Imperial College London

The Mars Rover Project

ELEC50008 Engineering Design Project 2

22nd June 2022

Year 2 Group AAB2

Jiaqi Ge 01846577 - EEE
Qing He 01866590 - EEE
Yuhe Zhang 01899323 - EEE
Shizhe Zuo 01863983 - EEE
Tianyu Zhao 01874783 - EIE
Saifullah Ijaz 01854110 – EIE
Yiwei Feng 01712546 – EIE

Lecturer: Dr. Adam, Bouchaala

Word Count: 6361

Abstract

Over one hundred million miles from Earth, the Perseverance rover is exploring and mapping the surface of Mars to enhance our understanding of the environment. This project aimed to emulate a real 'Mars Rover' in a lab setting by building a small-scale rover that could autonomously navigate an arena designed to reflect an alien environment. The rover consisted of multiple subsystems that each combined to form a fully functioning rover capable of exploring and mapping unknown terrain.

This group project involved students from the Electrical and Electronic Engineering discipline as well as the Electronic and Information Engineering discipline. The students were required to work together and distribute tasks based on subject strength in order to achieve an overall system that functioned as expected. Different subsystems required knowledge of different areas within Electronic Engineering, which is why a Systems Engineering approach was taken to design the mars rover from the ground up.

At the end of the project, students were required to provide a demonstration of their mars rover within the arena to show how well their rover functioned. Obstacles were set up in the arena to mimic an alien environment. The rover needed to successfully navigate through the arena, whilst avoiding obstacles and provide live updates to the user in the form of a map.

Contents

1 Introduction

The specific implementation of this project involved building and testing a solar powered mars rover capable of autonomously exploring the simulated Mars environment. Algorithms were developed to enable the rover to navigate and avoid obstacles as well as detect alien structures. Table tennis balls were used to represent aliens whilst paper structures denoted alien buildings. A fan placed under the arena indicated the presence of underground alien infrastructure. A variety of sensors were used on the rover to detect each of these obstacles and relay this information to the user in the form of a map that updates in real time.

2 Structure Design

2.1 High-Level Design

The top-down approach was taken when designing the overall mars rover system.

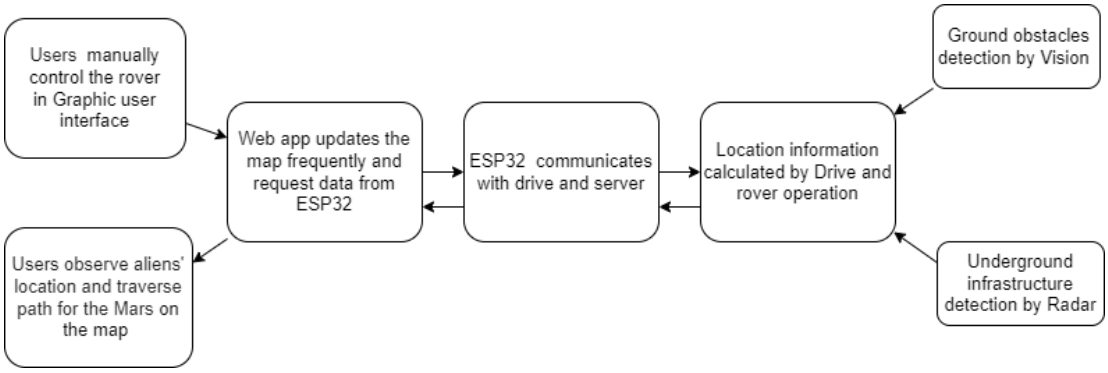


Figure 1 Relationship between each sub-system

2.2 Block Design

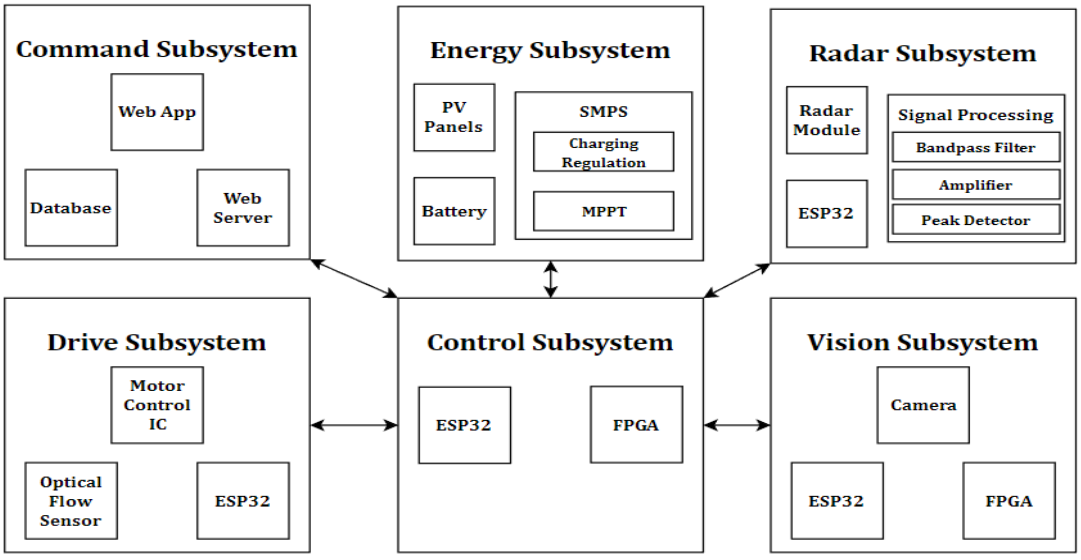


Figure 2 Block design of the whole rover

3 Function Design & Implementation

Considering the overall design objectives, the mars rover was split up into six inter-connected modules. Command displays the traverse path and locations of aliens on the front-end app. Control communicates between the different subsystems. Vision identifies objects of interest. Energy provides the rover's charging system. Radar locates underground alien infrastructure. Drive provides distance measurement, position, and speed control, as well as the algorithms for path finding and underground structure detection.

3.1 Drive

This module was designed to drive the rover to achieve specified distance and angle requirements and simultaneously output the rover's current position. The ESP32 sends a signal to the Motor Control IC (TB6612FNG) which monitors the rover, and the distance travelled measured by the Optical Flow Sensor (ADNS-3080) will be sent to the ESP32 through SPI ports.

3.1.1 Design process

Reading the datasheet and testing the sample code provided a solid foundation to design a fully functioning mars rover.

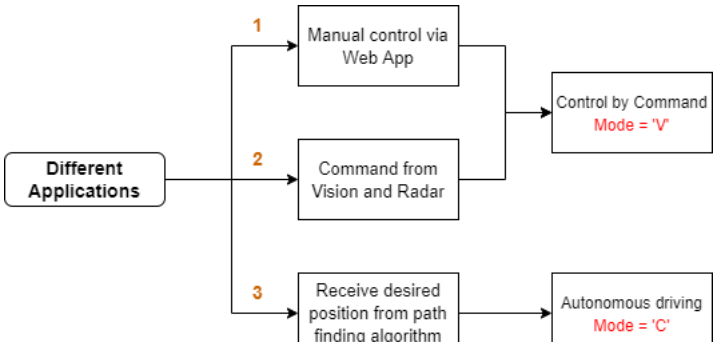


Figure 3 Different Applications for Rover

1) Rover moves when direction buttons are clicked on the web app.
2) Rover rotates until the largest signal given by Radar and Vision.
3) Rover advances autonomously to the desired position determined by the path finding algorithm.

Modes one and two are carried out using manual control signals sent by the command subsystem and mode three involves the autonomous driving functions.

3.1.2 Implementation

3.1.2.1 Control by command

The web page will display five buttons for manual control, which command five different actions on the rover: 'F' for moving forward, 'B' for backward, 'L' for left, 'R' for right and 'S' for stop.

The current coordinate is computed using trigonometry when the rover is moving forward or backward. Because dx and dy are absolute values, the direction of moved distance is indicated by a constant (1 for forward and -1 for backward).

$$moved\ distance = \sqrt{x^2 + y^2}$$

$$current\ y = current\ y + constant * distance * \cos(current\ angle)$$

$$current\ x = current\ x + constant * distance * \sin(current\ angle)$$

The rover's coordinates will not change when the mode is Right or Left, but angle changes can be determined using the chord length formula. It is the chord of the circle, not the arc, because dx and dy indicate the distance travelled since the last capture [1]. As the angle moved by the rover is utilised to determine the current coordinates, it is configured to be between -180° and +180°.

$$moved\ angle\ =\ \sin^{-1}\left(\frac{moved\ distance}{2 * \frac{rover\ length}{2}}\right) * 2$$

$$current\ angle\ =\ current\ angle\ +\ constant * moved\ angle$$

3.1.2.2 Autonomous driving

Autonomous driving is done by angle adjustment first then distance adjustment. The following three steps are considered to achieve desire outcomes:

1)PI controller

The angle error and distance are controlled using a PID controller. The derivative term generates output proportionate to the rate of change in the error signal; the faster the error signal changes, the greater the derivative output. After numerous testing, the findings revealed that the time difference between each measurement is nearly constant, indicating that a PI controller should be used instead of a PID controller. The values used were: $K_P = 0.35$ $K_I = 0.5$.

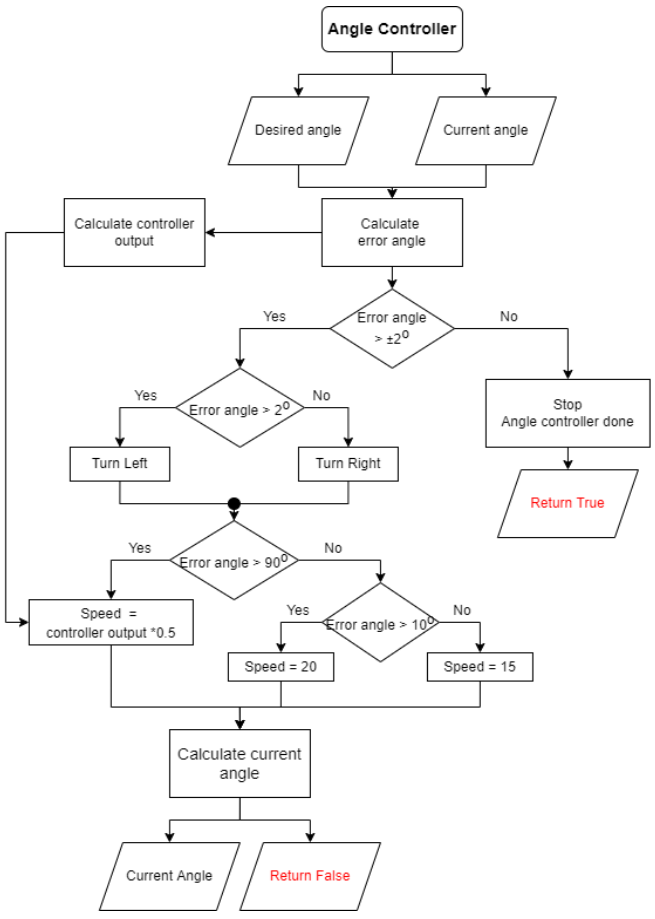


Figure 5 Logic explanation for angle controller

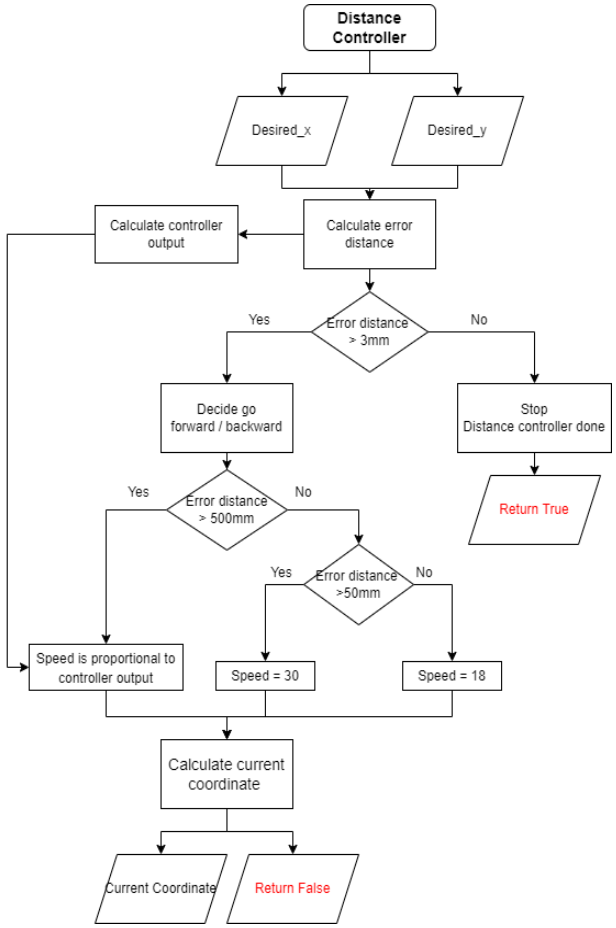


Figure 6 Logic explanation for distance controller

2)Angle adjustment:

The present angle and the desired angle are required to compute the angle adjustment. The required angle can be computed by subtracting the distance between the desired and present coordinates. The sign of the distance difference is determined by the rover's current quadrant. The angle is then simply inputted into the angle controller. Also, after testing, the rover's speed should be greater than 15.

$$desired\ angle\ =\ \tan^{-1}\left(\frac{x_{desired}-x_{current}}{y_{desired}-y_{current}}\right)$$

3)Distance adjustment:
After rotating, the rover will use the distance controller to reach the destination.

3.1.3 Path Finding

Extended from the basic driving functions, path finding is designed to establish an accessible path on the 'Mars' surface, circumnavigating all obstacles. Moreover, it can locate the position of the distinct obstructions present and send their coordinates respectively to the Command subsystem to map them out.

3.1.3.1 Core Algorithm of Path Finding

The fundamental principle behind the path finding algorithm is to optimize the route of travel for avoiding the obstacles based on an initially settled path.
To begin with, a default route which resembles a link of letter 'S's roughly covers the entire site and allows the whole arena floor to be scanned. Ultimately, the rover will return to the start point.
Inspection is conducted by the 'Vision' subsystem during the marching process, and a signal will be generated after determining whether the route to be taken is blocked. Then a reroute is supposed to be implemented, in which case, two coordinates will be generated to guide the rover.

3.1.3.2 Implementation and Code Realization

The movement algorithms for the rover were implemented in C++.
As figure 7 demonstrates, the default periodic 'S' like route (in half line) is set by a *'for'* loop, and the maximum distance the rover would travel due to the site restrictions were set with the consideration of the headroom for rover rotation.
For rerouting, as shown in figure 7.1, an isosceles triangle is modelled at the current position (as origin) in a coordinate system with its base overlapping the original designed path and the other vertices (A and

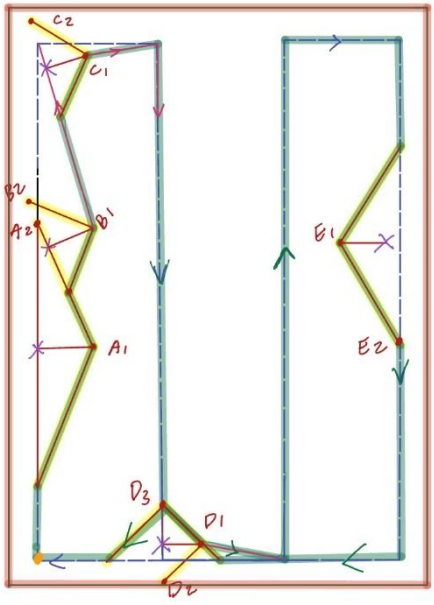


Figure7 Path Finding example

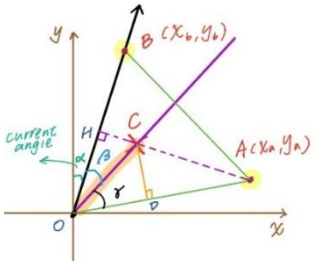


Figure 7.1 Rerouting Explanation

B) being guiding points. The line segment CD is set as a constant equal to 20cm guaranteeing that the rover will not hit the obstacles after changing direction.

$Point\ A: x_a = OA * \sin(\alpha + \beta + \gamma)\ \ y_a = OA * \cos(\alpha + \beta + \gamma)$

$Point\ B: x_a = OB * \sin(\alpha)\ \ y_a = OB * \cos(\alpha)$

The coordinate components above are respectively stored into a *'list'* in pairs. The declaration of iterator helps point to desired elements in the *'list'* and assign the current destination which is designed to be the first pair of elements. The current destination will update as the previous one is reached by removing the two elements in the front of the list.

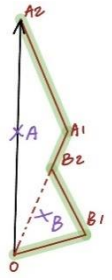


Figure 7.2 Rerouting

If another obstacle is detected successfully during rerouting, two more guiding points will be added into the 'list' to modify the avoidance route. As figure 7.2 shows, 'A' was detected first and 'A1' and 'A2' were in the list already, since 'A1' was not reached yet, the occurrence of 'B' makes 'B1' and 'B2' inserted before 'A1', then 'B1' replaces 'A1' being the current destination. Considering the behaviour limit of the rover and the space restriction, amendments will be made to adapt a more smooth and secure path. As figure 7 shows, the purple cross denotes the obstacles, and the yellow highlighted route is optimized to be the one in green. It is vital that the last point in the list must sit on the default path to bring the rover back to the normal marching pattern.

3.1.3.3 Underground Structure Detection

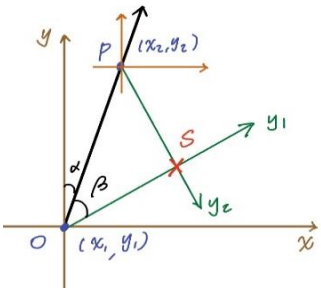


Figure 8 Underground structure detection

Not only will the ground obstacles and aliens be located, but underground terrain change should also be captured. As figure 8 shows, if the black arrow represents the travel route, and S is the target. At points O (the first rotating point is regarded as the relative origin) and P which are indicated by the radar, the rover will turn 360 degrees gradually to find the orientation (e.g.α+β at O) where the head of the rover points directly to the target.

find the line equations:

$y1 = \tan(current\ angle\ at\ O)$

$y2 = (1 - x_2) * tan\ (current\ angle\ at\ P) + y_2$

Then the 'S' coordinates will be the solutions to the simultaneous equations.

3.1.4 Problems Encountered and its Solutions

Following testing, it was discovered that when the present position is ahead of the desired position, the rover will first spin 180 degrees before adjusting the distance, rather than travelling backwards straight. A while loop is added to

check if the rover has reached the desired angle but not the destination. If yes, distance adjustment is applied, and the angle is no longer adjusted.

3.1.5 Evaluation

The performance of the Drive module is generally good, it can achieve the destination position and can also be controlled by command. However, the accuracy of the measurement from Optical Flow Sensor is not very high, and when the rover is near the destination, it oscillates around the desired position because the rover cannot reduce the speed any further.

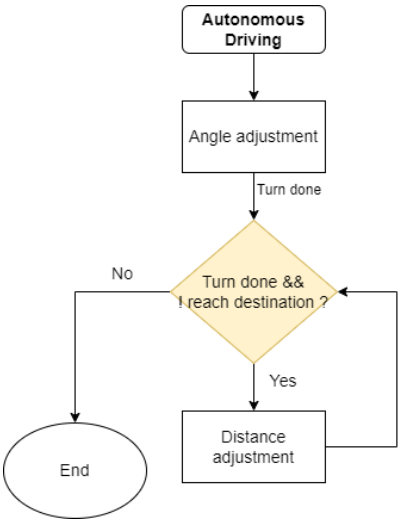


Figure 9 logic explanation for main loop

3.2 Vision

3.2.1 The overall goal of this subsystem

When the rover is moving towards its desired position, the control and drive subsystems require a relative position of the aliens and structure so that the rover can automatically avoid these obstacles. To achieve that, there are three separate functions necessary to be implemented: detecting the desired colours of the obstacles; calculating the distance between the rover and the target; transporting the message to the ESP32 which communicates to the control and drive modules.

Whilst avoiding the obstacles, the velocity and position of the Rover both changes, therefore a low latency system is required. The advantage of implementing visual capability on FPGA boards is that it can perform pipeline operations and achieve the highest real-time performance in image processing [2].

3.2.2 Specific Implementation of Designing and Communications

The progress of the subsystem is shown in the flowchart on the right.

The camera will input the image it has taken to the system through RGB light information. Using the colour information, the obstacle can be determined and bounded in a box, thus allowing the distance and angle to be calculated [3]. At end of the flow, the subsystem generates a message containing the colour and position of

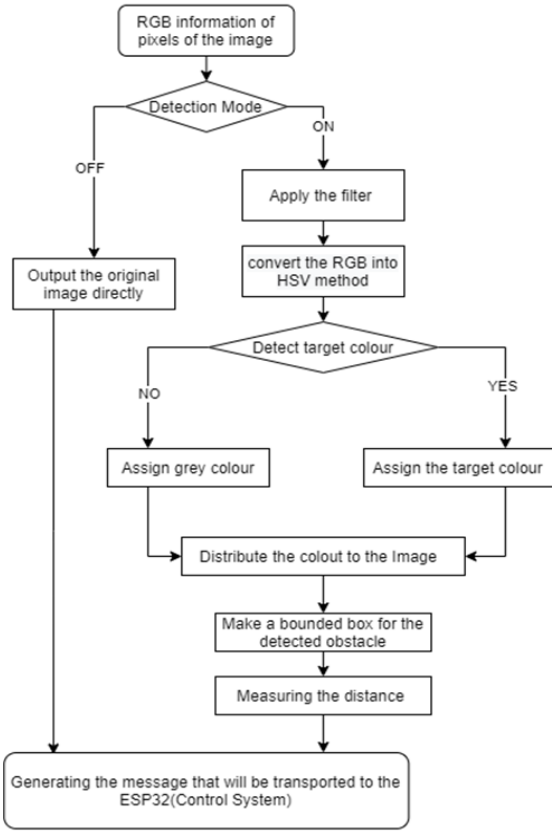


Figure 10 workflow for image processing

the obstacles.
The UART protocol in the softcore processor will then send the message to the ESP32 to update the database with the position of the object detected.

3.2.3 Performance and Improvement

The first version of this subsystem resulted in detecting too much colour at the target and there was a lot of noise in the image.
To set the colour more accurately, HSV colour space was introduced [4], which consists of 3 matrices, 'hue', 'saturation' and 'value'. These helped to adjust the parameters of targets with any change of the environment light.
To reduce the noise, a proper filter must be applied. The background noise contains pixels of the same colour as the balls. Therefore, the Gaussian filter would not perform well as it is more suited at removing white noise in an image. Here, the erosion filter is chosen [5]. The erosion filter generates a 3x3 kernel that involves the detection signal. The threshold count of detected pixels (signal of 1) is set as six here. Then, some of the noise will be removed and blocks of pixels with the same colour will remain highlighted in the output image (shown in figure11, the left side is filtered image, right side is unfiltered).

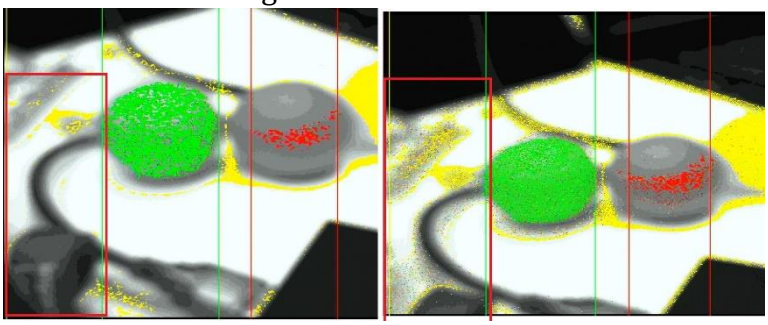


Figure 11 Image after erosion filtered

3.2.4 Testing and Potential Improvement

After adjusting the parameter of the detection colour, the figure below shows how the image has been processed.

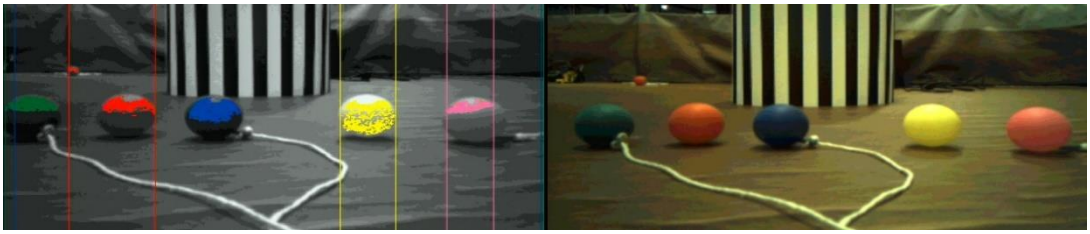


Figure 12 The final performance of Vision

The blue, yellow, and green ball detection produced an acceptable result, but the red and pink ball did not. Since these two colours were too close in colour space, distinguishing them would result in loss of information in the image.

3.3 Radar

The radar system aims to detect and locate an alien underground infrastructure, which is modelled by a fan with rotating blades, using a Doppler radar module that is integrated to the Rover.

3.3.1 Characteristics of the Radar Module

The primary parameter of radar module was tested, as individual modules may vary in performance. The signal from the radar module, when placed next to the fan, has 366Hz frequency, 12.5mV amplitude and -200mV DC offset. The signal varies between 323Hz and 371 Hz, as distance to the target and input voltage of the fan changes.

3.3.2 Design and Implementation of the Signal Processing Stage

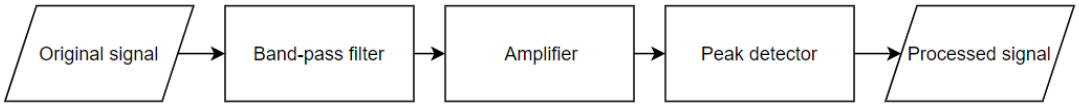


Figure 13.1. The Signal Processing Structure

First, a band-pass filter [6] with a centre frequency of 366Hz and a passband of 100Hz was chosen to reduce the clutter present in the radar signal. It was decided to use a Butterworth filter [7] to get maximally flat frequency response. Second-order filter was chosen to give large enough slopes and thus good attenuation for clutter frequencies outside passband, while having relatively simple structure. Since area available on Rover's breadboard was restricted, reducing components number is critical, circuit was built as shown in *Figure 13.2*. Op amp MCP6002 is used with single supply, with Arduino providing 5V [8], thus 2.5V was set to be mid-supply voltage reference. Two identical resistors, acting as potential divider, as shown in *Figure13.3*, were used to produce it.

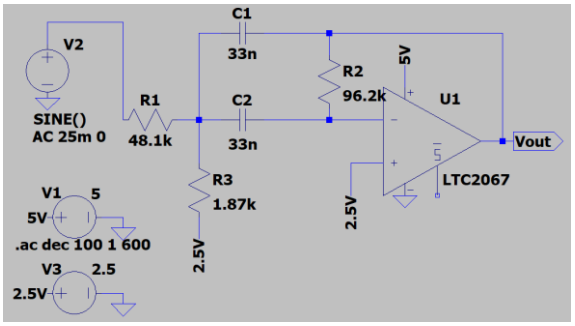


Figure13. 2. Band-pass Filter

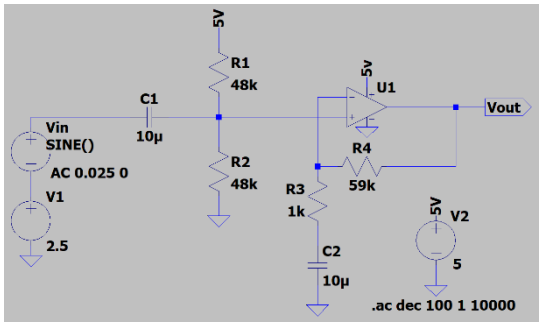


Figure 13.3. x60 Amplifier

The signal was still weak after filtering out undesired clutter signals, target detection remains difficult. Therefore, an amplification circuit, presented by *Figure 13.3*, was designed to boost the output signal of the filter. The largest potential gain of the circuit would be 128, with a signal amplitude of 12.5mV and the lowest input voltage of 1.6V. Yet it was experimentally found out that maximum gain obtainable with this op amp and Arduino 5V supply is only around 72. Thus, the final gain value was set to 60, which was within the actual obtainable gain and the tolerance of the chosen op amp (MCP6002) [9]. To avoid the unnecessary sign reverse, the design was simplified by using a non-inverting

amplifier. In addition, two capacitors, C1 and C2, were added to the circuit to block unwanted DC components from being amplified.

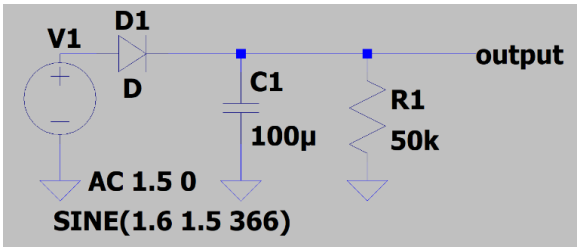


Figure13. 4. Peak Detector

A peak detector [10] was designed in the final stage to process the amplified signal. It would generate a DC signal whose amplitude is proportional to the amplitude of output AC signal from the amplifier, with very small ripples.

3.3.3 Integration with Other Systems

First, the analogue DC signal leaving the processing circuit is converted to digital signal using '*analogRead*' in ESP32. It is then compared with a threshold value to inform the Rover when to stop and rotate 360°. Since the processed signal has the greatest magnitude when the Rover is facing towards the target, the orientation of the target with respect to the Rover could be calculated from determining the angle at which the maximum signal value occurs. When the above process is repeated for two stopped positions, the target's exact coordinate could be determined by the Rover, with Underground Structure Detection algorithm, which is explained in Drive Module.

3.3.4 Problems Encountered and Solutions

Firstly, after characterising the radar module and testing with bandpass filter, the radar module was damaged accidentally, while the new radar module takes time to arrive. To continue work on this subsystem, signal generator from the Picoscope was used to model the radar signal, for testing the signal processing circuits and integration with drive module. Secondly, the potential divider could only produce a 1.6V voltage in actual implementation, rather than the ideal 2.5V. This is possibly due to the Arduino 5V port having a limited output power. To solve this, the signal processing circuits (mainly the peak detector) were modified to operate on DC voltage of 1.6V.

3.3.5 Evaluations and Reflections

Tests with implemented signal processing circuits show they work satisfactorily, attenuating frequencies outside passband significantly, giving roughly 59 gain, outputting final DC signal at 1.52V with around 35mV ripple, as given in appendix 7.7-7.12.

The radar system demands a better coverage area, while the vision system desires the fastest scanning speed, resulting in a priority conflict for Rover drive module. To improve, an optimal approach for determining the balance is crucial. Regarding the locating method, due to rotation of Rover needed and error inherent in the optical flow sensor, errors may occur when establishing the

coordinate of the target (at the intersection), which is discussed in Integration module.

3.4 Control

The control subsystem, with a few stand-alone features, functions as a connection hub to transfer information between different subsystems. In embedded system design, the communication protocols are the essential technical elements that facilitate a stable connection between the subsystems.

3.4.1 Serial Communication:

There are two types of interface modes: parallel interface and serial interface. The serial interface uses a smaller number of conducting wires, which would reduce cost on an actual mars rover. Since data is transmitted one bit at a time, there are fewer errors and stronger noise tolerance compared with parallel communication.

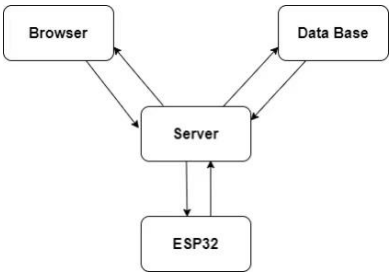


Figure 14 flowchart of control

3.4.2 UART

Considering hardware communication between the ESP32 and the FPGA, UART is one of the most widely used device-to-device communication protocols. Applying the two-way communication protocol, two wires are used for successful serial data transformation, which means that the esp32 and the motors were able to receive data at the same time. Asynchronous communication requires that the baud rate is synchronised. The data is transferred in the packet. A packet consists of five pieces: Start bit (1 bit), data frame (5 to 9 Data Bits), Parity Bits (0 to 1 bit) and Stop Bits (1 to 2 bits).

3.4.3 HTTP（Hypertext Transfer Protocol）

As a client-server protocol, HTTP provides communication between the ESP32 board and the server which is built into the Command subsystem so that the information collected from the Rover can be updated in the web app.

3.4.4 SPI

SPI (Serial Peripheral Interface) is one of the primary communication interfaces in embedded systems. SPI works in a synchronous, full-duplex, and master-slave fashion. Both master and slave are capable of transmitting data at the same time. Therefore, it is not required to add extra start and stop bits to each byte to keep the receiving end synchronizing. The wiring detail is given in the Appendix 7.13.

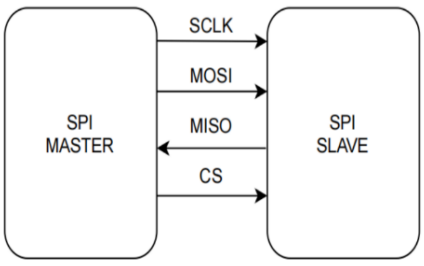


Figure 15 SPI Interface

3.5 Command Subsystem

3.5.1 Overview of Command Subsystem

The command subsystem consists of a web app, a web server and a database designed to allow the user to control the rover remotely. The web app displays the rover's current position as well as a map of the alien environment, including the positions of the aliens, their buildings, and their underground infrastructure. The web app connects to the web server and the database to enable two-way communication between the command subsystem and the other modules.

3.5.2 Design decisions

The web server was hosted locally instead of using AWS cloud servers, to avoid any propagation delays, since free tier AWS servers were located in Virginia. For the back-end, Node.js was paired with the Express.js framework [13] as there was lots of support and documentation available online. Initially, React.js was selected as the front-end library of choice. However, due to it relying on its own 'JSX' syntax as opposed to regular HTML, it was decided that regular JavaScript, HTML and CSS would be the front-end technologies of choice. MySQL was chosen as the database management system as it was simple to connect to the web application [14].

3.5.3 Implementation

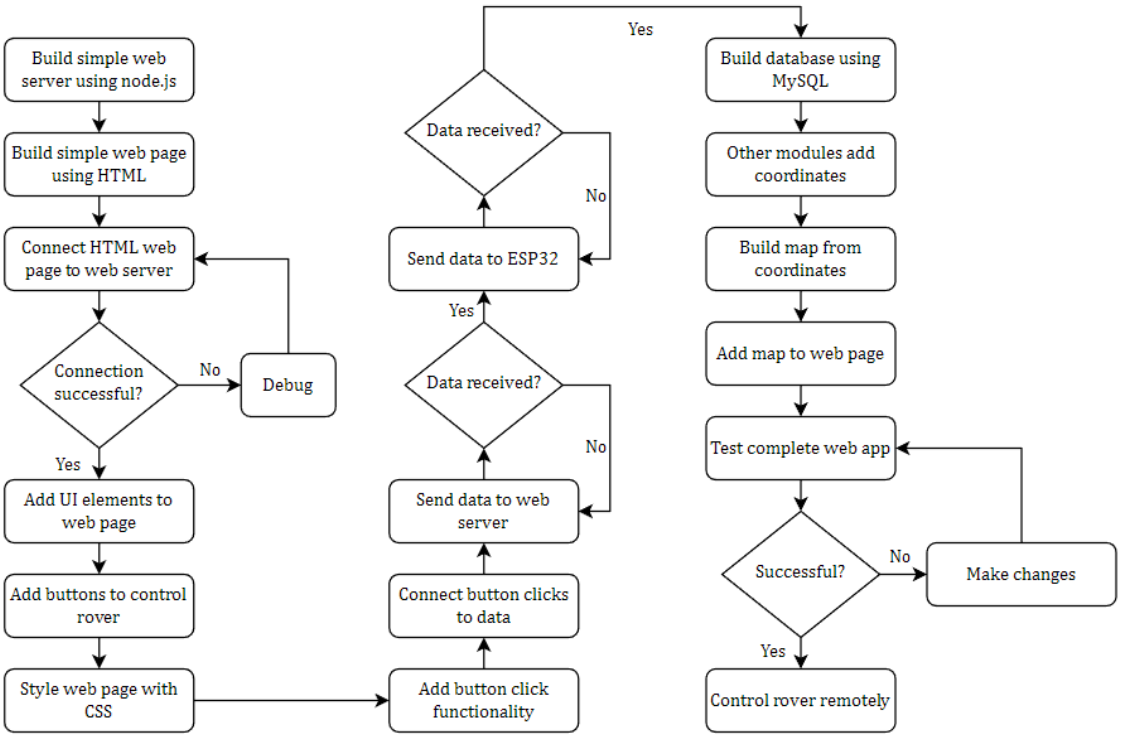


Figure 16 Implementation of Command

3.5.4 Successfulness of Command Subsystem

Overall, the command subsystem fulfilled the functional requirements that were set out in the specification. The web app was easy to use and connected well with the web server and the database. Initially, there were some issues connecting to the control subsystem, but these were overcome to enable remote manual

control of the rover. The map functioned as expected and provided live updates of the positions of the rover, the aliens, and their underground infrastructure.

3.5.5 Improvements

Security is one of the most important components of a real world project. The rover should guarantee that wireless data transformation is fully encrypted, and operation authority is always in the client's hand. The SSL (Secure Socket Layer) is ideal to use within the command subsystem which means that the HTTPS communication protocol could be used as it contains features of SSL. Secure connections, therefore, could be established by making use of the SSL certificate.

3.6Energy

The main principle of the system is to charge the battery pack effectively and efficiently, which will be put onto the Mars Rover and provide power to it after charging, using 4 PV panels that convert solar energy to electrical energy.

3.6.1 The Characterisations of PV Panels and the Battery

The characteristics of the battery pack was investigated by charging it with an external power supply. Battery input current varies linearly with input voltage between 4.8V to 5.2V, given in the Appendix 7.14. When out of voltage range, the current stays constant. Meanwhile, characteristics of PV panels and power-voltage relationships were tested, using one panel as representative, at various irradiance (modelled using different tilt angles), through adjusting the load resistance (given in the Appendix 7.15 and 7.16). The power would reach its peak around 4 to 5V, which changes based on solar irradiance, tilt angles, shading and temperature. Additionally, it was also tested out that any proportion of partial shading reduces panel output power significantly, aggravating as shading increases, given in the Appendix 7.19.

3.6.2 The Design and Implementation of the Charging System

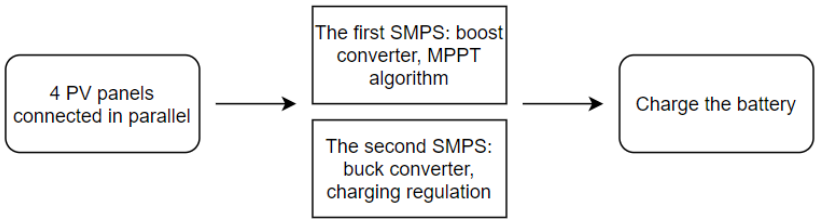


Figure 17. Structure of the Charging System Designed

Since the SMPS had 8V (port B) voltage limit, panels are restricted to be in parallel, producing nominally 5V voltage and 920mA current. To maximise panel efficiency, the Maximum Power Point Tracking (MPPT) algorithm [15] is used, and to avoid existence of local maximum power points due to partial shading [16], panels are placed as facing towards the sun as possible, with no shading on them.

Two SMPSs are used for the power electronics interface. The first SMPS executes the MPPT algorithm. Studies [17] suggested Perturb and Observe Method, which is a trial-and-error iteration that adjusts operating point on the curve through introducing small perturbations into the system. The duty cycle is saturated at 30%

to keep output voltage less than 8V, although the limit is almost never reached. Two SMPSs are connected in series. The second one adjusts its duty cycle, proportional to its input voltage (with coefficient named *gain*), to control the current flowing in the system, drawing a current that the first SMPS needs for MPPT. A consistent power could thus be ensured throughout the entire energy system. The second SMPS outputs voltage within the charging voltage range and it controls the relay that connects its output and battery pack, disconnecting battery pack from its output when output voltage is out of range. Design is detailed in *Figure 18* below.

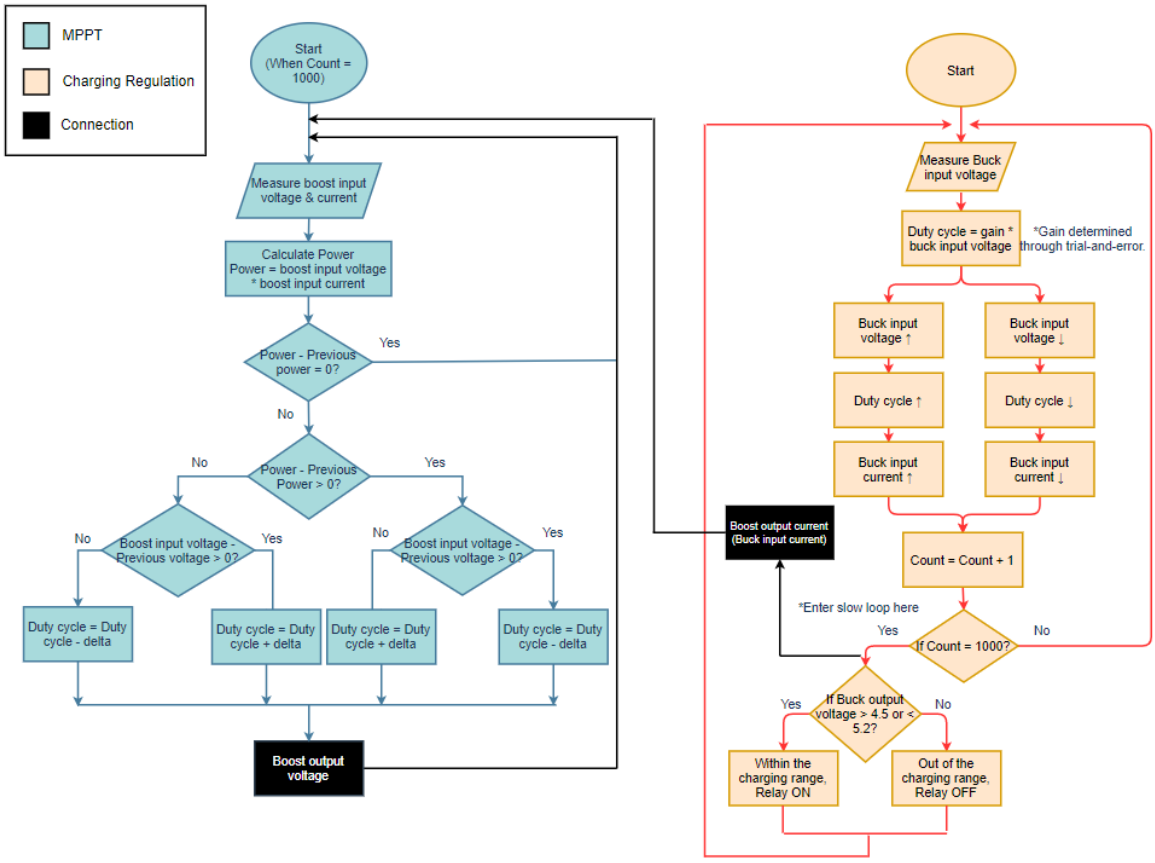


Figure 18. Structure of MPPT and Charging Regulation Algorithms Designed

The final decision on converter types was significantly influenced by the 8V voltage limit of SMPS circuit and existence of SMPS internal consumptions. Using both boost is not valid, as panels usually deliver 5V or higher and the battery requires input around and often less than 5V depending on state of charge. If using two bucks, converting between similar voltages may raise duty cycles up to 99%, leaving MPPT with very restricted range to control and track. Then, buck and boost were implemented and proven to be unsuitable, as the boost output voltage is lower than desired due to considerable internal power consumption of the boost converter. Therefore, boost and buck combination is chosen. It was planned to use both converters in synchronous mode to maximise efficiency and eliminate diode power loss, however, in the actual implementation, only buck is synchronous, and boost is asynchronous, which may be simpler to control.

3.6.3 Problems Encountered and their Solutions

Boost and buck input voltages are found to be usually higher than 5V and cannot be read correctly by Arduino analogue pins. To solve this, they are connected via potential dividers to analogue pins. Secondly, it was found out that battery only starts charging when its input voltage is from 4.5 to 5.2V, requiring a lower *gain* for buck duty cycle. After charging begins, the battery input voltage drops significantly depending on its state of charge, charging current increases as the proportional *gain* for buck duty cycle increases. Therefore, a different, higher value is set for *gain* than default value when buck output current is higher than a threshold. The buck output power and charging current are found to be significantly higher after this improvement.

3.6.5 Discussions and Evaluations

Average efficiency of the designed power electronics interface was satisfactory – 69.89%, given in the Appendix 7.22. Average output power was 0.41W before setting separate *gain* values for charging and not charging state and is 1.36W after the improvement. It was found out that voltages sometimes still saturate the Arduino's analogue pin's limit, even after connecting via potential dividers. This could cause the algorithm to not work correctly and the lower-than-ideal output power. Besides, a better control algorithm for the buck part could be implemented to increase the output power.

Many circuit connections were made using breadboard and jumper wires. Making these connections on a specially designed PCB board with tracks will possibly reduce power loss and improve efficiency.

Currently the Arduino and SMPS circuits are powered by USB cable or 5V power supply. Considering that stable power supply from laptop or plug socket does not exist on Mars, the station could be made to charge separate batteries when not charging Rover's batteries, then station can be powered by them after making connections.

Total power consumed and the power supply current by Rover are estimated to be 6.49W and 1.30A respectively, and thus battery run time will be 3.86hrs, given in the Appendix 7.20 and 7.21. Current way of powering the Mars Rover is good enough for demo in our arena, nonetheless, if the rover is designed to operate on Mars, algorithm should be implemented to inform the Rover the stored energy left and return to charge when needed. PV panels, power electronics interface and batteries could also be integrated to the Rover to avoid power station being damaged by strong Martian winds or risk of not returning to charge in time. PV panels could also be controlled to adjust its orientation and tilt angles to maximise output power.

4 Integration & Testing

The final stage was to assemble all the components into a fully functioning system. Integral testing was implemented under two driving modes; the first was to combine Command with Control for manual control, and the second was to

interact with Drive, Radar and Vision for autonomous control. The functionalities of the first stage were tested by illustrating the information received on the serial monitor.

In the automatic driving mode, the ESP32 is required to connect with WIFI and keep sending requests to the server to update the driving mode information. Once the ESP32 receives the mode information, the mode variable should be changed to 'C' as mentioned in 3.1.1.

The serial monitor illustrates: "Rover is in automatic mode".

In operation mode, ESP32 will send requests of movement-direction information from the server and sends back the Coordinate information of Rover simultaneously.

The serial monitor illustrates: "Mode is V, the direction is X; X should be F (Forward), B(Back), L(Left), R(Right).

It was found that if the rover is travelling in the same direction over 2 meters, the current angle will generate a small deviation, which is probably caused by the battery holder touching the ground, then a further checking is implemented. In addition, the change in the position is inevitable during the rover rotation, all related algorithms have to take it in to account.

5 Project management

At the project commencement date, task allocation was scheduled, and each member was given different roles and responsibilities for the project as given in appendix 7.1. It was agreed to have meetings in person on Tuesdays and Fridays for regular updates. Based on the design phase, the project timeline was created as shown in appendix 7.2.

A team was created with all members included on Microsoft Teams which allowed us to share files, documents, and plan meetings. GitHub was set up as a code repository, and each amendment to the code was pushed to the repository, so it was synced for each member. If necessary, the Imperial College EEE Department Staff were consulted for clarifications and issue resolutions.

6 Conclusions & Recommendations for future work

Overall, the project provided a good basis for working in an engineering team on a multidisciplinary project and the rover functioned as expected.

For future improvements, the real Mars environment could be considered. To avoid extreme weather damage, the auto-sleep and auto-wake-up function could be set up in the programme, making sure the rover only works under safe conditions. And because of the large temperature difference, the outer shell material should have a good thermal insulation performance and be as light as possible. More wheels could be used and equipped with a better suspension system to overcome the bumpy terrain of Mars.

7.Appendices

7.1 Task Division of Project Management

Task	Responsible Person
Implementation of web app, web server and database	Saifullah
Driving mode implementation, PI controller	Shizhe
Path finding algorithm, underground structure detection	Yuhe
MPPT & Voltage Regulation algorithms, characterisations of PV panels and battery	Jiaqi, Qing
Radar testing, signal processing & circuit design	Qing, Jiaqi
Detecting obstacles, measuring distance from rover to obstacle, communication with ESP32 by UART	Tianyu
Add WIFI API, implement HTTP connection with server	Yiwei

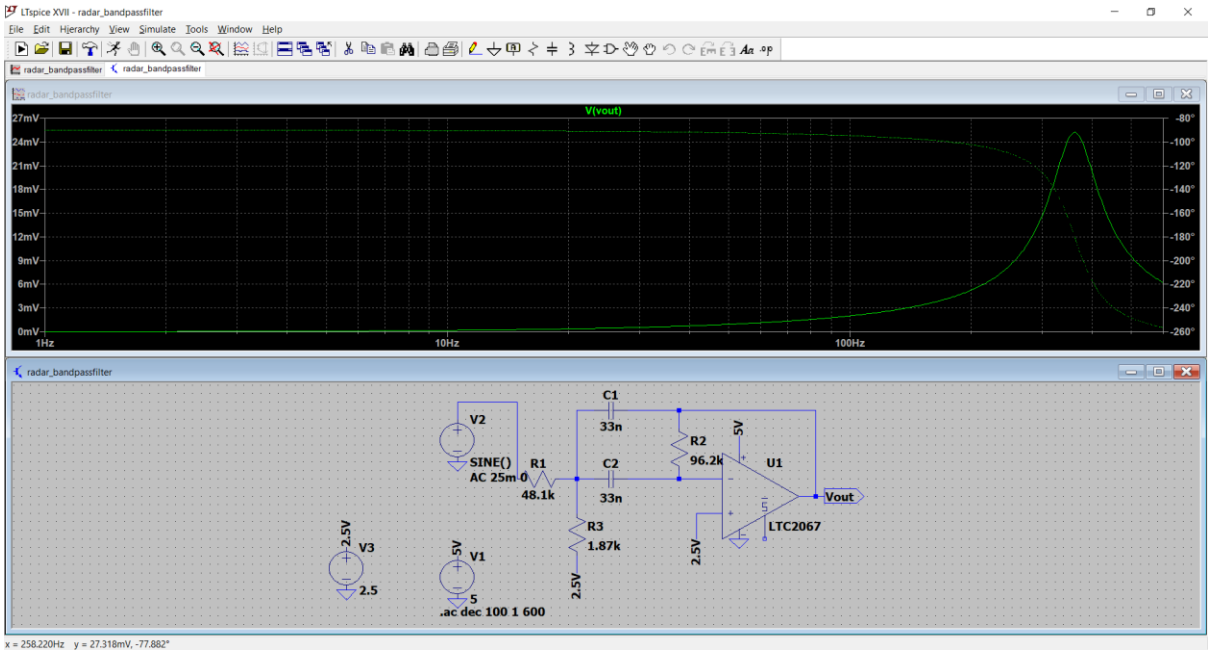
7.2 Timeline of Project Management

Time Point	Milestones
23/05/2022	Project overview, subsystem assignment, background research
27/05/2022	Sample code testing, algorithm research
01/06/2022	Preliminary testing, algorithm optimization
08/06/2022	Further testing, algorithm enhancement
15/06/2022	Report Finalization
22/06/2022	Final testing, demo preparation
28/06/2022	Demo Presentation

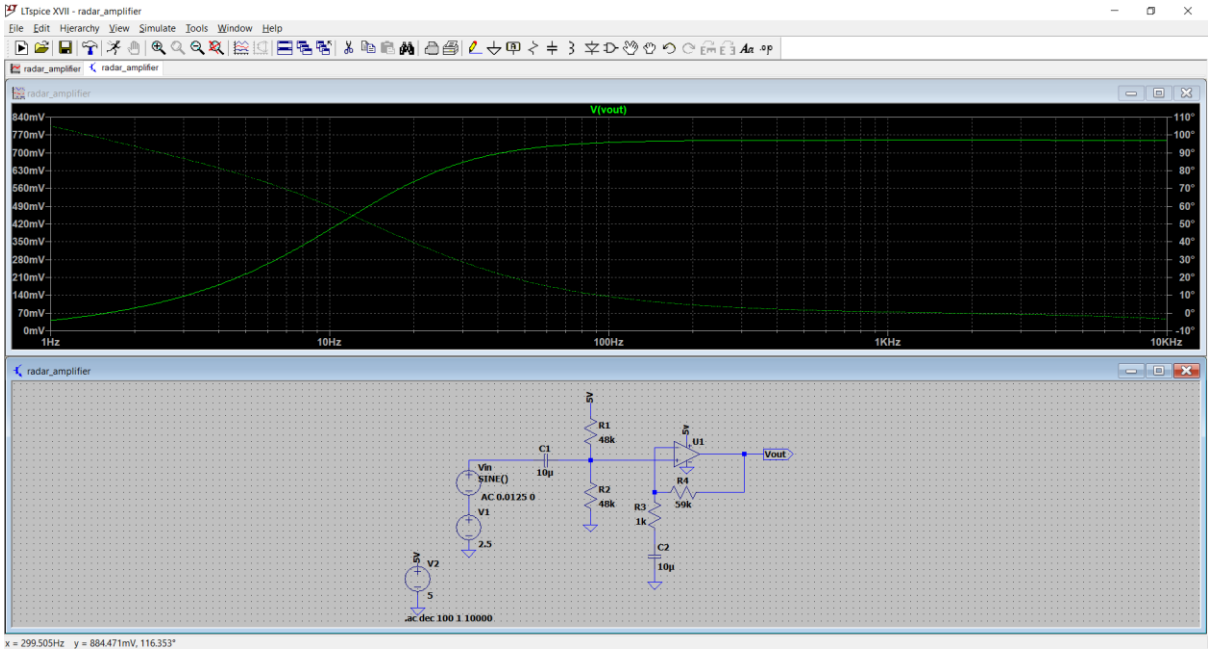
7.3 Radar Signal Tested at Different Distances and Fan Input Voltages

Distance to Target (cm)	Voltage (V)	Frequency (Hz)
5	10	323
5	10.5	328
5	11	348
5	11.5	357
5	12	371
7	12	370
10	12	371

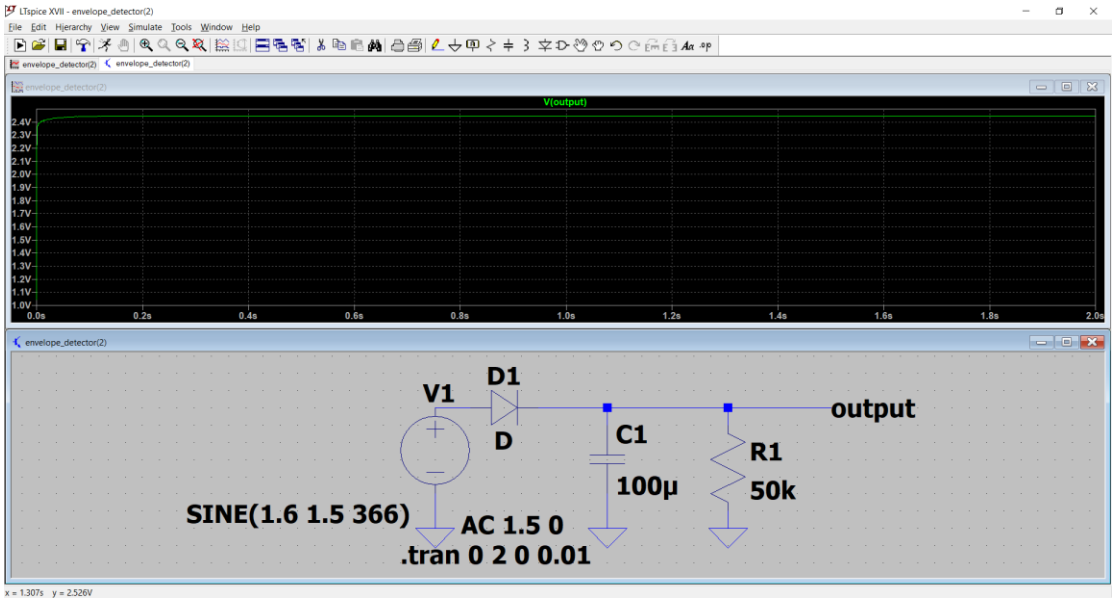
7.4 Band-pass Filter LTspice Simulation



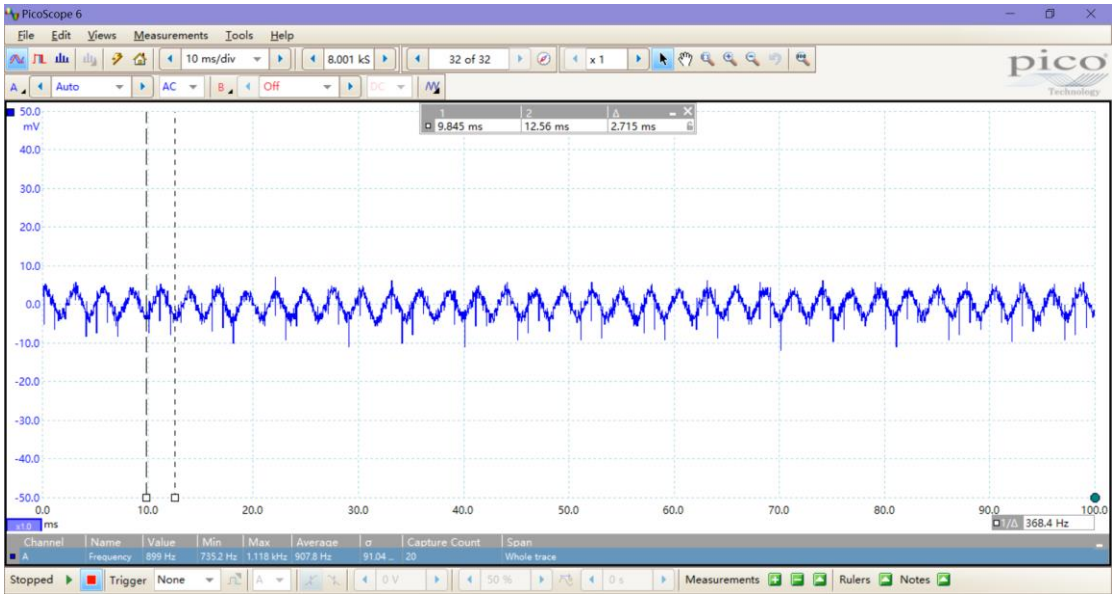
7.5 Amplifier LTspice Simulation



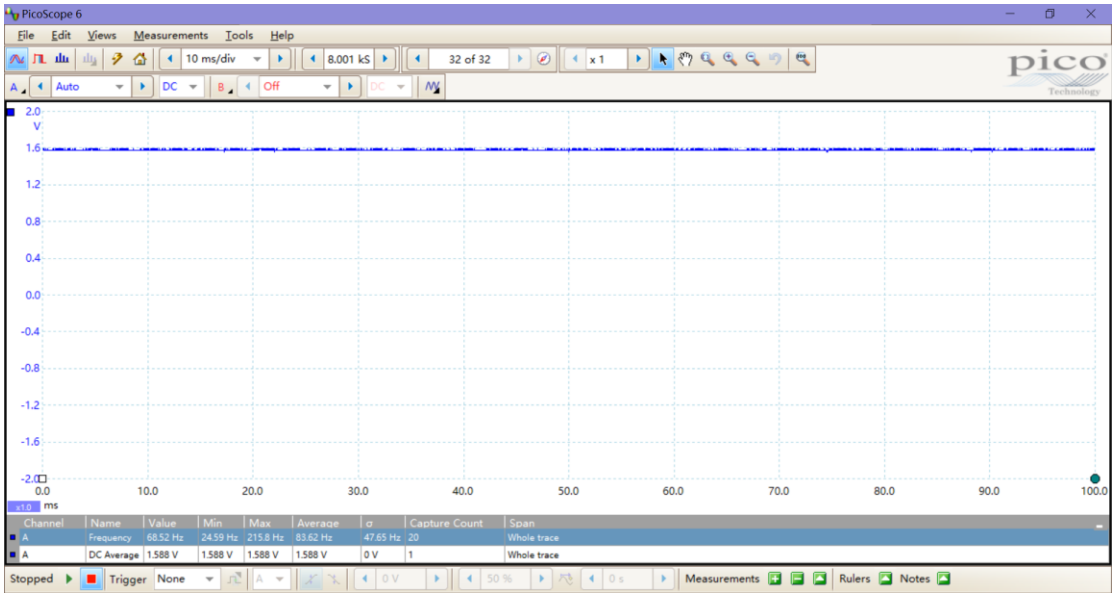
7.6 Peak Detector LTspice Simulation



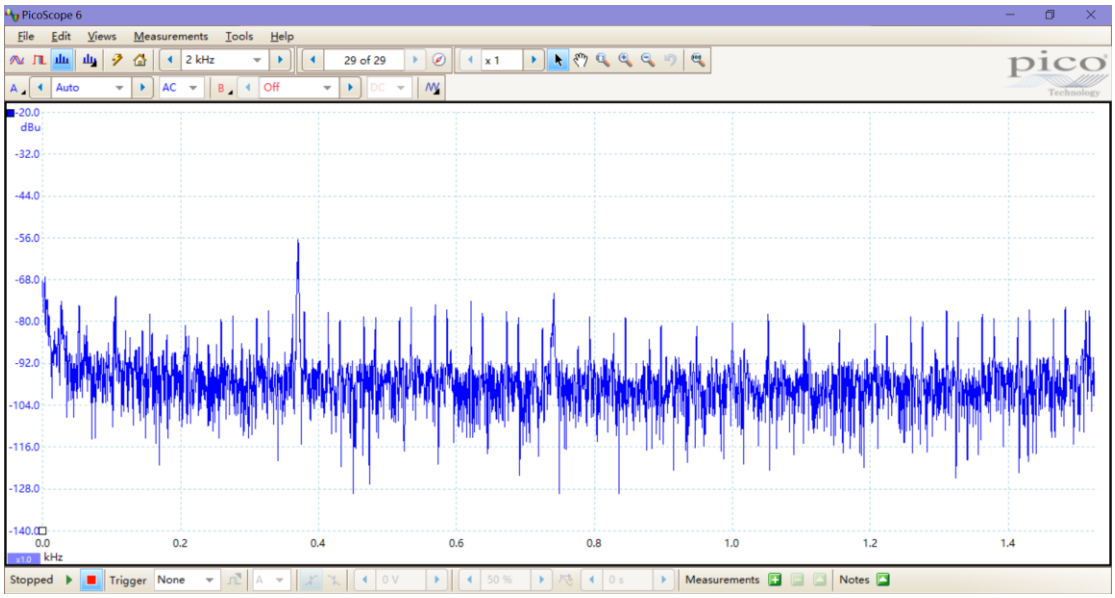
7.7 Bandpass Filter Output AC (Using Signal Generator)



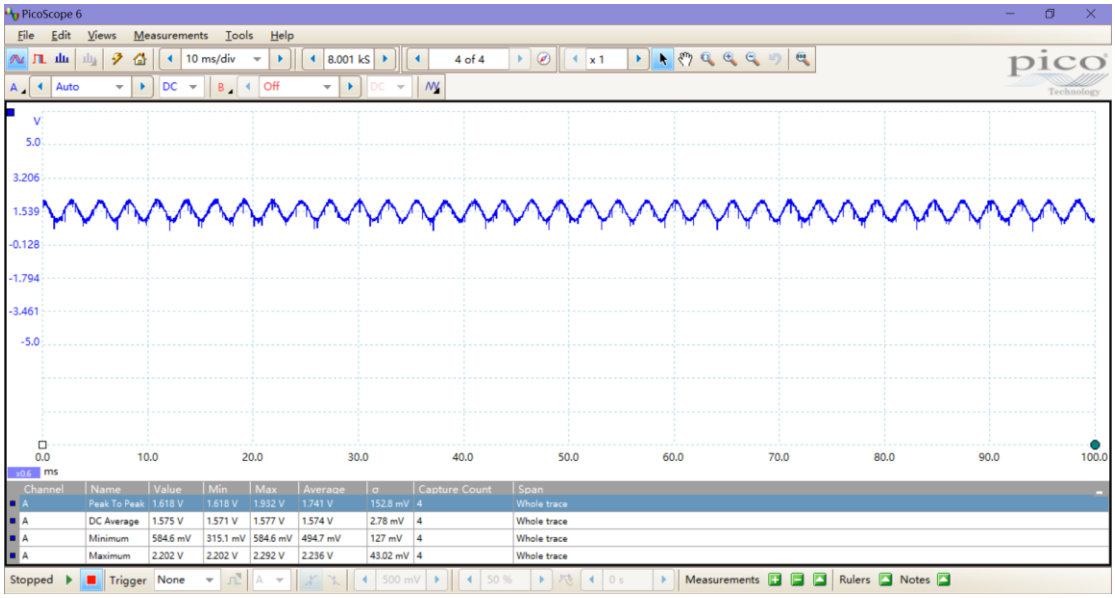
7.8 Bandpass Filter Output DC (Using Signal Generator)



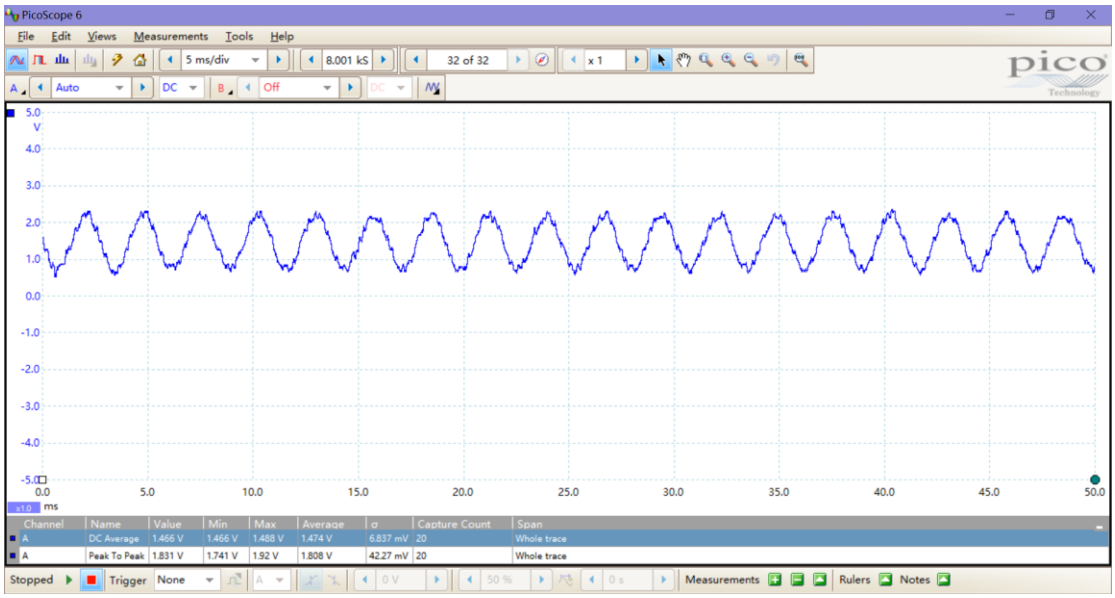
7.9 Bandpass Filter Output (Using Radar Module, Input 11.6V, Distance to Target 5cm)



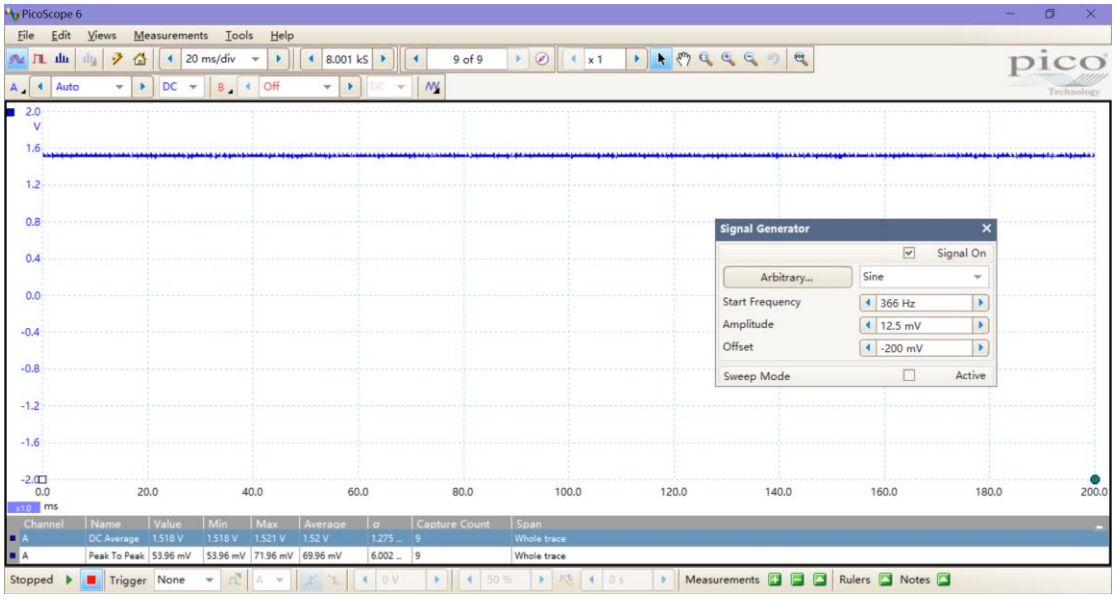
7.10 Amplifier Output, Gain 59, (Signal Generator)



7.11 Amplifier Output, Gain 72 (Signal Generator)



7.12 Peak Detector Output (Signal Generator)



7.13 The Four Essential Wirings for SPI Interface

In SPI interface, there should be at least four 4 wires:

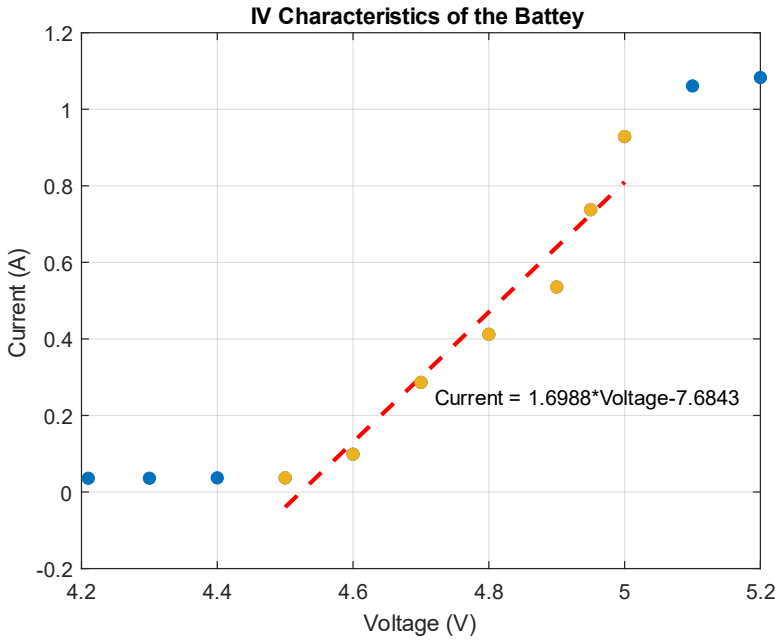
1)SCLK (Serial Clock): Generate clock signal from master.

2)MOSI (Master Output Slave Input): Transfer data from Master, receiving data from Slave

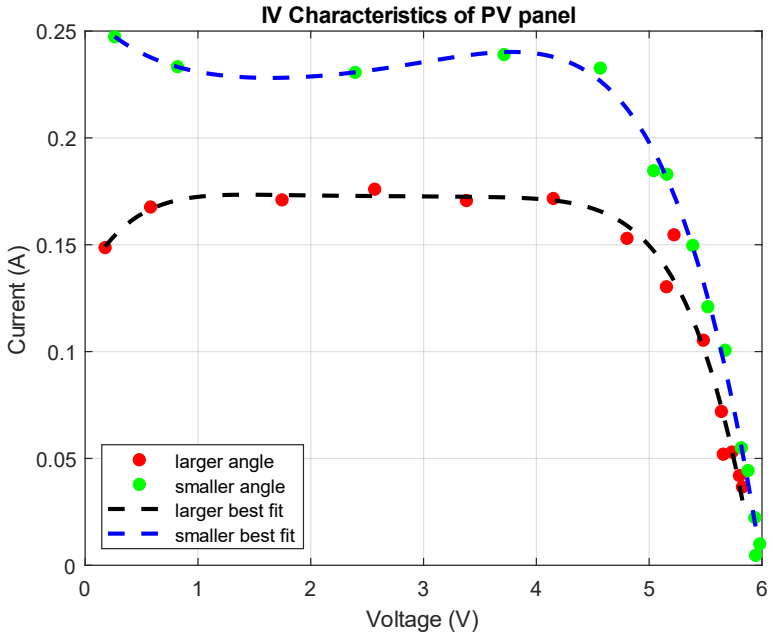
3)MISO (Master Input Slave Output): Transfer data from Master, receiving data from Slave

4)CS (Chip Select): Signal generated from master and enable the slave to receive signal.

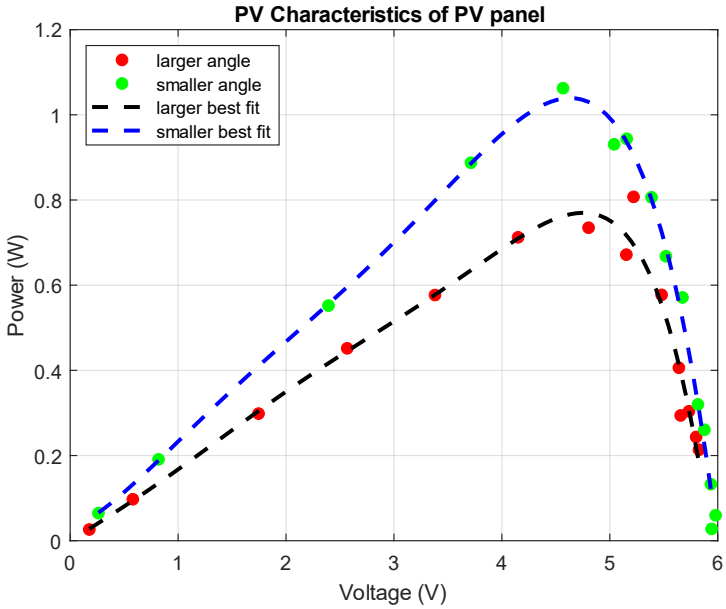
7.14 IV Characteristics of the Battery



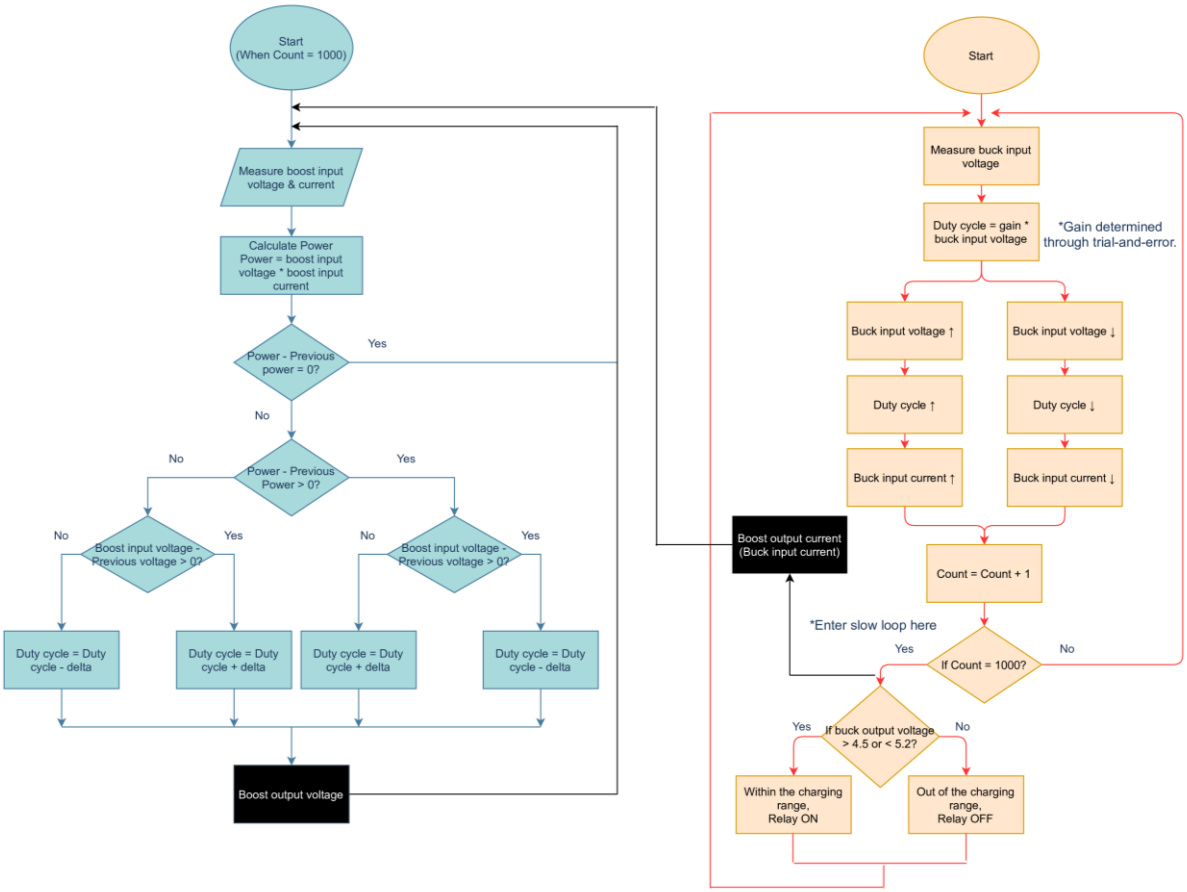
7.15 IV Characteristics of PV Panel



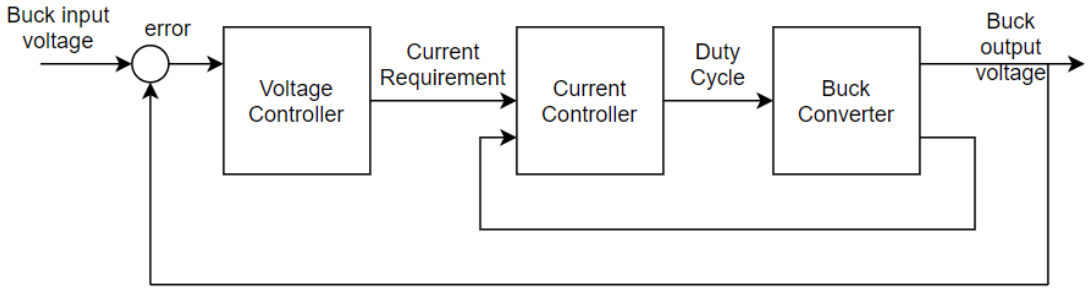
7.16 PV Characteristics of PV Panel



7.17 Overall Charging Algorithm



7.18 Other Buck Structure Attempted
[18]



7.19 Effect Of Partial Shading On Panel Output Voltage, Tested With Equal Load And Other Conditions Kept Approximately The Same

Output Voltage with no shading (V)	Output Voltage with $\frac{1}{4}$ of a Panel shaded (V)	Output Voltage with $\frac{1}{3}$ of a panel shaded (V)
5.362	4.549	4.192

7.20 Estimated Power Consumption of Rover [19] [20] [21] [22] [23] [24]

Component Name	Typical Power Supply Voltage (V)	Typical Power Supply Current (A)	Typical Power Consumption (W)	Component Name	Typical Power Supply Voltage (V)	Typical Power Supply Current (A)	Typical Power Consumption (W)
ESP32	3.30	0.50	1.65	Motor Control IC	-	-	0.89
FPGA	3.30	0.50	1.65	Optical Flow Sensor	3.60	0.05	0.19
Motor (*2)	6.00	0.32	1.92	Camera Package	-	-	0.20
Estimated Total Power Consumption (W)				6.49			

7.21Battery Nominal Output Power and Estimate Battery Run Time

Battery Nominal Output Voltage (V)	5.00
Maximum Discharge Current (A)	2.10
Rated Capacity (mAh)	5000.00
Maximum Power from Battery (W)	10.50
Estimated Supply Current (to the Rover)	1.30
Time that the Battery can power Rover (hrs)	3.85

7.22 Efficiency and Power of Power Electronics Interface

	Input Voltage (V)	Input Current (A)	Input Power (W)	Output Voltage (V)	Output Current (V)	Output Power (W)	Efficiency (%)
Horizontal	5.35	0.15	0.82	4.88	0.11	0.51	62.31
Lower Angle	5.72	0.11	0.62	4.56	0.10	0.44	71.63
Lower Angle	5.72	0.09	0.53	4.56	0.09	0.39	73.66
Upper Angle	5.72	0.08	0.47	4.56	0.08	0.35	73.97
Upper Angle	5.72	0.10	0.57	4.56	0.08	0.37	65.98
Improved	5.28	0.36	1.90	4.72	0.29	1.36	71.76

Average Efficiency (%)	69.89	Average Output Power (W) (before improved)	0.41	Average Output Power (W) (after improved)	1.36

8. References

[1] AVAGO Technologies, “ADNS-3080 High-Performance Optical Mouse Sensor,” ADNS-3080 datasheet, Apr. 2007.

[2] P. Chauhan, “Image Processing Using FPGAs”, 2021. [Online]. Available: https://medium.com/accelerated-image-processing-using-fpgas/image-processing-using-fpgas-8c703ef45808. [Accessed: 2-Jun-2022].

[3] S. Agarwal, “How Do We Calculate Distances of Objects Using Monocular Cameras?”, 2020. [Online]. Available: https://medium.com/all-things-about-robotics-and-computer-vision/how-do-we-calculate-distances-of-objects-using-monocular-cameras-67c4822c538e. [Accessed: 9-Jun-2022].

[4] “Why Do We Convert From RGB to HSV.” 2013. [Online]. Available: https://stackoverflow.com/questions/17063042/why-do-we-convert-from-rgb-to-hsv. [Accessed: 13-Jun-2022].

[5] Wikipedia, “Erosion (morphology).” 2021. [Online]. Available: https://en.wikipedia.org/wiki/Erosion_(morphology). [Accessed: 16-Jun-2022].

[6] ANALOGDEVICES, “Analog Filter Wizard.” [Online]. Available: https://tools.analog.c’om/en/filterwizard/. [Accessed: 1-Jun-2022].

[7] D. Raja, “Butterworth Filter: First Order and Second Order Low Pass Butterworth Filter,” 2019. [Online]. Available: https://circuitdigest.com/tutorial/butterworth-filters-first-order-and-second-order-low-pass-butterworth-filters. [Accessed: 3-Jun-2022].

[8] ARDUINO, “AnalogRead()”. [Online]. Available: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/. [Accessed: 7-Jun-2022].

[9] MICROCHIP, “MCP6001/1R/1U/2/4 – 1MHz, Low-Power Op Amp,” 2020. [Online]. Available: https://www.microchip.com/en-us/product/MCP6002. [Accessed: 5-Jun-2022].

[10] J. Lesurf, “The Envelope Detector”. [Online]. Available: https://www.winlab.rutgers.edu/~crose/322_html/envelope_detector.html. [Accessed: 6-Jun-2022].

[11] E. Pena and M.G. Legaspi, “UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter.” [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/uart-a-

hardware-communication-protocol.html. [Accessed: 2-Jun-2022].

[12] Mikegrusin, “Serial Peripheral Interface (SPI).” [Online]. Available: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all#whats-wrong-with-serial-portshttps://blogs.agu.org/martianchronicles/2011/09/28/how-do-we-know-where-rovers-are-on-mars/. [Accessed: 10-Jun-2022].

[13] “Node.js MySQL.” [Online]. Available: https://www.w3schools.com/nodejs/nodejs_mysql.asp. [Accessed: 14-Jun-2022].

[14] “ExpressJS Tutorial.” [Online]. Available: https://www.tutorialspoint.com/expressjs/index.htm. [Accessed:5-Jun-2022].

[15] G. Brown, “Key Solar Energy Measurement Terms: Irradiance, Insolation, TSRF, And More,” 2022. [Online]. Available: https://www.aurorasolar.com/blog/key-solar-energy-terms-irradiance-insolation-tsrf-and-more/. [Accessed: 27-May-2022].

[16] M. A. Ghasemi, H. M. Foroushani and M. Parniani, “Partial Shading Detection and Smooth Maximum Power Point Tracking of PV Arrays Under PSC,” in IEEE Transactions on Power Electronics, vol. 31, no. 9, pp. 6281-6292, Sept. 2016, doi: 10.1109/TPEL.2015.2504515.

[17] J. J. Nedumgatt, K. B. Jayakrishnan, S. Umashankar, D. Vijayakumar and D. P. Kothari, “Perturb and observe MPPT algorithm for solar PV systems-modeling and simulation,” 2011 Annual IEEE India Conference, 2011, pp. 1-6, doi: 10.1109/INDCON.2011.6139513.

[18] S. Motahhir, “Arduino Based MPPT Controller,” 2020. [Online]. Available: https://create.arduino.cc/projecthub/motahhir/arduino-based-mppt-controller-0560db. [Accessed: 2-Jun-2022].

[19] TOSHIBA, “Toshiba Bi-CD Integrated Circuit,” TB6612FNG datasheet, Jun. 2007.

[20] Espressif Systems, ESP32-WROOM-32d & ESP32-WROOM-32U datasheet, Nov. 2017 [Revised Mar. 2022].

[21] AVAGO Technologies, “High-Performance Optical Mouse Sensor”, ADNS-3080 datasheet, Apr. 2007.

[22] Intel, “Intel MAX 10 FPGA Device Datasheet,” M10 datasheet, Jun. 2018. [Revised Nov. 2021].

[23] Adafruit Industries, “DC Gearbox Motor – “TT Motor” – 200RPM – 3 to 6VDC”, TT DC Gearbox Motor 3777.

[24] OmniVision, “1/3.2n color CMOS 8 megapixel (3264 x 2448) image sensor with improved OmniBSI-2™ technology”, OV8865 (rev 1D) datasheet, Feb. 2014.