Intelligent Drone for Detecting Real-time Object using Machine Learning and Artificial Intelligence

Abhishek Kumar

A Thesis Stage II Submitted to

Indian Institute of Technology, Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of M.Tech. in Smart Mobility

Department of Smart Mobility (Interdisciplinary)

December 2021

Approval Sheet

This Thesis Stage II entitled Intelligent Drone for Detecting Real-time Object using Machine Learning and Artificial Intelligence by Abhishek Kumar is approved for the degree of M.Tech. in Smart Mobility from IIT Hyderabad

---

(Dr. G V V Sharma) Guide

Dept. of Electrical Engineering

IIT Hyderabad

Abstract

This thesis is proposed to serve as a concise and to-the-point report of my work in the domain of autonomous navigation of UAVs (Unmanned Aerial Vehicles) and UGVs (Unmanned Ground Vehicles). A driver or operator is generally required to operate a ground vehicle (car, bus, etc.) or an aerial vehicle (quad-copter, etc.). These operators/drivers must be trained, and obtaining a license is also a tedious process (especially for aerial vehicles). In reality, these factors prevent us from utilizing these machines to their full potential and exploring their wide array of valuable applications.

By automating ground vehicles, we could save on fuel, reduce accident-related costs, and improve overall transportation efficiency. An autonomous aerial vehicle, on the other hand, can be used for many purposes, including: Food and medical delivery in remote areas, Public safety in calamities, Bridge and power lines inspection, Agriculture, Road accident report, Traffic monitoring, and Transportation.

The main objective is to enable the UAVs to realize autonomous flight control, autonomous object tracking and obstacle avoidance with high accuracy. In order to achieve our goals, I have done few projects related to autonomous navigation.

Chapter 1 starts with basic micro-controllers which we have used in our projects. Chapter 2 defines the UAVs i.e, how flight dynamics works and which controllers and sensors are used to fly our drones or quad-copters. In chapter 3, we discussed a indoor navigation problem i.e, Beacon tracking using ESP32 and Raspberry Pi. This problem has been solved by two methods. We also compared these two methods one is moving average and other is Machine Learning based approach. In chapter 3, we implemented an application of lane detection using a simple approach i.e, Hough Transform algorithm on our UGV kit using Raspberry Pi as a main micro-controller.

Contents

List of Figures

Chapter 1

Introduction

1.1 Micro-Controllers and their specifications

1.1.1 Raspberry Pi 3B

Raspberry Pi (RPi) is an on-board computer and it is among the most popular development boards. The Raspberry Pi runs a customized version of Linux called Raspberry Pi OS. It has all the features of a personal computer. This board has USB interfaces that allow it to connect to peripherals (such as a mouse and keyboard), USB storage, etc. HDMI interface can be used to connect displays to the board having up to 4K resolution. Additionally, it has other ports/interfaces, including USB-C (power), a 2-lane MIPI CSI port (camera), a 3.5mm audio jack (audio), and RJ-45 (Ethernet). WiFi and Bluetooth 5.0 support are provided by the Raspberry Pi.

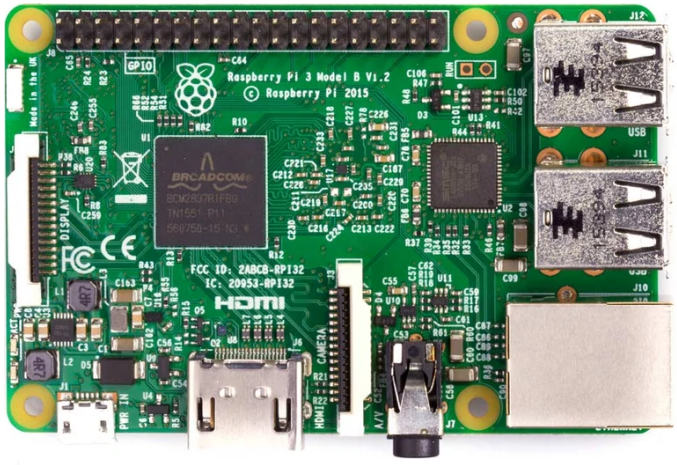


Figure 1.1: Raspberry Pi 3B

1.1.2 ESP32

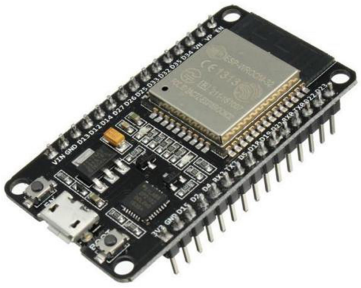


Figure 1.2: ESP-WROOM-32 Dev Kit

ESP32 is a family of low cost and low power system on a chip (SoC) micro-controllers. It is enabled with dual mode bluetooth and integrated Wi-Fi.It is a successor to the ESP8266 micro-controller. Tensilica Xtensa 32-bit LX6 is main processor inside ESP32. It has memory of 320 KiB RAM and 448 KiB ROM. It contains 34 programmable GPOIs, 12-bit SAR ADC up to 18 channels, two 8-bit DACs, four SPI, two I2S and I2C and three UART interfaces.I have used ESP32-WROOM-32 module for my projects.

Parameters	**Raspberry Pi 3B**	**ESP-32**
Processor	Quad-core Broadcom BCM2837 (4×Cortex-A53)	Xtensa Dual-Core 32-bit LX6 with 600 DMIPS
GPU	Broadcom VideoCore IV @ 250 MHz	-
Operating voltage	5V	3.3V
Clock speed	1.2GHz	26 MHz – 52 MHz
System memory	1 GB	<45kB
Flash memory	-	up to 128MB
EEPROM	-	-
Communication supported	IEEE 802.11 b/g/n Bluetooth, Ethernet Serial	IEEE 802.11 b/g/n
Development environments	Any linux compatible IDE	Arduino IDE, Lua Loader

Programming language	Python, C, C++, Java, Scratch, Ruby	Embedded C, C++
I/O Connectivity	SPI DSI UART SDIOCSI GPIO	UART, GPIO

Table 1.1: Comparison between Raspberry Pi 3B and ESP-32

Chapter 2

UAV (Unmanned Aerial Vehicle)

2.1 Quad-copter (Drone)

2.1.1 Introduction

Innovative work on unmanned aerial vehicles (UAVs) is getting high consolation these days, since the applications of UAV can apply to many areas such as military, salvage mission, film making, farming, and others. In remote areas like border areas or agricultural fields to monitor the situations on a daily basis, for surveillance in those areas, Unmanned Aerial Vehicle(UAV) plays an important role. UAV is also known as Drone and has a lot of attention nowadays. Drones are generally used at the border areas which cannot be done manually by military forces. The quad-copter works according to the force or thrust generated by four rotors connected to its body. It has four input and six yield or output states (x, y, z, θ, ψ, ω), and it is an under-activated framework, since this empowers quad-copter has to convey more load.

2.1.2 Quad-copter Motion Mechanism

Quad-copter can be described as a vehicle with four propellers joined to the rotor found at the cross casing. This goes for altered pitch rotors driven to control the vehicle movement. The velocities of these four rotors are independent. By controlling the pitch, roll and yaw angle, the position of the vehicle can be controlled effectively.

Quad-copter has four inputs, and essentially the thrust is generated by the propellers attached to the rotors. The speed of each motor is controlled independently, and the motion or direction of the quad-copter is controlled by varying the speed and direction of each motor.

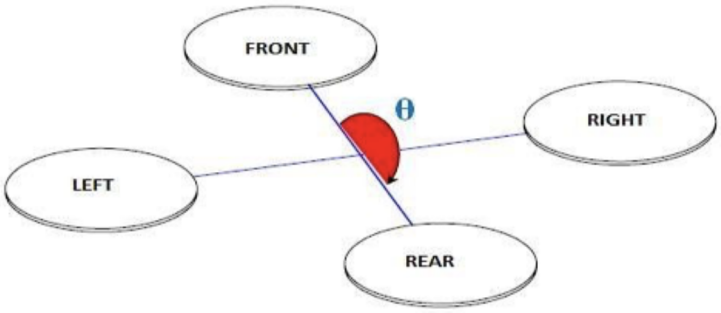


Figure 2.1: Pitch direction of quad-copter

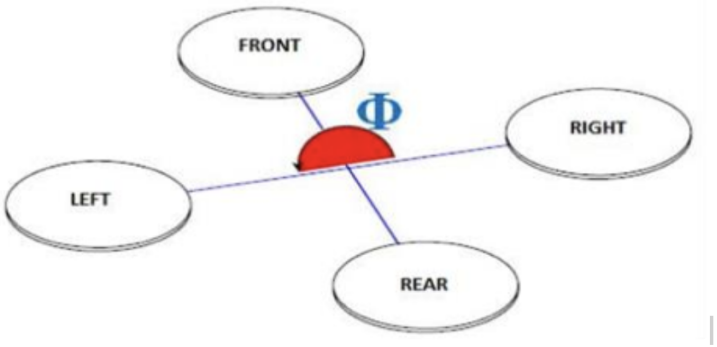


Figure 2.2: Roll direction of quad-copter

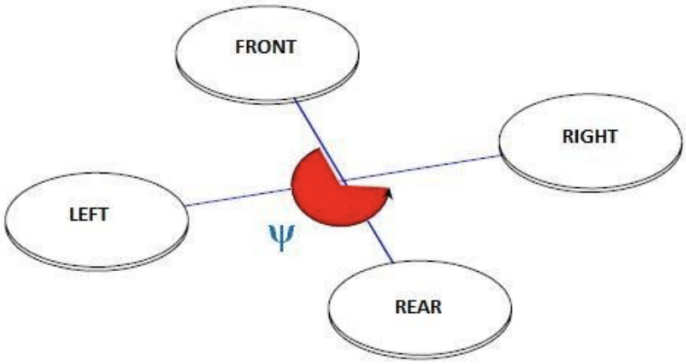


Figure 2.3: Yaw direction of quad-copter

2.1.3 Take-off and Landing Motion Mechanism

Take-off motion is the motion that lifts the quad-copter from ground to hover position. As shown in Fig.2.4 there are a total four motors, two rotating in the clockwise direction and two rotating in counter clockwise direction. To fly the quad-copter in hover position, increase the speed of each rotor simultaneously. For landing the quad-copter to ground decrease the speed of each rotor

simultaneously as shown in Fig.2.5.

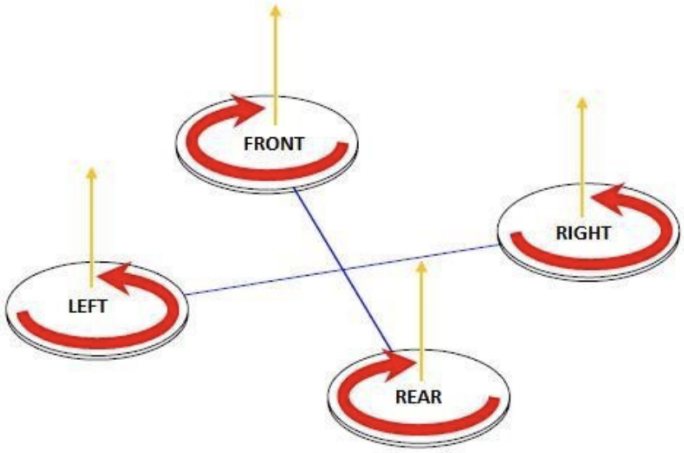


Figure 2.4: Take-off motion

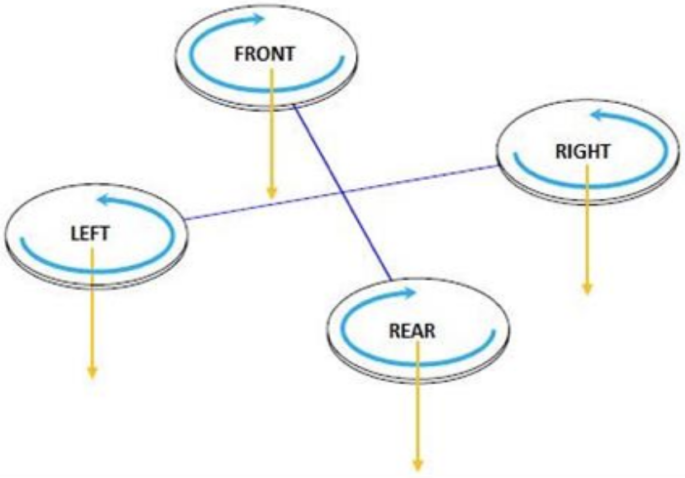


Figure 2.5: Landing Motion

2.1.4 Forward and Backward Motion Mechanism

Forward motion of the quad-copter is controlled by increasing the speed of the rear rotor and decreasing the speed of the front rotor simultaneously as shown in Fig.2.6. Backward motion of the quad-copter is controlled by increasing the speed of the front rotor and decreasing the speed of the rear rotor simultaneously as shown in Fig.2.7. Reducing the rear rotor speed and increasing the front rotor speed simultaneously will affect the pitch angle of the quad-copter.

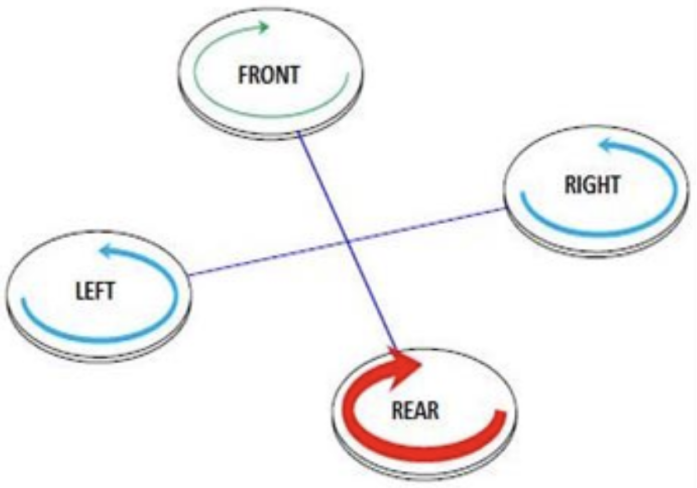


Figure 2.6: Forward motion

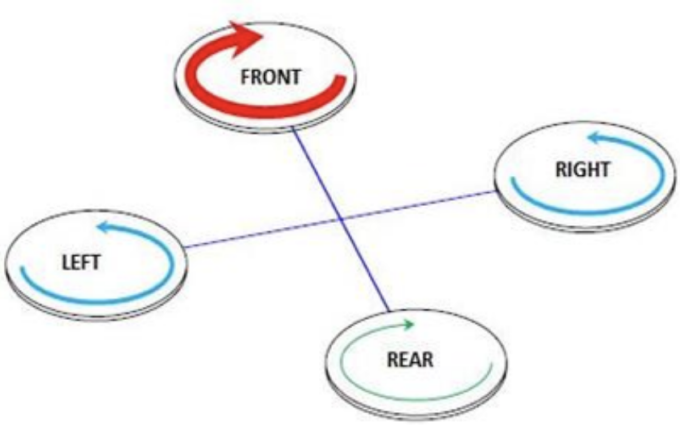


Figure 2.7: Backward motion

2.1.5 Left and Right Motion Mechanism

The left and right motions of the quad-copter are controlled by changing the yaw angle. By increasing the speed of the counter-clockwise rotor and decreasing the speed of the clockwise rotor simultaneously, the quad-copter moves to the left side as shown in Fig.2.8. Similarly by increasing the speed of the clockwise rotor and decreasing the speed of the counter-clockwise rotor simultaneously, the quad-copter moves to the right side as shown in Fig.2.8.

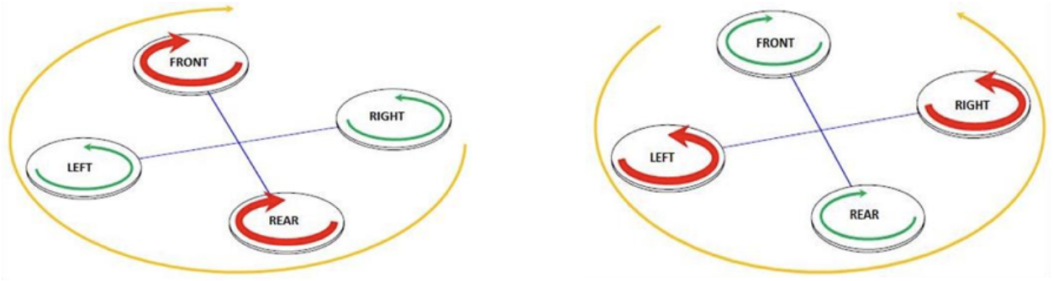


Figure 2.8: Left and right motion

2.1.6 Hovering and Static Position

When two pairs of counter-clockwise and clockwise rotors rotate at the same speed, the quad-copter moves to hover position. At that time, the total addition of reaction torque is zero which allows the quad-copter to achieve hover position.

2.1.7 Hardware requirements

Description	Make	Model
Frame(Quad-copter 4 axis)	-	F450
Motor (1000Kv)	-	A2212/13T
ESC (30 Amp)	Simonk	30A
APM Controller (2.8)	ArduPilot	APM 2.8
Transmitters and Receiver	Flysky	Fs-i6 2.4Ghz
Duracell Batteries for Transmitter	Duracell	Ultra (Alkaline AA)
GPS	Ublox	NEO 5 pin
Battery (2200mAh 3S, ,11V)	Shang yi	B3 - 2200 mAH
Battery Chargers (B3)	imaxRC	B3 pro
Propellers pairs (CW and ACW)	-	(10*4.5inch)

Table 2.1: Hardware components for UAV

Frame (Quad-copter 4 axis F450):

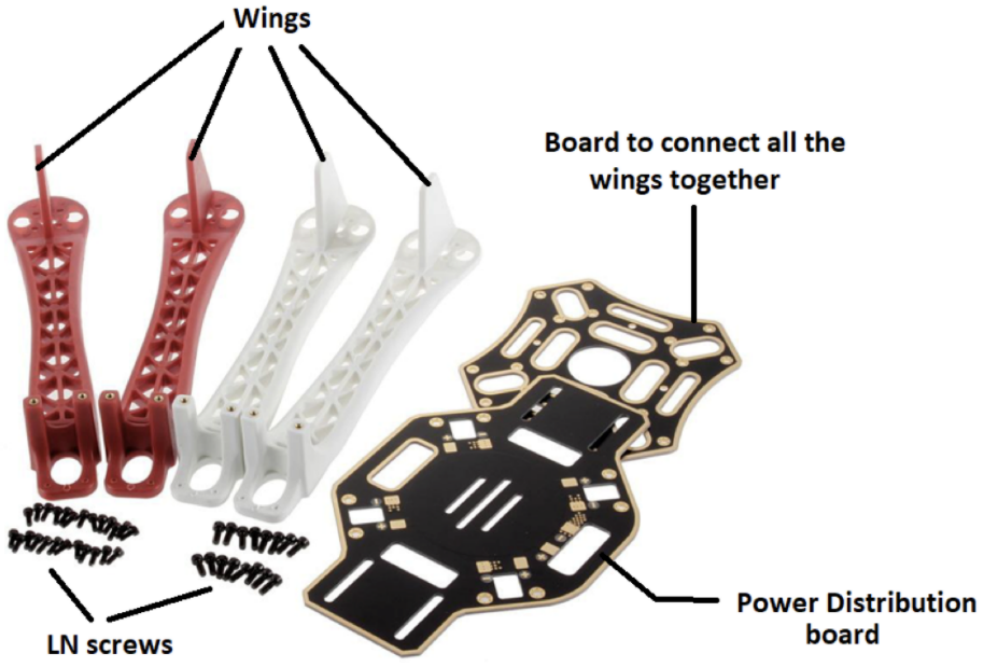


Figure 2.9: Frame (Quad-copter 4 axis F450)

Direction of Motion w.r.t. Frame:

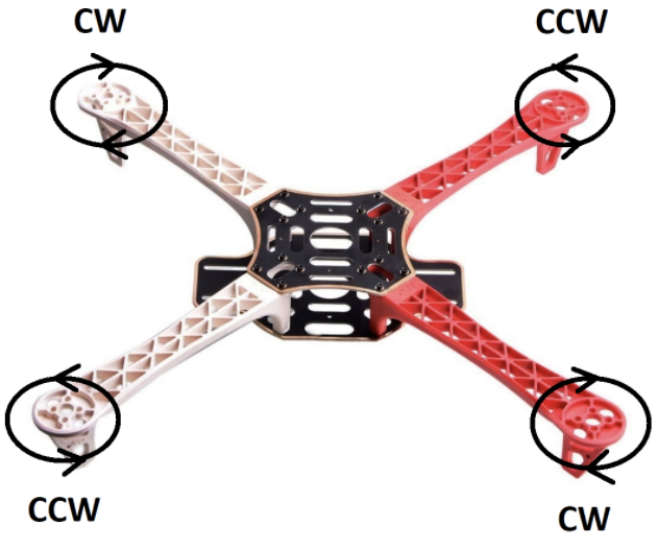


Figure 2.10: Direction of Motion w.r.t. Frame:

Motors (A2212/13T):

Figure 2.11: Motors (A2212/13T)

- KV rating : 1000 kV
- Operating voltage : 12V
- Idle Current: 0.5 A

- Motor Dimensions: 27.5 x 30mm
- Shaft diameter : 3.175mm.
- Weight: 47 g

Electronic Speed Controllers (ESCs):

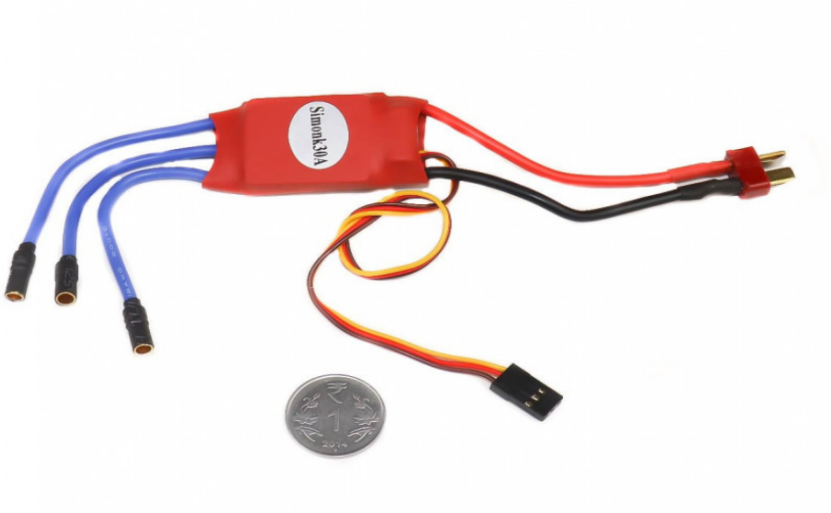


Figure 2.12: Electronic Speed Controllers (ESCs)

- Output:
  - 30A continuous
  - 40Amps for 10 seconds
- Input voltage: 2-4 cells Lithium Polymer
- BEC : 5V, 3Amp for external receiver and servos
- Max Speed:
  - 2 Pole: 210,000rpm
  - 6 Pole: 70,000rpm
  - 12 Pole: 35,000rpm
- Weight: 32gms
- Size: 55mm x 26mm x 13mm

Connection type	Wire Colour	Function
Power	Red	7.4 to 14.8V
	Black	Ground
BLDC Motor Connections	Three Blue Wires	BLDC ESC connections
Servo Connector	White	Throttle Input
	Red	5V, 2Amp Out
	Black	Ground

Figure 2.13: Connection between ESC and BLDC motor.

Transmitter (Fly-sky FS6iAB):

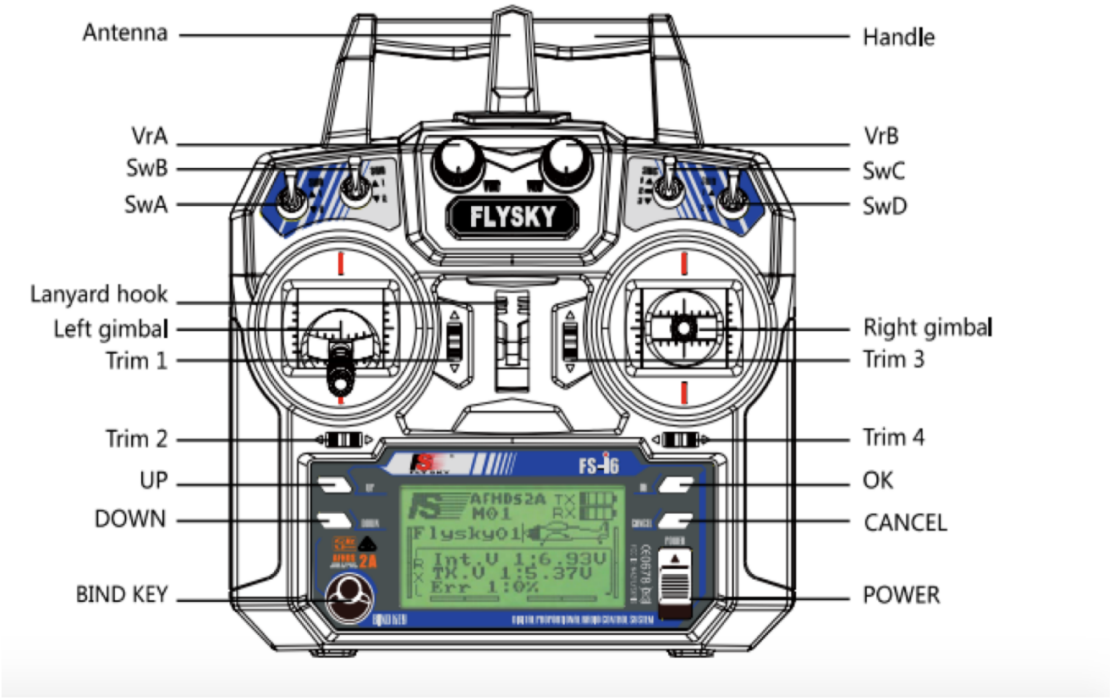


Figure 2.14: Transmitter (Fly-sky FS6iAB)

- Model type : Quad-copter
- RF range : 2.408 - 2.475GHz
- Bandwidth : 500 KHz
- Bands : 135

- RF power : Less than 20 dBm
- Protocol : AFHDS 2A
- Modulation type : GFSK
- PS2/USB Port : Yes (Micro-USB)
- Power input : 6V DC 1.4AA*4
- Weight : 392g
- Size : 174 x 89 x 190 mm
- Color : Black

Receiver (Fly-sky FS6iAB)

Figure 2.15: Receiver (Fly-sky FS6iAB)

- Channels : 6
- RF range : 2.408 - 2.475GHz
- Bandwidth : 500 KHz
- Bands : 135
- RF power : Less than 20 dBm
- Rx sensitivity : 105 dBm

- Protocol : AFHDS 2A
- Modulation type : GFSK
- Power input : 4 – 6.5 V DC
- Antenna length : 26 mm x 2
- Weight : 7g
- Size : 40 x 21 x 15 mm
- Color : Black

Controller (Ardupilot APM 2.8):

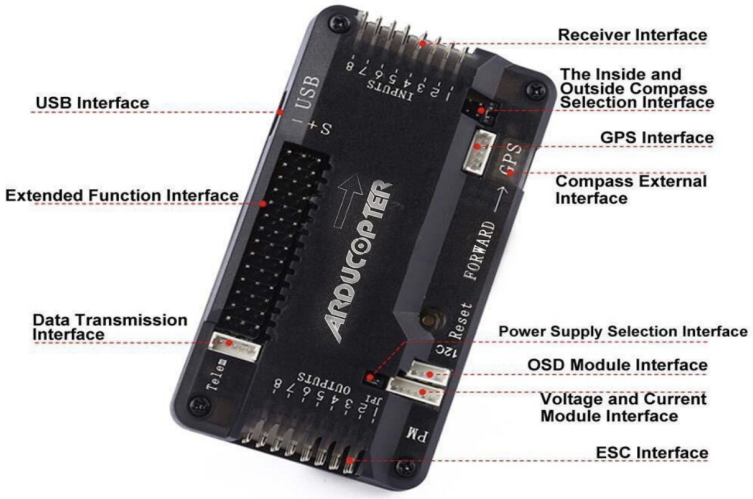


Figure 2.16: Controller (Ardupilot APM 2.8)

Model	: APM 2.8
Power supply	: LP2985-3.3.
Input Voltage (V)	: 12˜16 VDC
Sensors	: 3-Axis Gyrometer
	Accelerometer
	High-performance Barometer
Processor	: ATMEGA2560 and ATMEGA32U-2
Dimensions (mm) LxWxH	: 70 x 45 x 15
Weight (gm)	: 82
Shipment Weight	: 0.085 kg
Shipment Dimensions	: 9 × 3 × 2 cm

GPS Module:

Figure 2.17: GPS Module

- Model : FPV Ublox NEO-M8N
- Receiver Type : 72-channel Ublox M8 engine.
- Main Chip : Ublox NEO-M8N
- Sensitivity : Cold starts: –148 dBm. Hot starts: –156 dBm.
- Position Accuracy : Autonomous: 2.5 m SBAS: 2.0 m
- Acceleration : <4g

- Navigation Update Rate : up to 18 HZ.
- Operating Temperature Range : -24^{o}C 84°C
- Tracking Sensitivity : –167 dBm.
- Capture Time : 0.1s Average
- Dimensions (mm) : 50 x 12.8 (Diameter x W)
- Weight (gm) : 28
- Cable Length : 30 CM
- Supply Voltage (V) : 1.4 3.6

Battery (2200mAh 3s, 11V)

Figure 2.18: Battery (2200mAh 3s, 11V)

- Battery Type : Lithium Polymer 2200mAh 3S 25C
- Capacity : 2200 mAh
- Number of cells : 3S
- Total supply voltage : 11.1V
- Size : 24 x 34 x 108mm
- Discharge Rate : 25 C

- Burst Rate : 50C
- Weight : 183g

Overall Interconnection Diagram:

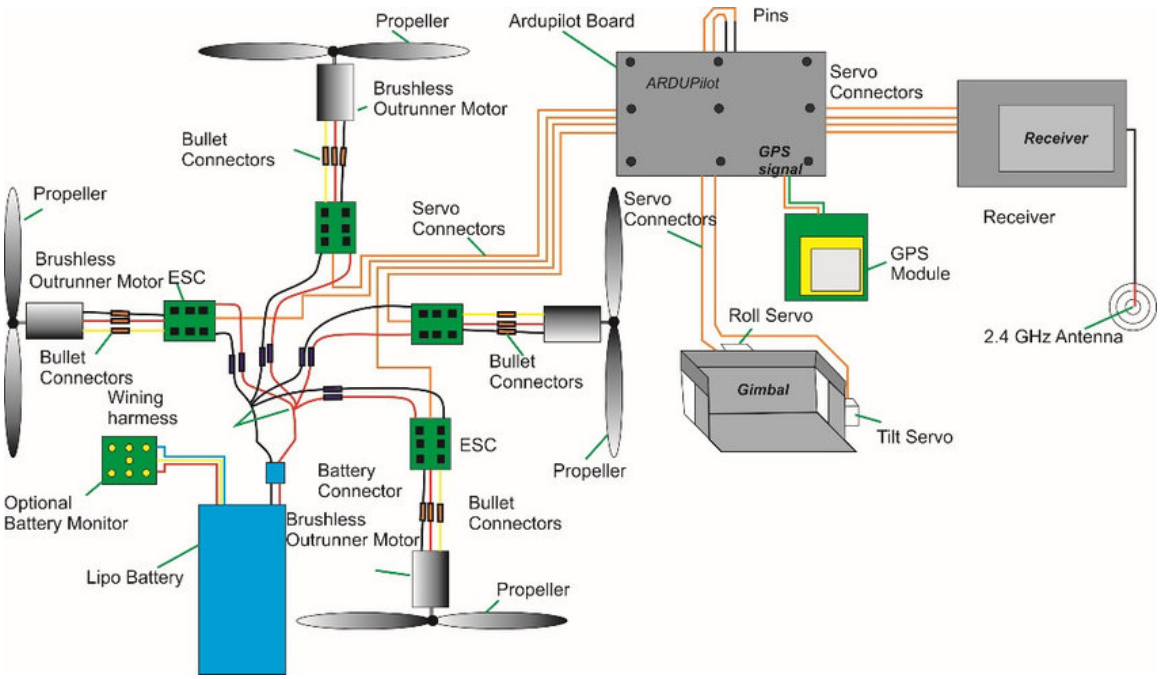


Figure 2.19: Overall Interconnection Diagram for quad-copter

Connection between ESC and motor:

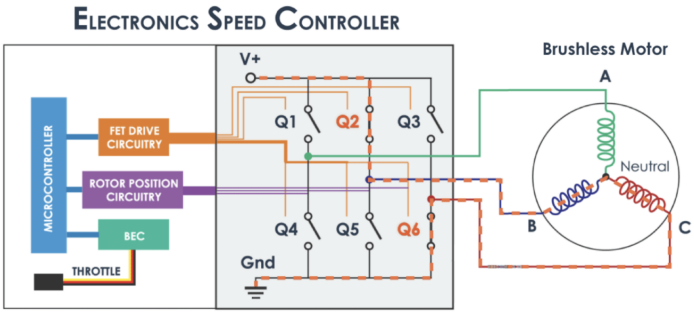


Figure 2.20: Connection between ESC and BLDC motor

PWM signal from Controller to ESC:

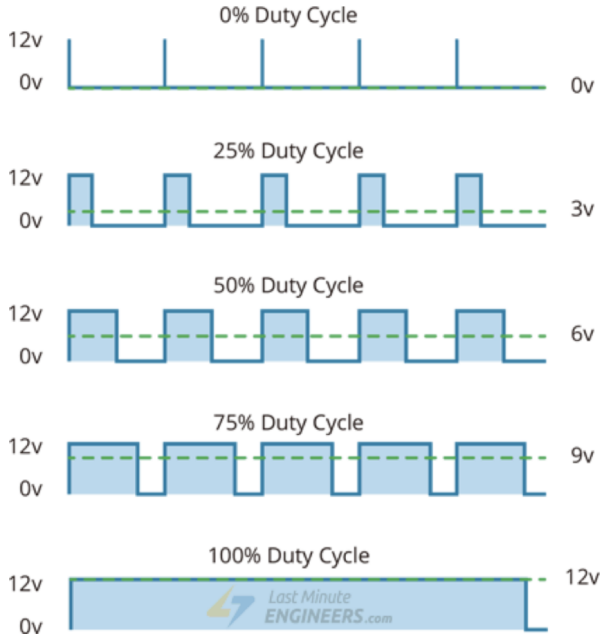


Figure 2.21: PWM signal from Controller to ESC:

- PWM Signal is sent from the controller to the ESC module in order to control the speed of the motor.
- Lesser the Duty cycle of the PWM signal, lesser is the RPM of the motor.
- Max. RPM achieved at 100% Duty cycle.

2.1.8 Configuration of Drone

All files and resources related to configuration can be found at below GitHub link

```
https://github.com/Abhishek-IITH/Drone-Building.git
```

Installing Mission planner on UBUNTU 20.04

STEP 1: Install Mono

```
sudo apt install mono−runtime libmono−system−windows−forms4.0−cil libmono−system−core4
    .0−cil libmono−winforms2.0−cil libmono−corlib2.0−cil libmono−system−management4.0−cil
     libmono−system−xml−linq4.0−cil
```

OR full Mono: (In case the upper command does not work use this one.)

```
sudo apt install mono−complete
```

STEP 2: Download Mission planner Download MissionPlanner.zip using the below steps:

- Get the latest zipped version of Mission planner here:

  ```
  https://firmware.ardupilot.org/Tools/MissionPlanner/
  MissionPlanner-latest.zip
  ```

- Unzip in the appropriate directory
- Open terminal, go into the directory where you unzipped your mission planner.
- Run Mission planner using the command:

  ```
  mono MissionPlanner.exe.
  ```

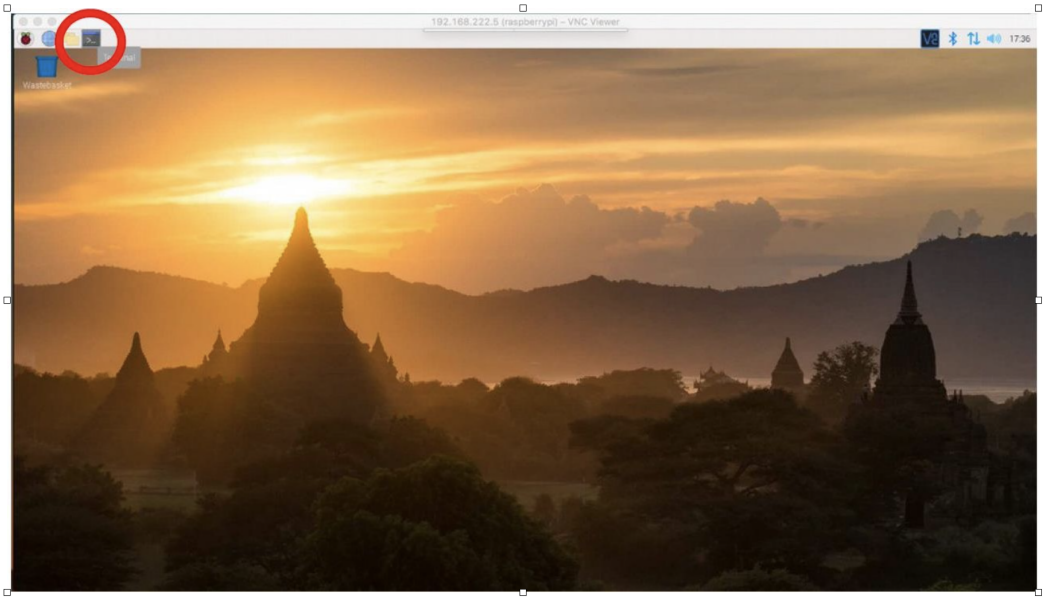
You can debug Mission Planner on Mono with the command:

```
MONO_LOG_LEVEL=debug mono MissionPlanner.exe
```

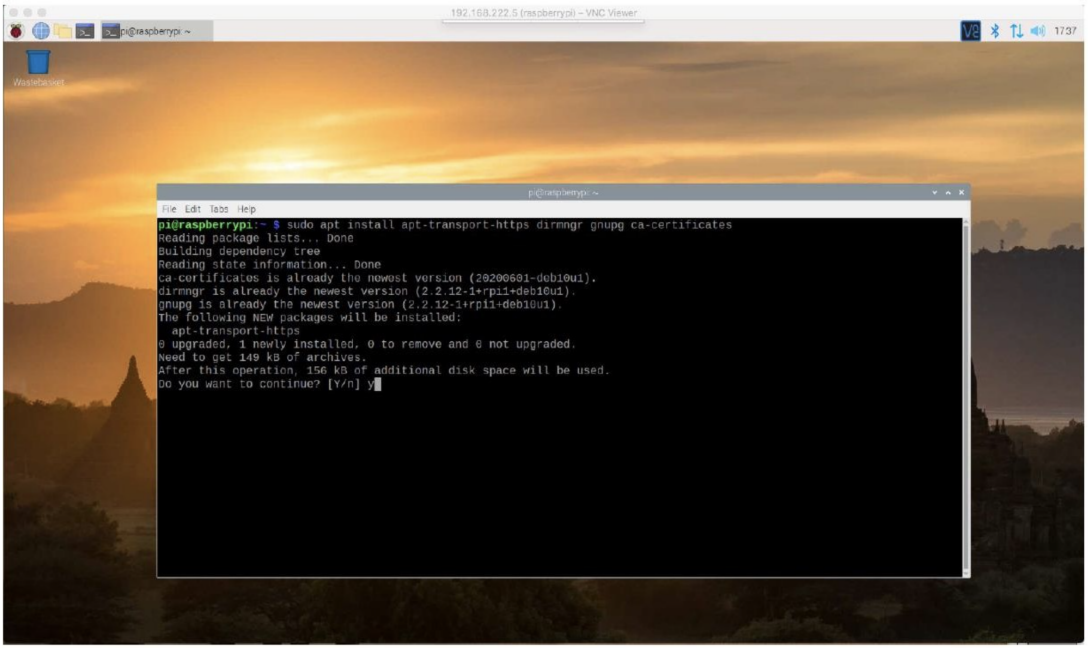
Installing Mission planner on Raspberry pi OS

STEP1: Install Mono

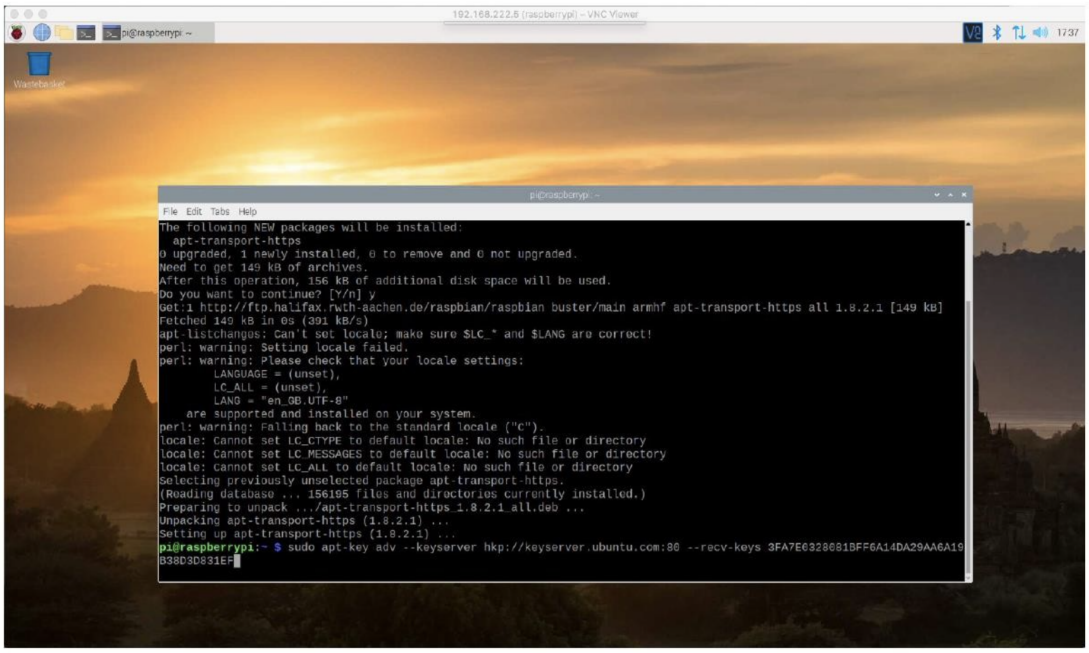
- First, open the Terminal by clicking the icon in the top left corner.



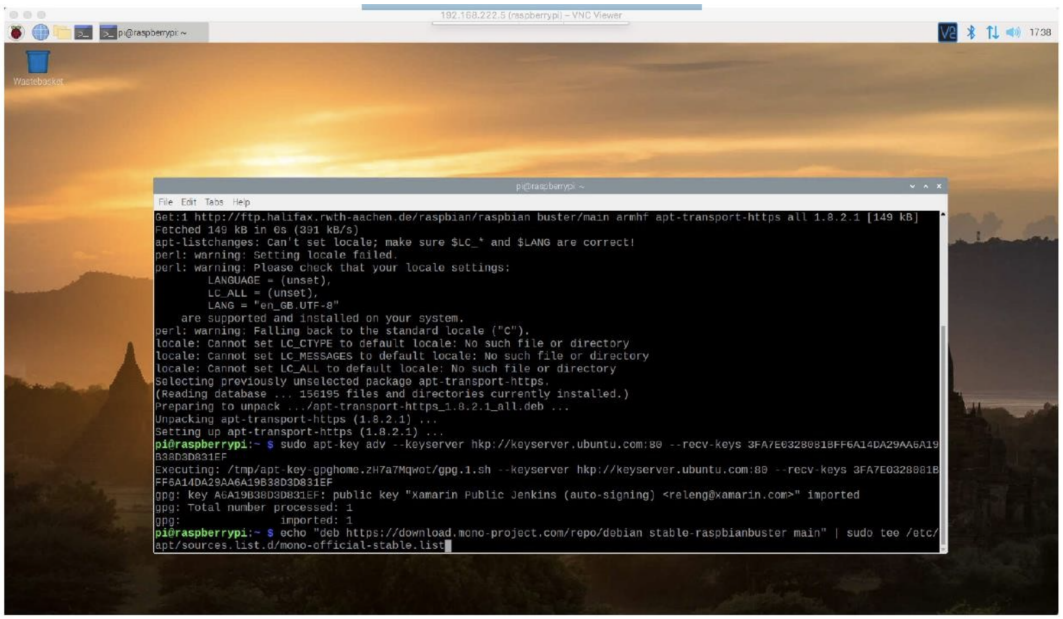
Command 1: "sudo apt install apt-transport-https dirmngr gnupg ca-certificates"



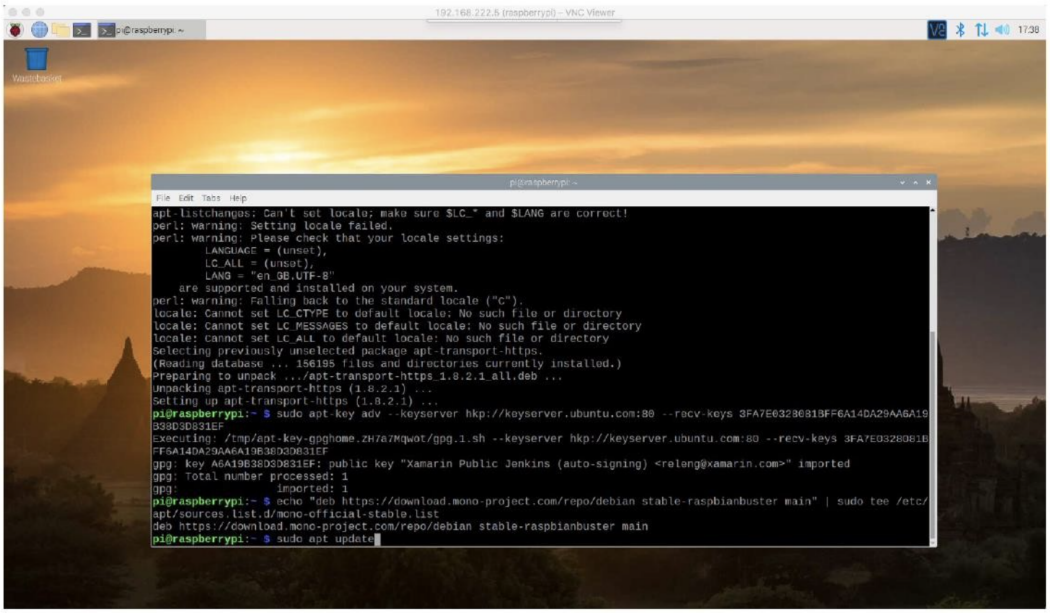
- **Command 2:**" `sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF`"



- **Command 3:** `echo "deb https://download.mono-project.com/repo/debian stable-raspbianbuster main" | sudo tee /etc/apt/sources.list.d/monoofficial-stable.list` (Enter both commands as one line!)

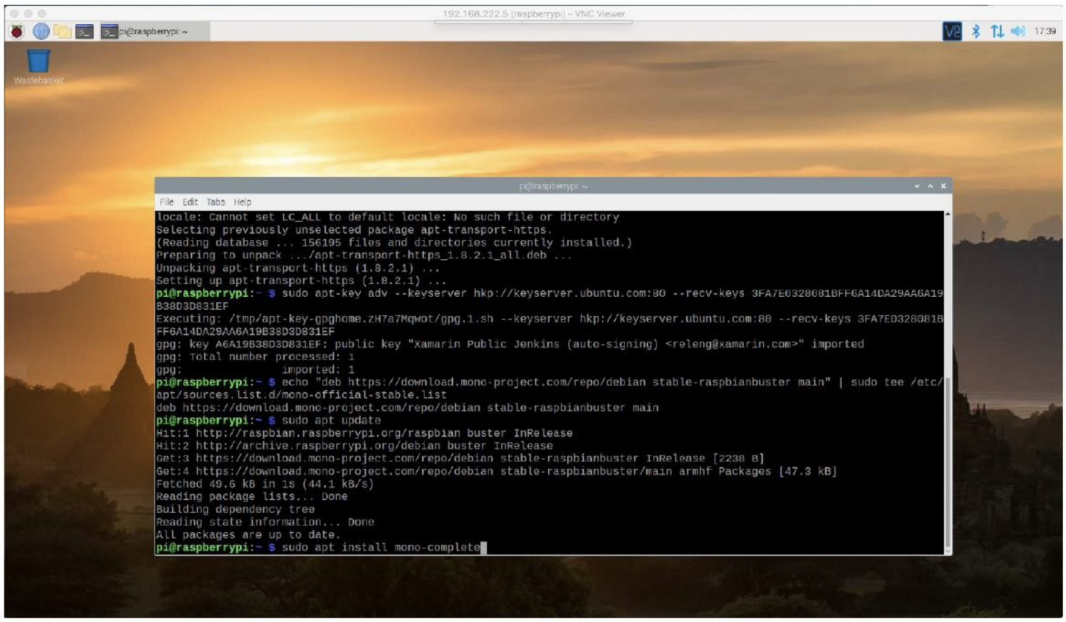


- Command 4: `sudo apt update`

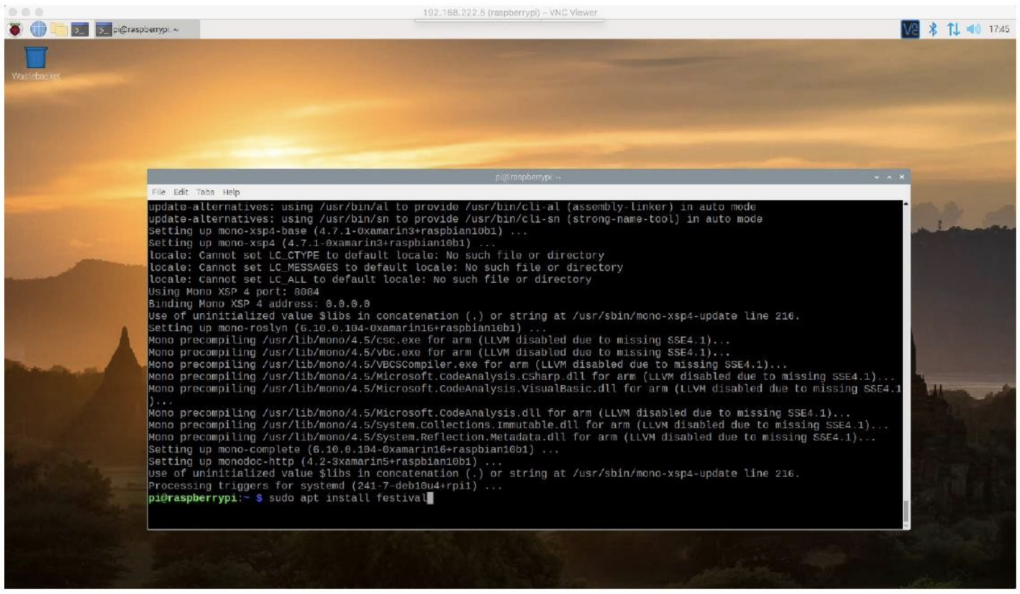


- After this, the terminal should tell you “all packages are up to date”. If this is not the case, please enter: “sudo apt upgrade”.

- Command 5: `sudo apt install mono-complete`
- Note: This step will download 300MB and take some minutes.

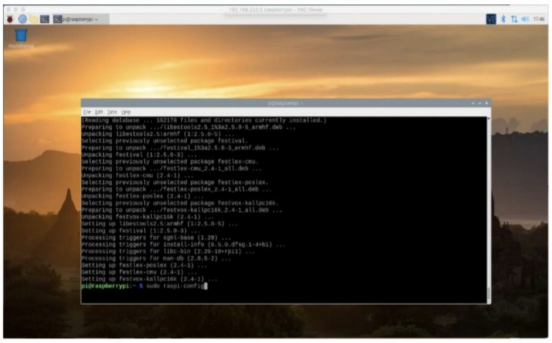


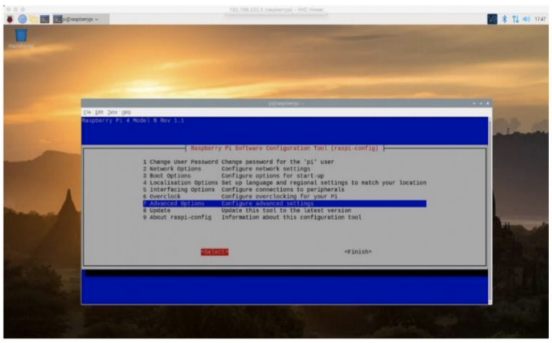
- When mono is finished, we need to install “Festival”. This is needed to give Mission Planner a voice.
- Command 6: `sudo apt install festival`



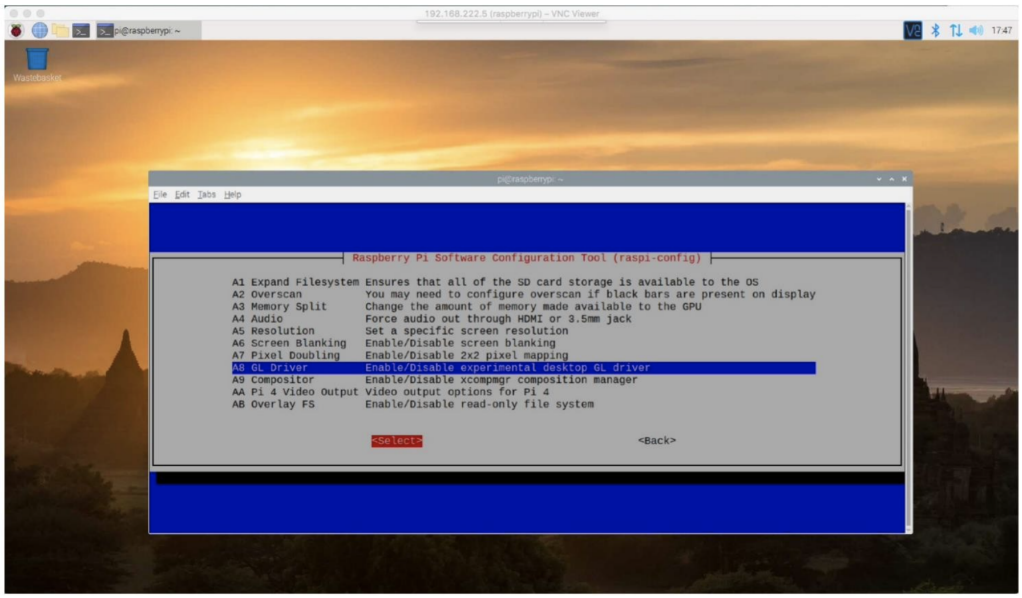
STEP2: Deactivate OpenGL

- Run command: `sudo raspi-config`
- Select “Advanced options”

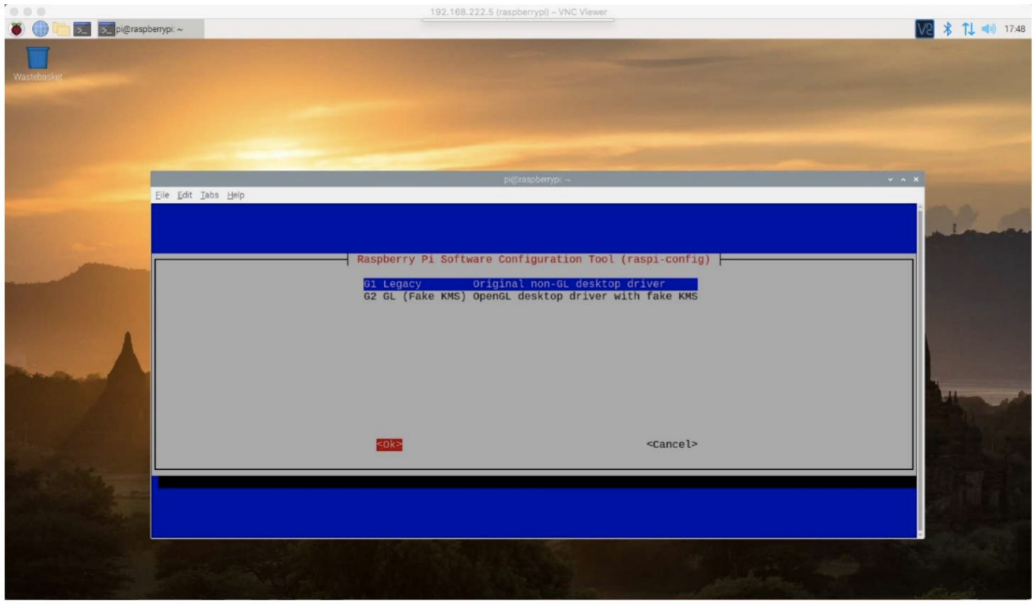




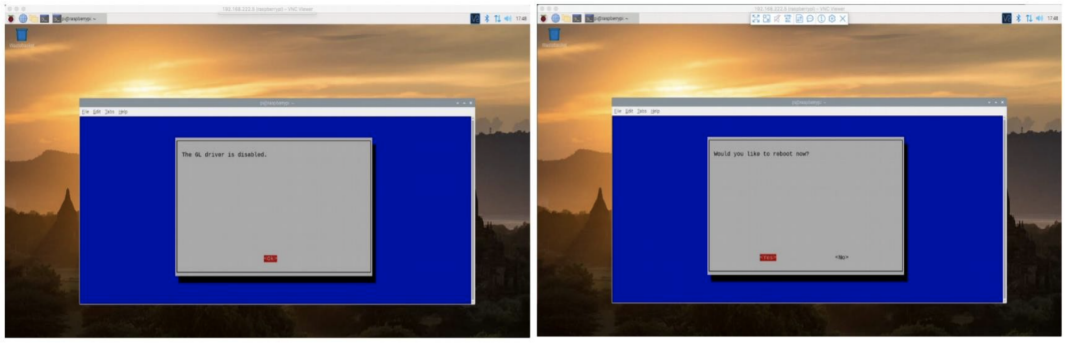
- Select “GL Driver”



- Choose “G1 Legacy”



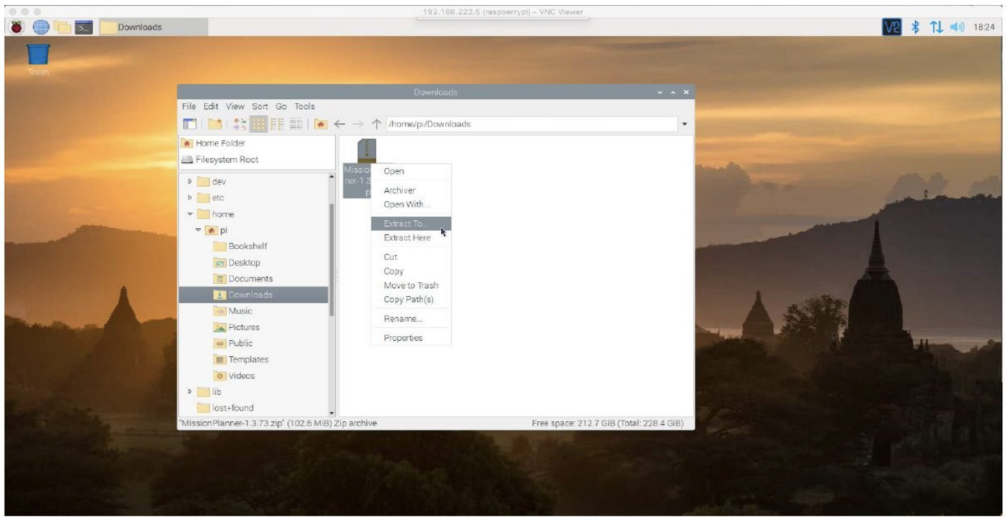
- It should say “GL driver is disabled”. Click “OK”
- Select the “Finish” option and select “Reboot” to reboot the system and apply changes.



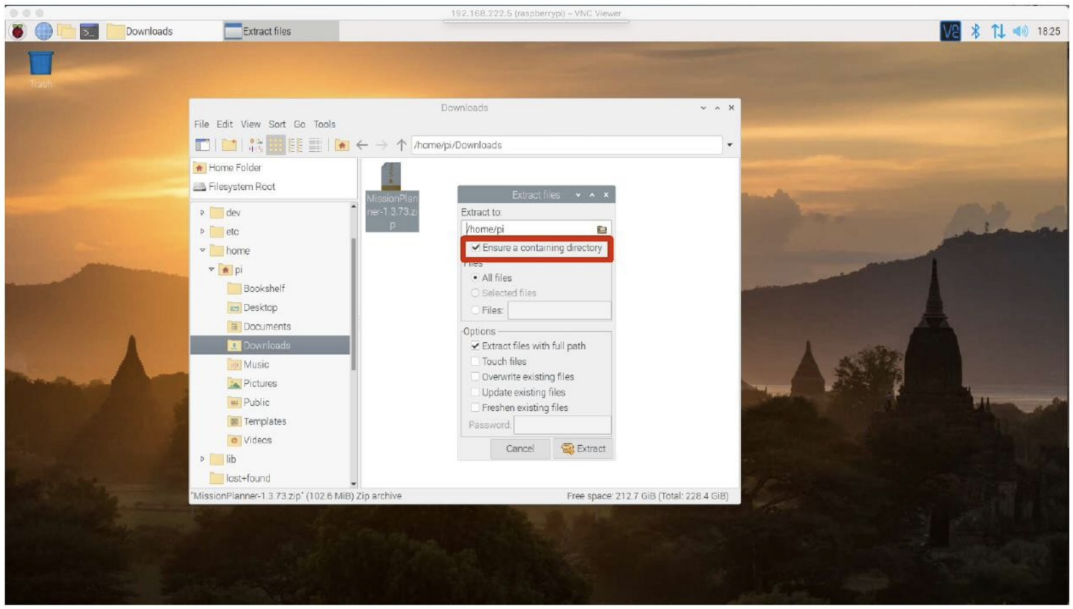
STEP3: Download Mission planner

Download MissionPlanner.zip using the below steps:

- Open Chromium web browser.
- Go to https://firmware.ardupilot.org/Tools/MissionPlanner/
- Download the latest version of Mission Planner. (1.3.74)
- Note: download the .zip file.
- Also, on the top left, you'll find your File Manager. Go to “home/pi/Downloads”.
- Right-click on the file → extract to.

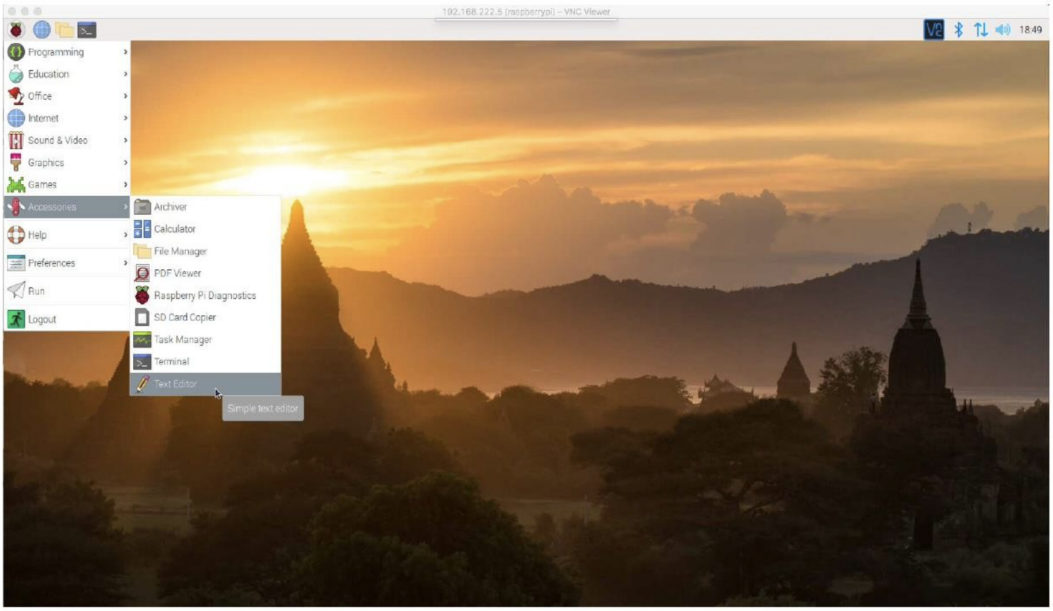


- Extract it at any desire location. However, you'll need to enter the path later to create a desktop icon for easy access.
- I extracted it straight to “home/pi/” where it will create its own directory (be sure to tick “ensure a containing directory”)



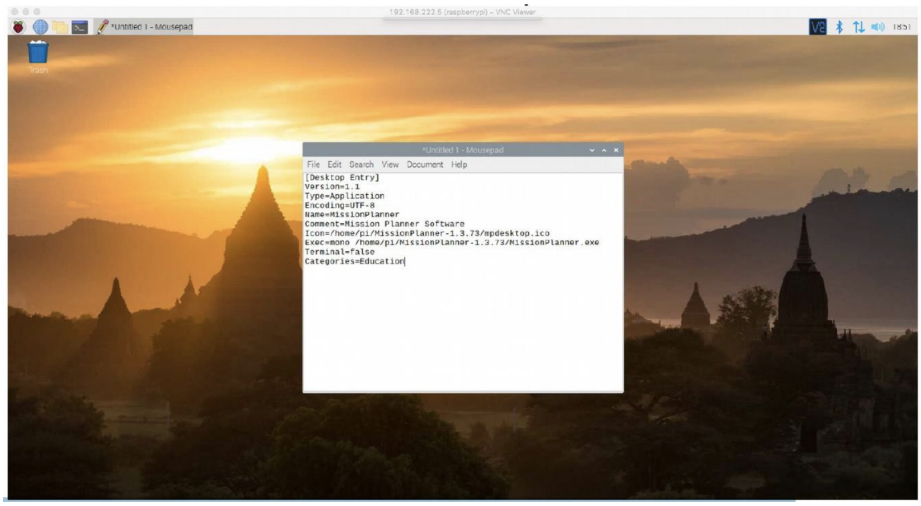
Follow these steps to create a desktop icon:

- Have a look under the Raspberry for Text Editor.

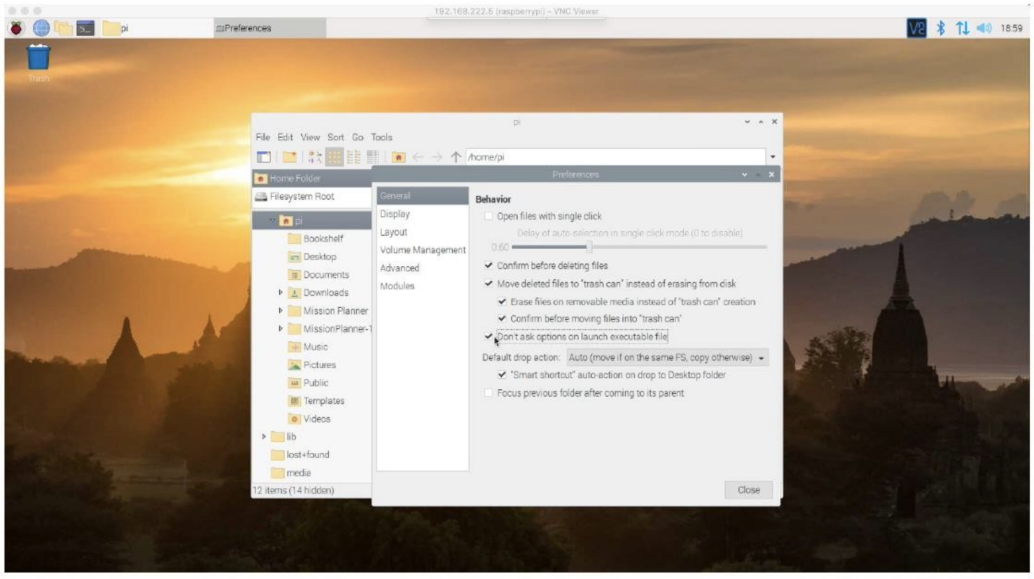


- We will copy the following lines into the Text editor and save that as “MissionPlanner.desktop” straight on our desktop. (Note: Change the path for the `Icon` and the `Exec` to the path of extraction. However, “`=mono`” is required) `[Desktop Entry]`

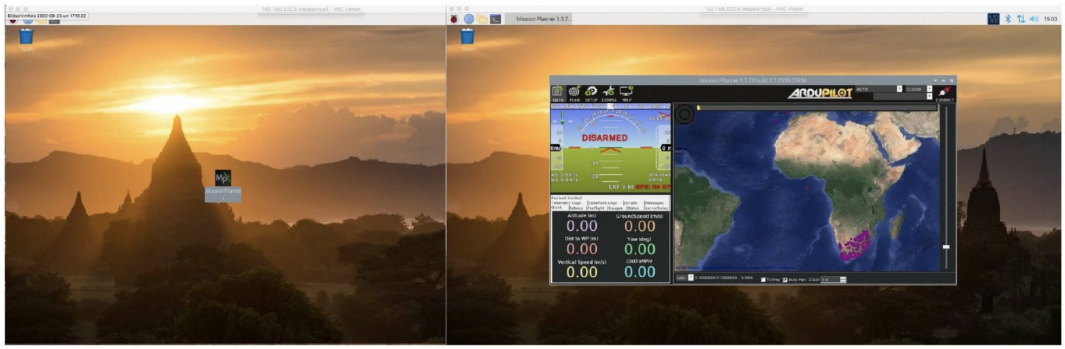
```
Version=1.1  Type=Application  Encoding=UTF-8  Name=MissionPlanner
Comment=Mission Planner Software  Icon=/home/pi/MissionPlanner-1.3.74/mpdesktop.ico
Exec=mono /home/pi/MissionPlanner-1.3.74/MissionPlanner.exe  Terminal=false
Categories=Education
```



- When you click on your new desktop icon, your Pi will prompt you with some rather annoying choices. We will now remove them.
- Inside your “File Manager” top menu → edit → preferences → general.
- Toggle the box “don't ask options on launch executable files”.



- The Download and install process is now completed.

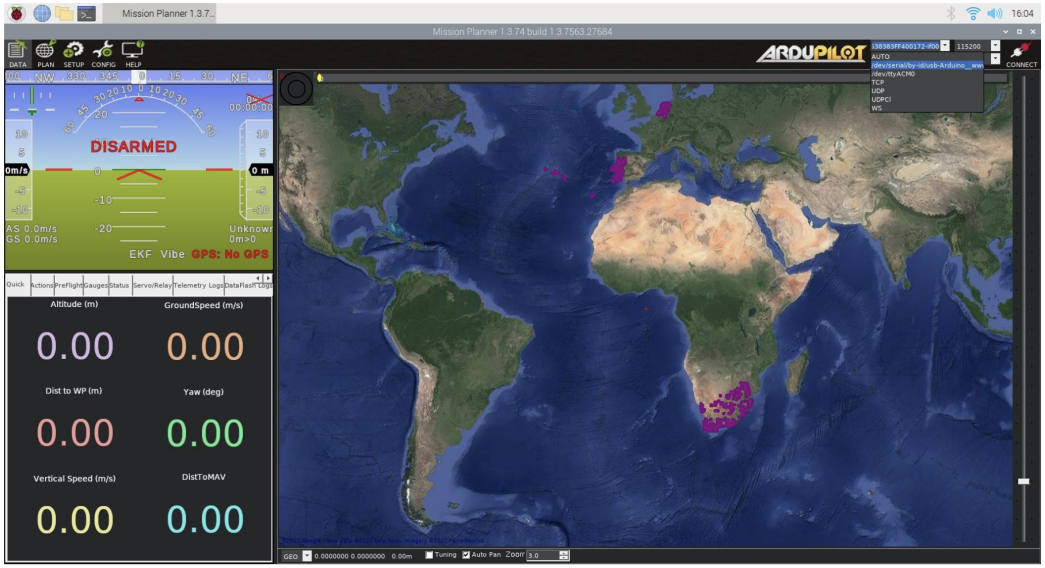


Calibration of drone using mission planner on Raspberry pi OS

STEP1: Getting started.

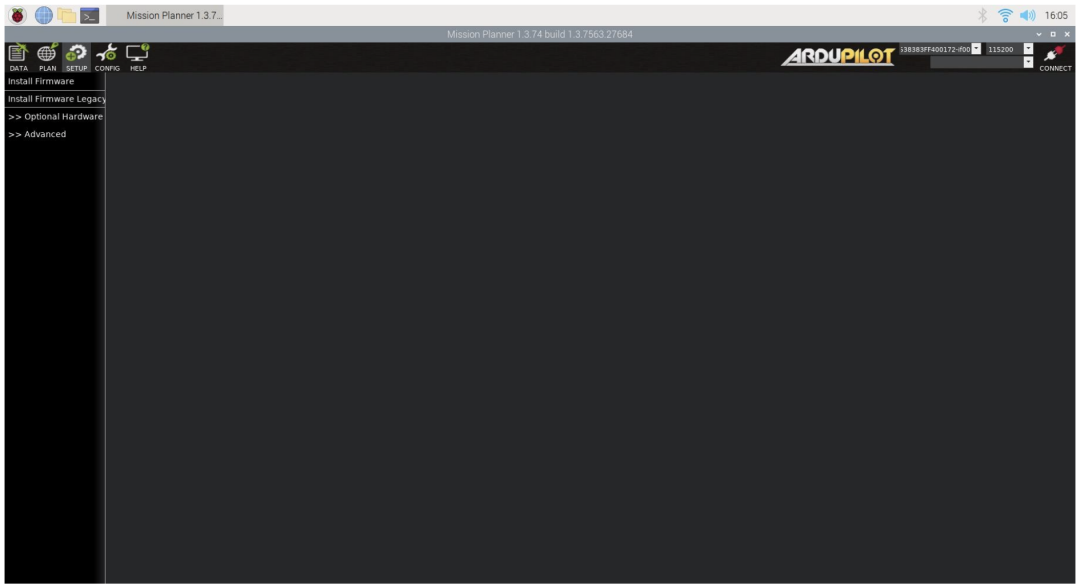
- Connect the Raspberry pi board to APM 2.8 using a Type-B USB cable.
- Open mission planner using the desktop icon created in the previous section.

- Select “/dev/serial/by-id/usb-Arduino_ww….” port from the dropdown menu in the top right corner.

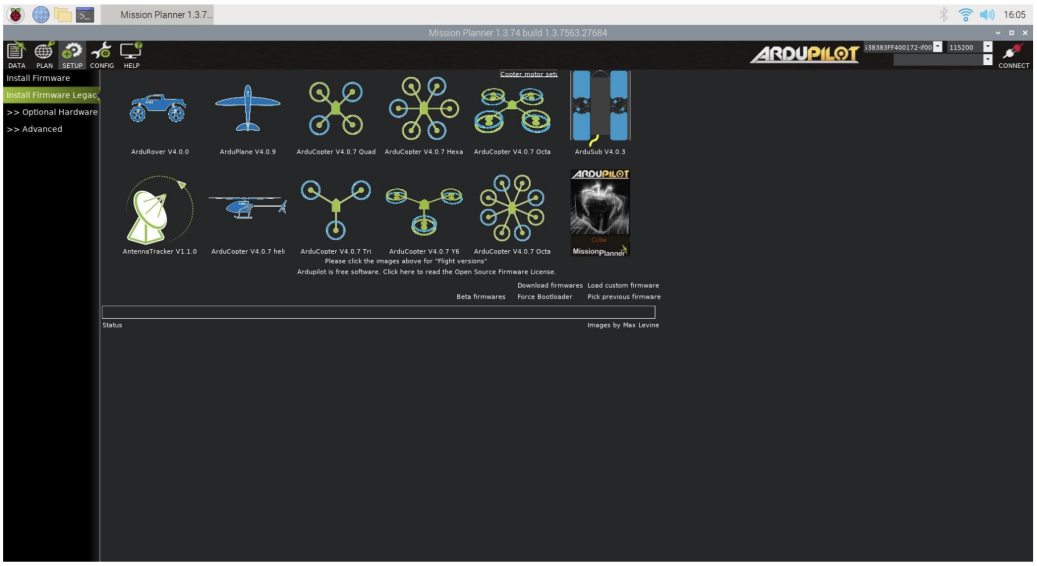


- Open the “Setup” menu from the top left corner, select “Firmware legacy”.

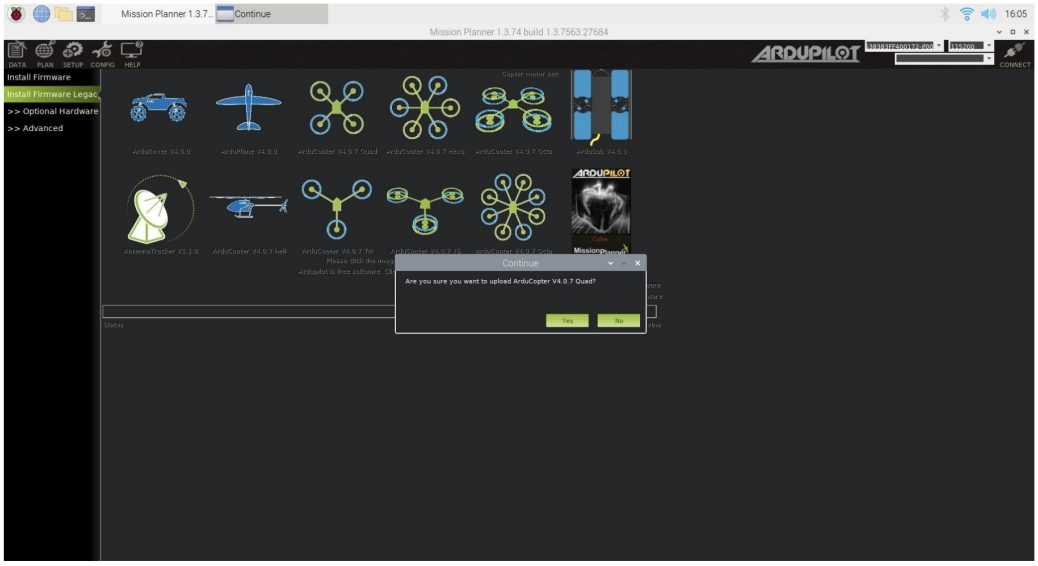
- Select “Firmware Legacy”.



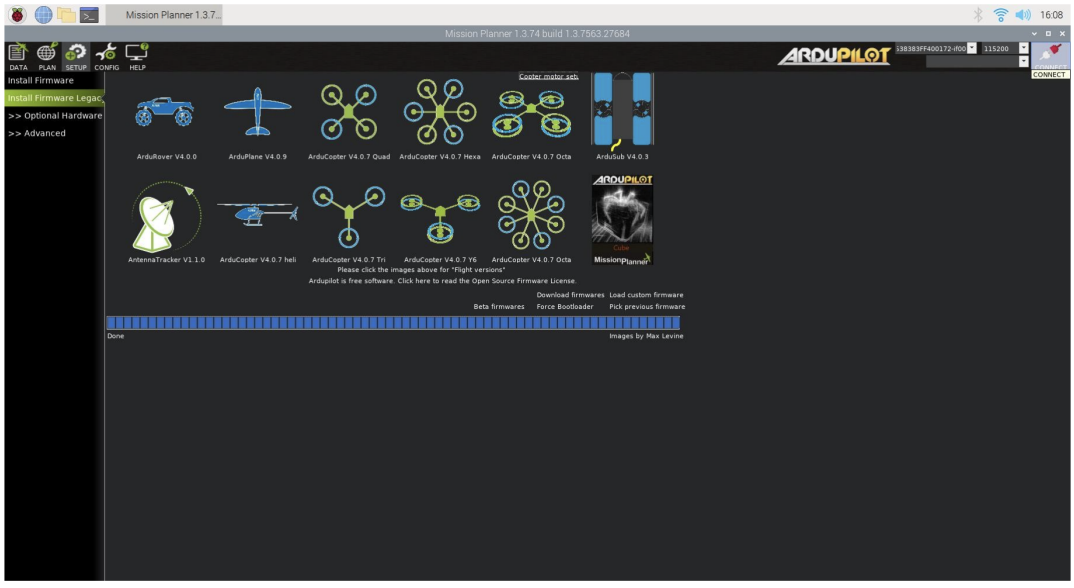
- Select the appropriate icon that matches your frame (i.e., Quad)



- Answer Yes when it asks you “Are you sure?”.

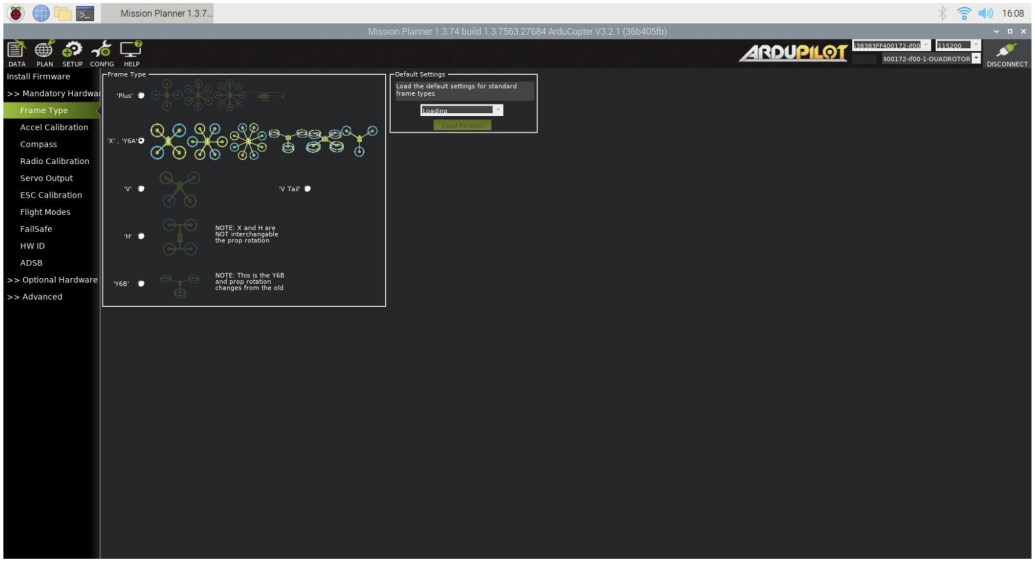


- If all goes well you will see some status appear on the bottom right including the words, “erase…”, “program…”, “verify…” and “Upload Done”. The firmware has been successfully uploaded to the board.
- It usually takes a few seconds for the bootloader to exit and enter the main code after programming or a power-up.
- Press CONNECT.



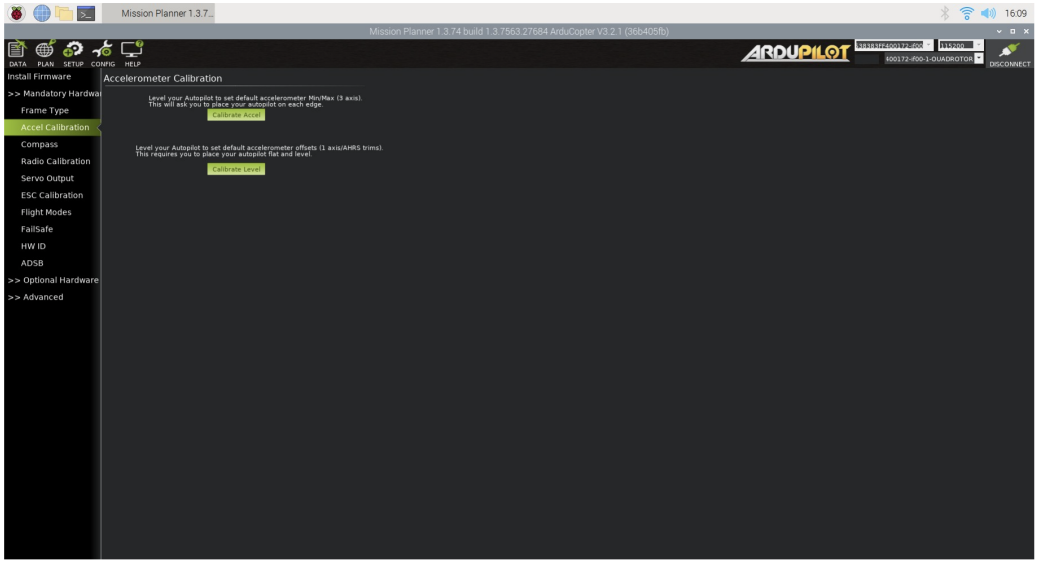
STEP2: Selecting Frame Type

- Under **Setup | Mandatory Hardware**.
- Select **Frame type** from the left-side menu and then choose “ ‘X’ . ‘Y6A’ ”.

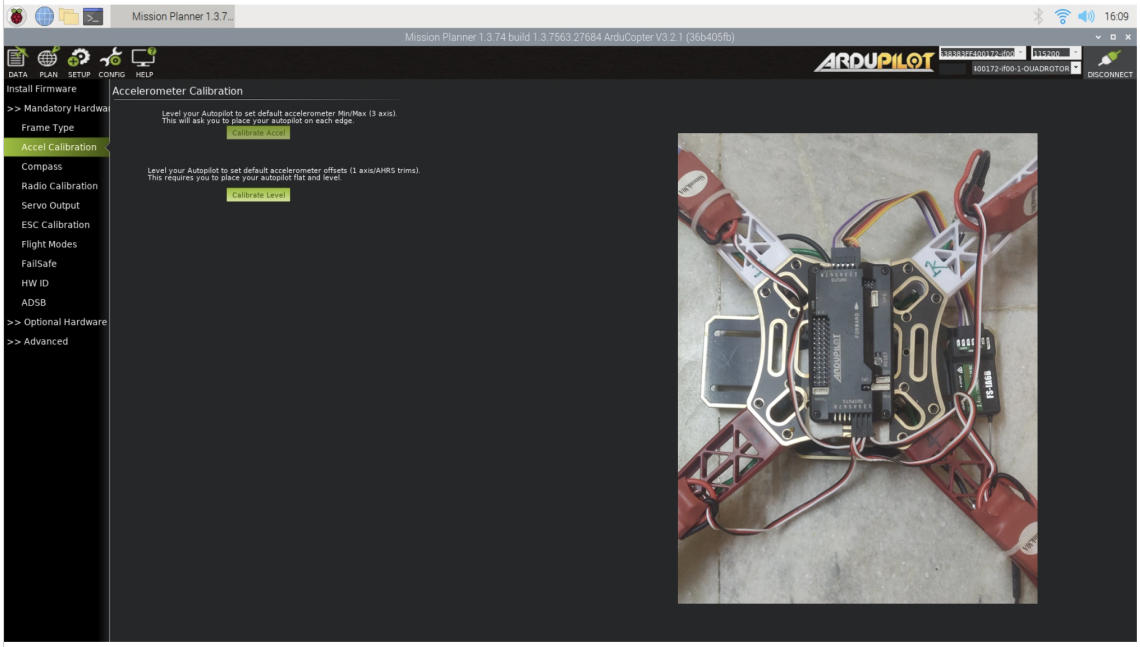


STEP3: Calibration of Accelerometer

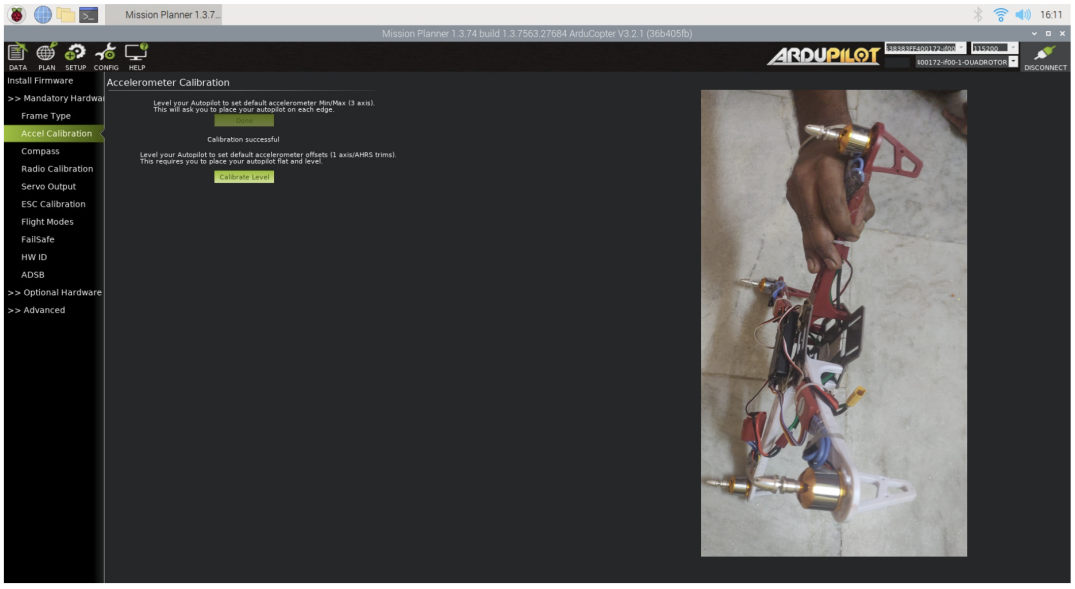
- Under **Setup | Mandatory Hardware**.
- Select **Accel Calibration** from the left-side menu.



- Click **Calibrate Accel** to start the calibration.

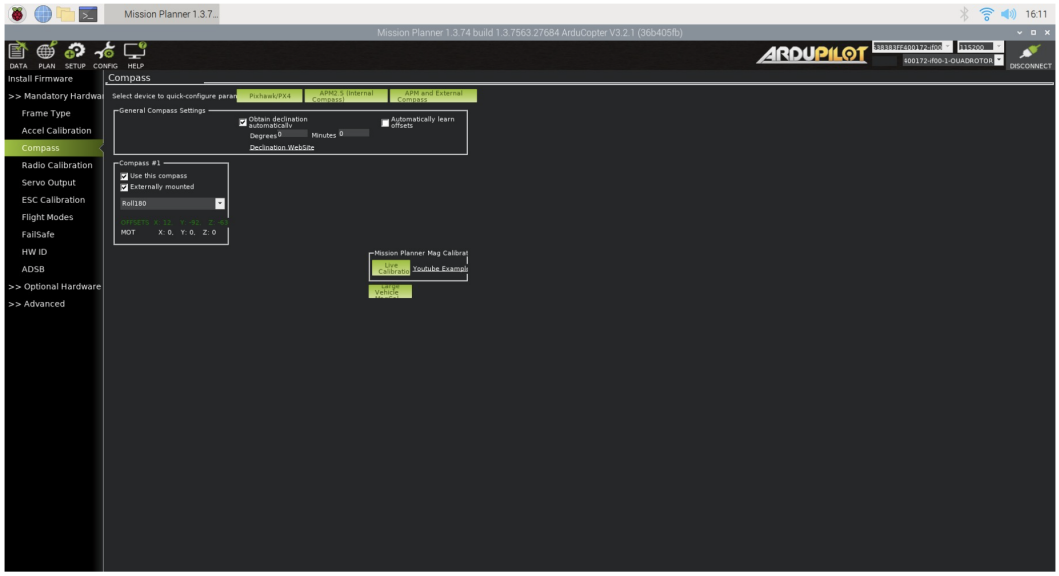


- Mission Planner will prompt you to place the vehicle in each calibration position.
- Press any key to indicate that the autopilot is in position and then proceed to the next orientation.
- The calibration positions are level, on the right side, left side, nose down, nose up, and on its back.
- Proceed through the required positions, using the Click when Done button after each position is reached.
- When you have completed the calibration process, Mission Planner will display “Calibration Successful!”

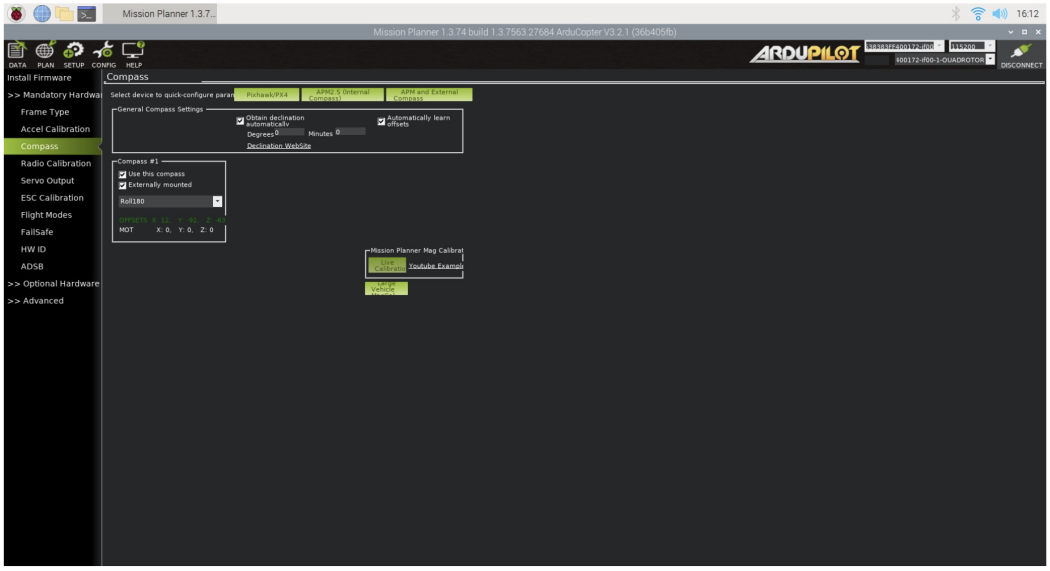


STEP4: Calibration of Compass

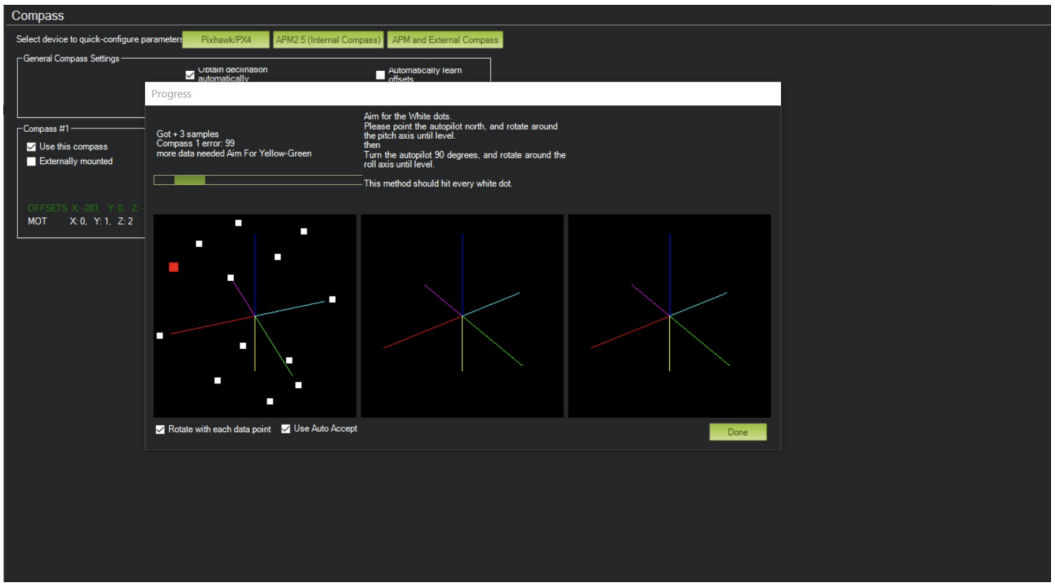
- Under **SETUP| Mandatory Hardware** select **Compass**.



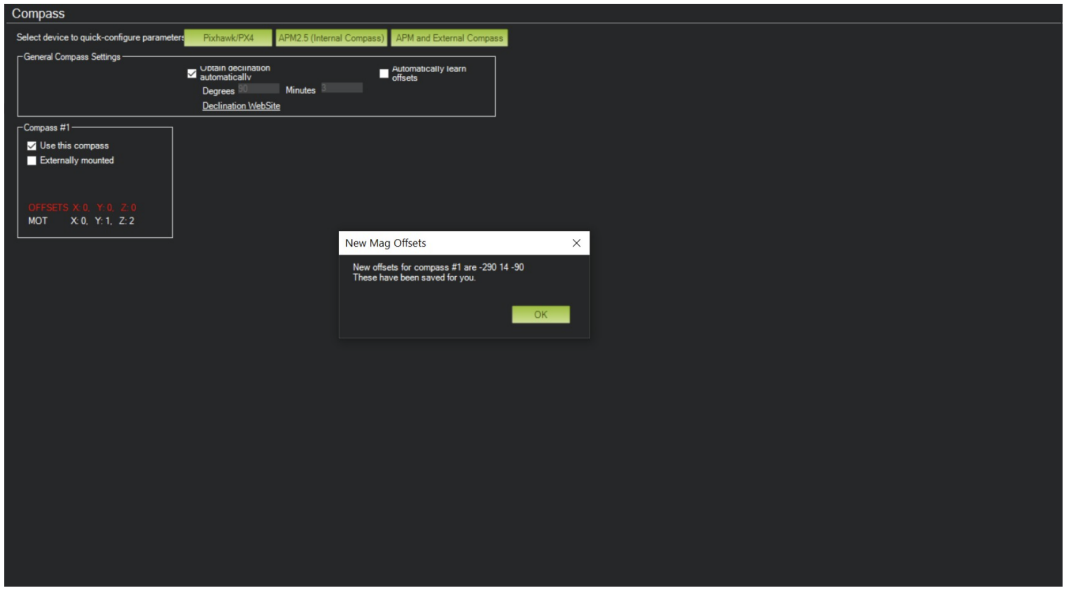
- Click on "**Live Calibration**".



- Rotate the Controller in all directions and make sure all the white spots are interconnected.



- Compass offsets will be recorded. Press **OK** to save the calibrated offsets.

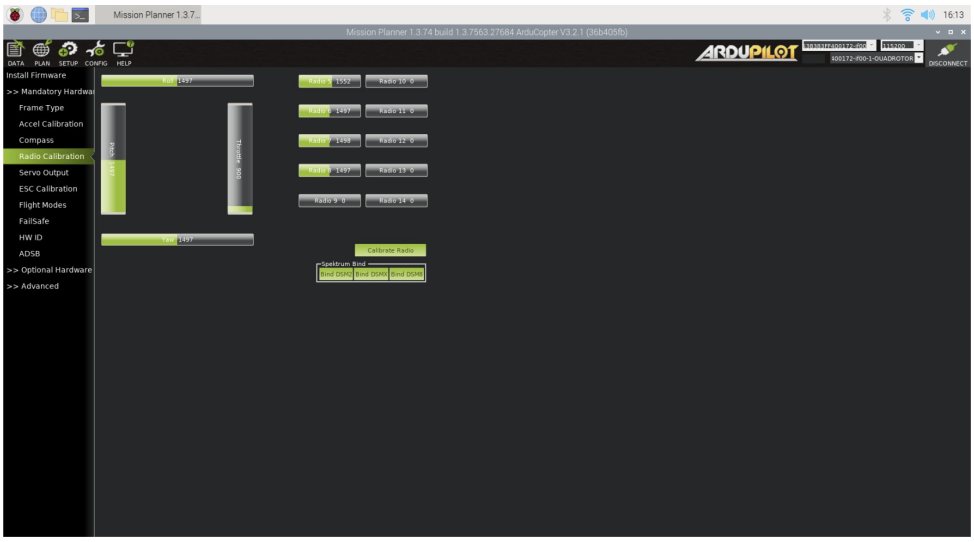


- After clicking on OK, the following screen appears showing the calibrated offset in green color.

STEP5: Calibration of Radio

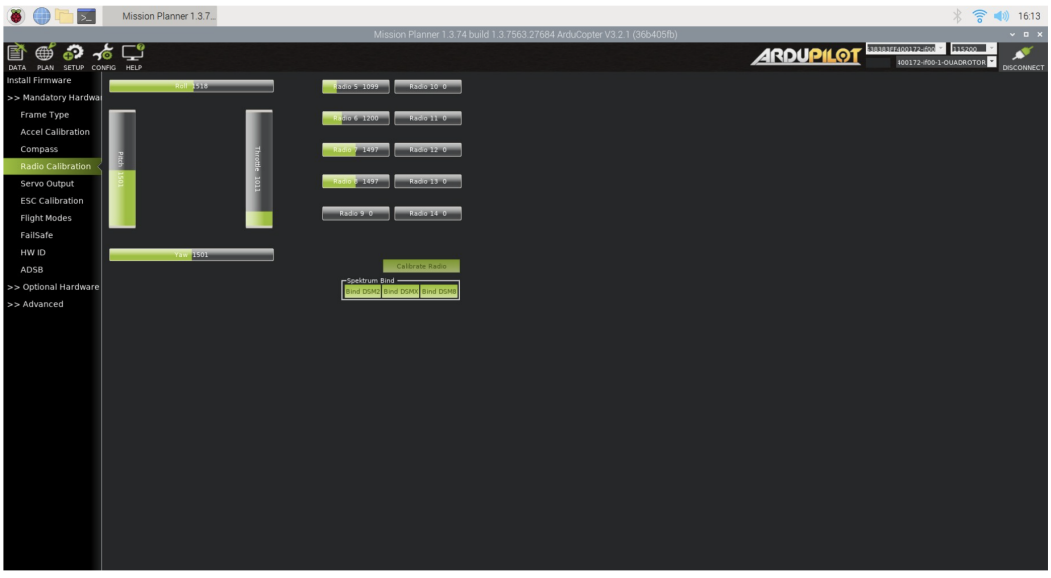
- Ensure the battery is disconnected (this is important because it is possible to accidentally arm the vehicle during the RC calibration process)
- Ensure the RC receiver is connected to the autopilot.
- Turn on your RC transmitter and if it has “trim tabs” ensure they are in the middle.
- Connect the autopilot to the PC using a USB cable.
- On the Mission Planner press the “Connect” button and open Mission Planner’s **SETUP** | **Mandatory Hardware** | **Radio Calibration** screen.

- Some green bars should appear showing the ArduPilot is receiving input from the Transmitter/Receiver.



- If no bars appear check the receiver's LED:
  - No lights may indicate that it is incorrectly wired to the autopilot. Look for connectors that may have been inserted upside down.
  - A Red or flashing LED may indicate that your RC transmitter/receiver needs to be bound. See the manual that came with your RC equipment for instructions.

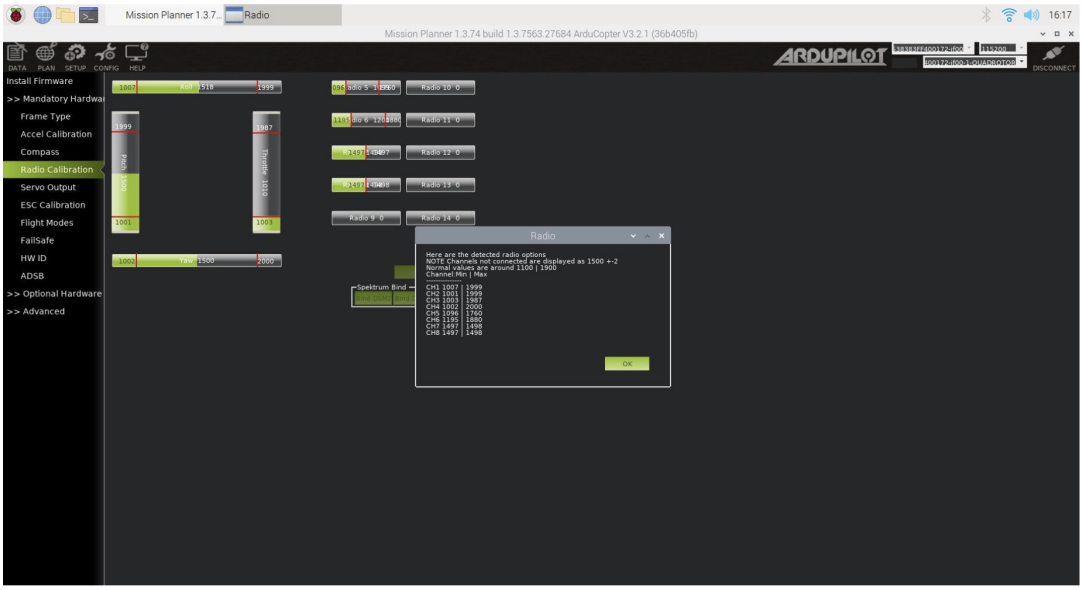
- Click on "**Radio Calibration**"



- Press "OK", when prompted to check the radio control equipment, is on, the battery is not connected, and propellers are not attached.

- Move the transmitter's control sticks, knobs and switches to their limits. Red lines will appear across the calibration bars to show minimum and maximum values seen so far. Select **Click when Done**.

- A window will appear with the prompt, "Ensure all your sticks are centered and the throttle is down and click ok to continue". Move the throttle to zero and press "OK".
- Mission Planner will show a summary of the calibration data. Normal values are around 1100 for minimums and 1900 for maximums.



STEP 6: Calibration of ESC (Electronic speed controller)

- Connect the ESC to be calibrated to the channel-3 of the receiver.
- Power ON the transmitter.
- Make the Throttle knob to its max position.

- Provide the power supply to the controller by connecting the battery.

- Now, Bring the Throttle knob position from max to min, you will **hear 2 beeps sound** which indicates the ESC under calibration is now calibrated.

- Repeat these steps for other ESCs.

Chapter 3

Beacon Tracking using ESP32 and Raspberry Pi.

3.1 Objective

The UGV should move towards the maximum vicinity(on the basis of RSSI or Received Signal Strength Indicator values) of beacon networks in an indoor environment in minimum time.

3.2 Required components/Software tools

- Raspberry Pi (3B model) for main controller
- ESP32 micro-controller with Type-B USB cable
- L293D Motor Driver IC
- UGV chassis with DC motors
- Breadboard
- Jumper Wires
- Android phone used as beacon

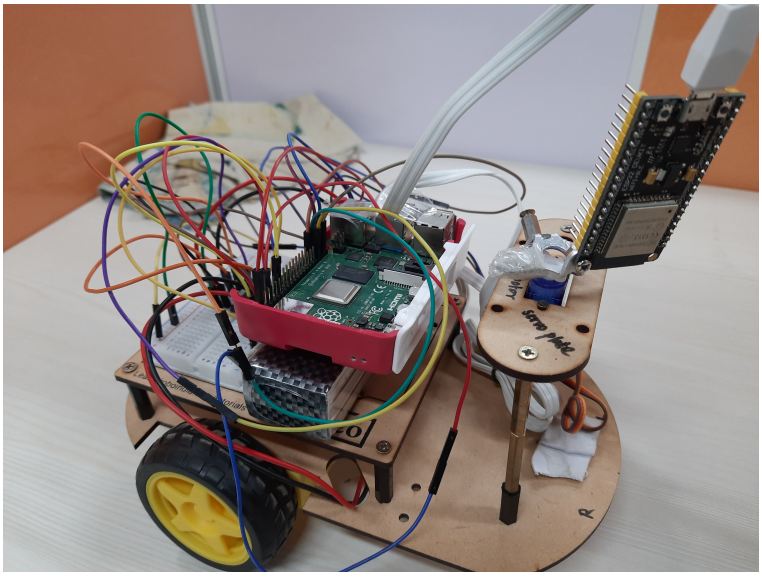


Figure 3.1: UGV kit assembly for beacon tracking

3.3 Methodology used:

3.3.1 Method 1:

Initially, we tried the moving average method on RSSI values received by the ESP32 module.

- Step 1: Initially, ESP32 mounted on the car will read RSSI (Received Signal Strength Indicator) levels in forward, right and left direction by suitable in-place rotation.
- Step 2: Average of 20 RSSI values are taken while measuring RSSI level in a particular direction. This is done in order to read stable RSSI values.
- Step 3: The car then rotates towards the direction having the highest RSSI level.
- Step 4: Further, It moves forward for a certain distance towards the beacon. By repeating the above steps again and again, the car navigates towards the beacon.

Note: Above method works but it takes more time to reach the beacon location. Hence, we moved towards the ML based algorithm i.e, KNN (K- Nearest Neighbours).

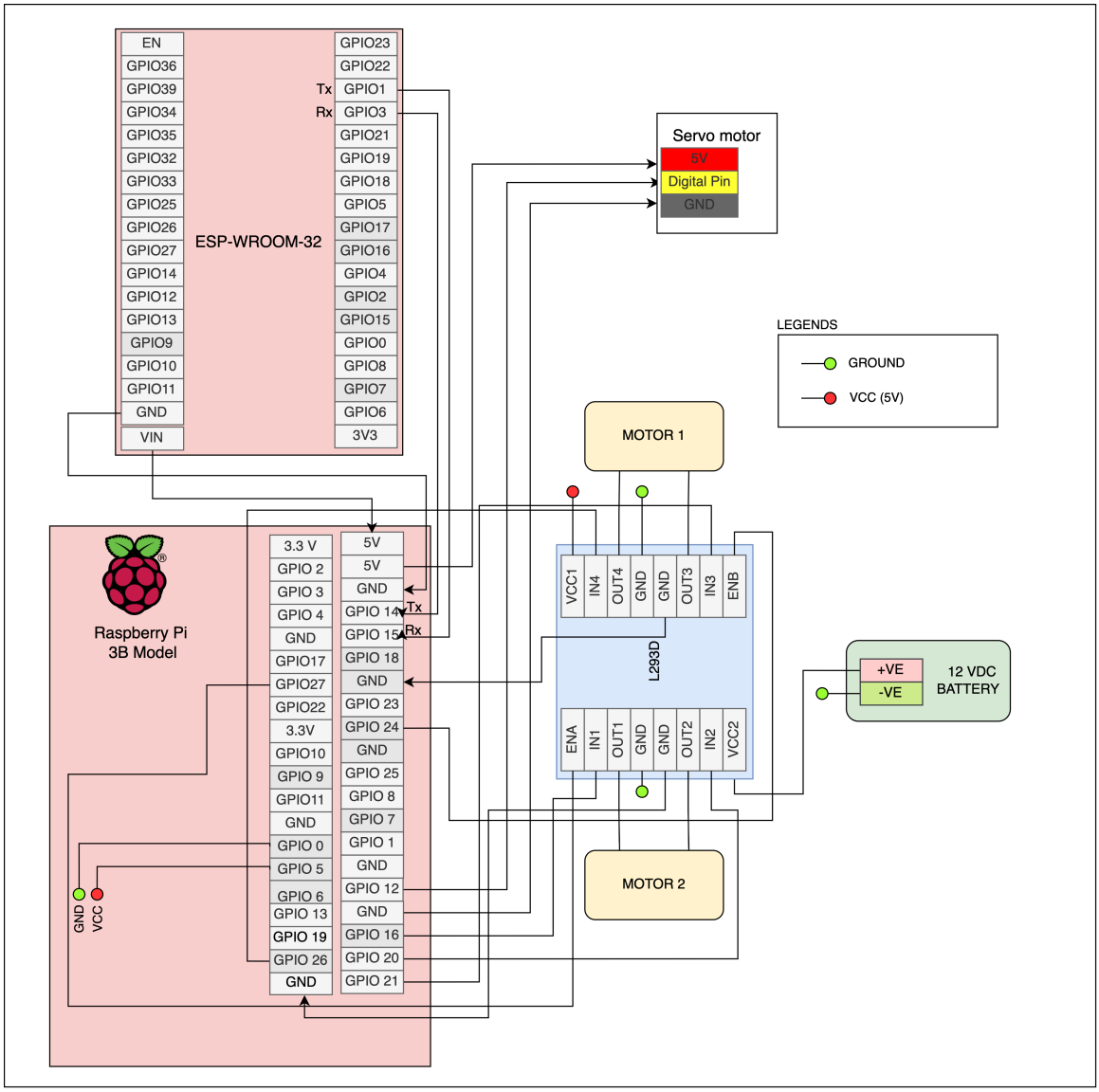


Figure 3.2: Wiring diagram for Beacon Tracking system (used in Method 2)

3.3.2 Method 2: KNN algorithm (Machine Learning based method)

- **Step1:** Collection of data in the lab. Data has been collected in an indoor environment (14x10 square feet area) Each block area is around 2 feet. (Refer Figure 3.3 and 3.4)

- **Step 2:** From each direction at each block (refer fig.3) , 15 continuous RSSI values were received by ESP32 module and sent those real-time values to our local machine(laptop) in a spreadsheet(for storing purpose) using raspberry pi. (Refer Figure 3.5)

- **Step 3:** We had collected (2040 x 5) data-set containing RSSI values coming from beacon, received by ESP32 in 5 different directions (eg. 0, 45, 90, 135, 180 degree angles) in spreadsheet

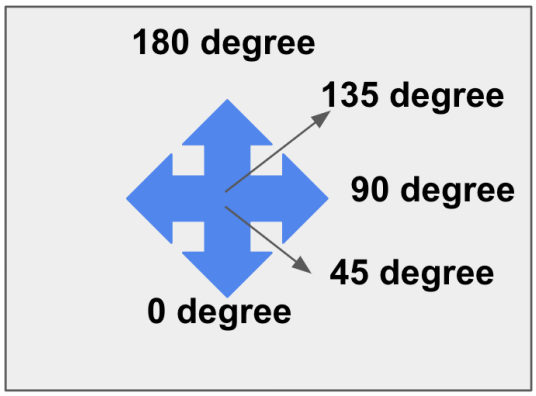


Figure 3.3: Angle reference from beacon on each block

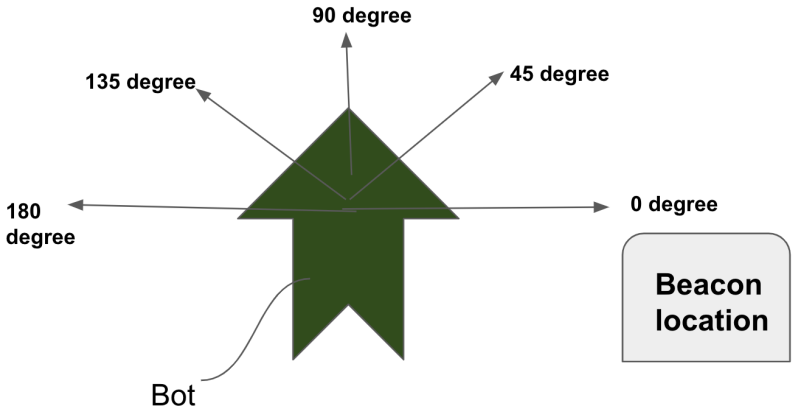


Figure 3.4: With reference to beacon, bot's required angle positions are described.

for future reference.(Refer figure 3.6)

- **Step 4:** We manually put encoded angular direction values for true label (total 5 classes):
  - 0 degree : 0
  - 45 degree : 1
  - 90 degree : 2
  - 135 degree : 3
  - 180 degree : 4
- **Step 5:** KNN Model training and testing
  - The K Nearest Neighbor algorithm comes under the Supervised Learning and is used for classification (most commonly) and regression also.

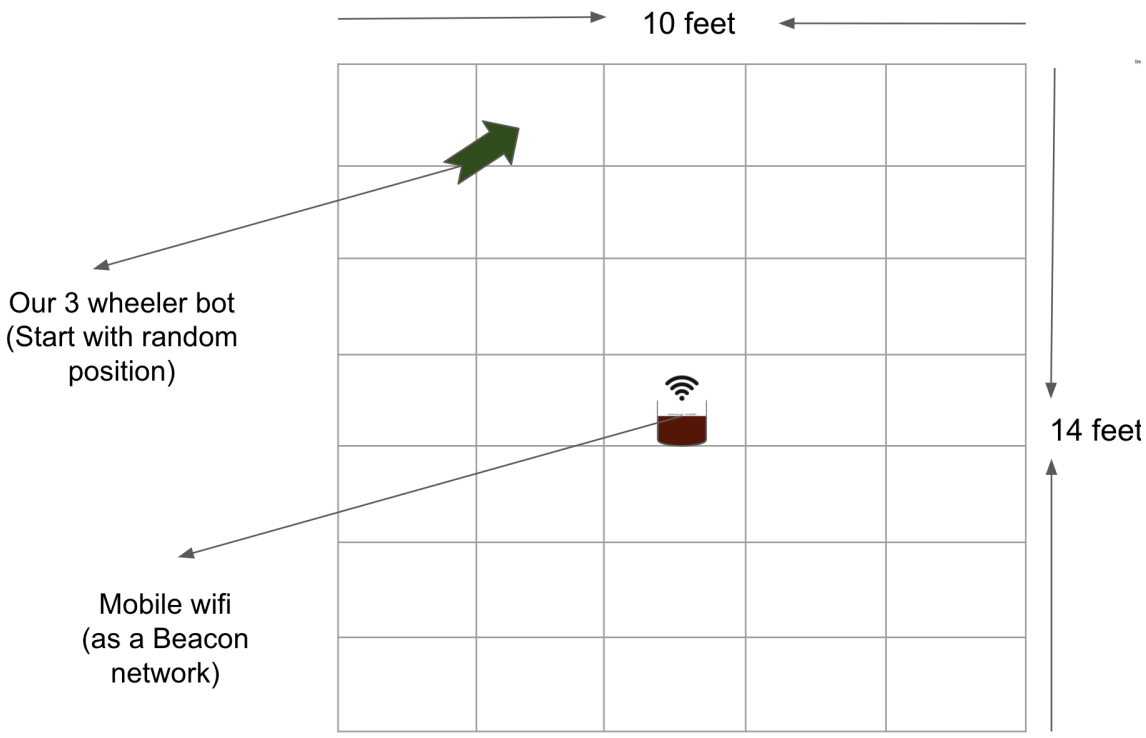
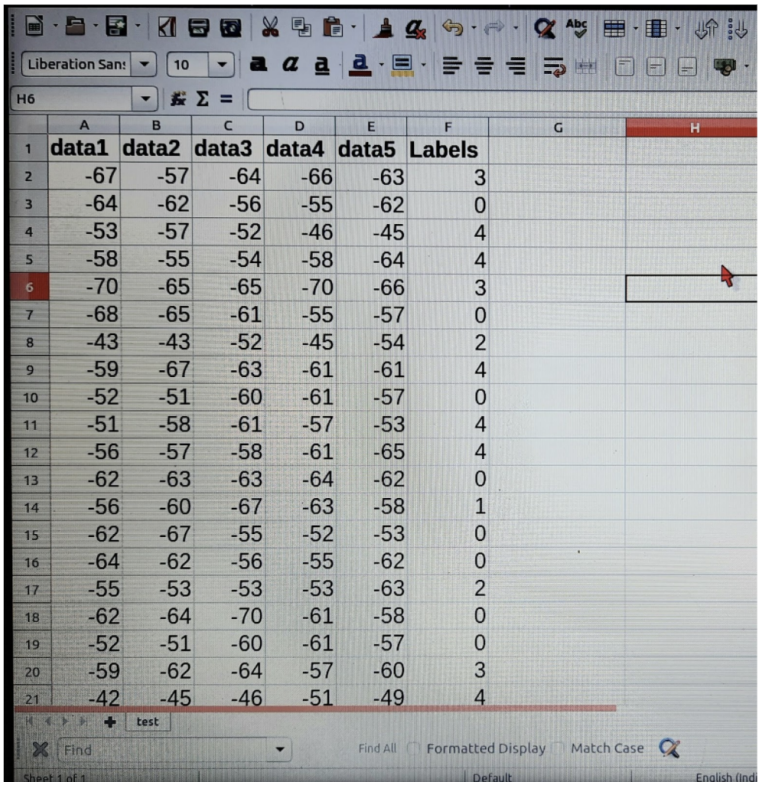


Figure 3.5: Arena for data collection

- It considers K Nearest Neighbors (data points) to predict the class or continuous value for the new data-point.
- In KNN, we do not learn weights from training data to predict output (as we do in model-based algorithms) but use entire training instances to predict output for unseen data.
- Step 1: The data-set which we have prepared, is splitted into 80:20 ratio for train test.
- Step 2: We have implemented the KNN algorithm on our data-set.
- Step 3: On new test data it gives approximately 77.08% accuracy (used sklearn library in Google colaboratory for model training).

Results and Conclusion: Although we tried KNN in order to reduce the time taken by the whole process as we did in method 1, KNN did not give consistent results on new test data points. Hence, we came upon the conclusion that we should solve this beacon tracking problem using method 1 with time and accuracy trade-off.



	A	B	C	D	E	F	G	H
1	data1	data2	data3	data4	data5	Labels		
2	-67	-57	-64	-66	-63	3		
3	-64	-62	-56	-55	-62	0		
4	-53	-57	-52	-46	-45	4		
5	-58	-55	-54	-58	-64	4		
6	-70	-65	-65	-70	-66	3		
7	-68	-65	-61	-55	-57	0		
8	-43	-43	-52	-45	-54	2		
9	-59	-67	-63	-61	-61	4		
10	-52	-51	-60	-61	-57	0		
11	-51	-58	-61	-57	-53	4		
12	-56	-57	-58	-61	-65	4		
13	-62	-63	-63	-64	-62	0		
14	-56	-60	-67	-63	-58	1		
15	-62	-67	-55	-52	-53	0		
16	-64	-62	-56	-55	-62	0		
17	-55	-53	-53	-53	-63	2		
18	-62	-64	-70	-61	-58	0		
19	-52	-51	-60	-61	-57	0		
20	-59	-62	-64	-57	-60	3		
21	-42	-45	-46	-51	-49	4		

Figure 3.6: A snap of our collected RSSI data

3.3.3 Code and Data-set link

```
https://github.com/Abhishek-IITH/Projects/tree/main/Beacon_tracking
```

Chapter 4

Implementation of Lane detection algorithm on UGV using Raspberry Pi.

4.1 Introduction

Lane detection is a primary step for vehicles to be autonomous. For self-driving cars, lane detection helps to navigate vehicles in predefined lanes. There are many algorithms which accomplish this task efficiently. It also helps the visual perception of vehicles in real-time navigation. I have used the simplest approach 'Hough Transform' algorithm for lane detection. The entire project is divided in 3 parts:

1. Gaussian Blur + Canny Edge Detection
2. Hough Transform
3. Displaying the lines + Thresholding

4.2 Software and Hardware Requirements

4.2.1 Libraries to be installed

- Numpy (used for performing mathematical computations)

- OpenCV (used to make computer vision applications)
- Matplotlib (used to visualize the images)
- Hough Transform Algorithm
- Raspberry Pi
- UGV kit (chassis, motors, motor driver IC, jumper wires)
- Pi camera

4.2.2 Hough Transform

In the Cartesian plane (x and y-axis), lines are defined by the formula y=mx+b, where x and y correspond to a specific point on that line and m and b correspond to the slope and y-intercept.

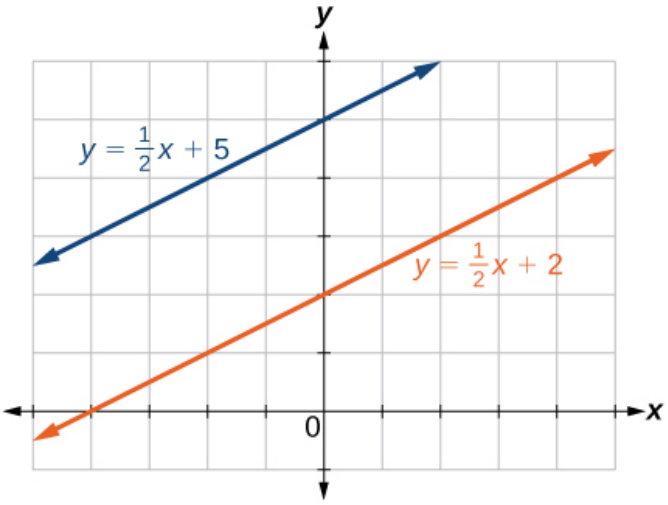


Figure 4.1: Lines in Cartesian Coordinate Space

A regular line plotted in the Cartesian plane has 2 parameters (m and b), meaning a line is defined by these values. Also, it is important to note that lines in the Cartesian plane are plotted as a function of their x and y values, meaning that we are displaying the line with respect to how many (x, y) pairs make up this specific line (there is an infinite amount of x, y pairs that makeup any line, hence the reason why lines stretch to infinity). However, it is possible to plot lines as a function of its m and b values. This is done in a plane called Hough Space. To understand the Hough Transform algorithm, we need to understand how Hough Space works.

Explanation of Hough Space

- Points on the Cartesian plane turn into lines in Hough Space
- Lines in the Cartesian plane turn into points in Hough Space
- We can find the line of best fit of two points in Cartesian space by finding the m and b coordinates of the POI (point of intersection) of the two lines that correspond with these points in Hough Space, then forming a line based on these m and b values.

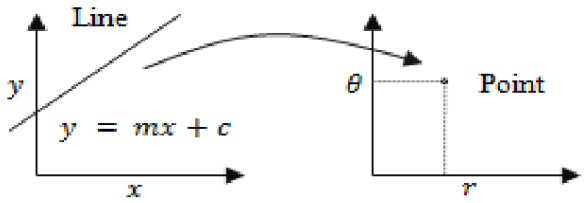


Figure 4.2: Explanation of Hough Space

A line is basically an infinitely long grouping of points arranged orderly one after the other. Since on the Cartesian plane, we're plotting lines as a function of x and y, lines are displayed as infinitely long because there is an infinite number of (x, y) pairs that make up this line. Now in Hough Space, we're plotting lines as a function of their m and b values. And since each line has its only one m and b value per Cartesian line, this line would be represented as a point.

4.3 Lane detection flow diagram

4.3.1 Steps

- Frame extracted from continuous stream of camera on Pi.
- Each frame is converted into an image array by using openCV library.
- Now, the image array is converted into a gray-scale image.This helps by increasing the contrast of the colours, making it easier to identify changes in pixel intensity.
- Using a Gaussian filter, we try to blur the image in order to reduce the noise in images.We do this because the gradients in Canny edge detection (in further step) are really sensitive to noise, so we want to eliminate the most noise possible.

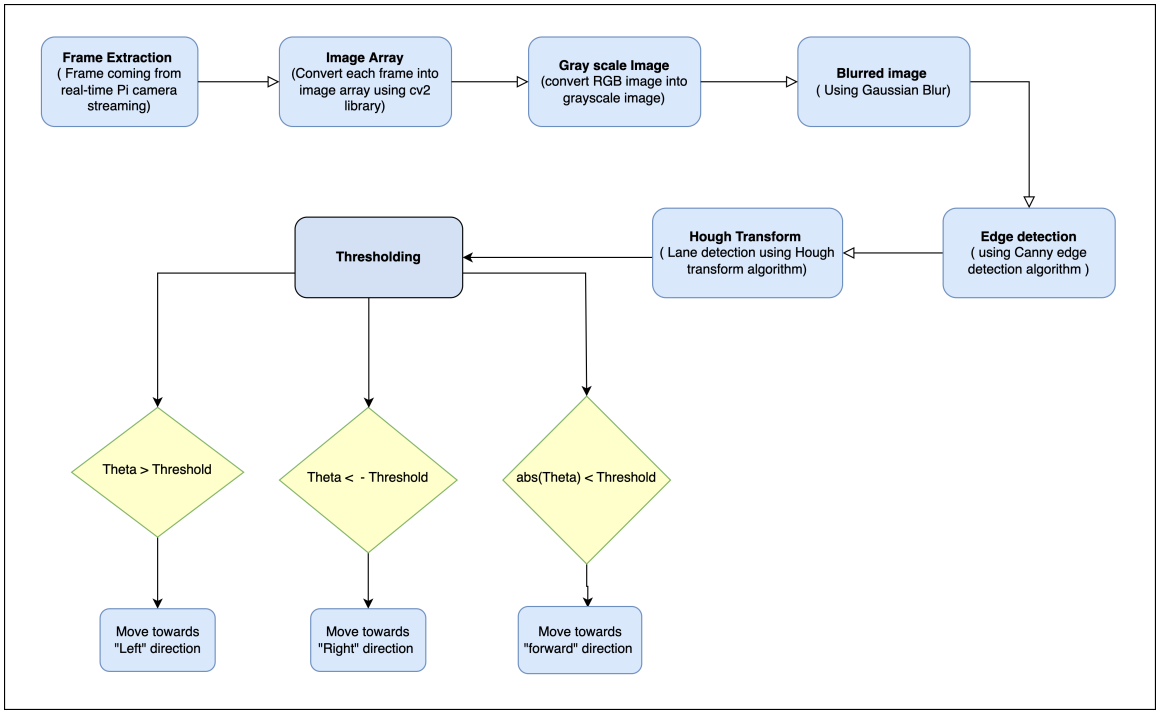


Figure 4.3: Flow diagram for Lane detection

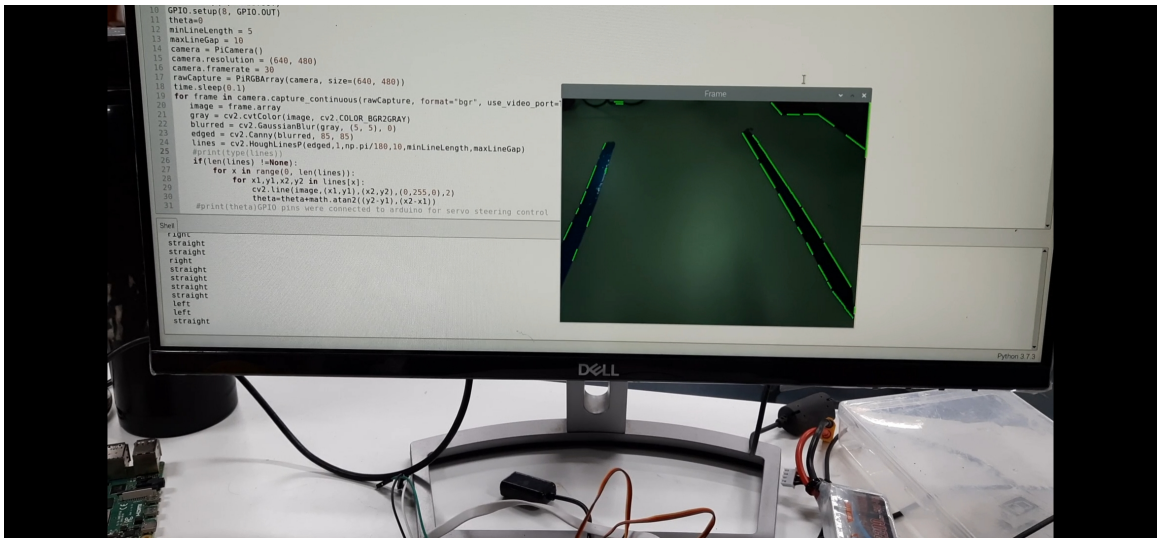


Figure 4.4: A glimpse of lane detection in lab

- Canny edge detection detects the edges present in images. It calculates the change of pixel intensity (change in brightness) in a certain section in an image.
- Now, we apply Hough transform on post canny edge detection images. This turns edges into lines that can be seen in images as detected lanes.
- After detection of lines, I have used thresholding techniques. The 'Theta' (averaging of lines)

is used to give command to motor driver IC to maintain the vehicle in lane (refer flow diagram for thresholding conditions).

- A continuous left, right or straight command is coming as output (for user reference) as per real-time scenarios. The vehicle takes those commands to maintain its navigation within lanes.

All files, codes and resources related to Lane detection can be found at below GitHub link

```
https://github.com/Abhishek-IITH/Lane-Detection-Project.git
```