for slight flexing when the rover is moving on a rough surface. The structure is strengthened by drilling additional holes for screws so that all the parts are securely anchored during testing and operation.
- Two EEEBug 3.0 chassis were combined together as the existing chassis would potentially not be large and stable enough to provide flexibility in mounting additional breadboards/circuits. Due to a college-wide printing malfunction(which prevented us from printing templates to prototype a new acrylic chassis), combining two EEEBug chassis together was seen as a more time-effective solution, reducing waste from unused parts from the lab skills module. Doing this increased the number available motors/wheels and available breadboard space to mount the circuits for the sensors.

- Subsequent to the testing of the dual EEEBug chassis, the team noticed a significant slipping of the rover to the right when in motion. This was due to uneven wheels and misaligned motors. The motor mounts were re-tightened, re-aligned, and the wheels were swapped out taller, wider tyres. This improved the traction of the rover and increased height clearance, allowing the rover to travel stably over small terrain imperfections.
- With the increased space of two EEEBug chassis, the rover has more space to incorporate an additional battery pack with capacity for 4 AA Batteries. This battery pack is used to power the two additional motors and the amplification and peak detection circuits for the sensors.
- In order to fit an additional breadboard, an 3D-printed L-bracket was utilised to support an additional breadboard over the Arduino board. The bracket can be moved aside to allow access to the wiring of the Arduino below for diagnostics or adjustment. This breadboard contains the sensor circuits being placed in a central location away from the ground prevents it from impacts while the rover is in operation.

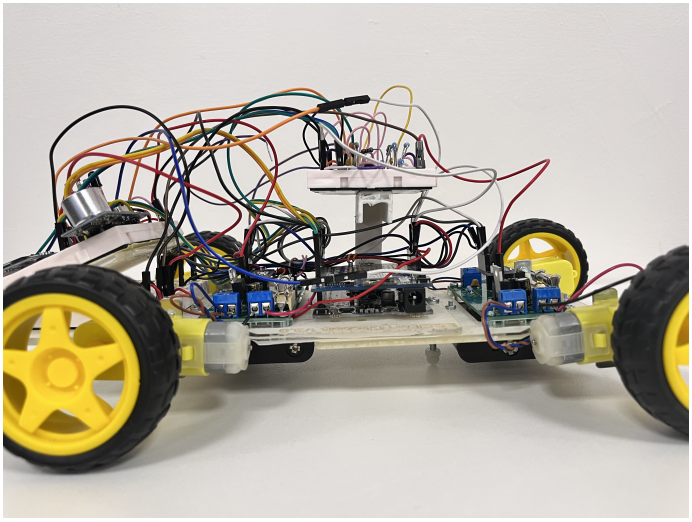


Figure 3: Prototype Rover Picture 1

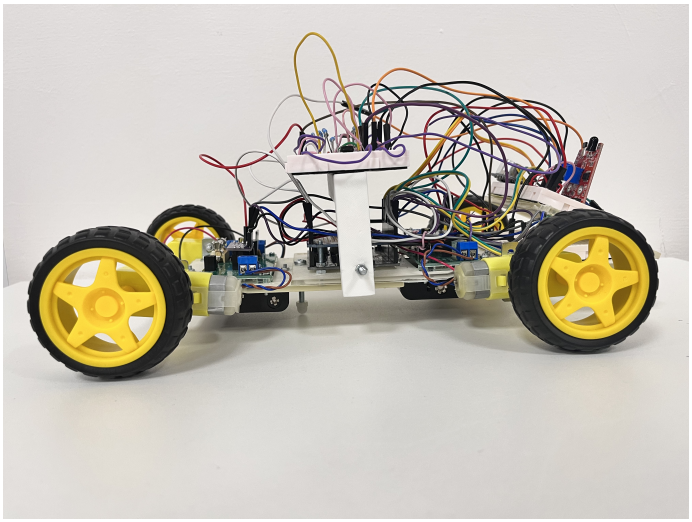


Figure 4: Prototype Rover Picture 2

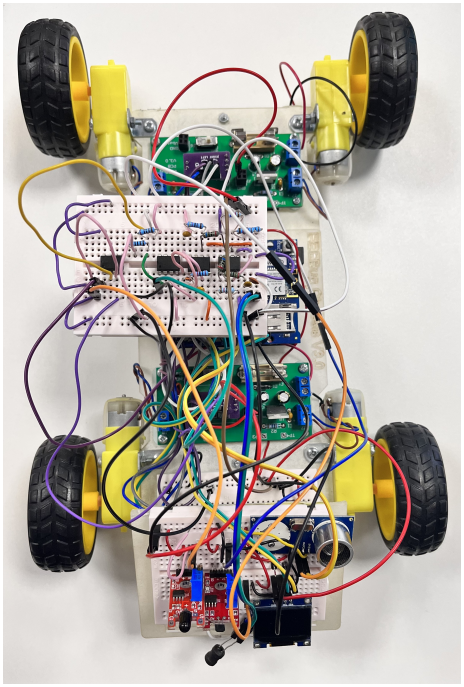


Figure 5: Prototype Rover Picture 3

4 Circuit Design and Implementation

4.1 Radio

According to *Property 1* in *Table 1*, rocks Gaborium, Lathwaite, Adamantine and Xirang emit electromagnetic signals in a form of radio waves respectively at different arrangements of carrier and modulation frequencies. Therefore, the project team has designed a circuit to amplify and demodulate the radio signals received in order to identify each rock precisely.

Figure 6: Signal processing method

Figure 6 demonstrates the stages of signal processing as the rover receives a radio signal. In total, there are 4 combinations of 2 carrier frequencies and 2 demodulation frequencies. Firstly, to receive the signal, the team is using a 47mH inductor as a radio sensor. After receiving, the signal will need to undergo amplification before it can be used to indicate detection, because the signal generated by the inductor will be inconveniently small. After amplification, the magnitude of the signal will be large enough to be processed by the Op-amp based peak detector, which is implemented to identify the demodulation frequency. At last, a comparator circuit is used to stabilize and filter out the distortion and noise within the processed signal.

4.1.1 Radio sensor

The signal received by the rover will be amplitude modulated radio waves consisting of different frequencies shown in *Property 1*. Originally, the team's first attempt in creating a radio sensor was a tuned coil antenna with a reasonably large diameter(0.15m), which will essentially act as a air-cored inductor. The diameter of the antenna is a key factor to its inductance due to the following relationship:

$$L_{coil} = \frac{\mu_r \mu_0 N^2 \pi r}{l}$$

L_{coil} = Inductance of coil(H) $\quad$ μ_r = relative permeability of the core
μ_0 = permeability of free space = $4\pi \times 10^{-7}$ (H/m)
N = number of turns $\quad$ r = coil radius(m) $\quad$ l = coil length(m)

The *LCR METER HM8018* is used to measure the inductance of the coil antenna,

which results in a reading of $60.5\mu H$. During the first two weeks of the project, the team used the coil antenna as the primary radio sensor during circuit prototyping stage. However, when subsequently planning to consolidate all the sensor components onto the rover, we found the coil antenna to be inconvenient and unwieldy in due to its size and space occupied. Therefore, the team settled on an alternate sensor using just an $47mH$ inductor as the sensor, as it occupies significantly less space and allows for more flexibility in placement.

To successfully detect both carrier frequencies, the radio sensor is required to be set at a resonance frequency between two carrier frequencies. For calculation, the resonance frequency formula is used:

$$f_r = \frac{1}{2\pi\sqrt{LC}} = \frac{1}{2\pi\sqrt{47\times10^{-3}\times91\times10^{-12}}} = 77 \text{ kHz}$$

4.1.2 Amplification

To amplify the radio signal received, the team has designed a three-stage negative feedback amplification circuit with a positive voltage supply of 5V, biased at 2.5V. The total theoretical gain of the circuit is 300. To assist simulation and design, the team uses the analogue electronic circuit simulator *LTspice*. All circuits are tested before being implemented in practice, which increases the efficiency of deign.

For circuit simplification purposes on *LTspice*, we have created label nets for common power sources and potential dividers that appear repeatedly within the circuit, which applies for all circuit figures in this report. Shown in the figure below:

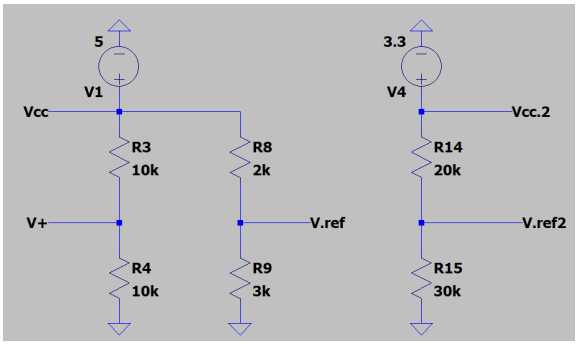


Figure 7: Common Circuit Components

As seen, *Figure 7* shows 2 voltage supplies: positive 3.3V $V_{CC.2}$ and positive 5V V_{CC}.

Furthermore, there are 3 potential dividers with different resistor values: $V+$, $V.ref$ and $V.ref2$.

The amplification circuit amplifies the signal 3 times by using 3 negative feedback operational amplifier circuits being connected consecutively.

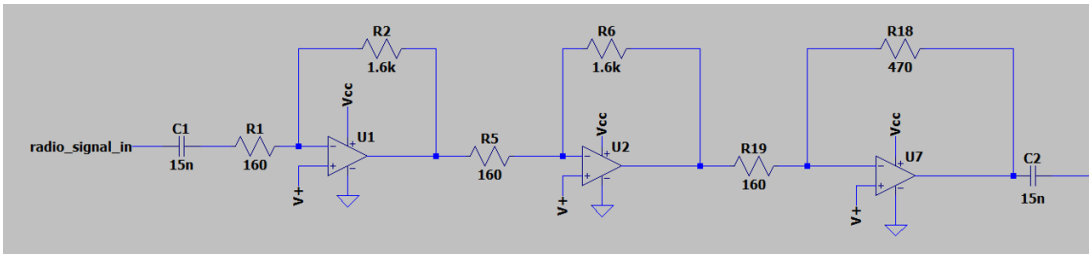


Figure 8: Radio Signal Amplification Circuit

In *Figure 8*, the circuit receives a radio signal from the rock at **radio_signal_in**. Originally, V_{CC} supplies a 5V voltage, which means the signal ranges from 0V - 5V. Therefore, the sensor will only be able to detect half-wave signals. In order to detect full-wave signals, signal biasing is needed to be at a quiescent point of 2.5V. A potential divider $V+$ is added to the negative input of the operational amplifier, where $V+$ outputs the node in between two 10k resistors, hence halves V_{CC} and biases the signal at 2.5V.

The circuit needs to block all DC signals being received to cancel the distortion it causes to the original AC radio signal. To do that, two capacitors of value 15nF **C1** and **C2** are added, before and after the amplification of the radio signal. In doing so, the circuit only processes AC signals because for DC signals travelling through a capacitor, once the capacitor charges up to the magnitude of the signal, the impedance of the capacitor becomes infinite.

Furthermore, **C1** and **R1** forms a high-pass filter with cut-off frequency 66.31kHz. To calculate this:

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \times 160 \times 15 \times 10^{-9}} = 66.31\text{kHz}$$

The reason for this value of cut-off frequency is because 66.31kHz is in between the two carrier frequencies 89kHz and 61kHZ.

To calculate the gain A, we use the gain formula for a negative feedback circuit:

For U1,

$$A_1 = -(\frac{R2}{R1}) = \frac{1600}{160} = -10$$

For U2,

$$A_2 = -(\frac{R6}{R5}) = \frac{1600}{160} = -10$$

For U7,

$$A_3 = -(\frac{R6}{R5}) = \frac{R18}{R19} = -2.94$$

Therefore, the combined ideal gain results in: $A = |10^2 \times 2.94| = 294$

The reason of the circuit requires three amplification stages is due to the limitations of the operational amplifier model. The *TL072* model is a Dual, 30-V, 3-MHz, high slew rate (13-V/µs), Vin to V+, JFET-input operational amplifier. Referring to the data sheet provided by *TEXAS INSTRUMENTS* for the *TL07xx Low-Noise FET-Input* [1] series, the maximum gain of the model within our signal frequency detection range is 10. Shown in the figure below:

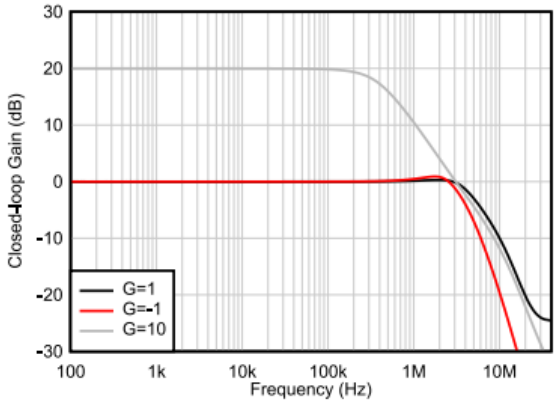


Figure 9: *TL072* Closed-Loop Gain vs Frequency

For that reason, the team decided to amplify the signal three times to reach a reasonably large gain of approximately 300. In doing so, the signal gain is within the Gain-Bandwidth Product.

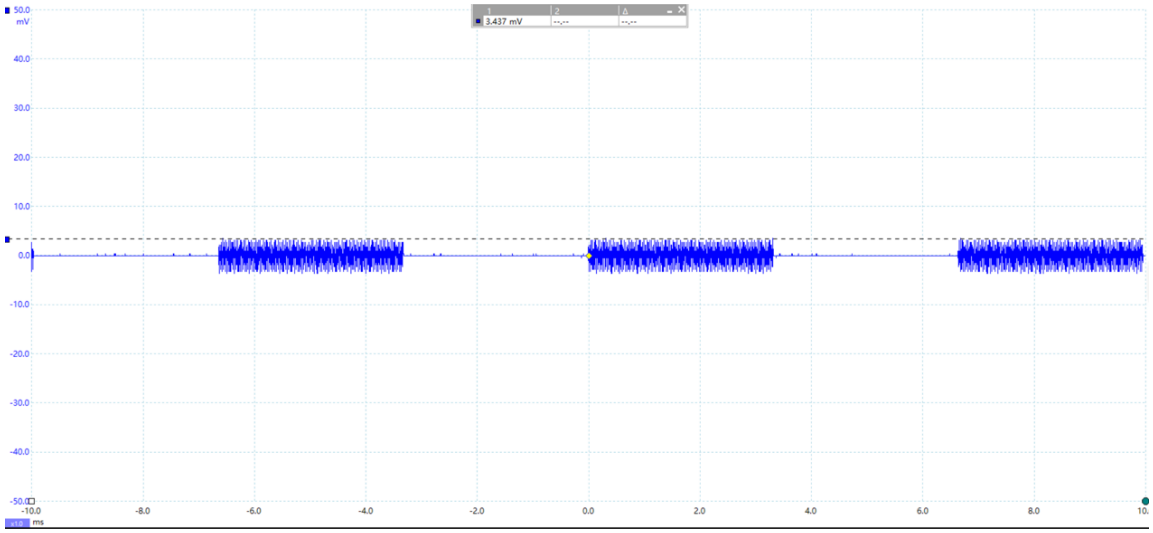


Figure 10: Initial Radio Signal
Initial radio signal before amplification

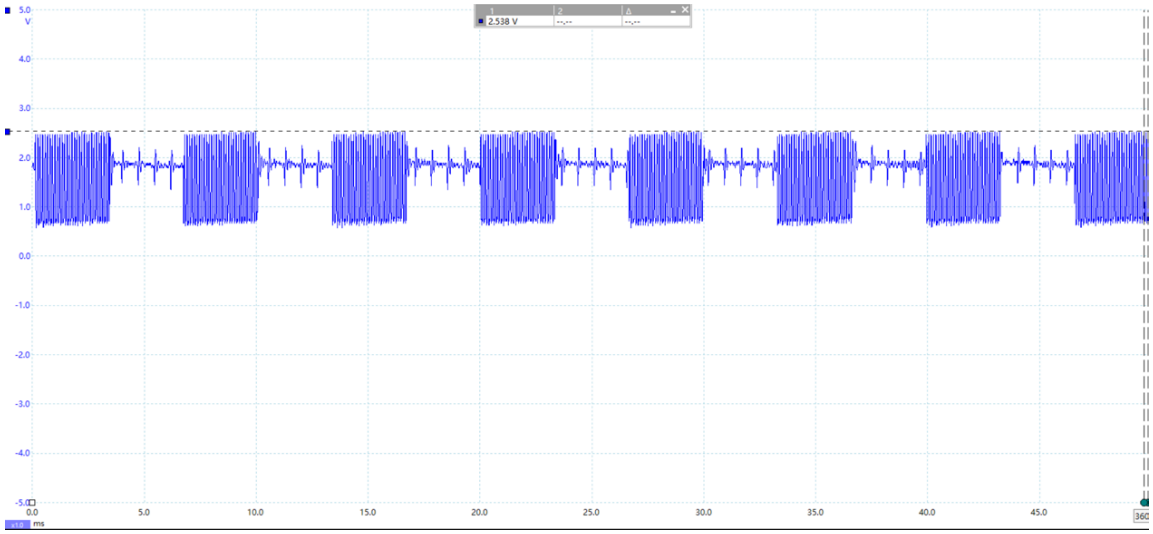


Figure 11: Radio Signal After Amplification
Radio signal after amplification

4.1.3 Demodulation

Signal demodulation is the process of extracting the original information-bearing signal, also known as the demodulation signal, from the carrier wave. Referring to *Table 1*, for each unique signal the four rocks emit, contains a carrier frequency modulated at a demodulation frequency.

To demodulate the carrier signal, the team has designed a Op-amp based peak detector circuit. By definition, a peak detector captures the peak value of an input signal. Furthermore, it is also a common circuit used for AM signal demodulation with the peak value signal being the demodulation signal. However, this decision was established after experimenting with the basic peak detector, which is comprised of a diode, a resistor and a capacitor, shown in the figure below:

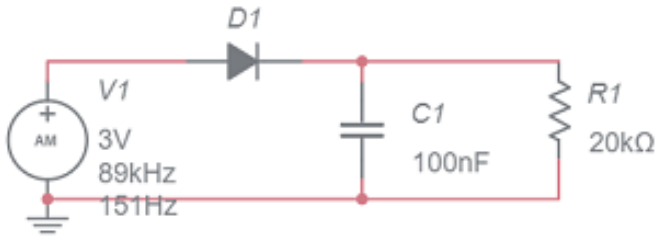


Figure 12: Basic Peak Detector Circuit

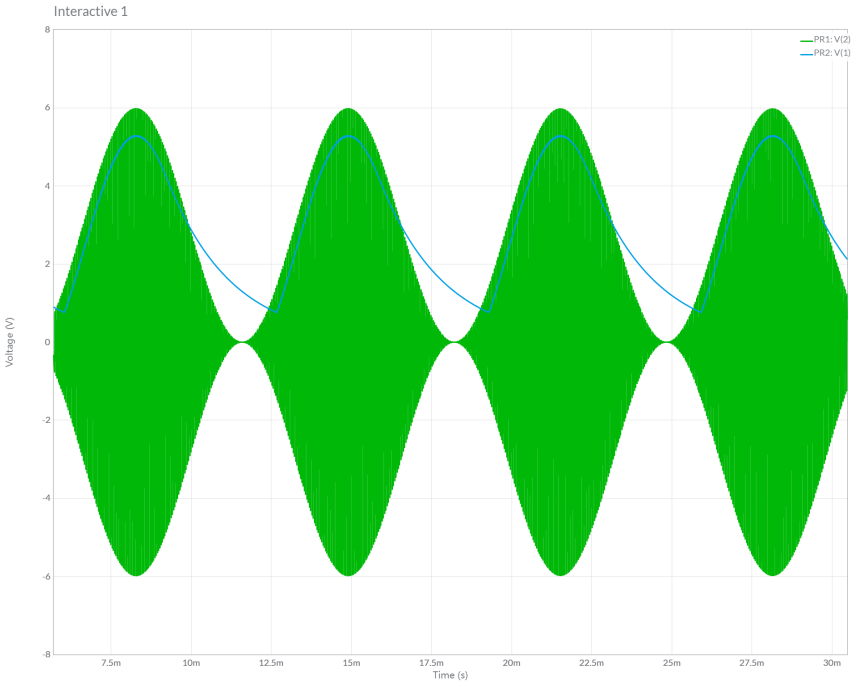


Figure 13: Basic Peak Detector Output

Initially when the input signal is at 0V, no current flows through the diode, hence the diode is turned off. As the signal amplitude begins to rise, the diode will be in forward bias but will not receive any appreciable current flow until the diode reaches its turn-on voltage(0.7V). Right when it reaches 0.7V, the circuit begins to form an output signal

with the same shape as the input signal, but with a 0.7V difference in amplitude. As the output voltage increases, the diode also charges up the capacitor **C1**. Once peak value is reached at the input signal, the voltage starts to fall. At this time, the capacitor is already charged up, and will start discharging at a DC level through the resistor following an RC time constant that is governed by the value of the resistor and capacitor. In this case:

$$\text{time constant} = \tau = RC = 2.0 \times 10^4 \times 1.0 \times 10^{-7} = 2^{-3}s$$

While the capacitor is discharging, the diode is in reverse bias, and output voltage is slowly dropping following the RC time constant. The output voltage will start rising again once the discharge reaches the turn-on value of the diode. After that, this phenomenon repeats periodically throughout the signal. The disadvantage of the basic peak detector is that the demodulated value will always be lower than the expected value, which makes the approximation imprecise. The reason being, there is a 0.7 difference between the peak values of the input and output signal because of the voltage drop across the diode. To evaluate this issue, we implement the Op-amp based peak detector, shown in the figure below:

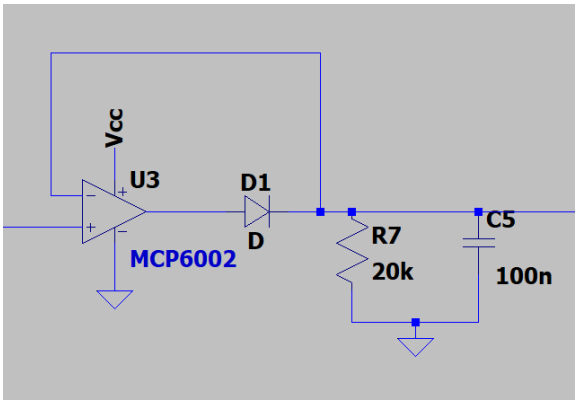


Figure 14: Operational Amplifier Based Peak Detector Circuit

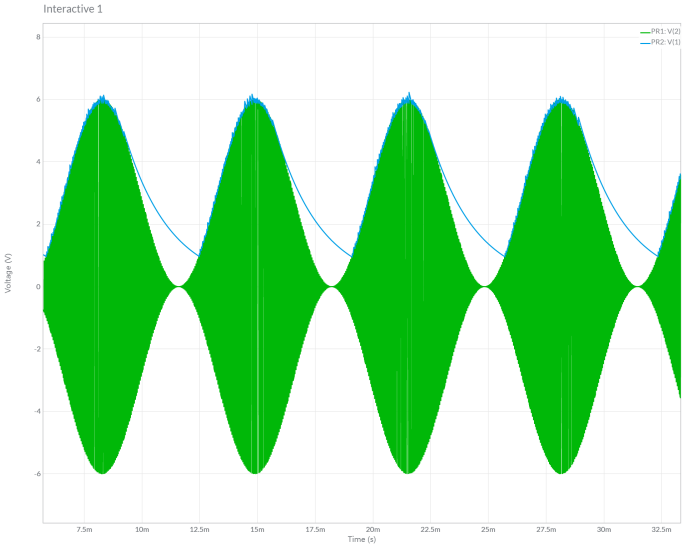


Figure 15: Operational Amplifier Based Peak Detector Circuit

By adding an Op-amp to the peak detector, we are utilising the negative feedback property to eliminate the voltage difference between the peak input and output signal. In the Op-amp, voltage at the negative and positive input are the same. Furthermore, voltage at the closed negative feedback loop is 0V, hence there is no voltage drop across the diode.

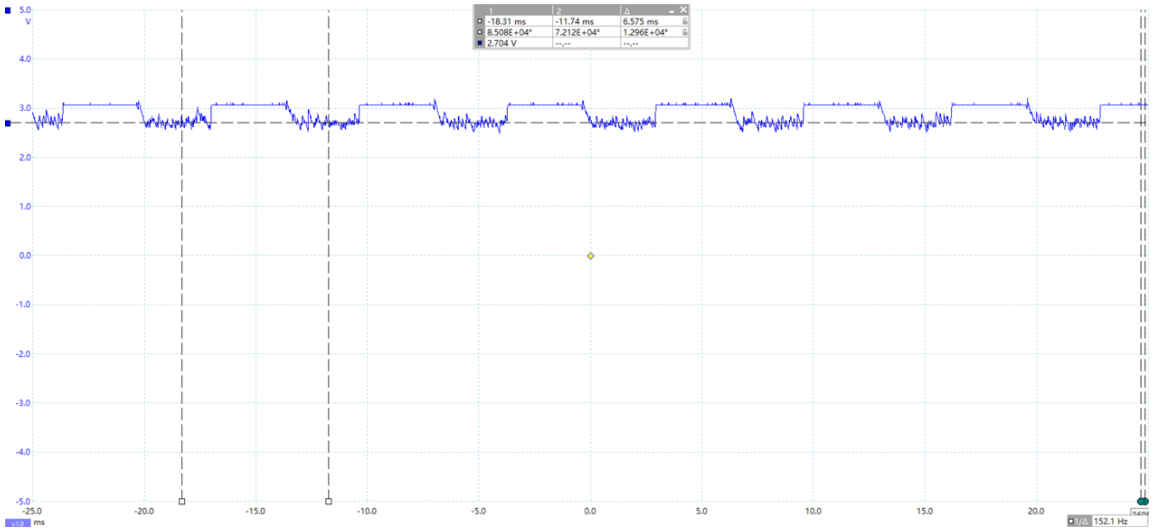


Figure 16: Op-amp Based Peak Detector Output Signal

4.1.4 Stabilization and Filtering

To minimise the effect of noise and distortion on the final signal output, the team has decided to implement a comparator circuit as the last stage of the radio signal processing circuit. In doing so, the signal output will appear more smooth and complete for detection.

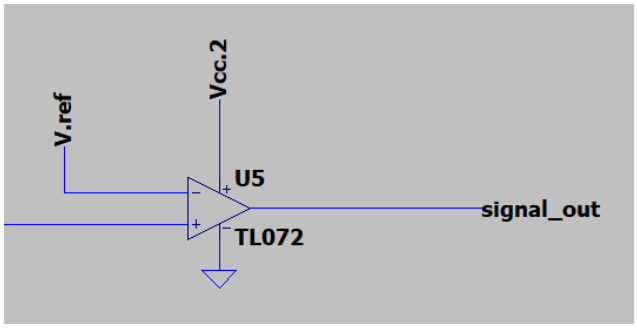


Figure 17: Radio Signal Comparator Circuit

The comparator circuit above indicates a non-inverting positive feedback amplifying circuit biased at 3V, with a potential divider connected to the negative input. To calculate the quiescent point:

$$V_{ref} = \frac{R9}{R8+R9} V_{CC} = \frac{3000}{2000+3000} \times 5V = 3V$$

The output of this voltage switches between its maximum and minimum output, depending on the input voltage. When the input is less than the DC voltage level at V_{ref}, the output switches to its minimum value 0V. As the non-inverting input is less than the inverting input, the output will be LOW and at the negative supply voltage, where V_{CC} results in a negative saturation of the output. When input is greater than V_{ref}, output will be HIGH, which results in its maximum value 2.2. Hence, the comparator essentially acts as a switch between 0V and 3V.

4.1.5 Complete Circuit

The complete circuit for radio signal processing consists of three stages, the amplification circuit for amplifying the carrier signal, the demodulation circuit to receive the demodulation frequency from the carrier frequency, and the comparator circuit to stabilize and filter the final output signal.

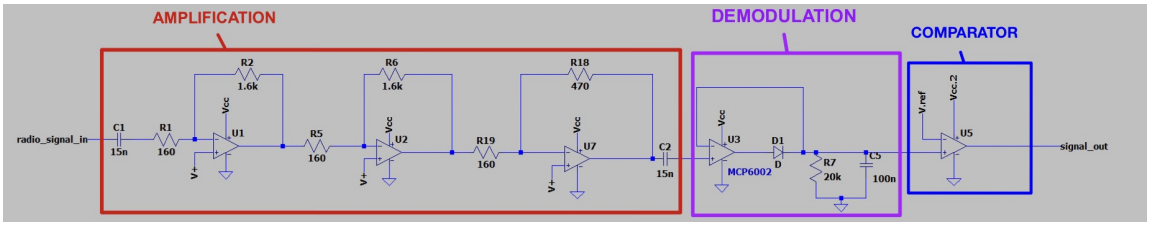


Figure 18: Radio Signal Processing Complete Circuit

4.2 Infrared

According to *Table 1*, the rover is required to detect infrared signals at two different frequencies, each representing *Property 1* of rocks Thiotimoline and Netherite. To accurately indicate detection, the project team has designed a circuit to amplify, stabilize and filter the signal received.

Figure 19: Infrared Signal Processing Method

Figure 19 demonstrates the signal processing stages for infrared signals. To receive the signal, the team uses the *KY-026 FLAME SENSOR* model as the sensor.

Equivalent to radio signal, initial infrared signals received have inconveniently low amplitudes, hence amplification is required for better indication of detection. After amplification, a comparator circuit is used to stabilize and filter the signal by minimising the effect of noise and distortion.

4.2.1 Infrared sensor

The team uses the *KY-026 FLAME SENSOR* model as the infrared sensor. The *KY-026 FLAME SENSOR* is packed with a photo diode that is sensitive to the spectral range of light created by an open flame. Hence, it is an appropriate sensor for infrared signal detection.

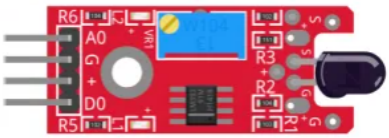


Figure 20: KY-026 FLAME SENSOR

Operating Voltage	$3.3V - 5.5V$
Infrared Wavelength Detection	$760nm - 1100nm$
Sensor Detection Angle	$60°$
Board Dimensions	$1.5cm \times 3.6cm[0.6in \times 1.4in]$

Table 2: KY-026 SPECIFICATIONS

As seen in *Table 2*, the sensor has an operating voltage between 3.3V and 5V. Therefore is suitable for the *Metro M0 Express* Arduino model used for this project. This module consist of a 5mm infra-red receiver LED, a LM393 dual differential comparator, a 3296W trimmer potentiometer, 6 resistors,2 indicator LEDs and 4 male header pins. The sensor has both digital and analogue outputs and the potentiometer is used to adjust the sensitivity.

4.2.2 Amplification

The *KY-026 FLAME SENSOR* contains a amplification circuit and has 3 main components on its circuit board.

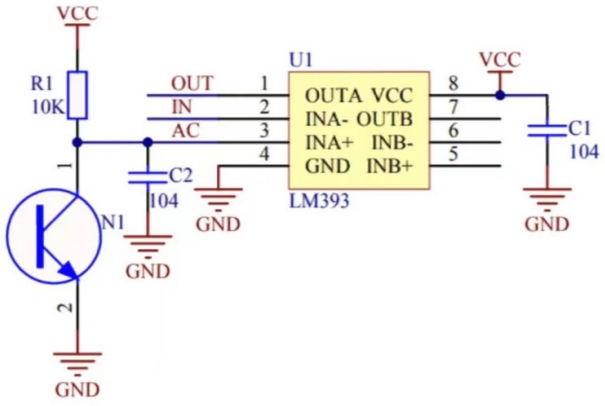


Figure 21: Infrared Signal Amplification Circuit

First, the sensor unit at the front of the module which measures the area physically and sends an analogue signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analogue output of the module. The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value. You can control the sensitivity by adjusting the potentiometer.

4.2.3 Stabilization and Filtering

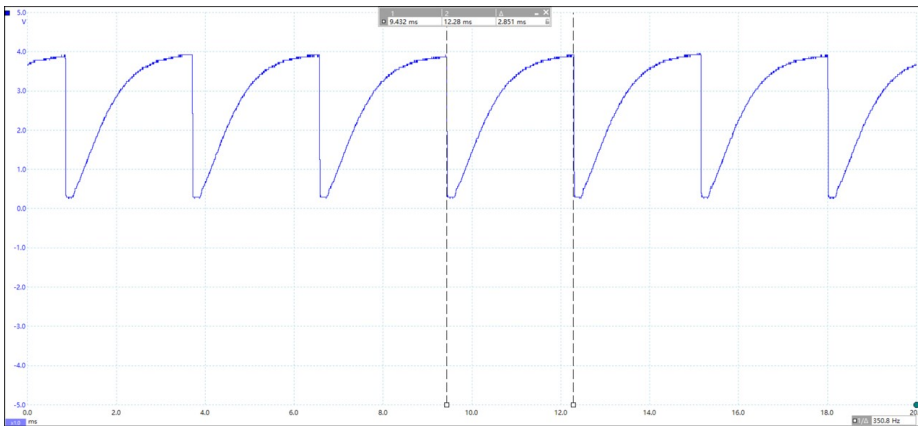


Figure 22: Initial Infrared Signal

Figure 22 shows the infrared signal output from the flame sensor. After being processed by the comparator circuit, the signal has already been stabilized and filtered, minimising the effect of external noise and distortion. However, the signal voltage has not reached the required 3.3V input voltage of the Arduino. As a consequence, the team has added a final comparator circuit to resolve this issue, shown in the figure below:

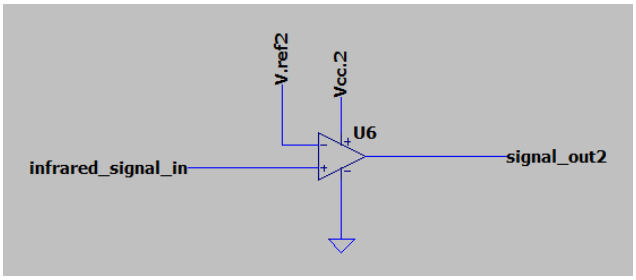


Figure 23: Infrared Signal Comparator Circuit

The circuit in Figure 23 functions equivalently to the radio comparator circuit, however with different component values.

In this circuit, the quiescent voltage can be calculated as follows:

$$V_{ref2} = \frac{R15}{R14+R15} V_{CC.2} = \frac{30000}{20000+30000} \times 3.3V = 1.98V$$

After biasing the signal, the circuit is able to switch between peak values depending on the input voltage. When **V+ V.ref**, the signal output will be the maximum output, 3.3V. When **V+ ¡ V.ref**, the signal output will be the minimum output, 0V. Essentially, the circuit acts like a switch between 3.3V and 0V, shown in the figure below:

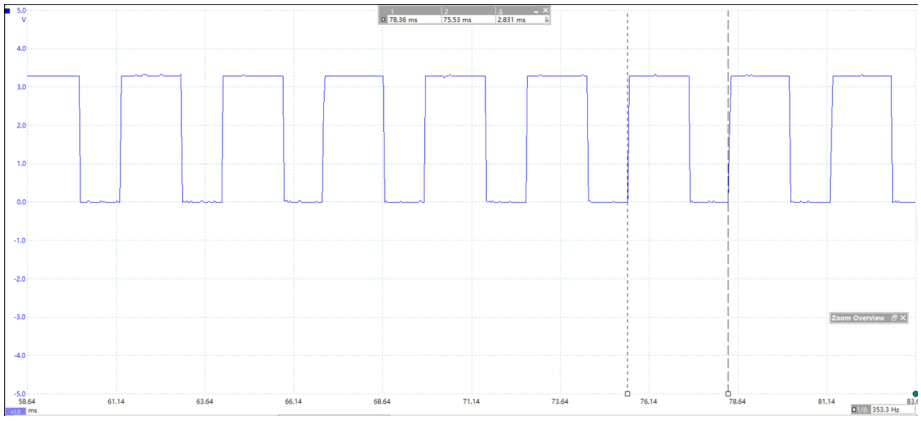


Figure 24: Infrared Signal Detected

4.3 Magnetic Sensing

- A coil of wire could be used to detect the static magnetic field produced by the Exorock. The coil has to be moving continuously to detect the magnetic field because the current would not be induced when the coil stays still. To implement this, an additional motor could be added to move the sensor, or detection could be carried out while the rover is in motion. However, adding another motor is inefficient considering the limited space of the rover, the weight and the extra current drawn. Additionally, detection and movement cannot happen at the same time due to limitations in programming. As a result, the KY-024 linear Hall effect sensor [2]is implemented to solve this problem. It is more energy efficient and takes up less space, withou requiring a motor.

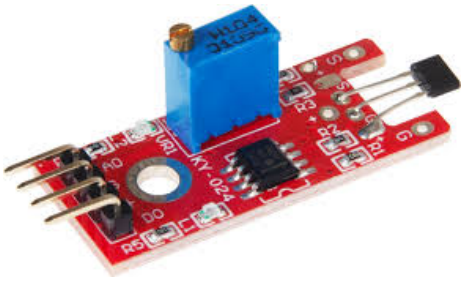


Figure 25: KY-024 Hall effect sensor

- There are two types of output in the Hall effect sensor: digital and analogue output. The digital output simply indicates the presence of magnetic field, synonymous with the LED on the sensor. The LED turns on when the magnetic

field is detected and turns off when it is not present. The Hall effect sensor also measures the current analogue value, which produces an amplified signal with a voltage value that depends on the strength of the magnetic filed. [3]. During testing, both digital and analogue out have similar performance and sensitivities. In the end, the analogue out was chosen simply because the Arduino did not have many digital pins available.

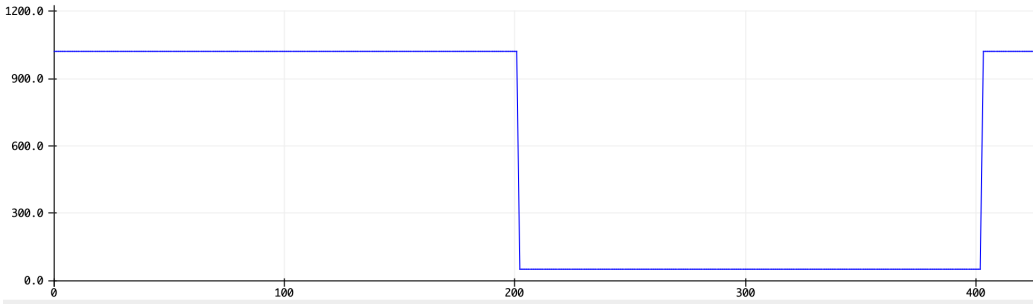


Figure 26: Analogue output from the Hall Effect Sensor. Around reading #200, a magnet with the North Pole pointing down was placed underneath the sensor.

- The Hall effect sensor was tested by connecting to Arduino with 5V input. *Figure 26* shows the analogue output. Initially, the analogue value stays between 800 and 1200 without the magnetic field. After it detects the downward magnetic field, the value decreases to around 50.
- A drawback of the sensor is that it can only detect a downwards-facing magnetic field, and would ignore an upwards magnetic field. To detect both the North and South poles of a magnet, two Hall effect sensors are used; with one being equipped upside down. Based on how each sensor responds, the orientation of a magnetic field can be determined.

Sensor 1	**Sensor 2**	**Output**
0	0	Neutral
0	1	North
1	0	South
1	1	Invalid

Table 3: Truth table for Hall Effect Sensor. Sensor 2 is placed upside-down

- The truth table, Table 3 shows how poles will be determined by using two sensors.

4.4 Acoustic Sensing

- In order to detect the acoustic signal, a 40kHz HC-SR04 ultrasonic distance sensor is used. The sensor consists of two ultrasonic transducers: a transmitter and a receiver. The transmitter converts the electrical signal to 40kHz ultrasonic sound pulses. Then, the receiver receives the sound and produces an output pulse which is proportional to the distance. This sensor is built for measuring the distance between the sensor and an object. However, it does not suit the rover use case as the sensor has to detect the acoustic signal from the rock, not from the transmitter.

  Due to the limitation of the ultrasonic sensor, it is impossible to use the receiver only, it is only able to be access if the transmitter is used at the same time. This is apparently a closed-source firmware issue, according to the Arduino forums. [4]

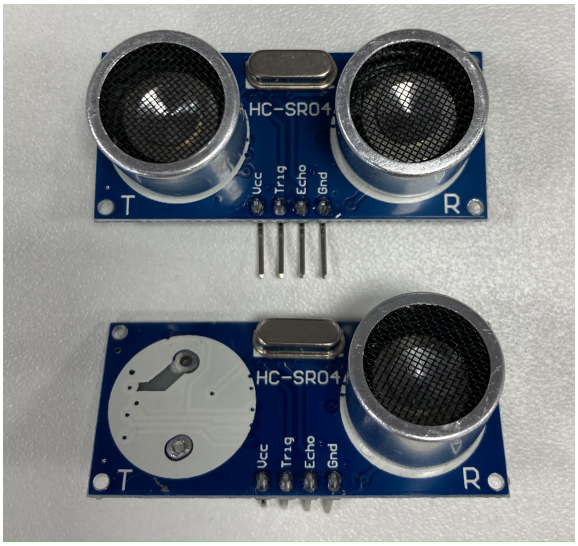


Figure 27: Ultrasonic Sensor - HC-SR04, second one with transmitter removed

- One workaround is the de-solder and remove the transmitter module from the ultrasonic sensor as shown in *Figure 27.* Then the transmitter is stimulated in the program (writing to the triggered pin), tricking the firmware into believing the sensor was triggered. This in turn enables the receiver (connected to echo pin) to receive a `pulseIn()` response. The concept is entirely made possible only because the rock emits 40kHz pulses.
- The time taken for the `pulseIn()` response is calculated and compared against

the measured duration when a 40kHz emitting rock is held close to the sensor.

4.5 Sensor Programming

4.5.1 Frequency Sampling (Radio and Infrared)

There are a variety of many methods to detect the frequency of sensors. Initial ideas considered by the team included various Fast-Fourier-Transform libraries and the Goertzel algorithm. Both techniques use mathematical methods to separate and identify frequencies at a reasonable computational cost.

Unfortunately during testing the rover sensor inputs were found to be very noisy and the sensor circuits' signal strength was not stable enough. This resulted in inaccurate and fluctuating signals.

Constant sampling and transmitting live data feeds at multiple exact frequencies would be too resource-intensive for the Arduino when coupled with a system for motor control. Eventually, given the processing limits of the Arduino, the team opted for a push-to-scan solution.

Idea	**Post-Testing**
The Arduino computes all the sensor frequencies in real time and transmits a live data feed to the remote constantly with exact frequencies	The Arduino takes frequency samples and outputs the result in discrete groups ranges when requested, via a single-threaded process.

Table 4: Table comparison of the concept and the final product

Due to the signal noise and the specification, it is more efficient to utilise frequency binning and sampling. In all cases there are only 2-3 possible group outcomes for a given property, for example, a 151Hz or a 239Hz radio signal, as shown in *Table 1*. During the detection process, the Arduino allows for tolerance in frequencies detected, and only returns the frequency group which is closest to the samples.

To turn the Arduino into an exact frequency meter (as depicted in the 'Idea' portion of Table *4*) would have been an excessive use of computation power for a superfluous feature.
In Snippet *1* a simplified algorithm concept is shown for the detecting an infrared signal of either 353Hz or 571Hz. Sample sizes and the duration of collecting samples can be controlled and increased in order to improve accuracy.

```
//Function constants
const int IRPin = 4;
const int IRSampleSize = 150;
const int reqIRSamples = 70;
int totalIRSamples = 0;
int samples353 = 0;
int samples571 = 0;

//Function to get IR Frequency
void infrared(){
  totalIRSamples = 0;
  samples353 = 0;
  samples571 = 0;

  while (totalIRSamples < IRSampleSize){
    int onTime = pulseIn(IRPin, HIGH, 20000);
    int offTime = pulseIn(IRPin, LOW, 20000);
    int timePeriod = onTime + offTime; //microseconds
    float frequency = 1 / (timePeriod / 1000000.0);


    if (frequency > 220 && frequency < 470){
      samples353 += 1;
    }
    if (frequency > 460 && frequency < 690){
      samples571 += 5;
    }

    totalIRSamples += 1;
  }


if (samples571 >= reqIRSamples){
    display.println(571);
    send("infrared", "571Hz");

  }
  else if (samples353 >= reqIRSamples){
    display.println(353);
    send("infrared", "353Hz");
  }
  else{
    display.println("Unknown");
    send("infrared", "N/A");
  }
  display.display();
}
}
```

Snippet 1: Frequency binning example

1. Measure the time taken a signal from the infrared sensor circuit to go from High to Low, and Low to High again. This is the period of the signal.
2. A rough frequency is calculated by dividing 1 by the time period.
3. If the frequency lies between 220Hz and 470Hz, it is likely the detected rock

frequency is probably 353Hz. If that frequency was between 460Hz to 660Hz, the detected rock frequency is likely to be 570Hz. Note that the groups overlap and one frequency could contribute to two tallies at once.

4. However, any of the readings could have been caused by noise and other factors, therefore a single reading by itself is not reliable. So a frequency tally is kept, and the process is repeated. Every time a frequency close to 353Hz (220-470Hz) is recorded the 353Hz tally incremented by one. If the frequency is close to 571Hz (440-660Hz), the 571Hz frequency tally is incremented by one.
5. This is repeated until the sample size is reached. In this case of the code, the sample size(variable `IRSampleSize`) is 150. The program than compares the tallies of both outcomes with each other and determines if one frequency tally passes the threshold(variable `reqIRSamples`), which is defined in the code as 70. If one does, then the result is the frequency corresponding to that tally.
6. In the event both tallies do not pass the threshold, there is no useful signal data to be gathered and the rock is likely emitting a different property.

4.5.2 Hall Effect Sensing (Magnetic)

As described in 4.3, the `analogRead` value of a Hall effect analogue output is between 800 and 1200 when unperturbed. If a value lower than than the given range is detected, there is a downwards facing magnetic field present.

After testing the sensor, it was observed that the signal from the Hall effect sensor had a very strong binary effect; only when the magnet is close by at a certain point, the `analogRead` value falls abruptly.

Given the very definite nature of the setup, no sampling is needed for detecting the direction of the magnetic field. Snippet 2 shows the team's prototype code for detection.

```
#include <Wire.h>

int pin1 = A0;
int pin2 = A1;
int x;
int y;

void setup()
{
Serial.begin(9600);
pinMode(pin1,INPUT);
pinMode(pin2,INPUT);
delay(1000);
}

void loop()
{
x = analogRead(pin1);
y = analogRead(pin2);

if ( ((x>800)&&(x<1200)) && ((y>800)&&(y<1200)) ){
  Serial.println("Neutral");
}
else if ((x>800)&&(x<1200) && !((y>800)&&(y<1200)) ){
  Serial.println("North");
}
else{
  Serial.println("South");
}

delay(1);
};
```

Snippet 2: Determining the pole of a magnet held to the sensor

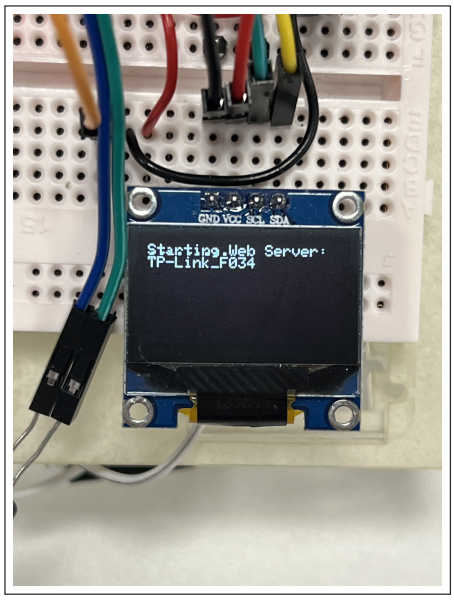
4.6 Remote System Design and Programming

Figure 28: System Environment

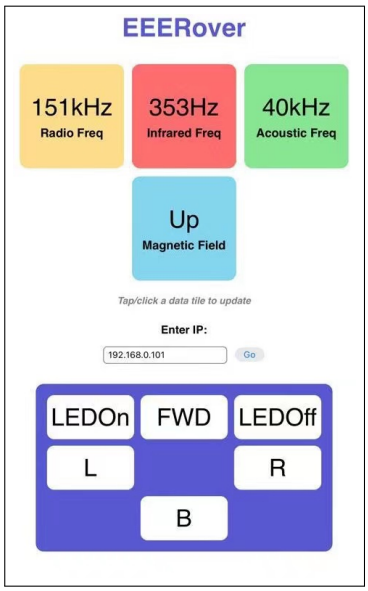
The system setup for the rover consists of a router, a remote control device (in most cases, a smartphone), and the rover.

A 2.4GHz router is used to provide the network environment for the remote, instead of the EEERover network, as the latter network is not always available. It allows for more flexibility in workflows. Both the Arduino and the remote control connect to the router.

4.6.1 User Interface



(a) OLED Screen On Rover



(b) HTML User Interface

- The rover has an OLED screen attached to the `SDA` and `SCL` pins of the Arduino. The screen is used to display diagnostic data including the rover's IP address, frequency sampling statistics, and any error logs.
- The remote control consists of four tiles that visualise the results of frequency detection from the Arduino.

The remote control device runs a web app created using HTML, Cascading Style Sheets (CSS), and JavaScript. All three languages are embedded directly into a single HTML file under a `<script>` and `<style>` tag for both JavaScript and CSS respectively. Combining all the files together was simple solution which allowed testing to be done quickly.

The web app interface (see Figure *30*) consists of three parts – the motor commands, the sensor view, and the IP address entry.

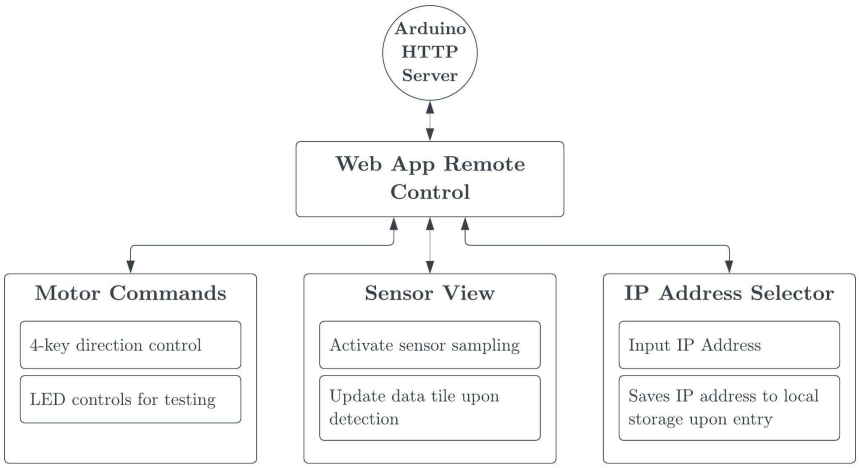


Figure 30: Frontend model for the remote control

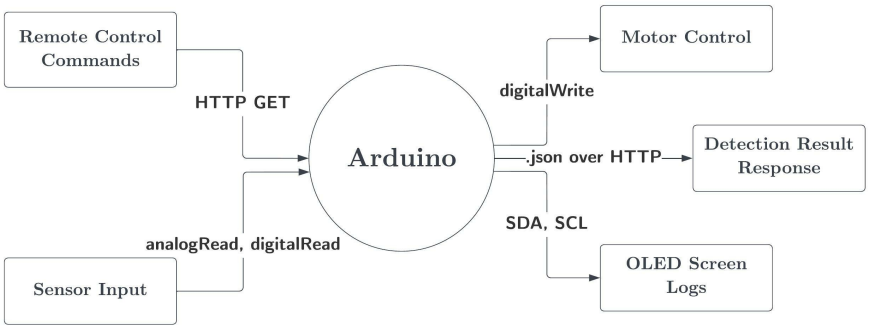


Figure 31: Arduino system context diagram

The Arduino acts as a HTTP server to stream the detection results from its sensors to the web app remote control. It also receives motor movement commands from the remote and directs its motors to run accordingly.

The pre-provided started code originally designates the Arduino to transmit the web app to the phone, unfortunately, the WiFi shield and its corresponding library both have a very small header limit on file transfer sizes. If the web app is too large (30 lines of code was enough to render it useless during testing) it will refuse to transmit over, causing the Arduino becoming unresponsive.

In order to solve this issue, the team pre-loads the web app onto the remote (using a web development tool), and it is stored locally. The benefit of this is that the remote is immediately ready on startup.

To operate, the phone is connected to the router and the web app. In most cases the web app knows the preset IP address of the rover, however it can be changed within the app itself. The remote control also uses the `Window.localStorage` Web API to locally store the last-used IP address so it can be retrieved again even after a refresh.

Whenever a function on the remote is interacted with, the remote sends a GET HTTP request to the Arduino. In response, the Arduino creates a static JSON file using the ArduinoJSON open-source library. This JSON file contains the results of the scan and transmits it to the remote control. The response text (accessed using

`this.responseText` under the XMLHttpRequest object) is parsed into a dictionary file structure in JavaScript and the remote's data tiles are updated accordingly. After the static JSON file is used, its memory is cleared and de-allocated to reduce strain on the Arduino's memory. This de-allocation is done automatically at the end of the function call. [5]

```
//Pre-requisites
#include <ArduinoJson.h>

//Function to send request responses within a JSON
void send(String key, String value) {
  StaticJsonDocument<300> doc;
  String response;
  doc[key] = value;

  serializeJson(doc, response);
  server.sendHeader("Access-Control-Allow-Origin", "*");

  server.send(200, F("application/json"), response);

}
```

Snippet 3: Sending a JSON HTTP response example

The ArduinoJSON library was chosen due to its performance being 10% faster and its memory use being 10% less than the official Arduino_JSON library. [6]

4.7 Testing

Every sub-modules are tested individually before assembled together. Throughout the process of testing the maneuverability of the moon rover, the team ran into several issues.

1. The rover was 'slipping' to the right. This was caused by imperfect wheels and uneven weight distribution resulting in one wheel gaining more traction than the other. We eventually decided to redesign the chassis with four motors and thicker wheels instead. Extending the design by adding an inverted EEEBug printed circuit board (PCB) with a reversed servo setup would partially reduce the unequal weight distribution. This would be farther alleviated by switching to larger tyres that would increase traction all around the rover, making any disparities in traction relatively smaller in comparison.

2. Due to our setup separating the remote control web app and the Arduino, the team ran into web server problems caused by modern browser security features which turned block any non-indicated Cross-Origin Resource Sharing (CORS) requests by default [7]. Whenever the Arduino sent a JSON data log to the remote control, the browser would terminate the request before any data could be parsed. Further research into the WifiWebServer Arduino library was carried

out and the solution was to send a header file that turned enabled CORS before sending over the JSON file. This solution has been implemented on line 11 of Snippet 3.

3. When prototyping the remote control, the original remote control web app made a HTTP get request to update all the data tiles every 100 milliseconds. Unfortunately this crowded the requests for movement controls, causing the rover to have a latency in between movements, and sometimes the remote control would become unresponsive. In the end, the team decided to no longer use a live data feed (see Table 4), and the sensor results area only sent over when requested.

5 Overall Performance Evaluation

6 Conclusion

References

[1] "Tl07xx low-noise fet-input operational amplifiers," https://www.ti.com/lit/ds/symlink/tl072.pdf?ts=1655552674558&ref_url, Sep. 1978, accessed: 19-6-2022.

[2] "Ky-024 linear magnetic hall sensor," https://pdf1.alldatasheet.com/datasheet-pdf/view/1284503/JOY-IT/KY024.html, Mar. 2021, accessed: 19-6-2022.

[3] "Hall effect sensor," https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html, Mar. 2020, accessed: 19-6-2022.

[4] "Using the ultrasound sensor in receive only mode," https://forum.arduino.cc/t/using-the-ultrasonic-module-in-receive-only-mode/896980, Aug. 2021, accessed: 19-6-2022.

[5] B. Blanchon, "arduinojson documentation," https://arduinojson.org/v6/how-to/reuse-a-json-document, 2021, accessed: 19-6-2022.

[6] ——, "arduinojson home page," https://arduinojson.org/, 2021, accessed: 19-6-2022.

[7] "Cross-origin resource sharing (cors)," https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS, 14 Jun. 2022, accessed: 19-6-2022.

7 Appendix

7.1 Product Design Specification

7.1.1 Performance

Our goal is to be able to maneuver the rover through the artificial moon terrain effectively and quickly while avoiding obstacles. Able to detect and classify rocks based on their properties. While minimizing lag delays between web-server and Adafruit wifi shield, so our commands from remote control will be time effective.

7.1.2 Environment

The testing environment should be smooth surface with some uncrossable obstacles. The testing moon rocks will be distributed across the environment. The room will be set at room temperature. The pressure of the room will be at atmospheric pressure. The lighting in the room could be maintained to prevent interference.

7.1.3 Testing Product Cost

The project should control the budget that does not excess 60 GBP. Identify ways to make the rover effective without purchasing many resources and remain eco-friendly.

7.1.4 Size and Weight

We desire our rover to be as lightweight as possible and should be less than 1.0 kilograms. Less weight is easier for maneuver on motors and extends the batteries life time.

7.1.5 Material

We are using only eco-friendly acrylic material to build the base and frame of our rover. It should be strong enough to lift its weight less than 1.0 kilograms and light enough to maneuver adeptly. It is also budget friendly that are able to fit in our beget that is less than 60 pounds.

7.1.6 Aesthetics, Appearance and Finish

We are aiming to design our rove to be lightweight. We will make sure all wires don't have any bare ends. Reduces any noise or interference on sensing. Insure everything are correctly screwed in and different components are fixed in place.

7.1.7 Ergonomics

The project team is aiming to design the rover and control systems as user friendly as possible and our user interface we design is easy to learn and use. It is also age-friendly that it has big buttons.

7.1.8 Quality and Reliability

We desire our sensors to be in high qualities, since signals tends to be very weak, and noises are not very desirable. We are aiming to design our rover to be very robust and circuits and sensors will not break during collisions with rocks or other rovers while manoeuvring.

7.1.9 Process

There are several steps to be taken before the assemble of the rover and they are the followings:

- Circuit implementations for radio wave, infra-red, acoustic and magnetic sensing.
- Remote control platform for the user to be able to maneuver the rover.
- Sensor programming to be able to determine the frequencies and identify which type of rock that are in front of the rover.
- Structure design aiming for designing a robust, small, cost effective and high quality rover.
- Testing is for simulate competition environments.
- Upgrade

7.1.10 Disposal

Dissemble and able to recycle rover parts such as sensors, cables, wires and batteries. The project team has recycled batteries used during the testing of the rover.