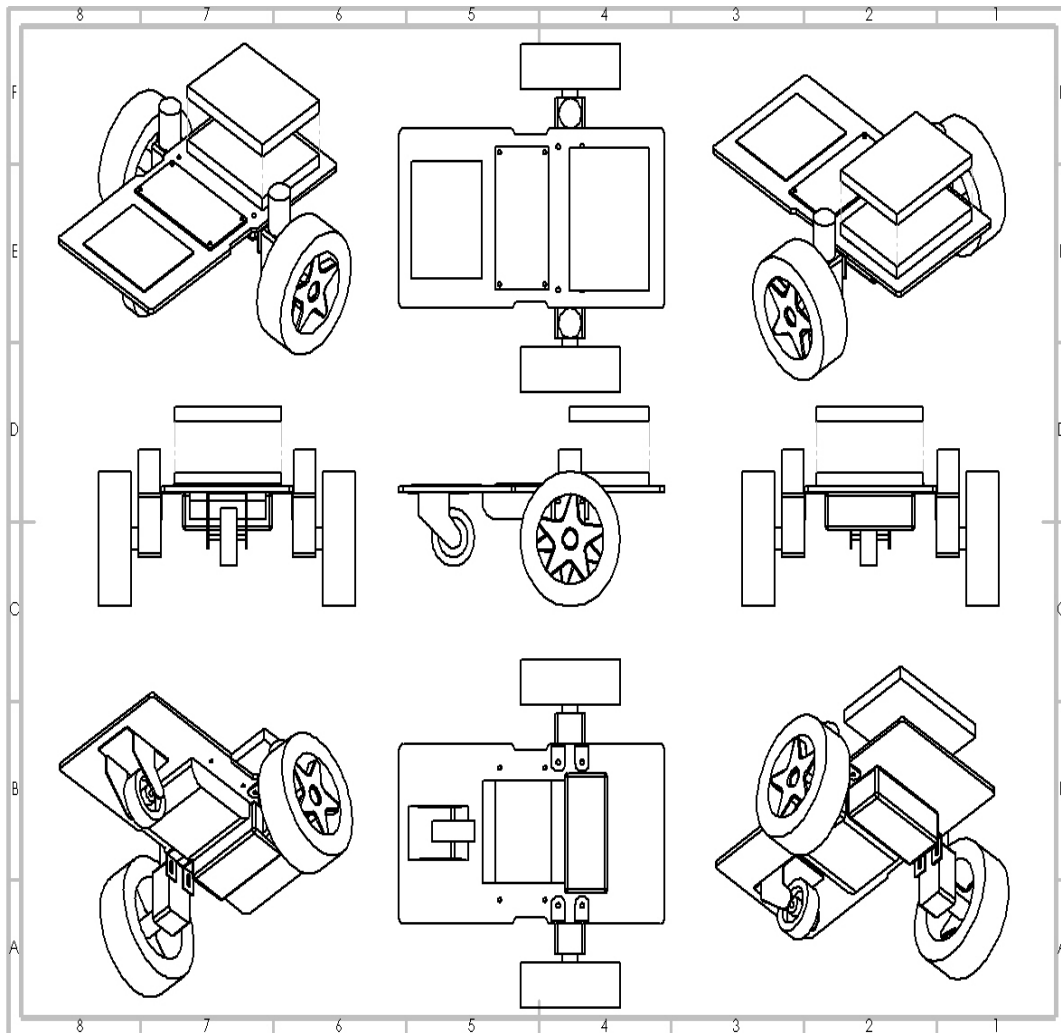


# Rover Project Report



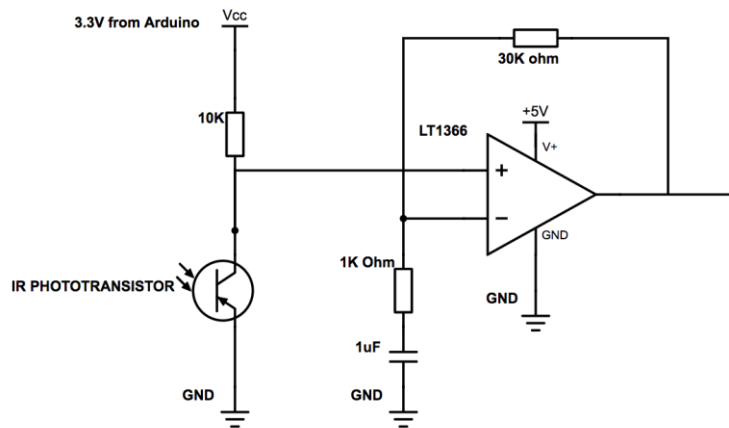
Team Indium

Electrical and Electronic Engineering  
Imperial College London

# Table of Contents

- IR phototransistor circuit
- Radio frequency detector circuit
- Wireless Communication
- Power
- Propulsion & Steering
- Chassis
- Budget & costings
- Project management
- Discussion & conclusion
- Appendices
- References

# IR Phototransistor Sensor Circuit

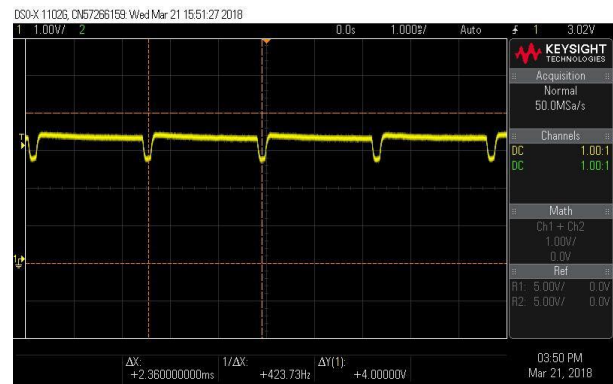
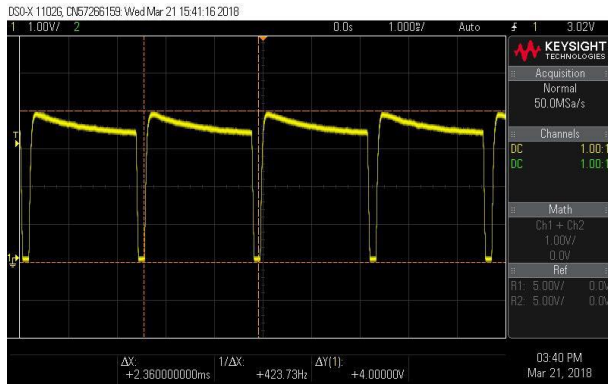


This circuit is used to detect the IR radiation from the Exoflask, the IR phototransistor's data sheet can be found in the Data Sheet section at the end of the report.

The 3.3V output pin from the Arduino is chosen deliberately as the analog input pins on the Arduino can only read between the range of 0 – 5V, therefore 3.3V would be more suitable to use as  $V_{cc}$  than 5V.

The 10k $\Omega$  resistor acts as a potential divider, when there is no IR radiation being detected, the output from the LT1366 op-amp will be constantly at around 3.3V. However, under the environmental conditions in the lab, there will be some background light/ radiation that reduces this voltage slightly. In regard to this problem we wrapped the head of the phototransistor with black paper to block out the visible spectrum of light that would interfere with the reading. When the IR signal is incident on the phototransistor, the op-amp amplifies it with a gain of 30. The resulting square wave can be clearly seen in the oscilloscope and the frequency is accurately calculated by the Arduino.

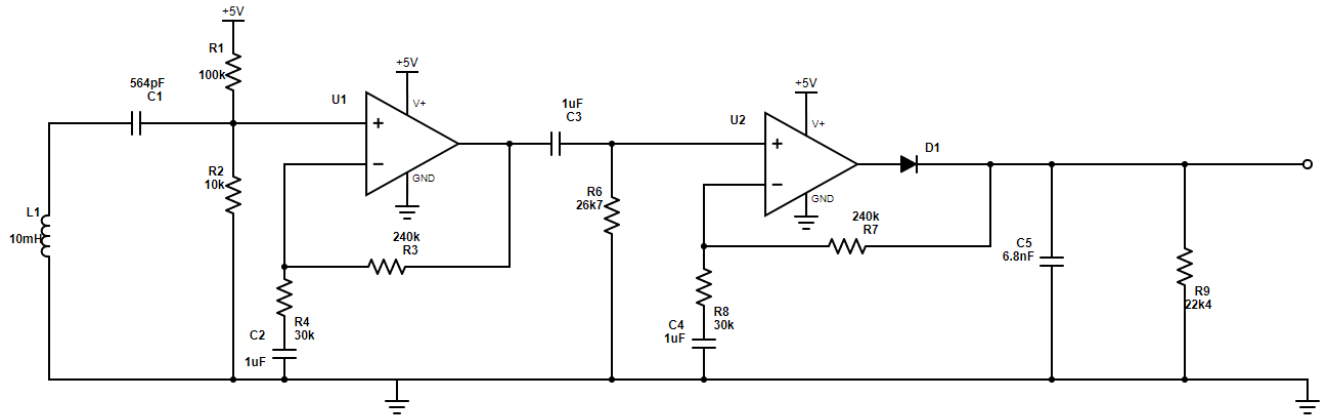
There is a 1 $\mu$ F capacitor connected with the feedback resistor and the ground which acts like a switch, when no signal comes is present it will act like an open circuit and therefore the op-amp behaves as a voltage follower with unity gain of 1, otherwise the output will be saturated at 5V.



On the right hand side is the waveform that the output terminal of the circuit displays when the Exoflask is 15cm away from the phototransistor with the phototransistor ideally orientated towards the IR emitter. The waveform on the left shows the output terminal of the circuit with the phototransistor non-ideally orientated towards the IR emitter. While the waveform is distorted and not perfectly square, the Arduino is still able to detect the wave and calculate the frequency.

# Radio Frequency Detector

**Figure 1:** Circuit to detect radio waves with 67KHz carrier frequency



**Figure 2:** Circuit to detect radio waves with 103KHz carrier frequency

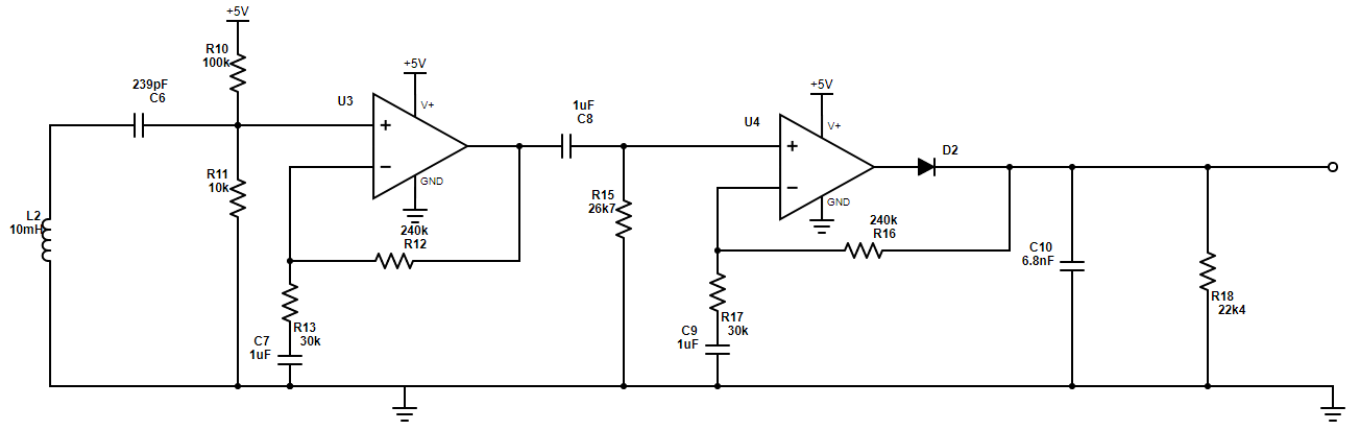
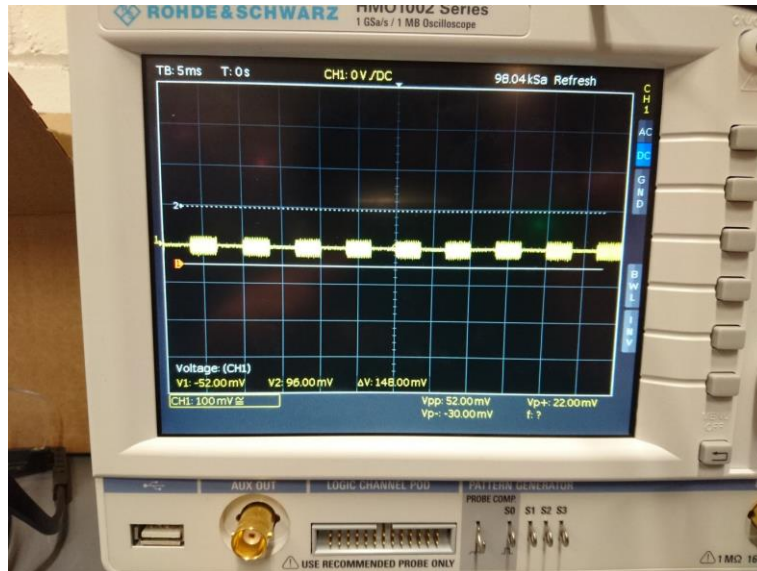


Figure 1 and Figure 2 display the circuit which will be used to detect the radio waves emitted by the Exoflask. The operational amplifiers are from a single MCP6004 chip, which contains four of these operational amplifiers. The benefit of using the MCP6004 chip is that it is very cheap, costing only 39p, but has the benefit of each operational amplifier having a GBP (Gain Bandwidth Product) of 1MHz, a slew rate of 0.6V/ $\mu$ s and requires 1.8V to 5.5V power supply which is ideal considering both the Arduino Uno and the PCB can supply up to a maximum of 5V<sup>[4]</sup>.

The inductor in series with the capacitor C1 or C6 forms our LC resonant circuit that resonates at the frequency of the modulated frequency, 67KHz or 103KHz. The angular resonant frequency,  $\omega$ , is given by  $1/\sqrt{LC}$ . For 67KHz and 103KHz, the capacitors should be 564pF and 239pF respectively. Figure 3 shows the waveform obtained when we use the 10mH inductor with the 564pF capacitor:



**Figure 3: Detection of modulated radio wave from Exoflask**

The resonant circuit is then connected to a cascading single supply non-inverting amplifier circuit. The resistors R1, R2, R10 and R11 are a potential divider circuit that sets the biasing such that there is sufficient bias current to use the operational amplifier, which at room temperature requires  $\pm 1\text{pA}$ .

In both Figure 1 and Figure 2, after the first amplifier, the capacitor is used to remove the DC component of the signal. This will be important for demodulation. However, we then need to connect a resistor from the non-inverting input of the second operational amplifier to ground. This is to provide a DC pathway for the input bias current of  $1\text{pA}$ . If we did not include this, the coupling capacitor at the input would charge up, until it reached or exceeded the maximum common-mode voltage rating of the operational amplifier. When this happens, the operational amplifier will no longer properly function and may not even function at all. It is a good rule of thumb to set the resistor that creates the DC path to be about equal to the parallel combination of the resistors  $R_f$  and  $R_g$ , where  $R_f$  is the resistor connected directly to the output of the operational amplifier and  $R_g$  is the resistor that is connected to  $R_f$ .

Resistors R3, R4, R7, R8, R12, R13, R16 and R17 are responsible for setting the gain of the operational amplifier. Since each amplifier block is in non-inverting amplifier configuration, the gain is given by  $1+(R_f/R_g)$ . In this current configuration, the gain for each amplifier is  $\times 9$ . This enables our rover to detect the source from a distance further away. Figure 4 shows how much longer the range is:



**Figure 4: Maximum range for detection of radio wave**

There are two reasons for which we have chosen  $R_f$  and  $R_g$  to be within the  $\text{k}\Omega$  region. The first is that, we want to avoid using large resistors as these can increase sensitivity to interference and if the resistances are too

small, we have very large currents that can exceed the limit of the operational amplifier's supply current. So, the circuit will not work. The second reason is that the capacitor connected to  $R_g$ , has impedance. For the impedance to not affect the gain significantly, the resistors must be about a hundred times bigger.

The capacitors at the non-inverting input terminals of the operational amplifiers are there to block any unwanted DC offset, and the capacitors that connect  $R_g$  to ground are there to ensure that the DC bias we have set, will not be amplified. This is important because if the DC offset is amplified too much, the operational amplifier may alter the signal that we want to demodulate (extract information from) or it may hit a supply rail and just saturate, which is no better.

The values of the resistors and capacitors at the non-inverting input of the operational amplifiers are not random. In Figure 1, the capacitor and the resistors form a high-pass filter that attenuates signals with frequencies of 31KHz. In Figure 2, the high-pass filter attenuates signals with frequencies of 73KHz. The significance is that while we can bias the input, we can also filter out unwanted signals. The circuit in Figure 2 takes advantage of this by being able to filter out the other carrier frequency of 67KHz.

The reason behind the decision to use cascading amplifiers is that a single amplifier block does not yield a sufficient gain for the signal. Since that the GBP of a single operational amplifier is only 1MHz and the largest frequency that we want to amplify is 103KHz, the maximum voltage gain a single operational amplifier can give is 9.7. If the amplification is not high enough then the range at which we can detect the signal will be too small to be of any practical use.

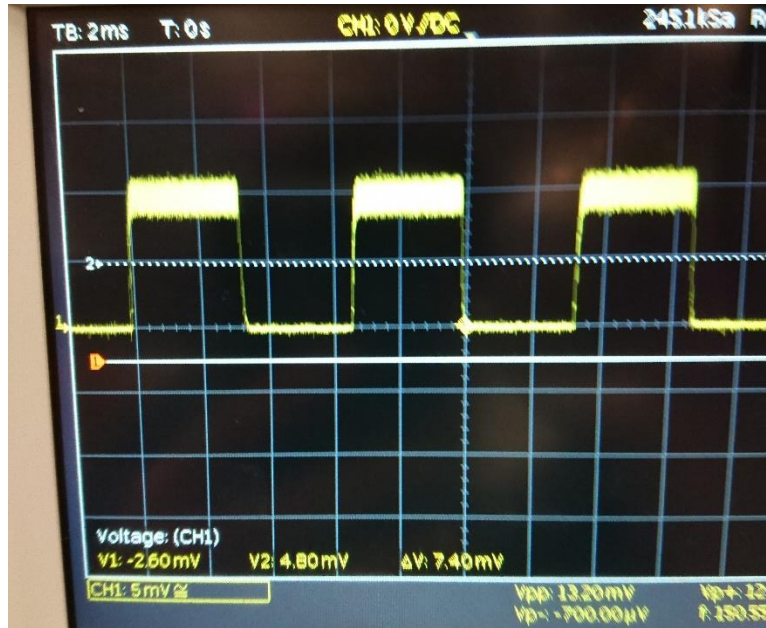
The final amplifier block is a precision rectifier with gain. The reason for this is twofold, the first is that we conserve more space on the breadboard, use less components and so reduce the manufacturing cost of the rover. The other reason is that we can now eliminate the 0.7V drop across the diode and in doing so would have a greater detection range.

The precision rectifier with gain is a part of the demodulating circuit, which extracts the useful information from the modulated radio wave. The diode removes the negative voltage portion of the radio wave and only the positive voltage portion of the wave is passed through. The capacitor that is connected to the diode will then charge up until it hits the amplitude of the signal that is being passed through.

However, when the voltage of the signal being passed is less than the charge on the capacitor, the diode switches to reverse bias and current cannot pass through the diode. The diode in reverse bias will have some leakage currents, but it is negligible and can be thought of as an open circuit. The capacitor is connected to a resistor, which the current will flow through as the capacitor discharges. At the same time the voltage of the signal rises as the capacitor is discharging. When the voltage of the signal and the capacitor are equivalent, the capacitor stops discharging. The voltage of the signal continues to rise, causing the capacitor to charge again until the amplitude of the signal is reached. This cycle of charging and discharging, can extract the modulating wave from the signal with fidelity.

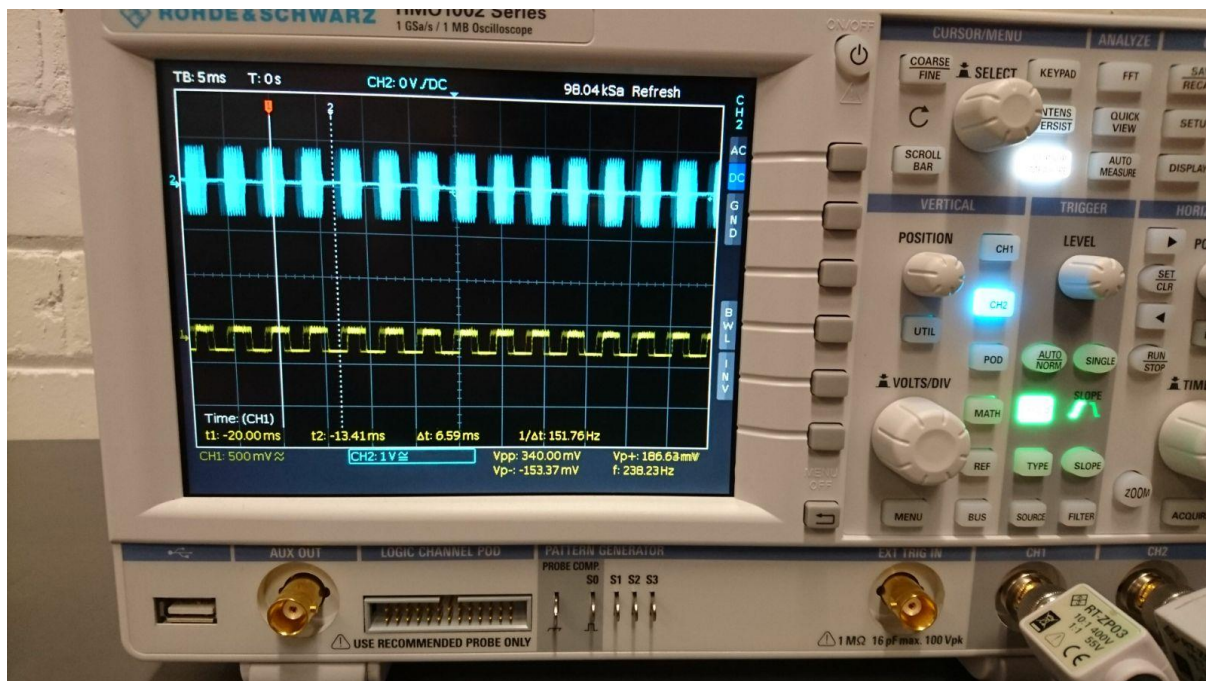
How quickly the capacitor charges and discharges depend on its capacitance and the resistance of the resistor it is connected to. The time constant  $\tau$ , is equal to  $RC$  and is the time required for the capacitor to discharge 67% of its full voltage or to charge up to 37% of its full voltage. This is important because if the capacitor charges or discharges too slowly then, we will not have a waveform that resembles a square wave. If the capacitor charges up or discharges too quickly then, there will be many ripples in the demodulated signal. However, this is not such an issue as the wave does resemble the modulating waveform and the cut off frequency of the Resistor-Capacitor circuit in the demodulator can accommodate both the 151Hz and 239Hz modulating frequencies. Figure 5 clearly shows that this is not an issue:





**Figure 5: A high time constant for demodulating a square wave**

Figure 6 shows the modulated signal (in blue) and the demodulated signal (in yellow).



**Figure 6: Detection at 12cm away from Exoflask**

The demodulated signal will then be fed into either pin 11 and pin 13 for carrier frequencies, 67KHz and 103KHz respectively. This way we can distinguish the four different radio waves as for each carrier frequency, there are two modulating frequencies. The Arduino Uno WiFi utilises a program to measure the frequency of square waves that is explained in the code section of the report.



## Outstanding work

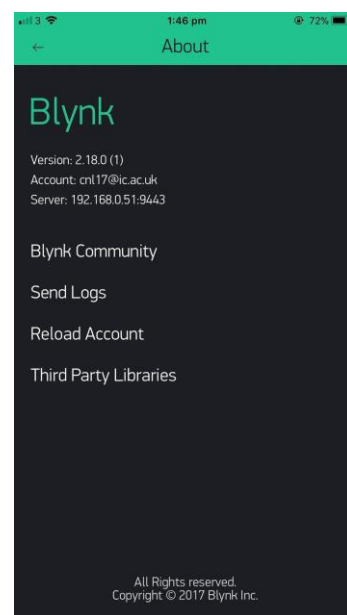
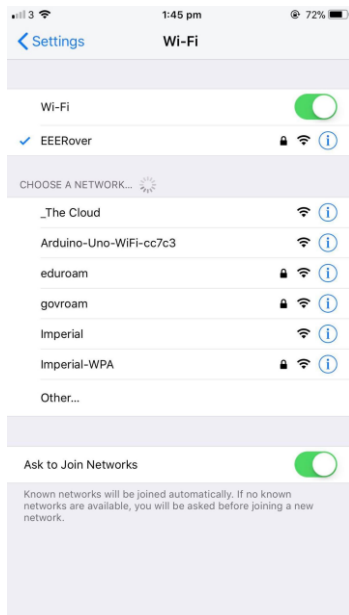
Initially, it seemed desirable to build a single supply Schmitt trigger and have the output of the demodulator circuit be the input to the Schmitt trigger. The reason for this is that the Arduino Uno can only take in 0 – 5V without causing any internal damage. Alternatively, we may just reduce the total gain of the cascading operational amplifiers from x81 to x60. The latter is cheaper than the Schmitt trigger and takes up less space on the rover.

# Wireless Communication

Blynk is the most popular platform with both an iOS and Android app to allow remote control of Arduinos, the app allows you to build a graphic interface on a digital dashboard by simply dragging and dropping widgets which can modify variables or read variables on an Arduino. It has a lots of useful widgets and tools which are very suitable for the rover's implementation of the Arduino Uno WiFi (and in general with most Arduino boards).

One can either connect automatically through the Blynk-Cloud Server or by setting up a personal server at home and entering the corresponding address and port number in the app when signing up. After that a unique “Authorisation Token” is generated, which acts as verification during the connection between the app and the Arduino at the time when it is being connected so that only the user can interact with their board.

These are the following settings that we used in order to connect to the Blynk server hosted on the EEERover network in the lab:

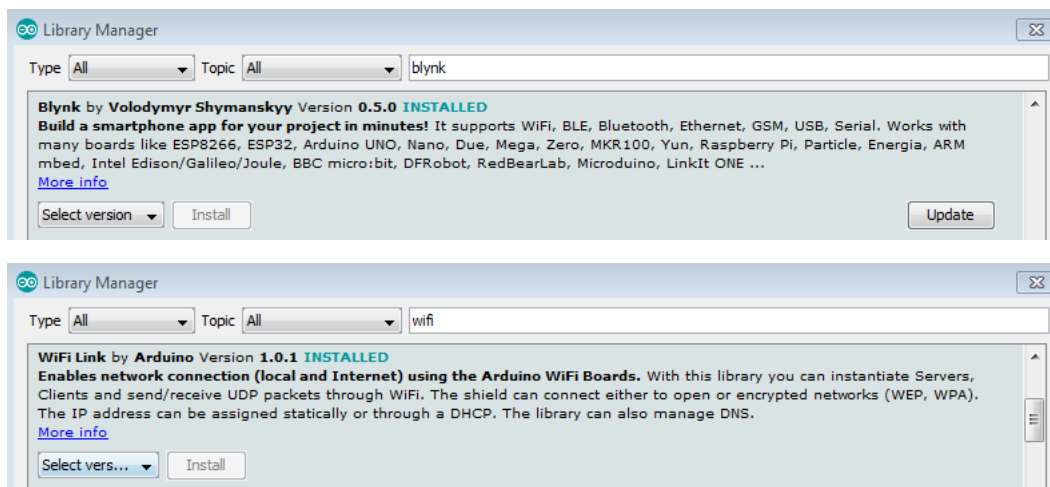


First of all, we connect to the EEERover WiFi network in the laboratory and change the Server IP and Port in the app settings. Then we double check in the Blynk app that we connected to the correct testing server and port. (This is the general case for different mobile phone operating systems.)

The local Blynk server on the EEERover network has been allocated the IP address of 192.168.0.51 so the Arduino connects to this when the script on it is initialised, and port 9443 indicates which port the server communicates on, hence the phone app & the Arduino must direct their traffic to this specific port as the server is only 'listening' for requests on this port, distinguishing Blynk communications from other sources that the server may receive data from.

## Program with Blynk and its library

Before explaining the use of the code and libraries, the user needs to download the WiFi Link (by Arduino) Version 1.0.1 and the Blynk Version 0.5.0 (or above) libraries, which can be done in the Library Manager in the Arduino IDE shown below.



In order to establish a connection between the Arduino and the server, the following shows the libraries used and their properties.

```
#define BLYNK_PRINT Serial
#include <SPI.h>
#include <WiFiLink.h>
#include <BlynkSimpleWiFiLink.h>
```

**#define BLYNK\_PRINT Serial**

This is used in testing to let the developer/user know whether the Arduino is connected to the destination server/network. Status of the connection and 'Blynk' will be printed in the Serial Monitor once connection is made. It streamlined the debugging process for us, allowing us to pinpoint whether any issues we were having were related to the connection to a network, the Blynk server or it was something else in our code/circuits.

```
#include <SPI.h>
```

Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances, this library allows us to communicate using this protocol, which is preferred to other protocols such as UART due to the increased data rate. [reference]. The details of this library can be found on the Arduino Website: [g](#).

```
#include <WiFiLink.h>
```

For this project we were given the Arduino Uno WiFi Board, which is a big but significant difference compared to the Arduino Uno with a separate ESP8266 WiFi Shield, or other Arduino boards with a WiFi Shield connected. WiFiLink enables network connection (local and Internet) using Arduino WiFi boards. The properties of the Arduino Uno WiFi Board will be introduced in the Arduino section.

```
#include <BlynkSimpleWiFiLink.h >
```

```
BLYNK_WRITE(Virtual Pin Number){ }
```

Inside the app it allows to set up which virtual pin the widget will correspond to. Blynk uses the code inside the BLYNK\_WRITE{ } to manipulate what variables the widget will modify or which variables will be read by and displayed by the widget.

```
param[].asInt();
```

This is a member function inside the Blynk library and it is used to get the values from the widgets in the app. It is used inside BLYNK\_WRITE and with specific virtual pin number put into the square bracket, corresponding to the ones in the app. More detailed and its actual implementation will be explained in the Arduino programming section.

```
Blynk.begin(auth, ssid, pass, IPAddress() );
```

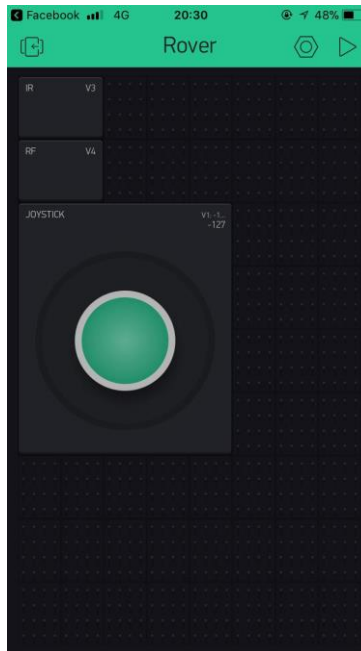
Necessarily there must be some function or code to get the Arduino to connect to a WiFi network with a particular IP Address. 'auth' is the token which the users are given after they successfully signed up an account, while 'ssid' and 'pass' are just the network name and password of the network. And the specific socket address on the server can just be put inside the IPAddress field.

```
Blynk.run();
```

This is a function which is embedded inside the Blynk library. It has to be put inside the void loop(){ } function to make sure that the Arduino is constantly communicating with Blynk app. There should be no delay() function running with it.

# Arduino Programming

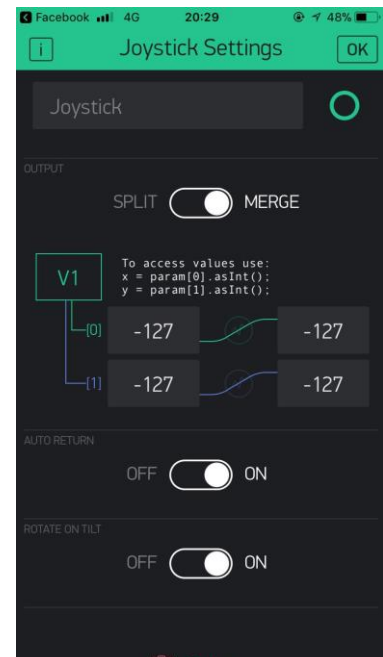
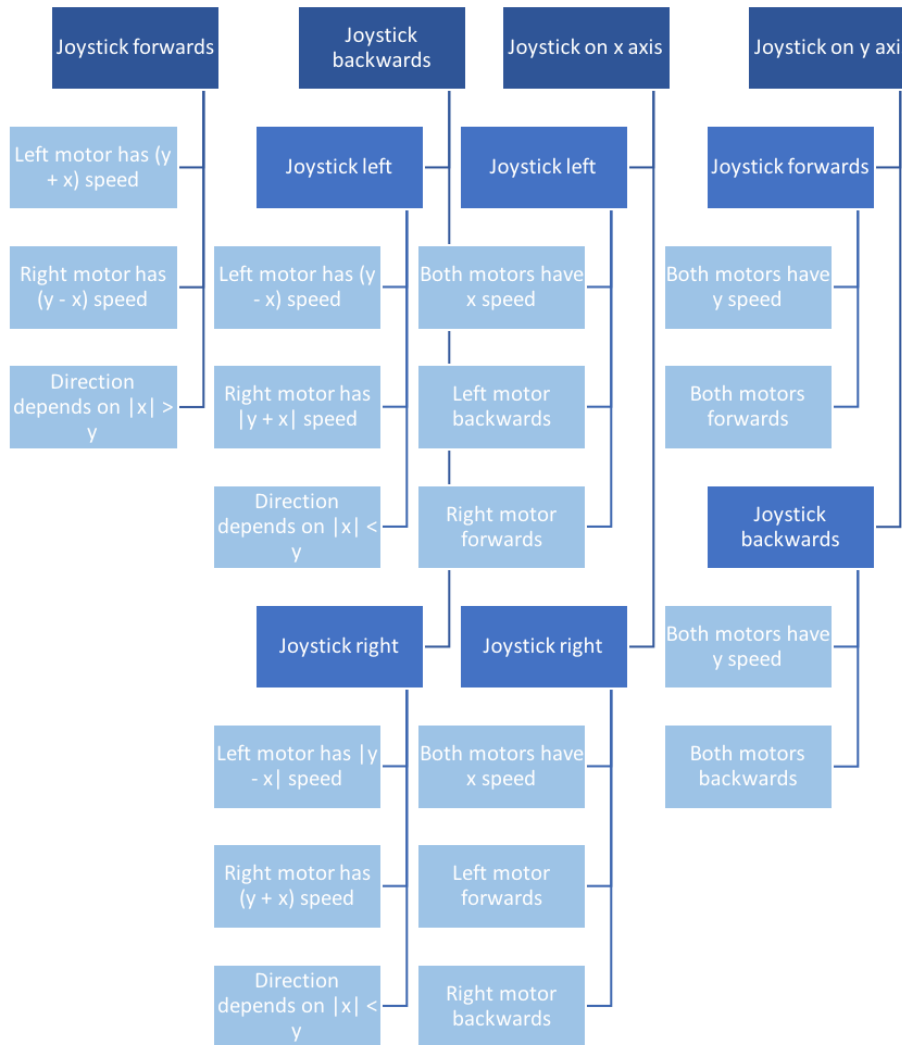
Even though we are using Blynk, we must think of the ways to detect and take measurements of the incoming data that controls the rover. We applied some functions which are already in the Arduino's default library and thought of the code to calculate the frequencies of the signals received. The code is in the Appendix.



## Rover Manipulation

Previously we mentioned those important libraries and functions from Blynk, one of them is `param[].asInt()` and it is implemented with `BLYNK_WRITE` with specific virtual pin written into it.

The joystick that we use to manoeuvre the rover outputs two variables, x & y which range from -127 to 127. The following flowchart briefly illustrates the overall purpose of the code corresponding to the joystick inputs, and therefore the overall movement of the rover.



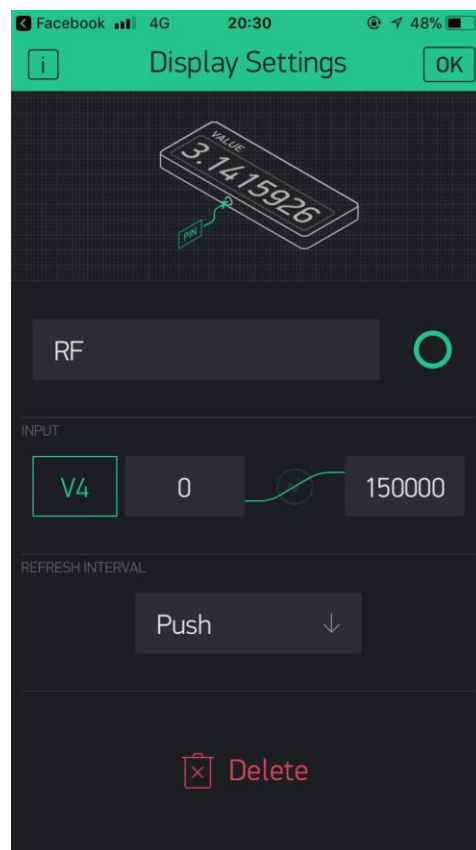
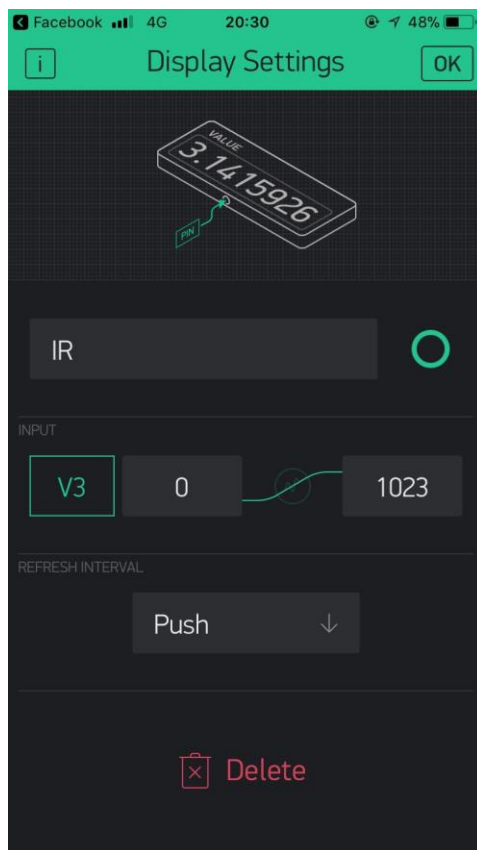
## PulseIn() Function & Frequency Calculation

The *pulseIn()* function measures the time that an incoming signal is *HIGH* or *LOW*. For example, if we wanted to measure how long it was *HIGH*, the function measures the time between when the incoming signal crosses the *HIGH* voltage threshold and when the signal crosses into the *LOW* voltage threshold. This can be used to measure the frequency of square waves as measuring the time that one wave is *HIGH* and *LOW* tells us the time period, regardless of the duty cycle of the wave. Then it is elementary to calculate the frequency.

We declare 6 unsigned long variables which are a, b, f, a2, b2 and f2 to help with the calculation of the frequencies in the code, with (a, a2) and (b, b2) measuring the time that the wave is *HIGH* and the time that the wave is *LOW* respectively. Since the function outputs the time in microseconds, we must multiply the time by a factor of 1000000 so our calculated frequency is in Hz.

## Sending data back to the app

Using “value displays” on the Blynk dashboard, we can send data back from the Arduino to the phone app. To implement one, all that needs to be done is to assign the variable that we wish to send back (i.e. the calculated IR and RF frequencies) to one of the virtual pins associated with the value displays. Then when the app is running and the Arduino is connected to the Blynk server, the values in each display are updated in real-time.



# Power

## High Level Design

There were two paths we could have taken with the power system given that, as a team, we had decided to re-use the EEBug PCB for power distribution. We either could retain the AA battery system that was already in place or move to a larger, rechargeable battery if the AAs were not capable of powering all of the rover's functions.

## Design Process

At first we researched the typical power consumption of the motors, Arduino and the op-amps we intended to use to see how long the batteries we were already using could support them. Our estimations were:

- 0.49W for the Arduino
- 6W for the motors
- 0.9W for the amplifiers in the sensor circuitry

As typical AA batteries can store around 9,000J of energy<sup>[3]</sup>, then the existing 4 AA batteries could support the rover for 80 minutes, which we concluded was sufficient for our needs.

However, the Arduino supply input  $V_{in}$  supports 7-12V, not the 5V that the EEBug PCB supplied, and connecting directly to the 5V pin would have bypassed the built-in voltage regulator<sup>[2]</sup>. Therefore we decided that it would be safer if we used an auxiliary power supply that connected to the Arduino through the DC power jack. Since there was ample space underneath the chassis to support it, we decided that a 9V battery in a battery holder could be mounted underneath the centre of the chassis.

## Final Design

The final power supply configuration carries over the EEBug power distribution PCB and batteries, whilst also adding a dedicated 9V battery to power the Arduino through the DC power jack. The 9V battery is kept in a battery holder that is taped to the underside of the chassis, between the underside of the PCB and the caster wheel.



# Propulsion & Steering

## High Level Design

There was a lot of early discussion on the method of propulsion that was to be used.

Our initial position was the existing EEBug layout: two powered wheels at the front with a stump at the back to support the rest of the chassis. Differential steering/torque vectoring for steering.

The initial modifications proposed around the time of the client meeting were:

- Two powered wheels at the front, one central caster wheel at the back. Steering with torque vectoring.
- Two wheels at the front, two wheels at the back.
  - Front wheels powered.
    - Steering by torque vectoring.
    - Steering by turning the front wheels (using servo motors/actuators).
    - Steering by turning the rear wheels (using servo motors/actuators).
  - Four wheels powered.
    - Steering by torque vectoring.
    - Four wheel steering (all 4 wheels pivot using servo motors/actuators).
    - Steering by turning the front/rear wheels (using servo motors/actuators).
  - Rear wheels powered.
    - Steering by torque vectoring.
    - Steering by turning the rear wheels (using servo motors/actuators).
    - Steering by turning the front wheels (using servo motors/actuators).

The idea that new motors should be purchased that can operate at higher speeds was also brought forwards, as this could allow us to accelerate and turn faster.

## Design Process

Initially we decided to delay our decision until we had finalised the chassis layout, as well as what payload the rover would be carrying because at the time we had no knowledge about the form of the sensors or sensor circuitry. After concluding that the EEBug chassis was sufficient for our rover, we decided that adding extra motors wasn't required.

The EEBug chassis and sensor circuitry were light enough so that the existing EEBug motors could be re-used. We decided that servo motors were unsuitable for the steering, therefore we decided to use differential steering to the rover. This meant that having a chassis with two additional fixed rear mounted wheels would increase the size of the turning circle, hence the optimum choice would be to use a single, centrally mounted, free-spinning wheel installed at the rear: such as a small caster. Replacing the stump at the rear of the chassis with this would boost the ride height and additionally reduce the frictional force that limited the top speed.

# Final Design

The Arduino receives two variables from the Blynk app, the x and y positions of the joystick, where both x and y range from -127 to 127 and they are centred at 0.

A function for controlling servo motor angle was written in the Arduino IDE and successfully tested, however it was not employed as we decided that servo motor steering would not be practical. This was due to the resulting power and chassis layout considerations, despite the fact that they provided more precise steering control and could facilitate steering at higher speeds (the existing EEBug motors do not vary their speed continuously when the voltage is varied, it has more discrete behaviour so use of differential steering is quite imprecise). Servo motor steering would have also separated the code for steering from the code for motor speed which would have greatly simplified the code structure as well as reducing the response latency.

The final design also incorporates the motor driver provided to us in the starter kit, which is fitted in the designated slot in the EEBug PCB. The code above returns four variables, two which range from 0 to 255 which correspond to the left & right motor speeds and two which are either *HIGH* or *LOW* which correspond to the left & right motor directions. The motor speeds are determined by the duty cycle of a PWM signal, so the Arduino's *analogWrite()* function is used to produce this and these are input into the *PWML* and *PWMR* pins of the motor driver, and the *digitalWrite()* function is used to control the inputs to the *DIRL* and *DIRR* pins of the motor driver.

The EEBug motors were retained and are connected to the EEBug PCB as before and the stump at the rear is removed and replaced by a caster wheel which is mounted underneath the Arduino on the chassis with screws.

# Chassis

## High Level Design

Before the client meeting, the propositions for modifications to the chassis were:

- The EEBug breadboard should be reused for the sensor modules.
- An extra shelf of Perspex above the existing chassis may be needed if the components prove too large to fit on the original EEBug chassis.
- The existing EEBug wheels provide far too little traction. Wider and/or larger wheels are required.
- The overall ride height should be raised as there may be difficulties navigating rough terrain/obstacles.
  - The existing motors can be rotated 90° to raise the distance of the chassis from the ground.
  - Larger wheels can also be used to raise the distance of the chassis from the ground.
- A bespoke chassis can be manufactured from Perspex to accommodate for all of these considerations.

# Design Process

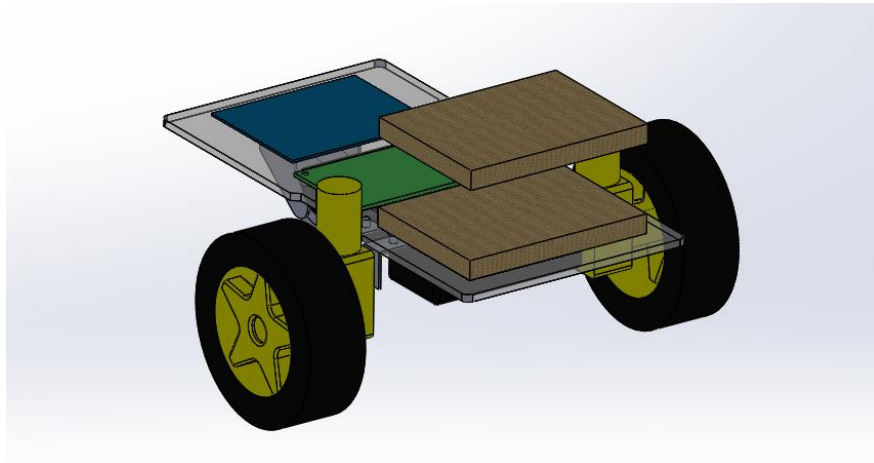
As the requirements for other areas of the project were unknown, such as the area required to mount all of the sensors, it was difficult to make a decision as to whether we needed a custom chassis or if the EEBug chassis would suffice. So we decided to continue as if we were using the EEBug chassis and if we encountered any issues with that decision in the future, we would re-evaluate our options at that point.

There was unanimous agreement that wheels with a larger diameter and width would be preferential to the existing EEBug wheels, as they would be unstable at high speeds. We also found rotating the motor assembly by 90° was an achievable task that gave us a significant amount of ground clearance

Later on in the spring term, once the sensor circuitry was prototyped and tested, we realised that we wouldn't be able to fit both the RF and the IR circuitry on the breadboard that came with the EEBug. Hence we discussed some ideas that we could implement in order to mitigate this issue:

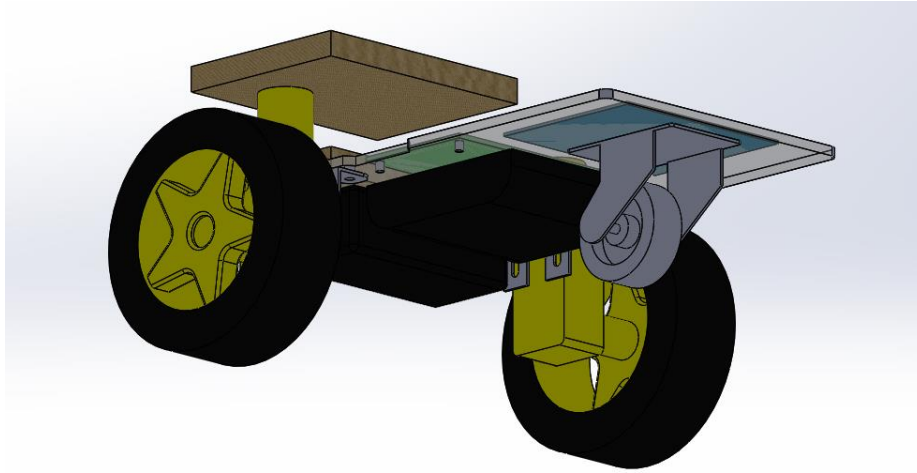
- Quickly design and laser cut a larger chassis that could accommodate a much larger breadboard.
- Design an additional Perspex 'shelf' that would be mounted on the EEBug chassis and sit above the PCB and Arduino so that there was extra mounting space for another breadboard.
- Design or purchase brackets that allow a second, identical breadboard to be mounted above the existing one, so that they are stacked above one another.

In the end we decided that the brackets were the best design choice as they minimised the added weight and meant that time did not need to be diverted away from the more crucial aspects of the rover. As is discussed later on in the Propulsion & Steering section, a single caster wheel was mounted on the rear of the rover to replace the stump that was carried over from the EEBug.



## Final Design

The final chassis design is mostly identical to the original EEBug chassis, the only modifications made are the installation of larger drive wheels, rotated motors, a caster wheel mounted at the rear and a secondary breadboard mounted on brackets placed on each corner of the existing breadboard.



## Budget & Costing

From the £50 budget, £14.82 was used to order 6 components.

|   |                                    |                            |                     |                         |                     |
|---|------------------------------------|----------------------------|---------------------|-------------------------|---------------------|
| EErwig 2015-2018 - Amine.Halimi ver 4.2 |                                    |                            |                     |                         |                     |
| <b>Project group</b><br>Indium          | <b>Your Full Name</b><br>Lam Chun  | <b>Your role</b><br>Leader |                     | <b>Total orders :</b> 6 | <b>New :</b> 0      |
| <b>Total Budget</b><br>£50              | <b>Total Expenditure</b><br>£14.82 | <b>Balance</b><br>£35.18   | <b>Members</b><br>5 | <b>Approved :</b> 6     | <b>Rejected :</b> 0 |

## Components Ordered

### 1. IR Phototransistor x 1

Osram Opto SFH 309 FA-4/5 24 ° IR Phototransistor, Through Hole 2-Pin 3mm (T-1) package

RS Stock No.: 654-8019 | Mfr. Part No.: SFH 309 FA-4/5 | Brand: OSRAM Opto Semiconductors



| 3380 in stock for FREE next working day delivery |          |           |
|--|----------|-----------|
| Price Each (In a Pack of 10)                     |          |           |
| <b>£0.253</b> (exc. VAT)                         |          |           |
| <b>£0.303</b> (inc. VAT)                         |          |           |
| Units  | Per unit | Per Pack* |
| 10 - 10  | £0.263   | £2.63     |
| 20 - 90  | £0.218   | £2.18     |
| 100 - 490  | £0.182   | £1.82     |
| 500 - 990  | £0.14    | £1.40     |
| 1000 +   | £0.112   | £1.12     |

<https://docs-emea.rs-online.com/webdocs/08b2/0900766b808b2f2e.pdf>

### 2. Comparator x 2

**Texas Instruments LM339N/NOPB Quad Comparator, Open Collector O/P, 1.3µs 3 → 28 V 14-Pin MDIP**

RS Stock No.: 533-8237 | Mfr. Part No.: LM339N/NOPB | Brand: Texas Instruments




✓ **£10 in stock for FREE next working day delivery**

Price Each  
**£1.00**  
(exc. VAT)


**£1.20**  
(inc. VAT)

| Units        | Per unit     |
|--------------|--------------|
| <b>1 - 9</b> | <b>£1.00</b> |
| 10 - 49      | £0.58        |
| 50 - 99      | £0.52        |
| 100 - 249    | £0.45        |
| 250 +        | £0.41        |

There is no datasheet, but the specification is shown below.

<https://uk.rs-online.com/web/p/comparators/5338237/>

### 3. Battery x 2




| Stock No | Description                         | Price |
|----------|-------------------------------------|-------|
| B530     | BATTERY PP3 DURACELL PLUS MN1604 9V | £1.98 |

### 4. Red LED x 5

**Kingbright Solid State Lamp 660 nm Red LED, 5mm (T-1 3/4) Round Through Hole package**

RS Stock No.: 228-5988 | Mfr. Part No.: L-53HD | Brand: Kingbright




⚠ Stock check temporarily unavailable - call for stock availability

Price Each (In a Pack of 5)  
**£0.24**  
(exc. VAT)

**£0.29**  
(inc. VAT)

| Units         | Per unit     | Per Pack*    |
|---------------|--------------|--------------|
| <b>5 - 20</b> | <b>£0.24</b> | <b>£1.20</b> |
| 25 - 95       | £0.132       | £0.66        |
| 100 - 245     | £0.126       | £0.63        |
| 250 - 495     | £0.092       | £0.46        |
| 500 +         | £0.078       | £0.39        |

\*price indicative

<https://docs-emea.rs-online.com/webdocs/151e/0900766b8151e40a.pdf>

Quantity: 5

### 5. Battery Holder x 1



**Trupower SBH-9VAS+DC PLUG PP3 Switched Battery Holder DC Plug**

Order Code: 18-0296

☆☆☆☆☆ (Write a review)

Brand: Trupower

MPN: SBH-9VAS+DC PLUG

RoHS Compliant

Qty

Ex Vat\*\*

|      |       |
|------|-------|
| 1+   | £1.73 |
| 25+  | £1.42 |
| 100+ | £1.26 |

Get a quote for large quantities [here](#)

1 **Add to Basket**

\*\*Price per unit  
769 in Stock  
Same day despatch

<https://www.rapidonline.com/pdf/18-0296.pdf>

## 6. Castor x 2

**Guitel Swivel Castor 55021100, 12daN**

RS Stock No.: 306-4287 | Mfr. Part No.: 55021100 | Brand: Guitel



| ✓ 170 in stock for FREE next working day delivery                   |                            |
|---|----------------------------|
| ✚ 10% available from Europe for delivery within 1 working day (max) |                            |
| Price Each<br><b>£1.80</b><br>(exc. VAT)                            | <b>£2.16</b><br>(inc. VAT) |
| Units   | Per unit                   |
| 1 - 4   | <b>£1.80</b>               |
| 5 - 9   | £1.75                      |
| 10 +  | £1.70                      |

<https://docs-emea.rs-online.com/webdocs/1447/0900766b81447f99.pdf>

# Bill of materials

| Name of component   | Number used | Price (£) |
|---------------------|-------------|-----------|
| Chassis             | 1           | 2.00      |
| Motor               | 2           | 1.12      |
| Wheel               | 2           | 0.51      |
| Battery holder (AA) | 1           | 1.02      |
| 1.5V cell           | 4           | 0.24      |
| Breadboard          | 2           | 2.88      |
| Screw kit           | 1           | 0.92      |
| Main PCB            | 1           | 2.71      |
| Main PCB components | 1           | 1.74      |
| Motor driver        | 1           | 5.33      |
| Arduino Uno WiFi    | 1           | 25.20     |
| IR phototransistor  | 1           | 0.30      |
| Comparator          | 1           | 1.20      |
| Caster wheel        | 1           | 1.80      |
| Battery holder (9V) | 1           | 1.73      |
| 9V cell             | 1           | 1.98      |
| LT1366 Op-amp       | 1           | 3.06      |
| MCP6004             | 1           | 0.39      |

The total cost in the bill of materials is: £59.36

## Team structure

Team Leader: Chun Ngai Lam

1. H-bridge
2. Infrared detection circuit
3. Motor control
4. Code for sensors
5. Organize meetings and update Gantt chart

Treasurer: Hisham Nasser

1. H-bridge
2. Infrared detection circuit
3. Motor control
4. Code for sensors
5. Budget and cost for components ordered

Member: William Lam

1. Radio wave detection circuit
2. Band-pass filter
3. Demodulator

Member: Qiyu Xiong

1. Radio wave detection circuit
2. Motor control
3. Demodulator

Member: Snehil Kumar

1. Radio wave detection circuit

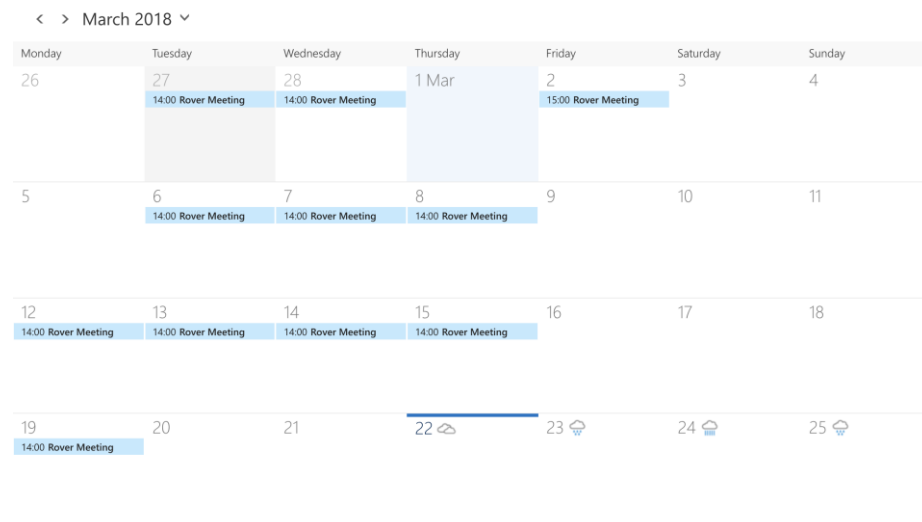
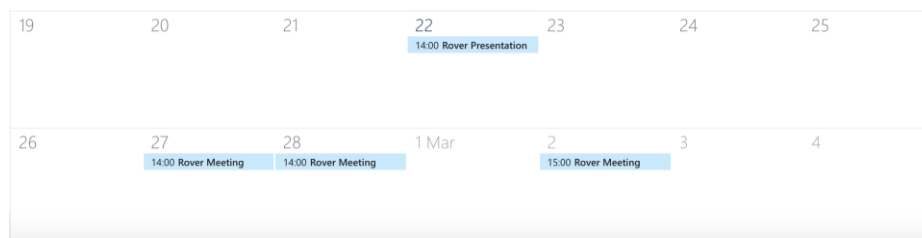


# Project Management

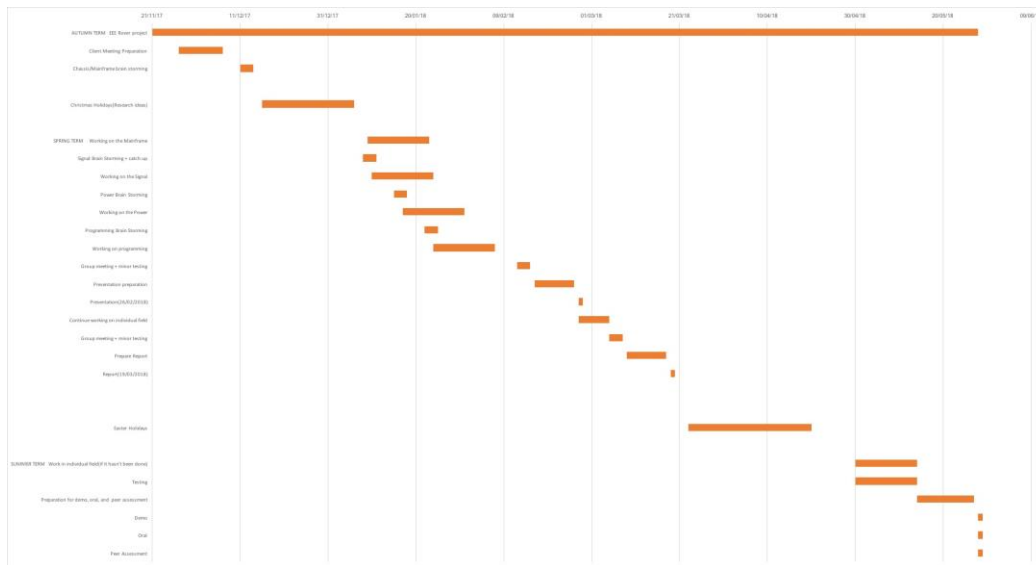
After receiving feedback from the client meeting and presentation we deeply looked into the problems and concerns we had, we learnt to organize things even better than in the beginning and have a better and firm decision for the project.

## Meeting Structure

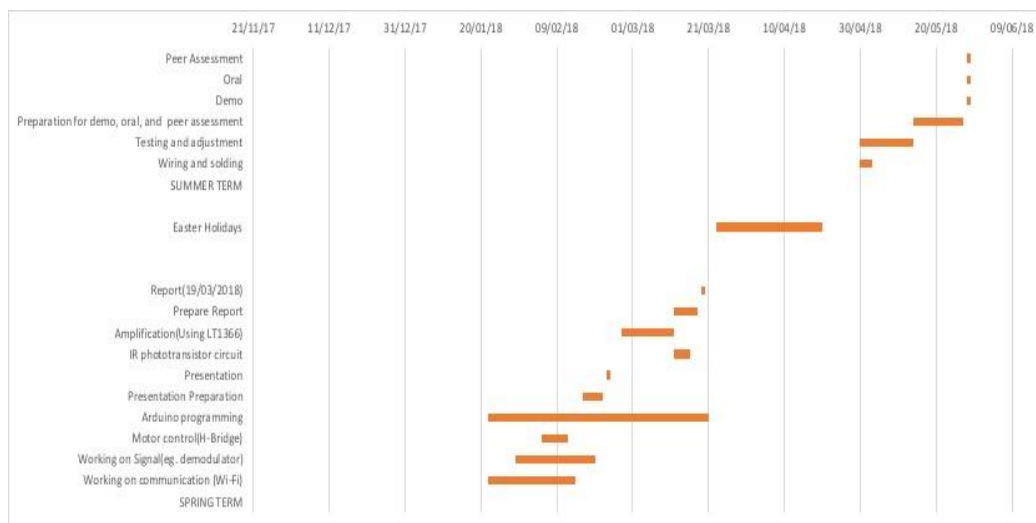
This is the meeting schedule we have had since our presentation, our usual meetings are on Tuesday and Wednesday from 2pm – 5pm, with extra ones on rest of the weekdays from around 3pm – 5pm or whenever the teammates are available.



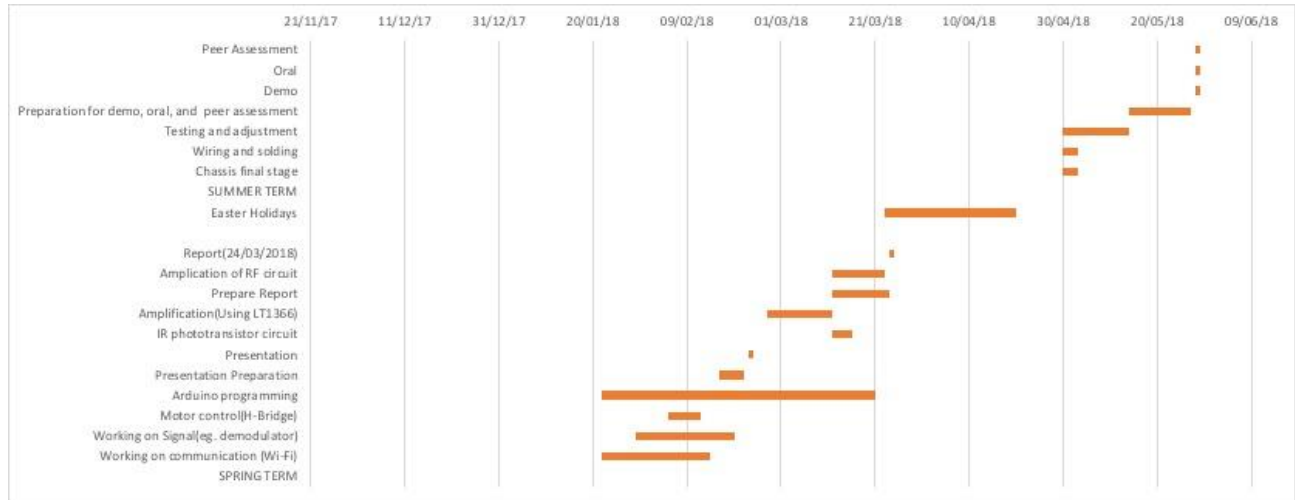
During the meeting we discuss what we are going to do for the project during the day, since the presentation we focused on the programming and sensors, surprisingly we finished these two major parts according to the timeline from the Gantt Chart we made.



*First timeline before any project work*



*Timeline before presentation*



*Timeline after presentation*

After the presentation we also found out that there were some slight technical problems and so in the actual timeline we spent more time fixing the issues.

# Appendix

## Code

Full Arduino script: <https://pastebin.com/NqAcLmAs>

## Assembly Drawing

The Assembly Drawing is in the cover of the report

## References

- [1] Blynk app: <http://docs.blynk.cc/>
- [2] Arduino Uno WiFi: <https://store.arduino.cc/arduino-uno-wifi>
- [3] Battery energy capacity: <http://www.allaboutbatteries.com/Energy-tables.html>
- [4] MCP6004: <https://www.mouser.co.uk/ds/2/268/21733e-41017.pdf>

