Mashreq University
Faculty of Engineering
Department of Mechatronics

Plant Disease Monitoring By Agricultural Drone Using Deep Learning

Final year project submitted as partial fulfillment of the requirements for the degree of the BSc (Honor) in **Mechatronics Engineering**

Prepared by:

1- ABUBAKR BADRELMAARIF NOURELDAIM
2- ABDULLAH ALNOOR AHMED
3- ABDALAZIM ABKAR ADAM

Supervisor:

Dr. ABDULNASER MUHAMMED ZAIN

Jan / 2023

بسم الله الرحمن الرحيم

الآيـــــــــــــــة

قال تعالى :

)هُوَ الَّذِي بَعَثَ فِي الْأُمِّيِّينَ رَسُولًا مِّنْهُمْ يَتْلُو عَلَيْهِمْ آيَاتِهِ وَيُزَكِّيهِمْ وَيُعَلِّمُهُمُ الْكِتَابَ وَالْحِكْمَةَ وَإِن كَانُوا مِن قَبْلُ لَفِي ضَلَالٍ مُّبِينٍ (

صدق الله العظيم

سورة الجمعة -الآية.2

الإهداء

إلى من تعجز حروف اللغة عن تعبير الشكر والإمتنان إليهم ، إلى من علمونا معنى النضال والكفاح ، إلى من كانوا لنا عونا وسندا منذ أن إستقبلت أعيننا ضياء الكون...

أمهاتنا وآباءنا

إلى معلمتنا الجميلة ، مصدر فخرنا ومنبع علمنا...

جامعة المشرق –كلية الهندسة

إلى من منحونا الحب والصدق والأمان وفجروا في دواخلنا الإحساس الرائع بالأشياء وغرسوا كل قيم الجمال والإنتماء والتفاني في نفوسنا...

أساتذتنا الأجلاء

إلي من برفقتهم تجلت كل معاني الروعة والوفاء ومن منحونا أجمل الذكريات المنحوتة في أقصى ذاكرتنا...

زملائنا الكرام

وعرفان شكر

إلى فيها نعود وقفة من الجامعية الحياة في الأخيرة خطواتنا نخطو ونحن لنا لابد
الكثير لنا قدموا الذين الكرام أساتذتنا مع الجامعة رحاب في قضيناها أعوام
نتقدم نمضي أن وقبل ، المعرفة ناصية تمليكنا في كبيرا جهدا بذلك باذلين
جميع إلى والمحبة والتقدير والإمتنان الشكر ايات بأسمى
،،، الأفاضل أساتذتنا

والتقدير بالشكر ونخص

عبدالناصر محمد زين .د

المَشرُوع مُشرِف

البحث هذا إكمال في ساعدتنا معلومات من مايملك كل لنا ويقدم معنا ليقف وقته من استقطع و إليه نحتاج ما كل منحنا الذي
قال رسول الله (صلى الله عليه وسلم): "مَنْ صَنَعَ إِلَيْكُمْ .جزيل الشكري وفائق التقدير والاحترام منا فله ، الصورة بهذه وإخراجه
نسألك بنور وجهك الذي اللهم . (مَعْرُوفًا فَكَافِئُوهُ، فَإِنْ لَمْ تَجِدُوا مَا تُكَافِئُونَهُ فَادْعُوا لَهُ حَتَّى تَرَوْا أَنَّكُمْ قَدْ كَافَأْتُمُوهُ". (رواه أبو داوود
في حرزك وحفظك، وجوارك وتحت كنفك، اللهم ألبسهم ثياب الصحة والعافية، وارزقه من أشرقت له السماوات والأرض، أن تجعله
.واسع رزقك، وتقبل أعماله بالقبول الحسن يا الله

Abstract

Agriculture is an extremely important sector for the sustainability of the nations and hence many technologies have been developed and utilized to improve the production in agricultural lands. Some of these technologies are drones and deep learning programs. This research project focuses on interfacing these technologies together to help farmers and farming companies identify and locate plant diseases in a more precise manner. Hence reducing the cost of spraying fertilizer and pesticides over the whole crop field. A drone have been used to gather images of the plants while recording their locations. The data gathered by the drone is then processed and by a deep learning model to identify the plant diseases and their location. This output of the model is then plotted to visualize the areas which are infected from the areas which are healthy. The research discusses the capabilities of each technology in relation to our meant application. Like the accuracy of the GPS in farms, the stability of the drone and the accuracy of the model to detect plant diseases in images taken by a tiny camera.

المستخلص

تعتبر الزراعة قطاعًا مهمًا للغاية لازدهار الأمم ، وبالتالي تم تطوير العديد من التقنيات والاستفادة منها لتحسين الإنتاج في الأراضي الزراعية.

بعض هذه التقنيات هي طائرات بدون طيار وبرامج التعلم العميق يركز هذا المشروع البحثي على ربط هذه التقنيات معًا لمساعدة المزارعين.

ومن ثم تقليل تكلفة رش الأسمدة والمبيدات على .والشركات الزراعية على تحديد وتحديد مواقع أمراض النبات بطريقة أكثر دقة حقل المحاصيل المحاصيل بأكمله

ثم تتم معالجة البيانات التي تم جمعها بـواسطة الطائرة .تم استخدام طائرة بدون طيار لجمع صور النباتات أثناء تسجيل مواقعها ثم يتم رسم ناتج النموذج هذا لتصور المناطق .بدون طيار ومن خلال نموذج التعلم العميق لتحديد أمراض النبات وموقعها .المصابة من المناطق الصحية

في المزارع ، واستقرار (GPS) مثل دقة نظام تحديد المواقع العالمي .يناقش البحث قدرات كل تقنية فيما يتعلق بتطبيقنا المقصود .الطائرة بدون طيار ودقة النموذج للكشف عن أمراض النبات في الصور التي التقطتها كاميرا صغيرة

List of Figures

List of tables

Abbreviations:

CNN ≡	Convolutional Neural Network
ANN ≡	Artificial Neural Networks
PC ≡	Personal Computer
SOM ≡	Self Organizing Map
FPV ≡	First Person View
APM ≡	Ardupilot Mega
Lipo ≡	Lithium Ion Polymer
AAD ≡	Average Amp Draw
AUW ≡	All-up weight of drone
GPS ≡	Global positioning system
UAV ≡	Unmanned Aerial Vehicle
RPM ≡	Revolutions Per Minute
BLDC ≡	Brushless Direct Current

List of Contents

Chapter 1
Introduction

1.1 General Introduction

Agriculture is the most important thing in many countries. It plays a critical role in the entire life of a given economy because it represents the main source of national income for most developing countries.

It employs many people as a result, the national income level as well as people's standard of living is improved. The occurrence of diseases on plants could degrade the quality and decrease the quantity so, recently many technologies and studies were developed to provide smart agricultural systems to the agricultural society and enhance productivity. One of these technologies are drones.

A drone is an unmanned aircraft. Drones are more formally known as unmanned aerial vehicles (UAVs) or unmanned aircraft systems. Essentially, a drone is a flying robot that can be remotely controlled or fly autonomously using software-controlled flight plans in its embedded systems, that work in conjunction with on board sensors and a global positioning system GPS.

Unmanned aerial vehicles are used in agriculture operations mostly in yield optimization and in monitoring crop growth and crop production. The aerial view provided by a drone can assist in the information of crop growth stages, crop health and soil variations in real time helping in any mitigation if required.

1.2 project overview

In this project a drone system have been implemented, it consists of drone and effective camera to capture the images and their locations then store them on sd card, besides this there is a machine learning model on PC to detect the diseases.

1.3. Problem Statement:

Failing to detect diseases early on can cause a lot of crop loss as some diseases are infectious and can spread through a large area. These disease can drastically reduce the crop yield and its quality. Manually inspecting plant leaves for diseases and recording their location can be extremely hard for a human especially in hot climates as in Sudan. Manual human inspection is also prone to errors from distraction.

1.4 Objectives

The main objective of this research is to assemble a drone that can fly over a field and take pictures along with their locations. Build machine learning model on PC which can detect diseases in crops.

1.5 Methodology

A drone will be manually controlled using a remote controller to fly it through the field. The pilot will capture images of the plant leaves by sending commands to the camera on the drone. These images as well as the location coordinates will the be sent back the ground station. Final the data will be processed by the deep learning model in the computer.

1.6 Thesis outlines

Chapter one gives an introduction to the project and explains the problem statement. Chapter two is about literature review, chapter three shows the block diagram, circuit diagram of the project and all components that have been used in the project and explains the methodology. Chapter four discusses the results obtained. Chapter five discusses the conclusions from this project and recommendations for future work.

Chapter 2
Literature review

2.1 Preface

In the past years, many researches and projects were done to interface drone technologies with machine learning and deep learning models in the agricultural field. These researches were done for different applications like detection and classification plant diseases, weeds and pests, while others focused on mapping the farm or monitoring the stages of crop for optimum time for harvest.

One of these projects is a drone and analysis system for detection of plant diseases by Lakshmi Narayana and Sai Divya. They developed a hexacopter with the raspberry pi interfaced camera to capture the diseased leaf images. Then those images are pre-processed to eliminate noise and color differences. It is followed by image segmentation and feature extraction to classify the disease based upon the features like shape, color, texture. classification is done using different classifiers like SVM, CNN, random forests, etc,. Clustering algorithms are used to extract features. So, the main goal of this paper is to decrease the tremendous efforts & risks of humans and stop the spreading of the disease at early stage of plant growth without ruining the entire crop production during the harvest period. [1]

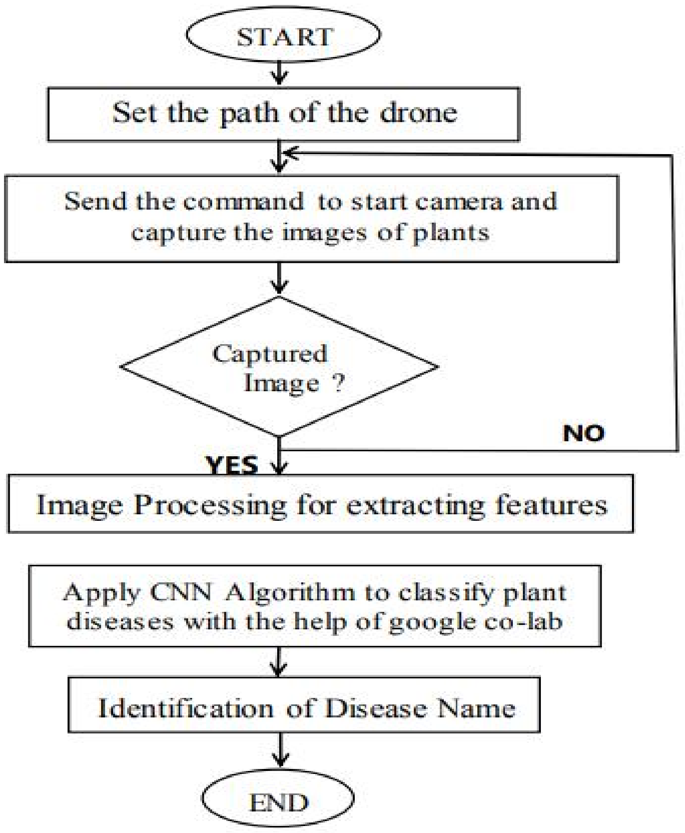

START
Set the path of the drone
Send the command to start camera and capture the images of plants
Captured Image ?
NO
YES
Image Processing for extracting features
Apply CNN Algorithm to classify plant diseases with the help of google co-lab
Identification of Disease Name
END

Figure (2. 1): Monitoring system flow chart

2.2 Convolutional neural network (CNN) and Artificial neural networks (ANN)

In Deep Learning research, CNNs are specifically applied for Computer Vision applications that involves Image Classification and Object Recognition. For example in a research about flower species recognition, the recognition is a combination of both Object Recognition and Image Classification, as the system must detect a flower in the image as well as recognize which species it belongs to. To recognize the flower species, an intelligent system must be trained with larger set of images, so that it could predict the flower species from its learned patterns. This approach is termed as "Supervised Learning" which requires an external dataset of images with labels to predict the label of an unseen image. This research work uses Convolutional Neural Networks (CNN) along with Transfer Learning as the intelligent algorithm to efficiently recognize flower species in real-time. The major difference between a traditional Artificial Neural Network (ANN) and CNN is that only the last layer of a CNN is fully connected whereas in ANN, each neuron is connected to every other neurons as shown in Figure below, ANNs are not suitable for images because these networks leads to over-fitting easily due to the size of the images. Consider an image of size [32x32x3]. If this image is to be passed into an ANN, it must be flattened into a vector of 32x32x3 = 3072 rows. Thus, the ANN must have 3072 weights in its first layer to receive this input vector. For larger images, say [300x300x3], it results in a complex vector (270,000 weights), which requires a more powerful processor to process.[2]

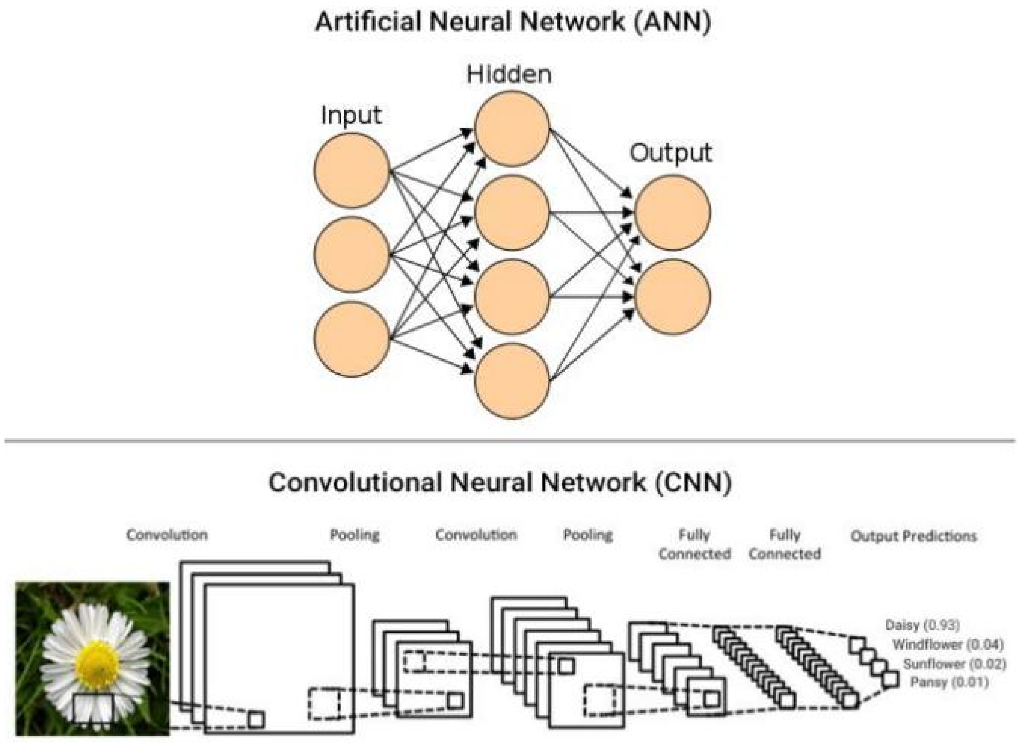


Figure (2.2): ANN and CNN work structure

Rice diseases, such as bacterial blight, brown spot, leaf blast, and sheath blight, are the major problems that threaten the cultivation. Many studies conducted to detect and classify rice diseases using different image processing techniques [3] ,used open source data from international research institute (ISSI) database and applied K-mean clustering algorithm to perform rice leaf segmentation of blast disease [4] while proposed Otsu’s threshold to segment the brown spot disease and leaf blast disease. Neural network was used to classify the diseases. [5] applied self organizing map (SOM) neural network to train a classifier. Other like [6] collected data manually by using digital camera to capture images. Then, they used Artificial neural network (ANN) and Fuzzy Logic method to classify the disease like blast, brown spot and narrow brown spot.

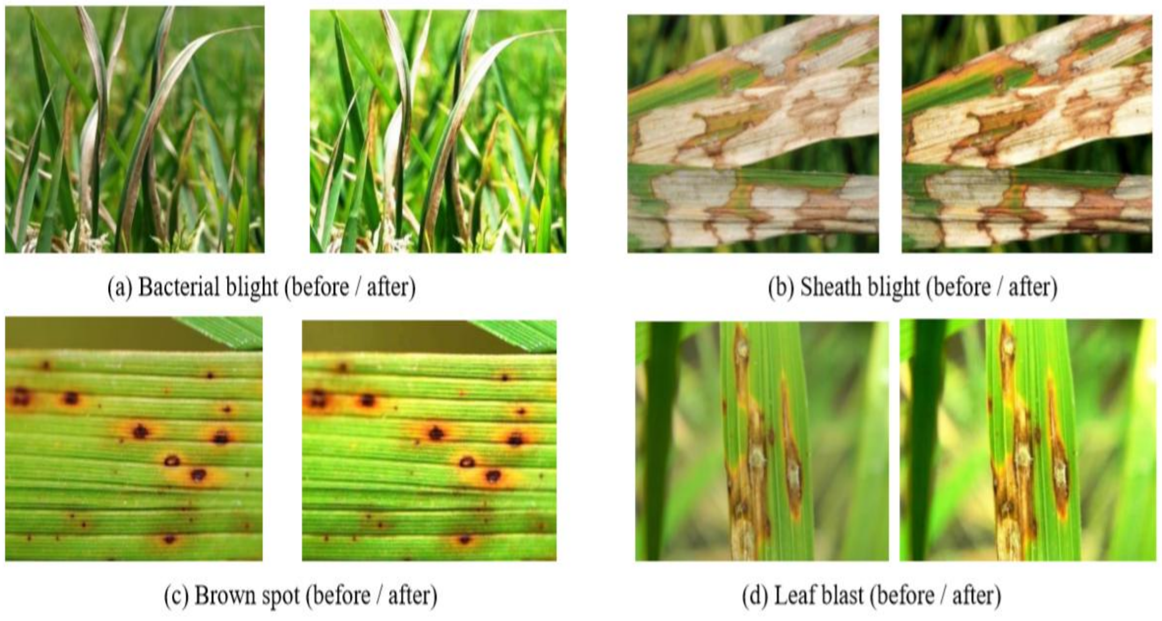

(a) Bacterial blight (before / after)
(b) Sheath blight (before / after)
(c) Brown spot (before / after)
(d) Leaf blast (before / after)

Figure (2. 3): CHARACTERISTICS OF FOUR RICE DISEASES

Table (2.1) features of the diseases

Disease	Color	Shape	Occurring Period
Bacterial Blight	Yellow to grayish white with black dots	Wavy margin toward leaf base	Tillering and heading stage
Sheath Blight	Gray-white center with brown margins	Oval or ellipsoid, 1-3 cm long	Young tillering stage
Brown Spot	Dark brown to purple, light brown to gray center with reddish brown margin	Circular to oval	All crop stages
Leaf Blast	White to grey-green in center with red to brownish border	Elliptical or spindle-shape	Tillering Stage

2.3 APPLICATION OF UAV FOR PEST, WEEDS AND DISEASE DETECTION USING OPEN COMPUTER VISION

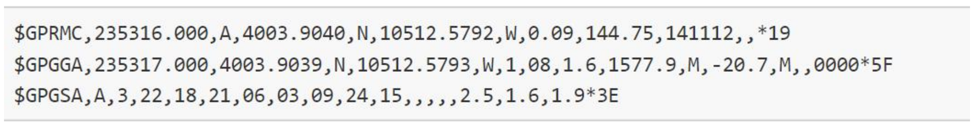
In research areas for creating sustainable agriculture, drones equipped with cameras can view every centimetre of a piece of land from several angles. It is a much more accurate way to monitor a farm than doing it by foot. Drones can create a digital map of a field, detect problems with crop health, find missing livestock[7]. Various image processing methods been successfully been applied for disease detection. Examples of such machine learning methods that have been applied in agricultural researches; Artificial Neural Networks (ANNs), Decision Trees, K-means, k nearest neighbours [8], support vector mechanism for pest detection [9] and NIR and Hui methods for weed detection.

2.4 GPS receiver

The data sent down to earth from each satellite contains a few different pieces of information that allows the GPS receiver to accurately calculate its position and time. An important piece of equipment on each GPS satellite is an extremely accurate atomic clock. The time on the atomic clock is sent down to earth along with the satellite's orbital position and arrival times at different points in the sky. In other words, the GPS module receives a timestamp from each of the visible satellites, along with data on where in the sky each one is located (among other pieces of data). From this information, the GPS receiver now knows the distance to each satellite in view. If the GPS receiver's antenna can see at least 4 satellites, it can accurately calculate its position and time. This is also called a lock or a fix.

2.4.1 Message Formats

GPS data is displayed in different message formats over a serial interface. There are standard and non-standard (proprietary) message formats. Nearly all GPS receivers output NMEA data. The NMEA standard is formatted in lines of data called sentences. Each sentence contains various bits of data organized in comma delimited format (i.e. data separated by commas). Here's example NMEA sentences from a GPS receiver with satellite lock (4+ satellites, accurate position)



```
$GPRMC,235316.000,A,4003.9040,N,10512.5792,W,0.09,144.75,141112,,*19
$GPGGA,235317.000,4003.9039,N,10512.5793,W,1,08,1.6,1577.9,M,-20.7,M,,0000*5F
$GPGSA,A,3,22,18,21,06,03,09,24,15,,,,,2.5,1.6,1.9*3E
```

figure 2.4. GPS data in NMEA format

For example, the **GPGGA** sentence contains the follow:

- **Time**: 235317.000 is 23:53 and 17.000 seconds in Greenwich mean time
- **Longitude**: 4003.9040,N is latitude in degrees.decimal minutes, north
- **Latitude**: 10512.5792,W is longitude in degrees.decimal minutes, west
- **Number of satellites seen**: 08
- **Altitude**: 1577 meters

Chapter 3
Design and Implementation

3-1 Preface

To build such a dynamic unmanned aerial vehicle(UAV) there is a need to attach many complex electronic devices. In this implementation, many intelligent electronic devices were used. The modules of the quadcopter are related together according to the below figure (3.1). The transmitter remote sends a signal to the receiver, which in turn sends the signal to the flight controller and controls the motors via the ESC and the GPS is to locate the drone. Pictures are taken by giving a signal from the phone to the Raspberry pi 3 wirelessly, and the GPS module saves the location of the pictures. The flow of these processes is described in the flow chart in figure (3.2).

3.2 Block diagram:

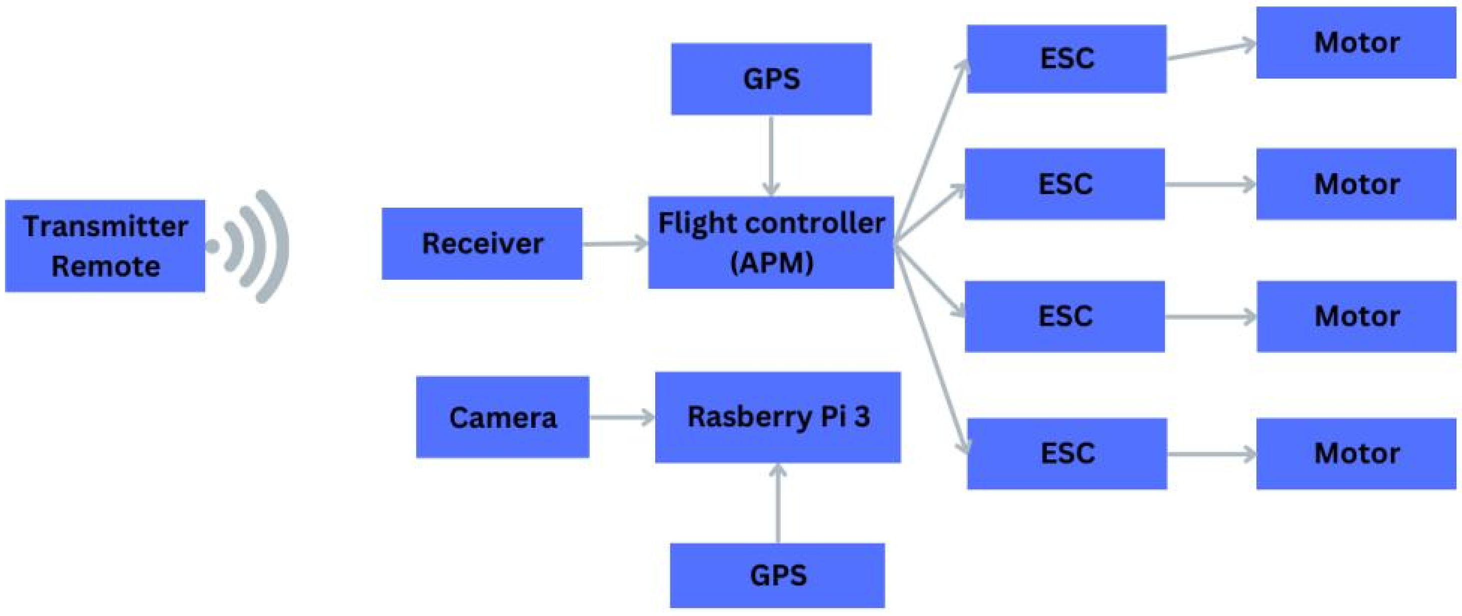


Figure (3.1) Block diagram of the quadcopter

3.3 Flow chart:

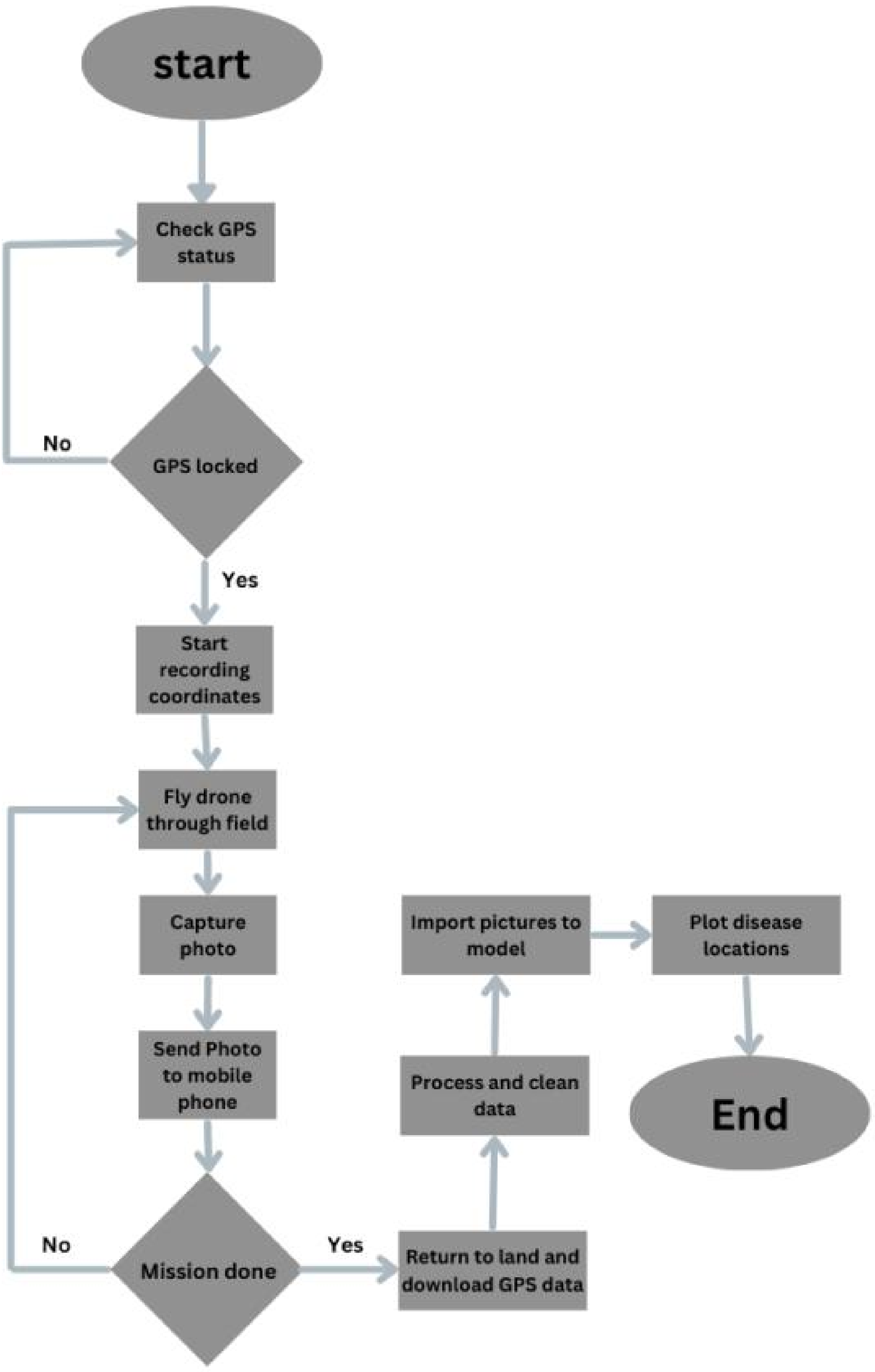


Figure (3.2) flow chart of the quadcopter

All parts of the drone have been assembled and the wires were soldered as shown in figure(3.3) according to the circuit diagram below in figure (3.4).

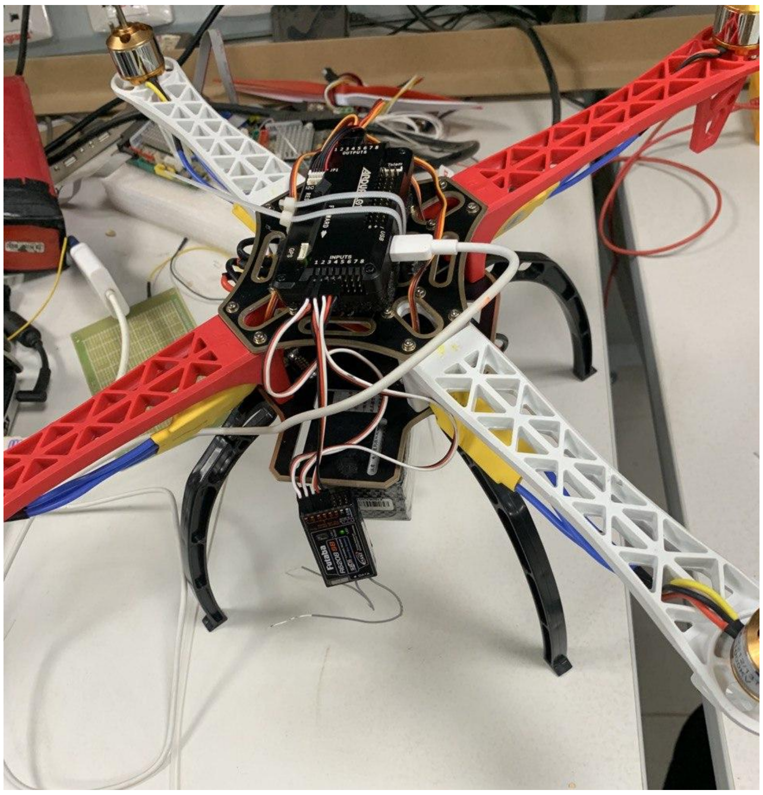


Figure (3.3) Assembled drone

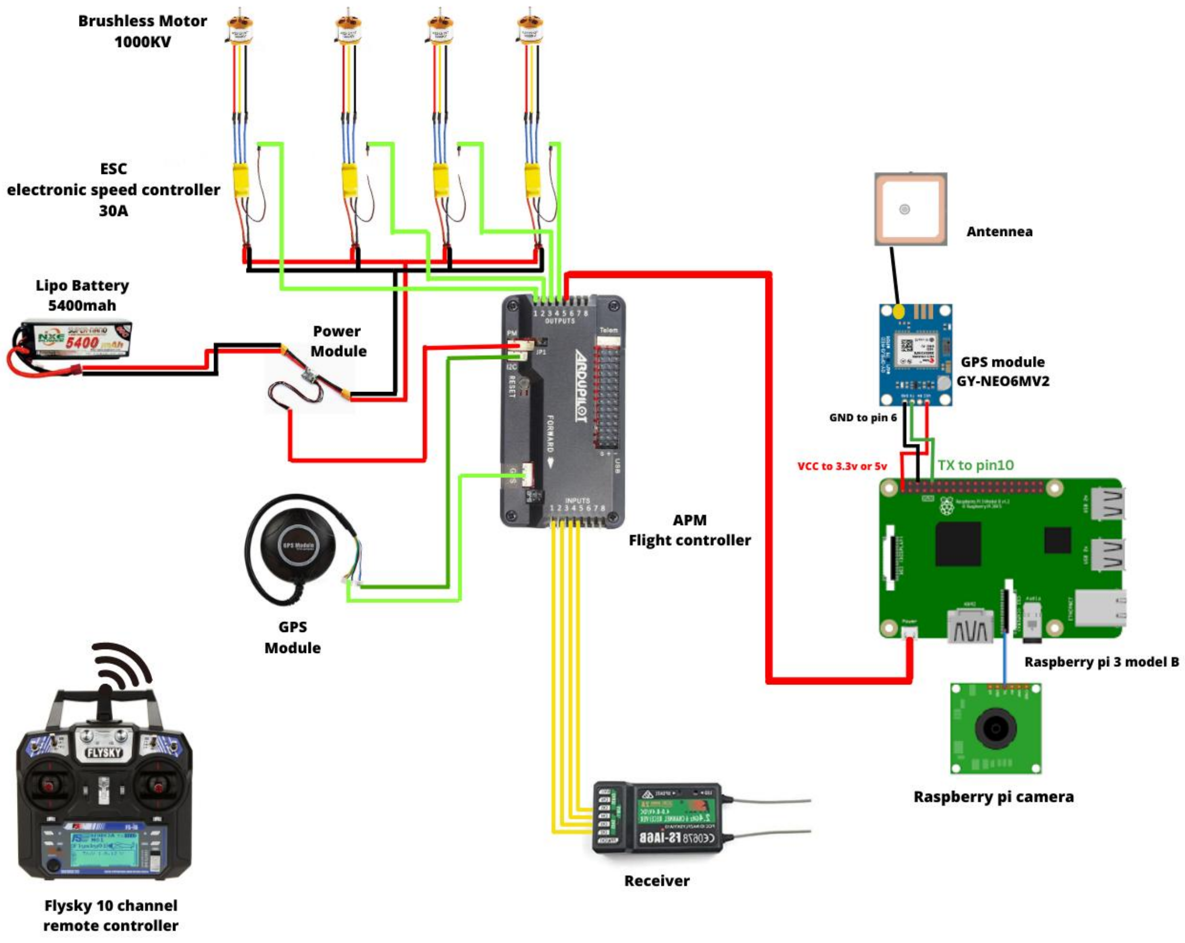


Figure (3.4) Circuit diagram

3.4 components:

3.4.1 brushless motors 1000kv:-

Used to provide thrust to the RC quadcopter. It is built with high quality bearings and stators. It has high efficiency and constant power to drive the propellers for a stable flight. Patented balancing technology ensures a high performance.

Figure (3.5) Brushless DC motor that is used in the project

Table (3.1) Specifications of BLDC motors

No of Cells:	2-3 li-poly / 6-10 NiCd/NiMH
Kv:	1000 RPM/V
Max Efficiency:	80%
No Load Current:	0.5A @ 10V
Resistance:	0.090 ohms
Max Efficiency Current:	4-10A (>75%)
Max Current:	13A for 60s
Max Watts:	150W
Weight:	52.7 g
Size:	28 mm dia x 28 mm bell length
Shaft Diameter:	3.2 mm
Poles:	14
Model Weight:	300-800g

3.4.2 30A Electronic speed controllers (ESC):-

A30 amp ESC Electronic Speed Controller with Connector is specifically made for quadcopters and multi-rotors,It provides faster and better motor speed control giving better flight performance compared to other available ESCs,STANDARD 30 amp ESC Electronic Speed Controller can drive motors that consume up to 30A current. It works on 2-3S LiPo batteries.

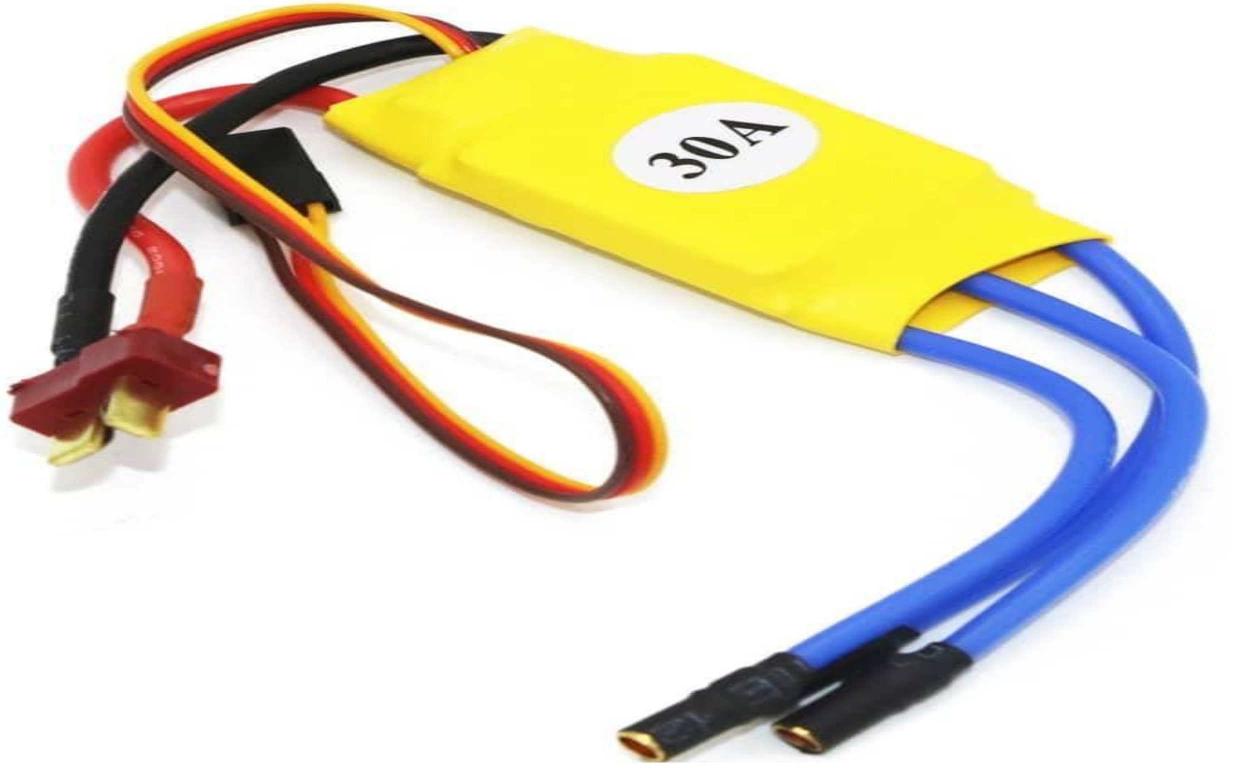


Figure (3.6) Electronic speed controller

Specifications:-

1. Input voltage: DC 6-16.8V(2-4S Lixx)
2. Running current:30A(Output: Continuous 30A, Burst 40A up to 10 Secs.)
3. Size: 26mm (L) * 23mm (W) * 11mm (H).
4. Weight: 32g.

3.4.3 F450 Fram*e:-*

The standard frame mostly used for the quadcopters are F450 frame because of its light weight and high strength. F450 frame has a shape of 'X' geometry which provides more room space for mounting of additional components on the quadcopter.

Figure (3.7) F450 frame

3.4.4 Propellers:-

This 1045 propeller can be used by brushless motors with a 800-2200 kv rating and with a low kv motor (800-1400 kv) this propeller offers smooth flights with longer fight time perfect for FPV and aerial photography with a high kv motor (more than 1200 kv) this propeller offers fast flights perfect for acrobatic flights.

Figure (3.8) 1045 propeller

This propeller can be used with our A 2212 1000KV 1800KV and 2200 motors and a 30 A ESC

Specifications of 10x4.5-inch propeller pair:

. Diameter: 10' (25.4cm)

. pitch: 4.5'(11.43cm)

. includes one CW rotating propeller and one CCW rotating propeller

3.4.5 The transmitter:-

Works in the frequency range of 2.405 to 2. 475GHz.This band has been divided into 142 independent channels each radio system uses 16 different channels and 160 different types of hopping algorithm.

This radio system uses a high gain and high quality multi directional antenna, it covers the whole frequency band. Associated with a high sensitivity receiver, this radio system guarantees a jamming free long range radio transmission .

Each transmitter has a unique ID, when binding with a receiver, the receiver saves that unique ID and can accept only data from the unique transmitter. This avoids picking another transmitter signal and dramatically increase interference immunity and safety.

Figure (3.9) *t*ransmitter

3.4.6 APM 2.8 flight controller:-

The APM 2.8 is a complete open source autopilot system and the bestselling technology. It allows the user to turn any fixed, rotary wing or multirotor vehicle (even cars and boats) into a fully autonomous vehicle; capable of performing programmed GPS missions with waypoints. This revision of the board has no onboard compass, which is designed for vehicles (especially multicopters and rovers) where the compass should be placed as far from power and motor sources as possible to avoid magnetic interference.

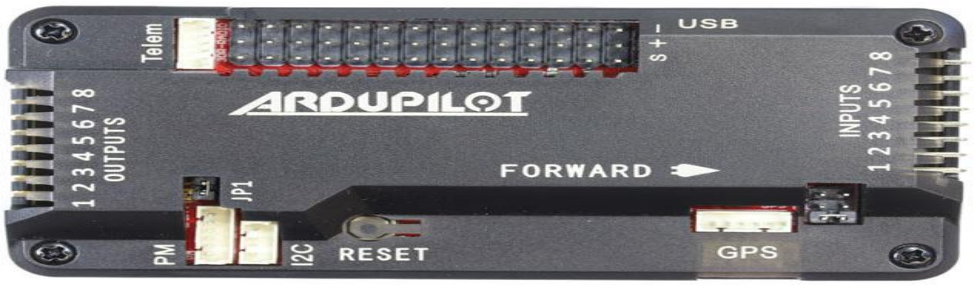


Figure (3.10) ArduPilot Mega 2.8 APM Flight Control that is used in the project

Specification:-

Needle type: Right angle (Soldered to the board)

Supply Voltage:

Size: 10x80x120 mm

Weight: 30g

Recommended software: Mission Planner, ArduPilot.

Inbuilt Compass: NO

Features:

Arduino Compatible!

Includes 3-axis gyro, accelerometer, along with a high-performance barometer.

Onboard 4 MegaByte Dataflash chip for automatic data logging.

Optional off-board GPS, uBlox LEA-6H module with Compass.

One of the first open-source autopilot systems to use Invensense’s 6 DoF Accelerometer/Gyro MPU-6000.

Barometric pressure sensor upgraded to MS5611-01BA03, from Measurement

3.4.7 The receiver:-

Suitable for FS-I6X FS-I6S I8 RC transmitters, etc. It can be used on airplane, glider or helicopter.
FS-iA6B adopts 26mm dual antennas that obtain radio-frequency signal.
The indicator light shows the power supply and working status of the receiver in real time.
2.4G modes: automatic frequency hopping digital system second generation.

Figure (3.11) The receiver

Specification:-

3.4.8 GPS NEO M8N MODUL:-

SIZE:

24.9 mm x 47.3 mm x 14.3 mm
0.98 in x 1.86 in x 0.56 in
WEIGHT:
13.8 g
0.49 oz

The NEO-M8N provides the best performance and easiest RF integration of all the NEO-M8 configurations. The NEO form factor allows easy migration from previous NEO generations. Sophisticated RF-architecture and interference suppression ensure maximum performance even in GNSS-hostile environments.

Figure (3.12) NEO *GPS M8N*

Specifications:-

Receiver Type72-channel Ublox M8 engine
Main Chip Ublox NEO-M8N

Input Supply Voltage (VDC)0.5 ~ 3.6
Sensitivity (dBm)Tracking & Navigation: -161 dBm
Position Accuracy (Meter) 2, 2.5
Operating Temperature Range (°C)-45 to 105

Raspberry Pi 3 Model B:-

The Raspberry Pi 3 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Whilst maintaining the popular board format the Raspberry Pi 3 Model B brings you a more powerful processor.

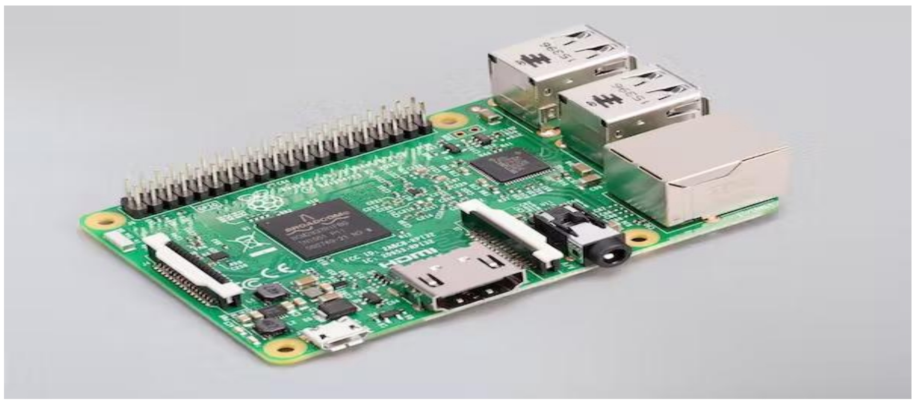


Figure (3.13) The Raspberry Pi 3 Model B

Specifications:

Processor Broadcom BCM2387 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.1 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE) GPU Dual Core VideoCore IV® Multimedia Co-Processor

3.4.10 Raspberry Pi Camera Board v1.3:

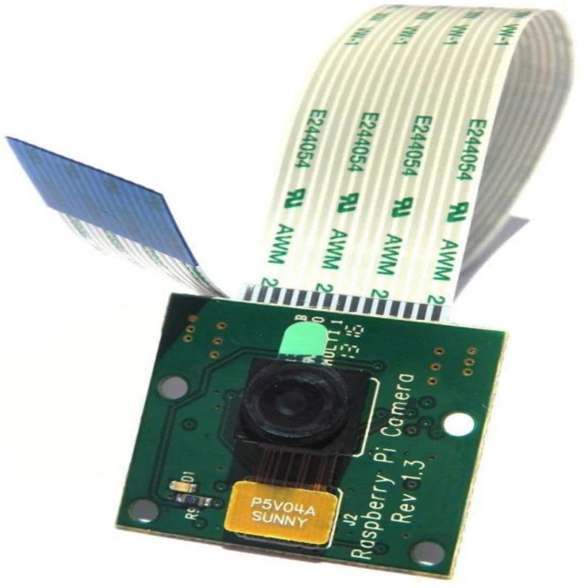


Figure (3.14) Raspberry Pi Camera Board v1.3

The Features:

Fully Compatible with Both the Model A and Model B Raspberry Pi

5MP Omnivision 5647 Camera Module

Still Picture Resolution: 2592 x 1944

Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording

15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board

Size: 20 x 25 x 9mm

Weight 3g

3.4.11 lipo Battery 5300mah:

Application: large-scale sport and aerobatic airplane models as well as big-sized Jet and 700 helicopter & helicam etc

DNOT PULL
ace
MODEL:5300mAh-30C 14.8V-4S1P 78.4Wh
HIGH DISCHARGE LI-PO BATTERY
5300 30C mAh
5C

Figure (3.15) lipo Battery 5300mah:

Specifications:

Minimum Capacity: 5300mAh

Configuration: 4S1P / 14.8V

Discharge Rate: 30C

Max Burst discharge Rate: 60C

Net Weight(±20g): 531g

Dimensions: 140mm Length x 43mm Width x 41mm Height.

3.4.12 power adapter:

Power adapter is used to regulate the power from the battery and distribute to the ESCs and to power the flight controller.

Specifications:

Maximum input voltage of 18V (4S lipo)

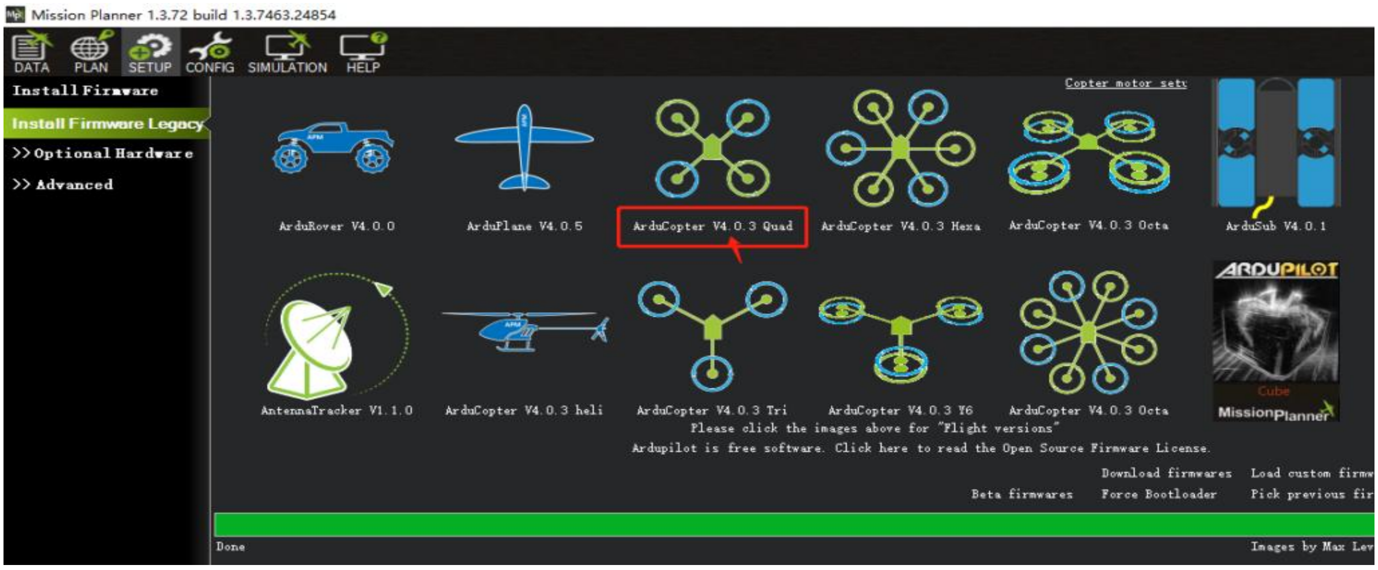
Maximum of 90 Amps (but only capable of measuring up to 60 Amps)

Provides 5.37V and 2.25Amp power supply to the autopilot

Figure (3.16) power adapter

3.5 Calibrations:

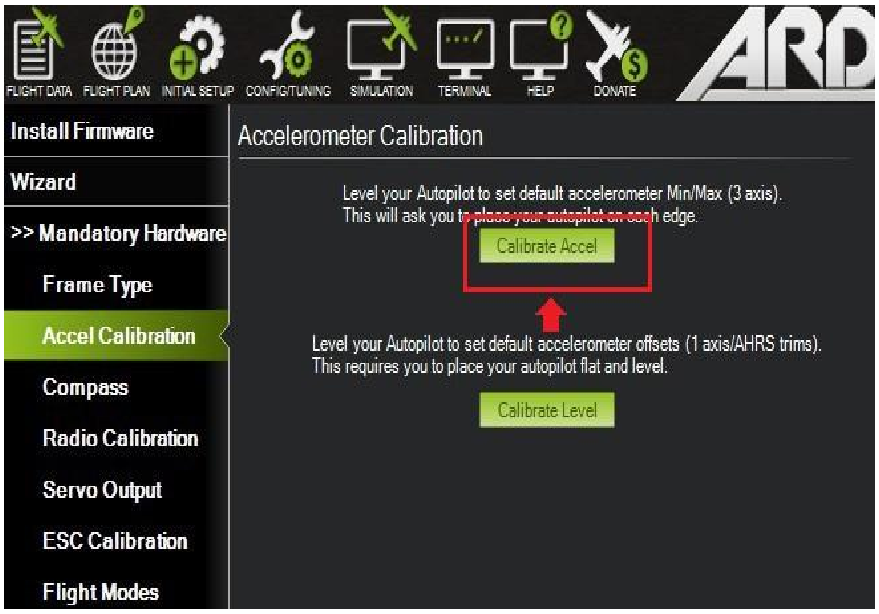
APM firmware is the brains of the autopilot operation and must be installed before using ardupilot. To load firmware on to ardupilot, mission planner application was installed on the ground station computer.



Please click the images above for "Flight versions"
Ardupilot is free software. Click here to read the Open Source Firmware License.

Figure (3.17) : Firmware of ardupilot from mission planner

Initial Setup was selected, the firmware was installed, and vehicle type was selected. Select Accel Calibration, check the box for AC 3.0+, select Calibrate, and followed the prompts to calibrate ardupilots's accelerometer. A couple of seconds were waited before moving the vehicle.



Level your Autopilot to set default accelerometer Min/Max (3 axis).
This will ask you to place your autopilot on each edge.

Level your Autopilot to set default accelerometer offsets (1 axis/AHRS trims).
This requires you to place your autopilot flat and level.

Figure (3.18) Accelerometer calibration

Radio calibration was done to teach ardupilot to work with your RC transmitter.

The transmitter was turned on, selected Calibrate Radio, and moved all sticks and switches to their extreme positions. clicked "Click when Done" once the red bars are set for all available channel.

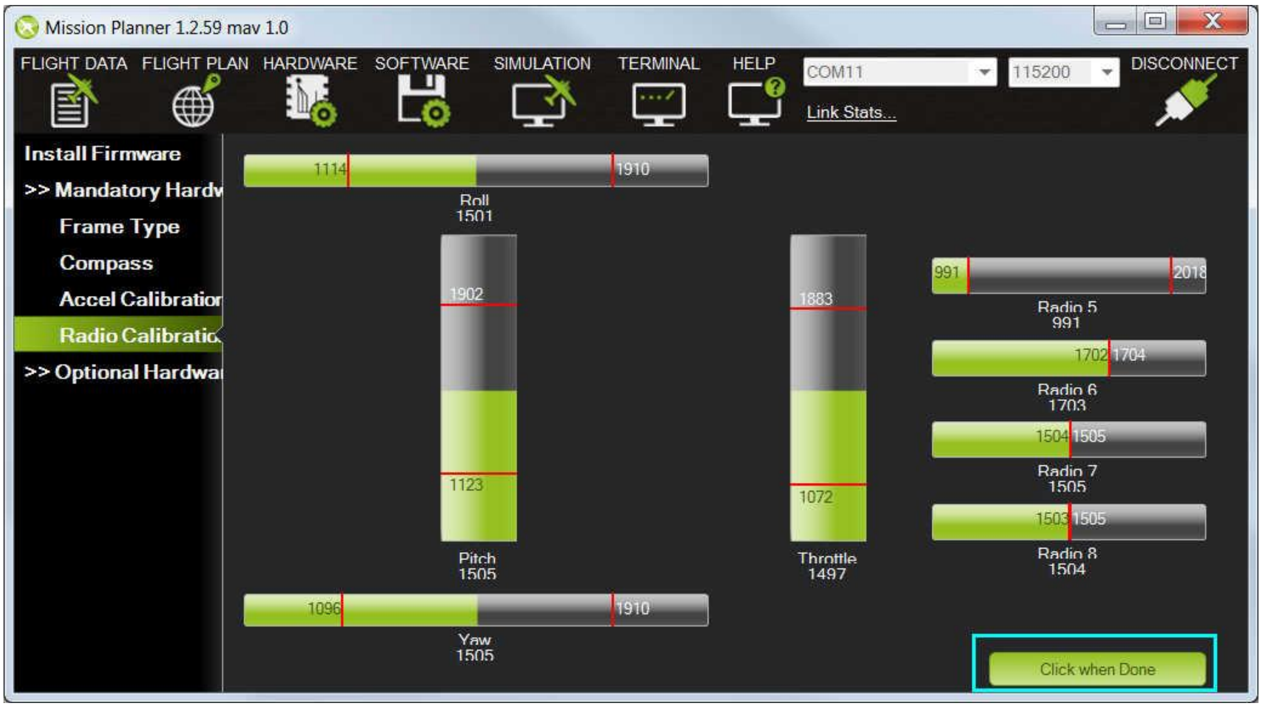
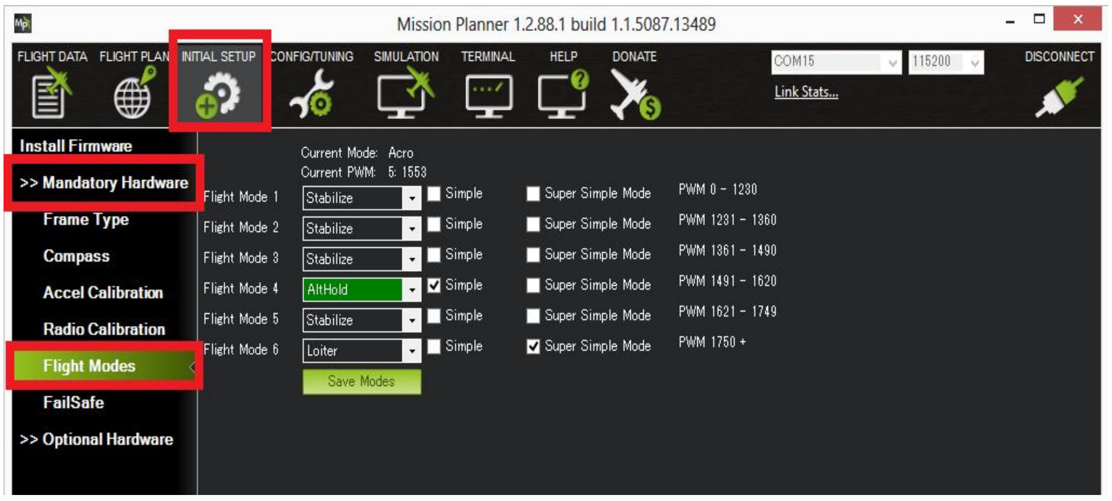


Figure (3.19) RC Calibration

For the modes calibration, moved each switch on the transmitter to its available positions. The mission planner indicated the currently selected position with green highlighting.

Select a mode for each switch position, and selected Save Modes to assign.

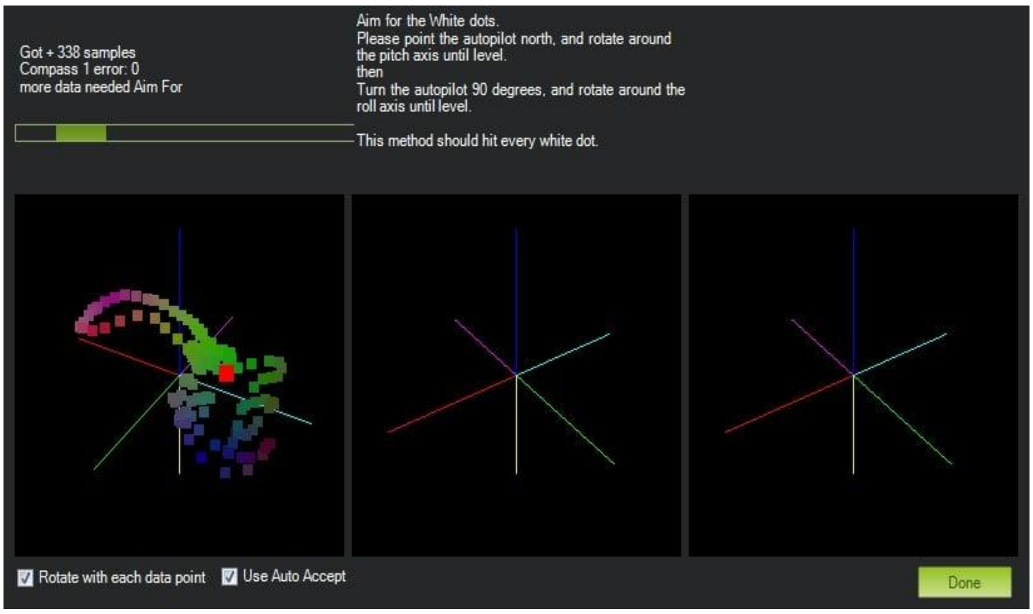


Current Mode: Acro
Current PWM: 5: 1553

Flight Mode 1	Stabilize	☐ Simple	☐ Super Simple Mode	PWM 0 - 1230
Flight Mode 2	Stabilize	☐ Simple	☐ Super Simple Mode	PWM 1231 - 1360
Flight Mode 3	Stabilize	☐ Simple	☐ Super Simple Mode	PWM 1361 - 1490
Flight Mode 4	AltHold	✓ Simple	☐ Super Simple Mode	PWM 1491 - 1620
Flight Mode 5	Stabilize	☐ Simple	☐ Super Simple Mode	PWM 1621 - 1749
Flight Mode 6	Loiter	☐ Simple	✓ Super Simple Mode	PWM 1750 +

Figure (3.20) Flight modes calibration

To do Compass calibration the drone was moved around all axes because the drone need to know where his position while it's flying .



Got + 338 samples
Compass 1 error: 0
more data needed Aim For

Aim for the White dots.
Please point the autopilot north, and rotate around the pitch axis until level.
then
Turn the autopilot 90 degrees, and rotate around the roll axis until level.
This method should hit every white dot.

✓ Rotate with each data point ✓ Use Auto Accept

Figure (3.21) Compass calibration

When the motors were supposed to be armed ,the motors did not arm. So the pre_arm check in the standard parameters was disabled by setting its value to zero.

Rashik's model was used as the base of the deep learning model and then modified by
using google colab and kaggle those websites give a free service for processing and training the model.
The model was modified to receive images of size 120*120 pixels rather than 80*80. A code was written that saves the weights of the model in the form of checkpoints so as to allow training the model several times in discontinued intervals.
A dataset from "Kaggle" was fetched and was used to train the model and then test it .

The camera connected to the raspberry pi captured the images and the GPS module recorded their locations and sent them to mobile phone. The pictures were sent to the phone immediately after each capture and GPS data was retrieved at the end of the flight. At the ground-station, a program was written to extract the time of each picture and the time for each GPS location frame. The data was then organized in tables. The pictures were then fed to the model to output their predictions.

The predictions along with the labels which were manually written for each picture were compared to calculate the accuracy of the model.

The data from the raspberry pi is processed in the computer of the ground-station according to the below flow chart in figure (3.22).

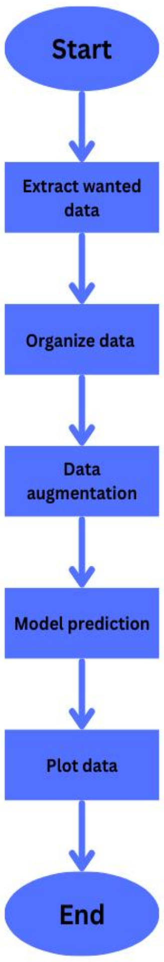
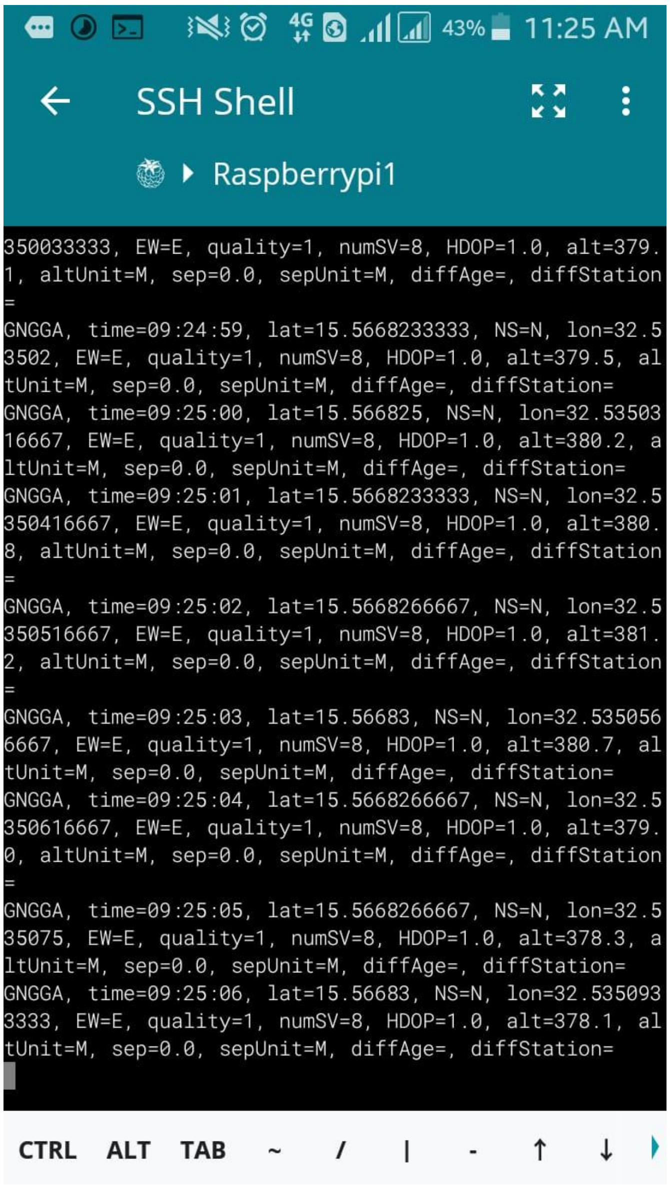


Figure (3.22) Data process flow chart

In this project data undergoes different processes by 5 different codes. The first code "GPS_working_file1.py" was used to fetch the GPS frame sentences from the RX pin and displaying it directly on command terminal of the raspberry pi. This allowed the user to know when the GPS got a fix or lock on the satellite. After getting a fix the user started "GPS_working_file2.py" to save the GPS sentences into a csv file format. These programs were started from a mobile app called "raspcontroller" as shown in Figure (3.23). The raspcontroller app was also used for capturing the images by controlling the camera on the raspberry pi and saving the images on the phone.

After finishing the flight mission the csv file containing the location data for each datapoint was

inputted into “GPS_data_processing” notebook in google colab to extract the wanted data only and organizing it into a dataframe and also extracting the time of capturing for each picture.
“Plant disease recognition” notebook in “kaggle” was then used to recognize the disease in each picture and and organizing the output in a dataframe and saving it to csv file.
Finally to plot the datapoint, the csv file containing the location points was inputted into “GPS_plot_map” notebook in google colab to plot the points in a map.



```
350033333, EW=E, quality=1, numSV=8, HDOP=1.0, alt=379.
1, altUnit=M, sep=0.0, sepUnit=M, diffAge=, diffStation
=
GNGGA, time=09:24:59, lat=15.5668233333, NS=N, lon=32.5
3502, EW=E, quality=1, numSV=8, HDOP=1.0, alt=379.5, al
tUnit=M, sep=0.0, sepUnit=M, diffAge=, diffStation=
GNGGA, time=09:25:00, lat=15.566825, NS=N, lon=32.53503
16667, EW=E, quality=1, numSV=8, HDOP=1.0, alt=380.2, a
ltUnit=M, sep=0.0, sepUnit=M, diffAge=, diffStation=
GNGGA, time=09:25:01, lat=15.5668233333, NS=N, lon=32.5
350416667, EW=E, quality=1, numSV=8, HDOP=1.0, alt=380.
8, altUnit=M, sep=0.0, sepUnit=M, diffAge=, diffStation
=
GNGGA, time=09:25:02, lat=15.5668266667, NS=N, lon=32.5
350516667, EW=E, quality=1, numSV=8, HDOP=1.0, alt=381.
2, altUnit=M, sep=0.0, sepUnit=M, diffAge=, diffStation
=
GNGGA, time=09:25:03, lat=15.56683, NS=N, lon=32.535056
6667, EW=E, quality=1, numSV=8, HDOP=1.0, alt=380.7, al
tUnit=M, sep=0.0, sepUnit=M, diffAge=, diffStation=
GNGGA, time=09:25:04, lat=15.5668266667, NS=N, lon=32.5
350616667, EW=E, quality=1, numSV=8, HDOP=1.0, alt=379.
0, altUnit=M, sep=0.0, sepUnit=M, diffAge=, diffStation
=
GNGGA, time=09:25:05, lat=15.5668266667, NS=N, lon=32.5
35075, EW=E, quality=1, numSV=8, HDOP=1.0, alt=378.3, a
ltUnit=M, sep=0.0, sepUnit=M, diffAge=, diffStation=
GNGGA, time=09:25:06, lat=15.56683, NS=N, lon=32.535093
3333, EW=E, quality=1, numSV=8, HDOP=1.0, alt=378.1, al
tUnit=M, sep=0.0, sepUnit=M, diffAge=, diffStation=
```

Figure (3.23) Raspcontroller terminal screen on smartphone

3.6 Calculations:

The motor:

There are two important elements in order to choose the appropriate motor:

The first: thrust to weight ratio; The ratio of thrust or lift to weight. In order for the drone to fly, the thrust generated by the motors must be 50% higher than the weight. The right of the drone includes all components at least. For example, if the weight of the drone was 1 kg, the total thrust generated by the motors would be 2 kg (ratio 2:1).

In the racing drone, the ratio is higher, and the ratio is recommended to be (5:1).

The second element: the kv

This means that every 1 volt generates the same RPM, for example, the 1000KV motor generates 1000RPM for every 1 volt of the battery. The higher the Kv, the faster the motor. For example, a 2200kv motor is faster than a 1000kv motor, and so on.

If you want the drone to be fast, then you should choose the KV motors, which are very high. There is an inverse relationship between torque and speed; The higher the motor speed, the lower its torque, and vice versa.

So a great torque is needed in the case of two wheels, the lifting or pushing force is large, and this is always the case in the case of the load, or the big load, or the weight is large, in this case more torque is needed than speed. For example: the 920kv motor has higher torque than the 1000kv motor.

In case of a drone what's more important is the speed.

Thrust calculations:

Total mass = 1300g

1000 Kv brushless motor provides

thrust = 7,848N
4 motors = 7,848 * 4 = 31,392 N
safety factor = 31,392 / 2 = 15,696N
extra weight = 15,696 - 12,753 = 2,943N

Calculation of the flight time :-

time = capacity × discharge / AAD

time – Flight time of the drone, expressed in minutes.

capacity – Capacity of the battery, expressed in milliamp hours (mAh).

discharge – Battery discharge that you allow during the flight. As LiPo batteries can be damaged if fully discharged, it's common practice never to discharge them by more than 80%.
AAD – Average amp draw of your drone, calculated in amperes.

Calculate the amp draw

AAD = AUW × P / V

- AAD – Average amp draw, expressed in amperes.
- AUW – All-up weight of your drone – the total weight of the equipment that goes up in the air, including the battery. It is usually measured in kilograms.
- P – Power required to lift one kilogram of equipment, expressed in watts per kilogram. Our drone flight time calculator assumes a conservative estimate of 170 W/kg.
- V – Battery voltage expressed in volts.

  AAD = AUW × I

  I - the current (in amps) required to lift one kilogram into the air

  Calculate the average amp draw:

AAD = AUW × P / V

AAD = 1.3 × 170 / 16 = 14.9 A

use the drone flight time formula:

time = capacity × discharge / AAD

time = 5.3× 80% / 14.9= 0.285 hrs = 17.1 min

Chapter 4

Results and Discussion

4.1 overview

This chapter reviews the results of the project while noting all of the parameters and conditions that led to these results.

4.2 GPS accuracy

The GPS module GY-NEO6MV2 was tested to measure it's accuracy in areas blocked by trees as well as areas open to the sky.

While capturing pictures by the raspberry pi camera, a code was running in raspberry pi in the background to collect GPS data points. These data points were then plotted on a map generated on www.gpsvisualizer.com as well as a map generated by python using the "folium" library. Figure (4.1) shows a rectangular path around the lemon farm to test if the collected datapoints will actually present a rectangle in the plotted map. The result was a fairly rectangular path but it was shifted North by about a meter.

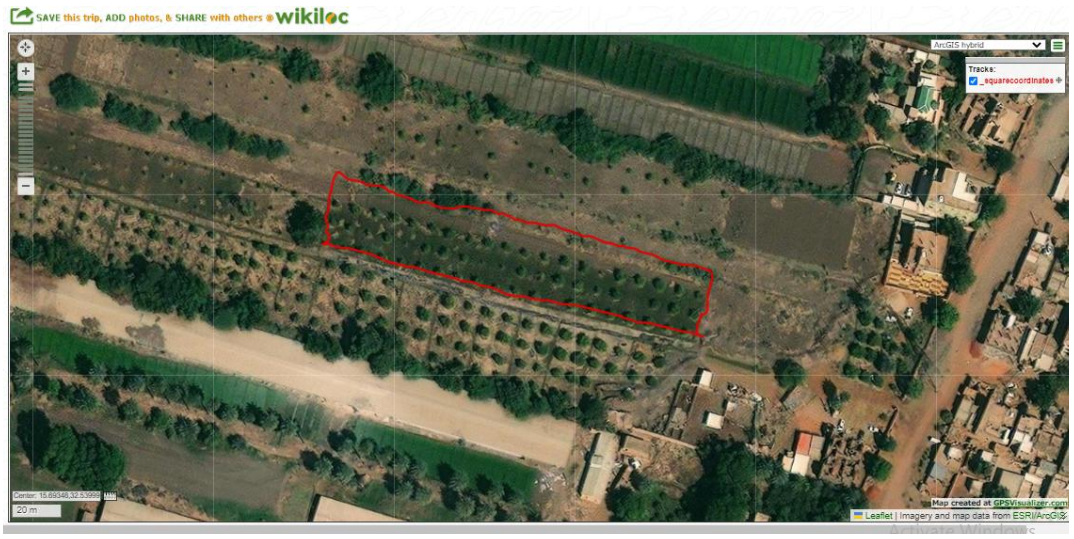

SAVE this trip, ADD photos, & SHARE with others
wikiloc
ArcGIS hybrid
Tracks:
_squarecoordinates
20 m
Map created at GPSVisualizer.com
Leaflet | Imagery and map data from ESRI/ArcGIS

Figure (4.1) shows a rectangular path around the lemon farm in Halfaia area

Figure (4.2) shows a test conducted at the University of Khartoum, college of agriculture. The datapoints were mostly at the correct coordinates but there were some points which went too far off of there actual locations. This was assumed to be due to the large trees covering the sky and buildings surrounding the area.

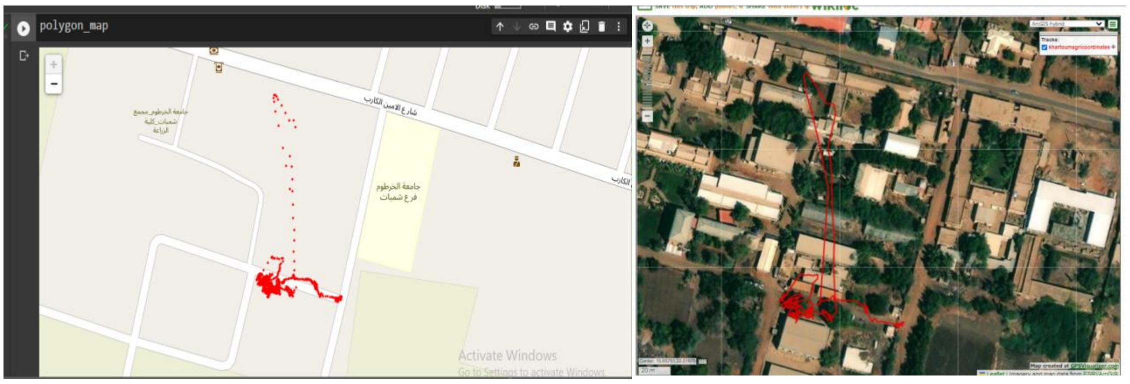


Figure (4.2) Accuracy test under trees and between some buildings

Figure (4.3) shows the locations recorded when capturing photos on the lemon farm. The data points were filtered to only show some points to properly view each individual point. The photos were taken of the first 4 trees only to review and analyse the results before proceeding on a larger area.

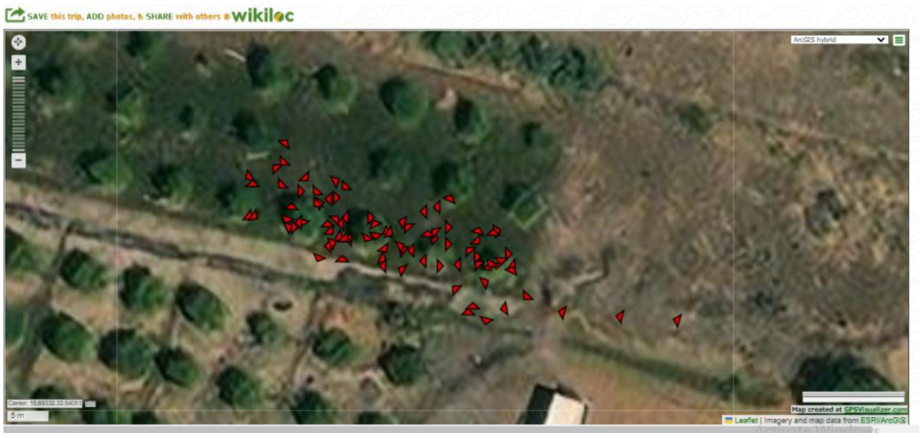

SAVE this trip, ADD photos, & SHARE with others wikiloc
ArcGIS hybrid
5 m
Map created at GPSVisualizer.com
Leaflet | Imagery and map data from ESRI/ArcGIS

Figure (4.3) Location points when capturing photos for the lemon farm

Figure (4.4) shows each and every location point plotted using python on Google Colab. The folium library provided an interactive map which can be zoomed and panned to view the neighboring streets which can help the user understand where exactly are the points plotted in relation to the city or area.

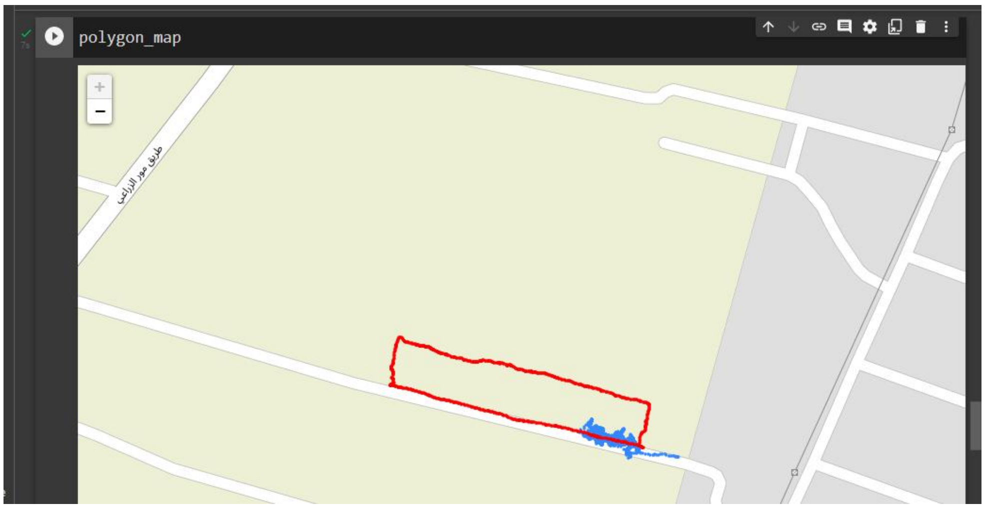


Figure (4.4) The map generated on google colab with all of the points color code

4.3 Plant disease recognition model

The model was trained and tested the first time on a dataset gathered by Rashik Rahman on Kaggle consisting of 1,532 images, 1,322 images for training, 150 images for validation and 60 images for testing. Figure 4.5 shows samples of the 3 classes the model was trained on as well as sample of photos taken of plants in the university by the raspberry pi and were named dt_test1.

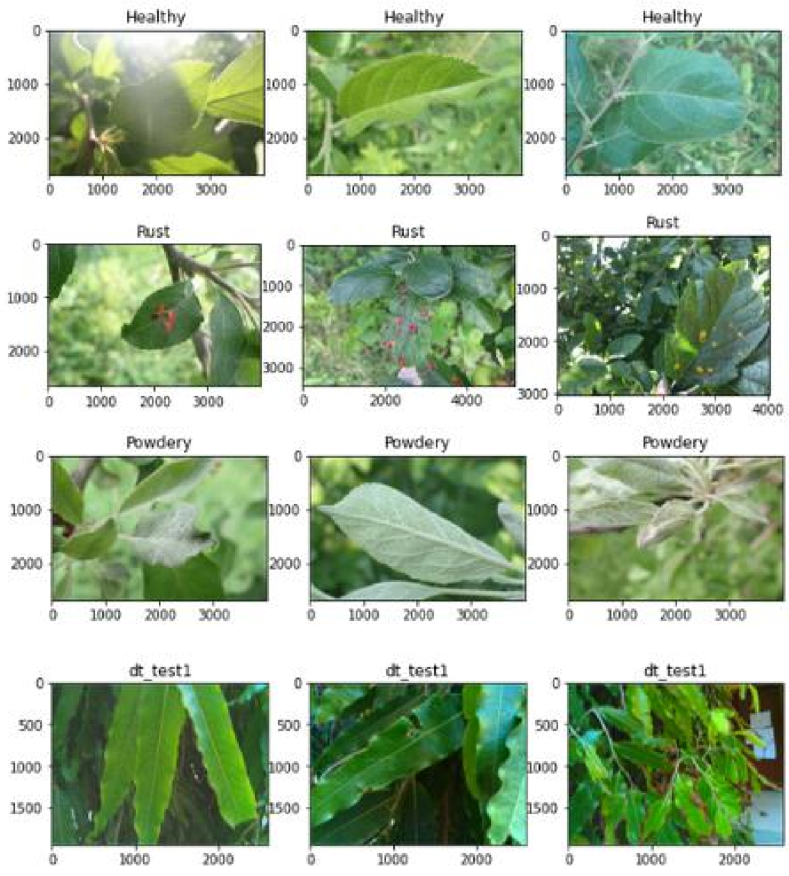
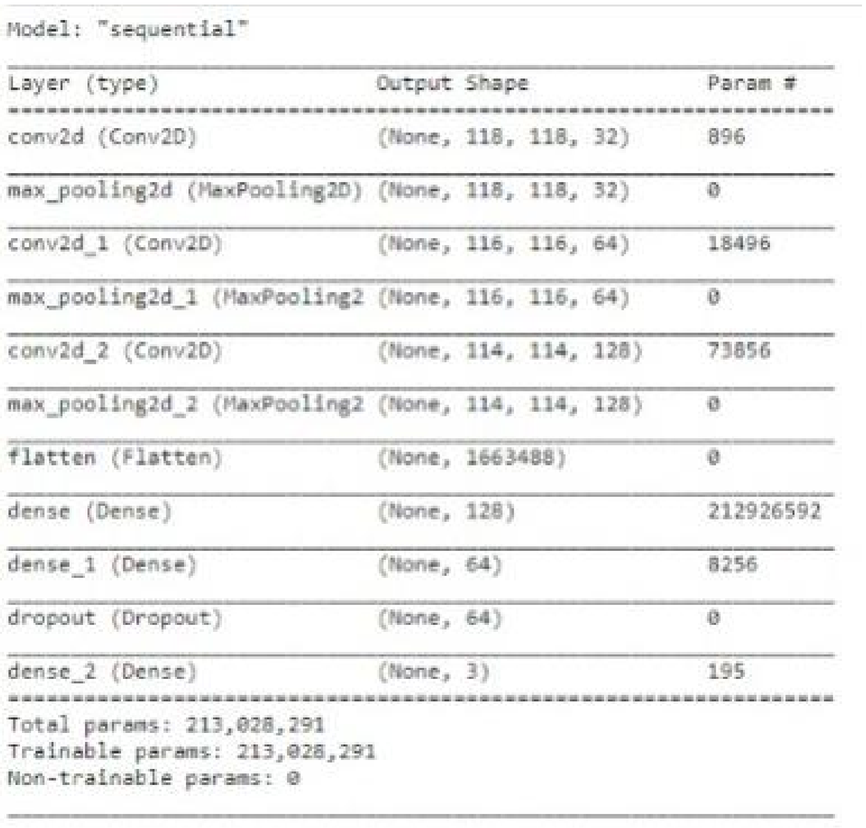


Figure (4.5) Samples of the pictures used

4.3.1 Model architecture

A sequential CNN model created by Rashik Rahman was modified slightly and used. The model was original designed to work with 80x80 pixels then modified to work with photos of 120x120 pixels in dimensions. Figure 5 shows the summary of the sequential model and figure 6 shows the model's architecture. A person can recreate the same results by compiling a model of the same architecture and loading the training checkpoint found in the output of the notebook at:
https://www.kaggle.com/code/abubakrbadrelmaarif/plant-disease-recognition-final/data?scriptVersionId=111656445



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 118, 118, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 118, 118, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 116, 116, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 116, 116, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 114, 114, 128)     73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 114, 114, 128)     0
_________________________________________________________________
flatten (Flatten)            (None, 1663488)           0
_________________________________________________________________
dense (Dense)                (None, 128)               212926592
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 195
=================================================================
Total params: 213,028,291
Trainable params: 213,028,291
Non-trainable params: 0
_________________________________________________________________
```

Figure (4.6) Model summary

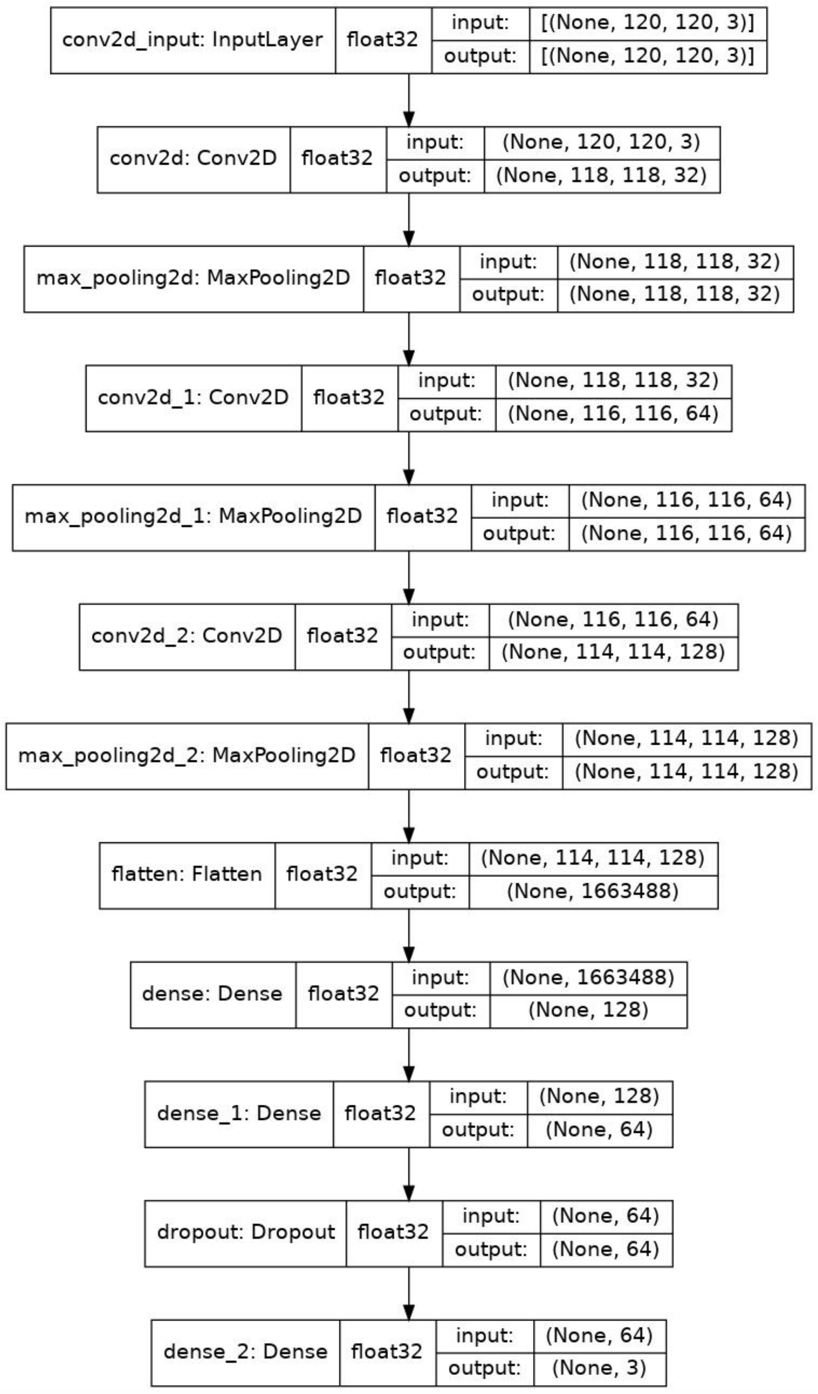
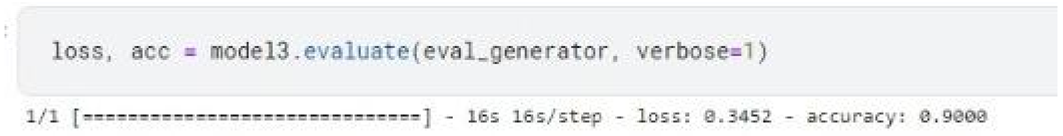

conv2d_input: InputLayer | float32 | input: [(None, 120, 120, 3)] | output: [(None, 120, 120, 3)]
conv2d: Conv2D | float32 | input: (None, 120, 120, 3) | output: (None, 118, 118, 32)
max_pooling2d: MaxPooling2D | float32 | input: (None, 118, 118, 32) | output: (None, 118, 118, 32)
conv2d_1: Conv2D | float32 | input: (None, 118, 118, 32) | output: (None, 116, 116, 64)
max_pooling2d_1: MaxPooling2D | float32 | input: (None, 116, 116, 64) | output: (None, 116, 116, 64)
conv2d_2: Conv2D | float32 | input: (None, 116, 116, 64) | output: (None, 114, 114, 128)
max_pooling2d_2: MaxPooling2D | float32 | input: (None, 114, 114, 128) | output: (None, 114, 114, 128)
flatten: Flatten | float32 | input: (None, 114, 114, 128) | output: (None, 1663488)
dense: Dense | float32 | input: (None, 1663488) | output: (None, 128)
dense_1: Dense | float32 | input: (None, 128) | output: (None, 64)
dropout: Dropout | float32 | input: (None, 64) | output: (None, 64)
dense_2: Dense | float32 | input: (None, 64) | output: (None, 3)

figure (4.7) The model's architecture

Figure (4.8) shows the test result on Rashik's dataset which resulted in 88.0% - 90.0% accuracy and about 0.34 loss.



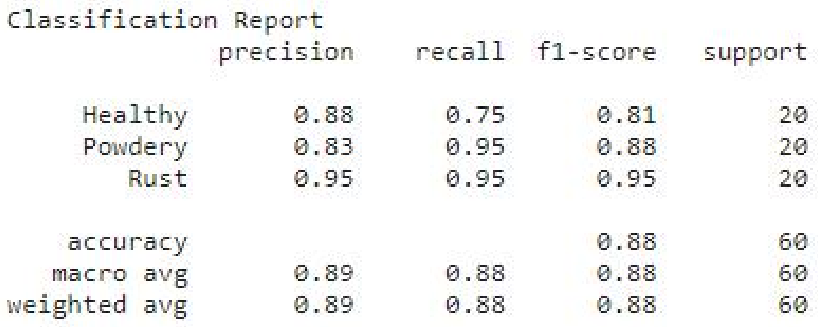
```
loss, acc = model3.evaluate(eval_generator, verbose=1)

1/1 [==============================] - 16s 16s/step - loss: 0.3452 - accuracy: 0.9000
```

Figure (4.8) Accuracy and loss metrics on test dataset

A classification report on the test dataset was generated using "sklearn" library. The report showed good values of precision, recall and f1-score but it can still be further improved if more training data was provided and the model was better refined.

Table (4.1) Classification report on the test dataset



```
Classification Report
              precision    recall  f1-score   support

     Healthy       0.88      0.75      0.81        20
     Powdery       0.83      0.95      0.88        20
        Rust       0.95      0.95      0.95        20

    accuracy                           0.88        60
   macro avg       0.89      0.88      0.88        60
weighted avg       0.89      0.88      0.88        60
```

The results of the prediction were then plotted on a heat map.

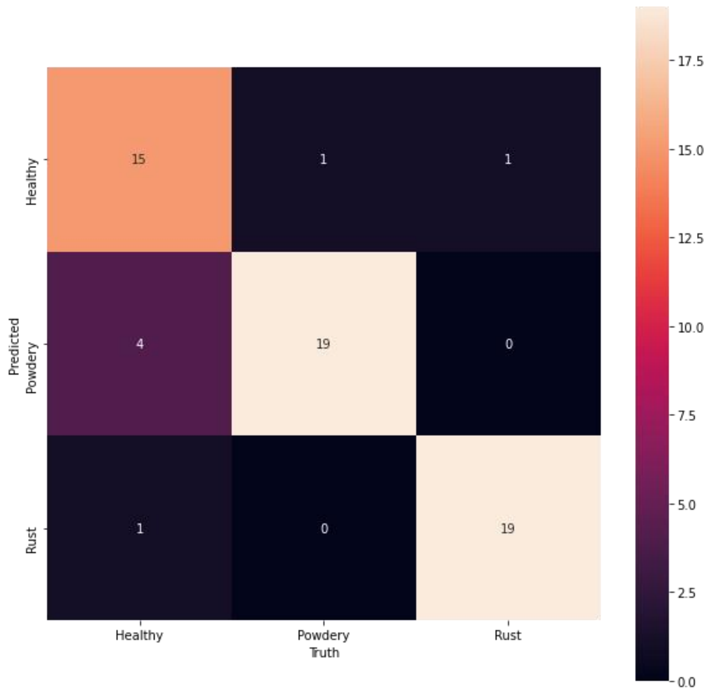


Figure (4.9) Heat map of the predictions of the model plotted against the actual values

The model was then tested on 84 images (dt_test1) of plants in Mashreq University and yielded the following results in figure 4.10 and appendix (1). The accuracy was observed to drop to about 54.7%. After cleaning the dataset (dt_test1) by removing all of the unclear and blurred images and thus creating another dataset named "dt_test2" consisting of 52 images. The model predictions on this dataset resulted in an improved accuracy of 75.0% . This result shows us that the input images to the model needs to be cleaned beforehand whether it is manually done or programmed in the preprocessing stage.

4.4 Drone stability

In the first test flight the drone was able to fly with a good stability without any fluctuations but it was slowly drifting with the wind. The first landing the drone crashed and the battery flipped out, this was due to the drone coming to land at an angle and not straight down. The battery was then reattached and a second test was immediately done. In the second test the drone flew smoothly and went as high as 16 meters and for a total time of about 4 minutes.

Figure (4.10) Drone test flights

Chapter 5
Conclusion and Recommendations

5.1 Conclusion:

The quadcopter was assembled and after calibration it was flight tested. The flight test showed good stability of the drone but the flight controller (APM 2.8) seemed to not respond well to flight mode changes. The photo capturing system and the GPS location tracking program were found to work very well except that the interface of the camera with the raspberry pi was observed to lag by a few frames. It was also concluded that the GPS operates at full capacity when the farm isn't surrounded by buildings or large trees. It was observed that the model needs clean data to achieve high accuracy ie. the pictures need to be filtered by removing blurry images and images focusing on unwanted subjects, like objects in the background. Overall the project can implemented on different farms successfully and achieve the desired results.

5.2 Recommendations:

As future research still some work needs to be done and this is mainly for machine learning model to further increase the accuracy. The more pictures that can be gathered and labeled to be fed to the model for training the better its ability to detect diseases. It is recommended to create a new dataset of about 1500 images and labeling them under supervision of a botanist and then training the model further on the dataset. Furthermore a good camera like a phone's camera should be used to capture image rather than the raspberry pi camera. Telemetry should be used in the case of unexpected loss of vision to the drone or use GSM module to send the GPS location via internet or SMS. During flight tests it advised to test the drone in controlled and safe environment to avoid crashes and accidents from loss of control of the drone.

References:

[1]Lakshmi Narayana Thalluri, Sai Divya Adapa, Priyanka D, Sri Nithya N, Yasmeen, Addepalli V S Y Narayana Sarma, Srikhakolanu Naga Venkat "Drone Technology Enabled Leaf Disease Detection and Analysis system for Agriculture Applications" Proceedings of the Second International Conference on Smart Electronics and Communication (ICOSEC).

[2] I.Gogul, V.Sathiesh Kumar "Flower Species Recognition System using Convolution Neural Networks and Transfer Learning" 2017 4th International Conference on Signal Processing, Communications and Networking (ICSCN -2017)

[3]. K. S. Amit, A. Rubiya, and R. B. Senthil, "Classification of Rice Disease using Digital Image Processing and SVM Classifier," International Journal of Electrical and Electronics Engineers (IJEEE), vol. 7, 2015.

[4] S. Phadikar, J. Sil, and A. K. Das, "Classification of Rice Leaf Diseases Based on Morphological Changes," International Journal of Information and Electronics Engineering, vol. 2, no. 3, 2012, pp. 460-463.

[5] S. Phadikar, and J. Sil, "Rice Disease Identification using Pattern Recognition Techniques," International Conference on Computer and Information Technology (ICCIT), 2008.

[6] R.P. Narmadha. and G. Arulvadivu, "Disease Symptoms Identification In Paddy Leaf Using Image Processing," Advances in Computational Sciences and Technology, ISSN 0973-6107, vol. 10, no. 8, 2017, pp.2213-2223.

[7] M. Reinecke and T. Prinsloo, "The influence of drone monitoring on crop health and harvest size," 2017 1st International Conference on Next Generation Computing Applications (NextComp), Mauritius, 2017, pp. 5-10

[8] Al-Hiary, Heba & Bani-Ahmad, Sulieman & Ryalat, Mohammad & Braik, Malik & Alrahamneh, Zainab. (2011). Fast and Accurate Detection and Classification of Plant Diseases. International Journal of Computer Applications. 17. 10.5120/2183-2754

[9] Gondal, Danish & Niaz, Yasir. (2015). Early Pest Detection from Crop using Image Processing and Computational Intelligence. FAST-NU Research Journal ISSN: 2313-7045.

APPENDIX:

Raspberry Pi programs

GPS_working_file1.py:

```
from serial import Serial
from pynmeagps import NMEAReader

try:
   while True:
       stream = Serial('/dev/ttyAMA0', 9600, timeout=3) #stream data from port ttyAMA0 which is
RX at baudrate 9600
       nmr = NMEAReader(stream) #read NMEA format
       (raw_data, parsed_data) = nmr.read() #extract raw data and parsed data

       raw = str(raw_data) #convert to string data type
       parsed = str(parsed_data) #convert to string data type

       if raw[0:8]=="b'$GNGGA": #read only $GNGGA sentences
          print(parsed[6:-2])

except KeyboardInterrupt:
   Exit
```

GPS_working_file2.py:

```
from serial import Serial
from pynmeagps import NMEAReader
import csv
import random


gps_list=[]

try:
    while True:
        stream = Serial('/dev/ttyAMA0', 9600, timeout=3) #stream data from port ttyAMA0 which is
RX at baudrate 9600
        nmr = NMEAReader(stream) #read NMEA format
        (raw_data, parsed_data) = nmr.read() #extract raw data and parsed data

        raw = str(raw_data) #convert to string data type
        parsed = str(parsed_data) #convert to string data type

        if raw[0:8]=="b'$GNGGA": #read only $GNGGA sentences
            gps_list.append(parsed[6:-2].split(", "))

            with open('datafarm.csv', 'w') as csv_file:
                writer = csv.writer(csv_file)

                writer.writerow(gps_list)


except KeyboardInterrupt:
    #df = pd.dataframe(gps_list)
    #print(df)
    exit
```