A Project Report

On

**QuadCopter Navigation**

BY

**Saksham Subhash Yadav**

**2020A8PS2156H**

Under the supervision of

**Dr.Abhishek Sarkar**

**SUBMITTED IN FULLFILLMENT OF THE REQUIREMENTS OF**

**ME F366:LABORATORY PROJECT**



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)**

**HYDERABAD CAMPUS**

**(DECEMBER 2022)**

# ACKNOWLEDGMENTS

I appreciate the faith shown in me by the department of Mechanical Engineering to give an opportunity to me to work on this project which involves working on a QuadCopter along with a Pixhawk flight controller and a Raspberry Pi.  
I would like to thank Dr.Abhishek Sarkar for allowing me to work on this project and allowing to work on this project in accordance to my time schedule by allowing me to work with the components after lab hours.  
I am also grateful to Dr.Abhishek Sarkar and George Yuvaraj sir for guidance and help and for providing alternate solutions to the problems faced during the project work.

Lastly, I would like to thank my batch mates who helped me in formally,Bhavya Jain,Kalash Paripurnam, Pushkar Sapre and Abhiram Ravi along with Pranav and Pradyut who all helped me while debugging and flying drone.

****

**Birla Institute of Technology and Science-Pilani,**

**Hyderabad Campus**

# Certificate

This is to certify that the project report entitled “**QuadCopter Navigation”,** submitted by Saksham Subhash Yadav (ID No. 2020A8PS2156H), fulfills the requirements of the course ME F366, Laboratory Project, embodies the work done by him under my supervision and guidance.

**Date: 10th December,2022 (Dr.Abhishek Sarkar)**

BITS- Pilani, Hyderabad Campus

# ABSTRACT

Through this project, we intend to establish communication between drones or drones to the ground station, which serves as the backbone for the other work to be commenced on other subsystems, which is the first step needed to be explored before diving deeper into more niche subsystems of implementation based drone functionalities. Therefore, the major objective of the project is to first fly assembled drones and use motion planning to plan the trajectory of flight and collect sensor data at the ground station and lastly, keep a record of the steps involved in assembling the drones, the errors faced while working on it, the solutions to those problems and the learnings to have a handout ready for anyone to refer to who works on Pixhawk based drones.

Towards the end of the semester, We intend to connect Raspberry Pi with Pixhawk and establish data communication between the ground Station and the drone along with the flight logs which could be used for further analysis.

# CONTENTS

[**ACKNOWLEDGMENTS**](#_vkownqv2tald) **2**

[**Certificate**](#_d0x56k7gyywe) **3**

[**ABSTRACT**](#_6bucv4vgkkfa) **4**

[**CONTENTS**](#_9jmddnvq6a8z) **5**

[Components and their Specifications](#_kp2db34pww3a) 7

[Q450 Quadcopter Frame](#_uwhy579r4uk5) 7

[Pixhawk](#_wq1o9yk0lt5u) 7

[DJI 2212 920KV Brushless DC Motor](#_yeqhs687qwy6) 8

[Raspberry Pi 3 Model B+](#_5j46ave1s6pd) 8

[1045(10×4.5) SF Propellers Black](#_ac7rlcovnhmj) 9

[SimonK 30A BLDC ESC Electronic Speed Controller](#_xwlrprdhxfy0) 10

[Flysky Transmitter](#_4g2g1aan7plu) 10

[GPS Module](#_5zp93srjiq0q) 11

[Lipo Battery](#_hpjegoil7v1q) 11

[Assembly](#_v33n5u4ju73d) 13

[Step1:Soldering Wire Connections](#_id2wz9ygp68j) 13

[Step2: Assembling the Frame](#_4fp4tzhodcwt) 13

[Step3:Mounting Motors](#_4uibqwctwxai) 13

[Step4:Mounting ESCs](#_v6oi3g1rpvdm) 13

[Step5:Making Mock Connections](#_aomzvbcuvdt2) 14

[Step6:Labeling Motors](#_1hzl6mdorkbn) 14

[Step7:Placing the Flight Controller and the GPS module](#_wce3298d5jkp) 15

[Step8:Making Final Connections](#_bg3ddmnishha) 15

[Mission Planner](#_ugr795fbgkwe) 18

[Installing Mission Planner](#_340ysexihwar) 20

[LED/Buzzer Meanings](#_1umjnk8myadj) 20

[Tone Identification](#_6xxqd39isfgj) 23

[Safety Switch](#_959xro2epsle) 24

[LED meaning(LED on the Switch itself):](#_6idhtvwqke99) 24

[Buzzer (aka Tone Alarm)](#_wsbqybkh8rtz) 25

[Calibration and Testing](#_h6ror5k5dqg7) 27

[Radio Transmitter and Receiver Binding](#_z21e30ut4p5k) 27

[Initial Parameter SetUp](#_c6k8kwt0u87i) 27

[Accelerometer and Compass Calibration:](#_s9cnm55qherw) 28

[Radio Calibration](#_x4wfvqarh3ce) 29

[ESC Calibration:](#_o10pljr07ex2) 30

[Flight Mode:](#_svxk9qsrthms) 30

[FailSafe](#_1e0jc6qj493m) 35

[GeoFence](#_afo8i0jml810) 35

[Parameter Setup](#_o3bi4xvb8rdk) 36

[Logging](#_d33131j1uado) 38

[ArmCheck](#_ih6q1vbq83ok) 39

[Raspberry Pi Communication](#_pmxtucpymla5) 40

[Raspberry Pi 3 As A Companion Computer For Pixhawk](#_rocs38vz0cp9) 40

[Installing OS:](#_pjvlkm8bqwe3) 40

[SSH using PuTTY:](#_328l5nu5b2i) 44

[Connections between Pixhawk and Raspberry Pi:](#_vv26uei3ts9l) 48

[Installing Libraries:](#_bjlceefedgnl) 48

[Conclusion](#_daue3zcvf8z2) 50

[Bibliography](#_wnqmwjxz0aem) 51

## 

## Components and their Specifications

### Q450 Quadcopter Frame

* Wheelbase: 450mm
* Material: Glass Fiber + Polyamide-Nylon
* Weight: 330 gm
* Motor Mounting Hole Dia.: 3 mm
* Arm Size: 220 x 40 (LxW) mm
* Landing Gear Material: ABS
* Landing Gear Height: 200 mm

### Pixhawk

| **Model** | PIX 2.4.8 32 Bit |
| --- | --- |
| **Input Voltage (V)** | 7V |
| **Firmware** | Mission Planner |
| **Sensors** | 3-Axis Gyrometer  Accelerometer  High-performance Barometer  Magnetometer |
| **Processor** | 32 bit STM32F427 Cortex M4 core with FPU  The 32-bit STM32F103 failsafe Co-processor |
| **Micro-SD Card Slot** | Yes |
| **Dimensions (mm) LxWxH** | 82 x 50 x 16 |
| **Weight (gm)** | 40 |

### DJI 2212 920KV Brushless DC Motor

* Motor KV: 920 RPM/V
* Motor Rotation: CCW for Silver Cap
* Motor Rotation: CW for Black Cap
* Thrust: Around 0.5 kg
* LiPO Batteries: 3S-4S
* Rated Voltage(V):7~12
* ESC: 30 A
* Shaft Diameter: 6 mm
* Dimensions(mm):28\*28\*46
* Weight: 60 g

### Raspberry Pi 3 Model B+

* The Raspberry Pi 3 Model B+ is the final revision in the Raspberry Pi 3 range.
* Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
* 1GB LPDDR2 SDRAM
* 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
* Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
* Extended 40-pin GPIO header
* Full-size HDMI
* 4 USB 2.0 ports
* CSI camera port for connecting a Raspberry Pi camera
* DSI display port for connecting a Raspberry Pi touchscreen display
* 4-pole stereo output and composite video port
* Micro SD port for loading your operating system and storing data
* 5V/2.5A DC power input
* Power-over-Ethernet (PoE) support (requires separate PoE HAT)

### 

### 1045(10×4.5) SF Propellers Black

* Length: 10″.
* Pitch: 4.5″.
* Weight: 14 gm.
* Shaft Diameter: 6 mm.
* Total length: 10 inch / 254 mm.

### 

### SimonK 30A BLDC ESC Electronic Speed Controller

* Model: SIMONK 30A.
* Constant Current: 30A (Max 40A < 10 sec).
* BEC: 5V 2A.
* Suitable Batteries: 2-3S LiPo.

### Flysky Transmitter

| No. of Channels | 6. |
| --- | --- |
| RF Range (GHz) | 2.40 ~ 2.48 |
| Bandwidth (kHz) | 500 |
| RF Power (dBm) | ≤20 |
| Sensitivity | 1024 |
| Low Voltage Warning (V) | ≤ 4.2 |
| Power | 6V (4 x 1.5AA Not Included) |
| Display mode | Transflective STN positive type, 128\*64 dot-matrix VA73\*39mm, white backlight |

### 

### GPS Module

* Locate performance
* These are Pre-configured, Flashed with the correct settings, and tested. To make them Plug and Play.
* Super Bright LED
* Backplane with Standard Mk style mounting holes 45mm X 45mm
* 38400 bps (Default) Changed to 115200 bps!
* Output GGA, GSA, and RMC frames
* 1Hz (Default) Changed to 5Hz!
* Permanent Configuration Retention
* compass on board
* 6-pin connector for EZ connect to MEGA BLACK
* 4-pin connector for only GPS use
* 4-pin connector for compass only use
* Can use both four pins at once.

### Lipo Battery

| Model No. | ORANGE 5200 / 3S-40 |
| --- | --- |
| Capacity (mAh) | 5200 |
| Weight (gm) | 360 |
| Output Voltage (VDC) | 11.1 |
| Charge Rate (C) | 1 ~ 3 |
| Discharge Plug | XT-60 |
| Balance Plug | JST-XH |
| Length (mm) | 137 |
| Width (mm) | 44 |
| Height (mm) | 37 |
| Max. Burst Discharge (C) | 80C(416.0A) |
| Max. Charge Rate | 5 C |
| Max. Continuous Discharge | 40C(208.0A) |

### 

## Assembly

#### Step1:Soldering Wire Connections

Take the bottom frame with the printed circuit board and solder wires to connect to ESCs later, which would be used to power the motors.

#### Step2: Assembling the Frame

Now keep the PCB frame as the bottom plate, and attach it to four arms with two screws each. Now attach the other plate on top with four screws each. On one of the arms, attach the GPS stand mount which would be used to keep the GPS module a few inches above the frame level. Take care not to fully tighten the screws before placing all the screws in place. Once you have all the (2\*4+4\*4)=24 screws in place, tighten all of the screws to achieve tension in the frame. In case it is tough to attach four screws on each arm owing to the alignment issue of the screws and slots, three screws would be enough.

#### Step3:Mounting Motors

We have two different sets of motors, the one rotating in the clockwise direction and the other rotating in the counterclockwise direction. So we need to mount the motors to the outer edge of the arm on the slots provided with four screws each. Take into account that the motors rotating in the same direction are supposed to be mounted on diagonally opposite arms to cancel out torque which would topple the drone if both the motors on the same side of the drone rotate in the same direction.

#### Step4:Mounting ESCs

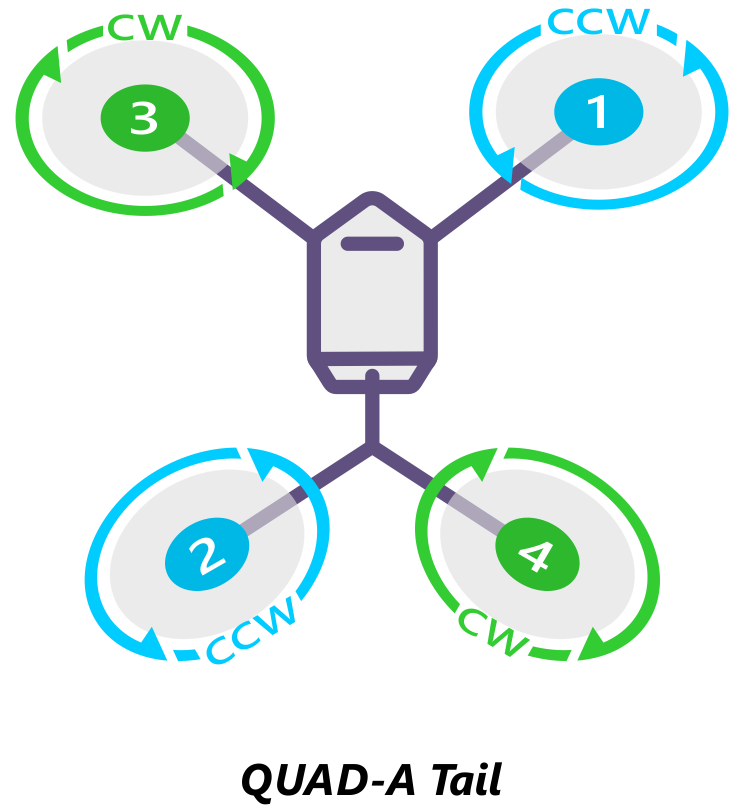
We can now attach the ESCs tightly on the arm's length with the side having two sets of three wires facing each of the motors with the aid of zip ties or insulation tapes. One set of three wires is supposed to be connected to the motors, while the three jumper wires are meant to be connected to the flight controller.

The lone set of two wires on the other side of the ESC is meant to be connected to the wires we soldered to the PCB on the lower plate of the frame earlier. Before attaching the ESC, check for the wires and if they are of the appropriate length to connect to the corresponding wires. For jumper wires we might need extra jumper wires to increase the length.

#### Step5:Making Mock Connections

Connect the ESCs to the motor, then the other set of wires to the soldered wire on the frame. After that, connect the jumper wires of the ESCs to pins on the side labelled as the MAIN OUT. The wiring colours on ESC are either orange, red and brown, or white, red and black corresponding to the signal, positive, and negative terminals. The top row of the Pixhawk slots is for the signal wire, and the bottom row is for the ground wire from the ESC. We

#### Step6:Labeling Motors

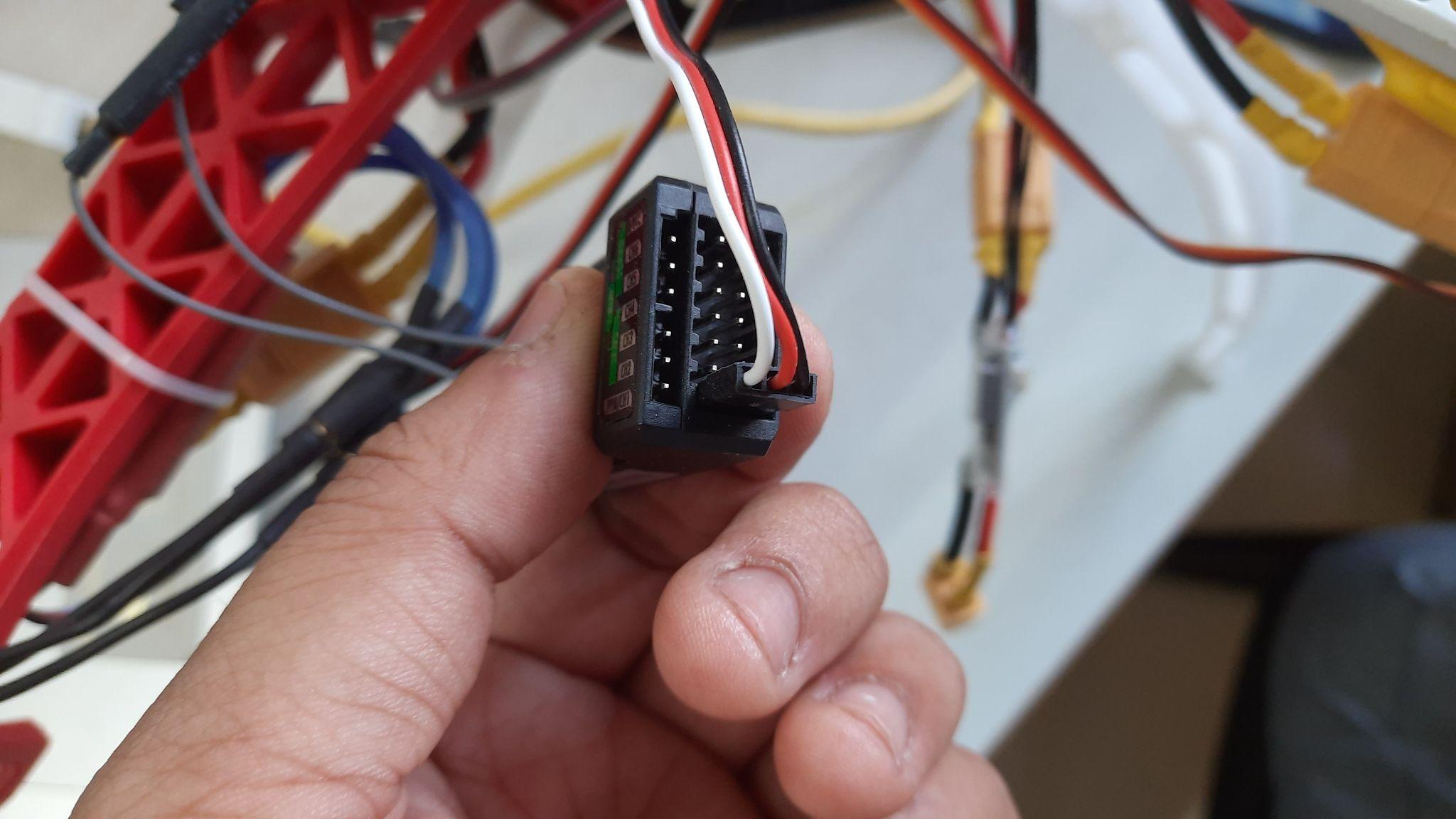
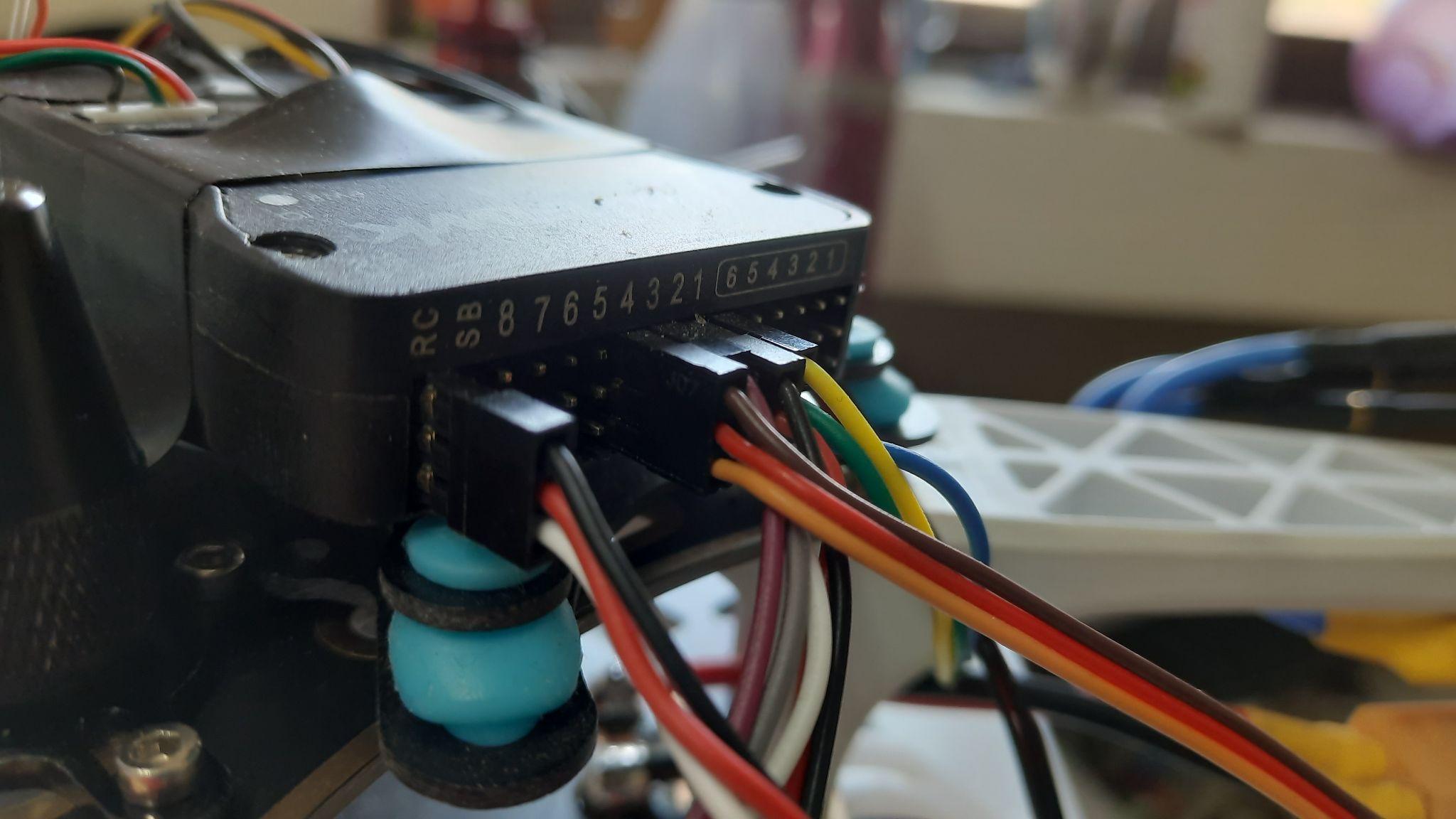
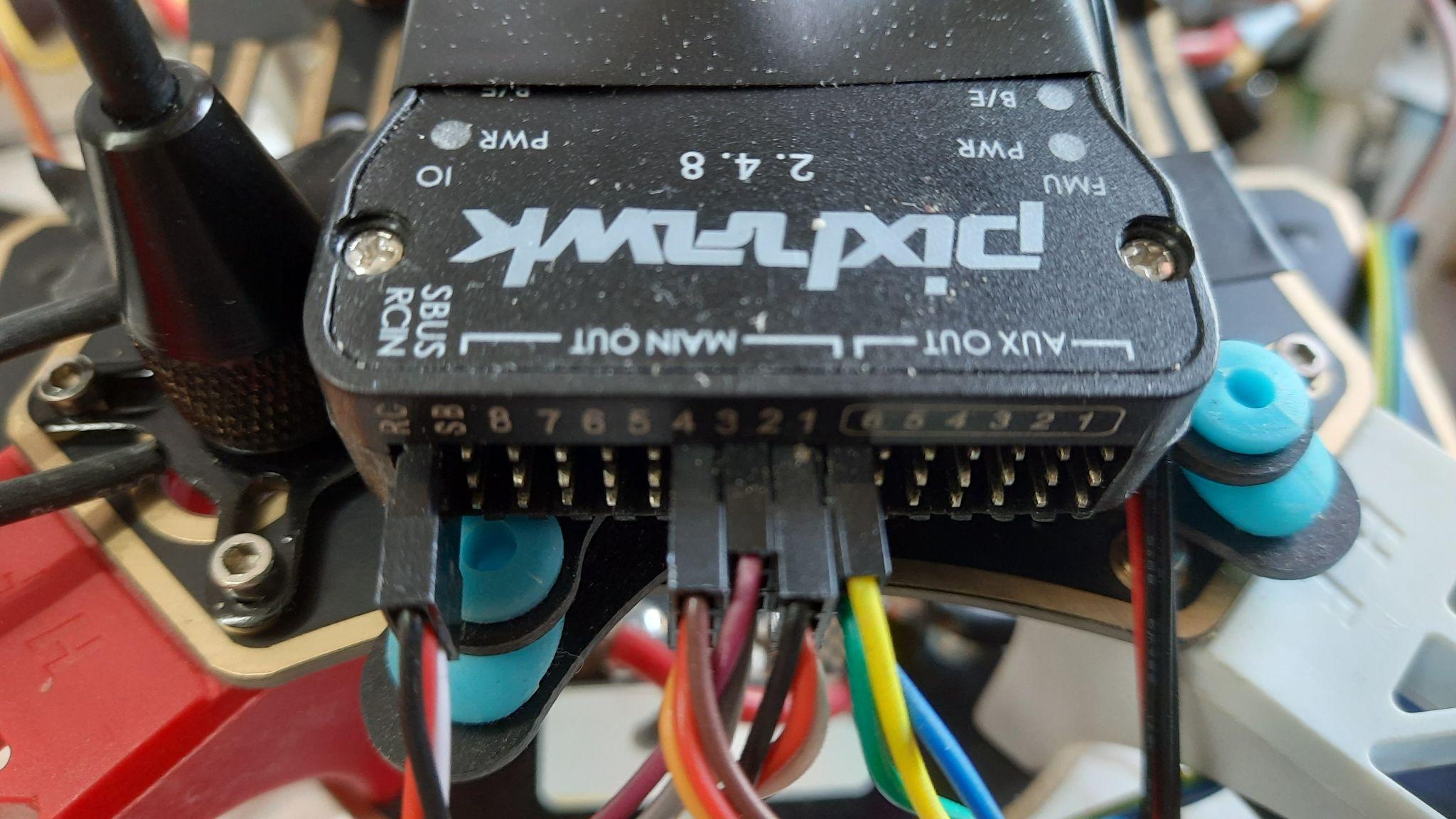


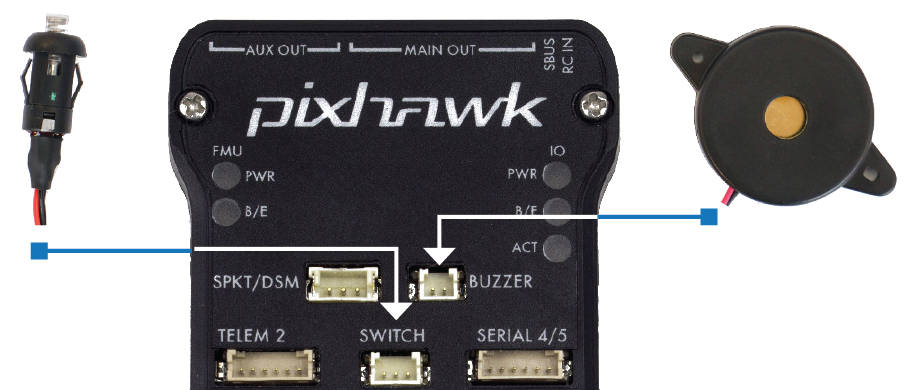
Now label the motor as shown in the figure and connect the wire from the esc in the same order from slot 1 to 4 of the MAIN OUT of the Pixhawk.

#### Step7:Placing the Flight Controller and the GPS module

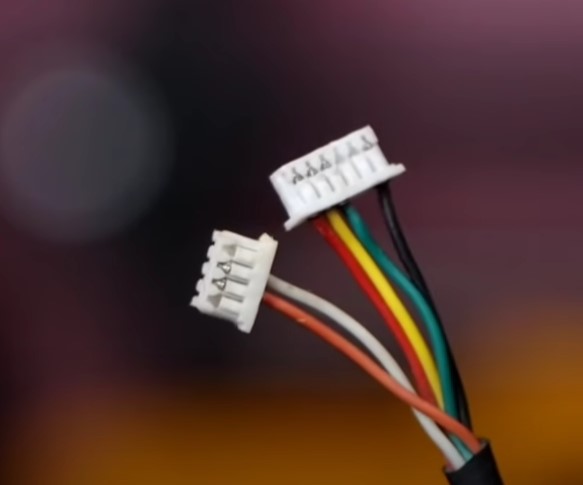
Attach the flight controller using double-sided tape on a shock absorber, which is attached to the frame and attach the GPS module on top of the stand.

#### Step8:Making Final Connections

Connect the ESC as labelled to the MAIN OUT of the flight controller. Connect the PPM slot of the receiver to the RC slots of the flight controller as shown in the figure.

Connect the switch and buzzer as shown in the figure below.

Connect the three wired pins of the GPS module to the i2c Socket of the pixhawk and the other pin to the GPS slot.

Lastly, connect the wire from the voltage supply to the power supply slot on the flight controller.

**Step9:**

Now that the whole assembly is done, we need to be aware of the few alerting mechanisms of the Pixhawk to prevent any harm to the drone. After that, we can calibrate all the sensors and be ready to fly them. We also need to be aware of the Mission Planner software we would use for calibration, mission planning and finding flight log files for further analysis.

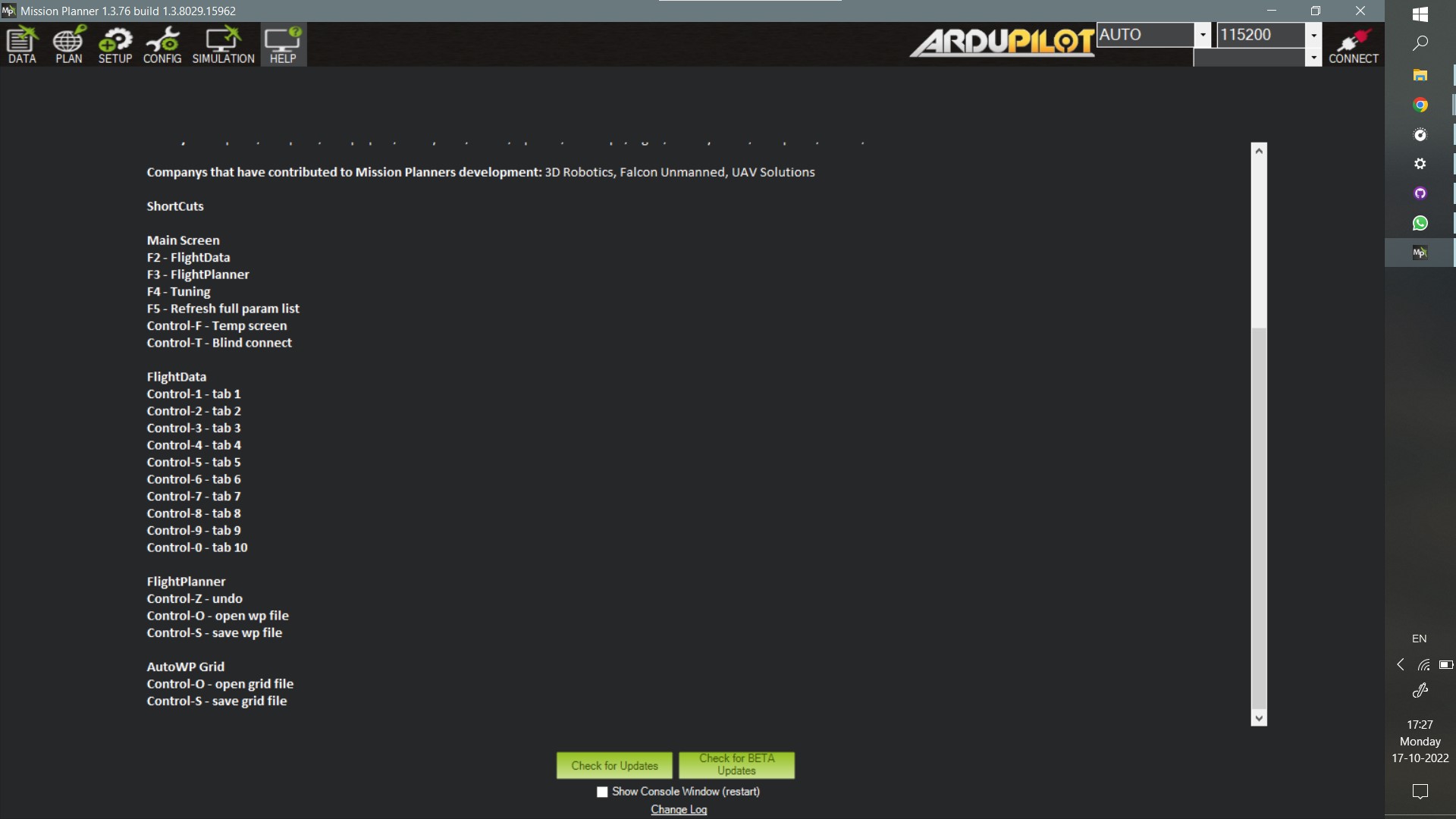
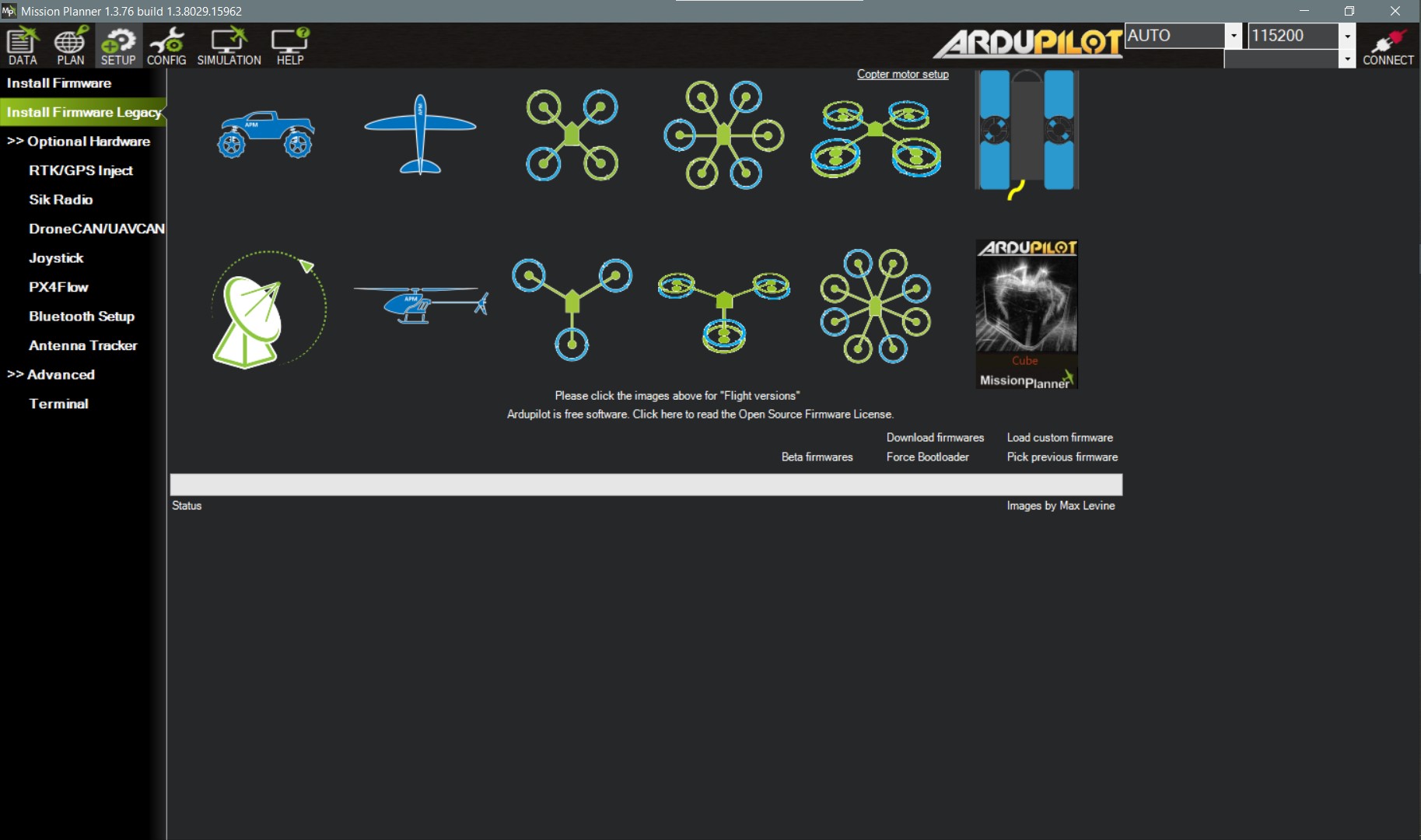
## 

## Mission Planner

Mission Planner is one of many control Station software available in the market. It specialises in ardupilot based flight controllers for Plane, Copter and Rover. The following screenshots are from the Mission Planner Software (V1.3.76).

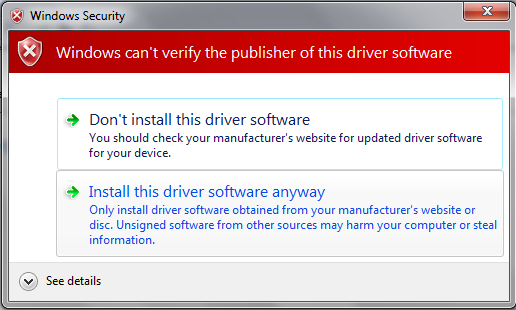






### Installing Mission Planner

Mission Planner was initially designed for Windows, but a beta version is available for Android, and a beta version for Linux is available too. Double-click the downloaded .msi file to run the installer from the [Ardupilot website](https://ardupilot.org/planner/docs/mission-planner-installation.html).

Follow the instructions to complete the setup process. The installation utility will automatically install any necessary software drivers. If you receive the warning pictured below, select Install this driver software to continue.  


## 

## LED/Buzzer Meanings

**Flashing blue and red:** Initializing gyroscopes. Hold the vehicle still and level while it initialises the sensors.

**Flashing blue:** Disarmed, no GPS lock found. Autopilot, loiter, and return-to-launch modes require GPS lock.

**Flashing green:** Disarmed (ready to arm), GPS lock acquired. Quick double tone when disarming from the armed state.

**Fast Flashing green:** Same as above, but GPS is using SBAS (so it should have a better position estimate).

**Solid blue:** Armed with no GPS lock.

**Solid green with a single long tone at the time of arming:** Armed, GPS lock acquired. Ready to fly!

**Double flashing yellow:** Failing pre-arm checks (system refuses to arm).

**Single Flashing yellow:** Radio failsafe activated

**Flashing yellow - with sharp beeping tone:** Battery failsafe activated

**Flashing yellow and blue - with high-high-high-low tone sequence (dah-dah-dah-doh):** GPS glitch or GPS failsafe activated

**Flashing red and yellow - with rising tone:** EKF or Inertial Nav failure

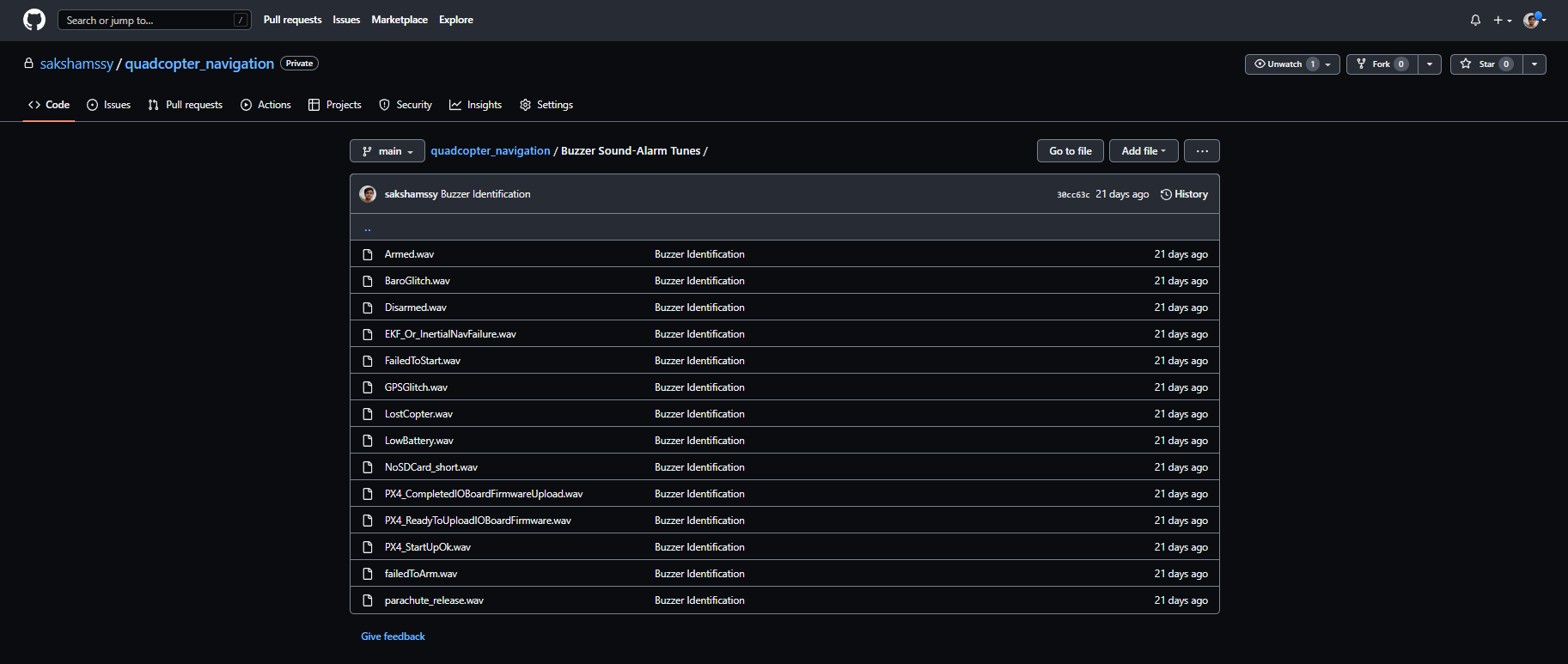
**Flashing Red, Blue and Green:** Copter ESC Calibration mode entered.

| **State** | **LED** | **Description** |
| --- | --- | --- |
| Initializing (calibrating gyros) | rgbinit | Blinking Red & Blue. |
| Save Trim or ESC Calibration | rgbsavetrim | Blue-Green-Red |
| Leak failsafe | rgbleakfs | Slow Yellow and White |
| EKF failsafe | rgbekffs | Slow Red and Yellow |
| GPS glitch | rgbgpsglitch | Slow Blue and Yellow |
| Radio/GCS/Batt failsafe | rgbcontrolfs | Fast Yellow and White |
| ARMED:3Dfix | rgbarmed | Solid Green |
| ARMED: No valid GPS fix | rgbarmednogps | Solid Blue |
| DISARMED:pre-arm checks failing | rgbprearm | Blinking Yellow and White |
| DISARMED: good DGPS fix | rgbready1 | Fast Blinking Green |
| DISARMED: good GPS fix | rgbready2 | Slow Blinking Green |
| DISARMED: bad GPS fix | rgbbadgps | Blinking Blue |

## Tone Identification

The tone alarm is used for indication of various states of the flight controller, ranging from EKF failure, startup failure, new firmware found, or no sd card found, indicating whether the drone is armed or not, whether the battery is low, is the GPS connected or not, with many more.

The audio files renamed as the state they stand for, can be found on the following [github](https://github.com/sakshamssy/quadcopter_navigation/tree/main/Buzzer%20Sound-Alarm%20Tunes) repository.



## Safety Switch



A safety switch can enable/disable the outputs of motors and servos.

### LED meaning(LED on the Switch itself):

* **Constant blinking** - the system is initialising
* **Intermittent blinking** - the system is ready. Press the safety switch to enable output to the motors.
* **Solid** - safety switch has been pressed, and motors and servos can move once the vehicle is armed.

**Using the Safety Switch to force I/O board firmware update**

If the safety switch is held down for the first few seconds after a Pixhawk is powered up, the I/O firmware is reloaded. This usually is not required, but in some rare cases is required after a firmware upload if you hear the “Start up Failed” sound after startup (see [Sounds wiki page](https://ardupilot.org/copter/docs/common-sounds-pixhawkpx4.html#common-sounds-pixhawkpx4)).

### Buzzer (aka Tone Alarm)



If an active buzzer is used, it can indicate the following:

| **Status** | **Pattern** |
| --- | --- |
| Arming | 3-sec long Beep |
| Arming Failed | Single Beep |
| Disarmed | Single Beep |
| Battery Failsafe | Single Beep repeating every 3 seconds |
| EKF Failure | Beeeep-Beeep-Beep-Beep |
| Lost Vehicle | Beep-Beep repeating every 3 sec |

**Mounting the buzzer:**

The sound from the buzzer can impact the accelerometers if placed too close to the autopilot which can lead to poor altitude hold performance when the buzzer is activated. The buzzer should ideally be mounted at least 5cm away from the autopilot, and the speaker (i.e. the open hole) should not be pointing directly at the autopilot.

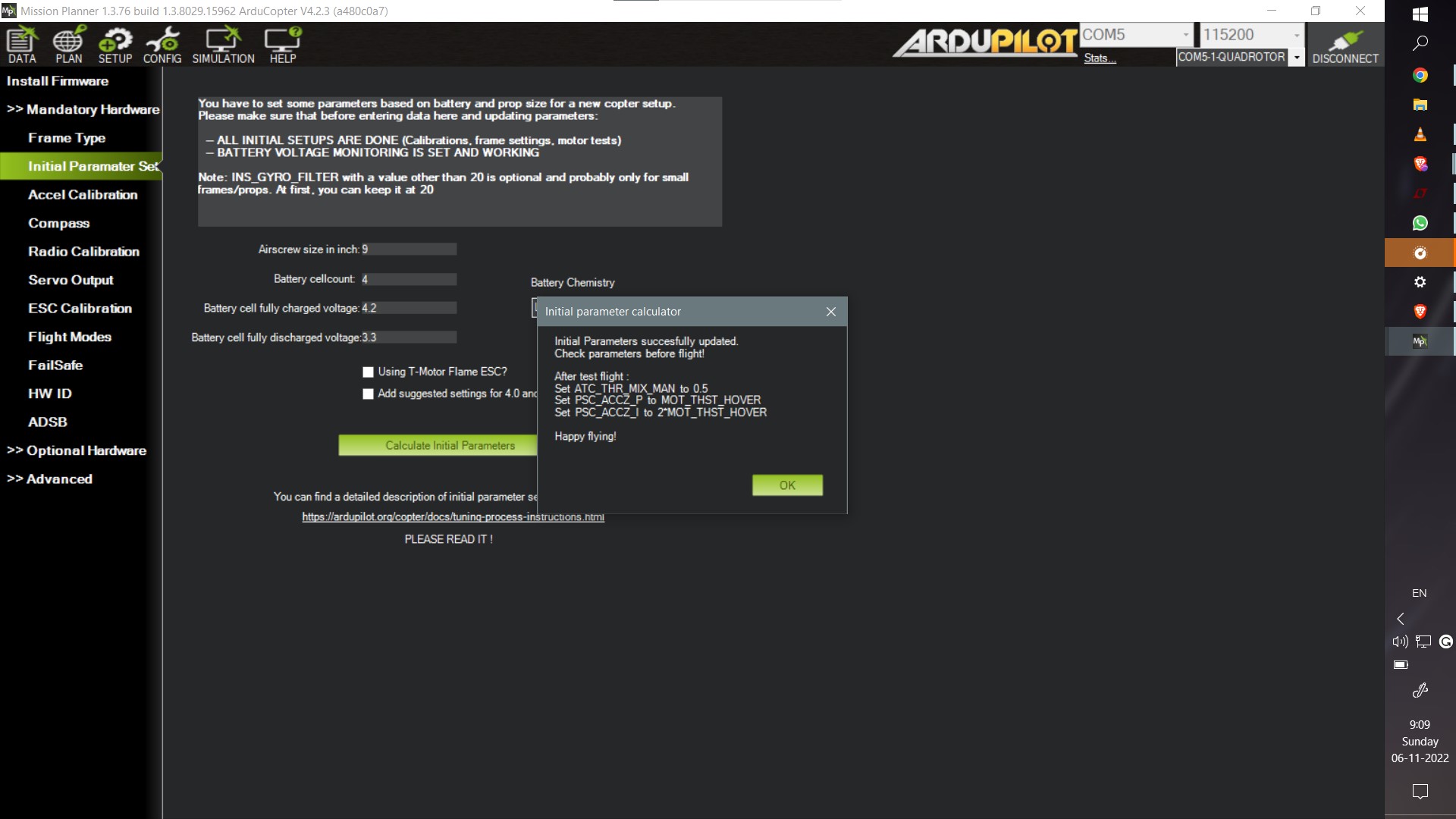
## Calibration and Testing

### Radio Transmitter and Receiver Binding

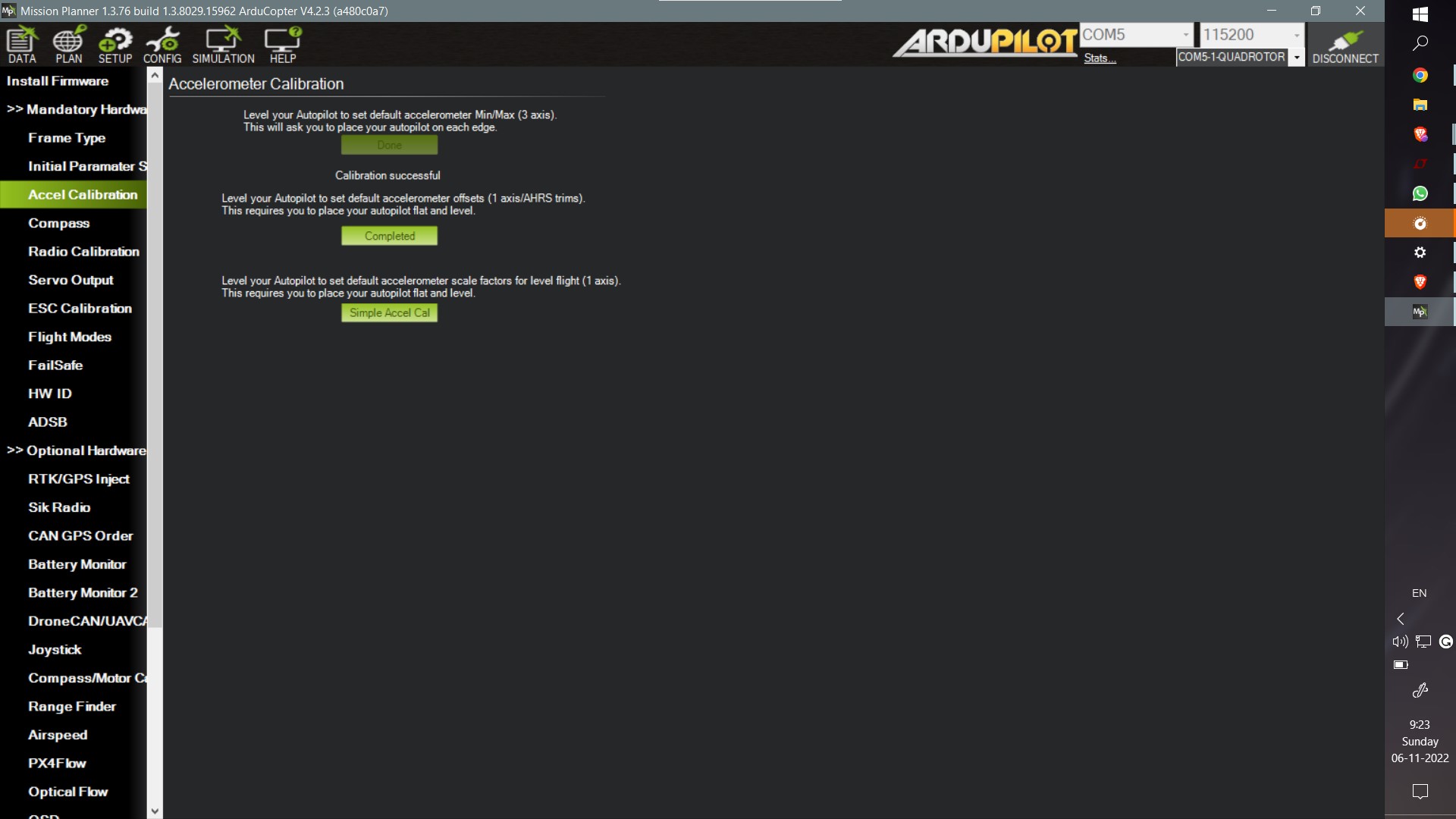
We first need to bind the transmitter and the receiver to ensure that they transmit and receive data from each other. To do this, we connect the first and the third pin of the B/VCC slot of the receiver from the bottom and power the receiver using any other channel. You would notice a blinking red LED signifying it is in binding mode. Switch on the transmitter by pressing the binding key while switching it on. Now on switching on the transmitter with the bind key pressed down, you would notice that the LED stops blinking, signifying that the transmitter and receiver are now bound to each other.

### 

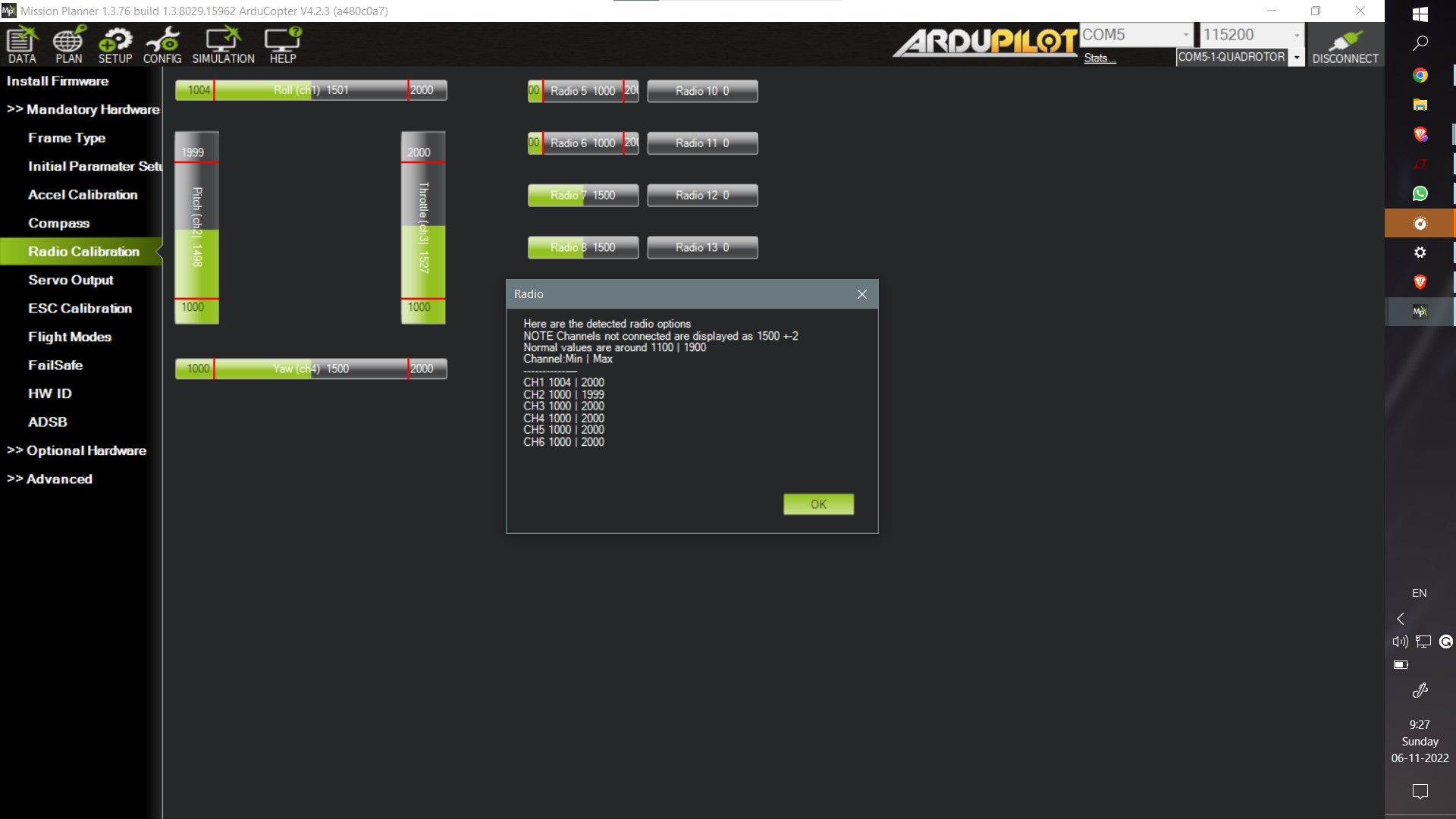
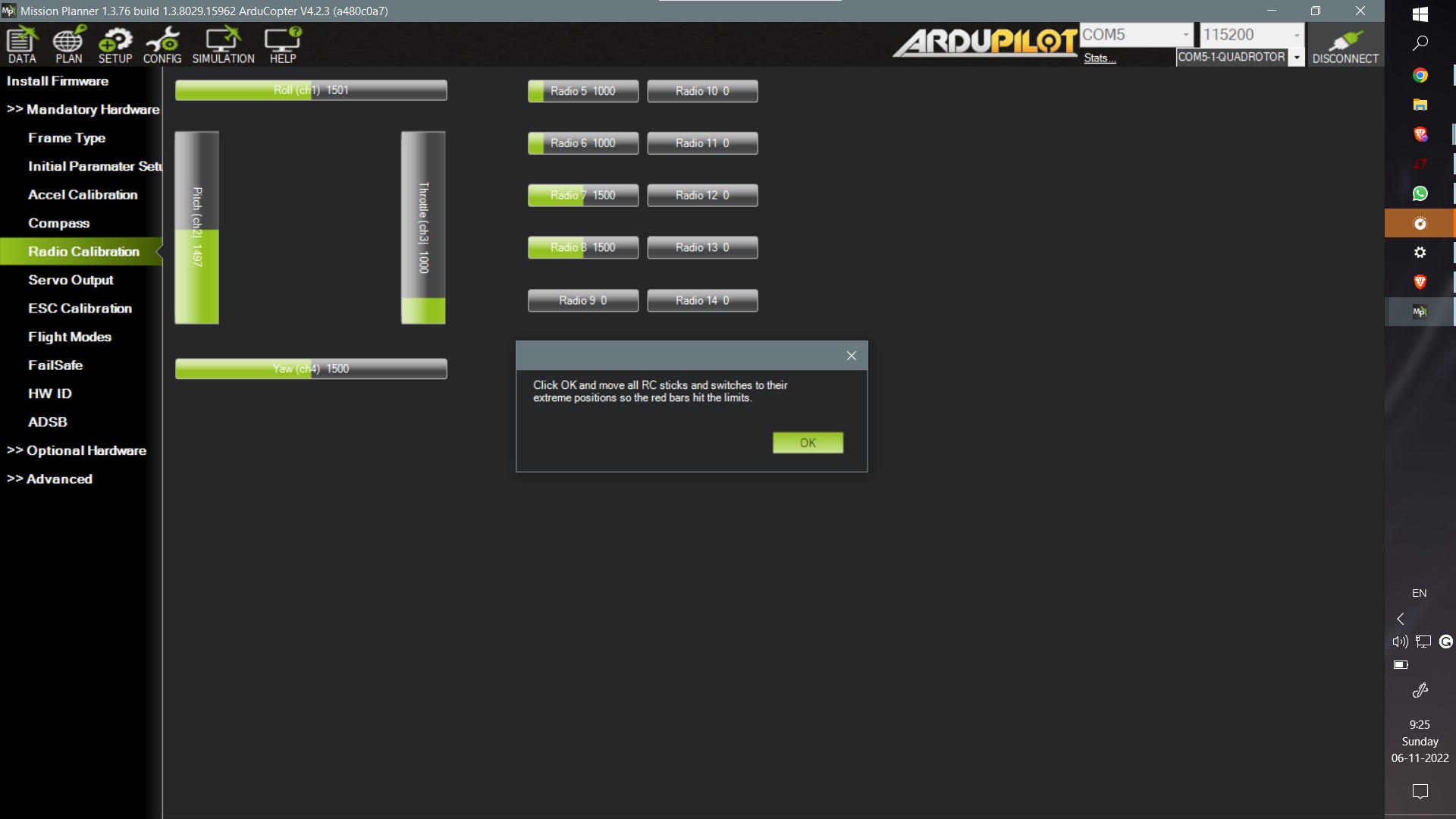
### Initial Parameter SetUp

Now connects your computer's flight controller through a MicroUSB cable to Mission Planner Software using the baud rate of 115200. After connecting, browse to the setup section of the software and choose the frame type as “X”, and change the initial parameters if required(if your battery has three cells and it is shown as four cells here, then make the appropriate change) or let it remain unchanged to its default value.

### Accelerometer and Compass Calibration:

Calibrate your accelerometer and the compass by following the commands provided by the software.

### Radio Calibration

After calibrating the accelerometer and compass, calibrate the radio as follows:

### ESC Calibration:

Similarly follow the steps given by the software for ESC calibration.

### Flight Mode:

You use 4 channel of data transmission for transmitting roll,yaw,pitch and throttle value.Therefore ,you can use the fifth channel of data for selecting the flight mode you want to fly the drone in. These additional assistance in flying is useful for a variety of use cases.For example **Loiter mode**, which holds the drone at a place and keeps it hovering at the same place if the throttle is kept at 50%, can be used for taking still footages.

**Altitude hold** is a similar mode in which we can move the drone in a XY plane with the altitude kept constant when throttle is kept at 50%.**RTL(Return To LaunchPad)** Mode uses the GPS data and returns to the point fom where it was launched.



Following is the list of all the flight modes and their basic description:

**Stabilize Mode**: Stabilize mode allows you to fly your vehicle manually, but self-levels the roll and pitch axis.

**Acro**: Acro mode uses the RC sticks to control the angular velocity of the copter in each axis. Release the sticks and the vehicle will maintain its current attitude and will not return to level (attitude hold). Acro mode is useful for aerobatics such as flips or rolls, or FPV when smooth and fast control is desired.

**Alt Hold**: In altitude hold mode, Copter maintains a consistent altitude while allowing roll, pitch, and yaw to be controlled normally. When altitude hold mode (aka AltHold) is selected, the throttle is automatically controlled to maintain the current altitude. Roll, Pitch and yaw operate the same as in [Stabilize mode](https://ardupilot.org/copter/docs/stabilize-mode.html#stabilize-mode) meaning that the pilot directly controls the roll and pitch lean angles and the heading.

**Guided**:Guided mode is a capability of Copter to dynamically guide the copter to a target location wirelessly using a telemetry radio module and ground station application.

**Loiter**: Loiter Mode automatically attempts to maintain the current location, heading and altitude. The pilot may fly the copter in Loiter mode as if it were in a more manual flight mode but when the sticks are released, the vehicle will slow to a stop and hold position.

**RTL**:RTL mode (Return To Launch mode) navigates Copter from its current position to hover above the home position.

**Circle**: Circle will orbit a point located [CIRCLE\_RADIUS](https://ardupilot.org/copter/docs/parameters.html#circle-radius) centimeters in front of the vehicle with the nose of the vehicle pointed at the center.

**Land**:LAND Mode attempts to bring the copter straight down

**Drift**: Drift Mode allows the user to fly a multi-copter as if it were a plane with built in automatic coordinated turns.The user has direct control of Yaw and Pitch, but Roll is controlled by the autopilot.

**Sport**:It was designed to be useful for flying FPV and filming [dolly shots](https://en.wikipedia.org/wiki/Dolly_shot) or fly bys because you can set the vehicle at a particular angle and it will maintain that angle.The pilot’s roll, pitch and yaw sticks control the rate of rotation of the vehicle so when the sticks are released the vehicle will remain in its current attitude.

**Flip**:Vehicle will flip on its roll or pitch axis depending upon the pilot’s roll and pitch stick position. Vehicle will rise for 1 second and then rapidly flip. The vehicle will not flip again until the switch is brought low and back to high

**AutoTune**: AutoTune attempts to automatically tune the Stabilize P, Rate P and D, and maximum rotational accelerations to provide the highest response without significant overshoot

**PosHold**: The PosHold flight mode is similar to Loiter in that the vehicle maintains a constant location, heading, and altitude but is generally more popular because the pilot stick inputs directly control the vehicle’s lean angle providing a more “natural” feel.

**Brake**:This very simple flight mode simply stops the vehicle as soon as possible using the Loiter controller. Once invoked, this mode does not accept any input from the pilot. This mode requires GPS.

**Throw**: This slightly dangerous flight mode allows the pilot to throw the vehicle into the air (or drop the vehicle) in order to start the motors. Once in the air, this mode does not accept any input from the pilot. This mode requires GPS.

**Guided\_NoGPS**:This variation of Guided mode does not require a GPS but it only accepts [attitude targets](https://mavlink.io/en/messages/common.html#SET_ATTITUDE_TARGET). Because it does not accept position or velocity targets like regular Guided mode it is generally not useful for regular users. This mode was created for use by companion computers that may want to fly the vehicle as if it was in AltHold mode.

**Smart\_RTL**: When switched into Smart RTL, like regular RTL, the vehicle will attempt to return home. The “Smart” part of this mode is that it will retrace a safe path home instead of returning directly home. This can be useful if there are obstacles between the vehicle and the home position.

**FlowHold**: FlowHold mode uses an [optical flow sensor](https://ardupilot.org/copter/docs/common-optical-flow-sensors-landingpage.html#common-optical-flow-sensors-landingpage) to hold position without the need for a GPS nor a downward facing Lidar. Better performance can be achieved by attaching a [rangefinder](https://ardupilot.org/copter/docs/common-rangefinder-landingpage.html#common-rangefinder-landingpage) and using regular [Loiter](https://ardupilot.org/copter/docs/loiter-mode.html#loiter-mode) mode instead

**Follow**: When switched into Follow, the vehicle will attempt to follow another vehicle (or anything publishing its position) at a specified offset. The vehicle lead vehicle’s position must be published to the vehicle in Follow mode using a telemetry system.

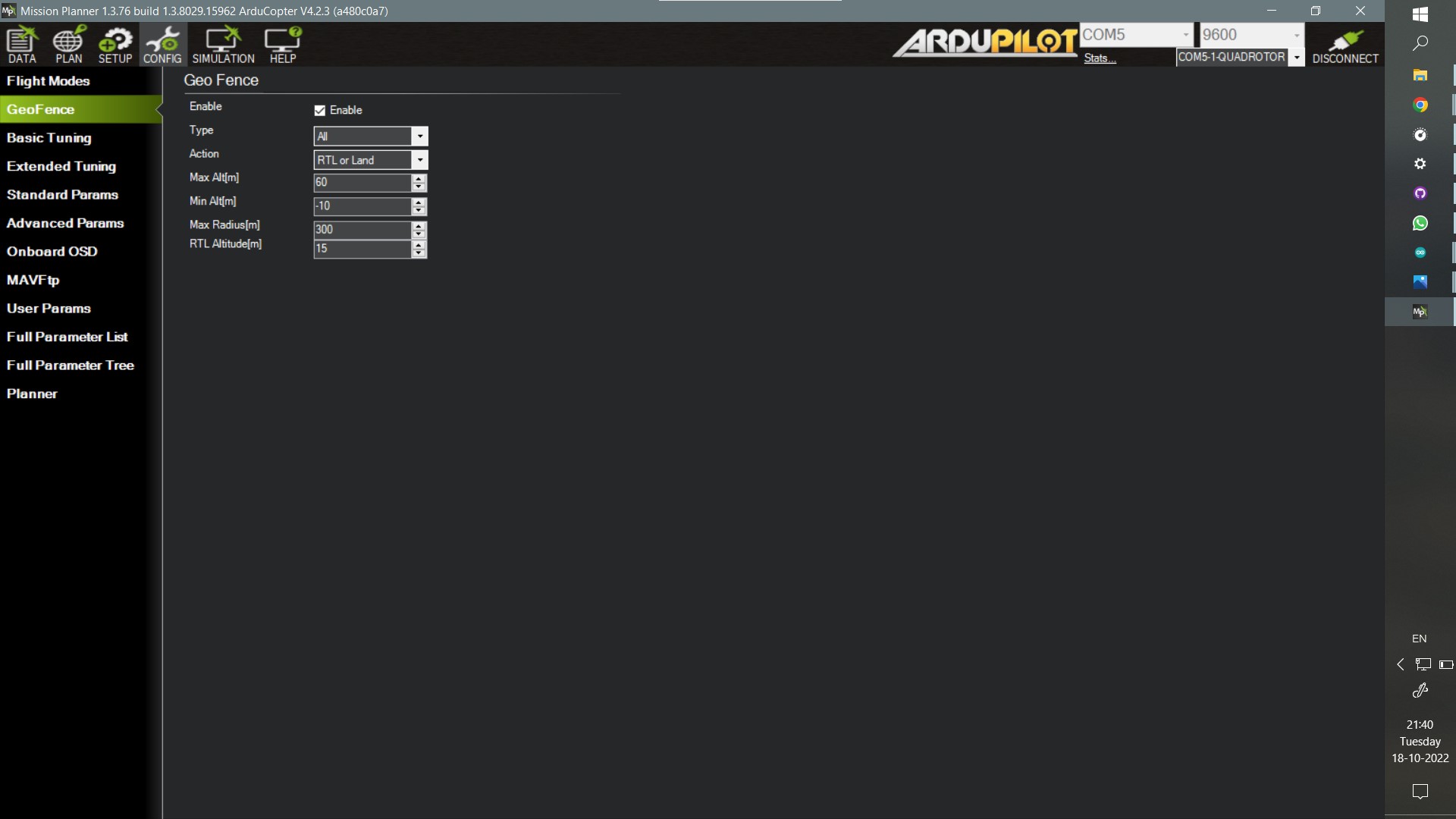
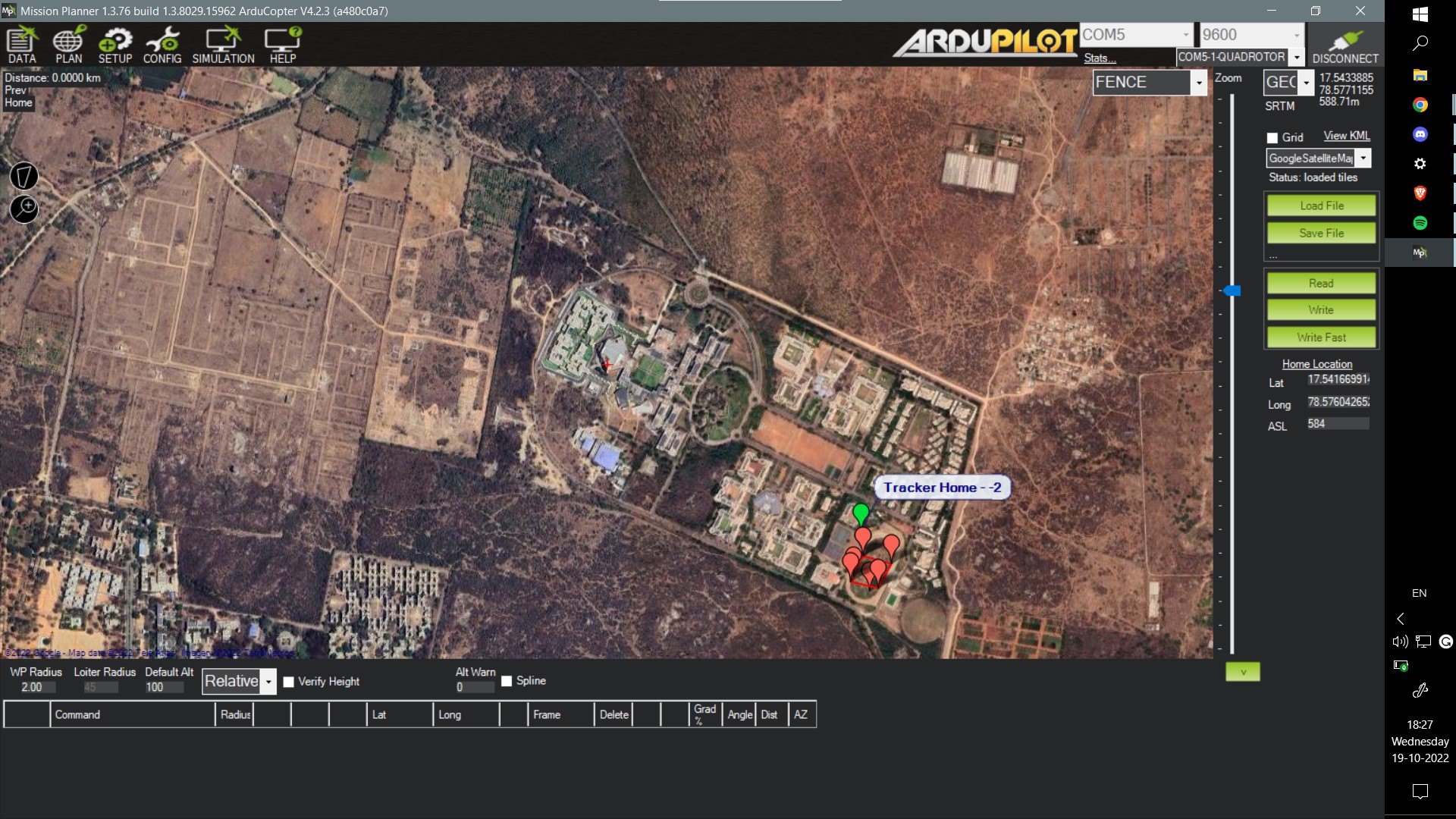
**Zigzag**: ZigZag mode is a semi-autonomous mode designed to make it easier for a pilot to fly a vehicle back and forth across a field which can be useful for crop spraying

**System\_ID**:This mode is for advanced users and provides a means to generate mathematical models of the vehicles flight behavior for model generation. It places the vehicle into a STABILIZE like mode, and generates bursts of signals (“chirps”) injected into the control loops at various points and logs the results for math analysis and model generation later.

### FailSafe

Failsafe mode is a precautionary step that ensures safekeeping of the quadcopter from system failures such as low battery,transmission signal loss etc. So along with the geofence, the fail safe mechanism can make the drone either land or return to the launchpad(as set by parameters), if the battery is below certain voltage or the drone has flown away from the designated geofence.

### GeoFence

Geofence is like a virtual perimeter for the real geographic area which we can assign the drone to be confined in or out of.In the Mission Planner Software, you can create polygons to mark area as inclusion area where the drone is supposed to fly or exclusion area where the drone is not supposed to delve into. 

## 

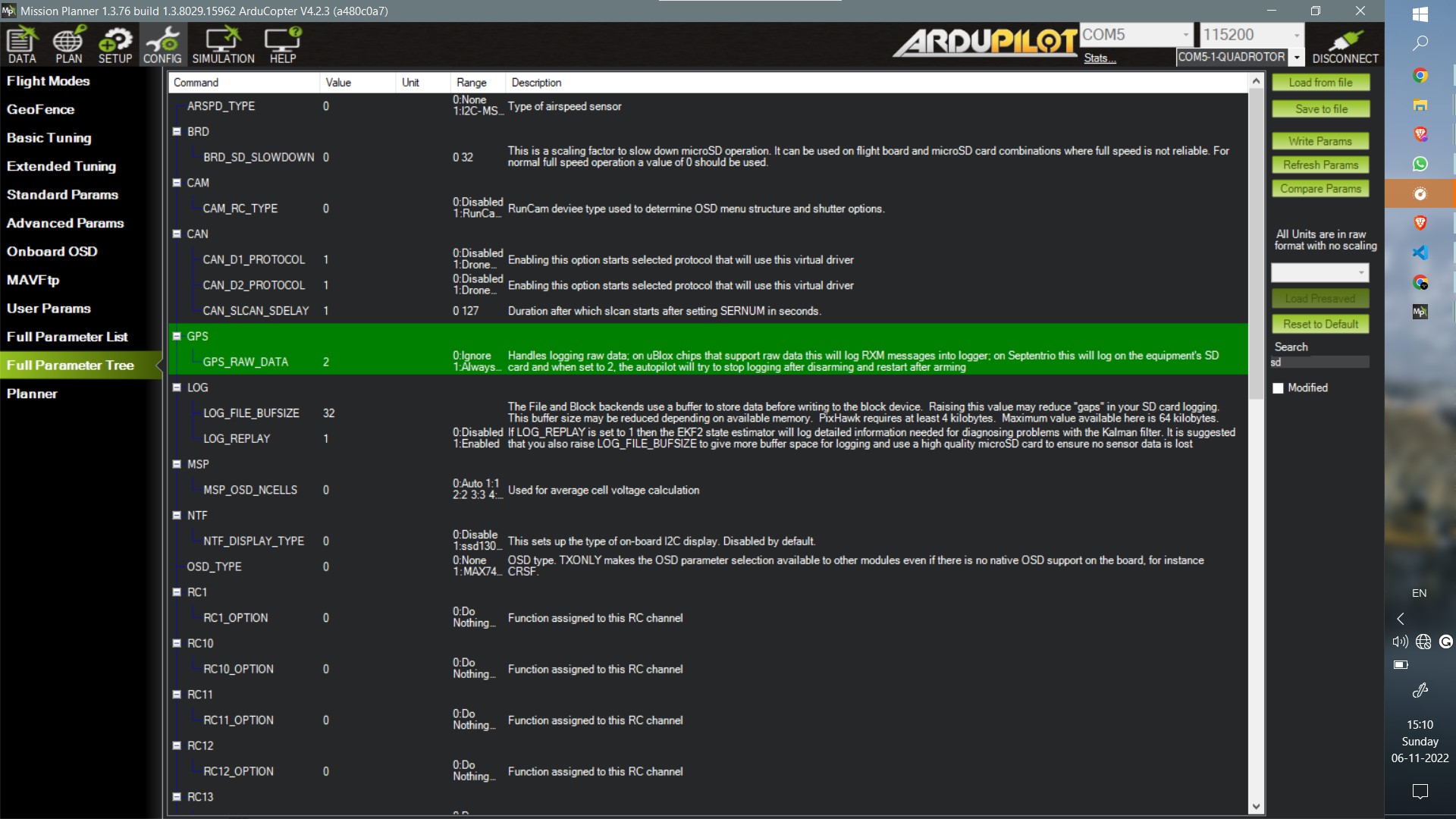
## 

## 

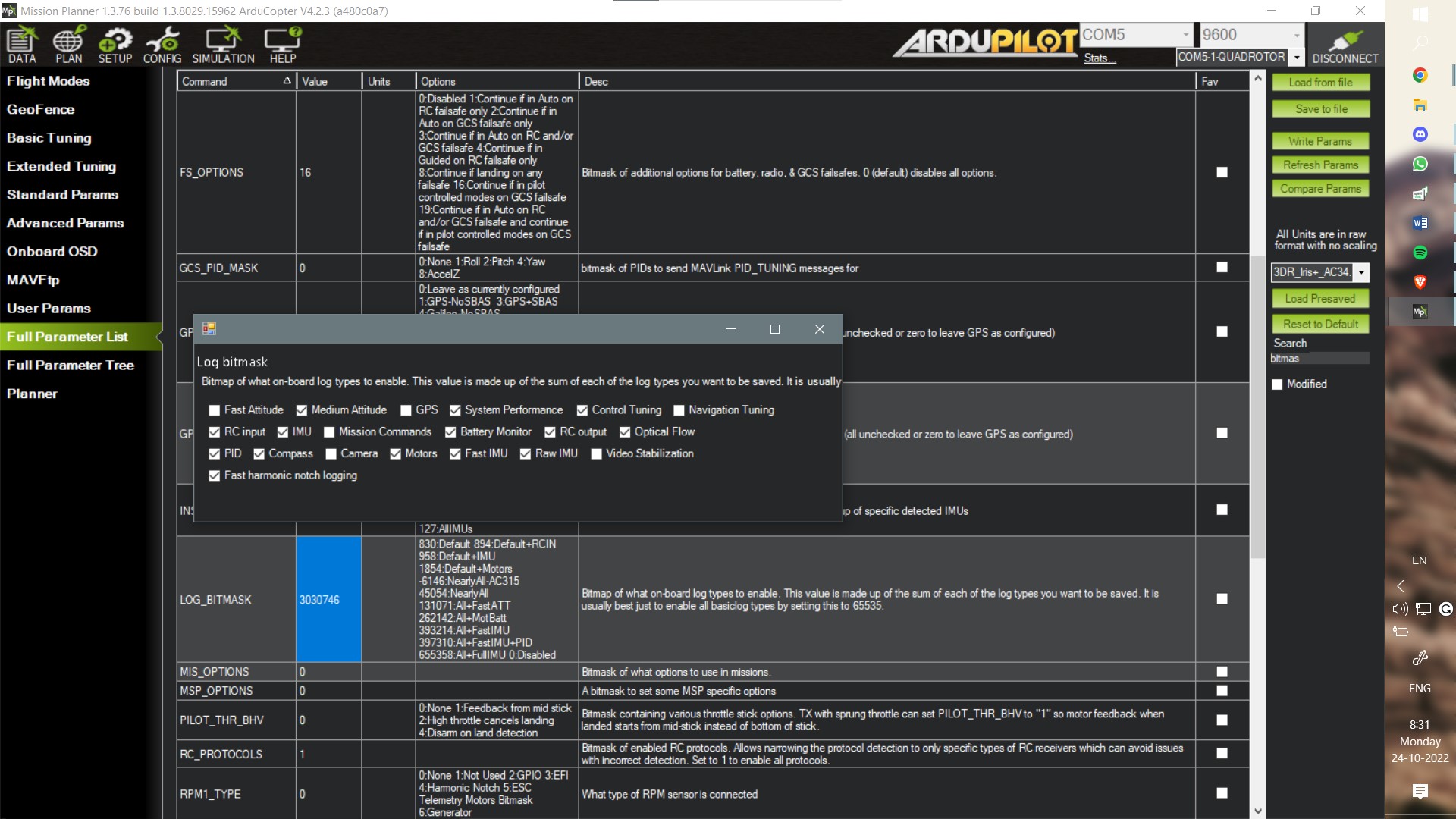
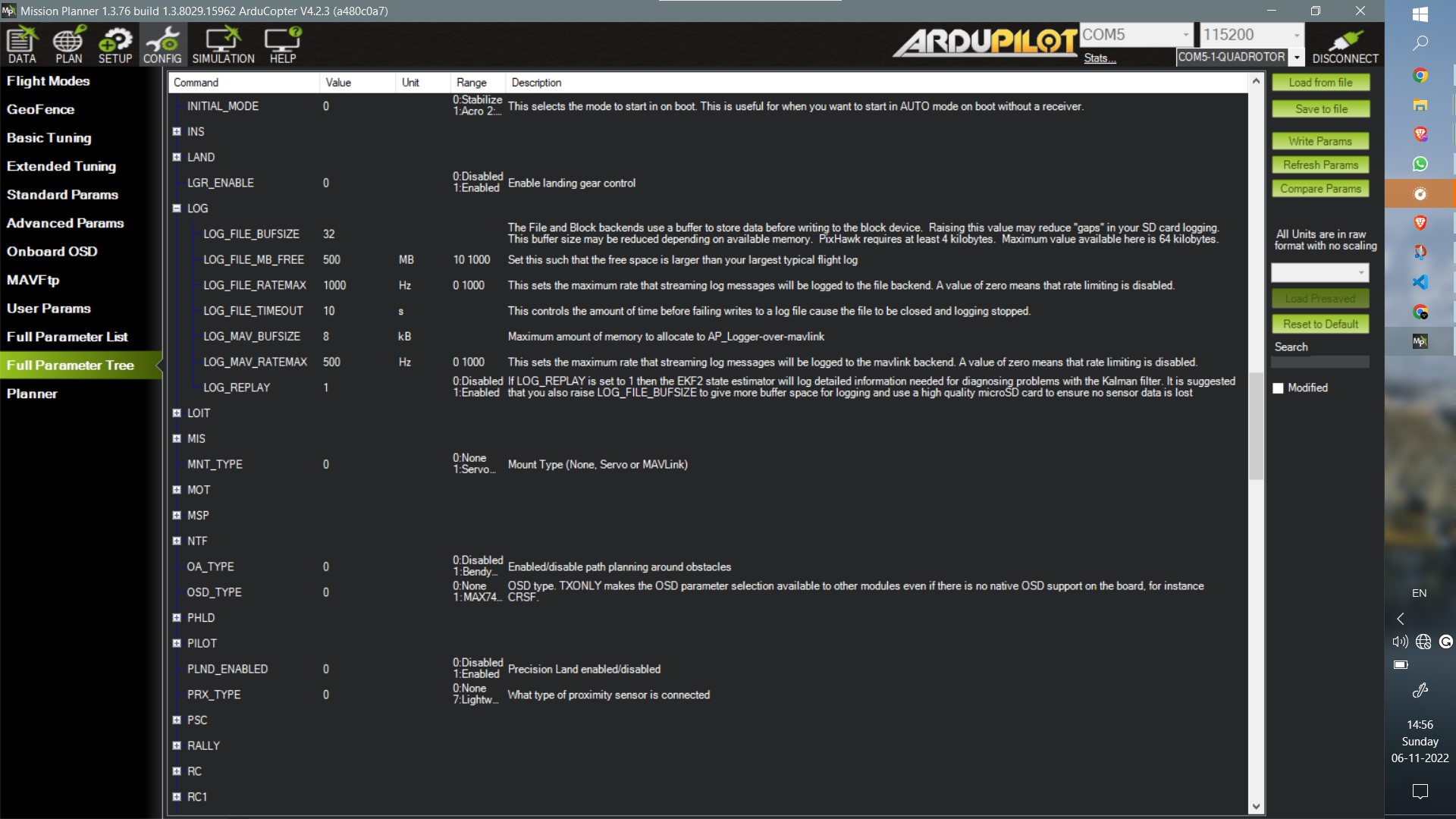
## 

## 

## Parameter Setup

There exist a number of parameters that control each and every parameter of the flying drone, from roll, pitch,yaw adjustments to whether or not we want safety checks before flying or do we want telemetry flight logs or not. We customize the working of the drone whichever way we want by changing these parameters.The whole list of parameters can be found in the config menu in a descriptive way in order to help us choose correct values of parameters we want to put in.

### Logging

The after flight log data comprising of the on flight data of accelerometer, compass, GPS, telemetry commands and many other sensor data can be of vital for after flight analysis for optimization.This data can be collected and stored in the sd card in pixhawk by changing a few parameters as shown in the picture.The exact parameters list used can be found on the github repository.

### ArmCheck

These are set of preflight precautionary checks in order to avoid malfunctioning of any system during the course of flight.

### 

### Raspberry Pi Communication

Set up the following parameters as follows:

* SERIAL2\_PROTOCOL ` = 1 (the default) to enable MAVLink on the serial port.
* SERIAL2\_BAUD ` = 921for the Pixhawk can communicate with the Raspberry Pi at 921600 baud.
* `LOG\_BACKEND\_TYPE ` = 3 if you use APSync to stream the data flash log files to the Raspberry Pi.

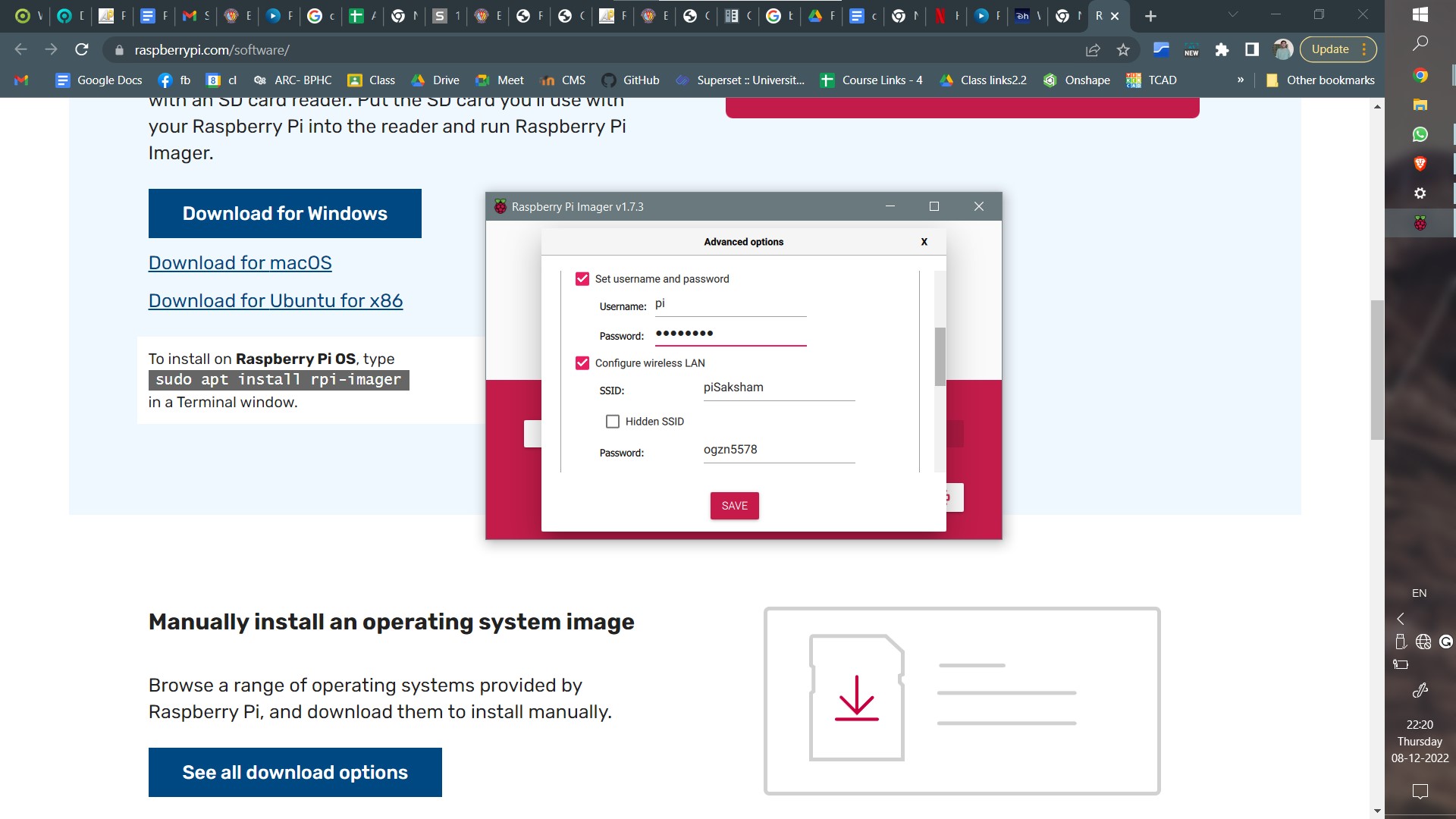
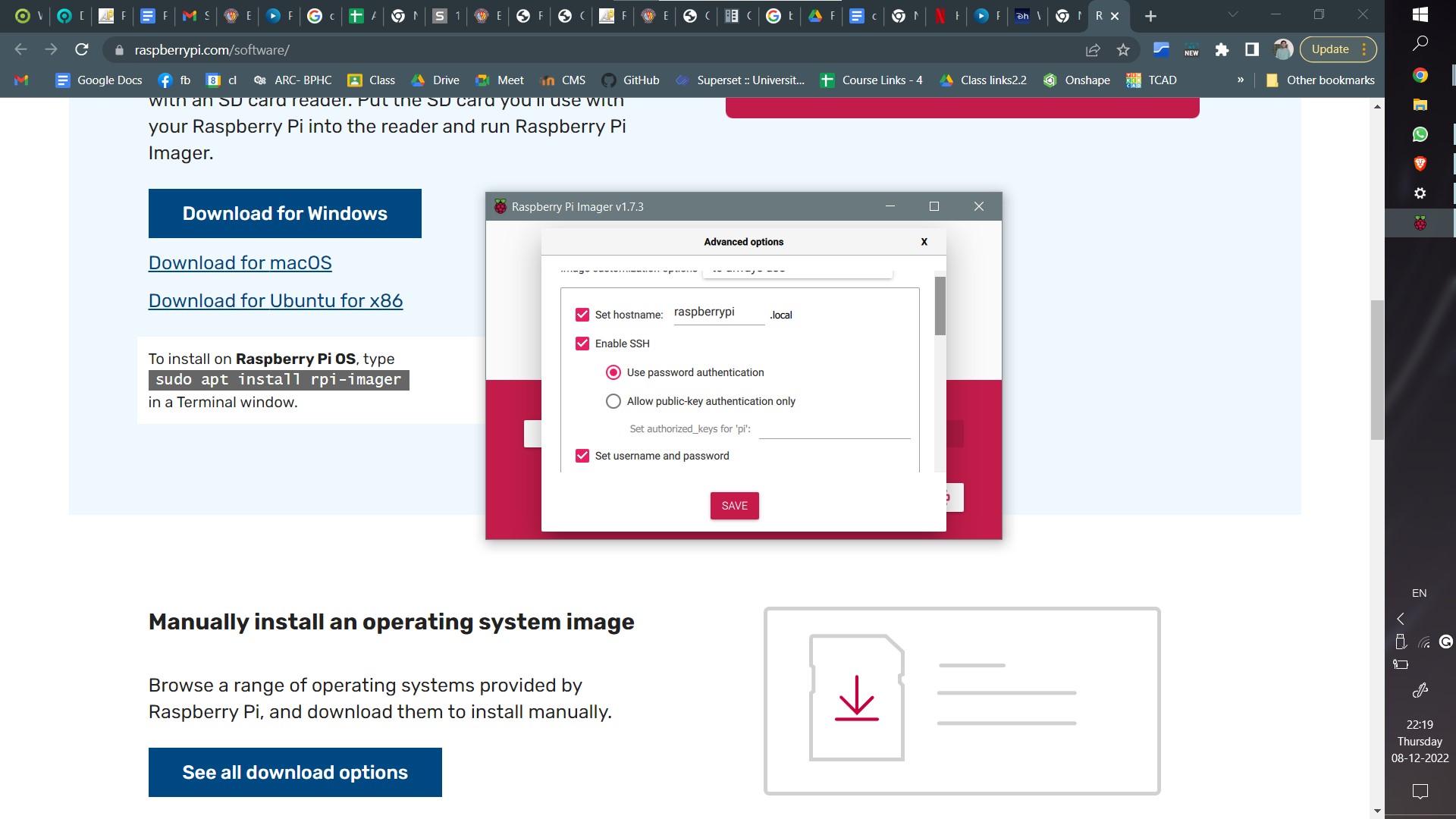
## 

## Raspberry Pi 3 As A Companion Computer For Pixhawk

### Installing OS:

The Raspberry Pi used as a companion computer needs an operating system. The operating system we are using is the Raspberry Pi OS(previously known as Raspbian), which is the official supported operating system.

We are using the Raspberry Pi Imager to install the operating system to the microSD card. After installing the Imager, insert the microSD card into the computer and open the Imager. Choose the Raspberry Pi OS as the operating system, set the hostname, and the password for authentication and the SSID and its password. We are using the mobile hotspot, the common hotspot to which the computer we use for SSH and the Raspberry are connected. So, set the SSID, and the password asked here is the same as your mobile hotspot.



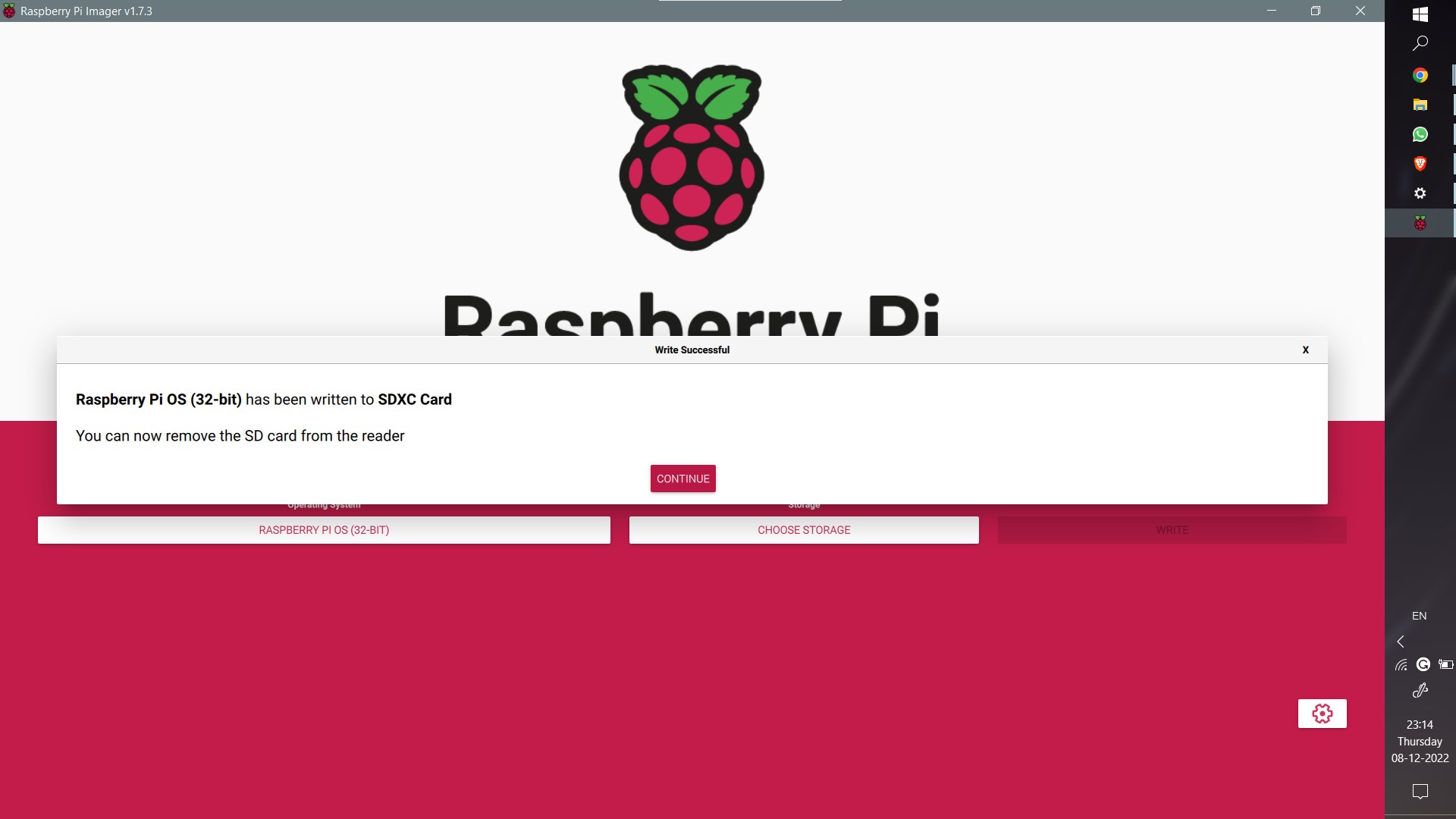
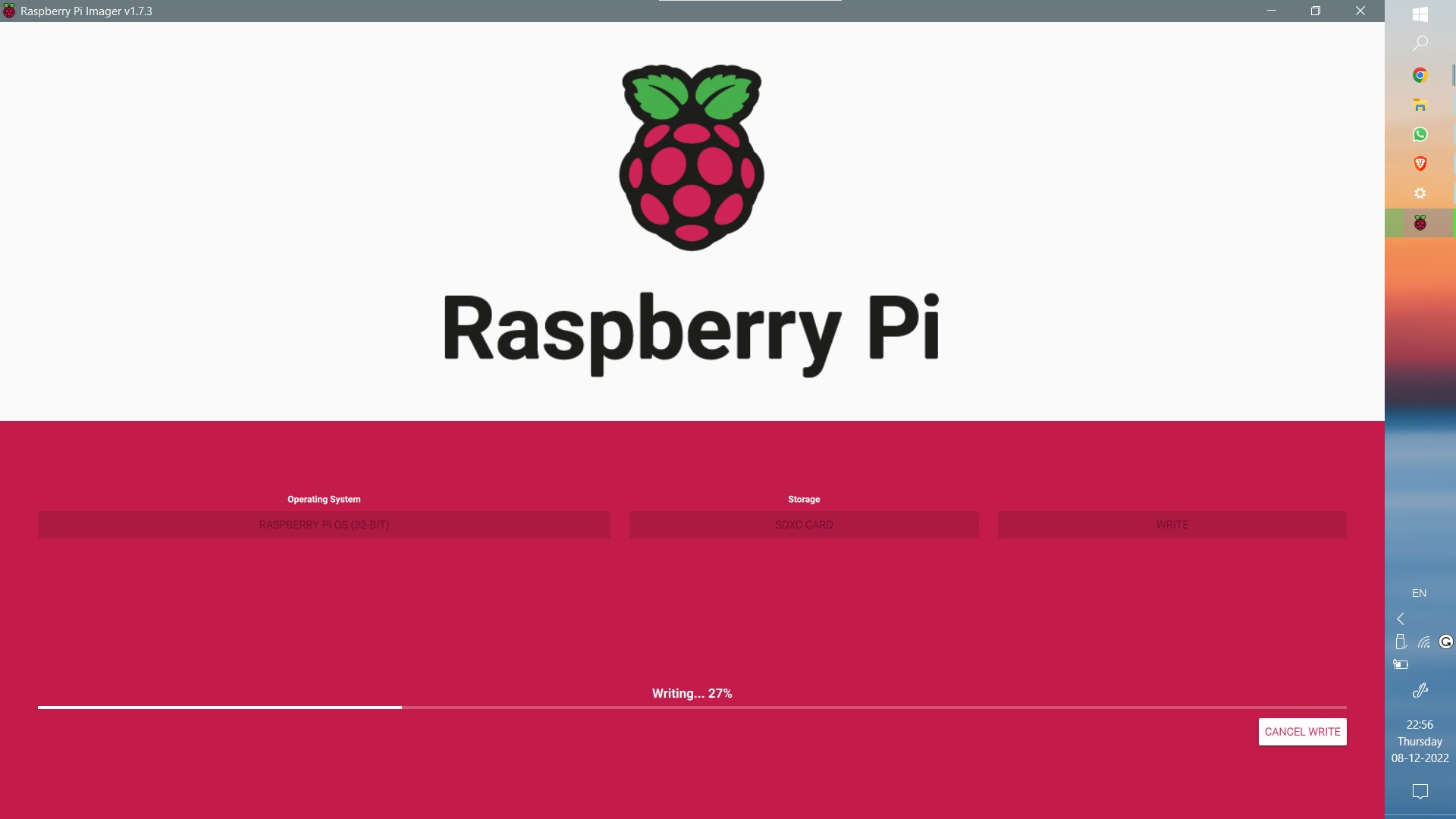
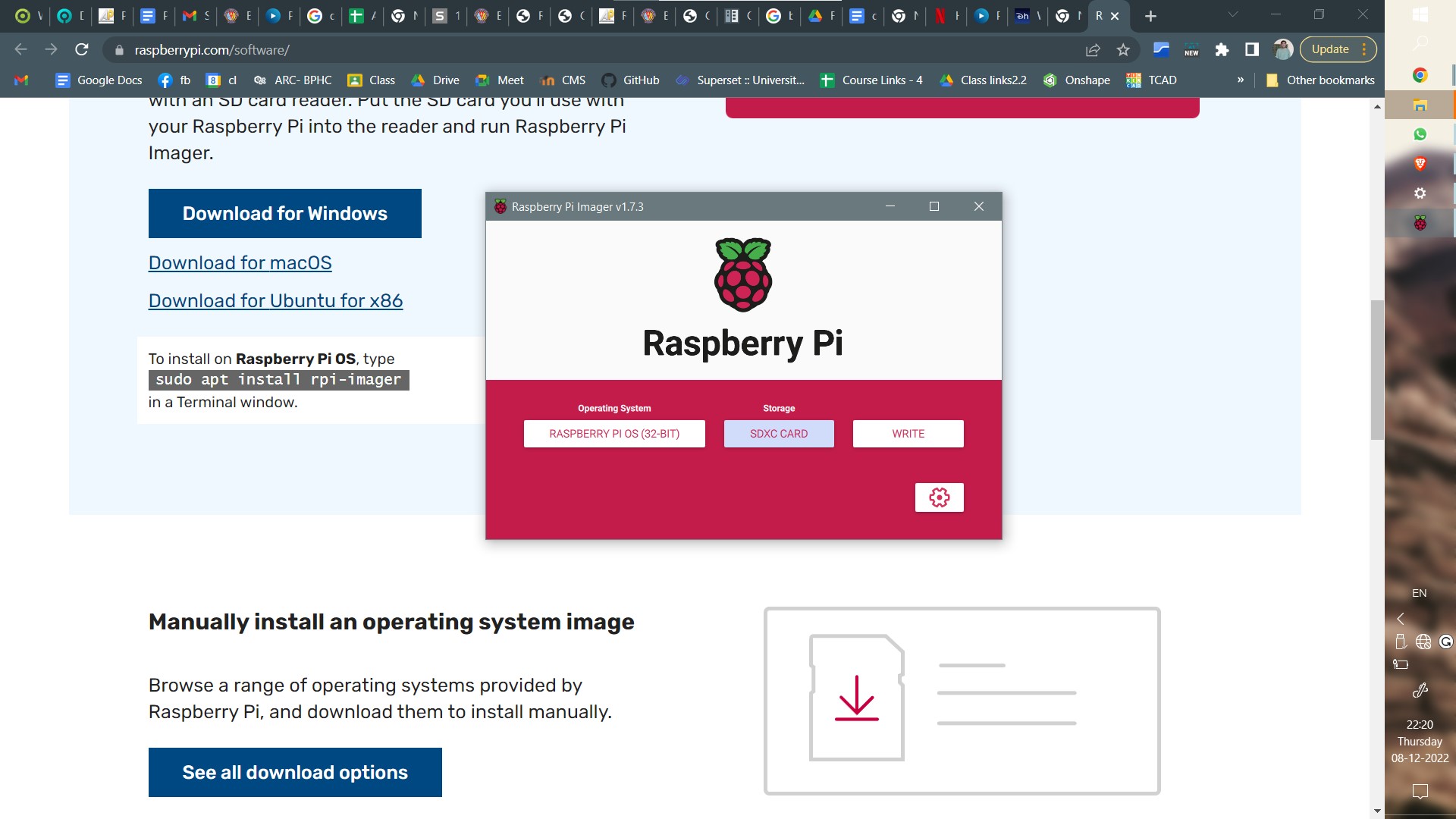
The credentials used for this Raspberry pi are:

Hostname: **raspberrypi.local**

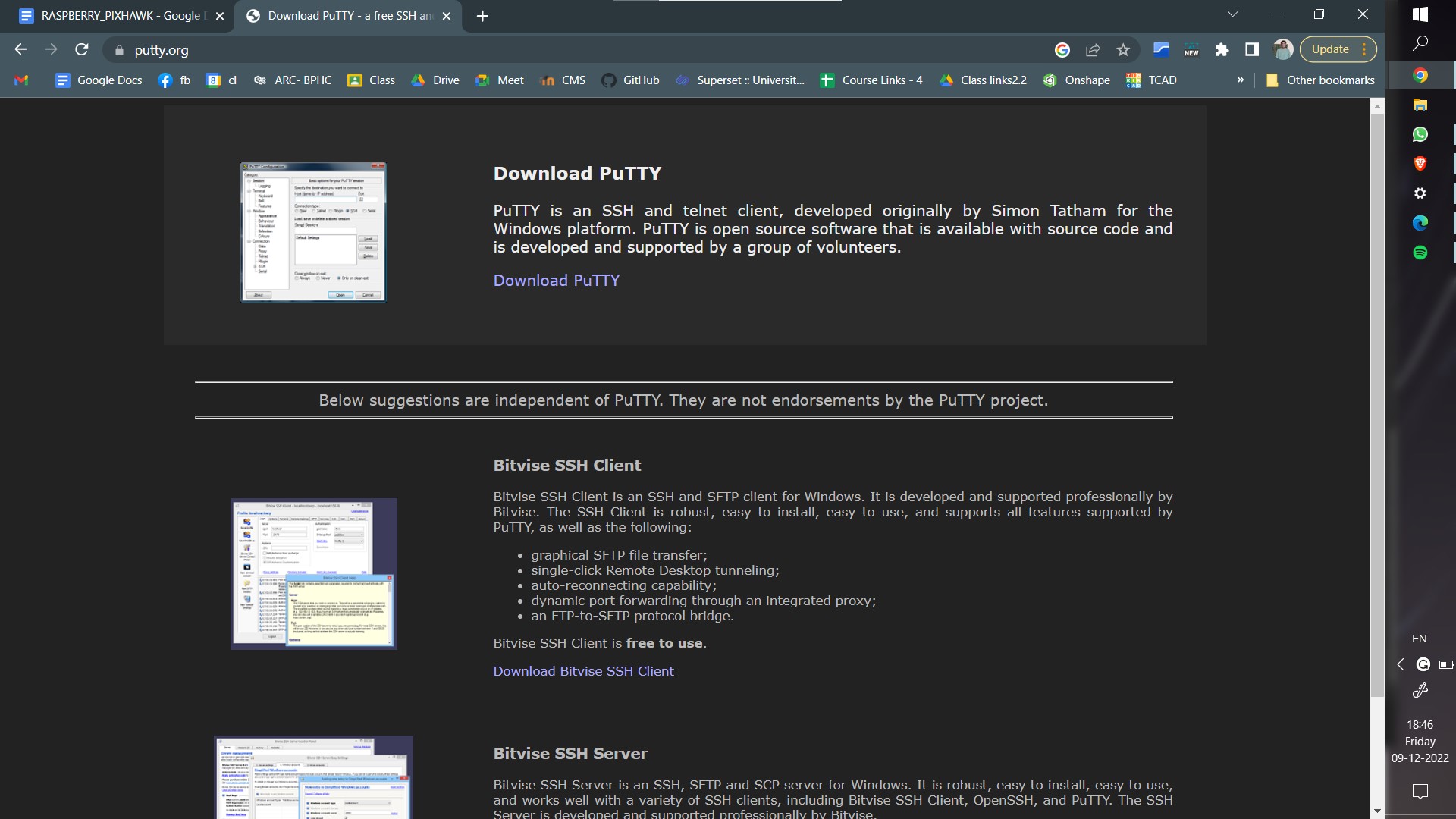
Login as: **pi**

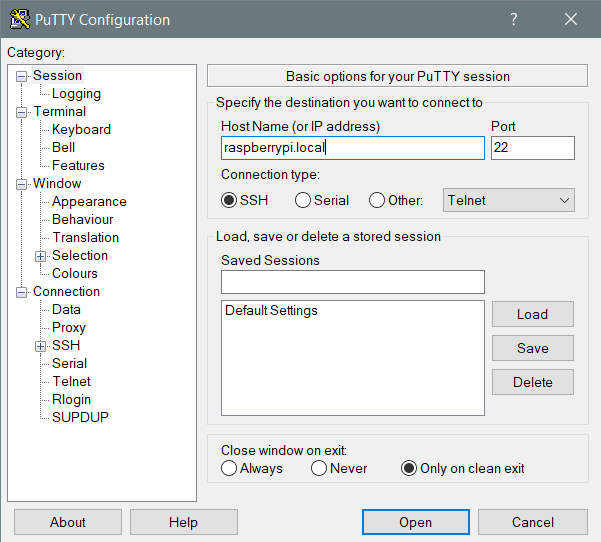
Password: **bitsmech**

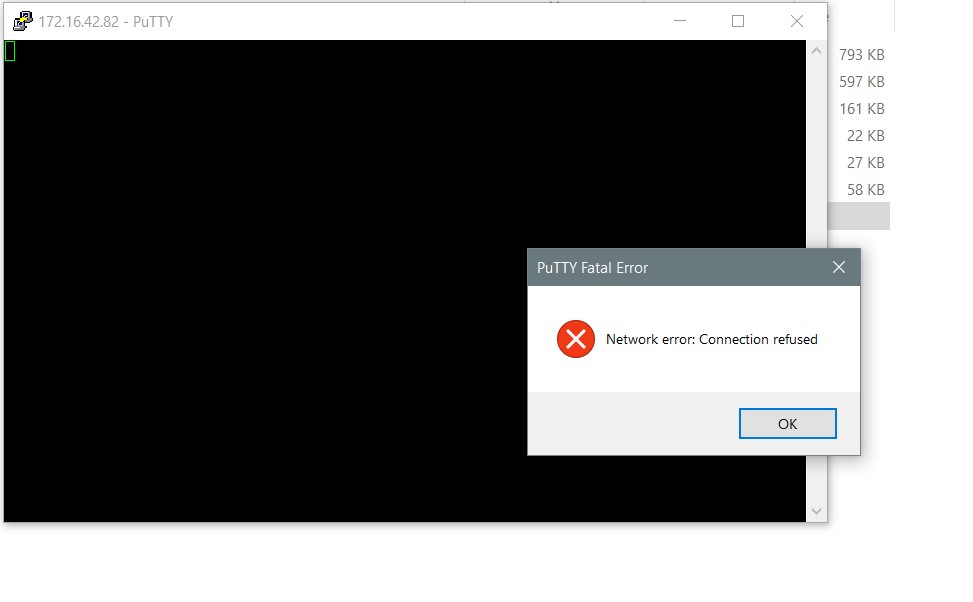
SSID: **piSaksham**

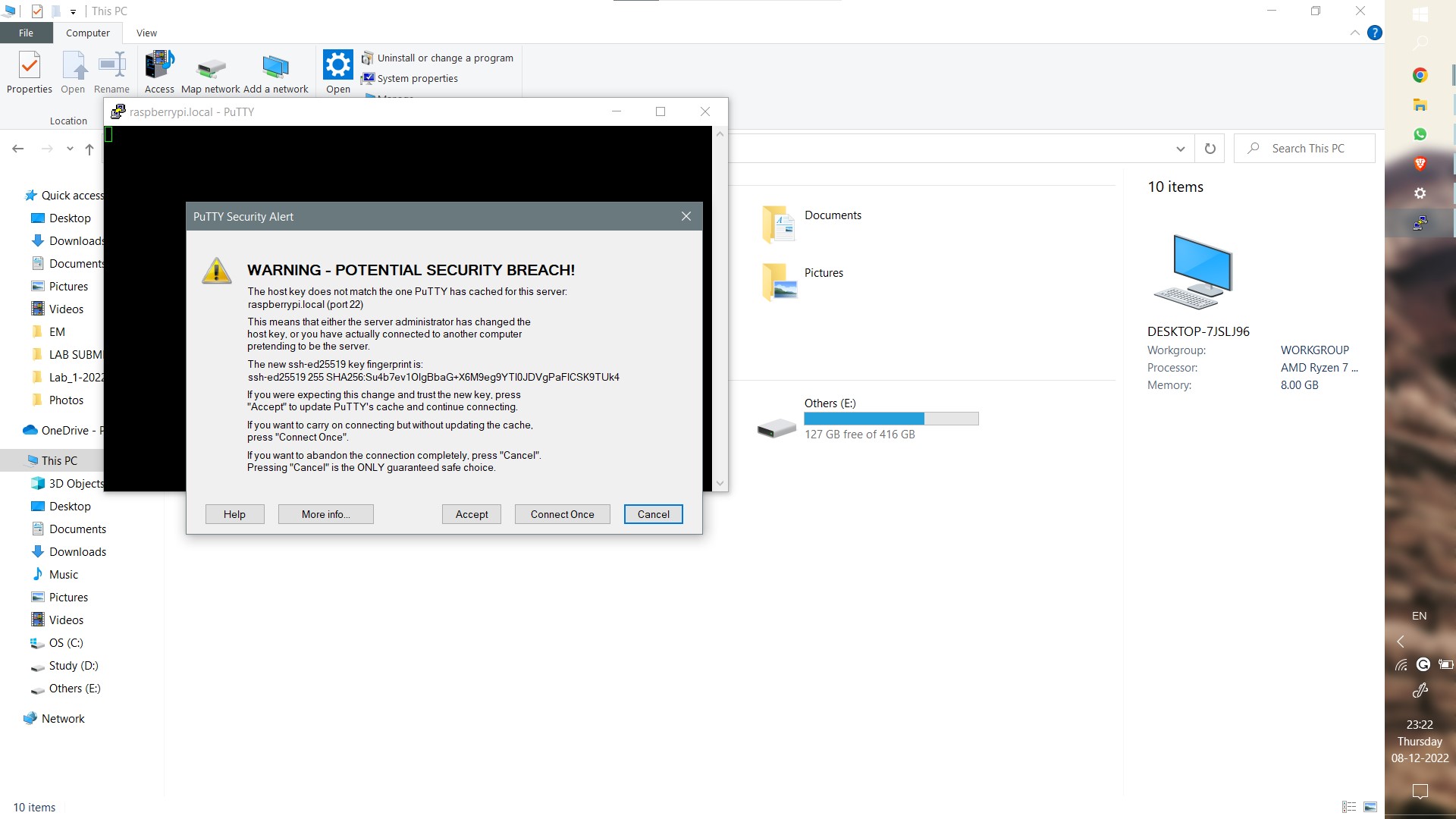
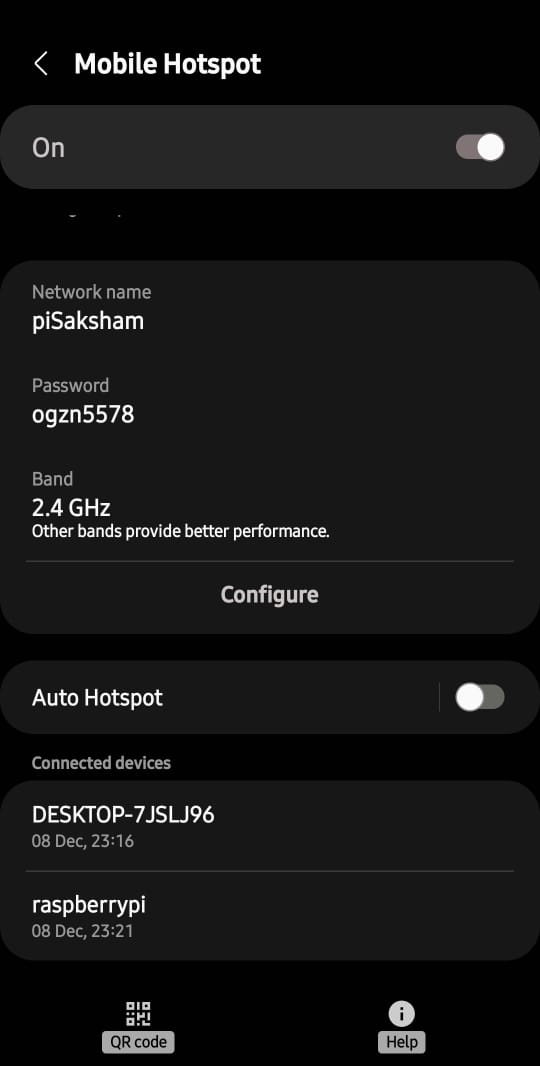
Password: **ogzn5578**After setting up these credentials, select the microSD card to which the OS will be installed and then select the write option. The writing process might take a few minutes. After the writing process, remove the microSD card from the computer and insert it into the Raspberry Pi.

### SSH using PuTTY:

We will now use PuTTY for SSH to establish communication between the Raspberry Pi and the computer. Download PuTTY, an SSH and telnet client for the Windows platform. The PuTTY alternatives for Mac include OpenSSH, mRemoteNG, and MobaXterm, majorly terminal emulators.



Open PuTTY and enter your hostname, raspberrypi.local, in our case. The following error might appear that PuTTY found no connection with the particular hostname. If one encounters this error, one must verify that both the computer and the Raspberry Pi are connected to the same network hotspot with the same SSID and password as mentioned during the installation of the OS. The Raspberry Pi automatically connects to the network with the proper SSID and password within 30-45 seconds.



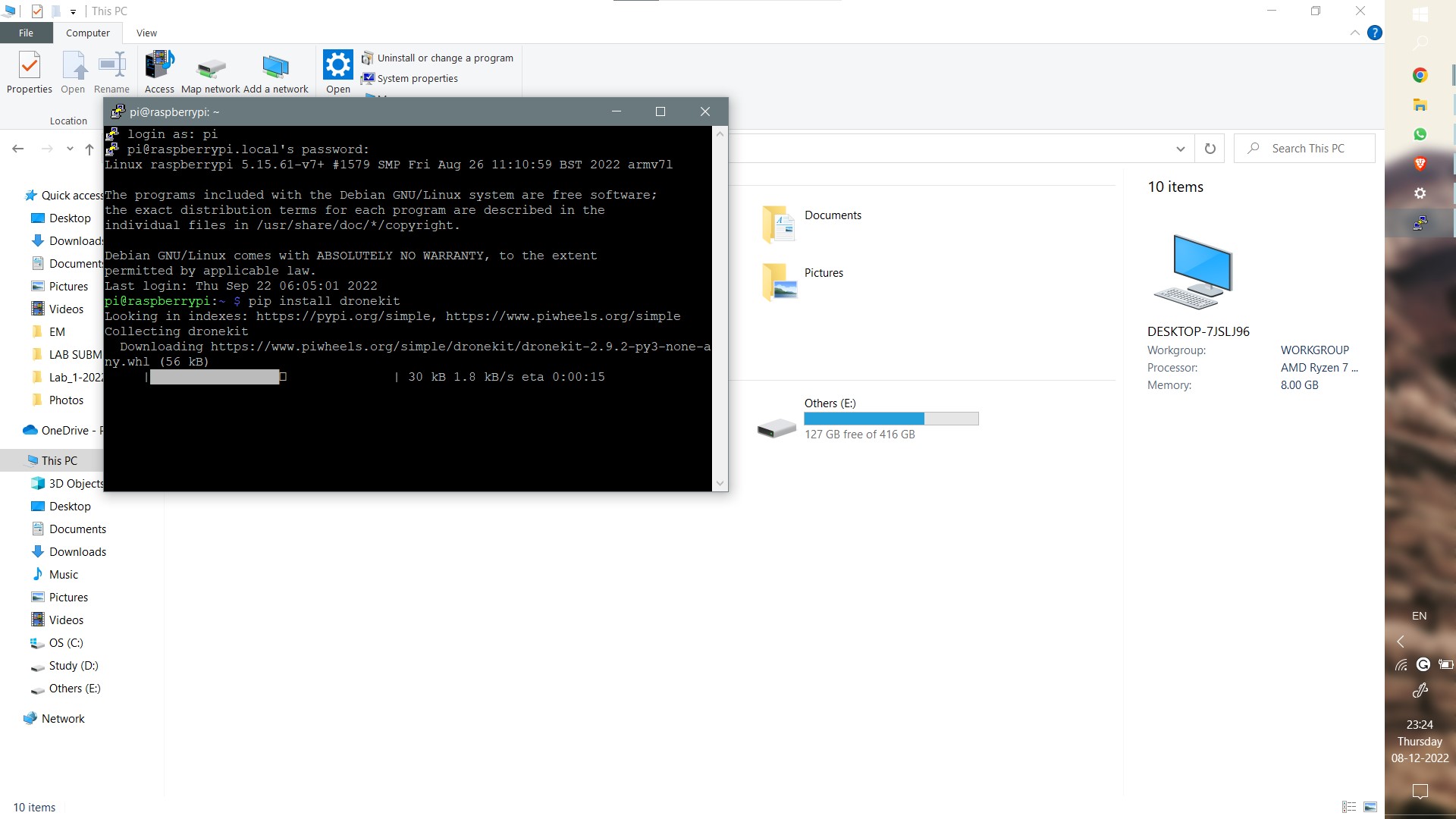
### Connections between Pixhawk and Raspberry Pi:

Connect the flight controller’s TELEM2 port to the RPi’s Ground, TX and RX pins as shown in the image above.The RPi can be powered by connecting the +5V source to the +5V pin or from USB in.

### Installing Libraries:

After successfully being able to connect to Raspberry Pi through SSH,we will install all the libraries required.The libraries we will be installing, will aid us

| sudo apt-get update  sudo pip install future  sudo pip install pymavlink  sudo pip install mavproxy  sudo pip install dronekit |
| --- |



### Testing the connection

## To test the Raspberry Pi and Pixhawk are able to communicate with each other first ensure the Raspberry Pi and Pixhawk are powered, then in a console on the Raspberry Pi type:

| sudo -s  mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --aircraft MyCopter |
| --- |

## 

## On some versions of Raspberry Pi ,the uart serial connection may be disable by default. To rectify it,and to enable serial connection on the Raspberry Pi, edit /boot/config.txt and set enable\_uart=1.The build-in serial port is /dev/ttyS0.

After this, explore the dronekit documentation to work on Raspberry Pi and Pixhawk based drone.

The camera module to be added can be done with help from Raspberry pi’s manual,which is given in the links.

## Conclusion

The work initiated on this drone this semester is the first step towards a more and more complex application-based drone project. This project report is intended to be used as a handout for anyone who takes up any work on drones involving Pixhawk, Raspberry Pi, or similar projects, for that matter of fact.

The project was intended towards planning the trajectory of the drone, which can be done through Mission Planner and let the drone fly autonomously by using the appropriate flight modes. The other major objective was to acquire telemetry data, which is now acquired from Pixhawk and stored in the microSD card and can be copied from the microSD card itself or downloaded through Mission Planner. The other way shown in the report is the automatic saving of the log file on Raspberry Pi.

Further works on this drone can include flying the drone in Autotune mode to tune its PID values and other parameters and then adding an FPV camera and storing the video footage on the Raspberry Pi itself, which can be done by inferring to the Raspberry Pi handout.

## 

## Bibliography

* [F450 Frame](https://robu.in/product/q450-quadcopter-framepcb-version-with-integrated-pcb-plastic-landing-gear-combo-kit-2/?gclid=CjwKCAjwtp2bBhAGEiwAOZZTuPNvjoUGYYbtOJGQNqHWazTdY3pi9mIC1HxRZmbPHcXwtGxjorTOohoCpR8QAvD_BwE)
* [GPS Unit](https://robu.in/product/ublox-neo-7m-gps-compass/?gclid=CjwKCAjwtp2bBhAGEiwAOZZTuDt-fR5tCmkfJER-Cx4NJr68WiY3Lm5CrC7bUD2fJmbm1zXCGnWPDhoCAeIQAvD_BwE)
* [Pixhawk](https://robu.in/product/pixhawk-radiolink/)
* robu.in
* [LEDs Meaning — Copter documentation - ArduPilot](https://ardupilot.org/copter/docs/common-leds-pixhawk.html)
* [Safety Switch — Copter documentation - ArduPilot](https://ardupilot.org/copter/docs/common-safety-switch-pixhawk.html)
* [Buzzer (aka Tone Alarm) — Copter documentation - ArduPilot](https://ardupilot.org/copter/docs/common-buzzer.html)
* <https://discuss.ardupilot.org/>
* <https://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html>
* <https://dronekit-python.readthedocs.io/en/latest/guide/quick_start.html#basic-hello-drone>
* <https://www.researchgate.net/profile/Saif-Aldeen-Alkadhim/publication/319555090_Method_to_control_Pixhawk_board_using_raspberry_pi_with_python/data/59b811f5a6fdcc68722c17ca/Communicating-with-Raspberry-Pi-via-MAVLink.pdf?origin=publication_detail>
* <https://www.raspberrypi.com/software/>
* <https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/>
* <https://www.putty.org/>
* <https://www.academia.edu/19259375/Raspberry_Pi_The_Complete_Manual>

All the resources related to this work can be found on the following github repository: [github.com/sakshamssy/quadcopter\_navigation](https://github.com/sakshamssy/quadcopter_navigation)