Imperial College London

ELEC50008 – Engineering Design Project 2 2020-2021

Mars Rover Project – Group 28

Authors:

Tianhao Bu, CID: 01723505

Meng Electrical and Electronic Engineering (Year 2)

Yujun Han, CID: 01726355

Meng Electronic and Information Engineering (Year 2)

Jingyi Liang, CID: 01708778

Meng Electrical and Electronic Engineering (Year 2)

Anqi Qiu , CID: 01733465

Meng Electrical and Electronic Engineering (Year 2)

Jingyi Wu CID: 01759400

Meng Electrical and Electronic Engineering (Year 2)

Sijun You CID: 01747112

Meng Electronic and Information Engineering (Year 2)

Instructors : *Adam Bouchaala*

Edward Stott

Zohaib Akhtar

Philip Clemow

Date of submission: 15.06.2021

Table of Contents

1. Introduction

The Mars Rover that can be controlled remotely and drive autonomously has been a staple of advanced manufacturing and received a great deal of attention over recent years. It is defined as an autonomous exploration vehicle that can achieve automatic obstacle avoidance and capturing images in various environments and is increasingly being utilized for entertainment, IoT, or even military purposes, such as search and rescue activity, uninhabited area monitoring, and planetary exploration.

The Mars Rover system is expected to complete some tasks by receiving commands from the Web app and give feedback to the remote app. It is also supposed to autonomously explore the local area using the sensors like the live camera and sending the relative information back. Therefore, the whole system integrates the power system, communication, computer vision, and rover control system.

This project aims to develop a lightweight portable autonomous rover which possesses the ability of remote operation and automatic obstacle avoidance. The rover is composed of five subsystems, including control, drive, vision, energy, and command subsystem, and each is capable of different functionality. Two modes are supported by the rover, one is command control mode, which allows the remote control of the rover by sending specific commands via MQTT protocol, the other is self-wandering mode, in which the rover drives autonomously and operates real-time obstacle detection and avoidance.

Dealing with varied terrain and background places extra demands on the vision subsystem, therefore, colour space conversion and spatial image filter are used to improve the accuracy of obstacle recognition. A powerful energy system is of great significance to the rover's performance as well, especially for the demand of driving the rover in uninhabited environment which does not have timely energy supply. Thus, Coulomb Counting algorithm and voltage look up table are used to calculate current battery stage and enables instant low-power warning. Additionally, solar panels are equipped and mppt tracking is applied which helps to achieve the maximum efficiency of power usage.

This report addresses the main architecture of the Mars Rover, the detailed design and implementation of the six subsystems of the rover, and how the rover performs under different conditions. Accurate results are acquired with the help of the video stream from the camera, and further possible improvements are explored to allow future study and development.

2. Overall Description

The main purpose of this section is to give a summary of the features and outcomes of the design. Mars Rover integrates five parts, including drive, energy, command, vision, and control subsystem, the connection between these modules is shown in Figure 2.1 below. With the collaboration of these subsystems, the rover can operate in two modes: the self-wandering mode where the rover runs autonomously and avoids the obstacles automatically using the vision subsystem, the command control mode, which requires manual actions such as pressing keys for controlling the rover movements or moving speed control bar to adjust the speed of the rover. During the rover's movement, the ball can be captured by the camera, and the relative distance to the ball is calculated and sent to the command side for updating the position of the balls in the real-time map.

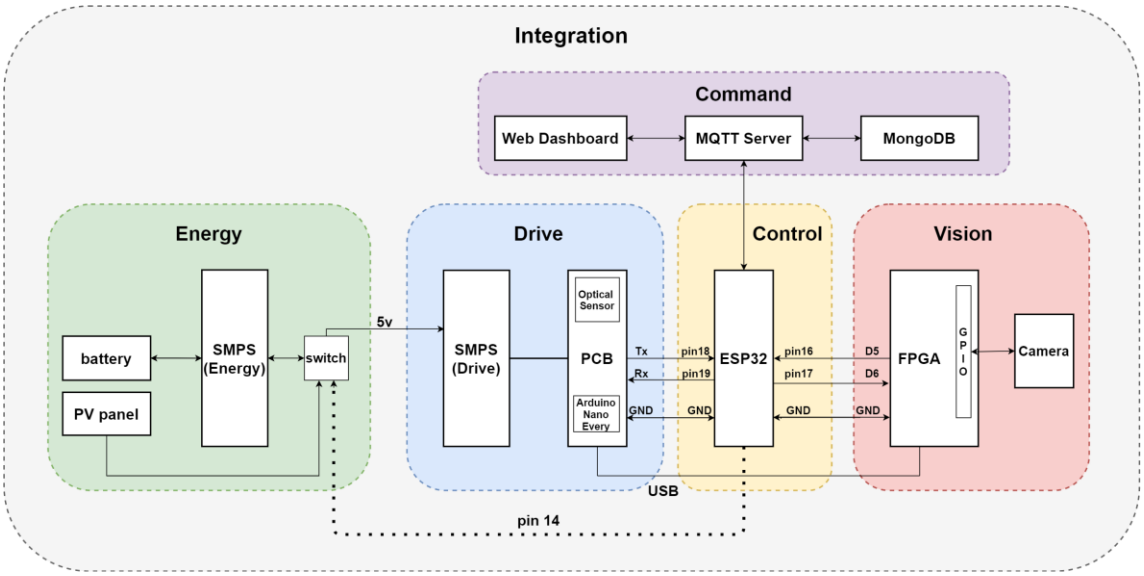


Figure 2.1, the overview of the physical connection between subsystems.

In addition, the Web app is also available to show the relevant information of the energy system, including state of charge, state of health, charging status, and remaining work time, they are displayed in a block on the Web app. The system is designed as the energy system supplies power continuously to the motor when it receives a 'high' command from the drive, and if it is not responding to the request, the battery is at a charging state. Figure 2.2 describes the structural components in the design, with the parameters for each component listed in the diagram.

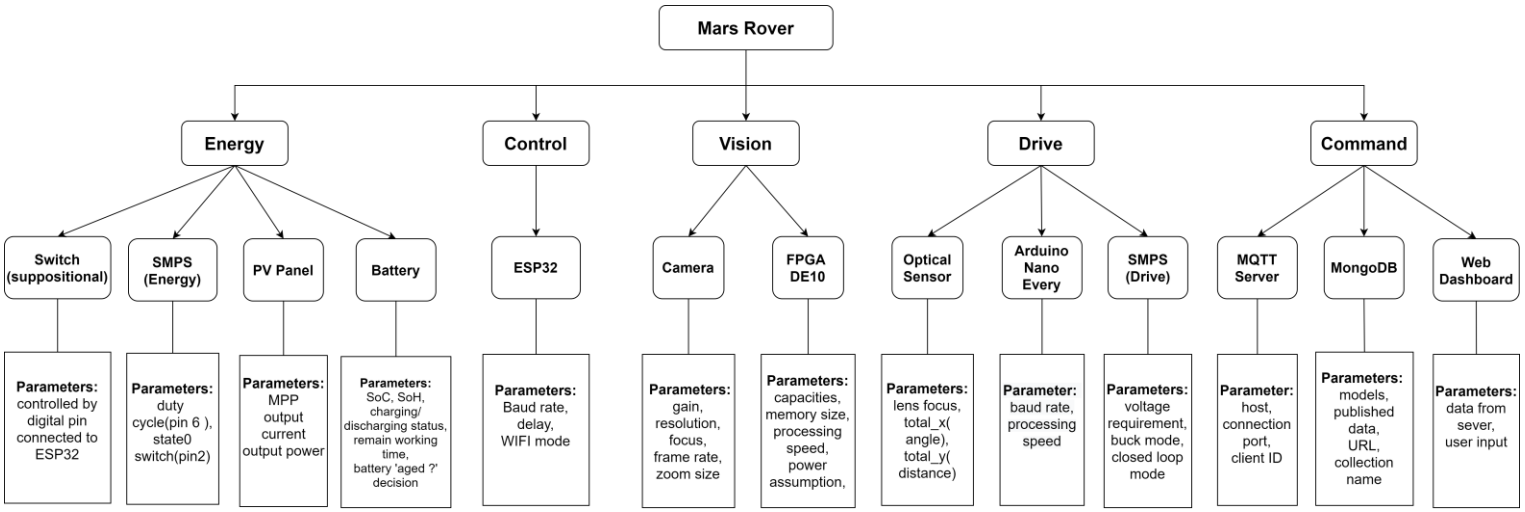


Figure 2.2, Structural Design of the Mars Rover.

In both operating modes, the data is collected by the vision subsystem and processed by the control subsystem before sending the information to the command subsystem, the location to process the data is carefully chosen to boost efficiency and reduce delays in the data transmission. To obtain real-time position of the rover, control and drive subsystem are connected via UART port, the raw data from the optical sensor will be sent from drive to command via control, the command subsystem processes these readings to find the coordinates of the rover and tracks its path, and the live position of the rover is displayed in the map. In self-wandering mode, the rover operates autonomously, and it turns left when there is an obstacle in front of it. The only difference between these two modes is that the autonomy of the rover's motion. Also, the real-time location tracking and path are still available in the Web app, with the location of the balls highlighted in the map. Both modes will be influenced by the surrounding environment, including background, lighting, terrain etc. Figure 2.3. detailly explains how the rover interacts with the environment and its life cycle.

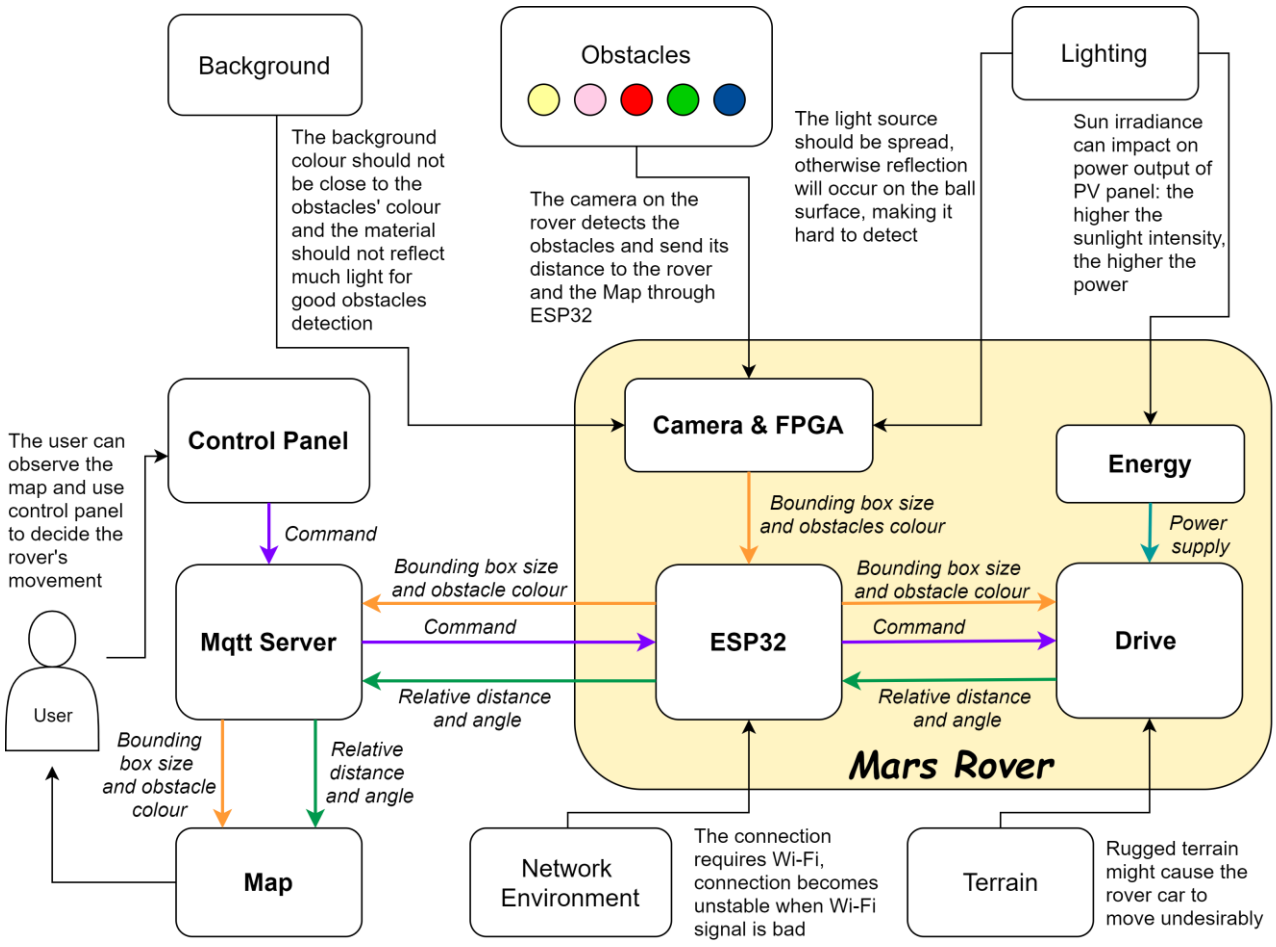


Figure 2.3, Context Diagram and the life cycle of the whole system

3. System Features and Requirements

Defining the functional and non-functional requirements is an essential part of the design since it is challenging at the start of the project when these requirements are poorly defined. For functional requirements, they specify the goal of the system and give directions on designing the product. These must be met to ensure the system working properly. Non-functional requirements state characteristics of the system, such as user experience and system features, they are built since functional requirements are fulfilled. Therefore, these requirements are discussed before progressing to the design and implementation, the details are provided in this section.

3.1. Functional & Non-Functional Design

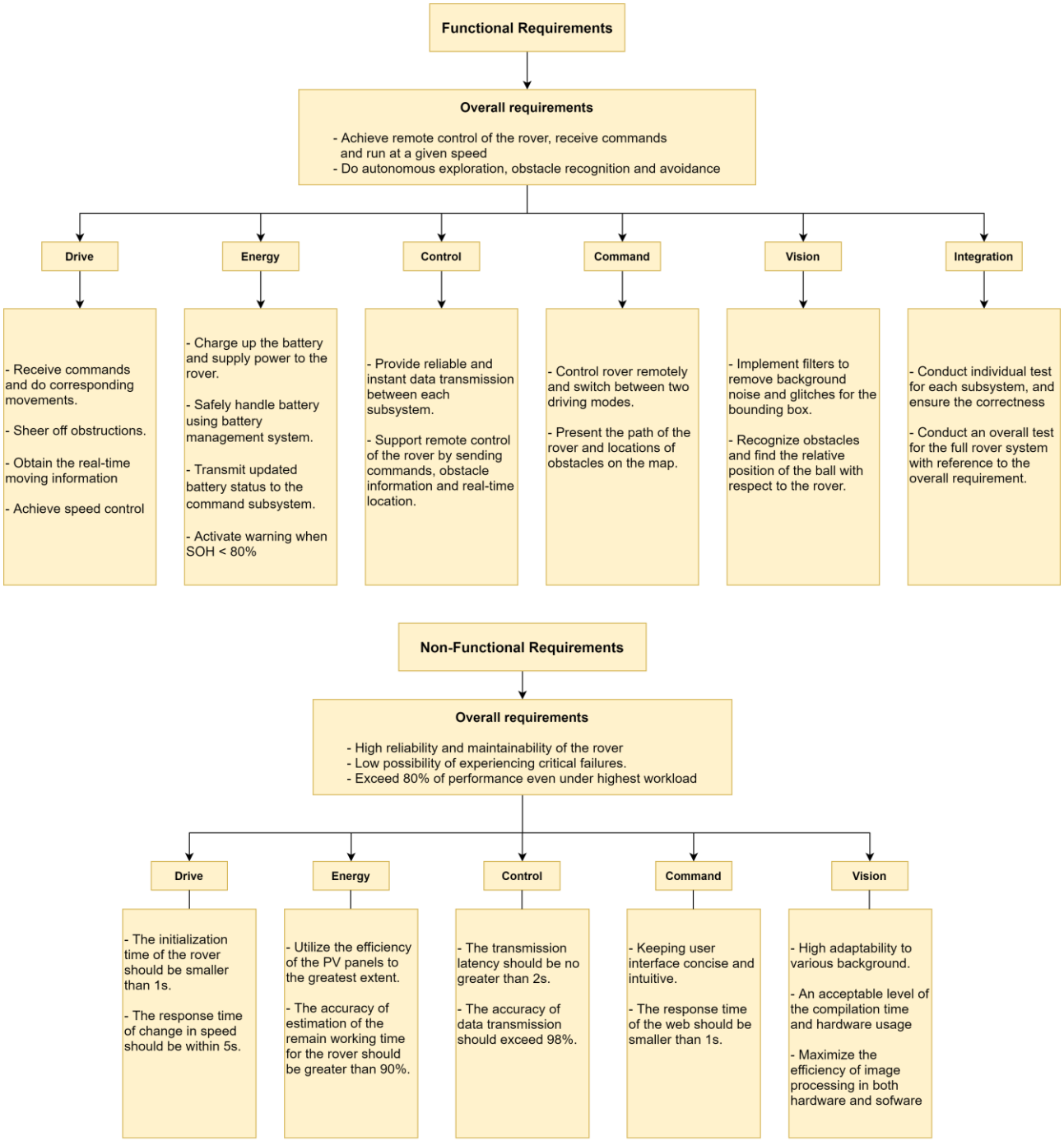


Figure 3.1, the diagrams for functional and non-functional requirements.

4. Design approach

4.1. Energy Subsystem

As the power source of the whole rover, characterizing the PV panels to extend their power output and design a Battery Management System (BMS) which provides a variety of functionalities that help improve the overall lifespan of the battery, including states estimation algorithms, are the main goals for energy subsystem,

4.1.1. Design Approach

Before starting the design, the priority process is deciding what outputs are required from the available inputs. Having investigated the datasheet of $LiFePO_4$ batteries [1] and PV cells [2] and discussed with fellows from other modules, the targets were confirmed: a constant 5V dc voltage should be applied to other subsystems from a nominal 3.3V battery input voltage, and the charging voltage from PV panels will be around 5V which is higher than the nominal voltage of a single battery.

The final circuit diagram for both charging and discharging of the batteries is shown in Figure 4.1.1 below: the synchronous SMPS acts as a buck to step down the voltage when charging the lower voltage batteries with PV panels and a boost to supply a higher 5V to other components. Ideally, the charging/discharging decision is made by a two-sided switch SW1 that is controlled by the control module (digital pin 14 from ESP32): SW1 is switched to the load when operation starts. However, without a switch available in the circuit, the charging and discharging process had to be tested with separate codes.

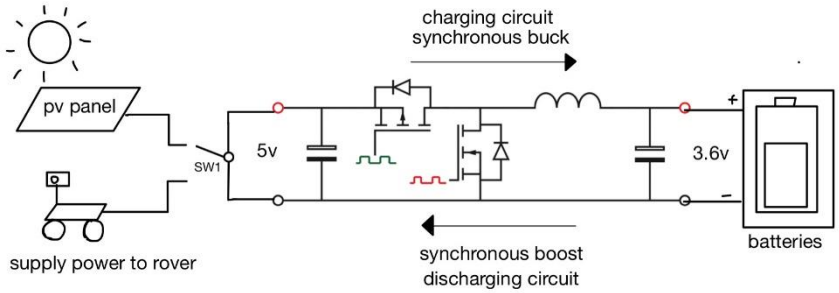


Figure 4.1.1, Charging and discharging circuit diagram.

In the next section, the process of investigating each component in this circuit will be fully discussed, starting from the characterization of the PV panel.

4.1.2. PV cell characterization

Photovoltaic (PV) panels have, for many years, been used as a method for harvesting energy and producing electricity.

In order to achieve maximum power output, electrical characterization of the cell as well as PV materials is performed as part of the research. The well-known function current versus voltage (I-V) and power versus voltage (P-V) curves of a solar cell have been implemented by incrementing the duty cycle of buck at the rate of 1% every 5 seconds to provide sufficient time for PV to stabilize, with the lamp placed 10cm to the solar panel: altering duty cycle is equivalent to changing the resistance of the load connected to the panel and therefore alternates its voltage and current which could be calculated from measurable V_{out} and I_{out} applying the buck equations:

$$V_{in} = \frac{V_{out}}{duty}, I_{in} = I_{out} * duty$$

Figure 4.1.3 below illustrates the measurement result curves (implemented with code 4-1 in appendix):

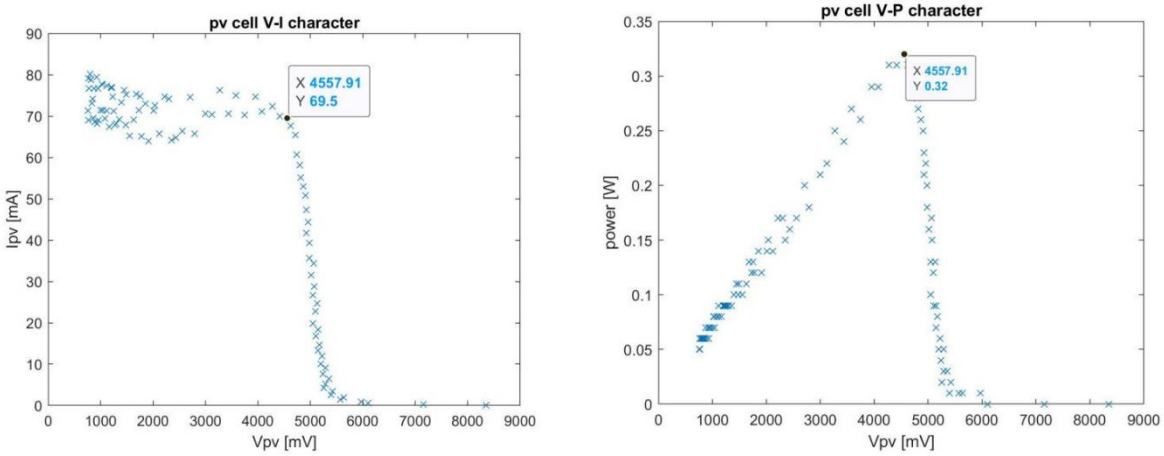


Figure 4.1.3, solar cell V-I and V-P curve measurement results (scattering in data points are measurement errors)

The curve shapes match expectation reasonably well and two important observations can be extracted: first, maximum power occurs at a particular voltage(~4.558V) which is the operating point; second, the current generated at MPP by a single cell is too low (~70mA) compared to the standard charging current of 250mA of the batteries. This reinforces the validity of using parallel PV cells in the formation of the final solar panel which boosts up the current.

The next section demonstrates the algorithm in tracking the maximum power point (MPP) and test results obtained with the solar panel built up by 4 cells in parallel.

4.1.3. PV panel Maximum Power Point Tracking (MPPT)

All solar panels have a maximum power point (MPP), which is the optimal condition where they produce the most electricity. The MPPT technology utilized was the conventional Perturb & Observe algorithm. See Figure 4.1.4 for the flow charge of the P&O tracking method.

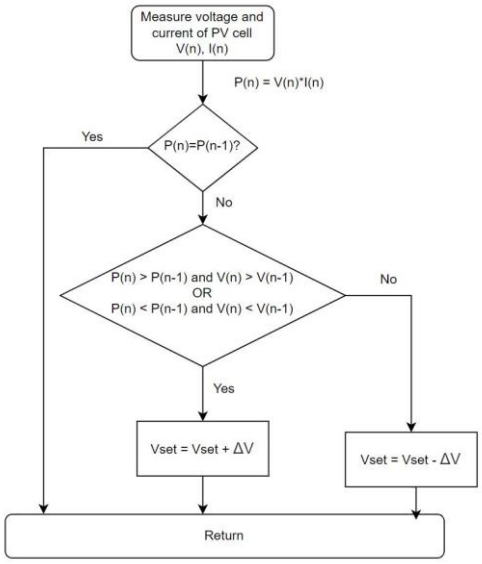


Figure 4.1.4 Perturb & Observe flow chart

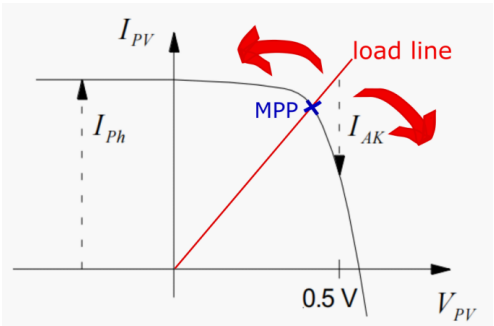


Figure 4.1.5 PV panel operating point

Basically, it is constantly watching the panels' MPP and then tweaking the panel's output voltage to optimize performance [3]. The Arduino implementation code is shown in code 4-2 in the appendix [4], where the duty cycle signal 'pwm_out'

is modulated at intervals of ±1.5% every millisecond. The gradient of the load line is then adjusted so that it intersects the V-I curve of the PV panel at MPP, as demonstrated in Figure 4.1.5.

The implementation result is presented in Figure 4.1.6 below: the power is successfully kept at 0.35W and the current is around 80mA, which isn't a large improvement compared to a single PV cell due to the limitation in using a Halogen bulb lamp to model sunlight: the irradiation concentrates on the one cell nearest to it. Furthermore, considering that the maximum solar irradiance on Mars is about 590 W/m^2 [5] and the total surface area of 4 solar panels =0.045m^2 [2], the peak power produced by the panel on Mars could reach 26.55W. As a result, considering the conflict in duty cycle control requirement for both MPPT and constant current (CC) charging, it was decided to keep MPPT dominant at low currents and for extreme cases where sunlight intensity is very high, the generated current might exceed the 0.5C standard of batteries, where MPP tracking is discarded and 250mA current PID controller is activated to charge the batteries, therefore prevent any danger caused by too rapid charging rate.

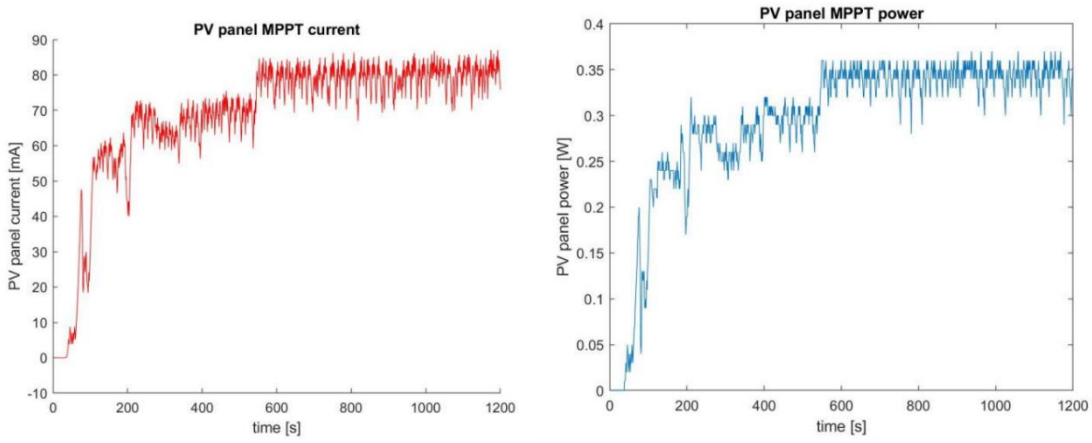


Figure 4.1.6, MPPT implementation result: Power and current output

Up to this point, the PV panel has been fully characterized and starting from the next section the design process of BMS will be demonstrated.

4.1.4. Battery State of Charge (SoC) estimation [6]

The efficient usage of a rechargeable battery requires the knowledge on how much charge is available in the battery, normally referred to as the state-of-charge (SoC) [7]. An accurate SoC indication improves the performance and reliability of the battery and ultimately increases its lifetime [8]. The general approach of SoC estimation during charging/discharging involves 2 parts: initial SoC indicated using a voltage look-up table and Coulomb Counting Method to continuously track SoC during the process. Therefore, the primary task is setting up the voltage LUT, where the concept of open-circuit voltage (OCV) of a battery should be investigated referring to the cell's 'Recovery Effect': a high load for a short period of time causes a high interface-concentrated gradient among electro-active species inside the cell, making the usable charge temporarily unavailable due to the lag between reaction and diffusion rates [9]. Thus, when the cell is allowed to 'rest' at zero discharge rate for some time, the voltage will rebound slightly from its original variation trend, as shown in Figure 4.1.7. The restored voltage those batteries terminates at is the desired OCV.

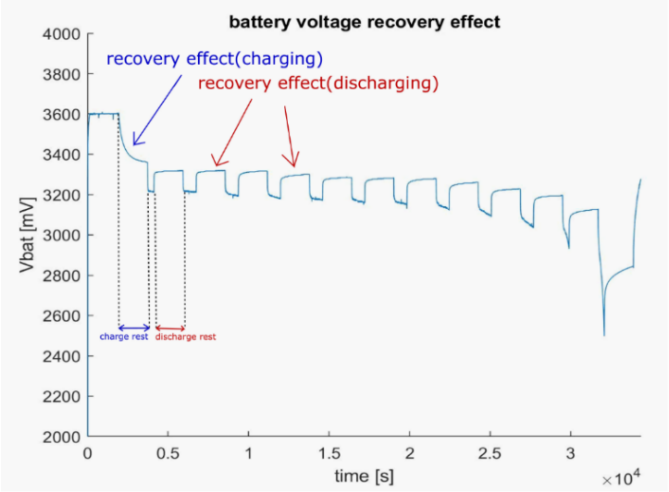


Figure 4.1.7,battery recovery effect (30min rest interval): when operation mode goes from charging to rest: experience voltage drop from original 3.6V to approximately 3.35V; when resting sequentially while discharging: the voltage that dropped temporarily goes back up

The OCV values show a pronounced and repeatable relation to SoC for most rechargeable battery types therefore are appropriate indicators of the battery's initial SoC. To construct a full table OCV values need to be determined for different SoC stages, where Coulomb Counting Method features. It accumulates current over time and adds it to the initial SoC as a fraction of total charge capacity, i.e indicating the percentage change in charge over the rated capacity.

The equation of SoC from Coulomb Counting Method is given by: $SOC(t) = SOC(t_0) + \frac{\int_{t_0}^{t} I_{bat} dt}{C_{rated}}$, where SOC(t_0) is the initial SOC, C_{rated} is the rated capacity, I_{bat} is the battery current [10] .

Therefore, accurate rated capacity needs to be measured. The 2 batteries used in the whole experiment, labeled Bat1 and Bat2, will each get charged to 3.6V 'apparent' voltage where they are assumed to be fully charged and then discharge continuously at a constant current of -250mA to their minimum 2.5V where they are considered to be empty. Total Crated obtained by accumulating current over the whole discharge period. For results see Figure 4.1.8. (Implemented using provided code Battery_Charge_Cycle_Logged_V1.1.ino).

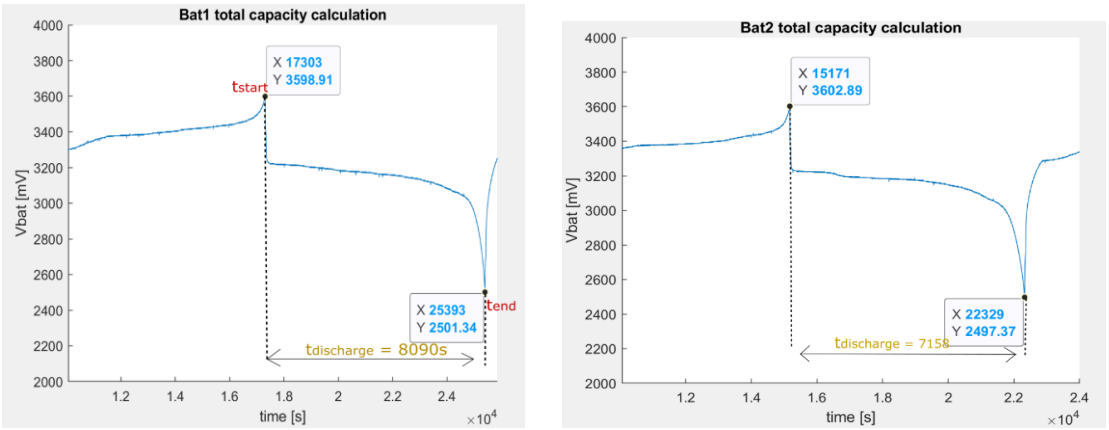


Figure 4.1.8 Bat1, Bat2 total capacity calculation

Bat1 discharge cycle: Crated_Bat1 = tdischarge1 * Idischarge1 = 8090/3600 * 250 mAh = 561mAh

Bat2 discharge cycle: Crated_Bat2 = tdischarge2 * Idischarge2 = 7158/3600 * 250 mAh =497mAh

Finally, a complete procedure in determining LUT using Bat1 as the reference is derived as follows:

- Step 1. fully charge the battery to 3.6V 'apparent' voltage and set SoC(t_0) = 100% ;
- Step 2. rest the battery for 30min before discharging: the voltage terminates at 100% OCV ;
- Step 3. start discharging at 0.5C constant rate and track SoC by Coulomb Counting; rest the battery each time SoC degrades by 10% to find out corresponding OCVs till SoC reaches 0%. (OCVs are measured during rest states by activating Rly and Mea ports of battery board through adding digital output pin D4 and analog input pin A1, which fully isolate the battery from the main circuit to prevent any current leakage.)

Figure 4.1.9 and Table 4.1.1 below show the state transition diagram and resultant SoC-OCV plot. (For core code see 4-3 in the appendix.

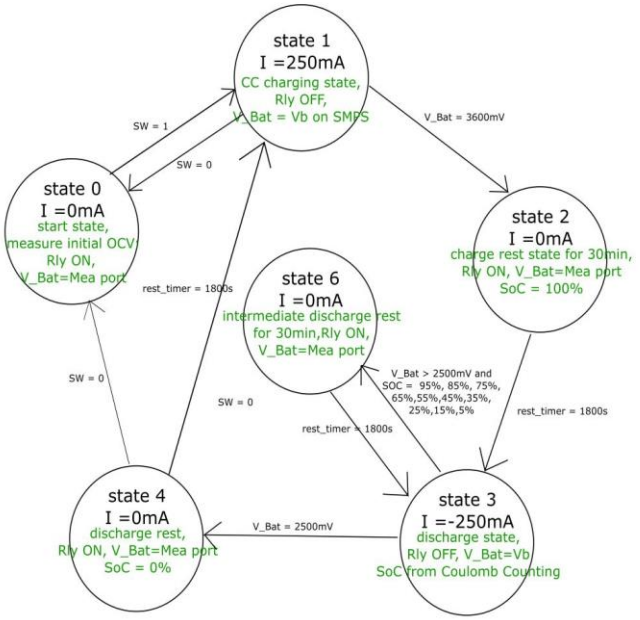


Figure 4.1.9 SoC-OCV implementation state transition diagram (Hidden transition: all state goes to state0 if SW=1)

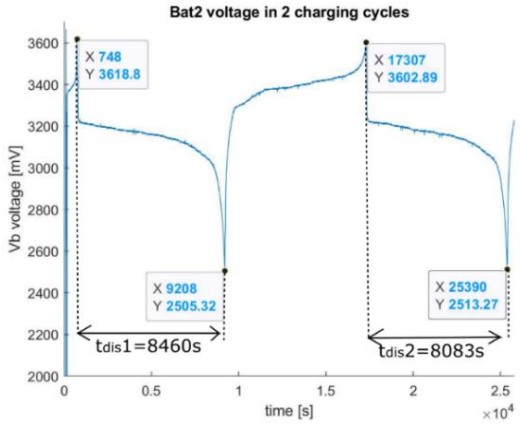
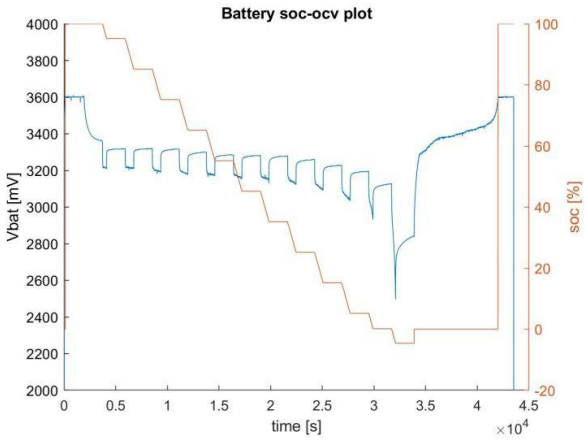


Figure 4.1.10 Bat2 aging effect.



SoC [%]	OCV [mV]
100	3340
95	3324
85	3320
75	3316.6
65	3288.7
55	3284.8
45	3284.7
35	3272.8
25	3249
15	3205.2
5	3113.8
0	2847.3

Table 4.1.1 SOC-OCV voltage look-up table

4.1.5. Battery State of Health (SoH) estimation

All batteries experience the aging problem, as justified in Figure 4.1.10 above where Bat2 shows degradation in total discharging time after 2 successive cycles. SoH is a measure of the degradation of a battery. It is customarily defined as

the ratio of its instantaneous maximum releasable capacity $Q_{max}(t)$ and its initial capacity Q_{new},

$$SOH(t) = \frac{Q_{max}(t)}{Q_{new}} \times 100\% \text{ [8]}$$

Generally, if SOH is below 80% the cell is at its end of life [7]. Therefore, tracking state of health of battery at the end of each fully discharged cycle is crucial to maintain the cell's long-term behavior. Its estimation method is similar to SoC which applies Coulomb Counting to calculate the total charge released. Further results will be shown in the following battery balancing section.

4.1.6. Battery Balanced Charging Method [11]

Due to the limitations on the number of output ports and the importance of keeping the status indicating LEDs, only 2 batteries can be used. The parallel combination of batteries faces the problem of unequal charging current between them, caused by the difference in their internal resistances. As a result, the batteries' OCV, i.e. SoC must be monitored regularly, and their charging current has to be adjusted in respect to the OCV difference between the two cells.

The initial thought was to apply passive balancing using the 'Dis' port. However, the rate of discharge of 'Dis' port was unacceptably low (especially when reaching the flat region of the discharge curve). An improved balancing method was the 'Low chasing High' Method described below:

1. Determine initial SoCs of batteries and balance by charging the lower voltage battery with full current from output port for a short time period and look at its OCV again after 2 minutes of rest whilst resting the higher voltage cell by turning on its 'Rly' and repeat until OCV difference between the batteries is acceptable (<10mV). As a result, the initial SoC of overall battery group takes the higher of the two.
2. Then move back on to the normal charging stage and repeats the checking process every 20min till reaching the mutual maximum 3.6V voltage at port B or SoC reaches 100%.
3. If Vb measured during the charging stage exceeds 3500mV, the batteries enter the region of rapid voltage rise and check the OCVs more regularly (every 5min)

Figure 4.1.12 demonstrates the state transition for this balanced charging method: the charging and rest time lengths in the balancing stage are determined experimentally to find the most suitable setting.

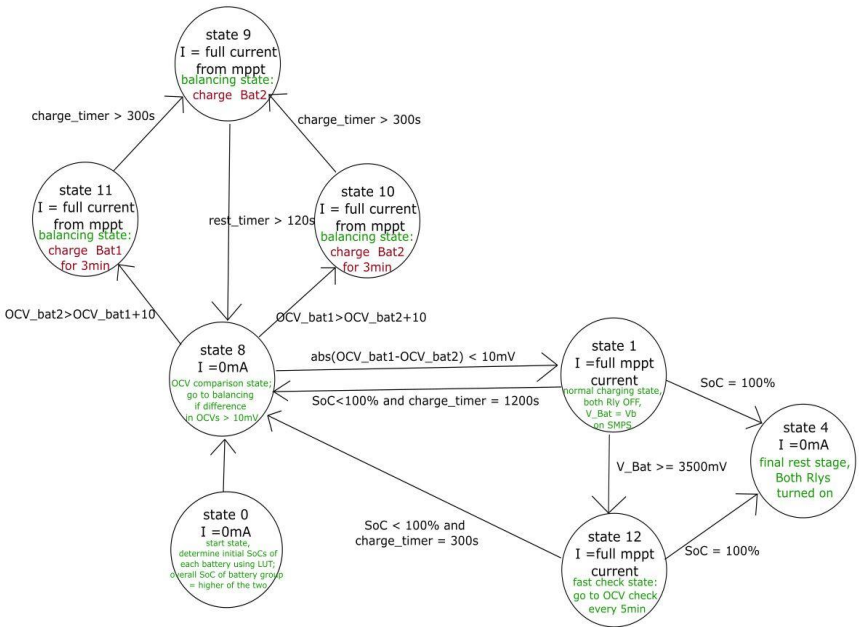


Figure 4.1.12 balanced charging method state-diagram

A few assumptions were made during this process:

- Assume the SoC-OCV correspondence in Table 4-1 is applicable to all batteries
- If initial OCV lies in between two values from the table then assume SoC varies linearly between the two corresponding SoCs and take the proportional value
- The total capacitance of the battery group = sum of each battery for parallel combination = 561+497 = 1058mAh
- The batteries are already after long rest before charging

Having charged the two batteries using the code charge_balance.ino that implements the procedure above, the SoC and V_{bat} measurement results are presented in Figure 4.1.13 below: the batteries are initially at around 95% SoC and after being fully charged the OCV of both cells terminates at around 3340mV which matches the 100% OCV value in Table 4-1 as expected (the sudden drops of voltages are due to the code setting: OCVs are set to 0V while balancing).

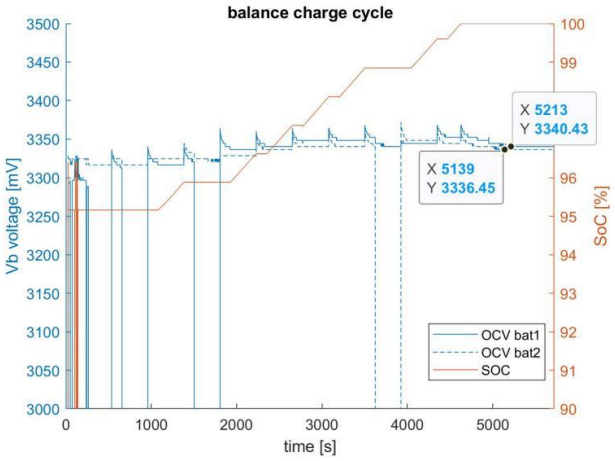


Figure 4.1.13 balance charging state OCVs of each cell and overall SoC variation

4.1.7. Final Battery Group Discharging Test

The batteries not only need to charge in balance, but also discharge. The good news is that having measured with a multimeter using different cells for several times, the two cells under parallel combination always discharge very evenly, i.e half of the total -250mA discharging current through each cell. Therefore, the first battery group discharging test is conducted without any balancing involved, and the result is shown in Figure 4.1.14.

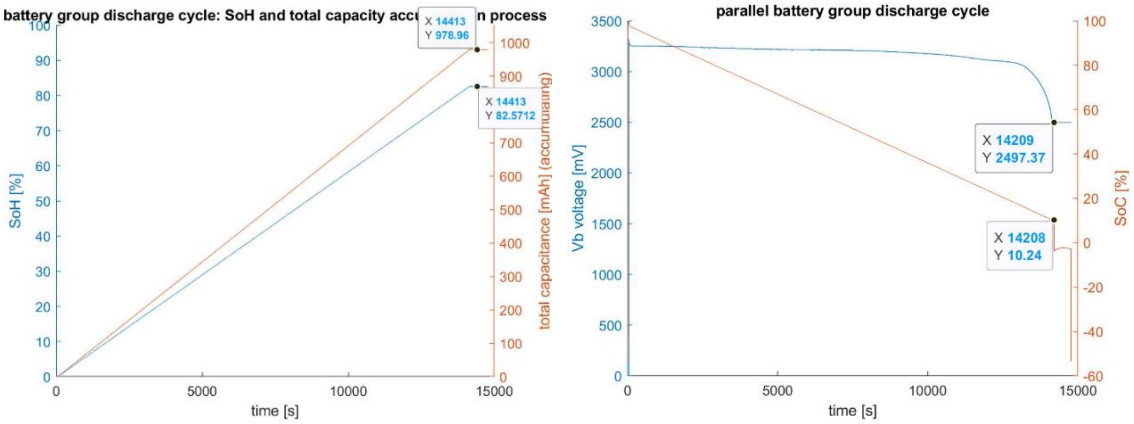


Figure 4.1.14 battery group full discharge cycle data, both Bat1 and Bat2 fully charged before discharging

We notice that the final SoH is only around 82% which is very close to the 80% red line. This verifies the aging problem

of cells where capacity has reduced by 18% compared to their brand-new state (each cell has experienced > 5 charging/discharging cycles) and SoC is no longer 0% when cells are fully discharged to 2.5V. This makes SoC estimation using Coulomb Counting very inaccurate and the degree of capacity degradation unpredictable. Whereas, by adding a few battery protection mechanisms, the chance of user damaging the batteries by inaccurate SoC information can be eliminated.

4.1.8. Data transmission to command module

In order to provide battery status information to the user interface, i2c communication is considered which links the output ports of Arduino to the ESP32 physically. However physical connection isn't possible as the integration module does not contain energy equipment, a data transmission test is conducted directly between energy and command module using the laptop as a medium. Here's the list of data we've decided to transmit:

1. SoC: tracked real-timely during the whole battery working process;
2. SoH: only generates non-zero significative value when discharging has finished;
3. A Boolean called 'aged': justify whether SoH is below 80%; if yes, aged = 1; otherwise, aged = 0;
4. Working mode of battery: 3 possible statuses: 'Charging', 'Discharging' and 'Rest';
5. Remain working time: provided during discharging process to indicate how long the rover can remain working; t_{work}= SoC*60/250mA [min] [12].

The data from COM port 7 (where Arduino data is outputted) is extracted into a .txt file using the port linkage software PuTTY and the latest line of data will be captured and transmitted by a JavaScript code imported, written by Command module fellow (details will be explained in command section later).

Furthermore, considering the issues of inaccuracy in previous sections, some extra settings were appended to the discharge code logic:

1. if V_{bat} is below 2.5V, SoC and t_{work} goes straight to 0% and 0min so the user will notice a sudden drop in remaining battery power at low SoC which prevents battery under-voltage;
2. if SoH is below 80%, the website will give a warning message reminding the user to replace the old batteries.

With all the mechanisms implemented, the energy module performed excellently.

4.2. Vision Subsystem – obstacle recognition

The vision subsystem plays an important role in detecting obstacles while the rover is moving. The main task for this part is to identify five ping-pong balls with different colours in real-time and predict the location of balls based on the size of the bounding box (a rectangle that surrounds the ping-pong ball). There are two components used, the D8M-GPIO camera and the FPGA DE10 board, D8M-GPIO collects the video data and sends it back to the FPGA, and the FPGA carries out the processing of the frame data and buffer the video data to the VGA for debugging purpose, the output video is available at the VGA output. The main analysis process is done within the EEE_imageproc IP component, in this component, every pixel is analyzed to predict whether it is part of the ball. The bounding box will be clearly shown in the video output. As shown in Figure 4.2.1, the ball detection algorithms contains mainly three parts, the conversion from RGB (red, green, blue) colour space to HSV(hue, saturation, value) colour space, applying an appropriate threshold on HSV values for the detect signal, and filtering the background noise. Once the bounding box is able to surround the ball, a moving average filter is applied to obtain a stable value at the end. The detailed implementations of these parts will be discussed later in this section.

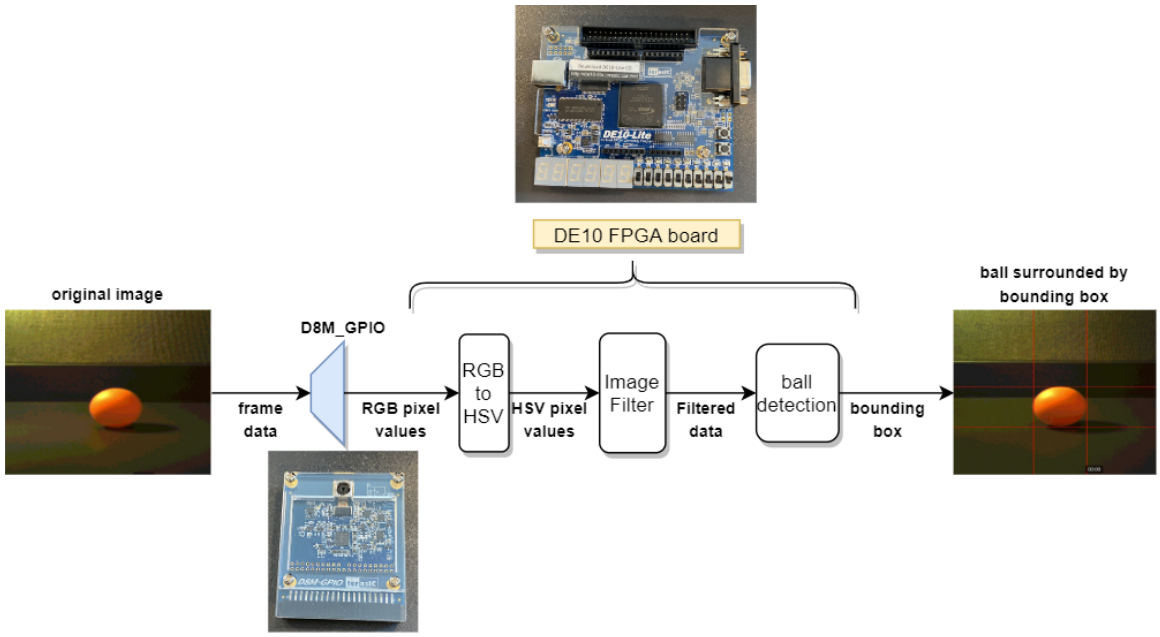


Figure 4.2.1, the overall flow for red ping-pong ball detection.

4.2.1. Design Choice

HSV Colour Model

The colour detection is mainly based on the HSV model, which stands for Hue, Saturation, and Value. Hue specifies the angle of colour on the RGB colour circle, Saturation is the amount of colour in the pixel, and value contains the information about the brightness of the colour [13]. Although the RGB (red, green, blue) are obtained directly from the camera data and the conversion requires more logic units, there are many reasons why RGB model is not preferred and the importance of introducing an efficient space for colour detection. Firstly, a pure colour segmentation normally won't appear in pixel values, which means that the blue and green components of the RGB value would be non-zero and also carry the information of the pixel colour, it would lead to non-precise results if only one component is analyzed. Also, the colour recognition of ping-pong balls is affected by illumination variations, this makes the colour detection even harder as the R, G, B of pixel values do not change linearly with

the brightness. Thus, a colour model transformation is required to combine the RGB colour information to a single variable and extract the brightness and saturation of the colour from RGB values, the HSV model was chosen to improve the tolerance of coloured balls detection under different light conditions.

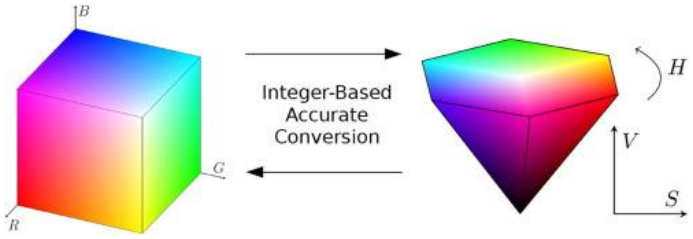


Figure 4.2.2, RGB colour space and HSV colour space illustrations[14]

Image Filter

The accuracy of the bounding box can be largely influenced by the noise in the background. The noise in the ball detection system is these pixels which are recognized as the coloured pixels randomly, resulting in a sudden change in the bounding box size. Therefore, a filter is needed to filter out the background noise while keeping the pixels of the ball as much as possible. Based on the requirements of the vision system, there are mainly two types of filters that are implemented and compared, finally, the average filter was chosen for its better performance at filtering the background noise. Since the main contribution to the noise is the random pixels from the background, the choice of the filer is based on how well the filter removes the noise while preserving the shape of the ping-pong ball.

Average Filter

The average filter is a linear and spatial domain filter, which means the pixels in the same frame are processed. There are two important properties that need to be considered, the size of the kernel and the values of each element in the kernel, an example kernel is shown in Figure 4.2.3. The pixel value in the center is calculated by multiplying the neighbouring pixels with the kernel element, the average of these pixels is placed in the center. It brings several effects to the image, it decreases the overall fault points in the image, thus the pixel values have lower contrast, thus the noise components are reduced. Since the average filter has a better performance at decreasing the differences between neighbouring pixels, it eventually reduces the chance to get a sudden change at constructing the bounding box. One drawback of this filter is that the overall image gets smoothed, and the edge of the ping-pong balls is affected. However, this influence is acceptable when the goal is to recognize the general shape of the balls, and it is still chosen for the satisfactory result at rejecting the background noise.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Figure 4.2.3 the example average filter kernel.

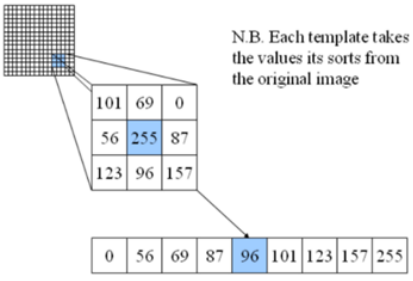


Figure 4.2.4, the median filter[15].

Median Filter

The median filter is a non-linear filtering technique, it is most effective at the removal of salt and pepper noise in digital images. The biggest advantage of the median filter is the preservation of the image edges, and the image

details are not influenced. Similar to the average filter, the median filter requires a filtering window to move along the pixels, normally a square one. and it works by replacing the central value with the calculated median pixel value, as shown in Figure 4.2.4. Thus, the process will replace the extreme pixels at the center of the filtering window, and the noise is removed. The reason why it is not chosen is that it gives a less satisfactory result for removing the random noise, and it requires more computation to sort and select the pixel values, resulting in an increase in logic units.

4.2.2. Implementation

HSV Conversion

Normally, the Hue which defines the colour has the variation from 0 to 360 degrees, Saturation and Value ranges from 0 to 100%. The range from 0 to 100% is not fit for the hardware implementation, thus it is changed to 0 to 255 during the colour space conversion. The Value is the maximum value for Red, Green, and Blue, the saturation is calculated by 255 multiply with the ratio of D(difference between the maximum and minimum value in RGB) and M(maximum value in RBG), and the Hue is dependent on the relationship between the M and RGB values, as shown in Figure 4.2.6. Since the Verilog language does not support the fixed-point calculation, a function that carries out the fixed-point division is invented to calculate the saturation and hue values. Even Hue and Saturation only need 9 bits and 10 bits, they are both assigned 32 bits for better precision. To shorten the long critical path introduced by the long division, the HSV values are put in the register, this introduces one clock cycle delay between the detection signal and x-y coordinates of the pixel, it can be eliminated by setting an offset at edge coordinates of the bounding box.

Figure 4.2.5, the HSV ranges for each coloured ball under demo environment.

Ball colour	Hue	Saturation	Value	Balls	Hue	Saturation	Value
Red	{0,18}	{130,255}	{60,155}	Green	{145,160}	{80,255}	{20,255}
Blue	{220,280}	{80,255}	{30,255}	Yellow	{45,65}	{5,120}	{80,255}
Pink	{0,20}	{5,120}	{60,255}				

$$M = Max(Red, Green, Blue)$$
$$m = Min(Red, Green, Blue)$$
$$D = M - m$$
$$Value = M$$
$$Saturation = \begin{cases} 255 * D / M, & if\ M \neq 0 \\ 0, & if\ M = 0 \end{cases}$$

$$Hue = \begin{cases} 0, & D = 0 \\ \frac{G-B}{D} * 60 + 0, & M = R\ and\ G >= B \\ 360 - \frac{B-G}{D} * 60, & M = R\ and\ G > B \\ \frac{B-R}{D} * 60 + 120, & M = G\ and\ R >= B \\ 120 - \frac{R-B}{D} * 60, & M = G\ and\ R < B \\ \frac{R-G}{D} * 60 + 240, & M = B\ and\ R >= G \\ 240 - \frac{G-R}{D} * 60, & M = B\ and\ R < G \end{cases}$$

$$Ranges:\ Value\ [0 - 255],\ Saturation[0 - 255],\ Hue\ [0 - 360]$$

Figure 4.2.6, formulae used for the Hue, Saturation, and Value

Pixel Buffers for Filtering

For both filtering methods, at least two rows of pixel values are buffered throughout the entire frame to complete the filtering window, as shown in Figure 4.2.7, the blocks highlighted in blue are the values stored in the external memory, the yellow blocks are the elements within the filter kernel. When the new values are fetched into the kernel

registers on the right side, the values at the left side are being shifted into the buffer memory, as illustrated in Figure 4.2.7. Since only the detection of the pixels is required, replacing the pixels with the new value is not essential for giving the solution. Therefore, instead of buffering all RGB values for one single pixel, only one single bit is stored, which indicates whether the pixel is recognized to be the ball colour. The pixel buffer and the kernel filter still apply here, the only difference is that less storage is required. There are several reasons for storing this bit, it reduces the computation cost and the size of memory needed to store the pixels. The detailed implementation and how this method works on filtering will be discussed later in this section.

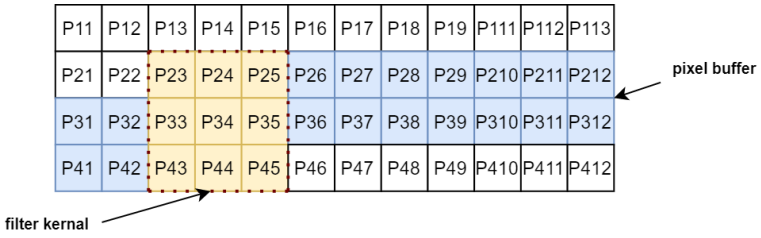


Figure 4.2.7, the pixel buffer

Filters (Image & Bounding box)

For the spatial image filter, an improved version of average filter has been applied to the image processing, where there is only one bit stored in the pixel buffer, and whether the surrounding pixels can meet HSV threshold is set before storing this single bit. As shown in Figure 4.2.8, the ball detect signal is only high when surrounding pixels are high, this is to follow the basic theory of the average filter. If any value in the filter kernel is smaller or larger than the threshold, the calculated pixels still do not meet the detection criteria. Since the RGB is fragile under certain circumstances, the detection is based on the colour ranges in the HSV model. Thus, the only difference between the traditional average filter and the one implemented in this system is that the order of applying detection criteria before or after the image filter. Also, the detection bit of the central pixel will not change and will be parsed into the pixel buffer. Since filter only works if there are two rows of pixels information are available, the first two rows and first two columns. All the pixels within the shade region will be filtered, thus most pixels in one single frame are able to give correct results. The functional correctness of this filter will be discussed later in this section.

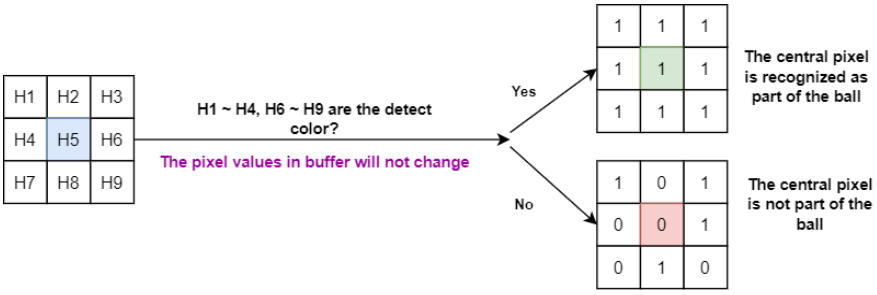


Figure 4.2.8, the implementations for the averaging filter

Due to the fluctuations in the pixels, the bounding box or its size changes rapidly in a short amount of time. To remove the variations, a moving average filter is applied before sending the information to the control side. This filter is chosen to be implemented in hardware to get a stable output for the bounding box size, and a shift register is used to store ten past values. When the SOP is high, all these registers are initialized as zero, and the new value would be buffered into the shift register and other values in the shift register are updated as well.

4.2.3. Testing

HSV Conversion

To ensure the correctness of the conversion, the codes used for the conversion are moved into a separate file and the outputs are evaluated by looking at the waveforms of each data bus. The results are also compared with the online conversion tool available at [b], results of five sets of RGB values are tested to ensure the correctness of the conversion diagram, as shown in Table 4.2.1. During the testing procedure, it is noticed that there are still some discrepancies between the outcomes of the module and the real results. However, these results are still considered to be accurate and acceptable for the calculation in hardware design. The colour detection of balls is built on the correctness of this conversion for determining the appropriate range for each colour.

Table 4.2.1, the list of values used for testing the correctness of the HSV conversion.

	Results			References			Percentage error / %		
Red, Green, Blue	Hue	Value	Saturation	Hue	Value	Saturation	Hue	Value	Saturation
{100,100,200}	240	200	127	240	200	126	0	0	0.00794
{125,240,100}	110	240	148	109	240	149	0.00917	0	0.00671
{230,100,200}	314	230	143	314	239	144	0	0	0.00694
{140,230,100}	102	230	143	102	230	144	0	0	0.00694
{100,240,145}	139	240	148	139	240	149	0	0	0.00671

Colour Detection & Image filter

The colour range for the ball is initialized with certain values based on the colour picker online and tuned by the trial method. The range changes slightly every time until a clear output from VGA video output and constant bounding box coordinates are observed. The pixels which are part of bounding box of the ball are highlighted with corresponding colours. The steps followed to tune a colour are:

- Using the HSV colour indicator to find an approximate range for Hue, Saturation, and Value
- Given the saturation a slightly larger range, and change Hue first until most pixels of the ball are recognized.
- Adjust the Saturation value by looking at how the image process changes when light condition changes.
- Value is modified based on how good the process until the light or black background noise is removed.

The test is carried out under a divergent light, and the direct light source above the ball is avoided to minimize the reflections at the surface of the ball. In addition, detecting the balls that have a similar HSV range is also tests. For example, the pink ball and red ball are only different in the saturation range, they are put together to ensure there is no mix between their colours. The quality of the image filter is determined by how well the background noise is removed when trying to detect a coloured ball. All colours have been tested on different grounds to guarantee the efficiency of the image filter, and Figure 4.2.9 and Figure 4.2.10 show the detection for all five balls.

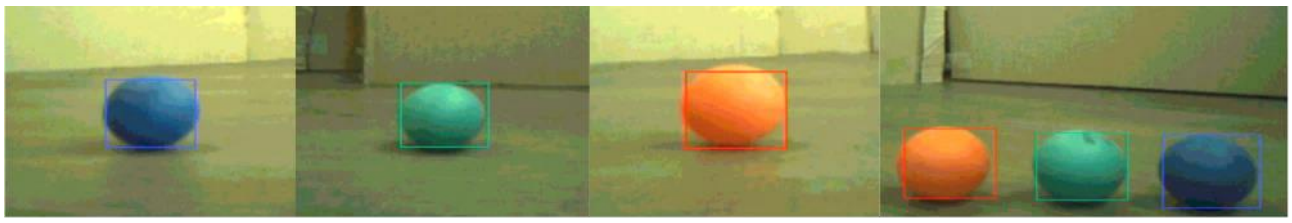


Figure 4.2.9, the images for blue(1), green(2), red(3), and three balls(4) detections.

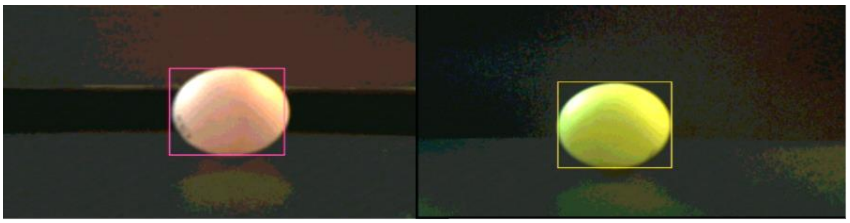


Figure 4.2.10, the images for pink(1), yellow(2) ball detection.

Bounding box

The tests for the bounding box consist of two parts, the stability of the reading when both the camera and the ball are stationary, and whether the bounding box has a steady change when the camera is moving. The first set of tests is carried out with some setup, both the camera and the ping-pong ball are placed on the surface which the positions of the camera and the ball are marked. This test only guarantees that the bounding box value is stable and reliable, and it would be sufficient to use just one coloured ball since the only difference is how the colour ball is influenced by the background, the graph representing the bounding box size data is shown in Figure 4.2.11. The second test is there to model the situation when the camera is trying to capture the balls while it is moving at constant speed with the Rover. The ball is at a location 100 cm from the camera at the beginning, then the camera is held to move towards the ball, the box sizes are captured from the nios2-terminal into a single file. Also, to ensure the validity of the testing result, this test is repeated for all five ping-pong balls. The relationship between the size of the bounding box and the distance recorded, and these data are used to find a mathematical relationship between the bounding box size and the distance. To reduce the hardware usage and compilation time, the bounding box size is directly given to the control side, and the calculations are done in ESP32, which supports more math functions, the detailed graphs and explanations are discussed in the control subsystem.

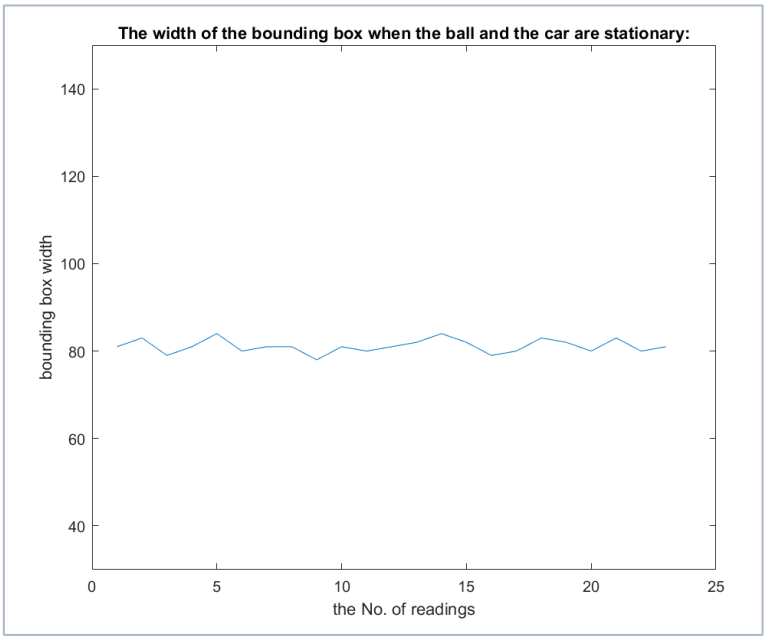


Figure 4.2.11, the variations in bounding box size when the camera and the ball are stationary.

4.3. Drive Subsystem

The focus on the drive subsystem is to control the motion of the rover using Arduino nano every, based on the command requirement. Drive consists of 2 modes, the first one is self-wandering mode which the rover detects the obstacles via vision subsystem and avoids the obstacles by rotation, by default the rover keeps moving forward. The second mode being the man-controlled mode, drive receives signal from command subsystem via control subsystem and takes corresponding actions based on the signals it receives. The optical sensor is initialized immediately after the codes being uploaded; it sends relative motion indicators (x,y, x for rotation and y for straight movement) with respect to the current rover movement.

4.3.1. Speed control

The speed of the motor is controlled by the DC voltage produced from the B terminal (vb in the Arduino) on the SMPS (SMPS is under closed-loop buck mode). From the original code the vref (reference voltage) value is defined in the sampling() function, together with the closed loop PID control (by feeding the voltage error ev into pidv) it adjusts the speed of the rover by changing the duty cycle of the SMPS (Vin is fixed at 5V thus vout is changing). To set the speed in a software perspective, (without tuning the blue knob physically on the SMPS) vref is set to be a parameter independent from the sensor ports, by giving vref a value manually in a range from 1.35 to 4 approximately, with 1.35 or below the speed is approaching 0 and 4 gives the maximum rover speed. Based on the command requirement, the range is adjusted from 1.4-3.9, a slider is used to change the rover speed, 0.1 at a time corresponding to 25 speed steps.

4.3.2. Optical sensor

The optical sensor gives 2 converted values of total_x and total_y; their values represent the relative movement of the rover, with a decrease in y (getting more negative) shows the rover moving forward and increase in y shows the rover moving backward. The same strategy applies for x, x measures the amount of rotation of the rover. The multiplication by 10 and the division by 157 convert the sensor readings into mm, by taking away the *10, everything is calibrated in cm instead.

4.3.3. Energy connection

In the void loop, drive defines a parameter as ‘count.’ This parameter will be high if the Arduino codes are running, it is sent to energy to indicate that the rover is consuming energy, requiring the battery being discharged.

4.3.4. Self-wandering mode

The approach was to let the rover move forward with respect to the camera face direction by default, the vision part updates the distance from the camera to the obstacles synchronously and if this becomes smaller than a manually set reference (means the rover gets close enough), the rover switches state and performs an anticlockwise rotation, when rotation finishes rover keeps on moving forward. (Note: the backward motion was not considered since the camera was not able to detect obstacles around 360 degrees, the optical sensor can only give the relative rover coordinates with respect to the original start point to indicate movements only) The state machine used in the Arduino nano is shown in Figure 4.3.1.[16]

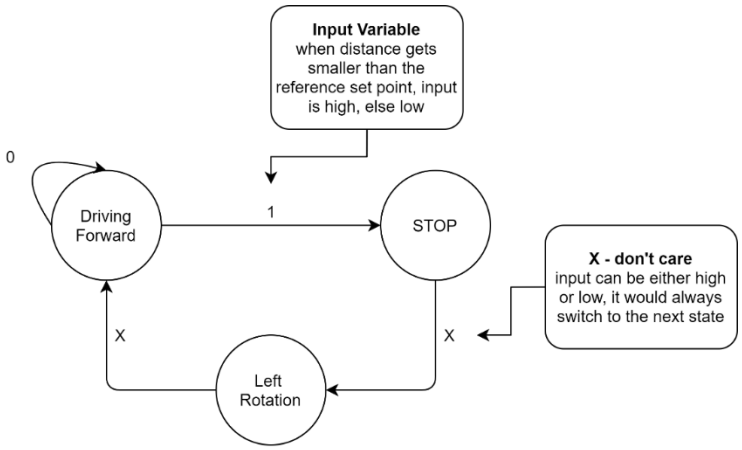


Figure 4.3.1. State machine in Arduino.

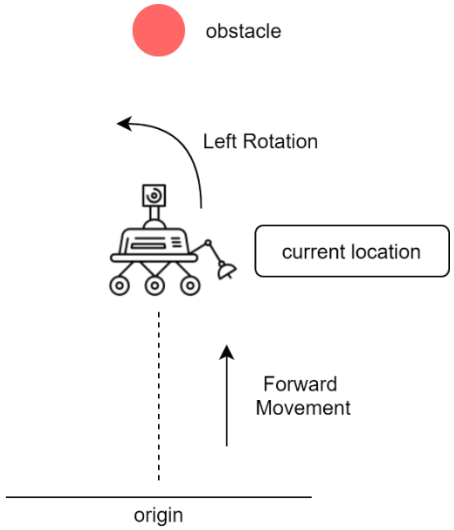


Figure 4.3.2. Rover moment logic

The state machine consists of 3 states: Forward, stop and rotation. Initially the rover is in FORWARD state, until the distance restriction is triggered. Rover then moves into STOP state, the rover is forced to stop and very quickly jumps into ROTATION. In ROTATION the rover rotates in an angle based on the total_x angle measurement and eventually falls back into FORWARD state, this process repeats over time.
In each state, especially with the rotation, the action is forced until the state is completed, the actions are not affected by loop interruptions.

4.3.5. man-controlled mode

Command sends 5 characters 'W' 'A' 'S' 'D' 'P' to control, the characters correspond to forward, left rotation,backward, right rotation and stop; whenever a key is pressed, the control subsystem uses Serial.write() to export the characters to drive (drive receives it with Serial.read() in order) and drive takes the corresponding action. For 'W', 'A', 'S' ,'D', digitalWrite(pwmr,) and digitalWrite(pwml,) are both set to high in order to enable the motors being driven, when the keys are released, character 'p' will be sent continuously and digitalWrites here will be both set to low just to cut the motor power off so that the rover no longer moves. This method is taken instead of setting vref to be below 1.35 is because the time taken for the rover to stop from a certain speed using vref method is long in comparison, for the enable method the stop action is almost immediate; but other than that, vref speed control is still used while the rover is moving.

4.3.6. Mode switching

A global variable with type char called 'direction' is used as a buffer to hold the direction and mode indicators before a change is made, when the variable is assigned as 'M' self-wandering mode is taken, (in this mode 'M' will be sent continuously until a switch in mode takes place) if the variable is assigned with 'W', 'A', 'S', 'D' and 'p', they correspond to direction commands which means man-controlled mode takes place, an else condition is inserted to ensure the rover does not end up in the undefined state, in this state the rover is set to stop using digitalWrite. Without the else condition, in the initial testing section the rover tend to fall into a left rotation movement as no signals were being sent from the command subsystem which the rover does not know which mode it is in until a key is being pressed.

4.4. Inter-module Communication – Control Subsystem

The inter-module communication is mainly based on the control subsystem, which works as a bridge connecting all the other subsystems based on ESP32 microcontroller. The ultimate goal of the inter-module communication is to provide stable and reliable communication channels, which is the base of the remote control of the Mars rover.

This section mainly describes how to construct the inter-module communication from problem definition to actual implementation, involving the way to test, the problems met during testing, and how were the problems being solved.

4.4.1. Design Approach

The main task of the control subsystem is to provide a smooth and reliable communication environment and ensure the accuracy and efficiency of the data transmission. Several problems should be considered before actual implementation. Firstly, which protocols should the esp32 use to communicate between each subsystem. Secondly, in what form should data be transmitted, for example, string, int or char, and what is the content of the data.

The communication among control, drive and vision subsystem are all based on physical pin-to-pin connections, and related protocols including UART, I2C and SPI. Ultimately, UART, a hardware communication protocol that uses asynchronous serial communication with configurable speed [17], is chosen to support the communication between esp32, FPGA and motor, as it provides full-duplex communication and is easy to be implemented.

Two-way communication is expected between the drive and control subsystem, where esp32 receives the location of the rover and the charge signal from the motor and sends user commands, the distance to the obstacle to the drive subsystem. The relative turning angle and distance moved of the drive subsystem is sent as the web needs this information to calculate and plot the driving path of the rover. This could be in the form of two integer type turning angle in degree and moved distance. Commands from the web can be sent in the form of string type, and these commands enable the remote control of the rover.

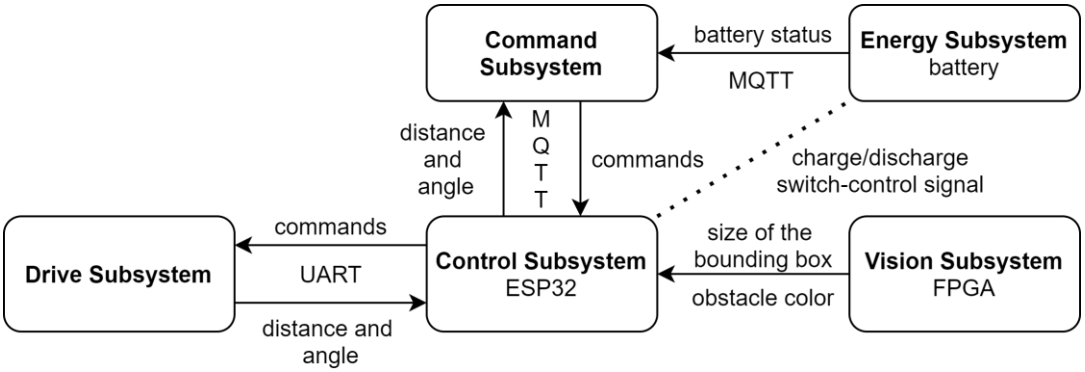
Additionally, the control and command subsystem should have two-way communication as well. It is desired that the control subsystem can receive the moving commands from the command webpage, as well as sends the current location of the rover and the location of the obstacle out to the remote server. Hence, remote control of the rover can be achieved, and the user could know how the rover moves. Based on such requirements, MQTT protocol is chosen to provide a stable connection, which is an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with minimal network bandwidth.[18]

Moreover, vision subsystem should send the size of the bounding box, and the colour of the obstacle to the esp32, as the calculating process of the actual distance between the obstacle and the rover would be done inside the control system. The colour can be transmitted in char type, and the size can be in int type. By finding the relation between the distance and the size of bounding box, the actual distance can be worked out and used for obstacle avoidance.

Furthermore, the battery should know when to charge the motor, therefore a pin-to-pin connection will be constructed, when the signal from the drive becomes high, the esp32 will write a HIGH signal to the corresponding pin in energy system. Also, the user should know the current status of the rover's battery, thus energy subsystem should send out the battery power, mode of the battery and the possible working time.

However, as the integration module does not have the battery component, and the battery is also too heavy to equip. Therefore, for the testing purposes, the energy subsystem sends out the data to the serial port of the computer and

then via a local MQTT client to directly send the data to the command subsystem, without transmitting through the control subsystem. Detailed explanation is presented in 4.1.9.



Overall, the structural design of the inner-module communication is shown below:

Figure 4.4.1, block diagram of the inner-module communication.

4.4.2. Implementation

4.4.2.1. Communication with FPGA

The communication between the control and vision subsystem is mainly about the transmission of the size of the bounding box and the colour of the obstacle as explained before.

On the control side, HardwareSerial library is used to map the serial port1 with pin16 and pin17, which is responsible for receiving and transmitting data, respectively. Serial.available() function is used to check whether there is any incoming serial data stored inside the receive buffer. If it is larger than 0, it shows that the data is arrived and can be read using Serial.read() function. On the FPGA side, RS232 component is added, and externally connected to the ARDUINO_IO[5] and ARDUINO_IO[6]. Based on the library "altera_up_avalon_rs232_regs.h", and the function IOWR_ALT_UP_RS232_DATA(RS232_0_BASE, <DATA>), the FPGA can send the data out via the assigned pins.

The data to be transmitted is the bounding box size and the colour of the obstacle. By recording the data of the box's size under different distances, one can come up with a diagram describing the relation between the distance and the size of the obstacle, is shown in Figure 4.4.2.

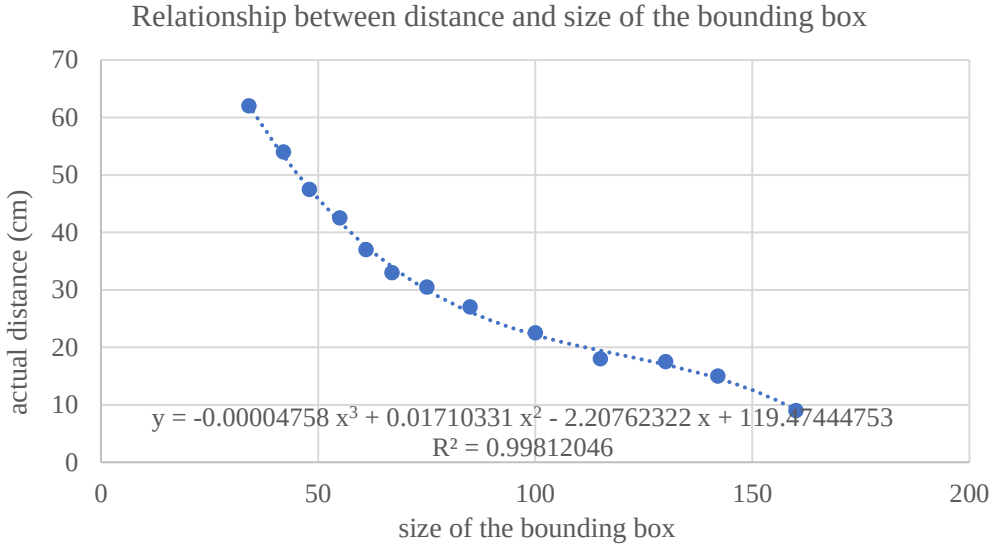


Figure 4.4.2. The relationship between the bounding box and distance from camera to the ball.

An approximated function describing the relationship is shown inside the diagram, by reading in the size of the bounding box, the actual distance from the rover to the obstacle is worked out, which is an important parameter for real-time obstacle avoidance.

4.4.2.2. Two-way communication with motor

For the transmission of real-time location from drive to the control subsystem, two values x and y coordinates are transmitted. Due to the feature that the coordinates can be larger than 256 and can be negative numbers, they are transmitted together inside a c string splitting by a comma, using function int snprintf (char * s, size_t n, const char * format, data). Also, to reduce the chance of data loss and improve the accuracy of data, two indicators '<' and '>'are used at the start and end of the string. The transmission code is shown below.

```
char location[40]; //defined as a global variable
/* Content of the code */
snprintf (location, 40, "<%i,%i>", x , y); [19] //print distance and angle into a c string
Serial1.write(location);
```

At the receiver end, characters keep reading in, when the start indicator is recognized, boolean start becomes true.

```
boolean start = false;
char tmp = Serial2.read();
if (tmp == '<') {  //indicator detected
      start = true;
}
```

Then the following incoming chars are stored inside a char array named location, until the end indicator is recognized.

```
char location[40];              //char array to store the location information
int i;                          //count
if (start == true) {            //set true above when start indicator is recognized
      if(tmp == '<' ){          //if start indicator appears inside string due to error
         i = 0;                 //empty the location array
       }
      else if (tmp != '>') {    //if the content of data is reading in
        location[i] = rc;       //store them
        i++;
        if (i >= 40) {          //if the data is larger than the given buffer
          i = 39;               //drop the data
        }
      }
      else {
        location[i] = '\0';     // reach the end of the data transmitted
        start = false;          // reset start and i
        i = 0;
      }
}
```

The commands used to control the rover is a, w, s, d, p, M and N, where a,w,s,d each indicates one direction, and M and N show the current mode of the rover, control mode and self-wandering mode respectively. Since all of them are one-character, the basic Serial.read() and Serial.write() functions can provide a successful communication.

4.4.2.3.Two-way communication with the command subsystem

The communication between the MQTT server and the esp32 should be implemented using library <PubSubClient.h>, which is a client library for MQTT messaging.

The commands given by the web are sent to the MQTT server, and then published into topic 'control', where the esp32 client subscribes. Given the ip address of the server and the corresponding port, the esp32 can connect to the server using function client.setServer (mqttServer, mqttPort). Due to the feature of MQTT protocol, the client can communicate with the server by subscribing a specific topic and using callback function to receive the arrived message of the topic. client.subscribe(<TOPIC>) and client.setcallback(callback) functions are used to achieve it.

```
client.setServer(mqttServer,mqttPort);
client.setCallback(callback);
while(!client.connected()){
  Serial.println("Connecting to mqtt server...");
  if(client.connect("ESP32Client",mqttUser,mqttPsw)){      // reconnect
    Serial.println("Connected !");
  }else{
    Serial.println("Fail ");
    Serial.println(client.state());
    delay(2000);
  }
}
client.subscribe("control");    //subscribe to the “control” topic
```

To send out the real-time location of the rover as well as the location of the obstacle to the server, client.publish(<TOPIC>,<MESSAGE>) is applied to publish the data to the server.

```
client.publish("location", location);
```

4.4.3. Problems met and solved

4.4.3.1. Value Transmission Problem

The first problem met during the implementation of the control subsystem is the transmission of the size of the bounding box from vision subsystem. Originally, the communication is based on the basic function Serial.read(). However, it is realized later that the bounding box can be larger than 256 if the obstacle is quite close to the rover, and due to the limitation of Serial.read() function which only reads in one byte a time, it is unable to transmit number larger than 256 successfully.

To solve this problem, the number is split into two bytes and then be transmitted and received individually.

The transmit code is

```
if(word > 256){
 uint8_t LSB = word;
 uint8_t MSB = word >> 8;  //shift MSB to the right
 IOWR_ALT_UP_RS232_DATA(RS232_0_BASE,LSB);
 IOWR_ALT_UP_RS232_DATA(RS232_0_BASE,MSB);
}else{
 IOWR_ALT_UP_RS232_DATA(RS232_0_BASE,word);
 IOWR_ALT_UP_RS232_DATA(RS232_0_BASE,0);
```

```
}
```

The receive end code is

```
uint8_t LSB1 = mySerial1.read();
uint8_t MSB1 = mySerial1.read();
int x = (MSB1 << 8) | LSB1;  //shift MSB to the left and add the LSB in
```

Then one can notice that data larger than 256 can be successfully transmitted and received.

4.4.3.2. Synchronization Problem

The second problem met is the difference of delay between each subsystem. If the delay of two system differs a lot, the data sent out from one system will get blocked in the receiving buffer of the other system, which leads to asynchronization and cannot achieve instant remote control.

Therefore, as the aim is to achieve synchronized transmission between each subsystem and ensure the accuracy and timeliness of data. it is important to ensure that the delay of each subsystem is almost the same. Thus, one should count the running time of each loop first, and then use delay() function to adjust the synchronization of data transmission.

The method to calculate out the run time of each loop is using millis() function, which counts the number of milliseconds passed since the program started. After several trials, an extra delay of 150ms is added to the drive subsystem, 120ms is added to the control code, and 120ms is added to the vision code.

4.4.3.3. Testing

The most important aspect of communication is to ensure low latency and high accuracy of data transmission. To test the performance, twenty trials are done for one set of test and there are 10 sets of test in total. The average accuracy and latency over 200 trials are shown in Table 4.4.1.

Table 4.4.1, The average accuracy and latency testing result.

Performance	To Command	To Drive	From Vision	From Command	From Drive
Accuracy	100%	100%	99.5%	100%	98.5%
Average Latency (ms)	115.20	209.45	73.40	56.00	379.68

From the above data, one can notice that the accuracy in general is larger than 98%. The wrong message being transmitted may be due to occasional data loss, blocked data in the buffer, and a sudden change in voltage level while the rover is launching.

In addition, the latency becomes larger in the bidirectional transmission of data between drive and control subsystem, possible reason is that great amount of data is transmitted and received at the same time, including control commands, speed, obstacle distance and location information, which in the long run slows down the speed of transmission. The latency to and from the command subsystem is acceptable as the MQTT protocol can provide a reliable transmission channel even under bad network conditions.

4.5. Command Subsystem

The command subsystem is the user interface side of this project. The user can use the website to control the rover car and observe the status of the car through a graphical plot and a status board. It is divided into frontend and backend where frontend is the webpage part, and the backend is the server and database.

4.5.1. Design Approach

Protocol

There are several requirements for the choice of protocol. It should be two-way communication. The connection is also desired to be connection-oriented so that generally there is no information loss unless the connection breaks. That is because each message from the web dashboard is sent once. Even though there is an overhead cost, the connection is usually maintained for a relatively long time. For the rover car, as the wires sometimes are loose and the fact that the team members are in different countries, it is very likely that the connection will suddenly break due to either physical or Wi-Fi issues. Hence, it is desired that the reconnection is easy to establish, and the message can be immediately received after reconnection.

Frontend

Frontend is the web dashboard design. It is designed to contain 4 sections: connection board, energy status board, control panel and a map plot. And there should be some pop-up alerts or notifications to inform the user.

Connection board is for the connection to the server. The user should be able to edit the host, port and ClientID. Energy status board is for demonstrating the message energy subsystem send which includes SOC, SOH, charging status and remain worktime. And when SOH reachs a certain dangerous level, a notification will pop up on the board to alert the user. The control panel is for controlling the rover car. The speed can be set. And there are 2 modes that the user can choose: self-wandering mode and control mode. In control mode, the user can use AWSD to control the movement of the car while in self-wandering mode the rover car itself decides where to go. Extra notifications are set to inform user when a ball is detected with its colour and the distance to the rover. The map plot is for plotting the path of the rover car and highlight its current location as well as the ball position. The path can be hidden or displayed depends on users' choice.

Backend

The server and database are all designed in backend part. Database is designed for backup information so users can revisit it. It is desired to connect to the user's own database so that each user will not be able to see other users' data and searching information in the database could be easier. For server, it is desired to handle receiving data and sending data fast.

4.5.2. Implementation

Protocol

The protocol chosen is MQTT which stands for Message Queuing Telemetry Transport. As it is based on TCP/IP, the connection is oriented, hence one of the requirements is fulfilled. It is a lightweight, publish-subscribe network protocol. The publisher can publish a message on a specific topic, and all the subscribers of that topic will receive

that message. The subscription process is fast and extremely easy. This subscribe and publish feature allows the clients to come and go at any time yet still able to receive the message once it subscribes the topic. The client can be both the publisher and the subscriber to either the same or different topics so two-way communication can be established.

Frontend

The frontend is built using React, Antd (Ant Design) and Echarts. JavaScript is mainly used. The overview of webpage is in Appendix D. The webpage is considered as a client and connected to server through http based MQTT. React is used to build the render tree and structural of the whole web dashboard. All the on Click event of buttons and input are handled using React. For better user interface, Antd is used. It modifies the layout, style, format, font etc. automatically and provide more types of alerts, inputs, and notifications. Figure 4.5.1 shows the example of Antd slider input used on website.

Figure 4.5.1, speed control bar for the control mode.

Echarts is for the map plot. Scatter plot is used for path and current location plotting. Special effect is added on the point with current location to make it remarkable. When move the mouse to current location point, the location info will be shown. The location of rover and the path is calculated by the website. The change in turning angle and moving distance is passed to webpage. The total rotated angle is recorded by the state variable 'angle' and is modified each time based on the turning angle. By using the total rotated angle, the move in x and y direction is calculated with sin and cos function and added onto the original position to give the new location. The main calculations are shown in the following code:

```
case 'location':
  var result = msg.split(",")
  var x = parseInt(result[0]) //10*angle change
  var y = parseInt(result[1])/3.2 //distance change
  if (x != 0 || y != 0){
    var newAngle = this.state.angle - x/10 //get original angle change
    var changeX = Math.sin(newAngle*2*Math.PI/360).toFixed(2) //change in x direction
    var changeY = Math.cos(newAngle*2*Math.PI/360).toFixed(2) //change in y direction
    var location = this.state.locationInfo[0] //get current location
    var newPath = [location[0]+changeX*y,location[1]+changeY*y]
    var newLoc = [newPath]
    var Path = this.state.path
    Path.push(newPath){
        this.setState({ //replace original info with new calculated ones
          angle: newAngle,
          locationInfo: newLoc,
          path: Path
        })
  }
```

The ball location is also calculated in frontend. The ball's colour and its distance to the rover is send to website. 2 assumptions are made: the ball is directly in front of the rover, and each ball detected has distinct colour. The main calculation process is below:

```
var location = this.state.locationInfo[0]
var angleNow = this.state.angle
var relativeX = distance*Math.sin(angleNow*2*Math.PI/360).toFixed(2)
var relativeY = distance*Math.cos(angleNow*2*Math.PI/360).toFixed(2)
var ballLoc = [location[0]+relativeX,location[1]+relativeY,colour]
var currentBallInfo = this.state.ballInfo
currentBallInfo. Push(ballLoc)
this.setState({
  colourSize: this.state.colourSize+1,
    colourHistory: history,
    ballInfo: currentBallInfo
})
```

Backend

The backend side used mainly Nodejs. The whole connection is shown in Figure 4.5.2. The server, as MQTT is implied, is acting as the broker of the MQTT subsystem. For broker building, 'Mosca' is used. The MQTT port is set to be 9090 and the http based MQTT port is 8888. Port 9090 is used for connection between ESP32, energy subsystem computer with server, while port 8888 connects the webpage with the server.

For database, Cloud MongoDB Atlas is used instead of the local database. That is because the amount of information stored is huge and therefore not suitable for storing them in local database. The user can also modify the URL in config folder to change to their own cloud MongoDB database. For storing data in database, models are built for each topic, specifying the structure of data stored in. These models can be changed based on the requirement of user to personalize their database.

An extra publisher is built for communication between command and energy subsystem. The energy subsystem will store the energy info message in a file in real-time, and the publisher will read the data string in the file and clear up the file again to wait for next data string. The time interval between each read and write is set to be consistent with the speed of Arduino writing to the file, so that full message can be grabbed.

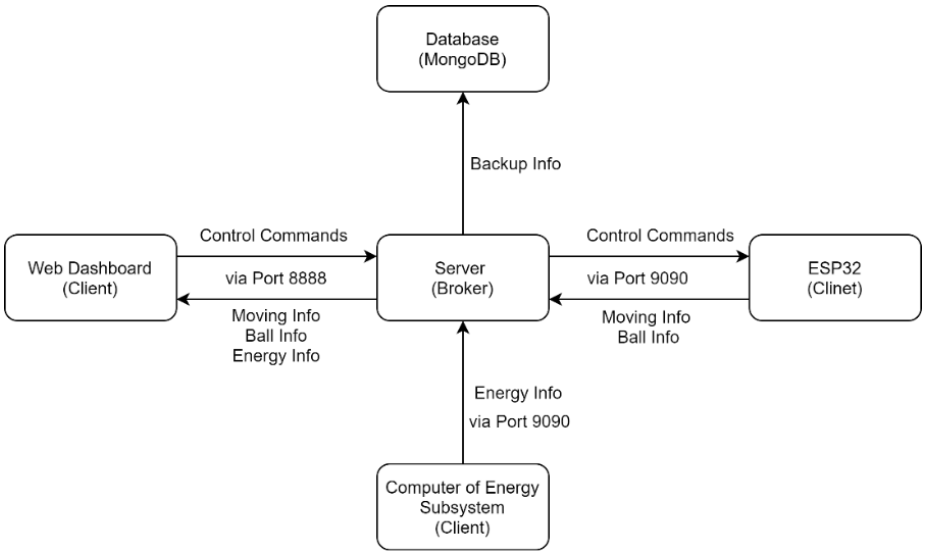


Figure 4.5.2, block diagram of connection of command subsystem

4.5.3. Testing

Four stages of testing are taken for the command subsystem before putting into the whole system.

The first stage is the testing of basic functions on the webpage, which is on buttons clicks and inputs. After that, the extra simple publisher and client are built. Some simple messages are tried to publish to a topic and the client is set to subscribe that topic to check if the message can be received. This is to test the basic connection functionality.

If all connections are fine and data can be transferred. Some possible output and incoming data are generated. This time the website is taken as both the client and publisher, and test along with the simple publisher and client. The testing checks the display of the result, map plotting and the ability to publish data from the webpage. The last stage is the test on the connection with the control and energy subsystem. This not only includes the accuracy of the messages sent but also accounts for the delay time. After this stage, the command subsystem is ready to put into the whole system for testing. And if there are any feedbacks or modification requirements, the website is modified and again go through the 4-stage testing before putting it into the whole system again.

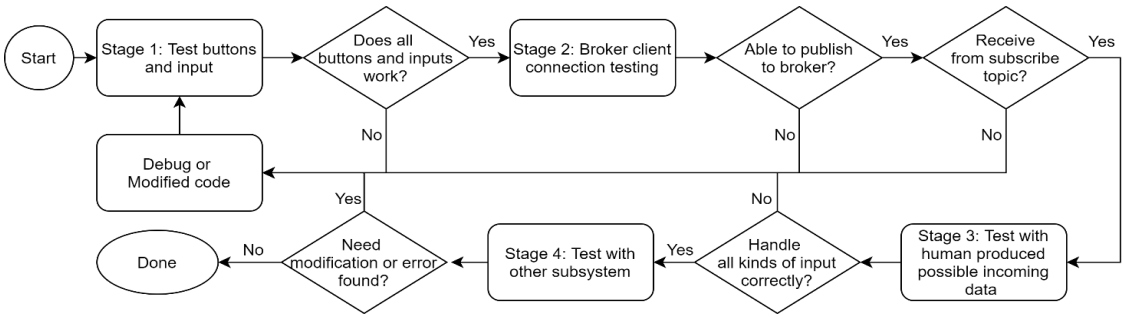


Figure 4.5.3, the flow of testing in Command subsystem.

Extra performance tests are also taken on the map plot. As re-render is always required when plotting a new point on the map and Echarts takes some time to reprocess the data and plot it. Hence, if the time interval between each data for plotting send is too small, the plot will not be able to react real-time and the current location point would look completely off track as shown in Figure 4.5.4. It would be back on track once data stops sending. The table shows the minimum time interval this distortion will not occur for local and AWS server with separate set input data when the scale of the map is automatically adjusted.

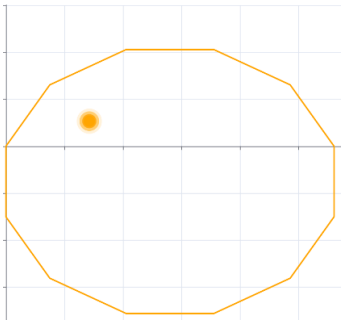


Figure 4.5.4, location track demo.

	Value of change in distance per interval	Value of change in angle per interval	Minimum time interval (ms)
Local Server	3	1	130
Local Server	3	30	310
Local Server	3000	1	480
Aws Server	3	1	200
Aws Server	3	30	360
Aws Server	3000	1	540

Table 4.5.1, the minimum time interval before the distortion occurs in Local and aws server

When the auto-scaled functionality is set, if the distance gets too large and affects the quality of real-time plot, it can be scaled down by divide the value with a constant factor. Yet, it cannot do anything about the angle in auto-scaled. Yet, the scale setting can be modified on the plot. Hence, the scale can be set fixed and large enough scale to avoid the off track looking. This can be done as the current location point has a fixed size on the map no matter of scale. Therefore, it is recommended that the data sending interval is above 600ms when auto scaled is used or set the x-axis and y-axis grid fixed and with large enough scale to avoid this error like looking.

5. Discussion -Integration Subsystem

The above section gives a detailed description of the design and implementations for each subsystem, and the unit test is covered before integrating with other modules. Since the functional correctness of each subsystem is guaranteed, this section is only concerned with the integration subsystem, where all the modules are assembled and investigated. The main purpose of the integration step is to record how the rover performs as subsystems are connected and assist in debugging between these systems. Thus, this section focuses on the overall result of the rover and the discussion of the accuracy and performance tests of the whole system.

5.1. Result

The first step towards the integration step is to ensure functional correctness when all subsystems are connected. Thus, before the results verification, several steps are taken to ensure the whole system behaves as designed. This is investigated by comparing the actual path of the Mars Rover and the one drawn by the Web app, and the position of the balls is also included in the map. This investigation also gives indicative evidence on the communications between drive, vision, control, and command subsystem. The communication across the whole system is tested by connecting every two subsystems first, and the next subsystem will be added to the system only when the test between the previous two gives satisfactory results. The whole process is repeated until the communication network is built and ready for data transmission. Therefore, how these two modes behave in the real world will be discussed in this section.

5.1.1. Command Control Mode

The command mode is aiming at controlling the rover using direction commands and set speed. Once the camera detects an obstacle, the user will receive a notification in the webpage which announces a ball is detected, and the user can choose to rotate the rover to avoid the obstacle. The Figure 5.1.1 below shows how the user moves the car once an obstacle is detected. Firstly, the rover is set at speed 4 and controlled to move forward, and then user receives the notification “a new red ball is detected (distance) cm”, as shown in Figure 5.1.2. Then the user decides to turn right to avoid the obstacle, and keeps going forward, until another obstacle (blue ball) is detected.

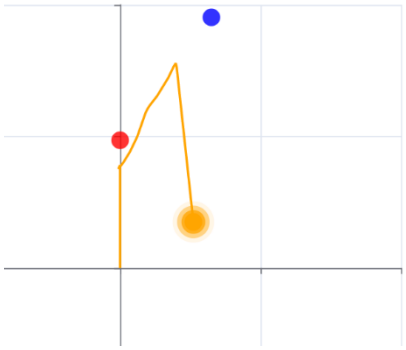


Figure 5.1.1, control mode path

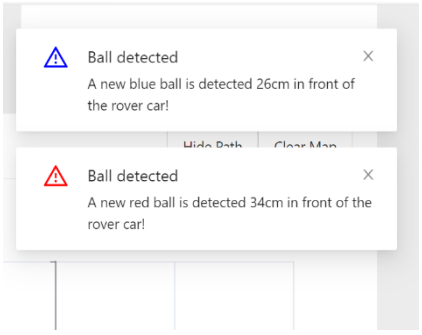


Figure 5.1.2, ball detection notification

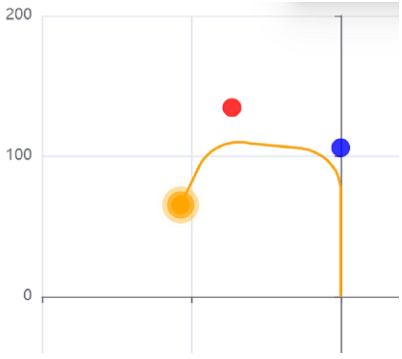


Figure 5.1.3, self-wandering mode path

5.1.2. Self-Wandering Mode

The self-wandering mode is to let the rover do autonomous exploration and decide its own path to move around while detecting the ball. During testing, one blue ball and one red ball are placed in the moving path of the ball. The rover drives from the origin and moves forward, once detects a blue ball, it turns left then keeps moving. Later, the red ball is detected, and the rover turns left again. The webpage draws the path of the rover and the location of the ball, as shown in Figure 5.1.3.

As discussed in this section, all the testing at this point is to ensure the functional requirements of the whole system are met. This similar process is repeated for several times to ensure there is no coincidences in the testing result, this result verification is followed by the accuracy test and performance test in next section.

5.2. Accuracy Test

The accuracy test aims to investigate whether the performance of the rover is within an acceptable range under real-life conditions, taking background factors into account. In addition, the impact of different factors on the rover has also been explored. One of the variables that can be directly influenced by the external factors is the distance where the rover is expected to turn left once the distance between the rover and the obstacle is within 30cm in the self-wandering mode. The next step of the rover would be moving forward until an obstacle is out of the screen of the camera. The obstacle avoidance performance with different colors of the ball and different traveling speeds of the rover are investigated in this case.

5.2.1. The capability to avoid obstacle at different travelling speed.

To keep the tests reliable and accurate, the setup steps and background such as the light source, are kept the same for every measurement. For all experiments, the rover and the obstacle are place in the same position, the obstacle should be at the center of the D8M camera to have a better object detection. As rover moving forward and travelling into a certain distance around the obstacle, the rover will turn left to avoid obstacle. The perpendicular distance from the rover to the ball at the point where the rover starts to turn is measured, this measurement is repeated ten times for each level of speed, and the blue ball is used for all experiments, these results are recorded in the table in Table 5.2.1, and the average and standard deviation are included to indicate the variations for the reading.

Table 5.2.1:

Speed / Distance (cm)	1	2	3	4	5	6	7	8	9	10	Average	SD
Level 1	28.0	29.5	28.0	27.0	27.5	29.0	28.0	30.0	30.0	28.5	28.6	0.99
Level 2	27.0	27.5	26.0	28.0	28.0	25.5	27.0	27.0	29.0	26.0	27.1	1.02
Level 3	26.5	26.0	24.5	27.0	28.0	25.0	26.5	25.0	29.0	27.0	26.5	1.33
Level 4	26.0	28.0	22.0	23.0	26.5	25.5	27.0	26.0	23.0	25.5	25.3	1.85

When the rover is operating at the level 1 speed, the average rotation speed is closest to the ideal value, and the variation is the smallest. The experiments give the expected result, since moving at high speed would largely influence the resolution of the camera, and the change in the bounding box size cannot be captured accurately. It takes time for the camera to update the increases in the bounding box size as the rover moves. Thus, the rover would move a little further until it receives the distance from the control system. By comparing the standard deviation for each speed level, the dispersion of distance at a slower speed is less than others which indicates that the performance is more reliable. However, the data for all speed levels are within the acceptable range, although the ball detection is highly dependent on the background and lighting conditions.

5.2.2. The capability to avoid different obstacles with different colour

To exam the effects of the ball colour on the rover's behavior, the previous measurement is repeated for all five colors, and the speed is kept at constant Level 4 speed. During the setup, one challenge could be finding a reliable environment that makes all coloured balls detectable. Thus, the dark-brown cupboard is used to surround the testing

area, and the direct illumination on the ball surface is avoided to reduce the chance of reflections. The starting points of the rover and the ball are kept in the fixed position, like the previous investigation. The perpendicular distance between the ball and the point that rover starts to move is measured, the results are summarized in Table 5.2.2 below.

Table 5.2.2:

Colour/ Distance(cm)	1	2	3	4	5	6	7	8	9	10	Average	SD
red	26.0	28.5	32.0	22.0	23.0	24.5	25.0	26.5	29.0	28.5	26.3	2.89
pink	30.0	29.5	31.0	28.5	28.0	29.0	29.0	32.5	30.5	27.5	29.5	1.50
yellow	33.0	31.5	29.0	30.5	28.5	28.0	32.0	28.5	31.5	27.0	30.0	2.00
blue	28.5	26.0	26.0	27.5	24.5	23.0	25.5	27.5	24.0	28.0	26.1	1.66
green	26.0	25.5	29.5	28.0	27.0	24.0	26.5	28.0	24.5	25.0	26.4	1.58

According to the testing results above, the distances vary significantly with the colour of the ping-pong balls. The balls with a lighter colour (pink and yellow) have blurred edges since the background colour shares a similar colour range in terms of the HSV color model, in some trials, the distance is even greater than 30cm which demonstrates that the competence to recognize light colour is weaker, and this results in larger bounding box size. However, the turning point of other balls with a dark colour is closer than expected, this problem may arise from the fact that the dark background increases the chance of capturing the noise and makes the edges less detectable. Therefore, the results are highly dependent on the testing environment and lighting conditions. More experiments could be carried out to investigate and find a suitable environment for the detection of five balls. In general, the fluctuations due the environmental factors are acceptable, based on the data shown in Table 5.2.2.

5.3. Performance Test

5.3.1. Overall Latency

An important aspect of performance test is the latency of data transmission. The rover is expected to respond to the commands as quick as possible, thus the overall delay from web to the server, to the control subsystem, to the drive system and then to the start of running is counted.

The test was taking place under two conditions, one is that the user uses web client in China to control the motor remotely, and the other is that the web client is used in London, while the server is set in East America. For each client location, five trials are done, and the latency is recorded, shown as the table below. The latency is recorded by using timestamp of the serial monitor at the Arduino ide, and the getHours(), getMinutes(), getSeconds() and getMilliseconds() functions in JavaScript to print timestamp for the web and server.

Table 5.3.1:

Client Location	No.	web to server(s)	server to control(s)	control to drive(s)	drive to run(s)	Total time (s)
China	1	0.073	1.439	0.053	0.336	1.901
	2	0.241	1.399	0.064	0.425	2.129
	3	0.136	1.518	0.056	0.465	2.175
	4	0.098	1.336	0.047	0.330	1.811
	5	0.167	1.422	0.043	0.267	1.899

Average		0.1430	1.4228	0.0526	0.3646	1.983
London	1	0.187	1.024	0.017	0.235	1.463
	2	0.163	1.251	0.073	0.422	1.909
	3	0.132	1.015	0.039	0.417	1.603
	4	0.145	1.343	0.062	0.331	1.881
	5	0.159	1.068	0.040	0.299	1.566
Average		0.1572	1.1402	0.0462	0.3408	1.6844

According to the testing result, the time taken to transfer data from server to control subsystem is dominated, while the actual latency may not be larger than one seconds. As for command subsystem, JavaScript is used and the time displays are all callback functions, the actual transmission time would be smaller than the amount calculated by the timestamp difference. The latency between drive system receives the commands and the rover starts to run is caused by the time taken for rover to charged up and the time taken to achieve the setting speed.

Additionally, the average latency is larger when the client is set in China (1.983s) rather than in London (1.6844s), which may be influenced by different network environment and the distance to the server. Overall, the latency is acceptable and would be imperceptible while actual running.

6. Intellectual Property

6.1. The importance of intellectual property

Intellectual Property is a work or invention that is the result of creativity, such as a manuscript or a design, to which one has rights. In simpler words, it refers to the ownership of intangible and non-physical goods. This includes ideas, names, designs, symbols, artwork, writings, and other creations. Types of IP including:

- Patent: prevent an invention from being created, sold, or used by another party without permission.
- Trademark: A distinctive sign which allows consumers to easily identify the particular goods or services that a company provides
- Copyright: not protecting ideas but tangible forms of creations and original work e.g., art, music, architectural drawings, or even software codes.
- Trade Secret: proprietary systems, formulas, strategies, or other information that is confidential and is not meant for unauthorized commercial use by others in business.

For technological projects/experiments, attention to patents and copyright is extremely important. All equipment and tools used, especially chips and PCB boards, are protected by their patents, this prevents any making, using, or selling of the same invention without direct permission from the inventor. In this project, the circuit design of the SMPS, layout of Arduino PCB, and internal configuration of Max10 FPGA are all protected by manufacturing patents to stop unauthorized reproduction and selling. The direct influence would be the necessity to connect the DE10 board to the computer while the rover is operating, since the IP cores are protected by the Quartus Prime.

Moreover, all references and articles online have copyrights which encourages creativity and innovation and enables producers to benefit financially. Considering our project report where lots of online research were done, no direct copying of contents or figures from articles or books should be done and proper source citation which lists the authors, publish date and website link must be included.

All in all, intellectual property is the engine of progress, and all creators should be respected.

7. Project Management

Mars Rover 2021

Name of the group: group 28 — Project Start: Wednesday, May 12, 2021

Members: Jingyi Liang, Tianhao Bu, Jingyi Wu, Sijun You, Yujun Han, Anqi Qiu — Today: Monday, June 14, 2021

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END	DAYS
Command					
Doing reseach on basic web structures, format, and other info		100%	12/05/2021	16/05/2021	5
Start to learn React for frontend, do research on protocol		100%	17/05/2021	23/05/2021	7
Use MQTT to build server(broker), make all function work on webpage		100%	24/05/2021	31/05/2021	8
Learn Antd and adding good-looking UI to webpage		100%	01/06/2021	06/06/2021	7
Tidy up codes, do posssible modifications if required		100%	07/06/2021	13/06/2021	7
Doing report and preparing for video taking		100%	01/06/2021	13/06/2021	14
Debug with the whole system and fix errors		100%	08/06/2021	14/06/2021	7
Control					
Study background knowledge and search information		100%	12/05/2021	16/05/2021	5
Build local uart/mqtt communication		100%	17/05/2021	23/05/2021	7
Test with vision, drive, command individually		100%	24/05/2021	30/05/2021	7
Integrate all the components, test controlling the rover through commands		100%	31/05/2021	06/06/2021	7
Test communication for self-wondering mode		100%	07/06/2021	13/06/2021	7
Prepare for the report and video		100%	31/05/2021	13/06/2021	14
Vision					
understand the requirements and the codes provideds		100%	13/05/2021	20/05/2021	8
write verilog codes to convert the RGB to HSV		100%	18/05/2021	25/05/2021	8
apply filtering to the image and compare the results		100%	22/05/2021	05/06/2021	15
starting writing report for the vision sub module		100%	25/05/2021	11/06/2021	18
adding FIR filter to get a stable result at nios2 side		100%	31/05/2021	11/06/2021	12
apply changes to the code to help for the self-wondering mode		100%	04/06/2021	11/06/2021	8
Drive					
speed control		100%	25/05/2021	30/05/2021	6
man control requirement		100%	28/05/2021	30/05/2021	3
mode switching		100%	04/06/2021	05/06/2021	2
recieving indicator		100%	25/05/2021	30/05/2021	6
overall self-wondering mode		100%	25/05/2021	11/06/2021	17
stop rover from driving		100%	27/05/2021	30/05/2021	4
optical sensor conversion		100%	16/05/2021	20/05/2021	5
Energy					
research on basic principles and data of PV cells and LiFePO4 batteries		100%	12/05/2021	15/05/2021	4
PV panel characterisation and MPPT implamentation		100%	16/05/2021	21/05/2021	6
battery charging method and SoC calculation		100%	22/05/2021	30/05/2021	9
battery balancing method		100%	31/05/2021	06/06/2021	7
battery SoH determination		100%	07/06/2021	08/06/2021	2
battery discharging test and final implementation		100%	08/06/2021	12/06/2021	5
report and test communication with command module		100%	09/06/2021	13/06/2021	4
Integration					
Testing communication from drive to control		100%	17/05/2021	25/05/2021	9
Testing Communication between command, control and drive		100%	25/05/2021	30/05/2021	6
Testing command control mode		100%	31/05/2021	05/06/2021	6
Debugging connection issue with camera		100%	31/05/2021	05/06/2021	6
Debugging optical sensor issue and delay issue between control and drive		100%	03/06/2021	08/06/2021	6
Testing different methods of transmitting location of the rover to command		100%	10/06/2021	13/06/2021	4
Debugging speed control of the rover		100%	11/06/2021	13/06/2021	3
Testing self-wondering mode using fixed coordinate		100%	27/05/2021	01/06/2021	6
Testing self wondering mode with vision		100%	01/06/2021	14/06/2021	14
Preparing for report and video		100%	01/06/2021	14/06/2021	14
Debugging speed control of the rover		100%	04/06/2021	08/06/2021	5

Figure 7.1, the project management chart for all subsystems

Project management is one of the key factors to improve the efficiency during a team project. As the project was separated into various subsystems before start, individual planning on progress and communication became extremely important. Instead of aiming at completing the tasks all in one go, a schedule was made at the beginning of the project, as shown in Figure 7.1. Group meetings were held weekly to discuss personal progress and how the project was understood by each subsystem.

For the first 2 weeks, the understanding of the whole system and the personal progress was more focused. That was

due to the fact that lots of research and learning were required to start up with each subsystem. The discussion on the whole system would provide guides on how the subparts could be done. Each subsystem would then work on its own functional requirements and prepare for the connection of the whole system later.
In the third week, the connection tests between subsystems started to take place. Discussions were taken place mainly on checking input and output format and the pins used for each subsystem. More small meetings between 2 subsystems were held apart from the whole group meeting which was held once per week. If an error occurs, the 2 subsystems would first debug their own code, then work together with the integration subsystem to figure out the causes and the way to fix it.
The last 2 weeks focused on building up the whole system. At the beginning of week 4, the whole system started to merge, and the performance testing started to taken place. Communication became significantly important and was the key here to boost the efficiency. For each subsystem, modifications were required to satisfy other subsystems' requirements or to provide work arounds for some existing problems. A shared GitHub repository was created so that every group member would be able to access other subsystems' work and maybe used them to test their own bits. The report was shared in OneDrive so that group members could edit it at the same time and help other members to refine their report contents.

8. Conclusion

This project fully discloses the main architecture and embodiment of the design of the unmanned intelligent Mars Rover and performed well under the two modes in both functional and power efficiency aspects.

The camera can detect all five colour balls provided even though there are requirements on the detecting environment. Communications between each subsystem can all be established, and data are able to pass from one subsystem to the other within one second. The delay varies with different network environments. Some small delays are also added in the purpose of synchronising the drive, control ,and vision subsystems. When running the rover, the instructions from command subsystems react fast in drive and the path of the rover can be clearly followed and displayed on the website. Speed control can also be accomplished on the website after some reaction time is required for speed to reach its set level. In self wandering mode, the rover can walk around on its own and detect the balls at the same time. Yet, the faster the speed, the time delay in veer action of rover after detecting the ball is longer, since the reaction speed of the rover is limited by the speed of distance change and only below a certain value of distance will the rover turn. Furthermore, to deal with special obstacles, for example balls that are lighter in colour (e.g. pink and yellow), the size of the bounding box is set to be slightly smaller than the actual size of the ball: the threshold distance between ball and rover for turning action to apply is shorter for lighter colour balls since their edges are harder to detect. Lastly, on the energy front, successive tracking in maximum power output of solar panel(~0.35W) was attained and battery group charging/discharging performance was fully tested in both the safety and efficiency aspects which could maintain the required power supply for at least four hours.

However, there are a few improvements that could significantly optimise the overall behavior of the rover but were reserved due to the time limit. All thoughts on further development are listed in the following section.

9. Future Work

9.1. Vision: Temporal Image filter

The image filter has the functionality to remove the noise and give a better resolution to recognize the Ping-pong ball. Two filters discussed in the vision subsystem part are the same type, the spatial filter, which means the filtering process is based on the pixel values in one single frame. Another type of filtering is the temporal image filter, the main idea is to use the data between consecutive frames [20]. The combination of the temporal and spatial filter may improve the ball detection when the rover is moving, as background noise is normally random and there is a low possibility that the pixel can be highlighted in two consecutive frames. However, the temporal filter would require a large amount of storage to store the pixel values from previous frames. Not only the storage requirement, but complex logical units are also needed to access the data stored. Therefore, with the time and technical limits, this is not introduced in the vision system design, it would be reasonable to attempt this filter if more time is allowed and a powerful FPGA is provided.

9.2. Energy: CC-CV charging Method

To increase the charging efficiency and extend capacity, the proposed multistage CC-CV (constant current-constant voltage) charging strategy should be applied, where the charging current is decreased to extend the charging process when terminal voltage reaches the charging cut-off voltage. However, having tried CV charging in the circumstance of this project the battery failed to maintain its 3.6V voltage even after long CV charging time, which was suspected that unmeasurable current or voltage peak occurred at the instance of mode changing which breaks the CV environment. Further improvements can be made.

9.3. Command: Better Database with API

So far, the location data, path as well as ball position data are sent to database as a backup and the data used for plotting is stored on the website. Yet, if the webpage is accidentally closed for some reasons, the data for plotting will all be gone and need to resend from a simple publisher to webpage. The data can be found by looking at the backup data in the database, however, it would still be troublesome if the amount of data is huge. Using API can enable the website to get data directly from database which is much suitable if long-time control and map plot is required. Therefore, in future work, API could be implemented on backend and link to the frontend webpage so that even when the webpage is closed, it can still be opened again with all the data required.

9.4. Drive: PID control on man-controlled mode

As in the connection process (both software and hardware limitations), it was complex and time consuming, the drive Arduino codes are designed to be simple to fit the connection requirements.

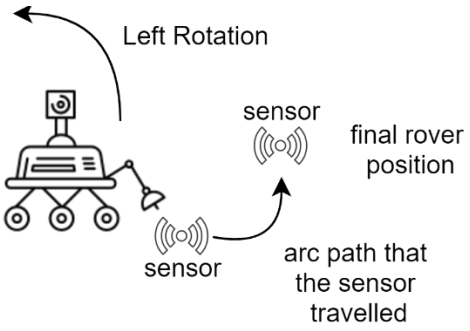


Figure 9.4.1, the autonomous mode.

Instructions can be added from command such as move a specific distance in cm or rotate a particular angle, allowing more options for the users, together with 90-degree angle rotation, they can both be implemented with PID control.

PID control on angle rotation can be achieved by setting a reference value (150), the subtraction between total_x will be the error, this error will be multiplied with a constant gain and the result will be compared with 0, the sign of the error indicates the turning direction, the error magnitude indicates the turning adjustment range, larger the error larger the turning range, the error will be approaching 0. This ensures the rotation angle is precisely 90.

PID can be applied on both man control and self-wandering, by having the PID control, it allows a precise action being taken as it has the self-tuning ability, users do not need to worry about physical interruptions such as wires pulling and unbalance weight distribution of the rover which interferes the rover movement.

The if condition was an issue, as rotating causes camera movement thus the position of the obstacle in the screen, Arduino functioning logic works as running the lines in loops repeatedly, which means FORWARD if condition would interrupt the rotation as the bounding box would change when the obstacle approaches the edge of the screen. It was solved using the state machine strategy, but PID control was not added in the end due to the timing consuming testing with control and vision subsystem, it would be nice to have PID and state machine in one in the future.

Reference

[1]. LiPeFO4 AA battery datasheet [provided by manufacturer , Brand: Ampsplus]: https://www.ampsplus.co.uk/ampsplus-14500-3-2v-500mah-battery-button

[2]. PV panel datasheet [provided by Philip Clemow, Feb 12 2021]

[3]. Perturb and Observe MPPT Alogrithm Implementation for PV Applications [Mamatha Gangavarapu; April 2015]: https://www.researchgate.net/publication/321245442_Perturb_and_Observe_MPPT_Alogrithm_Implementation_for_PV_Applications

[4]. Arduino based MPPT Controller [Saad Motahhir; March 20-2020]: https://create.arduino.cc/projecthub/motahhir/arduino-based-mppt-controller-0560db

[5]. Sun light on Mars: https://www.firsttheseedfoundation.org/resource/tomatosphere/background/sunlight-mars-enough-light-mars-grow-tomatoes/

[6]. Encyclopedia of Electrochemical Power Sources, Chapter: BATTERIES | Adaptive State-of-Charge Determination[Bergveld, HJ;Danilov, D;Notten, PHL;Pop, V;Regtien; 2009]: borrowed from Imperial College library

[7]. AN OVERVIEW OF STATE OF CHARGE(SOC) AND STATE OF HEALTH(SOH) ESTIMATION METHODS OF LI-ION BATTERIES [K. Saqli, H. Bouchareb, M. Oudghiri, N. M'Sirdi; 2019]: https://www.semanticscholar.org/paper/AN-OVERVIEW-OF-STATE-OF-CHARGE(SOC)-AND-STATE-OF-OF-Saqli-Bouchareb/514e28b47bba0daa287476e81a6349cb8c62daba

[8]. A Closer Look at State of Charge (SOC) and State of Health (SOH) Estimation Techniques for Batteries [M. Murnane, A. Ghazel; 2017]: https://www.semanticscholar.org/paper/A-Closer-Look-at-State-of-Charge-(-SOC-)-and-State-Murnane-Ghazel/964c93c82bf2ee272c6491b3cb85a06529e935d2

[9]. Lithium-Ion Battery Relaxation Effects [Marvin Messing, Tina Shoa, Saeid Habibi; 19-21 June 2019]: https://ieeexplore.ieee.org/document/8790449

[10]. Implementation of a Coulomb Counting Algorithm for SOC estimation of Li-Ion Battery for Multimedia Applications [Ines Baccouche, Asma Mlayah, Sabeur Jemmali, Najoua ESSOUKRI BEN AMARA; March 2015]:https://www.researchgate.net/publication/305641470_Implementation_of_a_Coulomb_Counting_Algorithm_for_SOC_estimation_of_Li-Ion_Battery_for_Multimedia_Applications

[11]. White Paper - Balancing cells by parallelling[Davide Andrea, publish date unknown]: http://liionbms.com/php/wp_parallel_balance.php

[12]. Scheduling of Battery Charge, Discharge, and Rest [Hahnsang Kim, Kang G. Shin; 1-4 Dec.2009]: https://ieeexplore.ieee.org/abstract/document/5369370_

[13]. Available at “Color models are color spaces”, https://programmingdesignsystems.com/color/color-models-and-color-spaces/index.html [Accessed:08-Jun-2021]

[14]. Graphical abstract in “Integer-based accurate conversion between RGB and HSV color spaces”, https://www.sciencedirect.com/science/article/pii/S0045790615002827 [Accessed: 08-Jun-2021]

[15]. University of Southampton “Computer Vision Demonstration Website” at https://www.southampton.ac.uk/~msn/book/new_demo/median/ [Accessed: 12-June-2021]

[16]. M. Aasvik, “State Machines and Arduino Implementation,” *Norwegian Creations*, 03-Mar-2017. [Online]. Available: https://www.norwegiancreations.com/2017/03/state-machines-and-arduino-implementation/. [Accessed: 11-Jun-2021].

[17]. E. Peňa and M. Legaspi, "UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices", *Analog.com*, 2020. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html. [Accessed: 29- May- 2021].

[18]. "The Standard for IoT Messaging," *MQTT*. [Online]. Available: https://mqtt.org/. [Accessed: 03-Jun-2021].

[19]. Robin2, "Serial Input Basics," *Arduino Forum*, 26-Dec-2014. [Online]. Available: https://forum.arduino.cc/t/serial-input-basics/278284. [Accessed: 15-Jun-2021].

[20]. Sang-Hee Yoo, Jae-Wook Jeon and Jung-Hyun Hwang, "Spatial-temporal noise reduction filter for image devices," 2008 International Conference on Control, Automation and Systems, 2008, pp. 982-987, doi: 10.1109/ICCAS.2008.4694641.

Appendix

A. The HSV conversion in Verilog

The implementation of the colour space conversion algorithms in Verilog is the low-level hardware design, the detailed codes for this conversion is shown below:

```
108   reg[7:0] value_reg;
109   assign value = value_reg;
110   assign saturation = saturation_reg;
111
112   reg [31:0] saturation_reg;
113
114  always@(posedge clk) begin
115     if(sop) begin
116        saturation_reg <= 0;
117        value_reg <= 0;
118     end
119     else if(in_valid) begin
120        saturation_reg <= (255*(div_1(difference,c_max))) >> 8;
121        value_reg <= c_max;
122     end
123  end
124
125  function [31:0] div_1(
126      input [31:0]  s1,s2
127  );
128  begin
129      div_1 = ((s1 << 16) / (s2 << 8));
130  end
131
132  endfunction
```

Figure A.1, HSV conversion in Verilog - first part.

```
147  always@(posedge clk) begin
148     if(sop) begin
149        hue_reg <= 0;
150     end
151     else if (in_valid) begin
152     if((c_max == 0) & (c_min == 0)) begin
153        hue_reg <= 0;
154     end
155     else if(c_max == c_min) begin
156        hue_reg <= 0;
157     end
158     else if(c_max == red) begin
159        if(green >= blue) begin
160           hue_reg <= ((div_1((green-blue)*60,difference)>>8)) + 0;
161        end
162        else begin
163           hue_reg <= 360 - (div_1((blue-green)*60,difference)>>8);
164        end
165     end
166     else if(c_max == green) begin
167        if(blue >= red) begin
168           hue_reg <= ((div_1(((blue-red))*60,difference)>>8)) + 120;
169        end
170        else begin
171           hue_reg <= 120 -((div_1(((red-blue))*60,difference)>>8));
172        end
173     end
174     else if(c_max == blue) begin
175        if(red >= green) begin
176           hue_reg <= ((div_1((red-green)*60,difference)>>8)) + 240;
177        end
178        else begin
179           hue_reg <= 240 - ((div_1((green-red)*60,difference)>>8));
180        end
181     end
182     end
183  end
```

Figure A.2, HSV conversion in Verilog -- Second part

B. Energy Arduino Code fragments

```
if (loop_trigger == 1) { // FAST LOOP (1kHZ)

 vb = analogRead(A0) * 4.096 / 1.03; //check the battery voltage (1.03 is a correction for mea
 current_measure = (ina219.getCurrent_mA()); // sample the inductor current (via the sensor ch

 Iin_pv = current_measure* pwm_out ;
 Vin_pv = vb / pwm_out;
 power_now = (Iin_pv/1000) * (Vin_pv/1000);
    if(inc_timer < 5000){//increment every 5s
        pwm_out =  pwm_out;
        inc_timer ++;
       }else{
        pwm_out = pwm_out + 0.01;
        inc_timer =0;
       }
```

Figure B.1, Code 4-1 pv V-I & V-P characterization.

```
 Iin_pv = current_measure* pwm_out ;
 Vin_pv = vb / pwm_out;
 power_now = (Iin_pv/1000) * (Vin_pv/1000);
 if(current_measure > 250){
      current_ref = 250;
      error_amps = (current_ref - current_measure) / 1000;
      pwm_out = pidi(error_amps); //Perform the PID control
 }else{
      //mppt using Perturb&Observe
      if (power_now > power_pre){
             if (Vin_pv > voltage_pre){
               //pwm_out =  pwm_out*0.9;
               pwm_out =  pwm_out-0.015;
             }else{
               //pwm_out = pwm_out*1.3;
               pwm_out =  pwm_out+0.015;}
          }else{
              if (Vin_pv > voltage_pre){
                //pwm_out = pwm_out*1.3;
                pwm_out =  pwm_out + 0.015;
              } else{
                //pwm_out = pwm_out*0.9;
                pwm_out =  pwm_out - 0.015;}
          }
       }
       power_pre = power_now;
       voltage_pre = Vin_pv;
```

Figure B.2, Code 4-2 MPPT by P&O implementation code

```
   case 3: { //Discharge state (-250mA and yellow LED)
       current_ref = -250;
       soc = soc + (current_measure/2019600)*100;//nominal capacity = 561mAh, total Q = 561*3600 = 2019600mAs
      if ((V_Bat > 2500) && (int(soc*10) != 951) && (int(soc*10) != 851)&& (int(soc*10) != 751) && (int(soc*10) != 651)&&(int(soc*10) != 551)
        next_state = 3;
        digitalWrite(8,false);
      } else if ((int(soc*10) == 951) || (int(soc*10) == 851)|| (int(soc*10) == 751) || (int(soc*10) == 651) || (int(soc*10) == 551) || (int(s
        next_state = 6;
        digitalWrite(8,false);
      } else { // If we reach full discharged soc=0 , move to final rest
        next_state = 4;
        digitalWrite(8,false);
      }
```

Figure B.3, Code 4-3 SOC-OCV implementation code

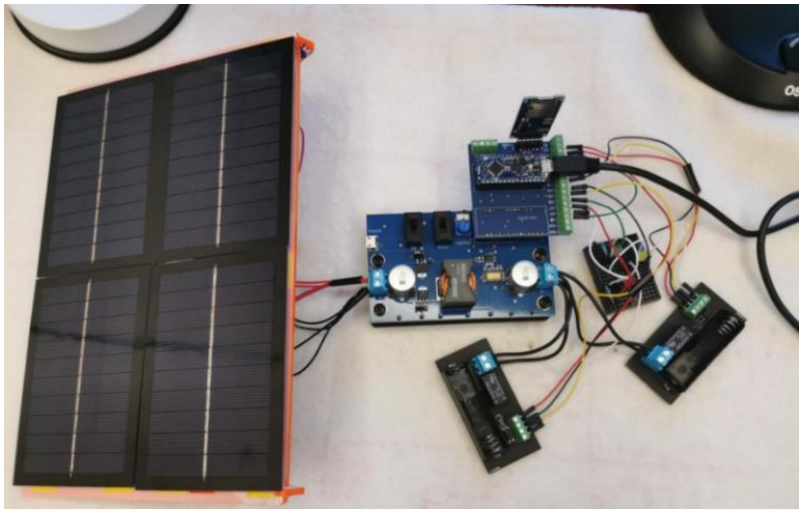


Figure B.4, battery charging circuit

C. Drive:

Initial testing environment:

Figure C.1, initial testing environment for Drive.

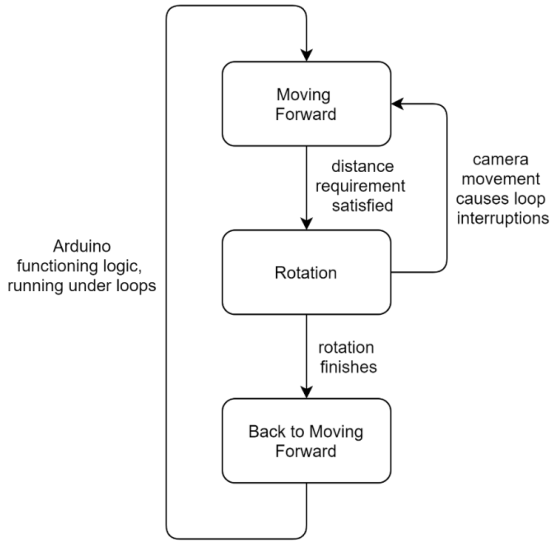


Figure C.2, Arduino interruption logic

The floor is marked with coloured stickers as shown in the picture. The red sticker represents the origin/initialization point, the yellow sticker surrounded by the cotton stick is the position of the obstacle.

D. Webpage Overview

This session is for demonstrating the webpage. Below is the whole preview.

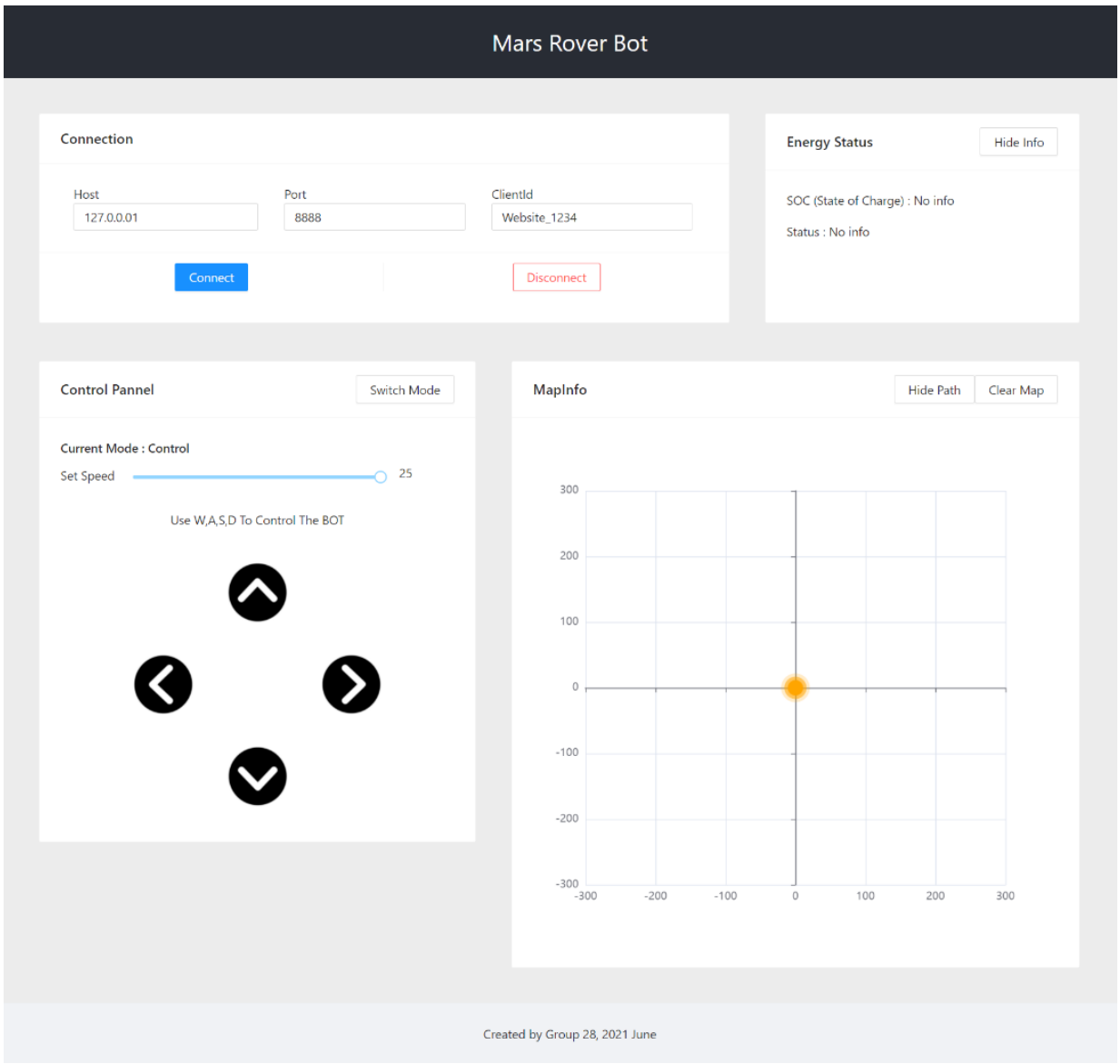


Figure D.1, Web App Overview

Possible notifications:

1. SOH notification will warn if SOH is below 80%

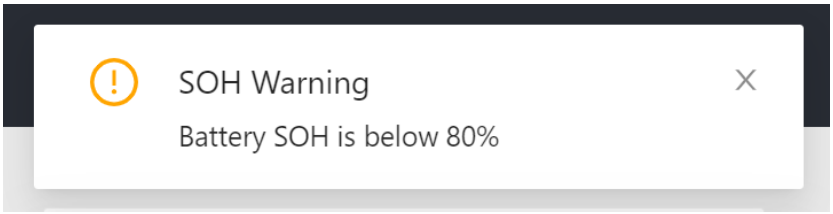


Figure D.2, SOH warning notification

2. Ball detect notification will warn if ball is detected with its colour and distance

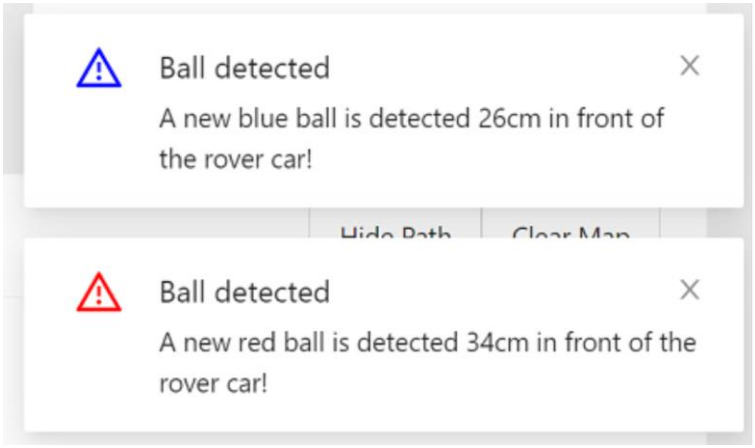


Figure D.3, Ball detection notifications

Another Overview when both buttons for changing display is clicked:

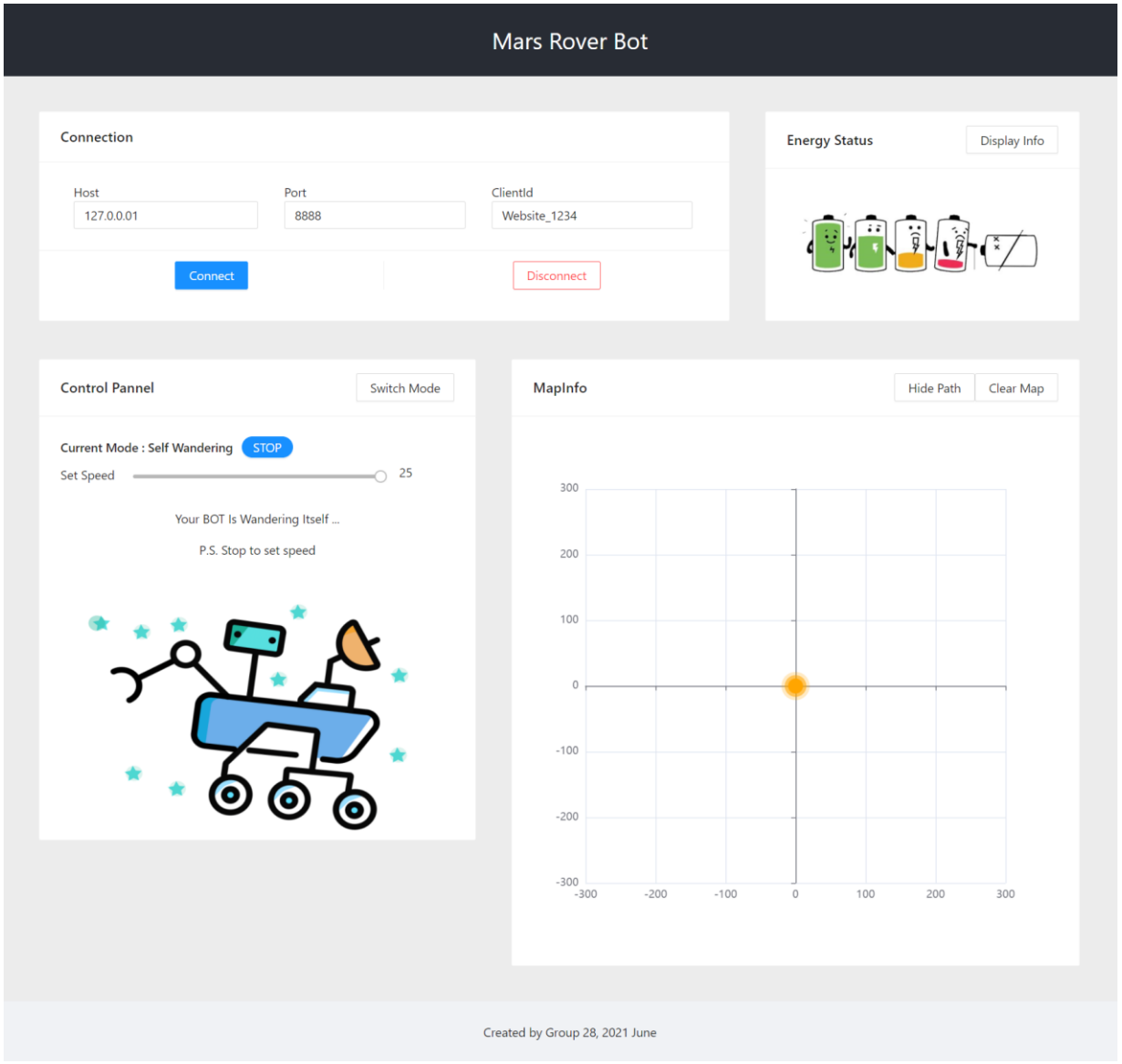


Figure D.4, Web App page overview when the energy block is activated