# E1 project & Engineering Design and Practice

## EERover project report

Submitted to

Dr Zohaib Akhtar

by


## Team Strontium

Daniel Gallego Rubio (CID: 01340223)

Albert Fleta Moya (CID: 01400775)

Constantinos Kourris (CID: 01341033)

Michael Figini (CID: 01350213)

Min Sung Kim (CID: 01361691)


Date of Submission: 24/Mar/2018

Word Count: 8221

Imperial College

# Contents

# 1. Introduction

In this project, a rover, remotely controlled via a mobile app, which can explore a hazardous environment and detect different materials encountered, was designed. As it must operate in an unpredictable and harsh condition, it is designed to manoeuvre on rough, uneven surfaces, go over small obstacles and have high mobility. The rover optimises 4-wheel 4-motor system, which allows the rover to have much better mobility and stability.

Since the orientation of the material is unknown, the rover has two sets of sensors, one built on the chassis and the other attached on the extension arm of the rover. Once the rover detects any material, it will give two types of feedback: visual feedback on the screen of the app, which is also used for controlling the rover, and acoustic feedback from the speaker of the rover, which is the extra flair of the rover.

## 2. High-level design

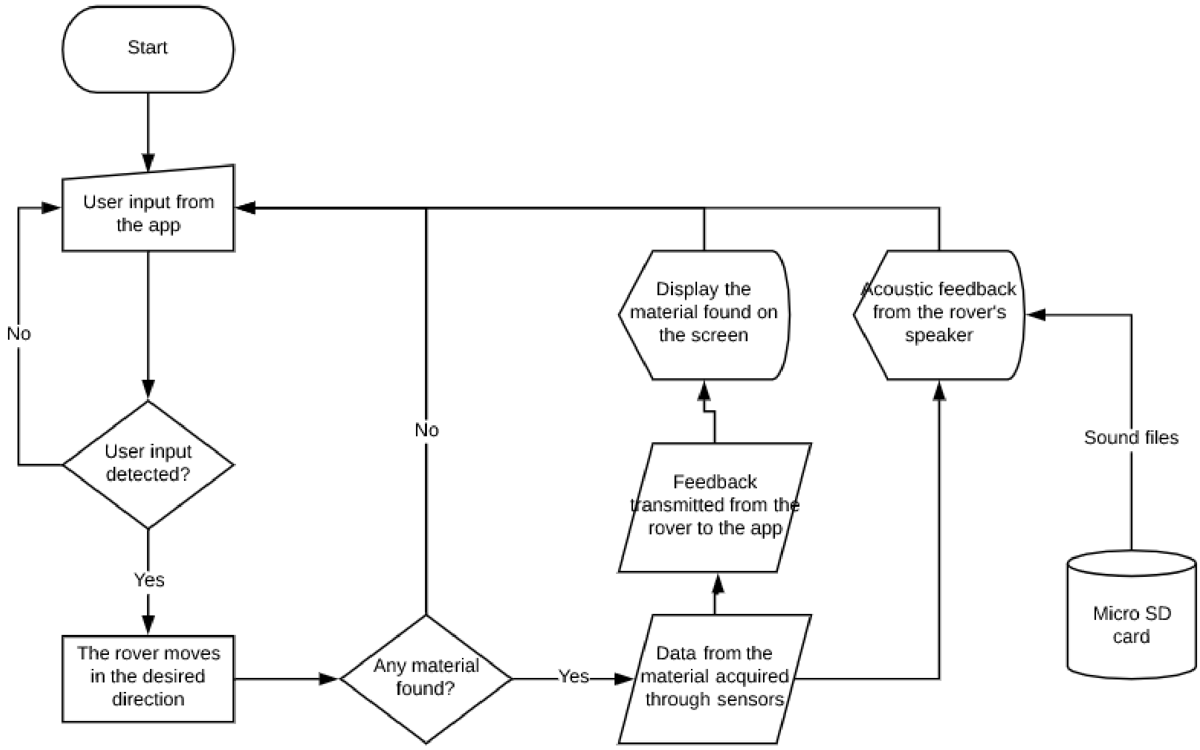



*Figure 1: system flowchart of the rover*

To begin with, it was necessary for the members to have an illustration of what the rover needs to do as it's much more efficient and coherent to define main systems and then work on their subsystems. Thus, a system flowchart of the rover was created as it gives a good illustration of the rover's tasks. In Figure 1, once the system starts, as in the rover is operating and the app is running concurrently, the rover waits for the input from the app. Once the user enters input via the app, the rover receives the input and moves in the desired direction at the speed set by the user. As the rover roams around the hazardous environment, if it finds any material, the information about the material is sent to the app. Once the app successfully receives the data, it displays the information of the material found on the screen while the rover outputs an acoustic feedback through its speaker. The system flowchart worked as a basis of the rover design, and details were added later as the team members worked on their parts.

## 3. Addressing the 32 PDS points

Performance: Four motors on the rover were used, and the app has the gear system; thus, the rover can operate at different speeds. This means that the rover can be both quick and manoeuvrable.

Environment: The wheels and steering system of the rover means that it can go around obstacles and has enough grip to deal with lower friction surfaces.

Life in service: The rover works for however long it's required to. There was no report from testing cases that the rover suddenly stopped working (After getting rid of the WIFI).

Maintenance: Regular maintenance on the rover was unnecessary as when a new component to the rover, the rover continued to work as expected.

Target production cost: Manufacturing the rover required less than £50. It is planned to exploit our budget as much as possible.

Competition: The rover aims to be the best rover in the class, possibly the best rover any 1st year class has seen at this university.

Shipping: The rover does not need to be shipped anywhere as it is made and tested in the EEE labs.

Packing: The rover was stored in one of the boxes from EEBueg experiment.

Quantity: 1 rover.

Manufacturing facilities: EEE lab and personal computers were used for manufacturing the rover and coding respectively.

Size: Chassis from EEBug experiment and two additional motors were used.

Weight: About 1Kg. Not the lightest, but light enough

Aesthetics: The rover has a speaker to output sounds when it finds certain materials, and during the summer term the rover might be upgraded with additional decoration such as LED before the manufacturing process is over.

Materials: The chassis, wheels and motors from EEBug experiment were used. Wires from EEE lab were also used.

Product life span: It should, at least, survive until the end of the summer term.

Standards and specifications: The rover meets the standards and specifications set for this project.

Ergonomics: The user uses a phone app to control where the rover moves, and the rover sends information back to the app to be displayed to the user.

Customer: The customer is the staff of the EEE department and it is assumed that their technical knowledge is advanced. It is also assumed that they will not what to fix the rover themselves; so, even though they have the knowledge, they will require the rover to be simple to use and have no need for them to fix.

Quality and reliability: The rover has been reliable so far and has performed well under testing, testing will be continued until the demonstration to make sure something doesn't go wrong.

Shelf life: The rover can last for long periods of time without being used and still works well after that time.

Processes: Wiring was used to connect all the sensor circuits together and used soldering to connect devices to the PCB.

Time-scale: The rover is on target to be completed before the demonstration in the summer term, this has been achieved by following the Gantt charts.

Testing: At each stage of the manufacturing process, individual parts of the rover were tested to check whether they operated properly. This has helped the members pick up and fix issues before the circuits became very complex.

Safety: All the testing and manufacturing has been conducted under the correct safety procedure, and it's ensured that the rover is safe to use.

Company constraints: There are several constraints in this project, such as £50 budget limit.

Market constraints: Not applicable as the rover won't be sold.

Patents, literature and product data: Patents don't have any impact in this project as the rover will not sold. However, when conducting research and writing a technical report, data or literature written by others may be used. Thus, careful referencing is required.

Political and social implications: Not applicable as the rover won't be sold.

Legal: Not applicable as the rover won’t be sold.

Installation: Installation of any kind isn’t required when the customer uses it.

Documentation: The secretary has taken minutes of every meeting the group has had and have also documented the manufacturing process of the rover.

Disposal: The entire rover will not be disposed but individual components may need to be disposed, and these will be dealt correctly.

# 4. Design Criteria

To add details to each subsystem of the rover, it was important to create design criteria in the beginning because not only does it define the standard of the final product, but also it makes the tasks brief and clear. By creating design criteria, the members could stay on track in terms of their progress as the criteria worked like a checklist of what the rover should be. Figure 2 shows the design criteria of each part of the rover.

I. Mobility/Operation
   a. Able to operate on a rough surface
   b. Able to smoothly turn
   c. Able to go over small obstacles
   d. High speed
   e. Able to go backwards
   f. Good stability

II. Sensors
   a. Able to detect all signals – ultrasound, magnetic, radio and infrared
   b. Good range of detection for each sensor
   c. Radio and infrared sensors should be able to detect two frequencies.

III. Power
   a. The batteries must provide enough power for the rover to operate.

IV. Application/Intelligence
   a. User can move the rover using the app.
   b. User can control the speed of the rover using the app.
   c. The app displays information of the material detected, say, carrier frequency, pulse frequency, etc.
   d. The intelligence unit should be able to recognise materials detected from the sensors.

V. Additional flair
   a. Sound system reproduces the name of materials when they are found.

*Figure 2: design criteria of the rover*

# 5. Concept generation processes

## 5.1. Concept generation

Now that the design criteria are defined, the next step is to generate solutions to meet each criterion. At this stage, two major methods were exploited: brainstorming and research. The first brainstorming session was done in one of the Engineering Design & Practice workshop sessions. In the session, various ideas regarding the high-level design of the rover were mentioned. Figure 3 shows the list of ideas regarding the structure of the rover from the session (Refer to **appendix I** for the pictures of the brainstorming):

- Spring suspension
- Centre of mass
- Wheels
  - Size
  - Number
  - Type
    - Spiked wheels
    - Continuous track
    - Omni-wheels
    - Rubber wheels
- Power
  - Speed control
  - Moving forward/backward
  - Steering system
  - Number of motors

*Figure 3: Ideas regarding the structure of the rover*

Some of the ideas have possible solutions while some don't. For example, the types of wheels that could be used are spiked wheels, continuous track and wheels with tyres. For the ones that have possible solutions already, concept selection, a stage where members evaluate the effectiveness of each solution and select which one to continue with, could be conducted. The complete list of possible solutions is shown in Figure 4.

Number of wheels

- 2 wheels
- 4 wheels

Size of wheels

- EEBug wheels
- Bigger wheels

Type of wheels

- Normal tyres
- Spiked wheels
- Continuous track

Steering system

- One wheel turns while the other stays motionless
- Two wheels simultaneously turn in the opposite direction

*Figure 4: Complete list of possible solutions to the structure*

Obviously, this is only a rough picture of what the rover could be; furthermore, it only covers the structure of the rover. More detailed and sophisticated solutions require research. This is where the second method, research, is exploited. One of the requirements of the rover is to detect various signals – ultrasound, magnetic, radio and infrared. Each sensor was allocated to each of the members, and they were asked to conduct research about what components and circuits are required to create a sensor for the specific signal they were assigned with. Then, in the next meeting, discussion and evaluation of possible solutions to each sensor were done to eliminate the ones that are unrealistic or less effective than others; this corresponds to the concept selection. Also, the meetings were done weekly to keep track of the development processes, encourage brainstorming between us and discuss any issue. For meetings that were important in the development process, recordings were taken. Records can be found in **appendix II**.

## 5.2. Concept selection

| | Possible solutions | | |
|---|---|---|---|
| Number of wheels & motors | A<br>2 & 2 | B<br>4 & 2 | C<br>4 & 4 |
| High mobility | -1 | 0 | 1 |
| Low power consumption | 1 | -1 | -1 |
| Stability | -1 | 1 | 1 |
| Turning | 1 | 0 | 0 |
| Sum | 0 | 0 | 1 |
| Continue? | No | No | Yes |

*Figure 5: example of decision matrices for the structure (wheels & motors)*

| | Possible Solutions | | |
|---|---|---|---|
| Which infrared receiver? | A<br>TSOP4838 | B<br>phototransistor from EEBug | C<br>BVP22F |
| Cost-effective | 1 | 1 | 1 |
| Directionality | 1 | -1 | 1 |
| Sensitivity | 1 | -1 | 1 |
| Filters out ambient light | 1 | -1 | 1 |
| Doesn't filter out necessary component of the signal | -1 | 1 | 1 |
| Sum | 3 | -1 | 5 |
| Continue? | No | No | Yes |

*Figure 6: example of decision matrices for the sensors (infrared receiver)*

To eliminate inefficient solutions and select the best solution, “decision-matrix method” was optimised. Figure 5 and 6 shows examples of the method for the structure and sensors respectively. In each figure, each possible solution was scored based on the criteria shown in the rows. + indicates that the solution meets the criterion while - indicates that the solution fails to do so; a zero implies that the solution has to relation to the criterion. The sum of all + and - signs is the score of the solution, and the solution with the highest score was considered the best solution. By using this method, the optimal solution to a specific problem was selected when given many options to solve it. The rest of the selection matrices of the app and sensors can be found in each section, and the ones of the structure of the rover can be found in **appendix III**.

# 6. The app

By communicating with the rover, the app enables the user to control the rover and displays information about the material discovered. MIT app invertor was chosen to create a simple app. App inventor is easy to use, and it reduces the load in the Arduino as the interface is stored in the app instead of the Arduino server. The app has a welcome screen, an introduction screen used to check that communication between the app and the Arduino is possible, and a cockpit screen.

## 6.1. Why This App

There were other alternatives to the use of AppInventor that were discarded after careful consideration:

- Direct communication between a C++ desktop windows program and the Arduino. This solution was very complicated compared to the use of app inventor: Knowledge on how to access WIFI from both windows in C++ and the Arduino, as well as on how to create user interfaces was needed.
- Exclusive use of a HTML web server: This solution was easy to implement by knowing some HTML and would allow to concentrate all the programming side of the project on the Arduino, instead of having two separate programs for the rover and for the controller. However, this solution involved loading a whole webpage into the Arduino, which could have affected its performance when taking care of other tasks.

| | Possible Solutions | | |
|---|---|---|---|
| Implementation method | AppInventor | HTML Interface | C++ Control Program |
| Easy to implement | 1 | 1 | -1 |
| Stable | 1 | -1 | 1 |
| High performance | 1 | 1 | 1 |
| Sum | 3 | 1 | 1 |
| Continue? | Yes | No | No |

*Figure 8: selection matrix of implementation method for the app*

## 6.2. Design and Functionality

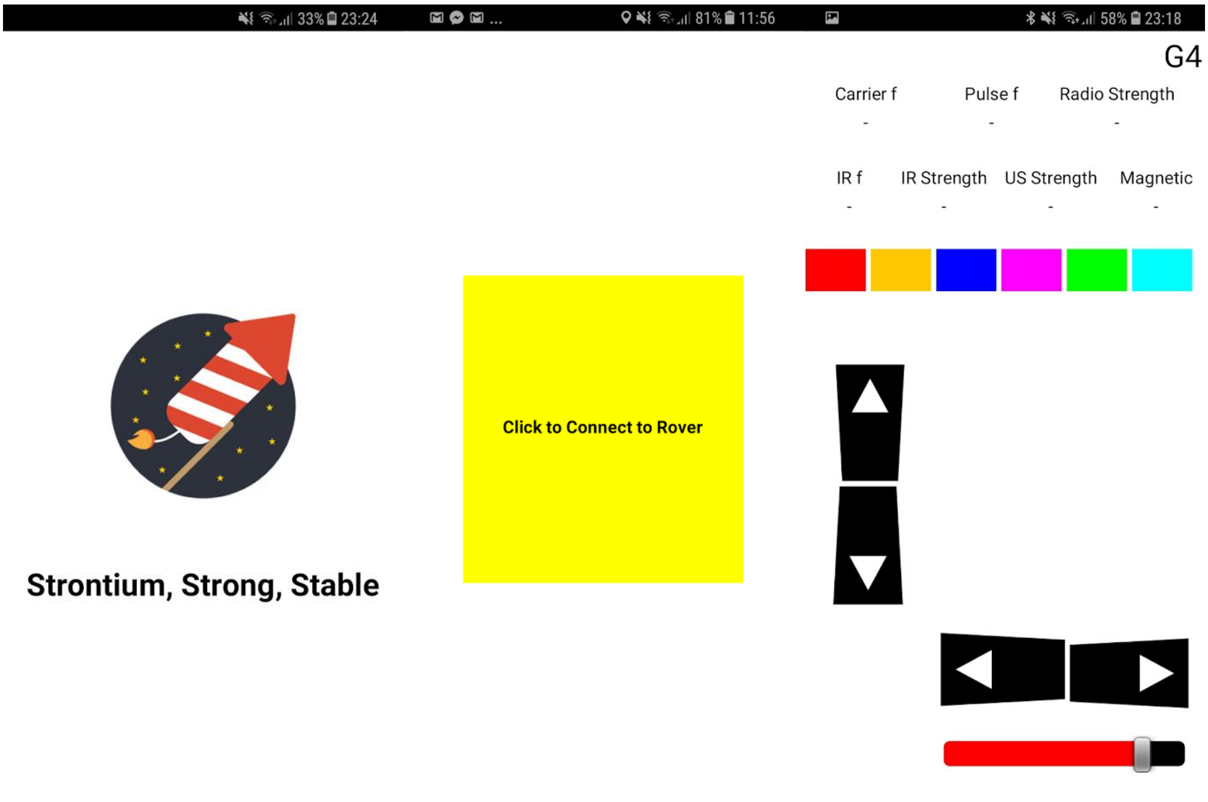



*Figure 7: screenshots of the app*

**Screen 1 (Left)**

A simple presentation screen shown for 1 second when the app is initialized.

**Intro Screen (Middle)**

This screen has a button used to go to the cockpit. When the app detects that connection has been lost while in the cockpit, it goes back to this screen and notifies the user that connection has been lost.

**Cockpit Screen (Right)**

Elements in the cockpit:

- Numerical information about the state of the sensor inputs.
- Visual information about materials being found. Figure 9 shows a hypothetical case where all 6 materials are being recognized at the same time.

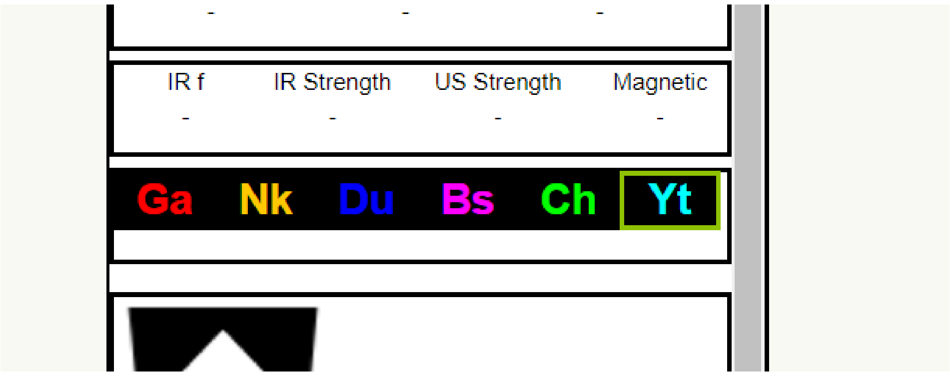



*Figure 9: Materials being found*

- A text element that will display the gear selected (G1 to G4).
- A slider to change gear (And with it the speed of the rover). Some intelligence had to be applied on the way that the app sends information to the Arduino based on the position of the slider. This is because while the variable containing the slider's position is very precise (contains decimal values) and therefore will change very quickly as the slider is dragged. Sending values too quickly over Bluetooth is not desired as it can cause instability. Therefore, the app has been designed to only send Bluetooth commands when the slider changes integer threshold (i.e.: if it changes from 1.99 to 2.00)
- Four directional arrows. When the arrows are pushed, they change their colour.
- While in the cockpit, the app checks every second whether there is information available to be received from the Bluetooth. More on this on the "Rover to App control" section

Natively, app inventor does not allow multitouch, but multitouch needed to be implemented in the app, so the left or right arrows can be pressed at the same time as the back or forward arrows to allow for all motion combinations. The solution was to use an additional extension [1].

It works by creating a 3x3 grid and it allows each of the 9 cells to be touched at the same time. The arrows have been placed in the top left, middle left, bottom middle and bottom right cells of the grid.

### 6.3. Material Detection Intelligence

As it will be discussed later, the Arduino will be able to provide data about the strength of radio or IR pulses even if it cannot identify correctly the pulse frequency. This fact, combined with information from the second property of each material, can be used to provide a second possibility to identify some materials in case the Arduino is not performing very well at reading signals. The logic from figure 10 has been implemented in the app, and tests have confirmed that this is a reliable recipe to correctly detect all 6 materials.

| **Material** | **Possibility A** | **Possibility B** |
|---|---|---|
| Ga | Carrier frequency 77k and pulse frequency 151 | Radio strength more than 0 and US strength more than 0 |
| Nk | Carrier frequency 67k and pulse frequency 239 | Carrier frequency 67k and magnetic field strength more than 0 |
| Du | Carrier frequency 103k and pulse frequency 151 | Carrier frequency 103k and magnetic field strength more than 0 |
| Bs | Carrier frequency 103k and pulse frequency 239 | - |
| Ch | Pulse frequency 421 | - |
| Yt | Pulse frequency 607 | IR strength more than 0 and US strength more than 0 |

*Figure 10: App's material finding logic*

# 7. The Arduino Code

The implemented code can read the output of the sensors, control the motors and other peripherals and communicate back and forth with the app. All this while dealing with the limitations on memory and multitasking capabilities of the Arduino.

## 7.1. Reading the Analog Inputs

In the case of the magnetic field sensor and the ultrasound sensor, this is a very simple task: A reading is performed every time the main loop of the Arduino is executed. Initial tests showed how the analogRead from the Arduino often oscillated for a few values (10 or so). Therefore, to prevent false readings due to noise, the readings are set to 0 if they are smaller than 15. In the case of the magnet, the reading is subtracted 512 to compensate for the 2.5V offset of the output.

Reading values for the radio and the ultrasound is more complicated: Every half a second the Arduino takes a read at either IR, Radio A, or Radio B. Radio A and B are the names given to the two different resonating capacitor configurations that can be applied by switching the MOSFET on or off.

This is how a reading is taken:

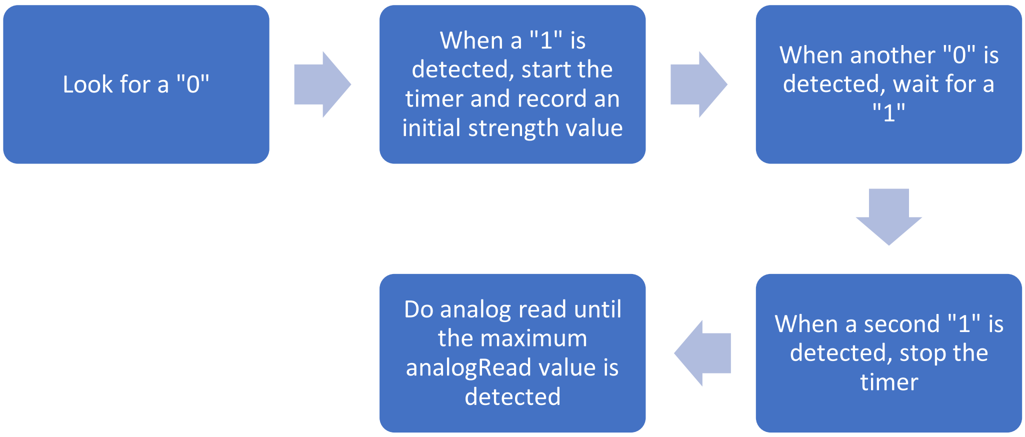



*Figure 11: Pulse reading algorithm*

A while loop allows the Arduino to focus strictly on listening to the analogue input for a short period of time (0.05 seconds). This ensures that the period of the square pulses will be read accurately. Tests seem to give values of the period slightly higher than what is expected, but differentiation between the two IR and radio square waves period is still perfectly possible.

The analogRead values for “0” are 10 for the IR, and 480 for the Radio. These “0” levels compensate for noise and the DC offset of the sensor’s output in the case of the radio.

The initial strength values taken for all three IR, Radio A and Radio B before the “maximum analogRead” is performed will allow the Arduino to tell the app that radio/infrared has been detected, and the carrier frequency, even if the pulses could not be sampled properly. This means that the app could still identify the materials using their second property (magnetic field or ultrasound)

After each read action is performed, the values of the radio strength with Radio A and Radio B are compared to extract the value of the carrier frequency.

## 7.2. The wireless communications system

For most of the time of development of this project, lots of hard work were put on creating a functional WIFI system. By updating to the 1.1.0 version the firmware and libraries, and Using HTML GET commands, plus some “tricks” where the Arduino stopped listening to the app once the essential information had been received, a virtually instantaneous response could be achieved when a command was sent from the app to the Arduino. However, this system was still unstable: The Arduino inexplicably stopped responding after a minute or so. This problem was also reported by some colleagues.

The solution was to switch from WIFI to Bluetooth. A HC-05 Bluetooth module was chosen. Fortunately, app inventor supports Bluetooth and making changes in both the app and the Arduino code was straight forward.

It was realized that even though the use of WIFI was the possibility suggested by the department, this is quite a bad idea: Bluetooth should have been the way to go from the beginning, as the advantages are numerous:

- Only the TX and RX pins of the Arduino (the ones that allow serial communication) are needed, therefore, the Bluetooth system will not require additional pins that will be free for other systems.
- Very easy to program
- Very stable
- We no longer rely on the EEERover WIFI router working properly: Communication is direct between the app and the rover.

- The Bluetooth module takes very little time to initialize: The rover can be controlled almost immediately after powering on.
- After switching from WIFI to Bluetooth, it was found that the Arduino code was taking much less memory than before. It seems like the WIFI libraries were taking quite a lot of memory. The fact that simply the serial is used instead of all the WIFI libraries probably also means that the load in the Arduino's processor is smaller, and therefore more time is available for other systems.

## 7.3. App to Rover Control

Information is only sent by the app when a button changes its state. Therefore, it's necessary for the Arduino to remember what the current state is to allow for directional arrow combinations. Logic is performed in the code with the following variables:

dir variable -> 0 when rover stopped, 1 when moving forwards, 2 when moving backwards

l variable -> true when rover moving left, false when not moving left

r variable -> true when rover moving right, false when not moving right

| **Action** | **Command** |
|---|---|
| Forward Pressed | F |
| Back Pressed | B |
| Forward/Back lifted | S |
| Right Pressed | R |
| Left Pressed | L |
| Right/Left lifted | s |
| Slider in Gear 1 | 1 |
| Slider in Gear 2 | 2 |
| Slider in Gear 3 | 3 |
| Slider in Gear 4 | 4 |
| Information to be sent from arduino to app | I |
| Material found, tell the Arduino to play some sound | g, n d, b, c, y |

*Figure 12: table of action and command for controlling the rover*

The fact that all the commands involve a single character makes the process of reading the serial from the Arduino much simpler.

### 7.4. Rover to App Control

Every second, the Arduino takes the processed numerical values from the sensors and creates a long string with them, leaving a space between value and value so the app can recognize where one piece of that stops and the next one start. Upon testing, it was found that simply storing the data in the string was not a very stable solution. Therefore, an additional "$" character was added so the app knows where the string is supposed to start:

| **Index** | **Data** |
|---|---|
| 1 | Initial Character "$" |
| 2 | Carrier Frequency |
| 3 | Radio Pulse Frequency |
| 4 | Radio Signal Strength |
| 5 | IR Pulse Frequency |
| 6 | IR Strength |
| 7 | Ultra Sound Strength |
| 7 | Magnetic Field Strength |

*Figure 13: How the data is organized within the string*

# 8. The Sound System

It was decided to include a speaker to reproduce sound files. The idea is to reproduce the name of the materials when these are found, as well as any other kind of sounds (for instance, wise quotes from the current president of the united states). A speaker from Farnell UK [2], which has enough power and a reasonable range of frequencies was used.

The 8-bit .wav files are stored in a micro micro SD card, and an adapter [3] is used to connect the card to the Arduino. A few libraries must be used for this sound system:

- SD.h: Allows the Arduino to use SD cards.
- SPI.h: SPI is a protocol used by microcontrollers like Arduino to communicate with peripheral devices.
- TMRpcm: Takes files and produces a PWM (Pulse width modulation) signals that can be fed onto the speaker. A very interesting property of this library is that sound signals will be emitted in the "background", i.e.: the Arduino will continue executing its main loop instead of stopping doing everything else while music is being played.

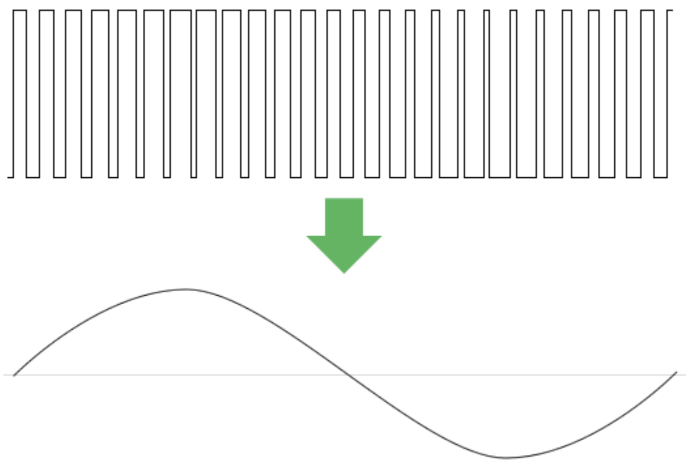


*Figure 14: how PWM produces sounds as an analogue signal*

The pins of the Arduino can only supply a maximum current of 20mA [4]so a very simple switch had to be implemented using a BJT that allows current to flow from the main power supply to the speaker when the Arduino speaker pin is "ON".

# 9. Sensors

## 9.1. Radio

### 9.1.1. How it Works

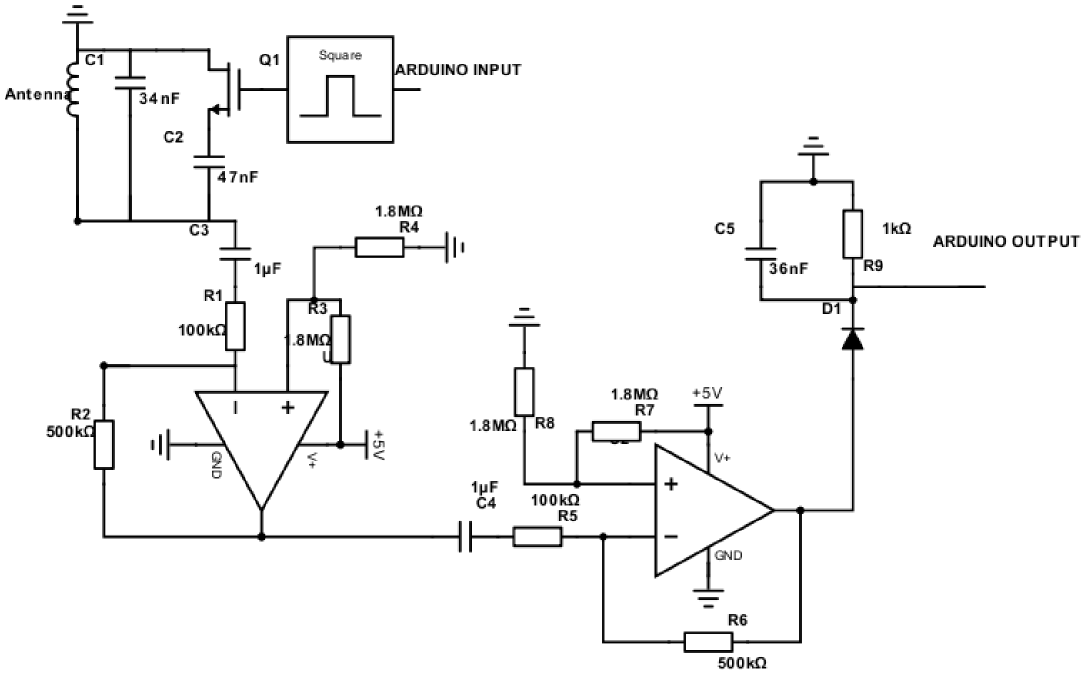



*Figure 15: circuit diagram of the radio circuit*

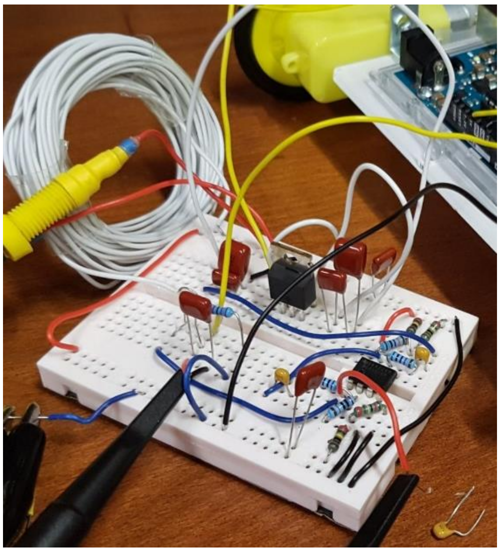


*Figure 16: picture of the radio circuit*

The radio input first enters the circuit through the antenna. The antenna is just an air cored inductor joined together with tape. As it can be seen in Figure 15, this inductor is connected to one set of capacitors of 34nF and to another set of capacitors of 47nF which are preceded by a

MOSFET. If the MOSFET is OFF, the signal goes through the 34nF set, this set of capacitors resonates at the 103kHz carrier frequency. If the MOSFET is ON, then the antenna is connected in parallel with another set of parallel capacitors that add up to 77nF. This circuit resonates at a frequency of 67kHz. As shown in the diagram, the function of turning ON and OFF the MOSFET is carried out by the Arduino, that can input either 0 or 5 volts (5 Volts happens to be above the threshold voltage of the MOSFET, which is what we want).

The next stage is the amplification stage. It is made up of two completely identical Op-amp amplifiers. First, the signal goes through a DC blocking capacitor. Both Op-amp have biasing voltage of +2.5V, the positive input is biased by using two 1.8MΩ resistors connected to the positive input terminal (+5V) as a potential divider, using these resistors provides a high input impedance. After the capacitor, the signal is connected to a 100k resistor that goes to the negative input terminal ($V_{in-}$) which is also connected to a 500k resistor that is connected to the output. These resistors provide negative feedback which sets the gain of the circuit. The total gain of the two stages of amplification is around 10.

Finally, the output of the second stage of amplification is connected to an envelope detector. This circuit oversees demodulation of the AM signal. The signal goes through a diode that is connected to a resistor and a capacitor in parallel (see **appendix VII** for more detail). The output of the whole circuit is the node connecting the diode with the resistor and the capacitor.

We had to run different tests at distinct stages of the circuit to ensure everything worked. It began with testing the antenna; one end of the antenna was connected to ground, and the other end was the output of the test. As soon as the flask was on, it was observed that it was picking up a signal, there wasn't any modification needed. Then, the impedance of the antenna was calculated, and based on the result, adequate capacitors were chosen to make it resonate at the various carrier frequencies. The same test was done but this time with the capacitors calculated in parallel with the antenna. The calculated values were found to not resonate much: The test had to be repeated several times for each of the two frequencies making small changes to the capacitance until propper resonance appeared.

Then, the Op-amp amplifier was designed, the same test was done, now connecting the output of the previous test to the input of the amplifier. The Output of the amplifier was measured. Although no signal was observed at first, by changing the gain of the circuit, it was discovered that trying to put too much gain on one stage didn't work; therefore, two stages with smaller gains were optimised. After adjusting the gain of each stage, the signal was amplified enough to make it detectable at around 30 centimetres. The enveloper stage has a low input impedance, which contributed towards the need for a large gain.

When testing the envelope detector, trial and error method was used to find the right resistor and capacitor values, always bearing in mind that the time constant had to be larger than 1/RC. Slightly deformed square waves were obtained because of the slow discharging of the capacitor, Furthermore, sometimes, a small fraction of the original signal was observed in the demodulated signal because the time constant was too small. However, after many trials, the right values were found.

During the designing stage of radio sensor, it was concluded that the sensor could be made without ordering stuff online by using components and materials from the lab. The goal was to minimise as much as possible the cost of the sensor and the circuitry related with the sensor. To design the antenna, the option of buying one was dismissed because our hand made antenna worked well enough.

*Figure 17: the first prototype of the antenna*

### 9.1.2. Innovative Ideas

When finding ways to improve the antenna, one innovative idea came from the equations that describe the behaviour of an inductor. These say that the inductance of an antenna is proportional to the turns of the inductor squared. An antenna with three times as much turns (Figure 18) as the first prototype (Figure 17) was built and tested. This new antenna gave the desired output.

*Figure 18: the final antenna design*

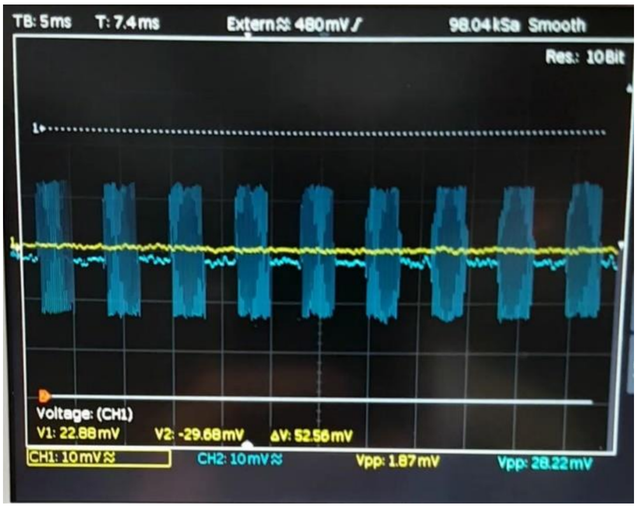



*Figure 19: Image of output of antenna when resonating at 103kHz*

The next innovative stage was to distinguish the two carrier waves. First idea was trying to use filters, which didn't give good results.  The next two figures show the brainstorming process that gave rise to this idea.

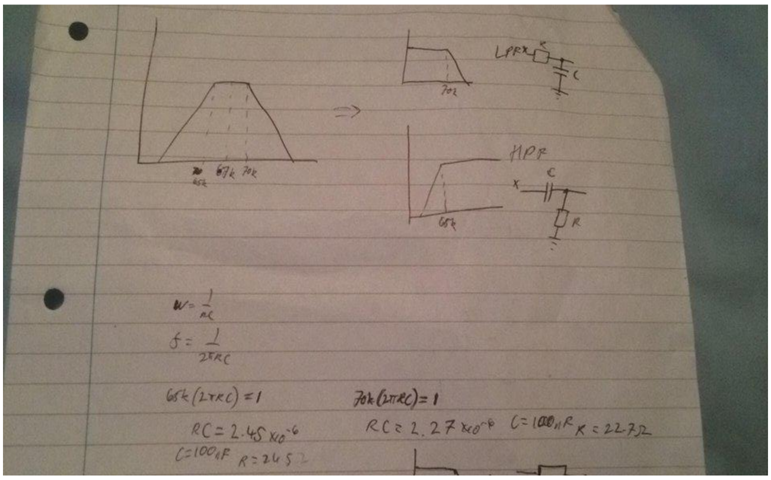



*Figure 20: image of sketch for filter for 67kHz*

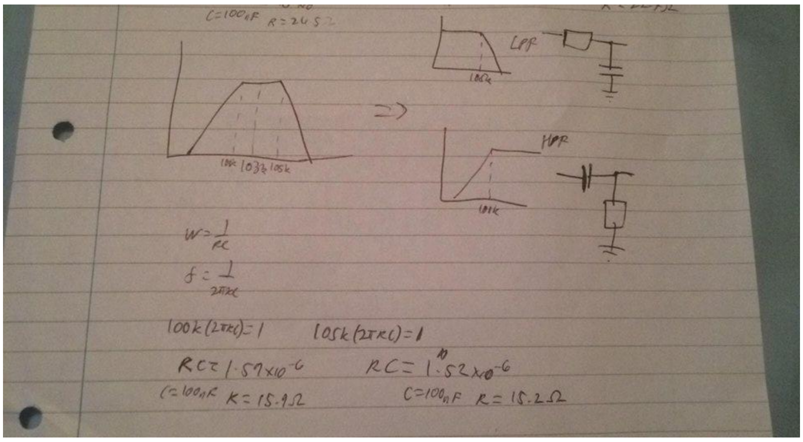



*Figure 21: image of sketch for filter for 103kHz*

After this, resonance was considered to increase the signal at the right frequency, and good results were obtained. The inductance of the antenna was measured experimentally by using a known resistance, applying a signal, and measuring the voltage difference between the signal and the resonant circuit. Once the inductance was known, the capacitance, which, in series, with the inductor, would make the circuit resonate at 103kHz and 67kHz, was calculated. It was discovered that the capacitances needed were 34nF for the 103kHz signal and 77nF for the 67kHz signal.

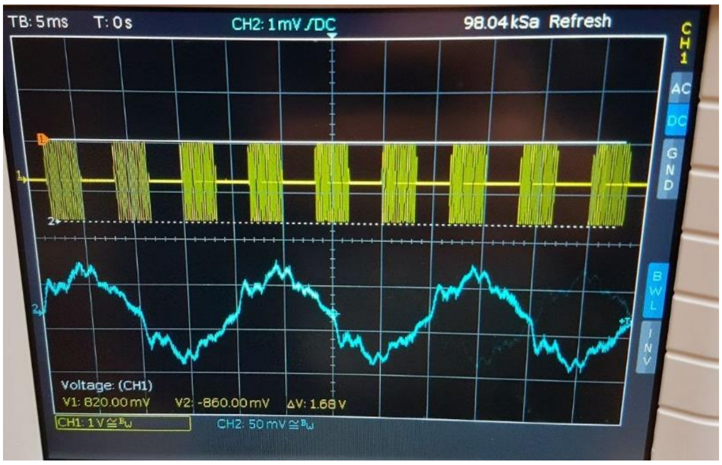



*Figure 22: Image of output of antenna when not resonating*

Although the amplitude of the signal looked high enough just by making the circuit resonate, some aspects like increasing the detection range of the circuit, the 0.7V that the diode of the envelope detector[1] takes away, and the low input impedance of the envelope detector itself, required the use of a good amplifier. The three ways of amplifying signals seen in analysis of circuits and analog electronics  modules were proposed as ideas: MOSFETs, BJTs and Op-amps. Op-amps were chosen after a few tests because circuits involving them are easy to design, they are less prone to malfunction, and they help to reduce noise.

Different antennas that could be made were discussed. One option was to do an antenna with a circle made of cardboard and coil the wire around the cardboard (this idea was quickly discarded), another idea was to coil the cables and join them with tape and leaving just air in the centre of the radio. Antennas with large diameters and small diameters and ones with a lot of turns and fewer turns were all considered. The idea of building an antenna without the protective layer of the cable and using a special varnish was also brought up. Figure 17, 18 and 23 show the different antennas, which were made and taken into consideration.

*Figure 23: image of the second prototype of the antenna*

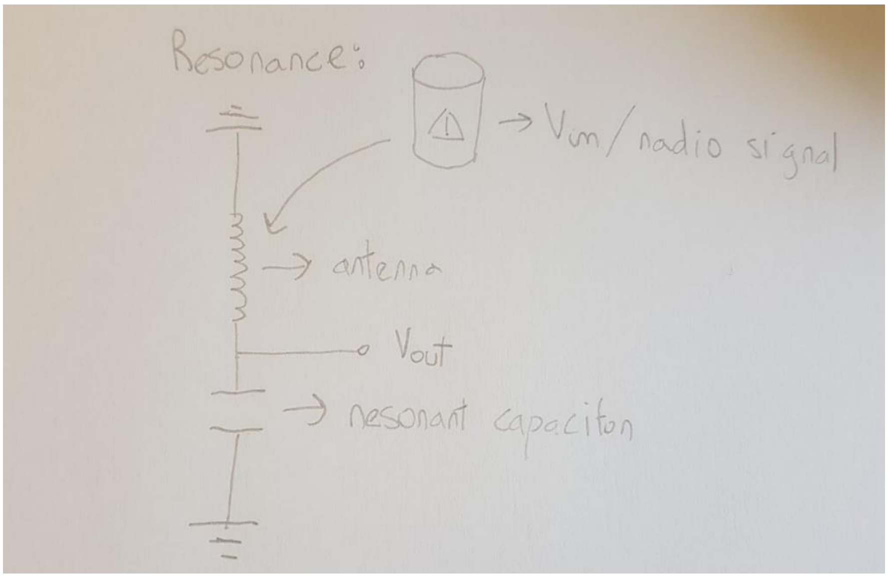



*Figure 24: image of initial sketch of resonance circuit*

The final innovative step was to make the Arduino detect the different carrier frequencies. The problem was that the sensor had two different sets of capacitors, one for each frequency, but they needed to be put together. The solution was the use of a MOSFET as mentioned above.

The final radio sensor can detect radio signals up to a range of 20cm. Another strength of the sensor is that excluding the MOSFET, which costed £0.68, this sensor is completely made of lab materials, which means that the impact of this sensor within the budget is minimal.

On the other hand, the antenna can only receive a signal when the flask is pointing to the front or back side, if the flask is emitting directly to the edges of the antenna, this will not pick any signal. This was resolved by putting the antenna right at the front of the rover, and the part of the antenna that detects the signals was directed to the front. This way, when the rover is moving directly to the flask it will the detect the signal.

## 9.2. Ultrasound sensor

The ultrasound circuit comprises of four main parts; the sensor, the amplifier, the filter and the rectifier. The sensor an ultrasonic receiver that is calibrated to detect 40kHz was found. After the testing the ultrasonic receiver, it was clear the output was not high enough to be sensed by the Arduino. Amplification was required, and a biased inverting amplifier was selected. The signal from the output of the sensor goes through a capacitor and resistor into the $V_{in-}$ terminal of the op-amp. There is a potential divider at the $V_{in+}$ terminal which splits the 5V input in half. There is also a very large resistor, 10MΩ, to provide negative feedback, connecting output to $V_{in-}$. To get the output of the amplifier back to 0 DC offset, a High-Pass filter was used. The HPF is comprised of a capacitor and a resistor. Then a rectifier was used to make the signal a constant DC signal, so that the Arduino could easily tell if it was getting closer to the source of the ultrasound. For this the signal was passed through a diode. Then, a capacitor and resistor in parallel connected to ground clear the output signal. The output signal sent to the Arduino is, when viewed on an oscilloscope, a straight horizontal line with a DC offset.

Our initial thoughts about how to make this circuit included using a microphone or something similar, to detect the ultrasound and then put the output signal through a filter. When searching for a microphone, sensors that work specifically for our ultrasound frequency (40KHz) were found. After testing the sensor and amplifying the output, a way of making the signal easier for Arduino to understand was discussed. This led to using the rectifier so that the output is a horizontal line on the oscilloscope, the voltage of the output increases the closer the sensor gets to the source. During testing, it was found that the sensor could detect at about a 45° angle from the normal (see Figure 25), and had therefore a pretty good directionality.

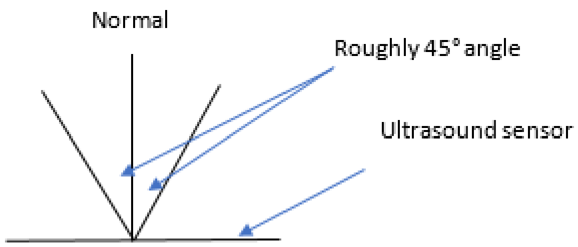



*Figure 25: Ultrasound sensor angle of detection*

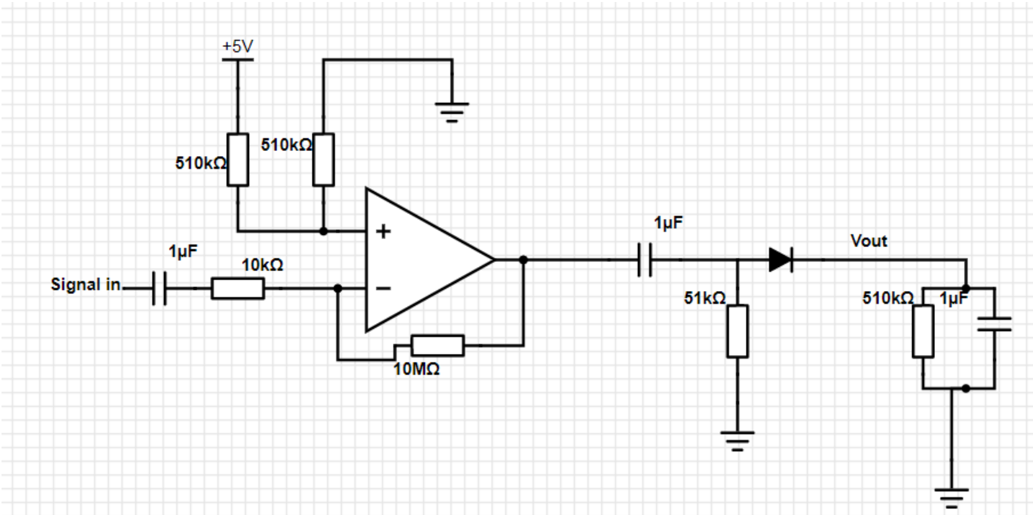



*Figure 26: Ultrasound amplification circuit*

### 9.3. Infrared sensor

#### 9.3.1. Design and Development

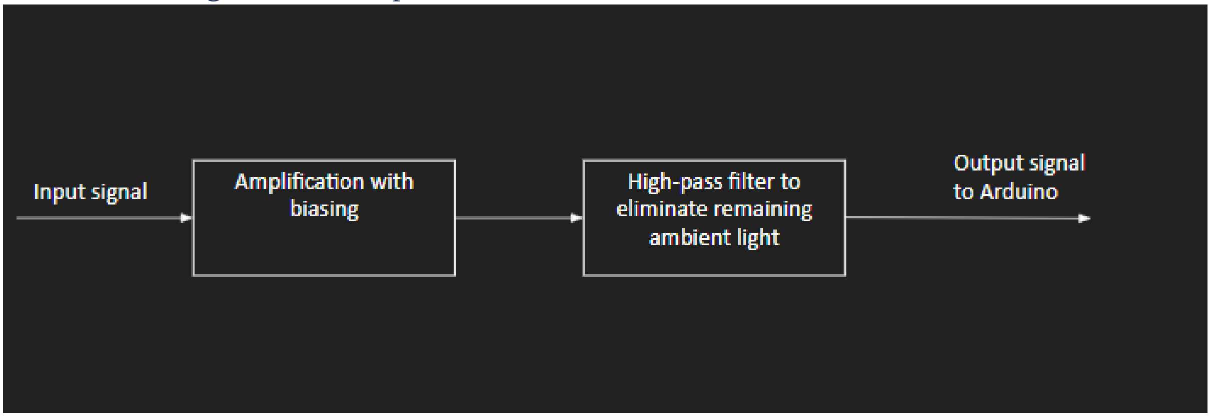



*Figure 27: Conceptual design of infrared sensor*

As Figure 27 shows, the photodiode detects infrared signal and sends an input signal to the amplification device. The amplified signal is then filtered by high-pass filter to get rid of ambient light. Then the output signal goes to Arduino.

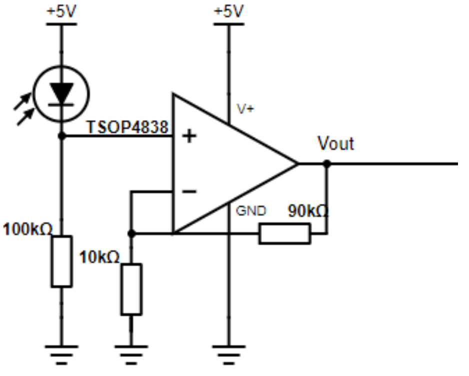



*Figure 28: Circuit diagram of the initial infrared sensor circuit (TSOP4838)*

Initial design of the circuit is shown in Figure 28. The TSOP4838 infrared receiver is connected in series with a 100kΩ resistor to ensure the input to the amplifier is as close to 5V as possible (The sensor will absorb some potential). A non-inverting amplifier was implemented. The gain of the circuit is:

$$\frac{V_{out}}{V_{in}} = A_v = \frac{10k + 90k}{10k} = 10$$

However, since TSOP4838 contains integrated circuits that filter out some components of infrared signal. This was a sensor designed for purposes such as TV remote control: It was concluded that the sensor is not suitable for the purpose.

We then proceeded to work with BVP22F, a photodiode. As well as not containing any integrated circuit, BVP22F had good directionality and peak sensitivity at about 950nm of wavelength [6]. Furthermore, this is a cheaper component.

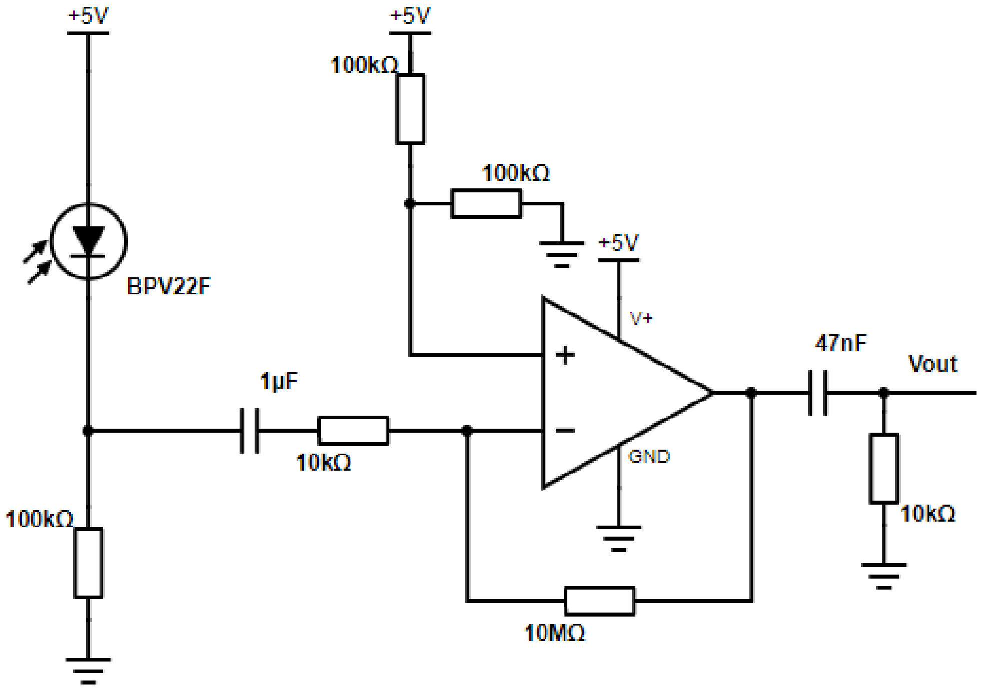



*Figure 29: Circuit diagram of the full infrared sensor circuit (BVP22F)*

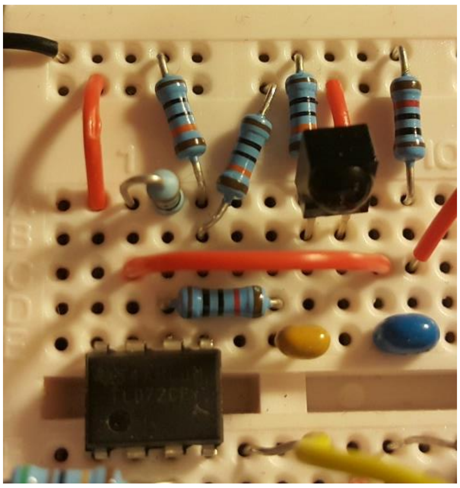


*Figure 30: Picture of the full infrared sensor circuit*

### 9.3.2. Stages of the Circuit

The photodiode in the circuit has 2 legs – one connected to positive 5V power supply and the other connected to the non-inverting amplifier. If the photodiode detects any infrared signal, it will allow some voltage flow into the circuit from the +5V power supply. The amount of voltage that flows into the circuit is determined by the intensity of the infrared signal, which depends on two components – the proximity of the signal source and the direction in which the infrared signal is transmitted. The input signal appears as a square wave, which needs to be amplified as it is too small for Arduino to detect.

As shown in Figure 29, the input signal is amplified using an inverting amplifier circuit. There are several reasons why an inverting amplifier was chosen over a non-inverting amplifier. First, the phase of the output signal doesn't interrupt Arduino's signal detection. This gives a choice between inverting and non-inverting amplifiers. Also, because there is ambient light from undesired sources, having a non-inverting amplifier would cause a huge voltage drop as it has a high input impedance; this would lead to high signal-to-noise ratio and less efficiency. Therefore, by using an inverting amplifier, which has a low input impedance, reduction in the voltage drop caused by ambient light and improvement of the efficiency of the circuit can be achieved.

The gain of the amplifier is:

$$\frac{V_{out}}{V_{in}} = A_v = -\frac{10M}{1k} = -1000$$

Furthermore, there is a biasing of 2.5V at the positive terminal of the operational amplifier because the power supply cannot supply negative voltage. DC offset of the output signal was hence 2.5V; thus, Arduino can detect the full output signal.

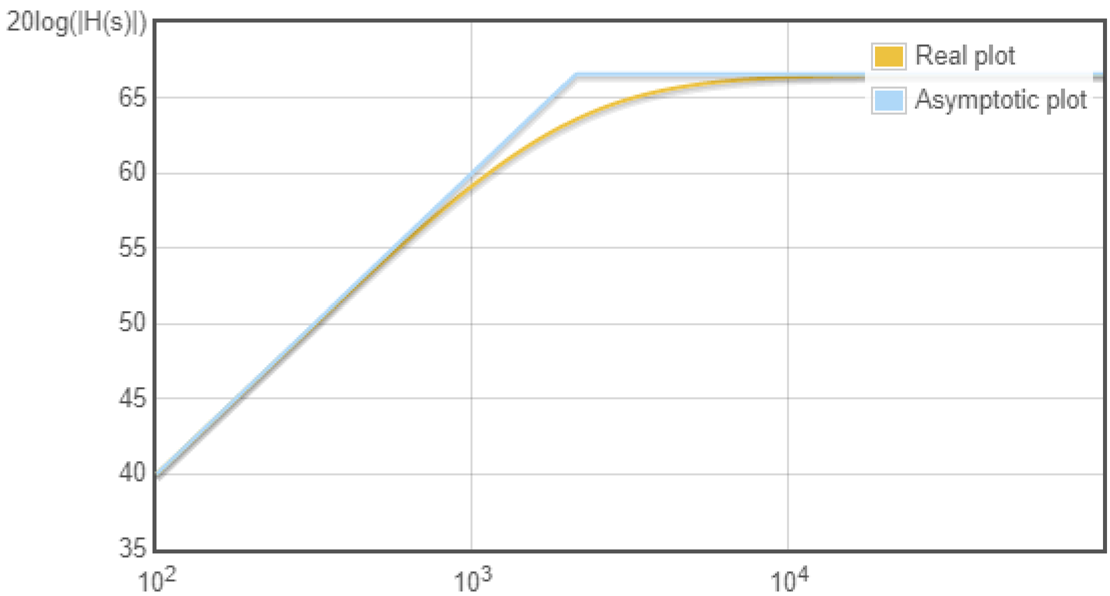



*Figure 31: Magnitude response of the high-pass filter*

Now, eliminating the rest of the ambient light is needed. Since ambient light has a frequency about 50Hz, optimising a high-pass filter can eliminate it. Hence, a high-pass filter, whose corner frequency is 2127.66rad/s or 338.63Hz, is implemented after the amplification stage. As shown in Figure 31, any signal whose frequency is lower than the corner frequency should be eliminated, and since the typical frequency of ambient light lies well below 338.63Hz, the ambient light will be eliminated. Finally, the output signal from the circuit is sent to Arduino.

### 9.3.3. Limitations of the Photodiode

There are several limitations of the circuit. The first one is that the directionality is not sufficient to detect infrared signal from any direction. Since infrared signal is transmitted in one direction, the photodiode must be placed at a spot where the sensor can detect the infrared signal. If the flask is upright, the photodiode must be placed high enough. Also, some flasks may be lying flat on the ground. This leads to using 2 sets of sensors as well as an extension of the rover's chassis.

### 9.3.4. Test


*Figure 32: output waveform of infrared sensor (without the lid on)*


*Figure 33: output waveform of infrared sensor (with the lid on)*

When testing the sensor without the flask's lid on, the waveform was clear and large as shown in Figure 32; furthermore, the sensor could detect infrared from a long distance, say, 1 meter. With the lid on, the sensor's detectable range significantly decreased to about 10 centimetres, and the waveform gets somewhat deformed as shown in Figure 33. This is not optimum, but it's still good enough to guarantee that the Arduino will detect the signal.

## 9.4. Magnetic field sensor

The rover must detect a static magnetic field. Two ideas can fit this description. One, is to have a moving wire in the design of the rover, where the sensor would detect the magnetic field through electromagnetic induction, or two, investigate IC sensors that can detect a magnetic field. In deciding, it was necessary to investigate the advantages and disadvantages of each system.

In implementing the first option using the rover as a source of motion, the costing of the sensor would be zero since all components needed could be found in the lab, however, accessibility to the flask would be a big factor since the conductor must be located rather close to the magnet and perpendicular to the magnetic field for ideal results. Since accessibility and magnet orientation within the flask was not well defined, there were high risks of the system not being functional. The second option proved to be more versatile and budgeting of the rover proved that it was within the limits to have some expenses on magnetic sensors, which were found to be rather cheap. Thus, the focus was on the second option and held the first one as buck-up.

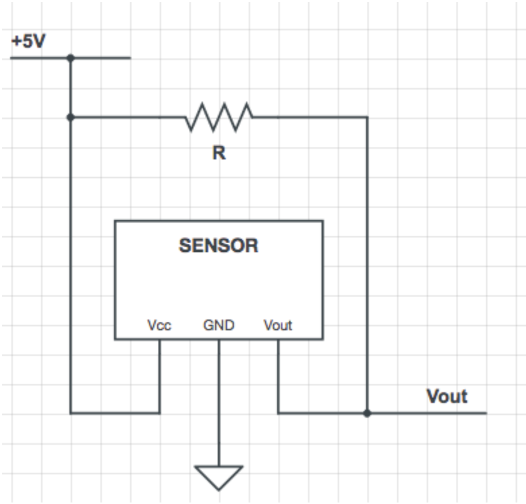



*Figure 34: First test circuit*

The sensor to be used should be able to detect a static magnetic field regardless of polarity and provide an output, easily interpreted by the Arduino. An omni-polar hall effect switch proved to be ideal. A sensor that is powered by 5V and shorts the Vout pin to the GND pin in the presence of a magnetic field above its operating point value (100 Gauss for this sensor). Using the circuit shown in Figure 34, in the absence of a magnetic field, Vout is connected to +5V, and since there is no current through R, Vout would be at 5V as well. In the presence of a field, Vout will be simply grounded. The sensor was simple to implement, was easy to interpret by the Arduino, and held a small footprint on the breadboard.

In testing, the sensor proved impractical, because the operating point was achieved within just 5mm from the magnet. Since the magnet would be located inside the flask and its exact location wouldn’t be known, a more sensitive sensor was required to retest the circuit. A sensor was obtained with an operating point of 7 Gauss. It provided a greater range of detection (1 cm). Although the results had improved and were sufficient to be used as the final magnetic sensor implementation, the systems limitations left the results rather unsatisfactory. The sensitivity was fixed, and more than one sensor would be needed to determine if a magnetic field is present.

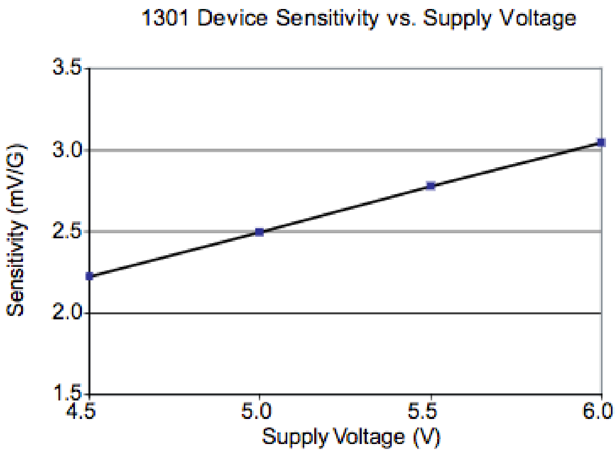



*Figure 35: linear sensor sensitivity*

With further research in IC sensors, it was found that a linear hall effect sensor could resolve the limitations that the initial design had. This type of sensor has a quiescent output of half the Vcc and varies linearly in the presence of a magnetic field, increasing or decreasing in the presence of a south or north magnetic field respectively. Using Figure 35 found in the sensors data sheet [7] a change of 2.5 mV per Gauss was expected.

The Arduino can sense a change of up to 10mV so keeping that requirement in mind, using the oscilloscope, a sufficient change in output voltage is achieved at 1cm, however this setup allows for modifications.

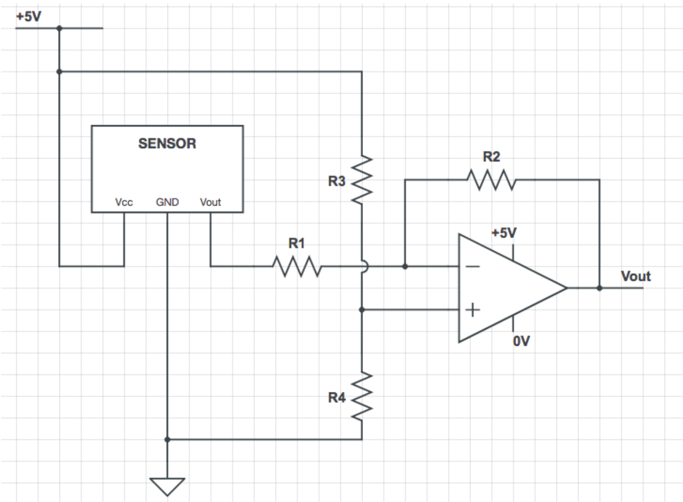



*Figure 36: Final circuit design*

In Figure 36, you can see the final circuit. With this sensor, it was possible to amplify the signal and therefore increase its range, since the output voltage increased proportional to magnetic flux density. An inverting amplifier was used since its minimal implementation would allow for the smallest footprint. Since the quiescent value is half of the Vcc, it was required to bias the non-inverting input of the op-amp to the same value. Thus, a potential divider where R3=R4 was utilised. The gain would be R2/R1, so the sensitivity of the sensor can be easily varied. Since the non-inverting input of the op-amp is biased at 2.5V and voltage values bellow 0 aren't required, a single supply configuration could be used, i.e. ground the lower power supply of the op-amp. Using values of R2 = 5.1kΩ and R1 = 1kΩ gave a satisfactory range to work with of about 3 cm. So, per increase in magnetic flux density of 1G, the sensor would deviate from 2.5V by 2.5mV, the op-amp would amplify this difference and output a change of 2.5 x 5.1 (= 10.25mv) which will be detected by the Arduino and indicate the presence of the magnetic field.

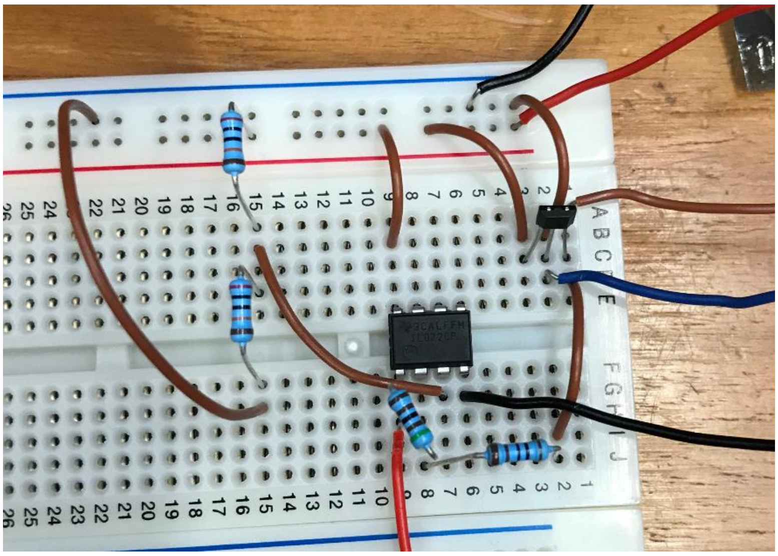


*Figure 37: image of final circuit*

| | Possible Solutions | |
|---|---|---|
| Ideas | Electromagnetic induction | IC sensor |
| Cost-effective | 1 | 0 |
| Footprint on the rover | -1 | 1 |
| Versatility | -1 | 1 |
| Sum | -1 | 2 |
| Continue? | No | Yes |

*Figure 38: selection matrix of magnetic field sensor*

# 10. Motion and structure

## 10.1. Wheels

A lot of brainstorming and testing went into the selection of the motor and wheel configuration. Starting from the wheels, there were three options: two, three or four wheels. Two wheels would require a third contact point with the ground. Three wheels configuration would be implemented using a third smaller wheel or an omni-wheel (Figure 39). However, since it is required for the rover to maneuver through obstacles and be able to operate on rougher terrain such as sand, these options were not as favourable. A four-rubber-wheel configuration would provide increased traction and stability. (Figure 40)

*Figure 39: omni-wheel*

*Figure 40: Rubber wheel used*

The four wheels can be implemented with either two or four motors. Four motors of course come with the disadvantage of higher power consumption but have the advantage of greater mobility and precision turning. Two motors would mean that non-driven wheels would require a free spinning mechanism. That would either be a bearing fastening the wheel to the chassis, or an axel to hold the two wheels together within a housing. Both concepts come with added complexity to the manufacturing process. Keeping this in mind, as well as the fact that we are not mechanical engineers, it was agreed that the rover would use the four-motor configuration.

| | Possible Solutions | |
|---|---|---|
| Size of wheels | A<br>Use the EEBug wheels | B<br>Get bigger ones |
| High mobility | -1 | 1 |
| Stability | 1 | 1 |
| Off-road capability | -1 | 1 |
| Ability to go over obstacles | -1 | 1 |
| Sum | -2 | 4 |
| Continue? | No | Yes |

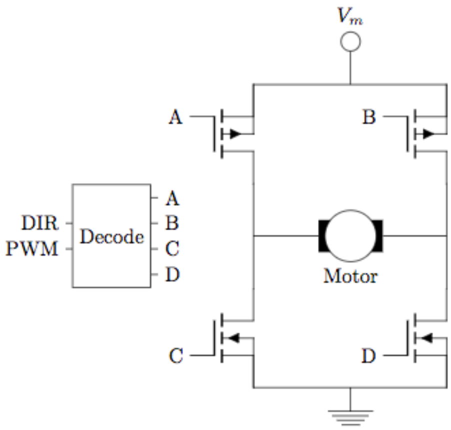
| | Possible Solutions | |
|---|---|---|
| Steering system | A<br>One wheel turns while the other stays motionless | B<br>Two wheels turn in<br>the opposite direction |
| High turning speed | 1 | -1 |
| Small turning radius | 1 | -1 |
| Sum | 2 | -2 |
| Continue? | Yes | No |

| | Possible Solutions | | | |
|---|---|---|---|---|
| Type of wheels | A<br>Normal tyres | B<br>Spiked wheels | C<br>Continuous track | D<br>Omni-wheels |
| Turning | 1 | -1 | -1 | 1 |
| Ability to go over obstacles | 0 | 1 | 1 | -1 |
| Off-road capability | 0 | 1 | 1 | -1 |
| Cost-effective | 1 | -1 | -1 | 0 |
| Mobility on a smooth surface | 1 | 0 | 1 | 1 |
| Sum | 3 | 0 | 1 | 0 |
| Continue? | Yes | No | No | No |

## 10.2. Chassis

To minimize expenses, it was decided to use the acrylic chassis provided with the EEBugs. To accommodate all our circuits and components, two of the acrylic chassis are going to be used one on top of the other. Low center of gravity and low weight are the main components that the design has to satisfy.  In Figure 42, you can see a prototype of the arm that will be used to mantle the sensors and scan vertical flasks. Since the radio wave sensor must be vertical, it will also be mounted vertically on the arm.

Setting up the control system for our 4 motors was very straight forward by using the H-Bridge given to us in the starter kit and feeding its inputs to the Arduino. Two inputs will indicate direction, and two inputs will indicate speed. The two left and the two right motors are connected to the same h-bridge outputs.




| DIR | PWM | A | B | C | D | Motor |
|---|---|---|---|---|---|---|
| x | 0 | $V_m$ | $V_m$ | $V_m$ | $V_m$ | off |
| 0 | 1 | 0V | $V_m$ | 0V | $V_m$ | forwards |
| 1 | 1 | $V_m$ | 0V | $V_m$ | 0V | reverse |

*Figure 41: single channel of module and truth table*

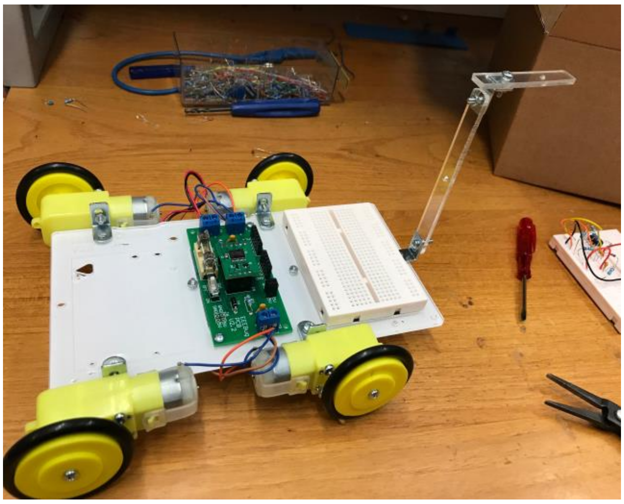


*Figure 42: arm prototype*

## 11. Conclusion

Throughout the project, the priority has always been maximising the reliability of rover. By doing so, it was ensured that the rover is versatile, durable and competent in performing its duty. “Reinventing the wheel” wasn’t tried, and simple, easy solutions, when these were good enough, were always the first option. (e.g. keeping the same EEBug chassis). Even if there were some difficulties in building the rover, with the aid of concept generation and selection, detailed planning and time-tracking, the project is successfully heading towards completion.

Also, there were other several proposed ideas: proximity sensor for detecting obstacles, image recognition algorithm for auto-pilot mode, differential steering system for better turning, etc. However, due to time and budget constraints, those ideas couldn’t be implemented. If the constraints were more flexible, the rover would be designed with above additional features as well as a better-looking chassis for aesthetics.

# 12. Components List

## 12.1. Rover Chassis (excluding sensor circuits)

| Description | Manufacturer | Quantity | Price (if ordered) |
|---|---|---|---|
| 3mm Clear Acrylic Robot Chassis V2.1 | Imperial College | 1 | £0 |
| Motor and Gearbox assembly | Imperial College | 2 | £0 |
| EEBug wheel | Imperial College | 2 | £0 |
| Rubber O-ring | Imperial College | 2 | £0 |
| Rubber wheel | Multicomp | 2 | £3 |
| 4-way LR6 Alkaline Cell Holder | Keystone | 1 | £1.48 |
| Solderless Breadboard 390 Points | K&H | 3 | £12.51 |
| EEBug Printed Circuit Board V2.2 | Imperial College | 1 | £0 |
| Alkaline-Manganese Dioxide Battery, LR6, 1.5V | Energizer | 4 | £3.99 |
| 100nF Multilayer Ceramic Capacitor | Vishay | 1 | £0.189 |
| Silicon Rectifier Diode, 400V, 1.0A | ON semi | 1 | £1.5 |
| 20mm Fuse Holder | CamdenBoss | 1 | £0.17 |
| Cartridge Fuse, F, 1A, 5.2 x 20mm | Littlefuse | 1 | £0.126 |
| 2x2 way Receptacle, 2.54mm pitch | Truconnect | 1 | £0.08 |
| 6-way Receptacle, 2.54mm pitch | Truconnect | 1 | £0.98 |
| 2-way Screw Terminal, 5mm pitch | Multicomp | 3 | £0.54 |
| 1Ω Resistor, 0.25W | Vishay | 2 | £0.404 |
| SPDT Slide Switch | ALPS | 1 | £0.3 |
| Arduino Uno | Arduino | 1 | £3.71 |
| Speaker | Visaton | 1 | £2.29 |
| BJT | Multicomp | 1 | £0.0604 |
| Total | | | £31.3294 |

*Figure 43: component list of chassis*

## 12.2. Ultrasound Circuit

| Description | Manufacturer | Quantity | Price (if ordered) |
|---|---|---|---|
| Ultrasonic sensor, 40kHz | Prowave | 1 | £4.71 |
| TL072 dual operational amplifier | Texas instruments | 1 | £0.501 |
| 1μF multilayer ceramic capacitor | Multicomp | 1 | £0.567 |
| Diode | Multicomp | 1 | £0.07 |
| 10kΩ Resistor, 0.25W | Vishay | 1 | £0.095 |
| 51kΩ Resistor, 0.25W | Vishay | 1 | £0.243 |
| 510kΩ Resistor, 0.25W | Vishay | 3 | £0.381 |
| 10MΩ Resistor, 0.25W | Vishay | 1 | 0.378 |
| Total | | | £6.945 |

Figure 44: component list of ultrasonic sensor

## 12.3. Infrared Circuit

| Description | Manufacturer | Quantity | Price (if ordered) |
|---|---|---|---|
| BPV22F photodiode | Vishay | 2 | £1.308 |
| TL072 dual operational amplifier | Texas instruments | 1 | £0.501 |
| 47nF capacitor | Multicomp | 1 | £0.182 |
| 1μF capacitor | Multicomp | 1 | £0.185 |
| 10kΩ Resistor, 0.25W | Vishay | 2 | £0.19 |
| 100kΩ Resistor, 0.25W | Vishay | 3 | £0.624 |
| 10MΩ Resistor, 0.25W | Vishay | 1 | £0.378 |
| Total | | | £3.368 |

*Figure 45: component list of infrared sensor*

## 12.4. Radio Circuit

| Description | Manufacturer | Quantity | Price (if ordered) |
|---|---|---|---|
| TL072 dual operational amplifier | Texas instrument | 1 | £0.501 |
| Diode | Multicomp | 1 | £0.07 |
| MOSFET (N-channel enhancement) | Vishay | 1 | £0.68 |
| 34nF capacitor | Multicomp | 1 | £0.182 |
| 36nF capacitor | Multicomp | 1 | £0.186 |
| 47nF capacitor | Multicomp | 1 | £0.191 |
| 1μF ceramic capacitor | Multicomp | 2 | £0.364 |
| 1kΩ Resistor, 0.25W | Vishay | 1 | £0.19 |
| 100kΩ Resistor, 0.25W | Vishay | 2 | £0.416 |
| 500kΩ Resistor, 0.25W | Vishay | 2 | £0.224 |
| 1.8MΩ Resistor, 0.25W | Vishay | 4 | £0.912 |
| Total | | | £3.916 |

*Figure 46: component list of radio wave sensor*

## 12.5. Magnetic Circuit

| Description | Manufacturer | Quantity | Price (if ordered) |
|---|---|---|---|
| TL072 Op-Amp | Texas instrument | 1 | £0.501 |
| A1301 Linear Hall Effect Sensor | Allegro microsystems | 2 | £0.837 |
| 100kΩ Resistor, 0.25W | Vishay | 2 | £0.208 |
| 5.1kΩ Resistor, 0.25W | Vishay | 1 | £0.201 |
| 1kΩ Resistor, 0.25W | Vishay | 1 | £0.19 |
| Total | | | £1.937 |

*Figure 47: component list of magnetic sensor*

## 13. A Possible Extra: LED Matrix

WARNING: This system is, at the time of writing of this report, on an early stage of development. Also, the only reason the idea is going forward is because it involves an interesting technical challenge to solve, not because it could serve any kind of practical purpose to the rover. That's why we might not end up including the LED matrix in the rover.

As an interesting way of doing something with the money left on our EEERover balance, and of applying knowledge from our digital electronics module, the following system could be incorporated to the rover: A 8*8 LED matrix.

Because the limited pins available in the Arduino, the challenge is to control the matrix using as few pins as possible. We wanted to use a system that works like the old cathode tube TVs: It will swipe very quickly across all the 64 LEDs, giving to each of them a 0 or 1 to form an image. If the swiping speed is big enough, the illusion of all LEDs being controlled at the same time will appear. This is the idea we came up with after doing some thinking about the problem:

A clock will drive a counter. The output of the counter will be a binary number from 0 to 8 that will be incremented with each tick of the clock. The three binary digits will be used to select in a demultiplexer which of the 8 pins much be connected to 5V. To prevent the LEDs from burning, a 0.1K resistor was placed between 5V and the "COM" pin of the column demultiplexer. The last digit of the first counter will also be used as a clock for a second counter that will do something similar: Connect to a demultiplexer that will select which row is going to be given access to ground.

Something interesting about how digital electronic systems must be used in practical applications was realized: Even if a pin is not being used at all, the system will be unstable if that pin is left "floating" instead of connecting it to the "0" voltage level or the "1" voltage level.

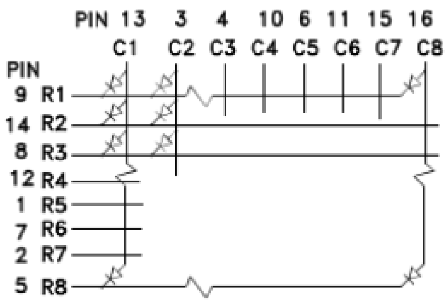
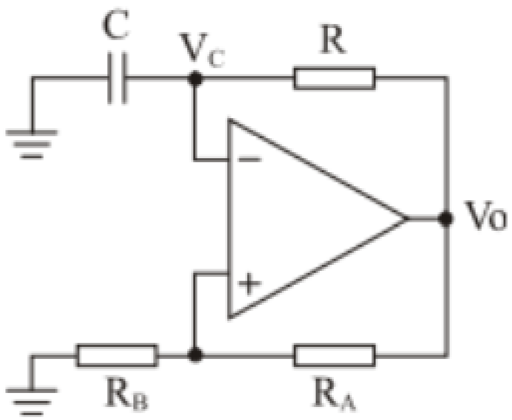



*Figure 48: Circuit Diagram of the Array*

To provide a clock for this system, a very simple switching oscillator was developed:




| | |
|---|---|
| R = 11K | C = 47nF |
| Rb = 10K | Ra = 15K |

*Figure 49: Circuit Diagram of the Clock*

DC biasing had to be applied once more: The Vcc+ of the op amp was at 5V, the Vcc- of the op amp was at 0V, and the terminals shown as ground on the circuit diagram were in reality connected to a voltage divider supplying 2.5V

Upon testing, the oscillator could produce a good quality square wave with a frequency of about 1.2KHz

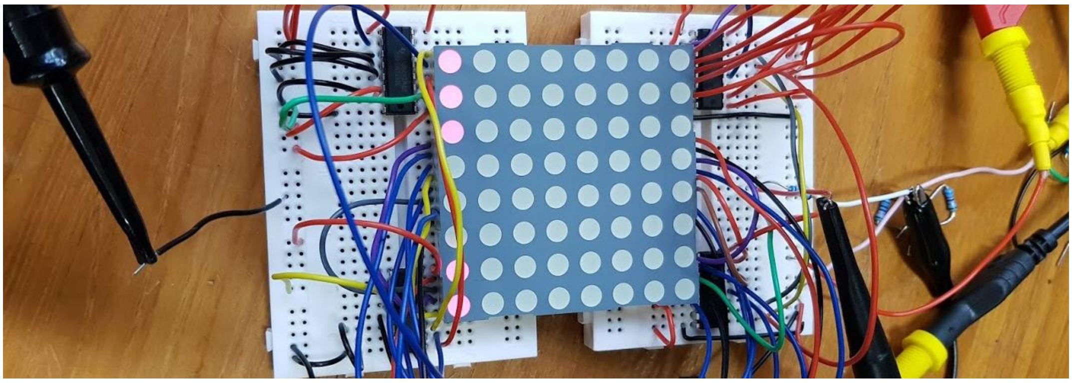


*Figure 50: The System is already capable of going through all 64 pins, one after the other.*

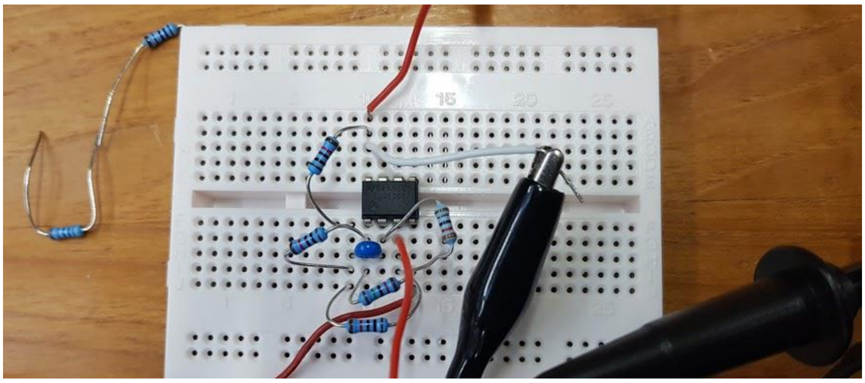


*Figure 51: The Clock*

So. How are images going to be displayed using the Arduino? Unfortunately, we have not been able to go very far before the date of submission of this report. One idea we have is to use an array of Booleans to store images, and interrupts to respond to the ticks of the clock: When the external clock ticks, the Arduino will send to the COM pin of the column demultiplexer the binary value of the next item in the array. Interrupts are a very handy tool that allows the Arduino to stop whatever it's doing when one of its digital pins changes state. After the interrupt's instruction is complete, the Arduino will return to the main loop.

Another possibility is to get rid of the external clock, and making the Arduino send a click, plus the next value on the array to the COM every few hundred microseconds. However, this cannot be done without modifying other parts of the code used now: As mentioned above, the current function to read square waves is programmed so the Arduino focuses exclusively on this task for 0.05 seconds every 0.5 seconds.

A pin of the Arduino might have to be used to reset the counters to 0 when starting to display pictures. Otherwise, there will be no way for the Arduino which LED the system is pointing towards at all time.

# 14. References


[1] J. Antonio, “Extension: Multitouch. You can touch concurrently zones in screen.,” 12 Mar 2017. [Online]. Available: https://groups.google.com/forum/#!topic/mitappinventortest/nOIgSpBkQ80.

[2] F. UK, “2901. - Speaker, Mini, 2 ", 500 mW, 8 ohm, 83 dB, 250 Hz to 10000 Hz,” 7 Jul 2017. [Online]. Available: http://uk.farnell.com/visaton/2901/speaker-miniature-k50-8ohm/dp/4662064.

[3] J. Wu, “Micro SD Card Micro SDHC Mini TF Card Adapter Reader Module For Arduino,” 14 Jan 2015. [Online]. Available: https://www.amazon.co.uk/gp/product/B00NNDBIRK/ref=oh_aui_detailpage_o00_s00?ie=UTF8&psc=1.

[4] Arduino, “Arduino Pin Current Limitations,” 2017. [Online]. Available: https://playground.arduino.cc/Main/ArduinoPinCurrentLimitations.

[5] A. A. Circuits, “Inductor Sizing Equation (Chapter 1 - Useful Equations And Conversion Factors),” All About Circuits, 2017. [Online]. Available: https://www.allaboutcircuits.com/textbook/reference/chpt-1/inductor-sizing-equation/ . [Accessed 29 Jan 2018].

[6] Vishay, “Silicon PIN Photodiode Datasheet,” 2015. [Online]. Available: http://uk.farnell.com/vishay/bpv22f/photodiode-50ma-silicon-pin/dp/1612301?ost=BPV22F&ddkey=http%3Aen-GB%2FElement14_United_Kingdom%2Fsearch. [Accessed 2018].

[7] A. microsystems, “Continuous-Time Ratiometric Linear Hall Effect Sensor ICs,” 2015. [Online]. Available: http://www.farnell.com/datasheets/1934044.pdf. [Accessed 2018].

# 15. Appendices

## 15.1. Appendix I: Pictures from the first brainstorming session

## 15.2. Appendix II: Records of the Meetings

### 15.2.1. Meeting 17/Jan/2018

**Radio Signals:**

- 2 modulation frequencies to filter (151Hz & 239Hz)
- Possible need of 2 antennas for 2 different carrier frequencies (67kHz & 103kHz)
- Tuned coil antenna for capturing radio signals (recommended by EERover tech-guide)
- Hertz test could be done to ensure the performance of the signal-receiving component.

**Filtering Circuits:**

- Designing filters by constructing circuits
- Problems with noise
  - Noise could be reduced or eliminated by using Fourier Transform.
- After filtering signals, amplify them using operational amplifiers.
  - Figure out what components are used in operational amplifiers **(Details are in Things to be done later)**.

**Things to be done later:**

- Making a voltage-frequency converter
- Research about which microphones to use
- Constructing H bridge
- Figure out what components are used in operational amplifiers.
  - Component types (Ex: resistors, capacitors and etc.)
  - Component values (Ex: resistance, capacitance and etc.)

**Research job assigned:**

- Daniel: Programming
- Constantinos: Magnetics
- Michael and Albert: Ultrasound and Radio Wave
- Alan: Infrared

### 15.2.2. Meeting 22/Jan/2018

**Radio**

- Choose an inductor with a resonant frequency, which is the same as the frequency to be measured.
- Rectification detects peak voltages.
- Amplification will be done using operational amplifiers.

**Magnetic field**

- Need a non-polarity-sensitive transistor that detects magnetic field.
  - It is some type of FET(Field Effect Transistor).
- The strength of magnetic field is unspecified.
- Not sure how sensitive the transistor should be.

## Infrared

- Amplification will be done using operational amplifiers.
- Carrier frequency doesn't need to match the actual frequency shown on the technical guide.
- Problem of phototransistors.
    - Directional sensitivity.
    - Ambient light.

## Programming

- Wifi is slow and unstable for now.
- Backup methods?
    - Bluetooth.
- Amplifying wifi signal may be done using antenna.

## Motor and powering

- How many wheels does the rover need: 4 or 2?

### 15.2.3. Meeting 31/Jan/2018

## Programming

- The application is done with the interface.
    - 4 gears (speed levels)
        - The speed of each speed level is not specified yet. They will be determined later.
    - D-pad is used for controlling the rover.
    - Analogue joystick is too complex to program and implement with the rover's components.

## Motor

- How do we smoothly turn the rover with 4 wheels?
    - We want the front 2 wheels to dictate the motion

## Ultrasound

- Prowave 400SR160 40kHz Aluminium Ultrasonic receiver
- https://www.rapidonline.com/prowave-400sr160-40khz-aluminium-ultrasonic-receiver-open-type-35-0175

## Magnetic

- Get 2 sensors – 1 sensitive and the other less sensitive.
    - This way we can figure out how far the each signal source is.

## Infrared

- AC to DC converter is required.
- Try to construct the circuit first without the receiver.

- TSOP4838 may be used.

**Radio**

- A coiled wire is needed
- Roughly 6.5 cm in diameter, 7 turns

**Assignment:**

- Constantinos: H bridge and Magnetic fields
- Alan and Daniel: Infrared
- Albert and Michael: Ultrasonic and Radio

### 15.2.4. Meeting 8/Feb/2018

**Magnetic**

- Magnetic sensor works well.

**Infrared**

- We need new infrared sensors.
    - BPV22F

**Motor and powering**

- 4 wheels
    - How to turn left and right
    - Sharp turning?

**Presentation (assessment)**

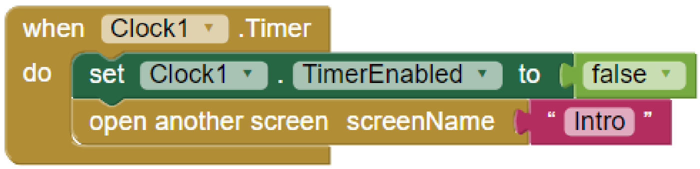
- What to include in the presentation
    - Responsibilities
    - How many times and when we have meetings
    - Communication methods among group members
    - Gantt chart
        - We have updated and been following it.
- Who presents what? (Might be changed later.)
    - Michael and Albert: ultrasound and radio
    - Daniel: wifi and app
    - Constantinos: magnetic and motors
    - Alan: Infrared
        - Mention TSOP4838, which contains an integrated circuit inside.
        - New infrared photodiode is called BVP22F.

### 15.3. Appendix III: Arduino Pin Map

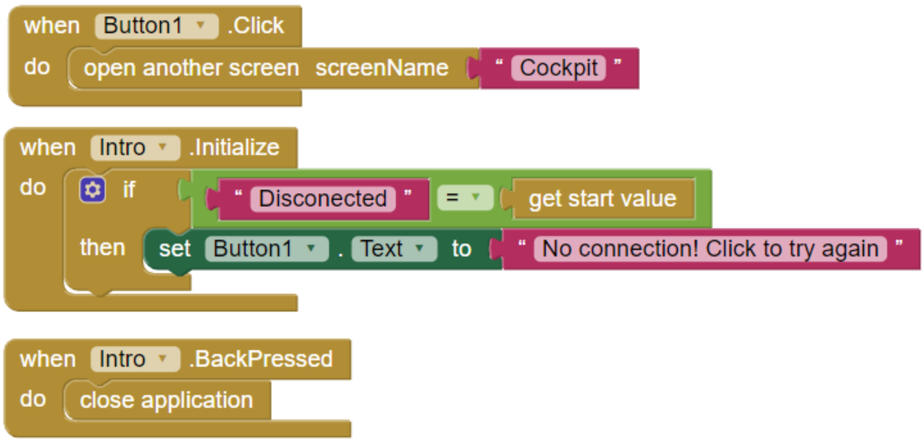
| **Pin** | **Function** |
|---|---|
| A0 | IR Iput |
| A1 | Radio Input |
| A2 | Ultra Sound Input |
| A3 | Magnetic Input |
| A4 | Available for LED matrix or others |
| A5 | Available for LED matrix or others |
| 0 | Bluetooth Shield TXD |
| 1 | Bluetooth Shield RXD |
| 2 | Left Motors Direction |
| 3 | Left Motors Speed |
| 4 | SD SD_CS |
| 5 | Right Motors Speed |
| 6 | Radio system MOSFET Gate |
| 7 | Right Motors Direction |
| 8 | Available for LED matrix or others |
| 9 | Sound Output |
| 10 | Available for LED matrix or others |
| 11 | SD MOSI |
| 12 | SD MISO |
| 13 | SD SCK |

## 15.4. Appendix IV: AppInventor Blocks

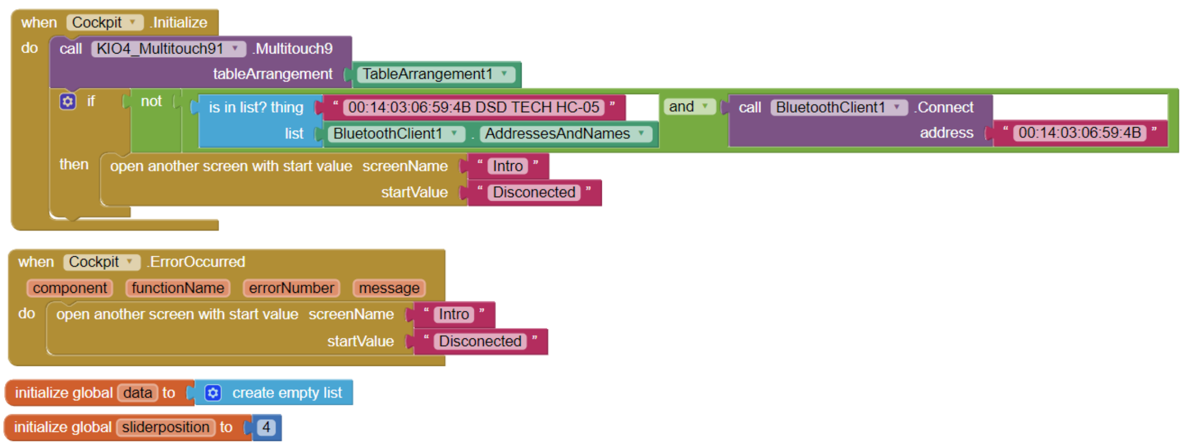
### Screen 1

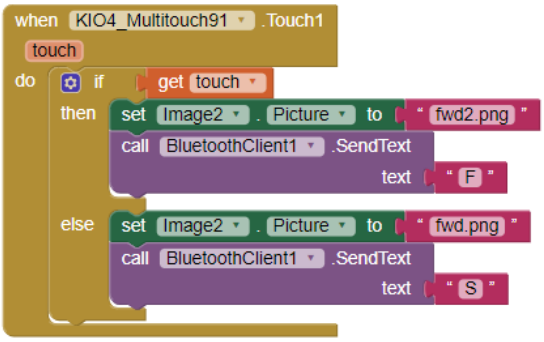


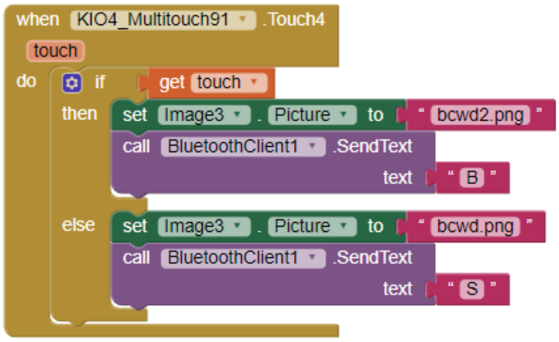

### Intro

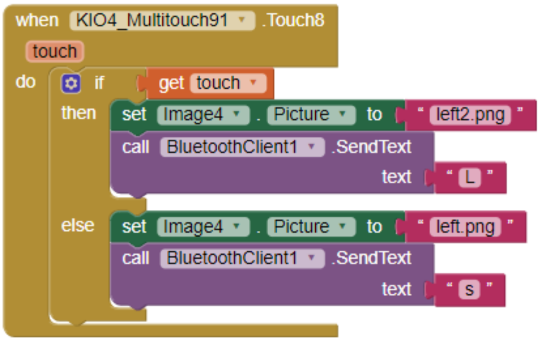


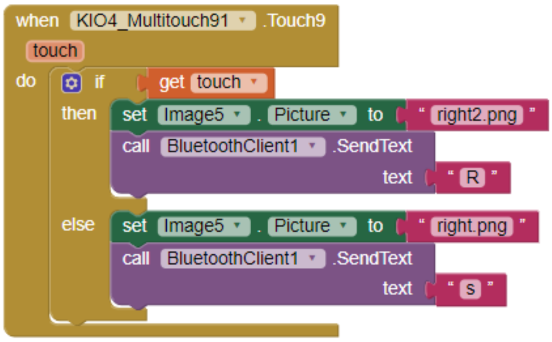

### Cockpit

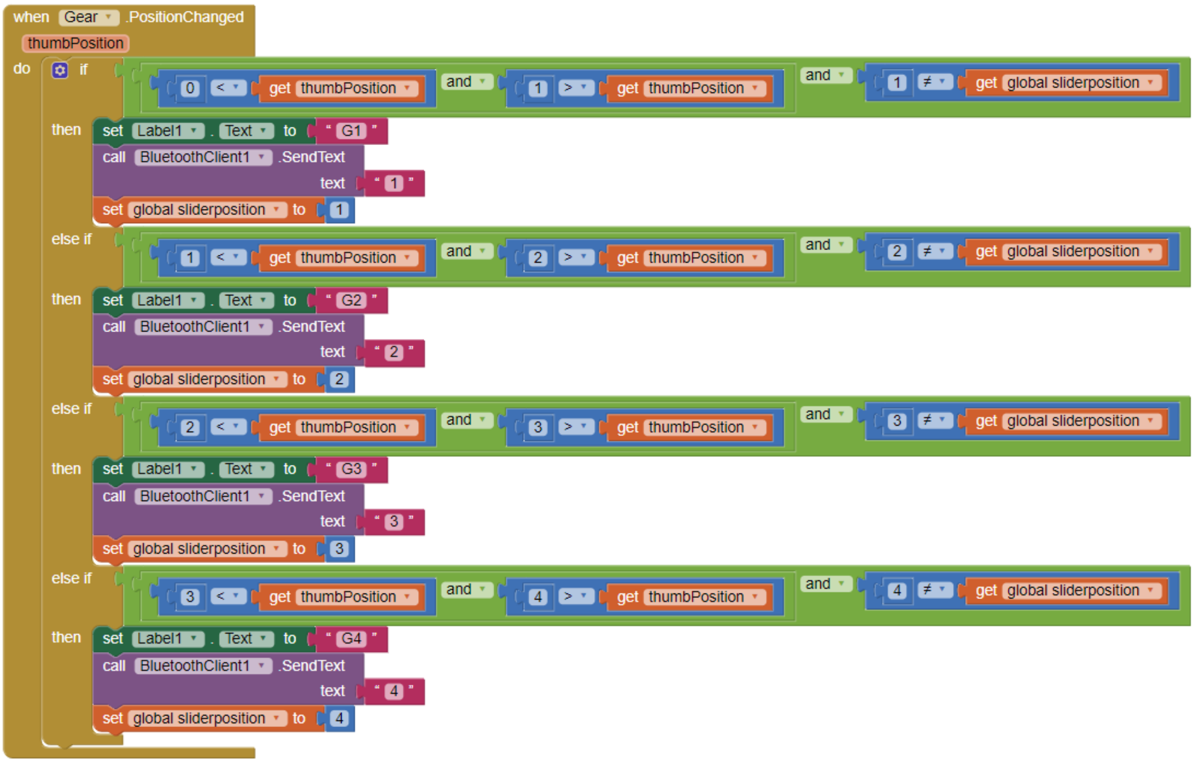



when KIO4_Multitouch91 .Touch1
touch
do if get touch
then set Image2 . Picture to " fwd2.png "
call BluetoothClient1 .SendText
text " F "
else set Image2 . Picture to " fwd.png "
call BluetoothClient1 .SendText
text " S "


when KIO4_Multitouch91 .Touch4
touch
do if get touch
then set Image3 . Picture to " bcwd2.png "
call BluetoothClient1 .SendText
text " B "
else set Image3 . Picture to " bcwd.png "
call BluetoothClient1 .SendText
text " S "


when KIO4_Multitouch91 .Touch8
touch
do if get touch
then set Image4 . Picture to " left2.png "
call BluetoothClient1 .SendText
text " L "
else set Image4 . Picture to " left.png "
call BluetoothClient1 .SendText
text " S "


when KIO4_Multitouch91 .Touch9
touch
do if get touch
then set Image5 . Picture to " right2.png "
call BluetoothClient1 .SendText
text " R "
else set Image5 . Picture to " right.png "
call BluetoothClient1 .SendText
text " S "


when Gear .PositionChanged
thumbPosition
do if 0 < get thumbPosition and 1 > get thumbPosition and 1 ≠ get global sliderposition
then set Label1 . Text to " G1 "
call BluetoothClient1 .SendText
text " 1 "
set global sliderposition to 1
else if 1 < get thumbPosition and 2 > get thumbPosition and 2 ≠ get global sliderposition
then set Label1 . Text to " G2 "
call BluetoothClient1 .SendText
text " 2 "
set global sliderposition to 2
else if 2 < get thumbPosition and 3 > get thumbPosition and 3 ≠ get global sliderposition
then set Label1 . Text to " G3 "
call BluetoothClient1 .SendText
text " 3 "
set global sliderposition to 3
else if 3 < get thumbPosition and 4 > get thumbPosition and 4 ≠ get global sliderposition
then set Label1 . Text to " G4 "
call BluetoothClient1 .SendText
text " 4 "
set global sliderposition to 4

## 15.5. Appendix V: Code

Please, go to https://github.com/danigr99727/code/blob/master/code.ino to find our Arduino code.

## 15.6. Appendix VI: How an Envelope Detector Works

The way the envelope detector works is the following. When the diode is forward biased, the capacitor charges and that follows the rising edge of the signal. When it is reverse biased; when the voltage falls, the resistor drops this voltage as the capacitor discharges. This action occurs continuously and performs the envelope detection of the signal. The time constant of the envelope detector has been adequately chosen to be significantly larger than 1/Wc (for Wc=2*π*carrier frequency), because otherwise the output would have still contained a little bit of the original signal. But also, it has been experimentally chosen to be small enough to avoid the capacitor taking too much time to discharge and give an undesired output. The final values used are: a capacitor of 36nF and a resistor of 1kΩ, the time constant is approximately times 10 greater than 1/Wc.