EERover Final Report

-Team Ytterbium-

1 Task Requirements

The aim of this project is to design and build a remote controlled rover that can explore a hazardous store room and identify unusual materials. Each material emits a unique combination of signals (figure 1) and is stored in an unlabelled container drum (figure 2).

Name	Property 1	Property 2
Gaborite	67kHz radio modulated at 151Hz	Acoustic signal at 40.0kHz
Nucinkisite	67kHz radio modulated at 239Hz	Magnetic field
Durranium	103kHz radio modulated at 151Hz	Magnetic field
Brookesite	103kHz radio modulated at 239Hz	None
Cheungtium	Infrared pulses at 421Hz	None
Yeatmanine	Infrared pulses at 607Hz	Acoustic signal at 40.0kHz

Figure 1*: Properties of each material (Department of EEE, 2017:p.9).*

Figure 2*: Container drum. It has a diameter of 58mm and a height of 112mm. (Department of EEE, 2017: p.6).*

The rover should possess the following qualities:

1. Good accuracy in identifying all materials
2. Cost and weight effective
3. Manoeuvrable enough to negotiate an artificial lab environment (unknown as yet)
4. Robust and reliable construction
5. Logical remote control interface that is easy to use

These are further elaborated in Design Criteria below.

1.1 Design Criteria

From the 32 PDS, 5 most important criteria was shortlisted: Performance, Cost, Size, Weight and Environment.

1.1.1 Performance

The rover's main function is to inform the user about the unknown material in the drum. To do so, it must be sensitive enough to pick up the emitted signals and also analyse the signals accurately to determine the material's identity. Then, it needs to transmit the information to the user in real time with little time-lag. This information should be easily interpreted by the user via a clear user interface. Similarly, the user should be able to move the rover towards the drum with little time-lag using the interface. Finally, the rover should be structurally stable and manoeuvrable enough to access the drum with no mishap.

1.1.2 Cost

All development costs are to be kept within £50. This limits our options for rover design.

1.1.3 Size

The rover should not be too big or too high- a good gauge based on the EEEBug would be 30cm x 20cm x 10cm. This restricts the space available for sensors and circuits. The arduino also has a limited number of pins - 6 analogue and 14 digital pins.

1.1.4 Weight

Weight of the rover should be evenly distributed to maximise stability. The rover should also be lightweight.

1.1.5 Environment

The environment is largely unknown. The drums could be lying in any orientation, and this affects the signal strength detected by the sensors which are held in fixed positions relative to the rover. The terrain of the floor could affect the traction of the wheels. Obstacles present could make it a challenge for the rover to access the drums.

2 Rover Design

2.1 High Level Design

The current prototype, with the sensors, circuits, movement, signal processing, communication and interface (figure 4) systems complete, is shown in figure 3. The structural design, however, is still in progress.

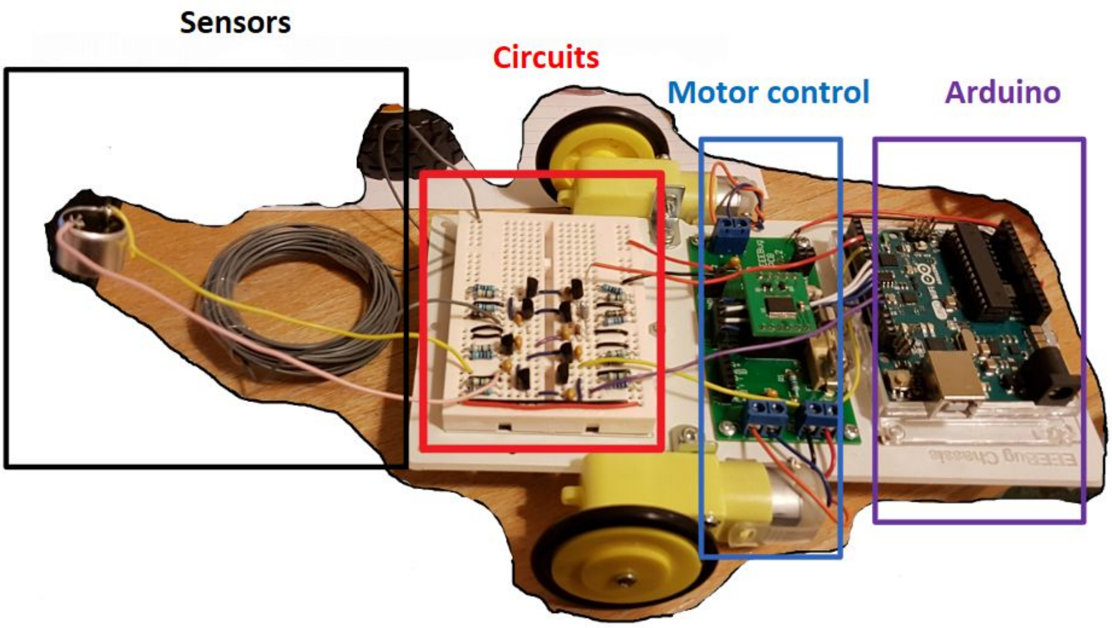


Figure 3: *Rover prototype*

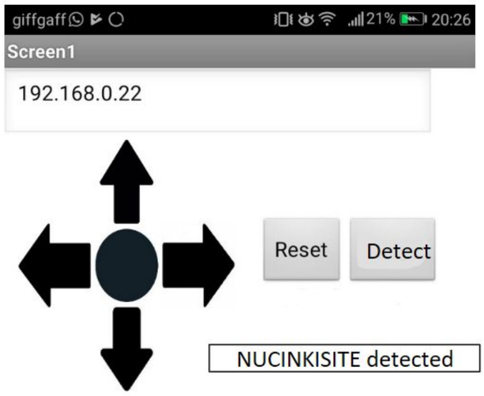


Figure 4 *Remote control interface*

The rover is controlled by a smartphone app via bluetooth. In order to identify the materials, the user will move the rover towards the drums. Once the rover is near enough, the user will press the “Detect” button (figure 4) to activate the signal identification algorithm, which then displays the material detected on the interface. Once the rover moves on to the next drum, the user will press “Reset” (figure 4) to clear the screen before detecting again.

A minimalistic design is employed to identify all the materials using just two sensors: a radio antenna coil and an ultrasonic sensor. Although some of the materials also emit infrared and magnetic signals, they do not have to be considered in order to identify the materials (see section 2.2.3).

The arduino pins are assigned to the following functions (table 1):

Arduino pin	D10	D9	D7	D8	D12	D5	D3
Function	Left wheel speed	Right wheel speed	Left wheel direction	Right wheel direction	Sound freq	Modulation freq	Carrier freq

Table 1*: Assignment of arduino pins*

2.2 Detail design

The rover can be subdivided into three subsystems: Sensors and Circuits, Signal Processing, and Movement.

2.2.1 Sensors and Circuits:

2.2.1.1 Radio sensor: Tuned coil antenna

The radio antenna is an air-cored inductor made up of coiled wire (see figure 55). A tuned circuit most sensitive at the resonant frequency is formed when an appropriate capacitor is added to the antenna.

Inductance of antenna, $L = 84.3\mu H$.

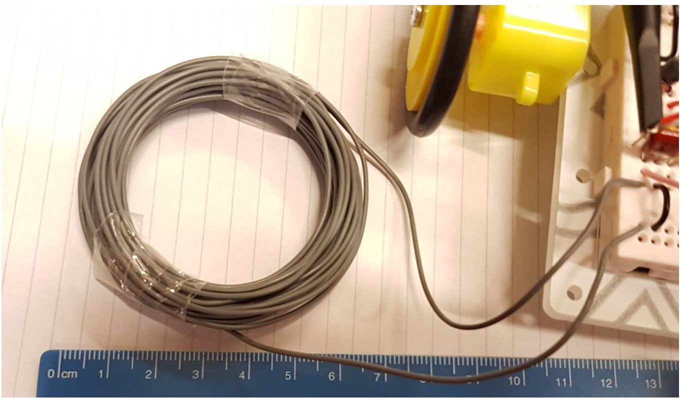


Figure 5*: Radio antenna, diameter = 5cm*

2.2.1.2 Ultrasonic sensor:

The ultrasonic sound sensor (figure 6) used is the receiver from the HC-SR504 ultrasonic ranging module.

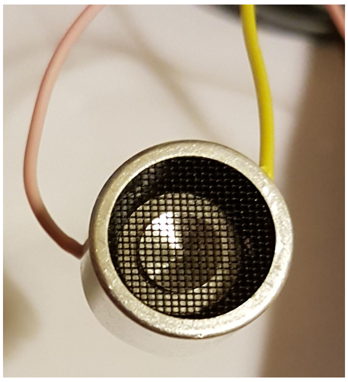


Figure 6*: Ultrasonic sensor*

2.2.1.3 Circuit Design

Our final design circuit (figure 7) contains 4 sets of amplifiers and 1 demodulation circuit. These circuits combine to form the top two sets of radio signal circuits and the bottom set of acoustic signal circuits.

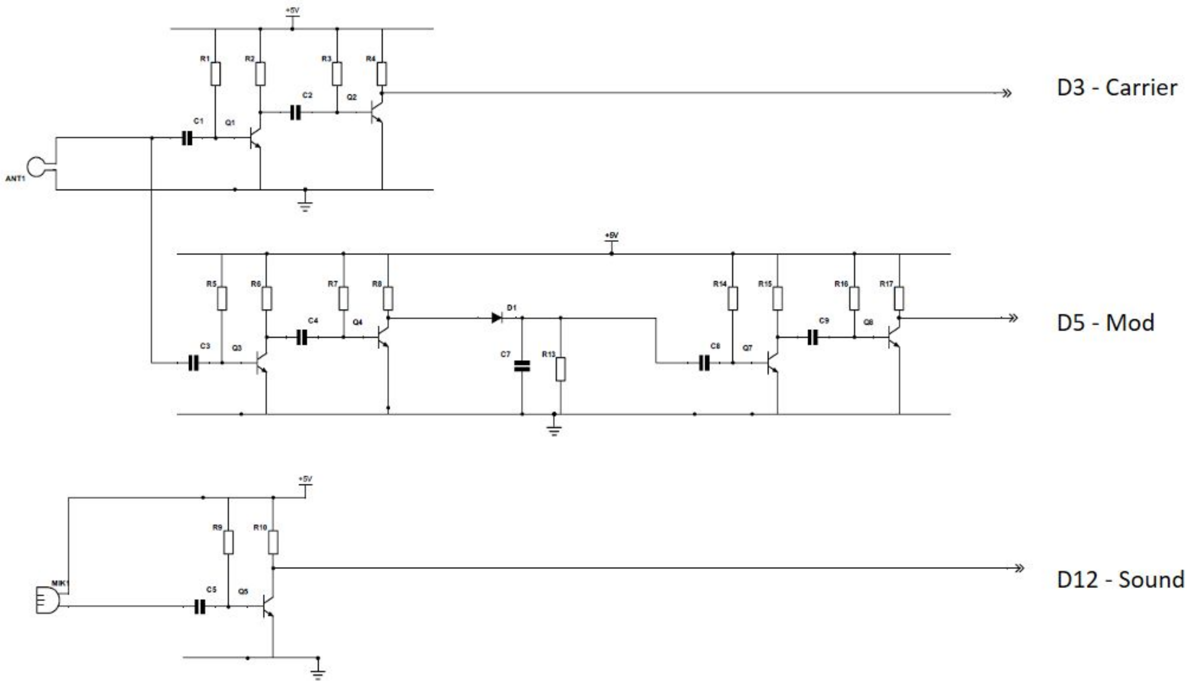


Figure 7*: Overall circuit design*

The inputs and outputs are shown in table 2.

Inputs	**Outputs**
Radio signal input	Carrier frequency
	Modulating frequency
Ultrasonic signal input	Acoustic signal frequency

Table 2: *Circuit inputs and outputs*

2.2.1.4 Radio signal processing circuit

Two distinct carrier frequencies (67kHz, 103kHz) and two modulating frequencies (151Hz, 239Hz) are to be detected. This can be achieved using the circuit below (figure 8).

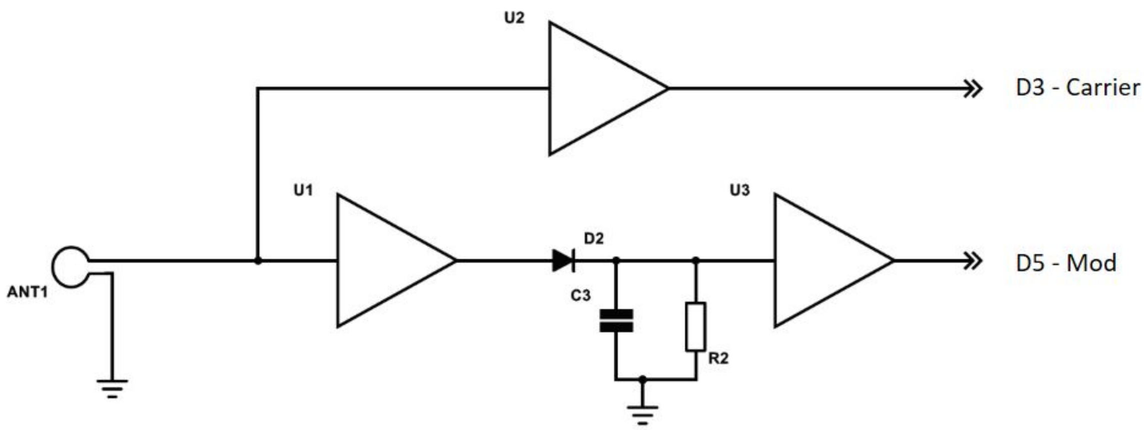


Figure 8: *Simplified radio circuit*

The radio signal input is directly amplified without the need for tuning capacitors as we are using the arduino determine the signal's frequency.

The demodulating branch also contains an amplifier after demodulation to boost the signal after it has passed through the diode, which is then sent to the arduino to read the modulating frequency. C and R values are chosen to be C = 1 μF and R = 620Ω. (See derivation in section 4.1.1.1.3).

2.2.1.5 Sound signal processing circuit

The ultrasound signal input is directly amplified and sent to the arduino (figure 9). It uses one less BJT than that of radio because the original input signal is relatively stronger.

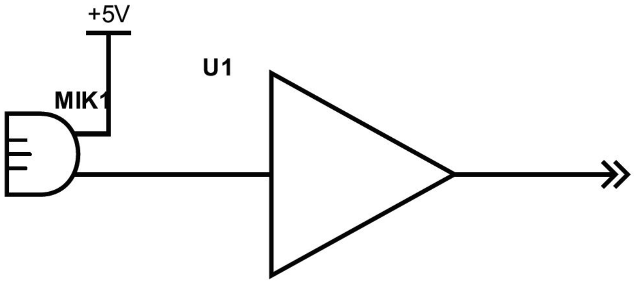


Figure 9*: Simplified sound circuit*

2.2.2 Amplification and Schmitt Trigger circuit

The amplifier (figure 10) consists of two identical cascading common-emitter BJT amplifiers. The BJT used is a BC337 NPN transistor. The R_B and R_C values are chosen arbitrarily: $R_B = 4.3M\Omega,\ R_C = 5.1k\Omega$. (See [section 2.4.3.1](#) on circuit explanation)

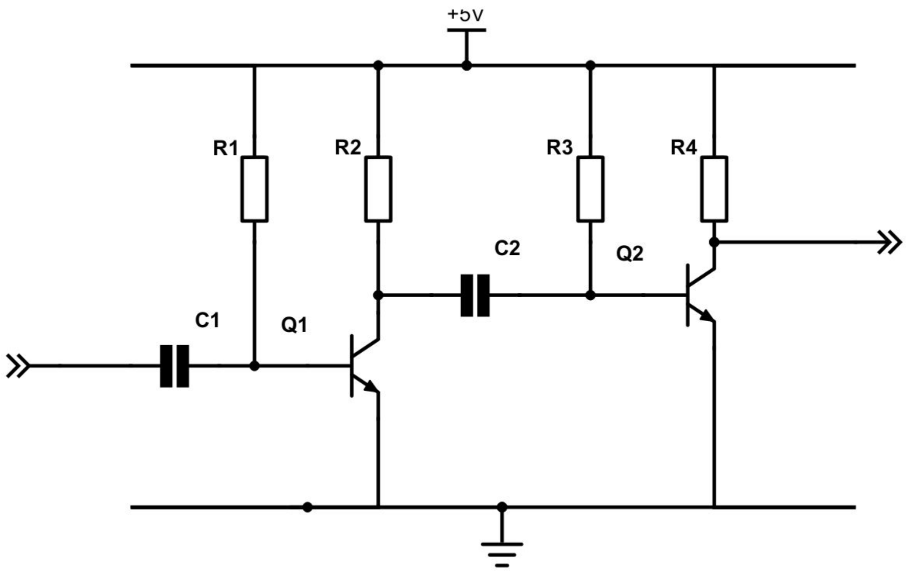


Figure 10*: Amplification circuit*

2.2.3 Signal Processing

For signal processing, the arduino has two main tasks: reading the signal frequency and identifying the material present based on the signals detected.

2.2.3.1 Measuring signal frequency

Two main methods were used to measure signal frequency: the pulseIn function and the FreqCounter library.

2.2.3.1.1 The pulseIn function

The *pulseIn* function is used to calculate the period and frequencies of the signals. This function works only on digital pins, so all the analogue signal outputs from the circuits are fed into digital pins. However, the digital pin will read low for any voltages below 1.5V and high for any voltages higher than 3V. Anything in between will be undetermined (change from high to low randomly). Hence, it is important that the input signal has a square waveform (i.e. only high or low) for *pulseIn* to work accurately.

Nevertheless, inaccuracy cannot be avoided because digital pins are used to read analogue signals which are not perfect square waves. Table 3 shows the deviations of the arduino readings from the actual frequency.

Frequency	**Lower reading**	**Higher reading**
67kHz	50000	63000
103kHz	80000	110000
40kHz	34500	39000

Table 3*: Frequency range returned when using pulseIn to measure frequencies*

Despite these inaccuracies, each frequency has a well-defined range, which will allow us to differentiate them. When using the sound sensor (40kHz) there was a background signal of around 33000, which could lead to identifying the wrong drum.

The *pulseIn* function works as shown in figure 11.

```
180        timeh = pulseIn(3, HIGH);
181        timel = pulseIn(3, LOW);
182        ttime = timeh + timel;
183        frq = 1000000 / ttime;
```

Figure 11*: Code using pulseIn function*

In this case, pin 3 is set to measure the carrier frequency. Line 180 will wait for the signal to go HIGH, and measure the time it stays HIGH, whereas line 181 will do the same when the signal is LOW. This time is measured in microseconds, and that’s why when calculating the frequency, the frequency is taken as 1000000 by the total time.

As for the modulating frequencies, since the difference between them is much smaller (around 90Hz), the *pulseIn* function is not able to differentiate them correctly. Hence another method was needed

2.2.3.1.2 FreqCounter library

The Arduino itself has a pin to measure frequency (for Arduino Uno Wi-Fi it is pin 5). For this the FreqCounter.h library[1] is used (figure 12).

```
FreqCounter::f_comp= 8;
FreqCounter::start(1000);

while (FreqCounter::f_ready == 0)
mod=FreqCounter::f_freq;
```

Figure 12*: Code using FreqCounter library*

The first line is used to compensate small errors in the readings, whereas the second line sets the resolution of the reading. In this case it is set to 1000ms, so that it measures to the nearest Hz (±1Hz). The third line waits for the counter to be ready and the last line just reads the frequency.

FreqCounter works accurately for the modulating frequency as seen in table 4.

Signal	**Reading**
151	152
240	239

Table 4*: Signal frequency vs actual reading*

2.2.3.2 Identifying the material present

Each material can be identified based on the presence or absence of radio and acoustic frequencies (table 5).

Material	**Signal**				
	Radio carrier 67kHz	**Radio carrier 103kHz**	**Radio modulating 151Hz**	**Radio modulating 239Hz**	**Acoustic 40kHz**
Gaborite	1	0	1	0	1
Nucinkisite	1	0	0	1	0

[1] Download link of the library: **http://interface.khm.de/index.php/lab/interfaces-advanced/arduino-frequency-counter-library/**

Durranium	0	1	1	0	0
Brookesite	0	1	0	1	0
Cheungtium	0	0	0	0	0
Yeatmanine	0	0	0	0	1

Table 5*: Logic used to identify drums*

2.2.4 Movement

The motors are controlled by the provided H-bridge through two different parameters per wheel: DIR and PWM. These can be connected directly to the digital input pins of the Arduino. The DIR pin will determine the direction the rover moves in (HIGH for rotating forward, and LOW for rotating backwards), and the PWM will determine the velocity (duty cycle). These two parameters are easily controlled by the Arduino with digitalWrite for DIR pins, and analogWrite for the PWM pins.

The code below (figure 13) shows how the arduino controls the motors in response to commands received from the user interface. Each command creates a different combination of active digital pins.

```
if (currentLine.endsWith("GET /forward")) {
  digitalWrite(dirl, HIGH);
  digitalWrite(dirr, HIGH);
  analogWrite(left, 255);
  analogWrite(right, 255);
  // forward
}
if (currentLine.endsWith("GET /left")) {
  digitalWrite(dirl, HIGH);
  digitalWrite(dirr, LOW);
  analogWrite(left, 255);
  analogWrite(right, 255);
  // left
}
if (currentLine.endsWith("GET /backward")) {
  digitalWrite(dirl, LOW);
  digitalWrite(dirr, LOW);
  analogWrite(left, 255);
  analogWrite(right, 255);
  // backward
}
if (currentLine.endsWith("GET /right")) {
  digitalWrite(dirl, LOW);
  digitalWrite(dirr, HIGH);
  analogWrite(left, 255);
  analogWrite(right, 255);
  // right
}
if (currentLine.endsWith("GET /stop")) {
  analogWrite(left, 0);
  analogWrite(right, 0);
  // stop
}
```

Figure 13*: Code showing how arduino controls motors*

3 Design Process

The design process is a cycle of generating new concepts, selecting ideas to implement and troubleshooting what went wrong.

3.1 Concept Generation

We learnt various concept generation techniques, and below is an example of how we used *Brainwriting* to generate ideas for the traction subsystem.

3.1.1 Brainwriting example: Traction subsystem

The terrain could be sandy or slippery, and thus we needed to design a traction system that would overcome such conditions. In order to produce creative and diverse ideas, we tried brainwriting, which is brainstorming without talking. Everyone had some post-its on which ideas were drawn out and pasted on the wall at their own time. In this way, the problem of discussions being dominated by certain people, leading to an unproductive session, is reduced. As we saw the ideas of other people, we became more inspired and more unconventional ideas were produced, including flooding the arena to eliminate traction totally. Eventually after about 20 minutes, we stopped to look at all the ideas in their entirety, and arranged related ideas together, which produced the following arrangement (figure 14).

Figure 14*: Brainwriting post-its arranged*

Following the session, we sorted out our ideas, and produced a rough preliminary sketch of the more feasible ideas (figure 15).

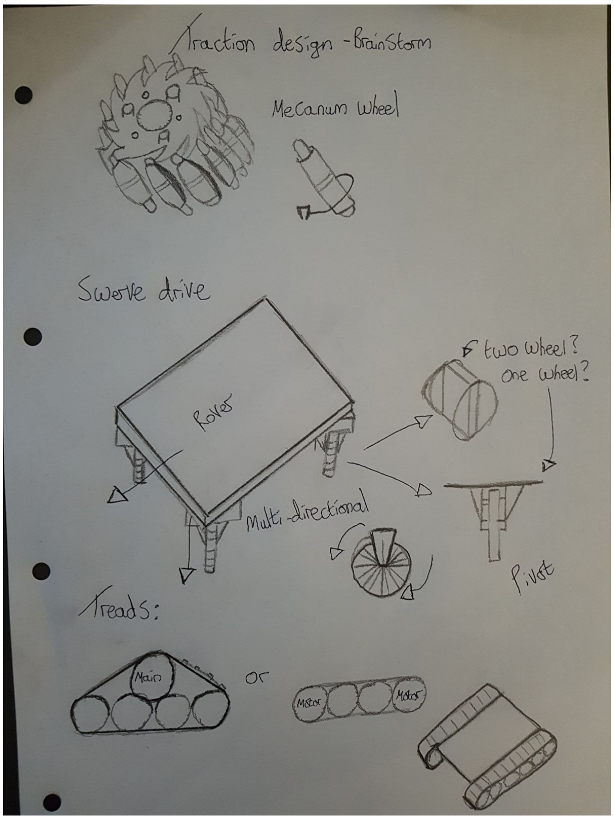


Figure 15*: Sketch of brainwriting ideas*

Ultimately, none of these ideas were implemented due to time, skill and cost limitations, but it was still a good exercise to explore what options we had.

3.2 Concept Selection

After generating many ideas, we needed to shortlist the best ones to implement.

3.2.1 Matrix Method: Selecting material identification strategy

3 material identification strategies were devised based on different combinations of sensors used, assuming that the arduino does not read the frequency directly. 6 criteria with different weights are considered: Cost, space, sensitivity, speed, complexity and ease of maintenance. The top 3 criteria - sensitivity, space and complexity - are chosen to score these strategies (table 6) because they are by far the most important constraints we are facing. While cost and speed are also relevant, all 3 strategies do not differ much in these respects and hence they were not chosen as scoring criterion.

		#1			#2			#3		
Sensor combi		Radio + IR			Radio + IR + Magnetic + Sound			Radio + Sound		
Criteria	Weight	Value	Score	W x S	Value	Score	W x S	Value	Score	W x S
Cost	5	IR: 0.518			0.518 + 2.47 + 3.5 = 6.488			3.50		
Space	7	A lot of space for many filtering circuits, on top of amplification and demodulation	1	7	Less space, only filtering of radio carriers required, no need demodulation but still need amplification	8	56	Less than #1, needs filtering and demodulation	6	42
Sensitivity	10	IR insensitive	1	10	IR, Magnetic insensitive	1	10	High	9	90
Speed	5	Fast			Fast			Fast		
Complexity	7	High, a lot of circuit design	1	7	Low, not much circuit design required	3	21	Relatively little circuit design except for radio	3	21
Ease of maintenance	7	Difficult, too many circuits, easy to make mistakes			Less difficult, but more sensors means greater chances of things going wrong			Low chance of error		
				24			**87**			**152**

Table 6*: Matrix Method to select best material identification strategy*

Clearly #3 is the best strategy due to the overriding weight of sensitivity as the test results (section 4) showed that with the drum cover in place, little to no signals were detected on the IR and hall effect sensor, and hence this is the one implemented, except with the arduino reading the frequency directly, thus negating the need for radio carrier filters.

4 Technical Development Process

Before finalising, we experimented with different options regarding material identification, communication and structural design.

4.1 Material Identification

We tried out three different combinations of sensors so as to identify all the materials efficiently.

4.1.1 #1: Radio + IR

At the beginning, we wanted to use just two sensors with filters to distinguish the six different drums: antenna for the radio waves and IR receiver for the IR pulses.

4.1.1.1 Radio

4.1.1.1.1 Radio antenna design

Initially, we researched on different types of radio antennas (USNA, 2013), such as the dipole antenna and small loop antenna so as to find the best design that is most sensitive and suitable for our frequency range. A lot of sources suggested that the antenna length should be equal to or at least a quarter of the radio wavelength to allow resonance (HOPERF, n.d). This confused us greatly since it would mean that to detect the 67kHz carrier frequency, we would require at least $length = 0.25 \times \frac{3\times10^8}{67000} = 1119m$, clearly impractical to implement. It was the wrong research direction, because the rover does not require such a high-performing antenna, and we eventually realised that a simple coil of wire that acted as an inductor sufficed.

We made a coil of 5cm in diameter with 30 turns using solid-core wire. It was measured to have an inductance $L = 84.3\mu H$.

4.1.1.1.2 Radio antenna performance

Since the signal picked by the antenna (figure 16) is in the range of tens of mV peak to peak depending on distance (table 7 and figure 17), we have initially used the TL072 op amp to amplify the signal.

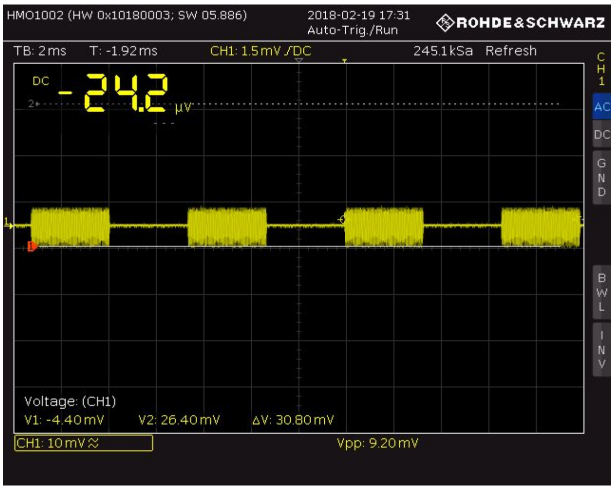


Figure 16*: Oscilloscope screenshot showing radio waveform detected by antenna*

Distance/cm	0	5	10	12	15
Peak-to-peak voltage/ mV	200	80	20	5	0

Table 7*: Distance between antenna and drum vs voltage detected without amplification*

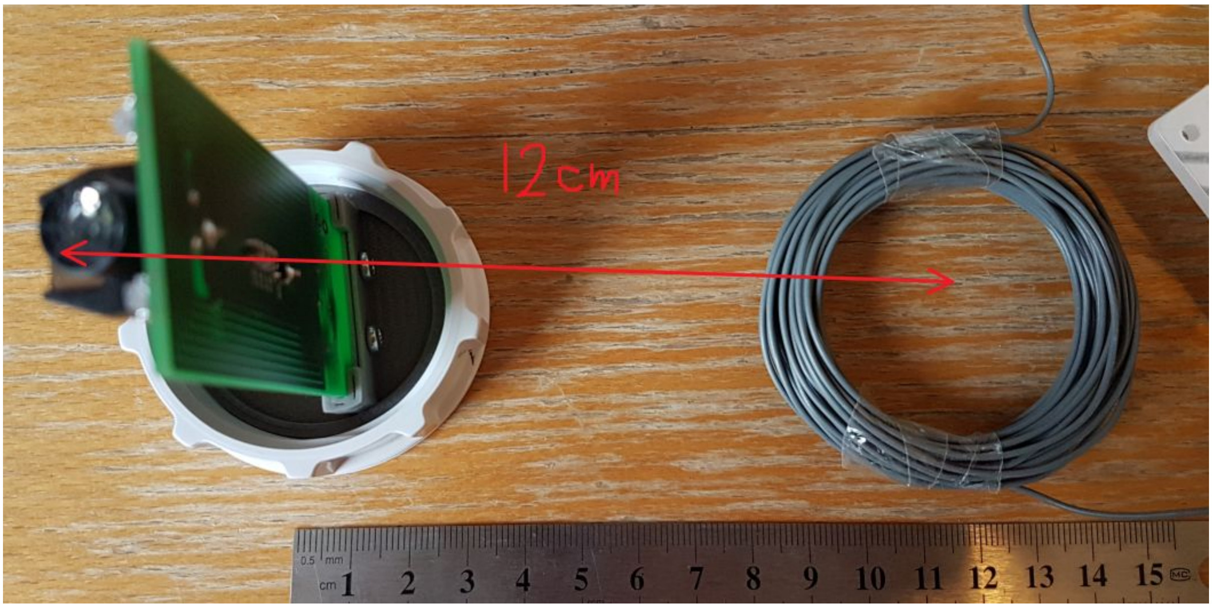


Figure 17*: How distance is measured*

The relative orientation between the drum and the antenna affects the signal strength (figure 18). However, this is not a problem with sufficient amplification.

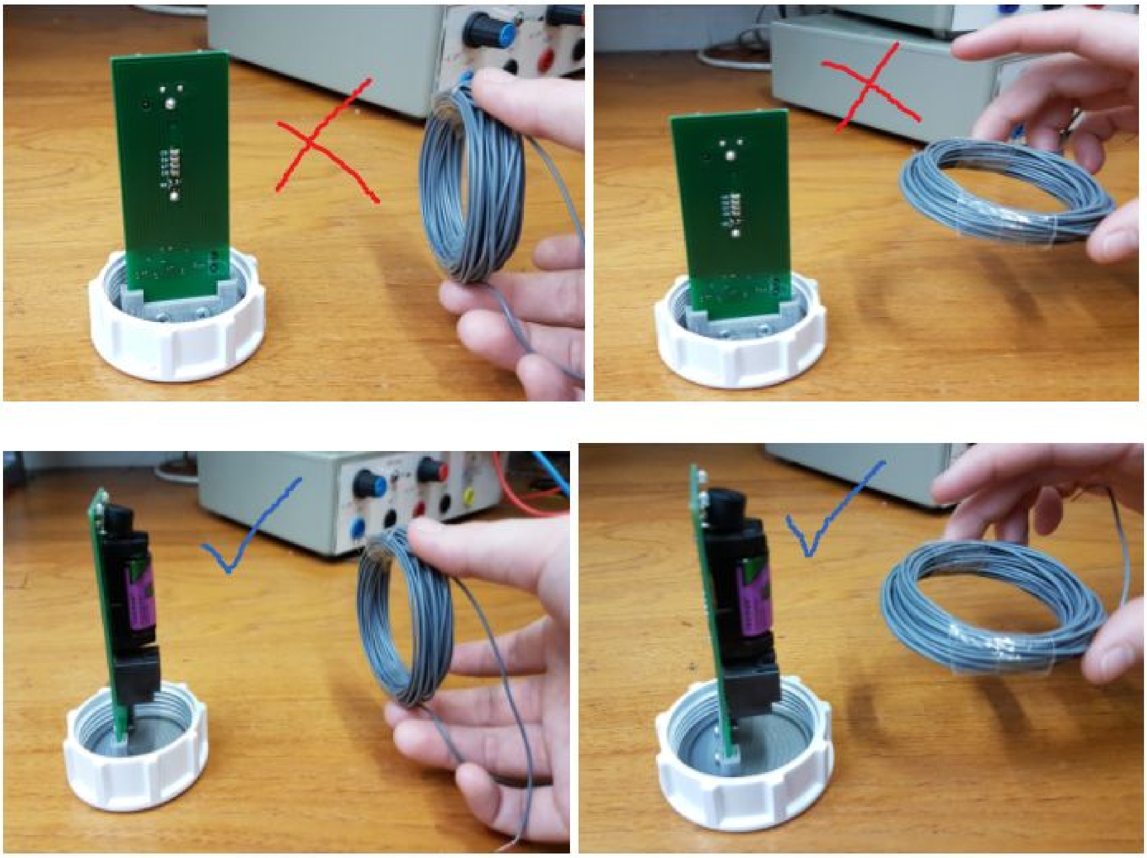


Figure 18*: Relative orientations between drum and antenna. Top - weak signal strength; Bottom - strong signal strength*

4.1.1.1.3 Radio wave demodulation

The same demodulation circuit used for the different radio frequencies comprises a diode, a capacitor and a resistor. (Figure 19)

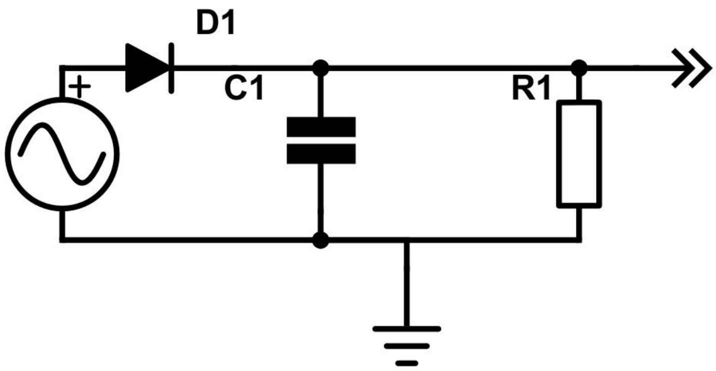


Figure 19*: Demodulation circuit*

The diode rectifies the modulated signal, and the charging and discharging behaviour of the RC network, as the AC signal varies, smooths the modulated signal, thus performing envelope detection (figure 20). In order for envelope detection to work well, the corner frequency of the RC network has to be much smaller than the smallest carrier frequency (67kHz) but larger than the largest modulating frequency (239Hz).

$$2\Pi f_m < \frac{1}{RC} << 2\Pi f_c$$

Accordingly, we used C = 1 μF and R = 620Ω, which gives a suitable corner frequency of 257Hz > 239Hz.

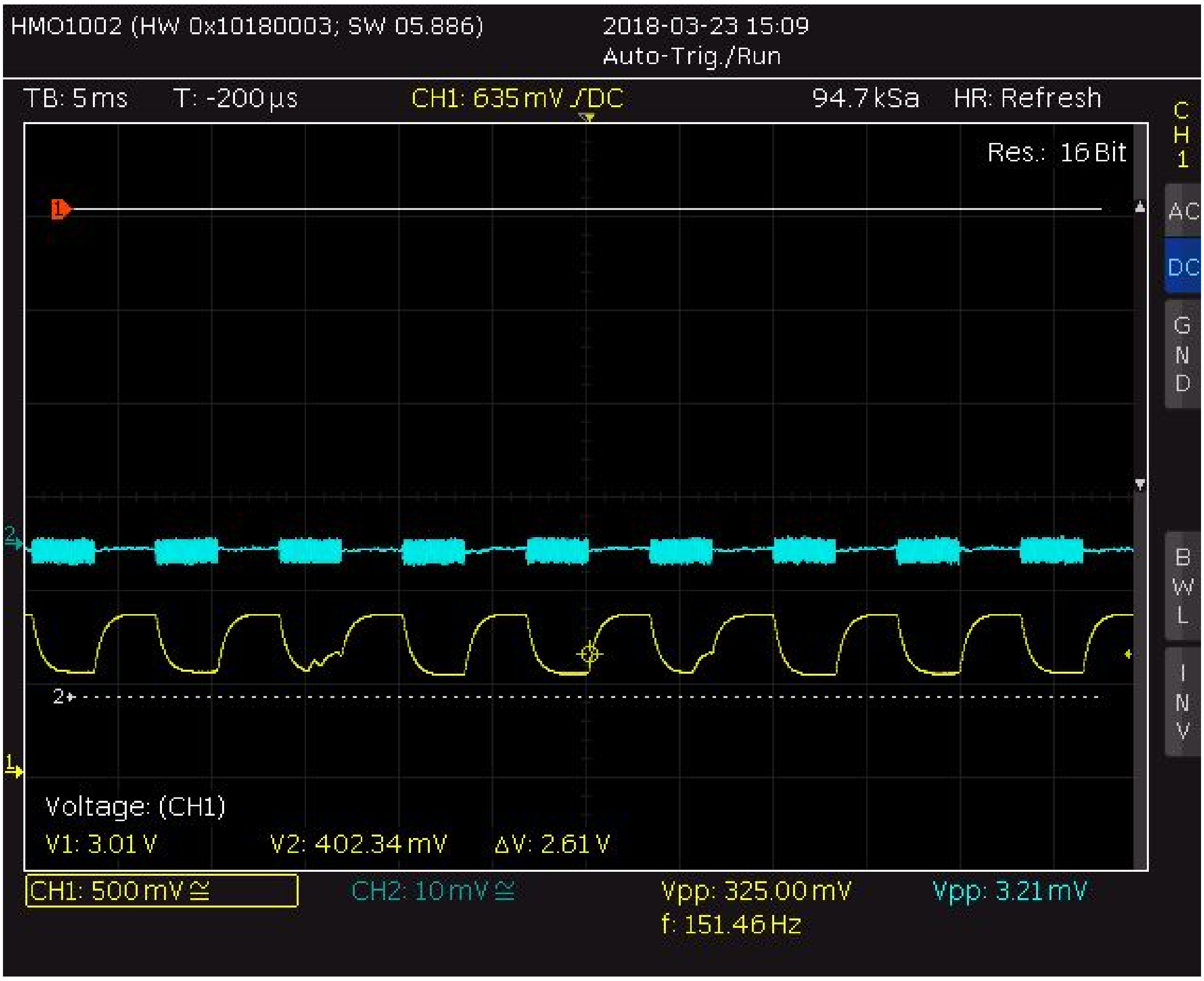


Figure 20*: Blue- original signal. Yellow- demodulated signal*

4.1.1.2 IR

There is a wide variety of IR sensors available. As a result, we had to consider which option - IR phototransistor, photodiode, light dependent resistor and IR receiver - is the most suitable. We compared them in table 8.

	IR [2] phototransistor	Photodiode	LDR	IR Receiver
Working principle	Transistor becomes active in presence of IR	Diode leakage current proportional to IR	Resistance proportional to light and IR	Contains a phototransistor
Sensitivity	Good	Bad	Bad	Good, peak sensitivity at 950nm
Speed	Relatively fast	Fast	Slow	Fast

Table 8*: Comparison between IR sensors*

Since the emitted IR signal is known to have a wavelength of 940nm, the IR Receiver is clearly the most suitable. Hence, we decided to use the IR Receiver TSOP 38238.

4.1.1.2.1 IR sensor performance

The IR Receiver can detect a weak signal (figure 21). However, the drum has to be uncovered and almost pressing onto to the IR in order for anything to be detected. Even with amplification, nothing can be detected once the drum is covered.

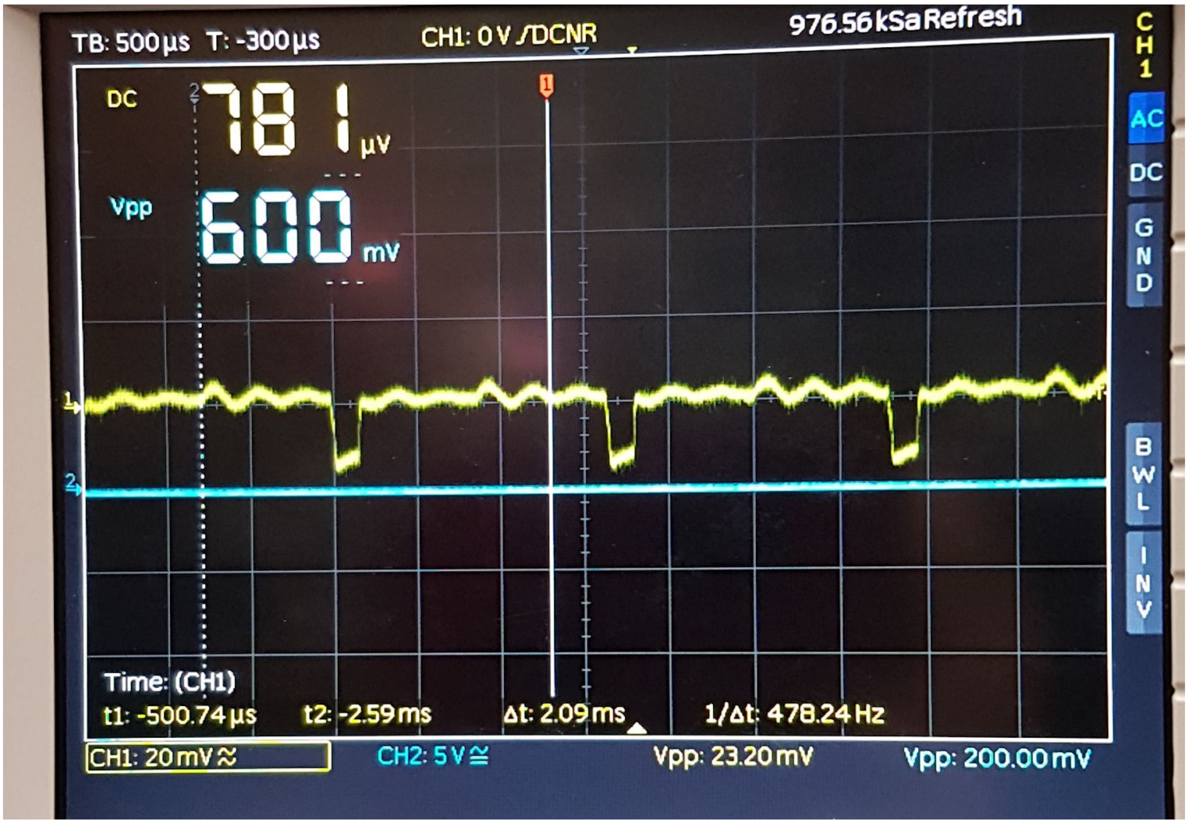


Figure 21*: Waveform of IR signal 607Hz*

[2] (Circuit Globe, n.d)

4.1.1.3 Filtering signals

Both the radio and IR signals require filtering to differentiate different frequencies. We tried out two types of filters: passive LC filters and active filters.

4.1.1.3.1 LC Filter:

The radio antenna simultaneously acts as the input signal source and an inductor. It forms a resonant network when placed in parallel with a capacitor (figure 22).

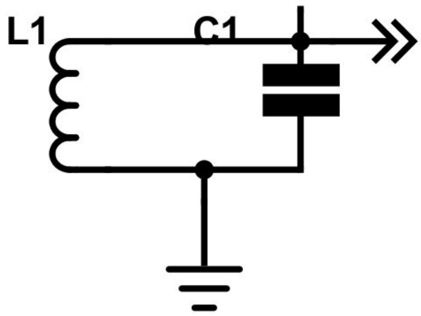


Figure 22*: LC Filter circuit*

The inductance is fixed, so the capacitor has to be chosen such that the tuning circuit's resonant frequency, $f_r = \frac{1}{2\Pi\sqrt{LC}}$, equals the carrier frequency - 67kHz or 103kHz. Signals other than the resonant frequencies will be attenuated, thus differentiating between the two carrier frequencies. Chosen capacitor values are shown in table 9.

Resonant Frequency	**67kHz**	**103kHz**
Capacitor Value	68nF	28nF

Table 9*: Capacitor values for each carrier frequency*

4.1.1.3.1.1 Performance of LC Filter:

The filter is not able to completely suppress the unwanted signal completely but there is a significant difference between the 67kHz and 103kHz. When the 68nF capacitor is applied, the antenna can detect the 67kHz carrier 20cm away but can only detect the 28nF from 12 cm away.

In order to get a sharper resonance curve, we also tried using active filters.

4.1.1.3.2 Active filters

We used the analog Wizard[3] to generate an active filter for the 421Hz IR frequency (figure 23).

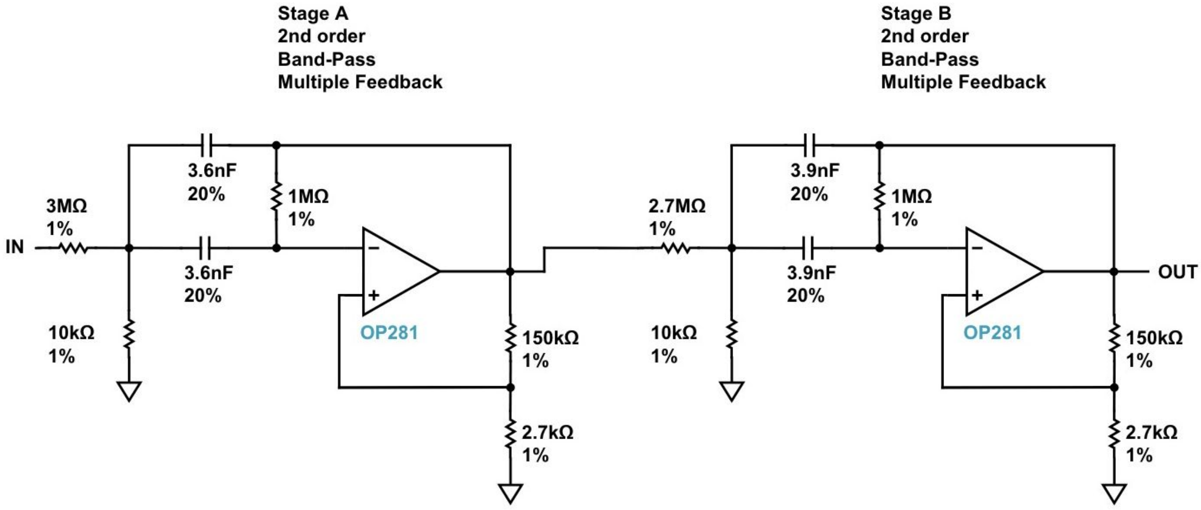


Figure 23: *Active filter circuit for 421Hz*

Implementing this circuit (figure 24), we obtained a output-frequency graph (figure 25).

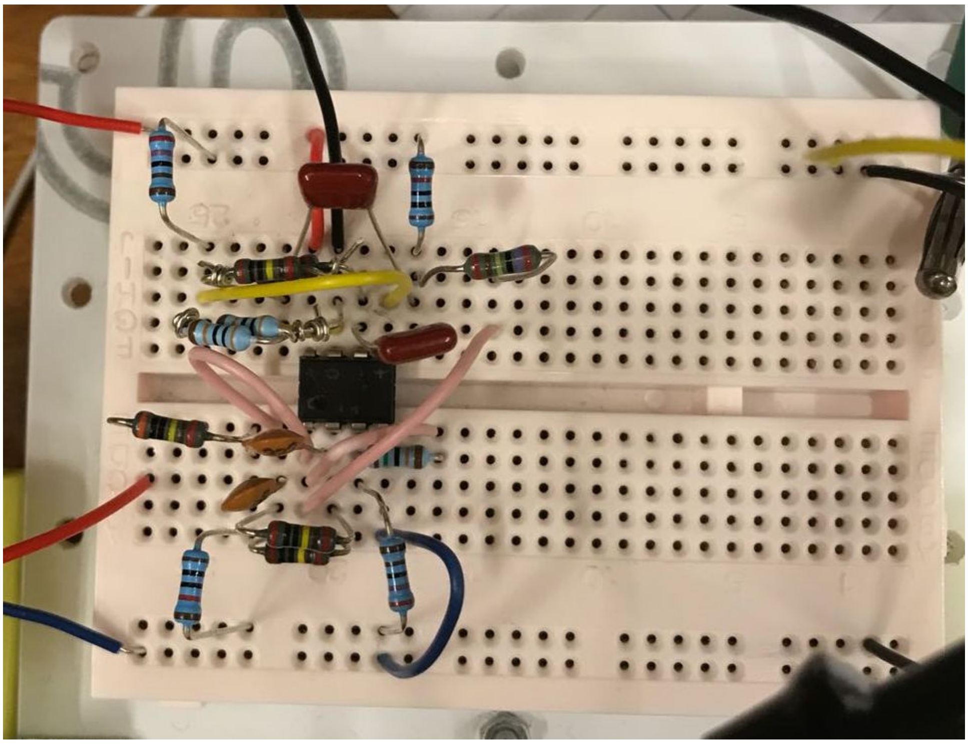


Figure 24: *Active filter implemented*

[3] http://www.analog.com/designtools/en/filterwizard/

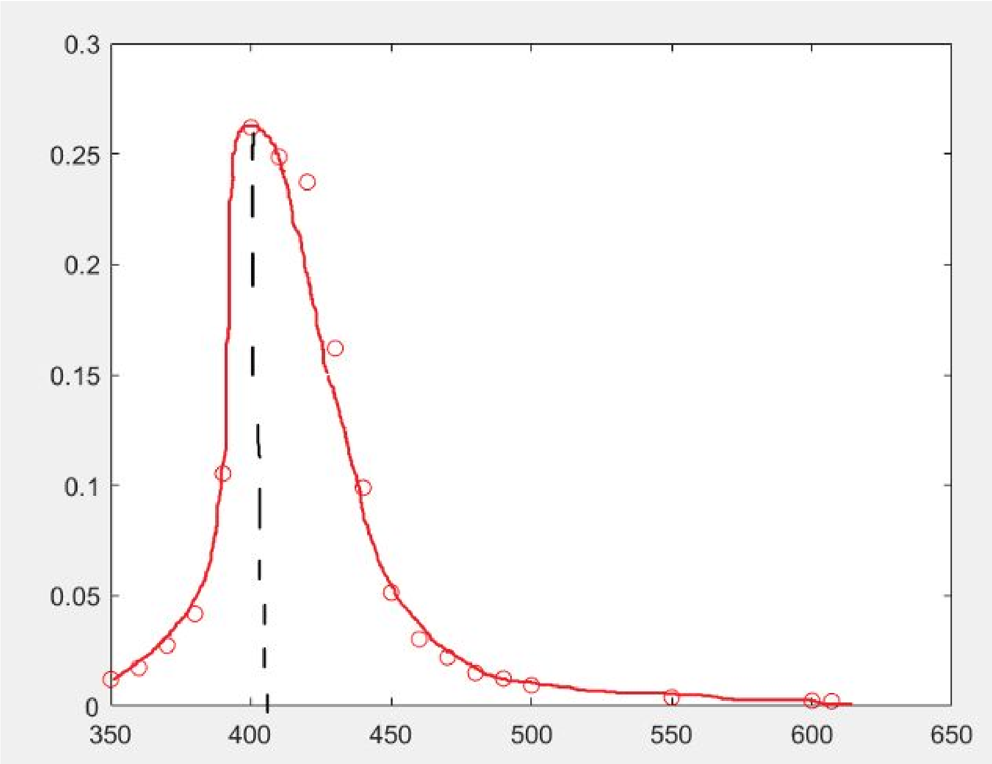


Figure 25*: Output-frequency curve*

4.1.1.3.2.1 Performance of active filter

Although the resonance peak is sharp which is ideal, the active filter takes up a significant amount of space.

In order to maximise space, we considered having an arduino controlled switch to switch between different components so that certain parts can be reused.

4.1.1.4 Arduino controlled switch to switch between different filters

One example where the switch can be used is to switch between the two tuning capacitors for receiving different radio signals (figure 26).

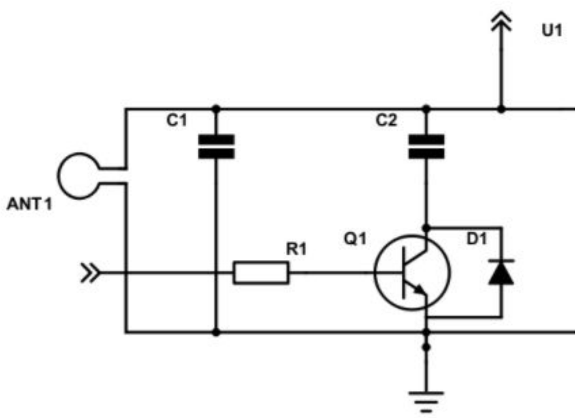


Figure 26*: Switch in the context of choosing tuning capacitors*

The BJT used is a BC337 NPN transistor, chosen because it is cheap (£0.15), easily available and good enough as a switch. The base is connected to the arduino digital input pin through a resistor[4], the emitter is grounded, and the collector is connected to the 40nF capacitor. When the arduino input pin is HIGH, base current is present causing the transistor to enter saturation mode and turn on. This connects the 40nF capacitor to ground and the total tuning capacitance is now 68nF, thus favouring the 67kHz signal. When the arduino pin is LOW, the 40nF capacitor is disconnected and the tuning capacitance is only 28nF which favours the 103kHz signal.

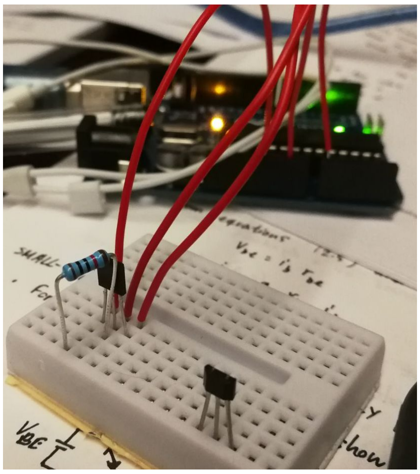
A diode is placed across the emitter and collector to prevent damage to the transistor.

4.1.2 #2: Radio + IR + Magnetic + Sound

We realised that just using two sensors in #1 meant that we will rely heavily on the filters for both. However, the breadboard space is limited. So instead, we decided to use the other two: magnetic and sound. Magnetic to distinguish Durranium from Brookesite, and Garborite from Nucinkisite; Sound to differentiate Cheungtium and Yeatmanine.

4.1.2.1 Magnetic

There is a number of magnetic sensors available. After eliminating the unipolar sensors, we chose a switch hall effect sensor 2SS52 which is omnipolar and thus more convenient to use as we do not know the magnetic poles of the drum magnet. This type of sensor produces a voltage when placed in a magnetic field. The one chosen is an active low device, which means it remains at 5V, and when it detects a magnetic field, the voltage will drop to 0V. Since this is not intuitive, we put a pull up resistor (figure 27) in the output of sensor to turn it into the more familiar active high mode.



[4] Resistor value was experimentally chosen to be 4.3kΩ. It is present to set the base current required to send the transistor into saturation mode.

Figure 27*: Switch Hall Effect sensor with pull-up resistor*

4.1.2.1.1 Performance of hall effect sensor

After, some testing with the magnet, we found out that it only detects a signal if the magnet is at a maximum distance of 5cm approximately, and it must be placed right on top of the sensor, to be able to detect it. Furthermore, if the magnet is place inside the drum, no signal will be detected. This means that the hall effect sensor is not a viable method of detecting the material and will eliminate the possibility of this combination of sensor being used.

4.1.2.2 Sound sensor

We had difficulty finding a suitable ultrasonic sensor because most sound receivers available [5] are too sophisticated and typically require large supply voltage. Eventually, we altered our search terms to search for ultrasonic ranging modules instead of just a receiver and ultimately chose the HC-SR504 ultrasonic ranging module and simply removed the receiver from it (figure 28). It is meant to work at 40kHz which fits our acoustic signal excellently.

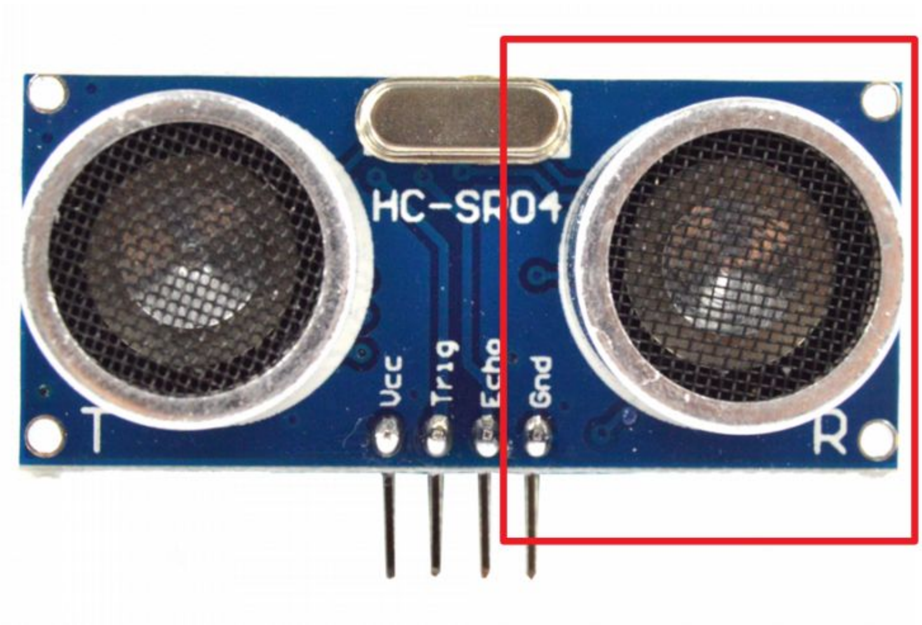


Figure 28*: HC-SR504 ultrasonic ranging sensor with receiver boxed up*

4.1.2.2.1 Performance of sound sensor

Without amplification, it detects a few mV a few cm from the drum, but sensitivity drops sharply at different orientations and larger distances, like the radio antenna. This is easily solved with amplification.

[5] https://uk.rs-online.com/web/p/ultrasonic-proximity-sensors/2370783/

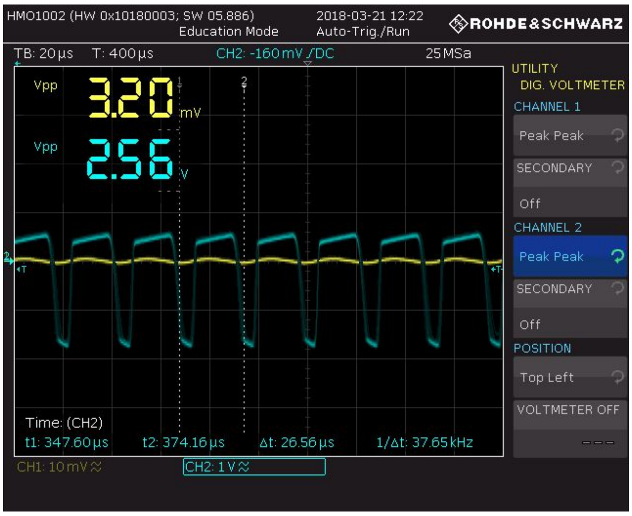


Figure 29*: Waveform of 40kHz acoustic signals before (yellow) and after amplification (blue)*

4.1.3 #3: Antenna + Sound

Out of all four sensors, only the radio antenna and the ultrasonic sensor work reasonably well, as the IR receiver and switch Hall Effect sensor fail completely once the drum is covered. Therefore, we can only identify the material through the radio and acoustic signals.

This time, we tried a different approach from #1 and #2. Instead of using filters heavily, we depended on the arduino code to measure the frequencies of the signals directly. Therefore, in our circuits, we only needed to ensure that the signal is amplified enough (>3V) to be detected by the arduino digital pin as high, and also transformed into a square waveform to improve the accuracy of the arduino code. This can be achieved using a Schmitt Trigger which amplifies and produces a square waveform simultaneously.

4.1.3.1 Schmitt Trigger and amplifier

The Schmitt trigger-amplifier (figure 30) consists of two identical cascading common-emitter BJT amplifiers.

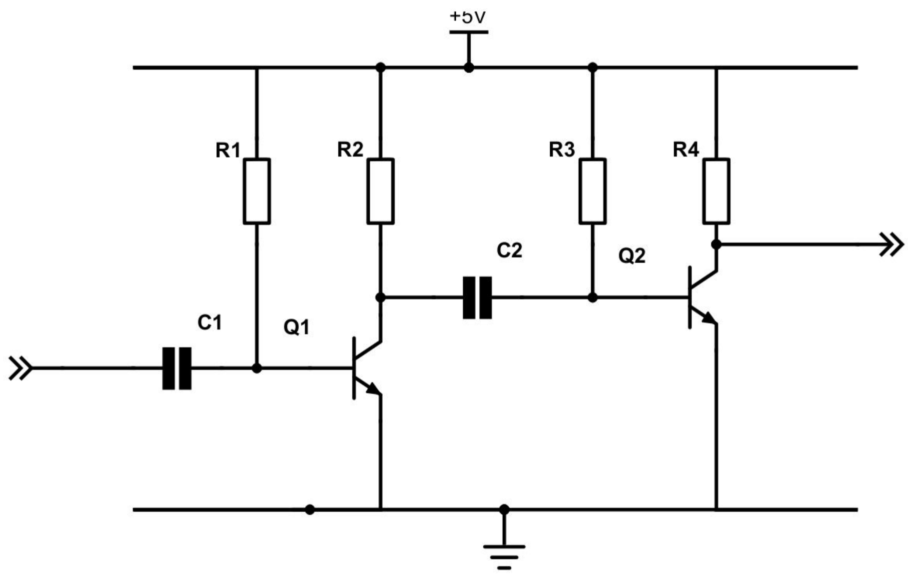


Figure 30*: Two cascaded BJT common-emitter amplifiers*

Circuit components used: BC337 NPN transistor, $R_B = 4.3M\Omega$, $R_C = 5.1k\Omega$, $C = 1\mu F$.

1 μF capacitors are placed at the base inputs to isolate the input bias voltages and yet still allow AC signals to pass through.

4.1.3.1.1 As an amplifier:

The BJT common-emitter amplifier is an inverting amplifier. As the input voltage increases from zero, the transistor turns on, leading to large collector current and consequently low output voltage. The gain magnitude of one common-emitter transistor, $A_v = g_m(R_C//r_O)$, is very high because of the large r_O value. Technically, to maximise output voltage swing, we should choose an R_B value that will set the bias collector voltage at 2.5V (midpoint of the power rails). However, as different discrete BJTs have different β values, the required R_B would be different as well, which makes things messy. Furthermore, with the arbitrary resistor values chosen, the signal is large enough to be detected by the arduino, hence we did not go to the trouble of calculating the most optimum resistor values. In fact, at 100Hz, the gain magnitude is 1300, and despite gain attenuation at high frequency, the gain magnitude for a 50kHz signal is 600 - still high enough.

4.1.3.1.2 As a Schmitt Trigger:

The two cascading BJTs also act as Schmitt Triggers. Since the common-emitter amplifier is inverting, two cascaded common-emitter amplifiers will lead to a phase shift of 360° leading to positive feedback. This produces a square waveform (figure 31), which makes it easy for the arduino to measure the signal frequency.

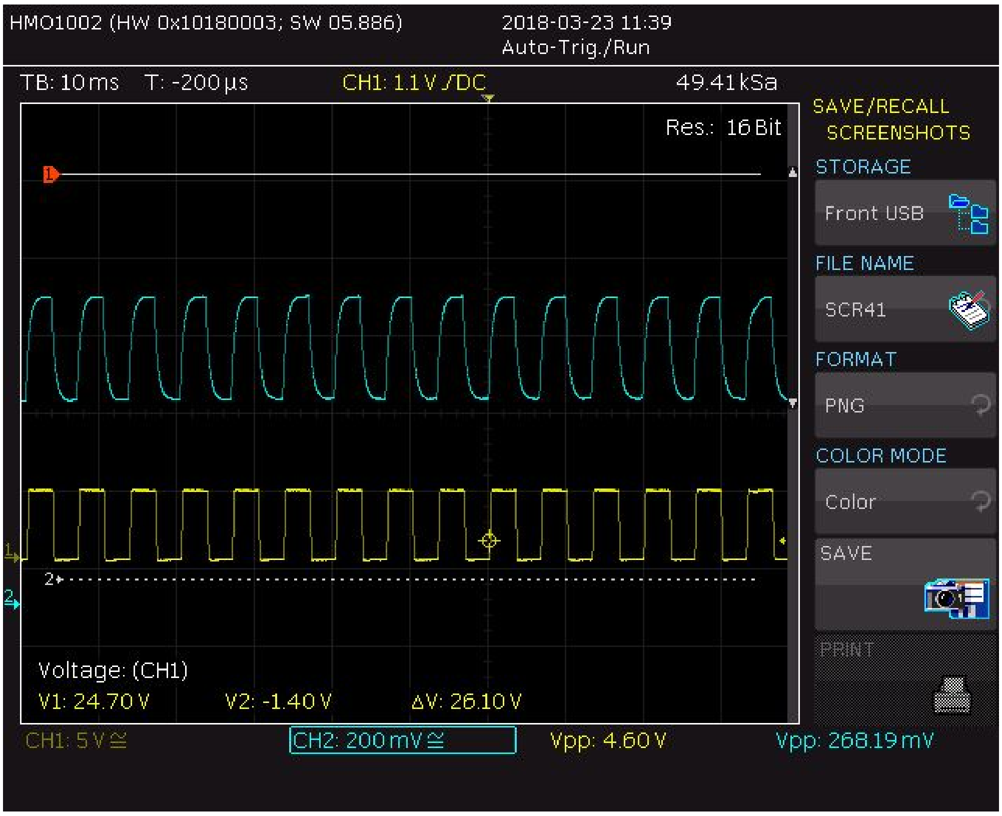


Figure 31*: Signal with (yellow) and without (blue) Schmitt trigger*

4.2 Communication and Remote Control Interface

For the remote control interface, we considered using a joystick but ultimately decided to use an android phone app as it is able to send and display information clearly. We considered different interface designs, and also two options for communication: Wifi or bluetooth.

4.2.1 Interface Design

4.2.1.1 GUI with signal graphs displayed

Our initial plan was to display signal readings in the form of graphs on the user interface, together with the material identity so that the user can visualise the signal waveform.

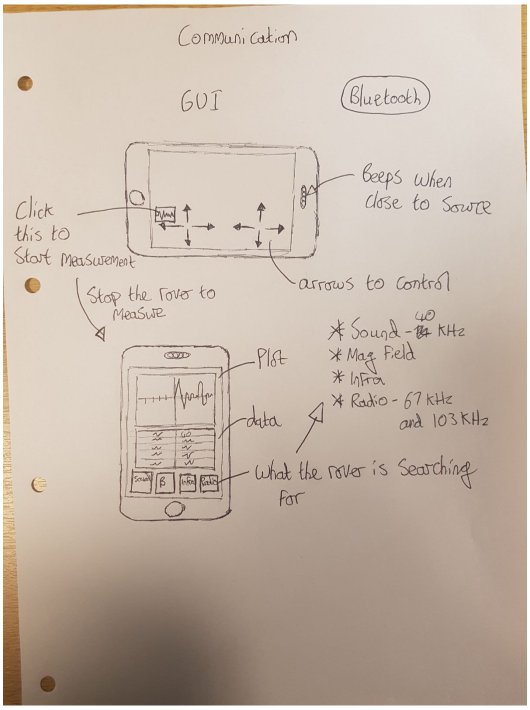


Figure 32*: GUI with graphs displayed*

However, we realised this is not feasible because plotting a graph would require too many data samples, hence this would cause serious time lag. Hence, we removed the plot and simplified the GUI as we only need to know if the signal is there or not. Measuring the signal strength would be hard and effectively pointless.

4.2.1.2 GUI during testing

During testing, we wanted to observe how far the arduino's frequency reading would deviate from the actual frequency. Hence, we created a temporary interface (figure 33) by creating an app with the *MIT app inventor* that is linked to the arduino via Wi-Fi.

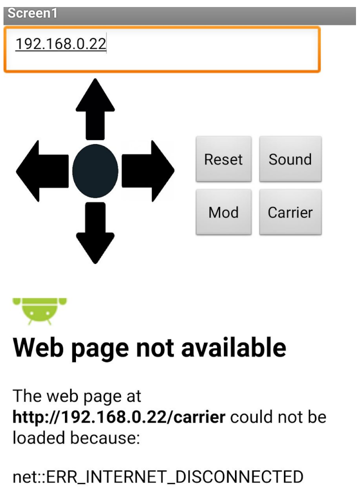


Figure 33*: Temporary remote control interface*

We can move the rover towards the drums by pressing on the arrows (it will move while the button is pressed and stop when no arrow is pressed). Once the rover is near enough, we just press either the “Sound”, “Mod”, or “Carrier” buttons to see the frequency reading for the respective signal type. We then refer to a table mapping each material to its frequencies in order to determine the material identity.

To make this app, we used the *MIT app inventor* which involves a programming language called scratch. This language is very intuitive and easy to use. It would not have been time efficient to build the GUI from the ground up considering platforms like the MIT app inventor come with tools that are tried and reliable. Note that the app only operates on android phones.

The code in figure 34 shows how the app gives instructions to the rover.

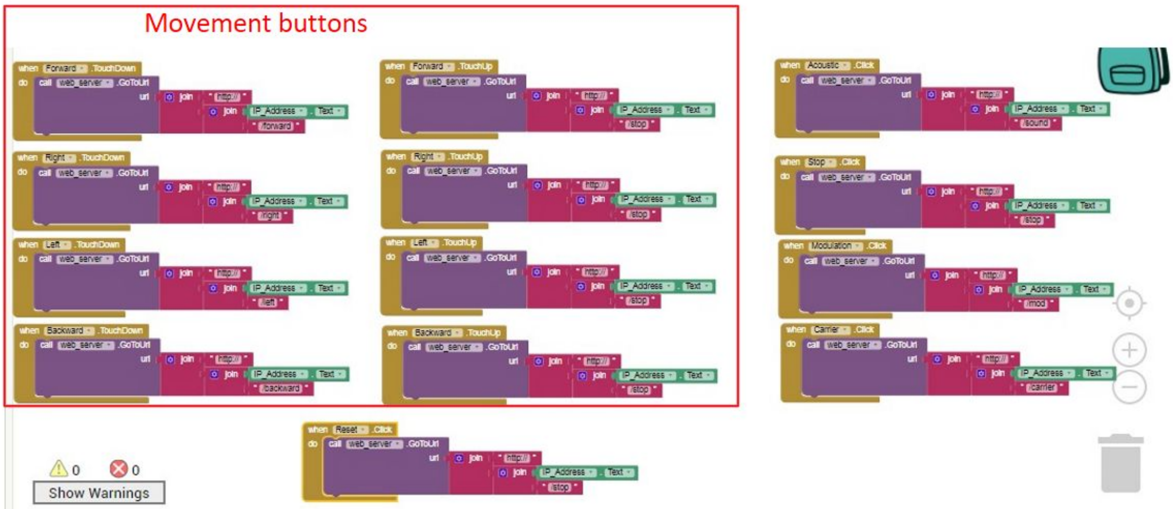


Figure 34*: Code for app in scratch for Wi-Fi*

Each block of code maps to a button on the interface. The two sets of four blocks on the left correspond to the arrow buttons for movement. Each movement direction requires two blocks to ensure that when the button is not pressed, the rover stops. The remaining five blocks correspond to “Stop”, “Reset”, “Sound”, “Mod”, “Carrier”.

4.2.1.3 Finalised GUI

In the finalised interface, the arduino determines the material directly, thus making it easier for the user (figure 35).

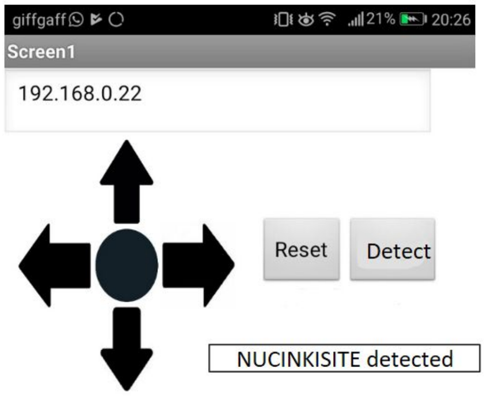


Figure 35*: Finalised GUI*

The user presses “Detect” when it is near enough to a drum, and presses “Reset” before going to the next drum. We decided to put in a “Reset” button instead of having only a “Detect” button so that when the material is detected, the user can see its name appearing on a blank screen, instead of the name changing from the previous material’s name. The more obvious change makes the user feel more secure that the app is working.

Aside from this, we also considered having no buttons to activate detection and simply keeping the detection on at all times. This will update the interface continuously with the new material name. However, the signal processing algorithm is only accurate at short distances, which means that a lot of erratic names will appear on the interface at inappropriate distances which will confuse the user. Hence, we would rather get the user to press two more buttons to achieve much better accuracy and clarity.

4.2.2 Wi-Fi

The Arduino has the Wi-Fi link firmware which allows the Arduino to be controlled via a webpage server. As we can see in figure 33, the app itself has a web server. By pressing a button, the server will go to a specific url. Then the arduino reads the url, and perform the actions associated with it.

We have also uploaded the UNO Wi-Fi firmware to the Arduino, to test which one is better. The UNO Wi-Fi firmware allows us to communicate to the Arduino via a monitor, similar to the serial monitor in the Arduino software. This allows us more flexibility when sending commands to the Arduino.

4.2.2.1 4.2.2.1Performance of Wi-Fi

The Wi-Fi presented some connectivity problems and lag, probably related to the EEERover network. There were times, when we were unable to stop the Arduino from moving, or even control the LED.

Due to the poor Wi-Fi performance, we decided to consider bluetooth, and leave Wi-Fi as a backup plan.

4.2.3 Bluetooth

The bluetooth module we are using is *Bluefruit LE UART.* Figure 36 shows a basic GUI connected to the arduino via bluetooth.

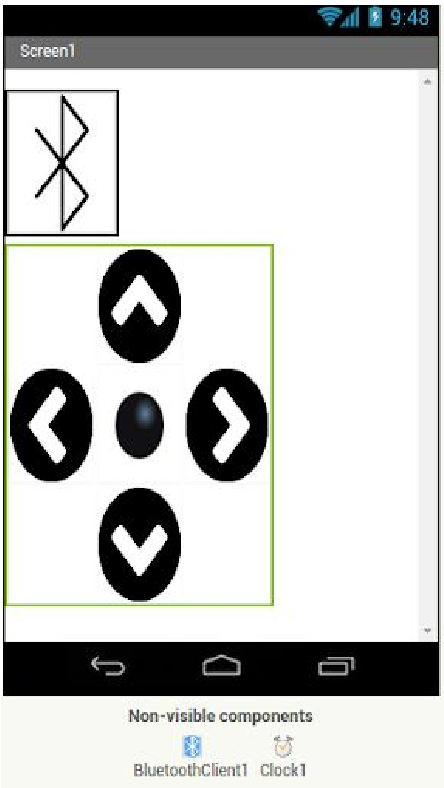


Figure 36*: GUI for bluetooth*

This GUI is similar to the ones we made for Wi-Fi, except that there is now a bluetooth button. It is pressed to connect the arduino bluetooth module to the phone. The arrows dictate the direction, once pressed the rover will move in that direction constantly until the centre button is pressed to make it stop.

The code in figure 37 shows how the app gives instructions to the rover.

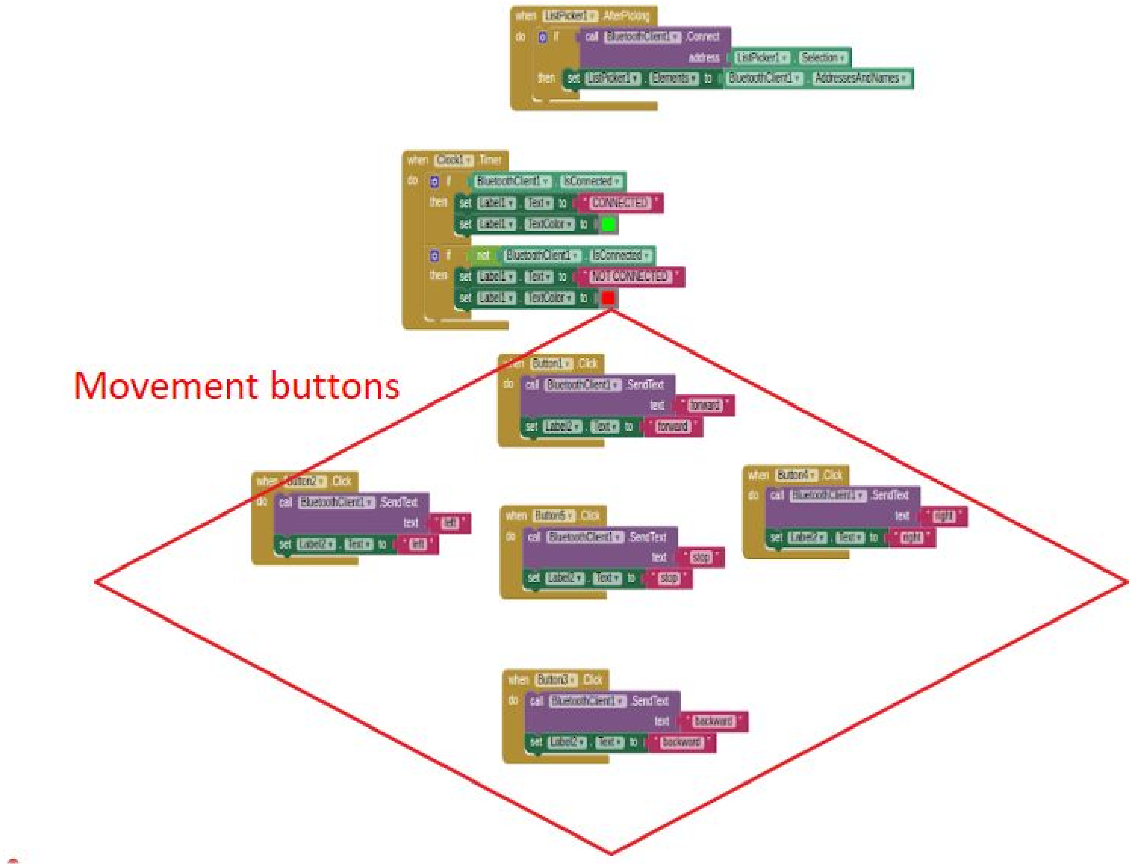


Figure 37*: Code for app in scratch for bluetooth*

The two blocks at the top are used to configure the bluetooth and ensure there is a good connection. In scratch it is as simple as placing a block called "BluetoothClient" down to get the bluetooth to operate. The five blocks below are for the movement of the rover. Each button on the GUI will send a different instruction to the rover. Button1 at the top will send "forward" to the rover which will set the rover in motion until the middle button Button5 is pressed to stop the movement with the command "stop".

4.3 Structural Design

4.3.1 Traction system

The traction system needed to be maneuverable, fast, and cheap. We had five choices in the beginning: swerve drive wheel, tank treads, mecanum wheels, swivel wheel and a normal wheel with treads (figure 38).

Figure 38*: Types of wheels considered - from left to right: swerve drive wheel[6], tank treads[7], mecanum wheels[8], swivel wheel[9] and a normal wheel with treads[10]*

The first three - swerve drive, tank treads and mecanum wheels - were eliminated due to cost, even though they are the best performing ones. We then remained with the swivel wheel and the normal wheel.

From these, we considered the following wheel systems: a 3-wheel system with the third wheel being a swivel wheel, and a 4-wheel system (figure 39). The presence of extra wheels increases stability.

Figure 39: 4-wheel system vs 3-wheel system[11]

The 4-wheel system uses 4 normal wheels, and the 3-wheel system uses 2 normal wheels at the back and a swivel wheel at the front. While a swivel wheel (~£5) is more expensive than the normal wheel (~£0.50), a 3-wheel system would be easier to implement as the swivel wheel does not affect the rear wheels. On the other hand, the front two wheels in a 4-wheel system would add a differential to the rear wheels when turning because the rear

---

[6] Swerve drive: https://www.andymark.com/product-p/am-3009.htm
[7] Tank treads: http://www.ducklearning.com/product/tetrix-max-tank-tread-kit/
[8] Mecanum wheels: https://www.vexrobotics.com/mecanum-wheels.html
[9] Swivel wheel: https://www.toolstation.com/shop/Ironmongery/d170/Wheels+%26+Castors/sd3036/Swivel+Wheel+Castor/p75764
[10] Normal wheel with treads: https://www.reddoko.com/product/bo-motor-wheel-big/s/S4D-273
[11] Prototypes from past year EEE1 Project

wheels will not rotate with the same speed and thus introduce complexity into the arduino code for controlling wheels. Therefore, we are planning to implement the 3-wheel system.

4.3.2 Plan for final structure

After having finalised the sensors and circuits, we are planning to have a protruding surface for the antenna coil and the ultrasonic sensor to rest on. This allows the radio antenna and the ultrasonic sensor to stick slightly out of the rover (figure 40), with its surface parallel to the floor in order to facilitate signal detection.

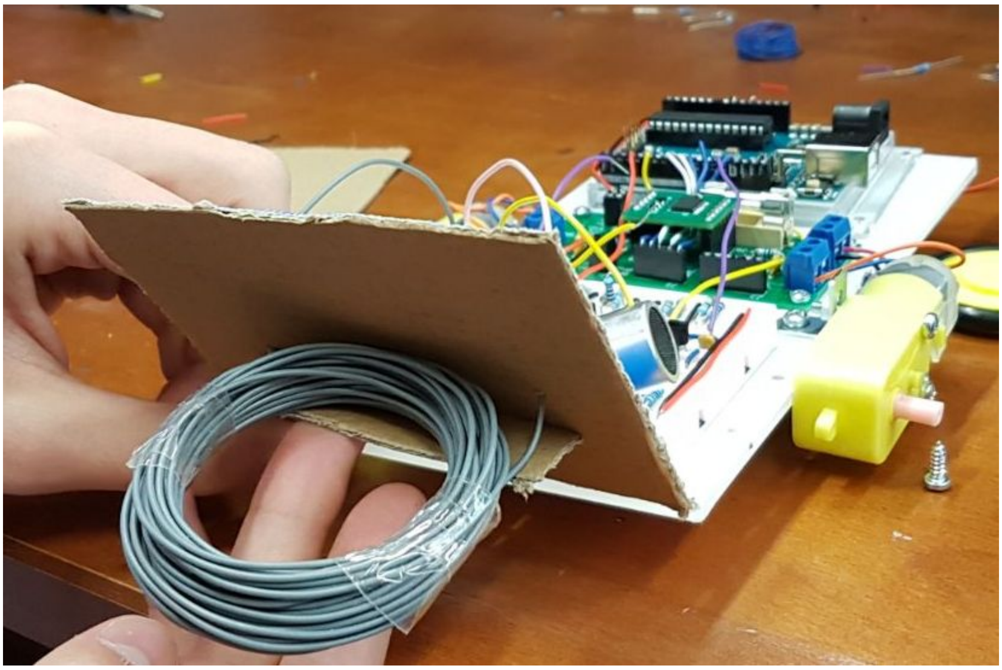


Figure 40*: Position of radio antenna relative to rover*

5 Budget and Development cost

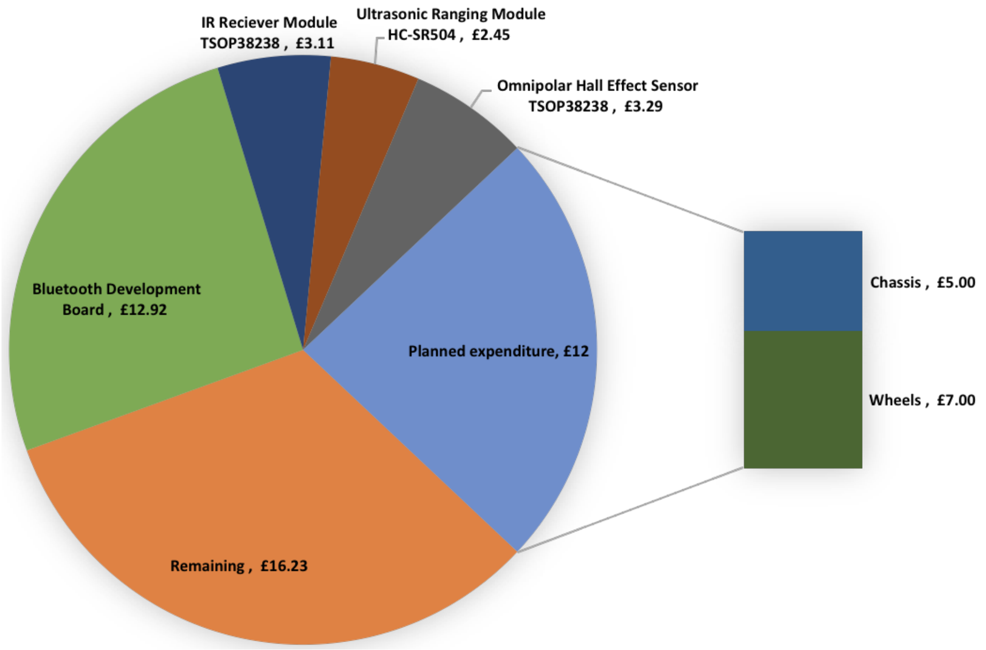


Figure 41*: Pie chart showing how the £50 was spent*

Of the £50 budget that we were allocated, £21.77 of it was spent on sensors to be tested and the bluetooth module to replace WiFi communication. We plan to spend £5.00 on laser cut panels to support the sensors and antenna on the main chassis. £2.00 will be spent on wheels that have already been chosen.

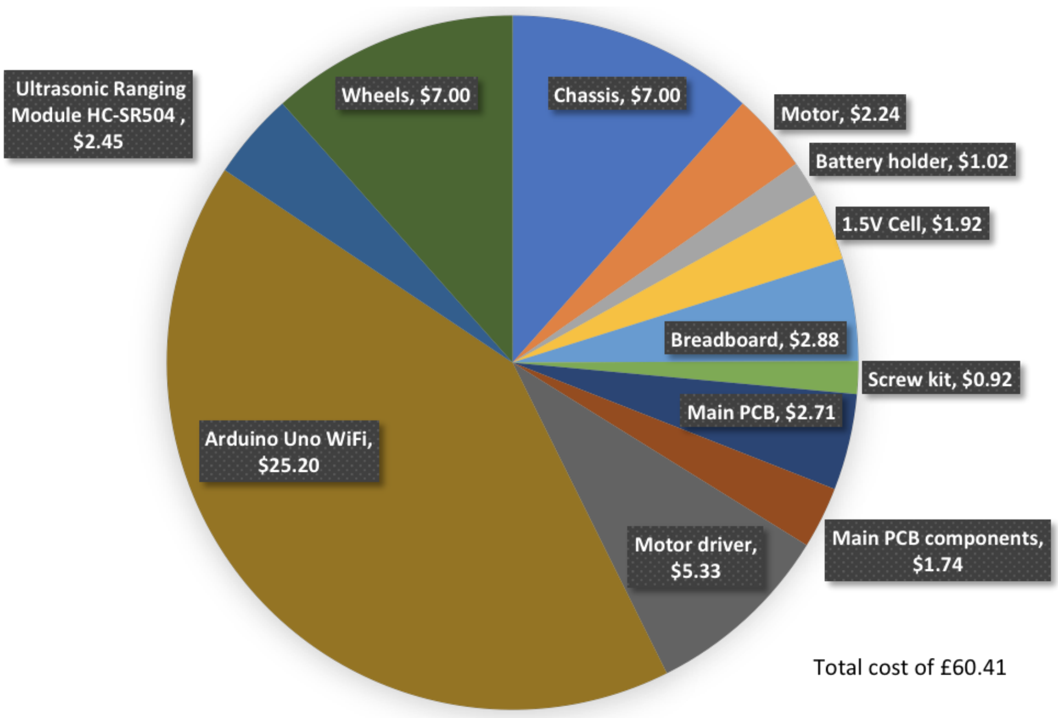


Figure 42*: Pie chart of the current total cost of the Rover (including prices of components supplied with the EEBug)*

6 Project Management since Presentation

After the presentation, we aimed to produce by Easter a basic functional rover that can be moved by a remote control, successfully identify all materials and send the information to the interface. Subsystems involved are Sensors and Circuits, Signal Processing, Communication and Movement.

To achieve this goal, we reduced the frequency of group meetings from the previous once a week to a more flexible schedule. This gives each individual more time to work on his respective subsystem. Instead of meeting as a group, we separated our group into smaller pairs and threes based on the nature of the subsystem. People in charge of sensors, circuits and signal processing collaborated together; communication and movement worked closely together. Under this arrangement, our progress since the presentations can be charted by the timeline below (figure 43).

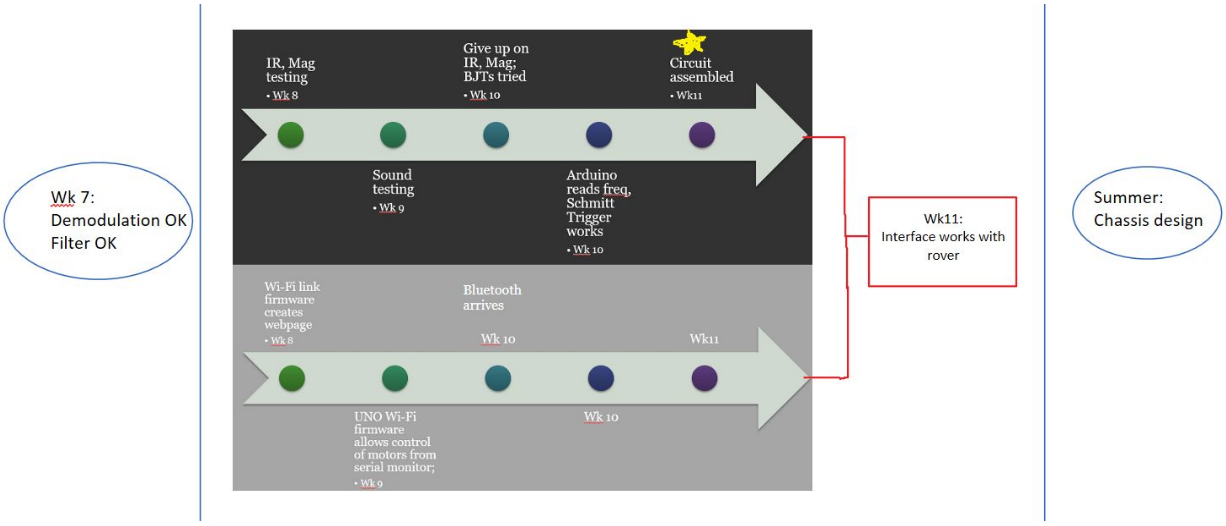


Figure 43*: Timeline since presentation*

The key milestone was having all the circuits assembled in Week 11. Once this was done, the two groups converged to test the interface and the rover together. We met our short-term goal before Easter break, and our next task is to focus on designing a robust chassis and traction system for the rover.

7 Discussion and Conclusion

Table 10 evaluates the current rover prototype against the task requirements.

Criterion	Achieved?	Plans
Accurate in identification	Yes	Do more testing in different conditions to get reliable error thresholds
Cost and weight effective	-	Buy the chosen wheels and implement the traction system that has been chosen
Manoeuvrable	Somewhat - can move, but primitive	Redesign traction system in summer
Robust and reliable construction	No	Redesign chassis, order extra parts to be added to the main chassis to support the sensors and antenna before 27 april
Logical user interface	Yes	Work on stability of bluetooth connection, further testing carried out when the traction is implemented

Table 10*: Evaluation of prototype against task requirements*

In summary, the prototype requires major changes to its structure and traction system, and minor improvements to identification accuracy and the interface.

Additional parts will be added to the structure to support the various sensors that will be used in identifying the material. The traction system will also be implemented once the chosen wheels are ordered and available.

Once the traction system has been implemented, further testing can be done with the bluetooth and wifi implementation to see if any improvements or changes are required.

8 References

Arduino Frequency Counter Library. (n.d.). Retrieved from: http://interface.khm.de/index.php/lab/interfaces-advanced/arduino-frequency-counter-library/

Circuit Globe (n.d). *Difference between Photodiode & Phototransistor.* Retrieved from: https://circuitglobe.com/difference-between-photodiode-and-phototransistor.html

Department of EEE (2017). Exoflask Images. *EEERover Technical Guide.* Page 7. Retrieved from: https://intranet.ee.ic.ac.uk/intranet/labweb/pdf/tech-guide.pdf

Department of EEE (2017). Material types from the design briefs. *EEERover Technical Guide.* Page 9. Retrieved from: https://intranet.ee.ic.ac.uk/intranet/labweb/pdf/tech-guide.pdf

HOPERF (n.d). *Antennas applications for RF module.* Retrieved from: http://www.hoperf.com/upload/rf/antennas_module.pdf

USNA (2013). *EE302 Lesson 13: Antenna fundamentals.* [Presentation]. Retrieved from: https://www.usna.edu/EE/ee434/Handouts/EE302%20Lesson%2013%20Antenna%20Fundamentals.pdf