Sudan University of Science and Technology

College of Engineering

School of Electronic Engineering

Enhancing The Efficiency of Agriculture by using Image Processing and Drones

A Research Submitted in Partial fulfillment for the requirements of the Degree of B.Eng. (Honors) in Electronic Engineering

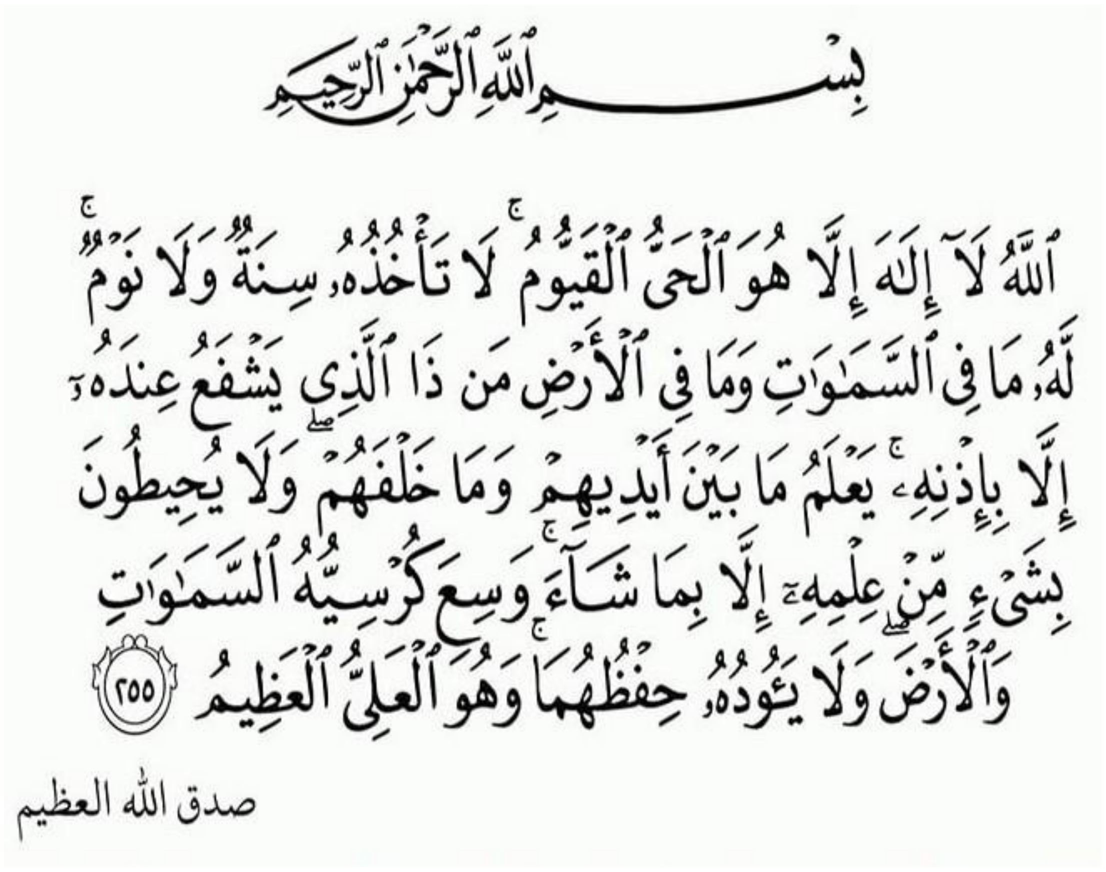
Prepared By:

1- Abubakr Mahdi Abdallah
2- Esraa Alnour Zakaria
3- Hiba Al-Daer Ibrahim
4- Motaz Khogali Alsaied

Supervisor By:
Dr. Hisham Ahmed Ali Ahmed

March 2022

الإستهلال



بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

ٱللَّهُ لَآ إِلَٰهَ إِلَّا هُوَ ٱلْحَيُّ ٱلْقَيُّومُ ۚ لَا تَأْخُذُهُۥ سِنَةٌ وَلَا نَوْمٌ ۚ لَّهُۥ مَا فِى ٱلسَّمَٰوَٰتِ وَمَا فِى ٱلْأَرْضِ ۗ مَن ذَا ٱلَّذِى يَشْفَعُ عِندَهُۥٓ إِلَّا بِإِذْنِهِۦ ۚ يَعْلَمُ مَا بَيْنَ أَيْدِيهِمْ وَمَا خَلْفَهُمْ ۖ وَلَا يُحِيطُونَ بِشَىْءٍ مِّنْ عِلْمِهِۦٓ إِلَّا بِمَا شَآءَ ۚ وَسِعَ كُرْسِيُّهُ ٱلسَّمَٰوَٰتِ وَٱلْأَرْضَ ۖ وَلَا يَـُٔودُهُۥ حِفْظُهُمَا ۚ وَهُوَ ٱلْعَلِىُّ ٱلْعَظِيمُ ﴿٢٥٥﴾

صدق الله العظيم

DEDICATION

First of all, we would like to dedicate this unpretentious effort

To our **Almighty GOD** who gave us strength and knowledge.

To **Our Parents** Who have an endless presence and for the never-endinglove and encouragement.

To **Our brothers and sisters** Who sustained us in our life and still.

To **Our teachers** Who lighted candles in our ways and provided us with thelight of knowledge.

And of course, to our beloved supervisor Dr. Hisham Ahmed Who was the perfect mentor for us.

Finally, our best friends and Our Classmates

Researchers…

ACKNOWLEDGMENTS

Thanking Allah before and after.

First and foremost; the greatest thanking to our teachers for their continuous support... and for their great efforts, they were the best guide and monitor...

Finally; thank our colleagues and workers at the College of Engineering for their cooperation...

ABSTRACT

Agriculture is an important factor for the development of any country, in addition to providing foodstuffs, agriculture is a primary source of raw materials that are used in several industries, a term known as smart agriculture has recently appeared where technology and modern techniques are used to better plan and manage crops. The research focuses on discovering one of the most dangerous cotton diseases, angular spot disease. which is also known as bacterial blight. It can cause production losses of up to 10% of the crop. The drone is used to take pictures of the agricultural field. It is a mechanical vehicle with four arms, and in each arm, a motor is connected to a propeller. Two of the propellers rotate clockwise, while the other two spin counterclockwise. Based on the images taken by the drone, the technique of filtering neural networks is convolutional Neural Network (CNN), which is used to detect the disease by performing operations on the images. This research contributed to helping to enhance the efficiency of cotton production by classifying the images taken by the drones into healthy images or bacterial blight diseases, training a CNN model using the data set that contains images of diseased and healthy cotton and we obtained an accuracy of 97.2% and thus is successfully classify the cotton images into diseased and healthy and return them to a map showing disease prevalence.

المستخلص

الزراعة تُعدّ عاملاً مهمّاً لتطوّر أيّ بلد، إلى جانب توفير المواد الغذائية تُعتبر الزراعة مصدراً أساسياً للمواد الخام التي تدخل في عدّة صناعات،ظهر مؤخر مصطلح يعرف بالزراعة الذكية حيث يتم استخدام التكنولوجيا والتقنيات الحديثة لتخطيط وإدارة المحاصيل بشكل أفضل . و يركز البحث على اكتشاف أحد أخطر أمراض القطن وهو مرض التبقع الزاوي. ويعرف أيضاً بإسم مرض اللفحة البكتيرية . ويمكن أن يسبب خسائر في الإنتاج تصل إلى 10% من المحصول. تم استخدام الدرون لأخذ صور للحقل الزراعي وهي عباره عن مركبة ميكانيكية لها أربعة أذرع وفي كل ذراع محرك متصل به مروحة. يدور اثنان من المراوح في اتجاه عقارب الساعة، بينما يدور الآخران في الاتجاه المعاكس لعقارب الساعة. استناداً على الصور المأخوذة بواسطة الدرون استخدمت تقنية الشبكات العصبية التلافيفية (سي أن أن) في اكتشاف المرض وذلك بإجراء عمليات على الصور. هذا البحث ساهم في المساعدة في تعزيز كفاءة إنتاج محصول القطن وذلك بتصنيف الصور التي التقطتها الطائرات بدون طيار إلى صور سليمة او مرض باللفحة البكتريا, تدريب نموذج سي إن إن باستخدام مجموعة البيانات التي تحتوي على صور القطن مريضة وصحية وحصلنا على دقة قدرها 97.2٪ وبذلك تم تصنيف صور القطن إلى مريض وسليم بنجاح وإعادتها إلى خريطة توضح انتشار المرض.

Table of Contents

List of Tables

TABLE NO **TITLE** **PAGE**

List of Figures

List of Symbols

symbols	**meaning**
a	lift slope
A	propeller disk area
A_c	fuselage area
A_u	Operational time (autonomy)
b	thrust facto
B_w	propulsion group bandwidth
c	propeller chord
C	propulsion group cost factor
C_{bat}	battery capacity
C_d	drag coefficient at 70% radial station
C_H	hub force coefficient
C_Q	drag coefficient
C_{Rm}	rolling moment coefficient
C_T	thrust coefficient
d	drag factor
g	acceleration due to gravity
h	vertical distance: Propeller center to Cog
H	hub force
i	motor current
$I_{xx,yy,zz}$	inertia moments
J_m	motor inertia
J_r	total rotor inertia is seen by the motor
k_e	motor electrical constant
k_m	motor torque constant
l	horizontal distance: propeller center to Cog
m	overall mass
m_{af}	airframe mass
m_{av}	avionics mass
$m_{bat_{av}}$	avionics' battery mass
m_{hel}	helicopter mass
m_{pg}	propulsion group mass
$M_{BAT_{max}}$	maximum battery mass possible
$M_{MAX_{possible}}$	maximum mass one motor can lift
$M_{MAX_{requested}}$	requested mass for one motor to lift
n	number of propellers
P_{av}	avionics' power consumption

P_{el}	electrical power
P_{in}	gearbox input power
P_{out}	gearbox output power
Q	drag moment
Q_{pg}	propulsion group quality factor
Q_{in}	design quality index
r	gearbox reduction ratio
R	rotation matrix
R_{rad}	rotor radius
R_{mot}	motor internal resistance
R_m	rolling moment
T	thrust force
T_w	propulsion group thrust/weight ratio
u	motor input
U	control inputs
V	body linear speed
x, y, z	position in body coordinate frame
X, Y, Z	position in earth coordinate frame
β	thrust/weight ratio
ζ	position vector
η	gearbox efficiency
η_m	motor efficiency
θ	pitch angle
θ_0	pitch of incidence
θ_{tw}	twist pitch
λ	inflow ratio
μ	rotor advance ratio
ν	speed vector
ρ	air density
Q_{equ}	percentage of time in equilibrium
σ	solidity ratio
τ	motor time-constant
τ_a	torque in body coordinate frame
τ_d	motor load
τ_m	motor torque
υ	induced velocity
Φ	roll angle
ψ	yaw angle
ω	body angular rate
ω_m	motor angular rate
Ω	propeller angular rate
Ω_r	overall residual propeller angular speeds

List of Abbreviations

Abbreviations	**Meaning**
AC	alternating current
ANNs	artificial neural networks
APM	Ardupilot Mega
BEC	Battery Eliminator Circuit
CCW	counterclockwise
CH	channel
CNN	Convolutional Neural Network
COM port	communication port
CPU	central processing unit
CSS	Cascading Style Sheets
CW	clockwise
DC	direct current
DNNs	deep neural networks
EMF	Electromotive Force
ESC	Electronic Speed Controllers
FETs	Field-effect transistors
FPV	first-person view
GND	Ground
GPS	Global Positioning System
GPU	Graphics Processing Unit
HTML	Hypertext Markup Language
Hz	hertz
ICSP	In-Circuit Serial Programming
IDE	integrated development environment
IMU	Inertial Measurement Unit
IOU	Intersection over Union
IP	Internet Protocol
LiPo	Lithium Polymer
MHz	megahertz
ML	Machine Learning
MLPs	multilayer perceptron’s
Ni-Cd	nickel–cadmium battery
NiMH	nickel metal hydride battery
NMS	Non-Maximum Suppression
NN	Neural network
PCB	printed circuit board
PDB	power distribution board
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
R-CNN	Region-Based Convolutional Neural Network
RPM	Revolutions per minute

RPN	Region Proposal Network
SDKs	Software Development Kit
SSD	Single Shot Detector
TF	TensorFlow
TL	Transfer Learning
UAV	unmanned aerial vehicle
USB	universal serial bus
VCC	Common Collector Voltage
YOLO	You Only Look Once

CHAPTER ONE
INTRODUCTION

1.1 Preface

Cotton is one of the most important crops grown in Sudan, as it is one of the most important industrial export crops. Industrially, cotton is used in the textile industry, while cottonseeds are an important source of oils and their residues are used to make feed.

Sudanese cotton has high quality and great historical fame, and to preserve the crop and increase its productivity, it must be protected from diseases and pests.

Bacterial blight is one of the most common diseases affecting cotton. Bacterial blight with angular, moist, waxy spots with a red to brown border on leaves, stems, and roses. Bacterial blight is one of the most dangerous bacteria found on cotton and causes losses and reduces the productivity of fields. Now, with the spread of technology in agriculture, the development of agricultural methods, the use of drones, and the frequent use of them in our daily life after it was a monopoly of military facilities, we see widespread use of this technology in agriculture.

The use of this technology in agriculture can lead to development in the agricultural field and reduce the spread of diseases that can be combated with this modern technology, which contributes to increasing crop productivity.

1.2 Problem statement

Bacterial blight is one of the most dangerous diseases that affect the cotton crop its infestation leads to a decrease in the product as a result of the affected leaves falling, the rotting of the nut, and the low degree of cotton as a result of its discoloration and discoloration.

1.3 Proposed solution

By use drones to take pictures from the field and analyze them later using digital image processing to detect the disease and collect enough information about it such as the extent of the disease and the identification of the affected area**.**

1.4 Research Aim and Objectives

Aim:

Design a drone for cotton disease diagnosis using image processing and machine learning.

Objectives of this work was to:

- Reduce the losses in cotton productivity caused by bacterial blight to the least possible and save time and effort.
- Develop agriculture using drones and new technologies.
- Develop an algorithm to perform image processing on crop digital images.

1.5 Methodology

The project consists of two parts, software, and hardware.

In the hardware part, the drone is designed and connect the components with the Ardupilot and connect the camera with the Arduino.

The software part, consists of two parts, a part for programming the drones, and the other part for machine learning and image processing, in which data is collected, and model training.

1.6 Project Layout

This project consists of five chapters:

Chapter One gives an introduction about the principles of the project, motivation.

Chapter Two discusses the background of computer vision, history of computer vision, machine learning, the components of drones, drone movement, and the software used.

The other section is related work

Chapter Three describes the system block diagram, hardware design for drones, and Software design for drone and machine learning.

Chapter Four discusses the results of simulation and implementation for the project.

Chapter Five explains the conclusion and the future ideas that can be performed.

CHAPTER TWO
BACKGROUND AND RELATED WORK

2.1 Background

2.1.1 Computer Vision

Computer vision is a field of artificial intelligence that trains computers to interpret and understand the visual world. Using digital images from cameras and videos and deep learning models, machines can accurately identify and classify objects — and then react to what they "see." [1]

2.1.2 Machine Learning (ML)

2.1.2.1 Machine Learning Overview

Machine Learning (ML) recently had breakthroughs in so many diverse areas due to the explosion in the availability of data, significant improvements in ML techniques, and advancement in computing capabilities. Machine learning is concerned with constructing computer programs that improve automatically with experience.

In traditional programming inputs and programs are given to the computer to find the output, but in ML the computer has the inputs and outputs, and the task is to find the relationship between them or to learn how it can reach the output given so many examples.

A computer program is said to learn from experience E concerning some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E[2, 3] For example in the cat dog classification problem, the task T is to tell whether the image is for a dog or a cat, the performance measure P is the percent of images correctly classified, and the training experience E is a database of images with given classifications. When the ML sees more images, it gets more experience, if the performance measure improves with more experience, that means the model is learning.

2.1.2.2 Machine Learning Components

ML algorithm components are a model, a dataset, an optimization algorithm, and a cost function. Most ML algorithms involve optimization of some sort. Optimization refers to the task of either minimizing or maximizing some function f(x) by altering x. Most optimization problems are usually phrased in terms of minimizing f(x). In that case, f(x) is called the loss function or the cost function. The derivative is useful in the task of minimizing a function f(x). We can reduce f(x) by moving x in small steps with the opposite sign of the derivative.

This technique is called gradient descent. Gradient descent is an iterative algorithm that starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function. The following figure shows an illustration of how this technique can minimize a function; we take f(x) = $\frac{1}{2}x^2$ as an example.

For a cost function f(x) depending on a parameter x, the optimization is performed by updating x using this formula: $x^{\circ} = x - (\tau * \frac{d(f(x))}{dx})$, where x° is the new value of the parameter x, and τ is the learning rate, which is a tuning parameter in the optimization algorithm that determines the step size at each iteration while moving towards a minimum of the loss function. When the training dataset is labeled (each example has a label or a target value) this is called Supervised Learning, and it’s called Unsupervised Learning when the dataset is unlabeled. The most common form of ML is supervised learning.[4]

Figure 2-1: shows the Gradient Descent.

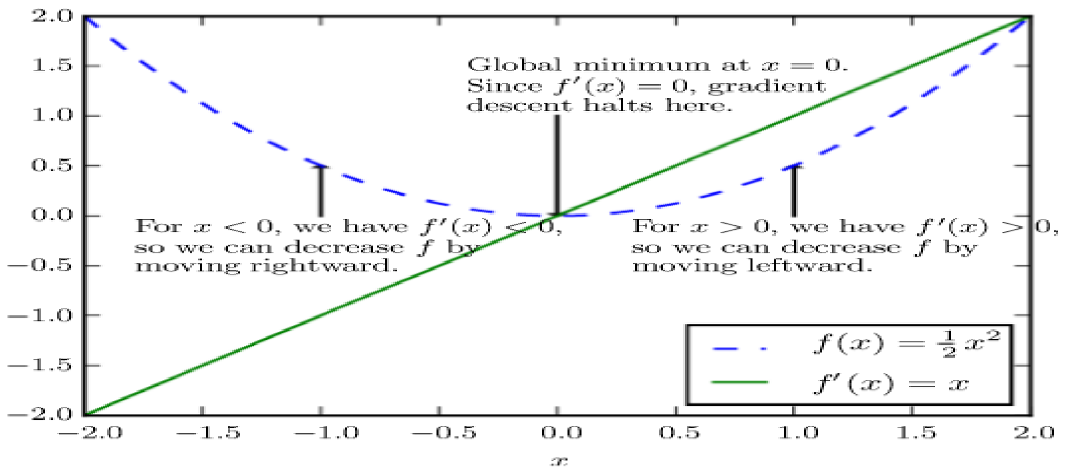


Figure 2-1: Gradient Descent

2.1.2.3 Unsupervised Learning

Unsupervised Learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset only from the experienced features, no supervision signal is given. The distinction between supervised and unsupervised learning is not formally and rigidly defined because there is no objective test to determine whether a value is a feature or a target provided by a supervisor. But informally unsupervised learning refers to most attempts to extract information from a distribution that does not require human labor to annotate examples.[5] Density estimation, clustering data into groups of related examples, learning to draw samples from a distribution, or denoise data from some distribution are all examples of unsupervised learning problems. K-means clustering is one of the simplest and most popular unsupervised machine learning algorithms. The objective of the k-means clustering merely is grouping similar data points together and discovering underlying patterns, which can be achieved by looking for a fixed number (k) of clusters in a dataset. A cluster refers to a collection of data points aggregated together because of certain similarities. The k-means algorithm identifies the k number of centroids (centers of the clusters). It then allocates every data point to the nearest cluster while keeping the centroids as small as possible.

2.1.2.4 Supervised Learning

Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future cases[6]. Supervised Learning algorithms experience a dataset containing features, but each example is associated with a label or a target. So given a training set of examples of inputs x and outputs y, the goal of supervised learning is to predict the right output y from data x that is not known before, based on earlier known data. Formally, the goal is to approximate the mapping function from inputs to outputs y = f(x). Supervised Learning can be divided into two categories of problems, classification, and regression. Both problems have as a goal the construction of a model that can predict the value of the output y from the input x value; the difference is the fact that the output y value is numerical for regression and categorical for classification. The goal in classification is to take an input vector x and to assign it to one of K discrete classes $c_k$where $k = 1,2, \dots . k$In the most common scenario, the classes are taken to be disjoint, so that each input is assigned to one and only one class. The input space is thereby divided into decision regions whose boundaries are called decision boundaries or decision surfaces. When there are only two categories, it’s called binary classification, an example of this is the cat-dog classification problem. The goal of regression is to predict the value of one or more continuous target variables t given the value of a D-dimensional vector x of input variables, and polynomial curve fitting is an example[7].

2.1.2.5 Machine Learning Metrics

Various metrics are proposed to evaluate ML model performance in different applications. Looking to a single metric may be tricky and may not give you the whole picture of the problem, then a group of metrics will be needed to have a concrete evaluation of the model. Rather than accuracy, performance in ML is measured using other concepts, precision, and recall.

$$\text{Accuracy}=\frac{TP+TN}{TP+TN+FP+FN}=\frac{\text{true predicted data}}{\text{total data}} \tag{1}$$

$$precision=\frac{TP}{TP+FP}=\frac{true\ predicted\ data\ labelled\ Y\ es}{total\ data\ predicted\ Y\ es} \tag{2}$$

$$\text{recall}=\frac{TP}{TP+TN}=\frac{true\ predicted\ data\ labelled\ Y\ es}{total\ data\ predicted\ truly} \tag{3}$$

Where:
TP: Actual class is YES (1), and the prediction is YES (1).
TN: Actual class is NO (0), and the prediction is NO (0).
FP: Actual class is NO (0), and the prediction is YES (1).
FN: Actual class is YES (1), and the prediction is NO (0).

2.1.3 Neural Networks (NN)

2.1.3.1 Neural Network Overview

Neural network (NN) learning methods provide a robust approach to approximate real-valued, discrete-valued, and vector-valued target functions. Neural networks are among the most effective learning methods currently known for certain types of problems, such as learning to interpret complex real-world sensor data [1]. Neural networks have generated a lot of excitement in ML research and industry, which results in breakthroughs in computer vision, speech recognition, and text processing.

A standard neural network consists of many simple connected processors called neurons, each producing a sequence of real-valued activations. Input neurons get activated through sensors perceiving the environment; other neurons get activated through weighted connections from previously active neurons. NN learning is about finding weights that make the NN exhibit desired behavior. Depending on the problem and how the neurons are connected, such behavior may require long causal chains of computational stages, where each stage transforms (often in a non-linear way) the aggregate activation of the network[8].

The study of artificial neural networks (ANNs) has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons. In a rough analogy, artificial neural networks are built out of a densely interconnected set of simple units, where each unit takes several real-valued inputs (possibly the outputs of other units) and produces a single real-valued output, which may become the input to many other units[2].

Figure 2-2 shows the biological neuron and the equivalent ANN.

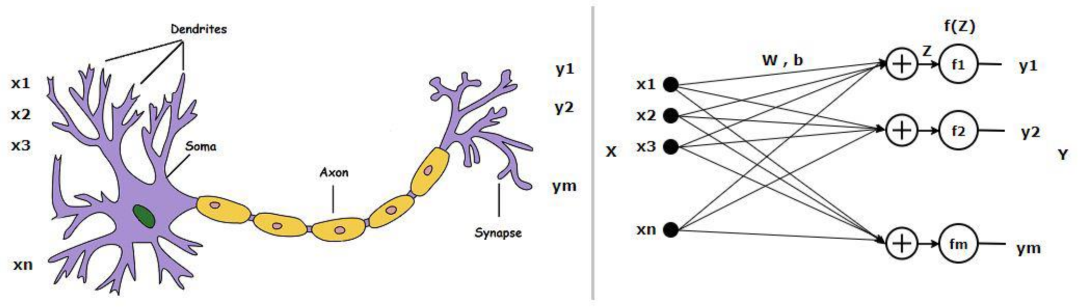


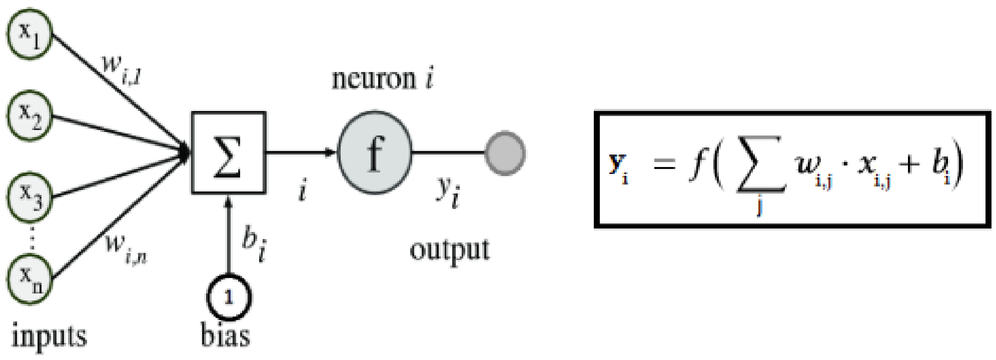
Figure 2-2: Biological neuron and its Equivalent ANN

2.1.3.2 Structure of Neural Network

The basic unit of computation in NN is the neuron, which is sometimes called a node or unit. The neuron receives input from an external source or another neuron and computes an output. Each input to the neuron has an associated weight w assigned based on the relative importance of this input. Additionally, there is another input 1 with weight b called the bias. The neuron computes the weighted sum of its inputs and then applies an activation function f to the result.

A neural network is formed in layers; each layer consists of multiple neurons. A neural network layer could be an input layer, an output layer, or a hidden layer. Input neurons receive information from an external source and pass them to hidden neurons. No computation is performed in input neurons. Output neurons are responsible for computations and transferring outputs from the network to the outside world. Hidden neurons have no direct connection with the outside world. They perform computations and transfer information from previous neurons to the later ones.

Figure 2-3: shows the computation of one-layer NN output.



$$\boxed{y_i = f\left(\sum_j w_{i,j} \cdot x_{i,j} + b_i\right)}$$

Figure 2-3: Computation of one-layer NN output

They are called hidden layers because the training data does not show the desired output for each of these layers, Instead, the learning algorithm must decide how to use these layers to the best implementation of an approximation of the mapping function[5].

Figure 2-4: shows the multiple layers NN

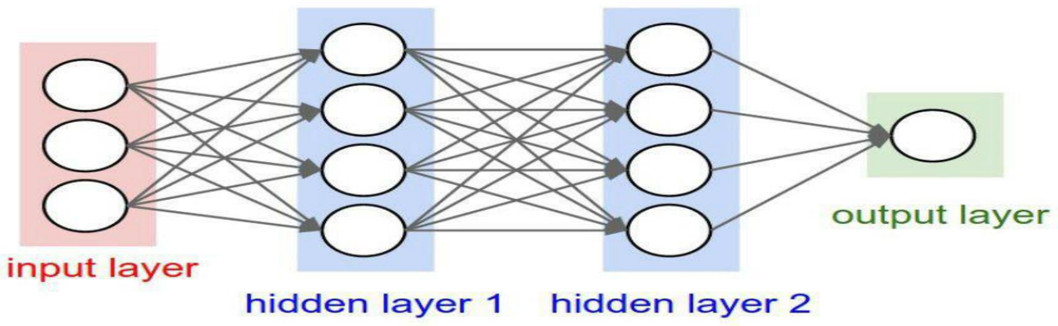


Figure 2-4: Multiple Layers NN

2.1.3.3 Activation Function

The Activation function adds non-linearity to neuron output, which is important because most of the real-world data is non-linear, and we want neurons to learn this non-linear behavior. The most popular activation functions used in neural networks are sigmoid, tanh, ReLU, and Leaky ReLU. In multi-class classification tasks in NN, the SoftMax function is used in the final layer (the output layer).

2.2 Deep Learning

2.2.1.1 Deep Learning Overview

A neural network that has only one hidden layer is called a shallow neural network, and those with more hidden layers are called deep neural networks (DNNs). From here, the term deep learning exists. Modern deep learning provides a compelling framework for supervised learning. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity. Most tasks that consist of mapping an input vector to an output vector, and that are easy for a person to do rapidly, can be accomplished via deep learning, given sufficiently large models and sufficiently large datasets of labeled training examples[5]. Figure 2-5 shows the structure of the DNN.

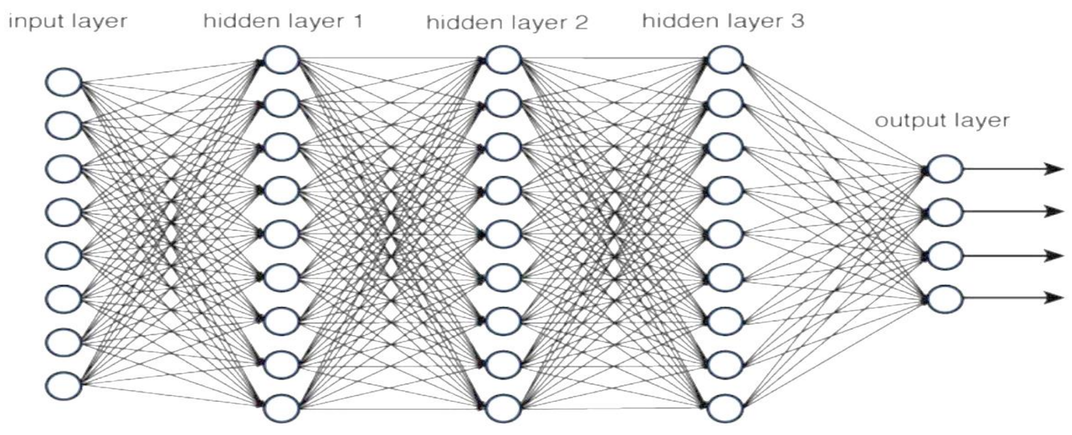


Figure 2-5: DNN Structure

2.2.1.2 Feedforward Neural Networks

Feedforward neural networks, or multilayer perceptron's (MLPs), are the quintessential deep learning models. The goal of a feedforward network is to approximate some function f_. For example, for a classifier, y = f_(x) maps an input x to a category y. A feedforward network defines a mapping y = f (x; W, b) and learns the value of the parameters W and b that result in the best function approximation.

These models are called feedforward because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself [3]. These feedback connections exist in Recurrent Neural networks.

2.2.2 Convolutional Neural Networks (CNNs)

2.2.2.1 CNN Overview

The Convolutional Neural Network (CNN) is one of the most notable deep learning approaches where multiple layers are trained robustly. It has been found highly effective and is also the most commonly used in diverse computer vision applications[9].

Dense layer or fully connected layer learns global patterns in their input feature space, but convolutional layers learn local patterns. Convolution learns different features from the input image, so the output of a convolutional layer is called a feature map. Feature maps have two spatial axes (height and width) as well as a depth axis (also called the channels axis). Properties of CNN[10]:

• Learned patterns by CNN are translation-invariant.

• They can learn spatial hierarchies of patterns.

• the Reduced number of parameters.

2.2.2.2 Convolutional Layers

Generally, CNN consists of three main neural layers, which are convolutional, pooling, and fully connected. The convolutional layer consists of various kernels or filters, of the same size. These filters are applied to (convolved with) the input image and the intermediate feature maps. In CNN's the filter or the kernel is mostly a 3D tensor of size k x k x c, where k is an integer and is usually a small number, like 3 or 5. And c is the number of channels of the input image or feature map. For an input image I with size w x h x c, and a filter K with size k x k x c, the output feature map S is evaluated by:

$$s(i,j) = (I * k)(i * j) = \sum_q \sum_m \sum_n I\,(m,n,q)K(i-m, j-n, q) \qquad (4)$$

The output size will be $(w-k+1)x(h-k+1)\,x\,n$, where n is the number of applied filters. This convolutional layer has $k.k.c.n+n$ parameters. The $k.k.c.n$ parameters are for the filter’s weights, and the n parameters are for the biases, one with each filter.

Figure 2-6: shows the operation for one filter.

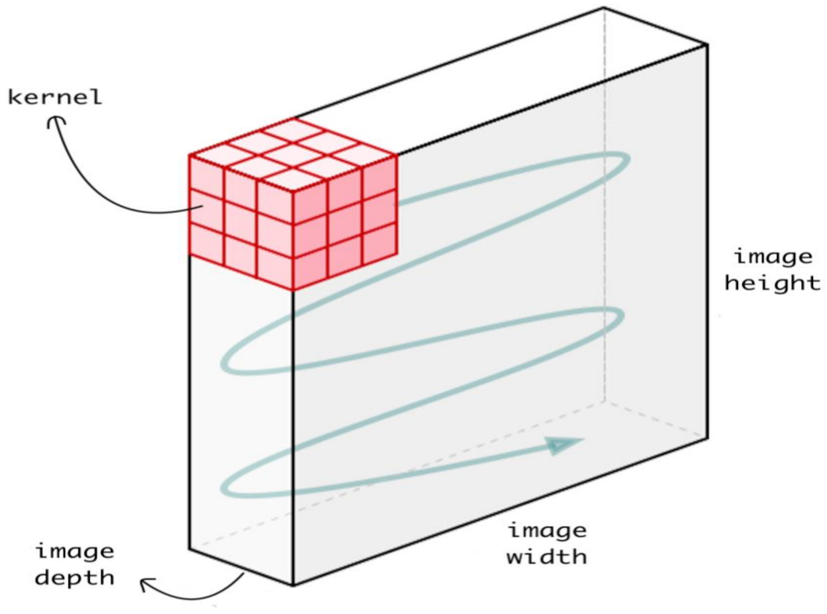


Figure 2-6: 3D convolution

The Convolutional layer has four hyperparameters, the size of the filter, the number of filters, the stride, and the padding. The stride is the number of rows or columns the filter is shifted by at each step. When using a stride s, the output size will be:

$$\left[\frac{w-k}{s}+1\right] X \left[\frac{h-k}{s}+1\right] x\, n \qquad (5)$$

The padding is to add a border to the input. Padding is applied to keep information in borders and to avoid dimension reduction. When padding is used input width and height are increased by 2p where p is the border size. The most commonly used padding is Zero Padding.

2.2.2.3 Pooling Layers

The pooling layer follows the convolutional layer and is used to down sample feature maps to reduce network parameters. Like convolutional layers, pooling layers are translation-invariant because their computations take neighboring pixels into account. Max pooling and average pooling are the most commonly used strategies. Pooling layers have no parameters to learn, so they are fixed functions. For w x h x c input, and a p x p pooling with stride s, the output size will be:

$$\left[\frac{w-p}{s}+1\right] X \left[\frac{h-p}{s}+1\right] X\, c \qquad (6)]$$

Pooling does not affect the number of channels because it is applied to each channel of the input independently. Feature maps resulting from convolution and pooling layers are flattened to a 1D feature vector, to get into fully connected layers that perform like a traditional neural network.

Figure 2-7: shows the CNN poling layers.

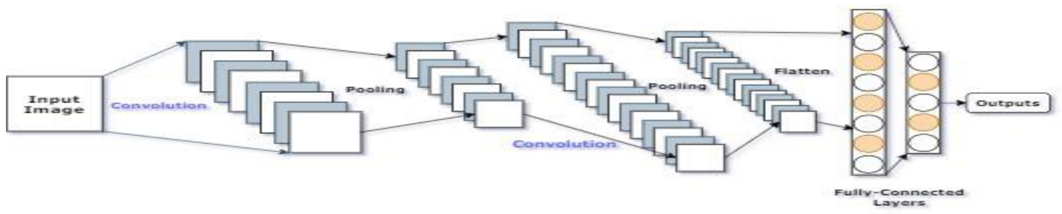


Figure 2-7: CNN

2.2.2.4 CNN Overfitting

Since there are a large number of parameters in deep NN, the network tends to overfit the training data, which was discussed in section (2.2).

In CNN, we can avoid overfitting by applying:

1-Batch normalization: which normalizes the mean and variance of the output activations from a CNN layer to follow a unit Gaussian distribution. It proves to be very useful for the efficient training of a deep network because it reduces the internal covariance shift of the layer activations[2].

2-DropOut: refers to dropping out units in a neural network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections. The choice of which units to drop is random[11].

3-Data Augmentation: It’s a very effective way of enhancing the generalization power of CNN models. Especially for cases where the number of training examples is relatively low. It is often utilized to generate additional data without introducing extra labeling costs by performing operations like scaling, cropping, rotating, horizontal reflections, or altering the RGB values of the training images. Besides overfitting, CNNs also face vanishing gradients of earlier layers, with the network depth increasing. This is solved by adding a skipped path or a shortcut to the main path in the network, which is called a residual connection[12].

2.2.2.5 Transfer Learning

For a model to get sufficient accuracy, it requires a lot of training data and computational power. One way to short-cut this process is to re-use weights from pre-trained models that were developed for another task or dataset where the input is the same, but the target may be different. This process is called Transfer Learning (TL). The re-used weights may be used for specific layers as the starting point for the training process. Transfer learning decreases the training time for a neural network model and can also reduce the generalization error. The reason behind using weights from trained models is that when a network is trained on images, the first layers tend to learn general features (i.e., edges, corners, and lines) which occur regardless of the dataset used. Transfer learning may be followed by a fine-tuning process to train the network well on the new dataset.

2.2.3 GPU

Graphics Processing Unit (GPU) is a processor in graphics cards, similar to a computer's CPU. GPU is designed specifically for performing complex mathematical and geometric calculations. Some of the fastest GPUs have more transistors than the average CPU. GPUs can process data several orders of magnitude faster than a CPU, due to the massive parallelism. So, they are best suited for repetitive and highly-parallel computing tasks, which makes the GPU the appropriate device in deep learning tasks (GPU is considered as the heart of Deep Learning).

2.2.4 TensorFlow (TF)

TF is an open-source software library for numerical computation using data flow graphs, initially developed by Google Brain Team, to conduct machine learning and deep neural networks research. TensorFlow provides an extensive set of functions and classes that simplify building advanced models from scratch.

Generally, TensorFlow can be thought of as a programming system in which computations are represented as graphs. Nodes in the graph represent mathematical operations, and the edges represent multidimensional data arrays (tensors) communicated between them. Some of the properties of TensorFlow are:

1. TF provides an accessible and readable syntax.
2. TF provides more flexibility. Thus, new functionalities can be defined.
3. TF offers great debugging tools.
4. Scalability and Pipelining.

2.2.5 RCNN Family

The R-CNN or Region-Based Convolutional Neural Network object detection approach uses the selective search algorithm[13]. to generate category-independent region proposals that define the set of candidate detections and then feed them into a detector composed of a feature extractor and a classifier[14]. R-CNN had many improvements that reduced its running time, which introduced Fast RCNN[15]. and then Faster R-CNN[16].

Faster R-CNN introduced Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. The output of the RPN predicts k proposals at each location, called anchors. Anchors cover different scales and aspect ratios. Using RPNs enables a unified, deep-learning-based object detection system that runs at near-real-time frame rates with competitive accuracy.

2.2.6 You Only Look Once (YOLO)

In the YOLO method, there is a single convolutional network, which simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO system divides the input image into an SS grid; each grid cell is responsible for detecting the object having its center in it. YOLO trains on full images and directly optimizes detection performance. This unified model is extremely fast since YOLO frames detection as a regression problem. Unlike sliding window and region-proposal-based techniques, YOLO sees the entire image during training and test time, so it implicitly encodes contextual information about classes as well as their appearance. YOLO makes a smaller number of background errors compared to other methods because they can't see the border context. Since YOLO is highly generalizable, it is less likely to break down when applied to new domains or unexpected inputs[17].

2.3 Drone Hardware

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 onboard sensors and a global positioning system (GPS).

The drone has a small, powerful computer (Ardupilot Mega) responsible for controlling other components of the drone:

2.3.1.1 Frame

The frame is the structure that holds all the components together. One of the most important parts of the quadcopter is its frame because it supports motors and other electronics and prevents them from vibrations. You have to be very precise while making it. They need to be designed to be strong but also lightweight.

Figure 2-8 shows: the quadcopter frame

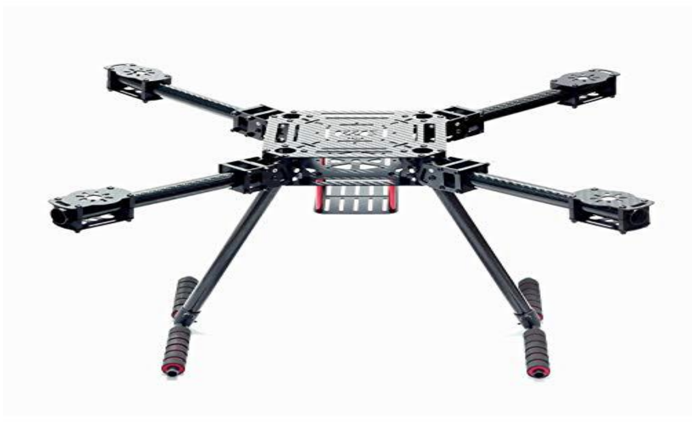


Figure 2-8: Quadcopter frame

2.3.1.2 The Microcontroller – Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. "Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index of boards.

Figure 2-9 Shows: Arduino Uno [18]

Figure 2-9: Arduino UNO Microcontroller Board

2.3.1.3 Electronic Speed Controllers (ESC)

An electronic speed control or ESC is a circuit with the purpose to control an electric motor's speed, its direction, and possibly also to act as a dynamic brake in some cases. ESCs are often used on electrically powered brushless motors essentially providing an electronically generated three-phase electric power, with a low voltage source.

An ESC interprets control information in a way that varies the switching rate of a network of field-effect transistors (FETs), not as mechanical motion as would be the case of a servo. The quick switching of the transistors is what causes the motor itself to emanate its characteristic high-pitched whine, which is especially noticeable at lower speeds. It also allows much smoother and more precise variation of motor speeds in a far more efficient manner than the mechanical type with a resistive coil and moving arm once in common use.

The ESC generally accepts a nominal 50 Hz Pulse Width Modulation (PWM) servo input signal whose pulse width varies from 1ms to 2ms. When supplied with a 1ms width pulse at 50 Hz, the ESC responds by turning off the DC motor attached to its output. A 1.5ms pulse-width input signal results in a 50% duty cycle output signal that drives the motor at approximately 50% speed. When presented with a 2.0ms input signal, the motor runs at full speed due to the 100% duty cycle (on constantly) output.

The correct phase varies with the motor rotation, controlled and monitored by the ESC. The orientation of the motor is determined by the back EMF (Electromotive Force). The back EMF is the voltage induced in a motor wire by the magnet spinning past its internal coils. Finally, a PID algorithm in the controller adjusts the PWM to maintain a constant RPM.

Reversing the motor's direction may also be accomplished by switching any two of the three leads from the ESC to the motor.

Ideally, the ESC controller should be paired to the motor and rotorcraft with the, Figure 2-10 Shows: ESC, following considerations:

1. Temperature and thermal characteristics.
2. Max Current output and Impendence.
3. Needs to be Equipped with a BEC (Battery Eliminator Circuit) to eliminate the need for a second battery.
4. Size and Weight properties.
5. Magnet Rating [19]

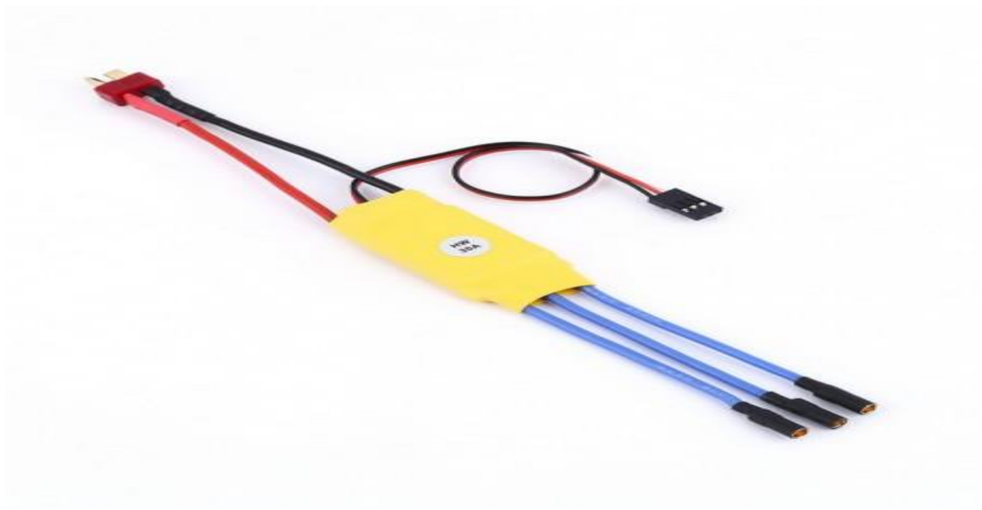


Figure 2-10:30A Brushless ESC

Additionally, the speed controller has fixed throttle settings so that the "stop" and "full throttle" points of all the various modes can be cut through cleanly. The controller produces audible beeps to assist in navigating through the program modes and troubleshooting logs.

2.3.1.4 The Battery Pack

Lithium Polymer (LiPo) cells are one of the newest and most revolutionary battery cells Available. LiPo cells maintain a more consistent voltage over the discharge curve when compared to Ni-Cd or NiMH cells. The higher nominal voltage of a single LiPo cell (3.7V vs. 1.2V for a typically Ni-Cd or NiMH cell); making it possible to have an equivalent or even higher total nominal voltage in a much smaller package LiPo cells typically offer very high capacity for their weight, delivering upwards of twice the capacity for ½ the weight of comparable NiMH cells, Figure 2-11: Shows LiPo Battery.

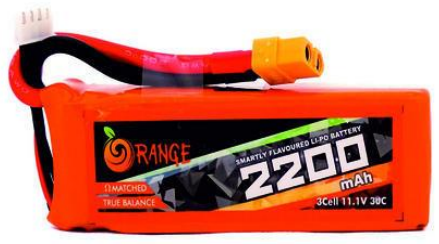


Figure 2-11:3S LiPo Battery

2.3.1.5 The Brushless Motors

The drone has four Brushless DC motors attached to a propeller. The Brushless motor differs from the conventional Brushed DC Motors in their concept essentially in that the commutation of the input voltage applied to the armature's circuit is done electronically, whereas in the latter, by a mechanical brush. As with any rotating mechanical device, it suffers wear during operation, and as a consequence, it has a shorter nominal lifetime than the newer Brushless motors.

Despite the extra complexity in its electronic switching circuit, the brushless design offers several advantages over its counterpart, to name a few: higher torque/weight ratio, less operational noise, longer lifetime, less generation of electromagnetic interference, and much more power per volume. Virtually limited only by its inherent heat generation, whose transfer to the outer environment usually occurs by conduction, Figure 2-12: Shows brushless motor [20]

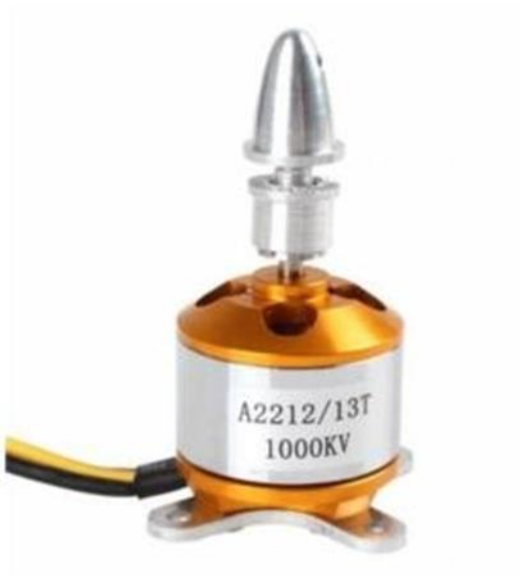


Figure 2-12: A2212/13T 1000 KV BLDC (Brushless DC Motor)

2.3.1.6 power distribution board (PDB)

PDB’s essentially distributed the power from the battery to the drone ESC. But the technology has improved so much in recent days that PDBs also distribute power to some other peripherals such as FPV Video Transmitters, FPV Cameras, and the Quadcopter Flight Controller itself. Some modern FCs have integrated PDBs, are limited by space, and can only accommodate so much that they do not do a very good job at filtering the voltage spikes from the insane current draws from our quads [21] The Figure 2-13: shows the power distribution board.

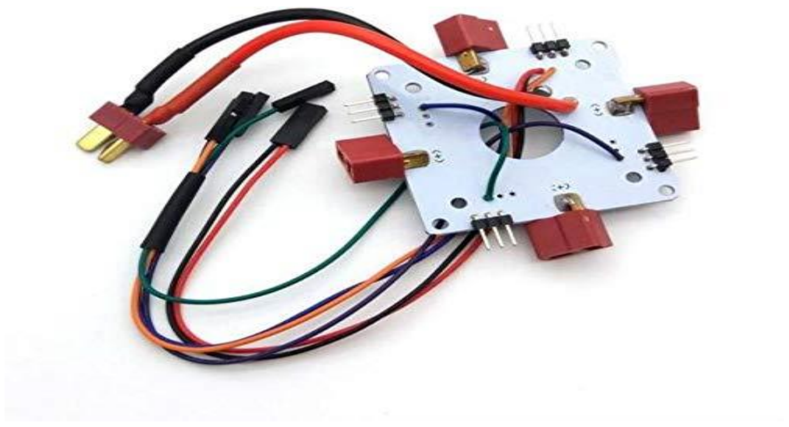


Figure 2-13:Power distribution board

2.3.1.7 Propellers

A propeller is a set of rotating blades designed to convert the power (torque) of the Engine into thrust.

The Quadrotor consists of four propellers coupled to the brushless motor. Among These four propellers, two are clockwise and the remaining two are counterclockwise. Clockwise and anticlockwise propellers cancel their torque from each other.

Propellers are specified by their diameter and pitch. The propeller used is 1045 Fixed-pitch, symmetric, tapered Normal Rotation Carbon Fiber Propeller.

figure 2-14: shows the carbon fiber propeller

Figure 2-14: 10x4.5 fixed-pitch, Carbon fiber Propeller

2.3.1.8 FLY SKY FS CT6B

The 2.4 GHz antenna is pretty standard fare, folding 90 degrees (via a 45-degree intermediate step) for storage, Figure 2-15 Shows: the remote control [22]

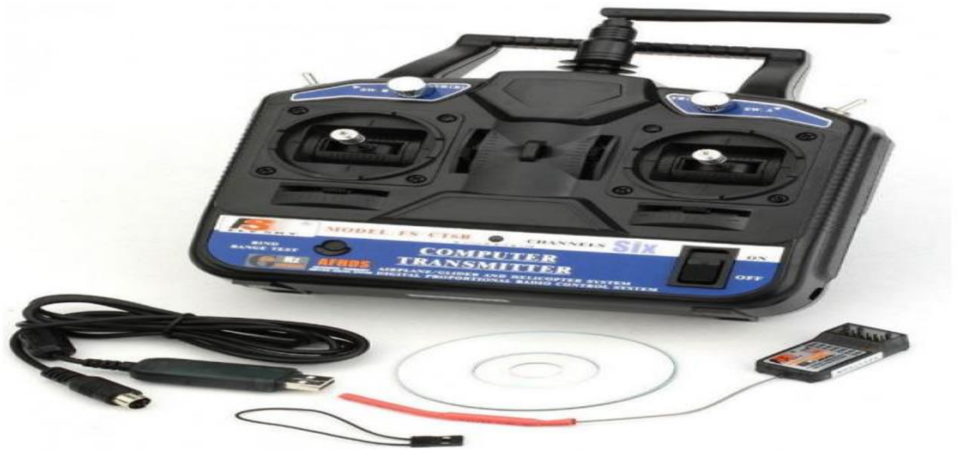


Figure 2-15:sky remote control

2.3.1.9 NEO-8N APM GPS

NEO-M8 modules utilize concurrent reception of up to three GNSS systems (GPS/Galileo together with Bei Dou or GLONASS), recognize multiple constellations simultaneously, and provide outstanding positioning accuracy in scenarios where urban canyon or weak signals are involved. For even better and faster positioning improvement, the NEO-M8 series supports the augmentation of QZSS, GAGAN, and IMES together with WAAS, EGNOS, and MSAS. The NEO-M8 series also supports message integrity protection, geofencing, and spoofing detection with configurable interface settings to easily fit customer applications.

The NEO-M8M is optimized for cost-sensitive applications, while NEO-M8N and NEO-M8Q provide the best performance. The future-proof NEO-M8N and NEO-M8J include an internal flash that allows future firmware updates. This makes NEO-M8N and NEO-M8J perfectly suited to industrial and automotive applications.

Figure2-16: show that the GPS module [23]

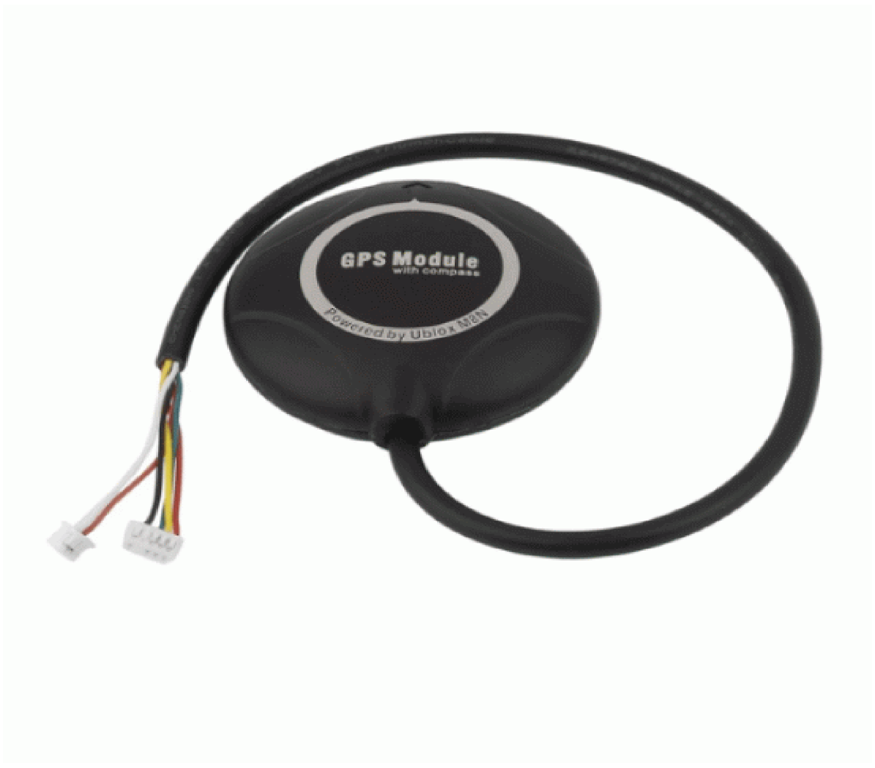


Figure 2-16: GPS module

2.3.1.10 Ardupilot Mega

(APM) is a professional quality IMU autopilot that is based on the Arduino Mega platform. This autopilot can control fixed-wing aircraft, multi-rotor helicopters, as well as traditional helicopters. It is a full autopilot capable of autonomous stabilization, way-point-based navigation, and two-way telemetry with Xbee wireless modules. Supporting 8 RC channels with 4 serial ports. ArduPilot Mega consists of the main processor board (the red one above) and the IMU shield which fits above or below it.

Figure 2-17 shows the Ardupilot [24]

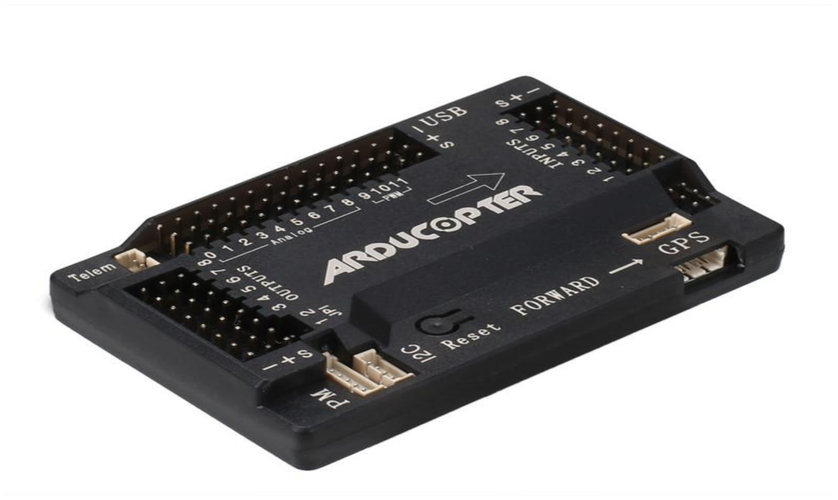


Figure 2-17: Ardupilot Mega

2.3.1.11 ESP32-CAM

ESP32-CAM is a low-cost development board with a WIFI camera. It allows creating IP camera projects for video streaming with different resolutions. ESP32-CAM has a built-in PCB antenna.

The figure 2-18:below shows the ESP32 Camera [25]

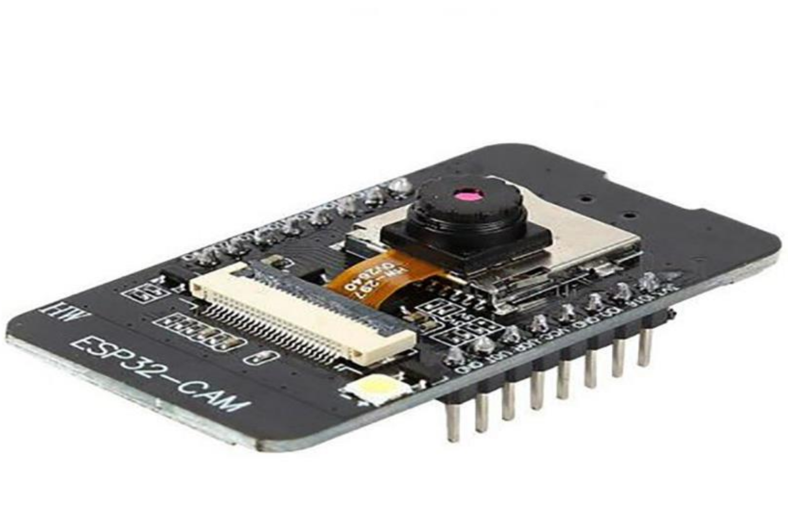


Figure 2-18: ESP32 Camera

2.4 Software tools

2.4.1 Arduino IDE

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board [26].

2.4.2 Mission Planner

Mission Planner is a ground control station for Plane, Copter, and Rover. It is compatible with Windows only. Mission Planner can be used as a configuration utility or as a dynamic control supplement for your autonomous vehicle [27].

2.4.3 PyCharm IDE

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python programming language. It is developed by the Czech company JetBrains It provides code analysis, a graphical debugger, an integrated unit tester [28].

2.5 Drone Movement and Control

2.5.1 Drone Movement

The basic idea of the movement of the Drone is shown in the following figure. It can be seen from the figure that the Drone is simple in mechanical design compared to helicopters. Movement in the horizontal frame is achieved by tilting the platform whereas vertical movement is achieved by changing the total thrust of the motors. But Drone arise certain difficulties with the control design [29].

Figure 2-19: shows the drone movement.

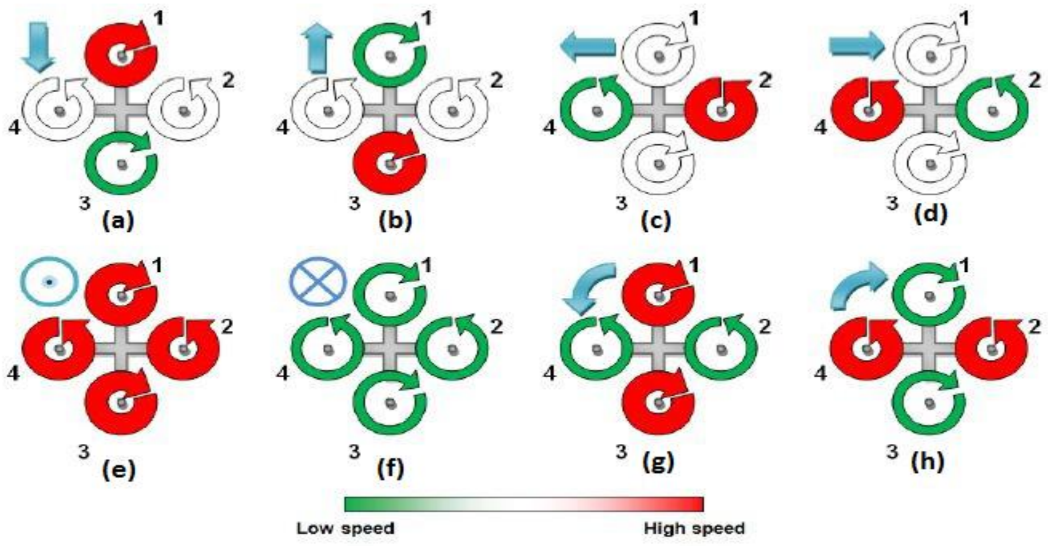


Figure 2-19: Basic Flight movements of a Quadcopter

2.5.2 Drone structure

The quadcopter has six degrees of freedom. This means that six variables are needed to express its position and orientation in space (x, y, z, φ, θ and ψ). The x, y, and z variables represent the distances of the quadcopter's center of mass along the x, y, and z axes respectively from a fixed reference frame. The other three variables are the three Euler angles which represent the quadcopter orientation. (φ) is the angle about the x-axis and is called roll angle, (θ) is the angle about the y-axis and is called pitch angle and (ψ) is the angle about the z-axis and is called yaw angle, Figure 2-20 Shows: quadcopter axis.

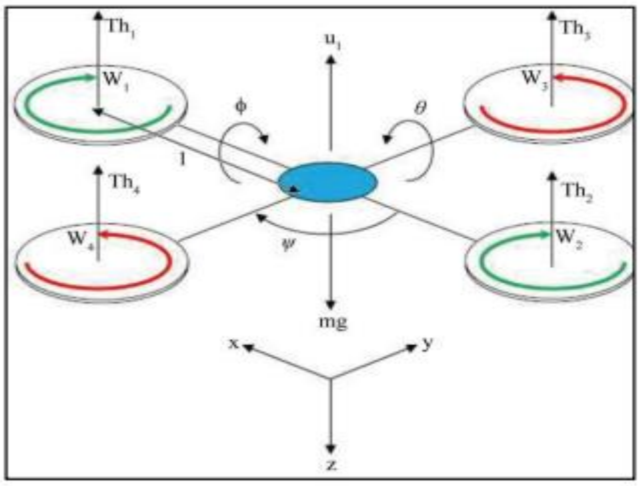
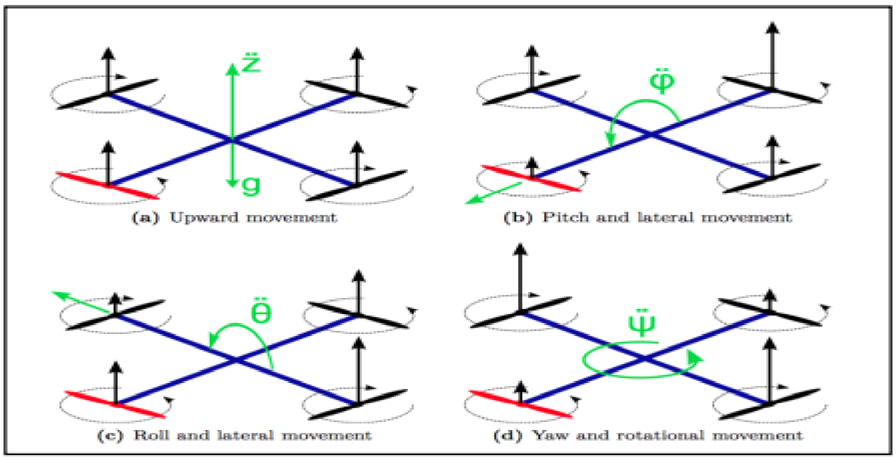


Figure 2-20: Quadcopter Axis

2.5.3 Drone mechanism

Each rotor produces both lift and torque about its center of rotation, as well as drag opposite to the vehicle's direction of flight. Lift is the force that directly opposes the weight of a quadcopter and holds the drone in the air. Lift is generated by every part of the quadcopter, but most of the lift on a normal airliner is generated by drone motors. Quadcopters generally have two rotors spinning clockwise (CW) and two counterclockwise (CCW). The spinning of the quadcopter propeller blades pushes air down. All forces come in pairs (Newton's Third Law), which means for every action force there is an equal in size and opposite in direction reaction force. Therefore, as the rotor pushes down on the air, the air pushes up on the rotor.

Figure 2-21: shows the quadcopter mechanism.



$\ddot{z}$
g
(a) Upward movement
$\ddot{\varphi}$
(b) Pitch and lateral movement
$\ddot{\theta}$
(c) Roll and lateral movement
$\ddot{\psi}$
(d) Yaw and rotational movement

Figure 2-21: Quadcopter Mechanism

2.5.4 Equations of Motion

1- Rolling Moments:
 -equation explains the Body gyro effect
 $\theta^{\cdot}\psi^{\cdot}(Iyy - Izz)$

- equation (2.2) explains the rolling moment due to forward flight $(-1)^{i+1}\sum_{i=1}^{4} R_{mxi}$
- equation (2.3) explains the propeller gyro effect $Jr\theta^{\cdot}\Omega r$
- hub moment due to sideward flight $h\sum_{i=1}^{4} H_{yi}$

- roll actuators action l$(-T_2 + T_4)$

2- Pitching Moments:

- Body gyro effect $\varphi^{\cdot}\psi^{\cdot}(Izz - Ixx)$
- rolling moment due to forward flight $h(\sum_{i=1}^{4} H_{xi})$
- propeller gyro effect $Jr\varphi^{\cdot}\Omega r$
- hub moment due to sideward flight $(-1)^{i+1}\sum_{i=1}^{4} R_{myi}$
- pitch actuators action $l(T_1 - T_3)$

3- Yawing Moments:

- body gyro effect $\theta^{\cdot}\varphi^{\cdot}(Ixx - Iyy)$
- hub force unbalance in forward flight $\boldsymbol{l(H_{x2} - H_{x4})}$
- inertial counter-torque $Jr\Omega^{\cdot}\, r$
- hub force unbalance in sideward flight $\boldsymbol{l(-H_{y1} + H_{y3})}$
- counter-torque unbalance $(-1)^{i}\sum_{i=1}^{4} Q_i$ [30]

2.5.5 Drone maneuverability

Drone rotate and move along the three principal axes; one running forward to back, another running right to left, and one running up and down. By rotating or tilting along these different axes, an aircraft can move forward or backward, left or right, or simply rotate in place.

Pitch: is the backward and forward movement of the drone. To pitch up the front two propellers, the RPM increased by receiving a signal from the control device to increase lift while decreasing lift of the two rear props and reducing lift, causing the nose to lift up while the quadcopter maintains altitude. To pitch down, the front propeller's speed is being sped down to reduce lift and speed on the rear propellers to increase the lift.

Roll: is the right or left movement of the drone. To roll right RPM keeps increasing on the left propellers and decreasing on the right ones, and the same opposite function when the drone trying to roll left.

Yaw: is simply defined as the rotation left or right of a quadcopter concerning the center axis, yaw refers to the movement of the drone clockwise or counterclockwise. A quadcopter's yaw is controlled by manipulating the reactive torque effect that each propeller has on frame-the very same effect that must be canceled by control surfaces in airplanes and helicopters.

Figure 2-22: shows the Drone maneuverability.

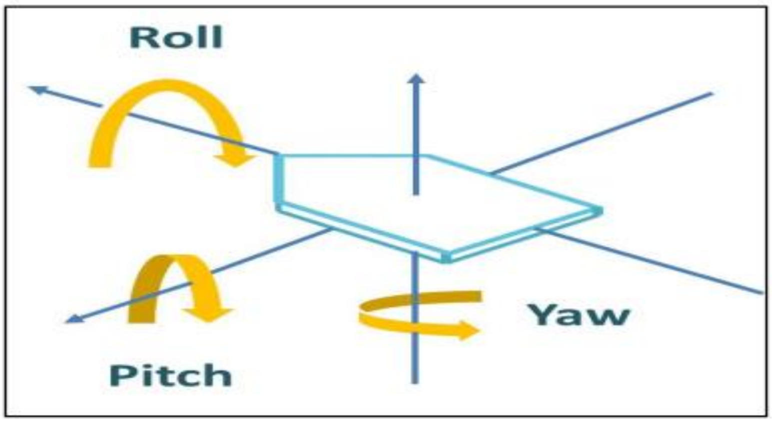


Figure 2-22: Quadcopter Roll, Pitch, and Yaw

2.5.6 Proportional Integral Derivative (PID)

A PID controller is a feedback control algorithm widely used in industrial control systems. A PID controller calculates the error value which is the difference between a measured process variable and the desired setpoint, the controller attempts to minimize the error by adjusting the process. It's part of a flight controller software that reads the data from sensors and calculates how fast the motors should spin to retain the desired rotation speed of the aircraft. The goal of the PID controller is to correct the error, the difference between a measured value (gyro sensor measurement), and the desired setpoint (the desired rotation speed). The error can be minimized by adjusting the control inputs in every loop, which is the speed of the motors.

The figure 2-23: shows the PID block diagram.

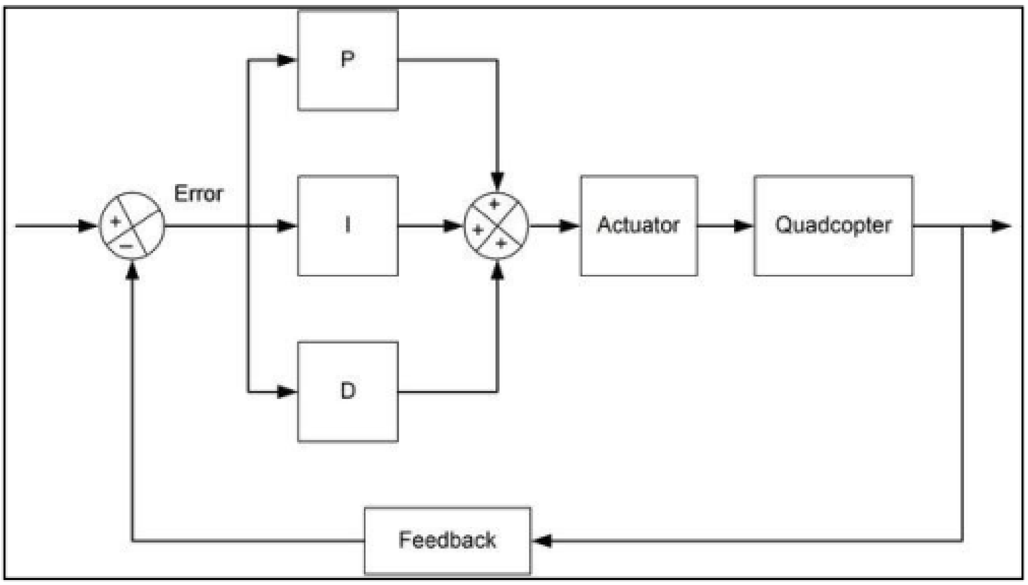


Figure 2-23: Proportional Integral Derivative (PID) Block Diagram.

There are three gains values in a PID controller, they are the ‘P’ term, ‘I’ the term, and ‘D’ term:

“P Gain” looks at the present error – the further it is from the set-point, the harder it pushes.

“D Gain” is a prediction of future errors, it looks at how fast you are approaching a set-point and counteracts P when it is getting close to minimize overshoot

“I Gain” is the accumulation of past errors, it looks at forces that happen over time; for example, if a quad constantly drifts away from a set-point due to wind, it will spool up motors to counteract it[31]

2.6 Related work

This set of experiments was designed to understand if the neural network actually learns the “notion” of plant diseases, or if it is just learning the inherent biases in the dataset. Using the deep convolutional neural network architecture, we trained a model on images of plant leaves to classify both crop species and the presence and identity of disease on images that the model had not seen before. Within the Plant Village data set of 54,306 images containing 38 classes of 14 crop species and 26 diseases (or absence thereof), this goal has been achieved as demonstrated by the top accuracy of 99.35%. Thus, without any feature engineering, the model correctly classifies crop and disease from 38 possible classes in 993 out of 1000 images. Importantly, while the training of the model takes a lot of time (multiple hours on a high-performance GPU cluster computer), the classification itself is very fast (less than a second on a CPU), and can thus easily be implemented on a smartphone. This presents a clear path toward smartphone-assisted crop disease diagnosis on a massive global scale [3]

This paper introduced the basic general principles used in designing a quadcopter UAV, which has wide applicability in modern agriculture. Quadcopter-collected color images are excellent for providing a gross picture of general field health and problems. The use of drones for surveillance of greenhouses can increase crop yields by minimizing the cost of traveling on very large areas and remediation the issues identified. Also, drones coupled with smart sensors can be developed to be an effective tool for the future.

The main goal is to achieve a stable flight of the quadcopter, and this is done using a linear PID control strategy.

The attitude and altitude stabilization system are composed of PIDs that are responsible for keeping the quadcopter in a desired angular state while keeping its altitude The PID controller uses a feedback loop to control the rotor angular speed output of the vehicle. Within this loop, the controller uses a combination of the previous output, the current error, and the previous two errors to estimate what adjustments must be made to reach or maintain the desired output. Using the aerial footage taken by drones, farmers can get some useful information about crops, as follows: may reveal patterns of irrigation or soil and fungal infestations that are not visible to the naked eye; the combination of multispectral images, infrared or visible, can create an image of crop that emphasizes healthy plants and those in need; by monitoring the crop at regular intervals, animations can be created, images that show changes over time, revealing problem areas or opportunities to better crop management [32]

We used the Ionic for the reason is the background support that works above Cordova to build hybrid applications. It allows to development of the application interface that can be compared as it were Web pages, i.e., using HTML, CSS, and JavaScript. and these Applications run inside the WebView of the original application.

Ionic Framework - is an open-source framework in Code that must be written once and executed on it Many mobile devices can work on different types of operating systems [33] where the ionic provides the Cordova tool and Capacitor is a cross-platform native runtime that makes it easy to build modern web apps that run natively on iOS, Android, and the Web. Representing the next evolution of Hybrid apps, Capacitor creates Web Native apps, providing a modern native container approach for teams who want to build web-first without sacrificing full access to native SDKs when they need it [34]

The proposed system helps in the identification of plant disease and provides remedies that can be used as a defense mechanism against the disease. using convolution neural network (CNN)over a proper dataset. a prototype drone was also designed which can be used for live coverage of large agricultural fields to which a high-resolution camera is attached capturing plants images. The goal was to reduce the attack of pests also provide a remedy for the disease that is detected [35]

CHAPTER THREE
METHODOLOGY

The objectives of this research study are to reduce losses in cotton productivity resulting from bacterial blight to the least possible, a disease also known as angular spot disease in cotton. It is one of the important cotton diseases with global spread and can cause production losses of up to 10% of the crops. The research also aims to develop modern agriculture with new techniques and raise the level of Sudanese agriculture to keep pace with global agriculture. We used drones that save effort and time for farmers, especially when the areas are large. We used digital image processing technology to analyze images taken by drones that show the extent of plant health. This chapter is an overview Detailed on the work performed on the project, which can be divided into software system parts, UAV movement, and control functions, it consists of hardware design and software design

Figure 3-1: shows system example.

Figure 3-1: System Example

3.1.1 Software system

The object detection process consists of Data Collection, Data Augmentation Operations, Preprocessing, Building the CNN model, Training the model, and Postprocessing Operations.

3.1.2 Method Overview

The software of building the dataset of CNN model classification dieses, preprocessing of images, Building the CNN model, and training the model, building the neural network layers, the second section is Interface show the Result of Processed image.

Figure 3-2: show the software of the system block diagram.

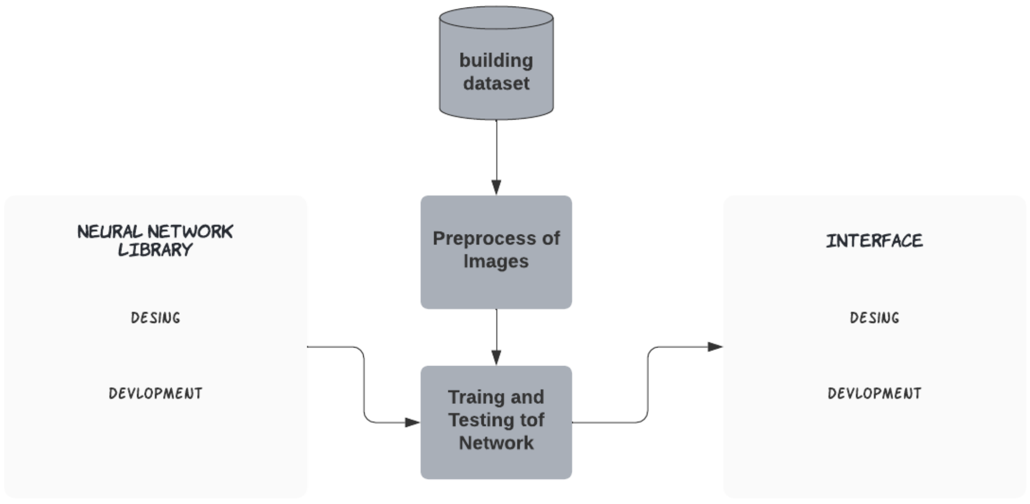


Figure 3-2: software system block diagram

3.1.3 Building of the Dataset

To train a deep learning network for our research problem, a dataset of diseased and healthy cotton leaf images had to be acquired the dataset was put together by downloading diseased cotton images on the internet from various sources Cotton plants are affected by diseases caused by, bacteria, and viruses and to damage by parasitic worms and physiological disturbances also classified as diseases. Cotton is threatened by different types of diseases, such as Bacterial blight. We have limited our study to only one main disease because of time constraints. A brief description of the disease is as follows:

3.1.4 Bacterial blight

It is the most devastating cotton crop disease. Cotton bacterial blight is caused by a bacterial called Xanthomonas citric subs malvaceous which survives in infected crop debris and seeds. It starts as an angular, waxy, and water-soaked leaf spot with a red to brown border on leaves, stems, and bolls. As the plant grows, the spots gradually turn into brown necrotic areas. If these diseases are left untreated, they will kill the plant. However, if they are diagnosed early, they can be treated, and the crop can be saved. The third category of classification is healthy cotton leaves to allow the model to tell the difference between a healthy and diseased cotton leaf.

Figure 3-3: shows the spread of bacterial blight disease

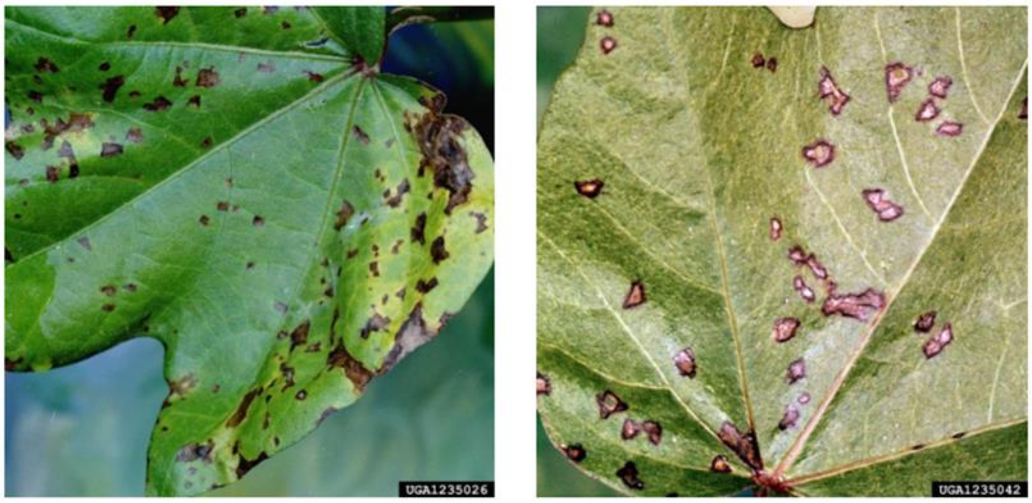


Figure 3-3: Bacterial blight

3.1.4.1 Dataset Annotation

After downloading the images, you must remove the duplicates and make sure that each image is placed in the appropriate file. The healthy image in the file. The healthy images (name) and the bacteria-infected images are in the bacteria-infected im age file. The step is very important and it must be ensured that it is obtained in the correct image to obtain Excellent training results.

Figure 3-4 shows the folders containing the diseased and healthy images.

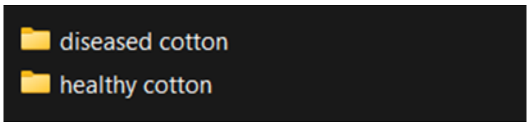

diseased cotton
healthy cotton

Figure 3-4: folders contain the diseased and healthy images

3.1.4.2 Dataset Division

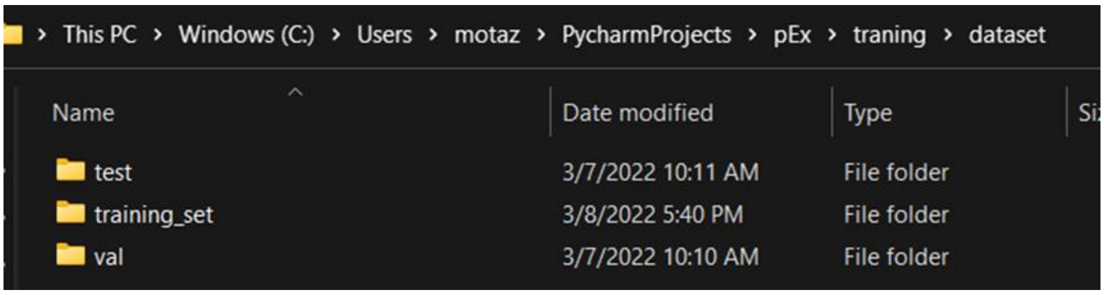
To train and test the model, three separate data sets are required. In the process, the home picture group together in the previous section is divided into the following groups:

1- Training set: It is the set that is used to train the model to know its hidden features such as weights and biases.

2- Validation set: The validation set is used to evaluate the model. These include among others the learning rate, batch size, and several epochs.

3- Test set: This set is used when the training phase is completed to evaluate the performance and accuracy of the final trained model.

Figure 3-5: shows the three sets test set and training set validation set


› This PC › Windows (C:) › Users › motaz › PycharmProjects › pEx › traning › dataset

Name	Date modified	Type	Si
test	3/7/2022 10:11 AM	File folder	
training_set	3/8/2022 5:40 PM	File folder	
val	3/7/2022 10:10 AM	File folder	

Figure 3-5: three sets test set and training set validation set

3.1.5 Preprocessing of Images

The next stage after building the dataset was the preprocessing of the images. This process is very important in deep learning to ensure that training data is standardized before it is fed into a training model. For example, images must be resized to match the input size required by the network.

3.1.5.1 Data Augmentation

To enrich the data set, several techniques were used to increase the number and diversity of images available in the data sets. Enhanced images increase the network's opportunity to learn more features and be able to accurately distinguish one category from another (Net Image Classification).

3.1.5.2 Dataset composition

Table 3-1: shows the division of the original images in the data into three different sections: training, validity, and testing for each classification of healthy and sick patients to be detected.

Table 3-1: dataset

Class	Original images	Original and augmented images	Training images	Validation images	Test images
Healthy leaves	**256**	**3870**	**3096**	**387**	**387**
Bacterial Blight	**430**	5982	4785	598	598

3.1.6 Neural Network Design

After the images are acquired and preprocessed, the next step is to design and train a model on those images.

3.1.6.1 Process Parameters

To begin model development, there are design options that must be considered. The main factor is the choice of architecture for the model. Using the architecture developed by the open-source "Indiana production" for educational purposes, it was possible to achieve high accuracy (98%) on the cotton crop. For this reason, "Indiana production" was selected as the preferred architectural design for this research. We use this architecture as a feature extractor but modify and tune it to support our disease class.

3.1.6.2 Model Design

We used the layers in the "Indiana production" as a feature extraction component. The "Indiana production" model was loaded without the classifier part of the model and changed a new Dense and Output layer to result in 2 classes altered for the requirements of our new dataset to predict the probability for 2 classes.

Figure 3-6: shows the block diagram design CNN model.

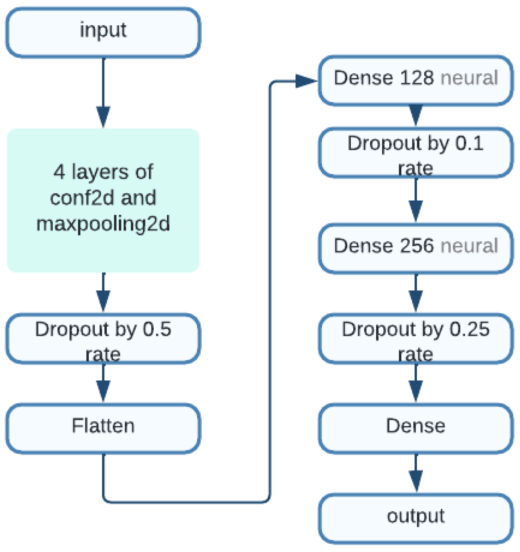


Figure 3-6: Deep neural network

The block diagram was drawn using lucid.app [36]. All code was written in the Python programming language. For the implementation of the deep neural network, Keras [37] library a python-based deep Learning library was used. Keras library runs on TensorFlow [38] backend. It was chosen as the backend for Keras because it offers high-performance numerical computations. The full code is attached in the appendix section of this report. Keras implementation workflow was as follows:

1- Import the libraries in Keras
2- -parameters the dataset
3- change new layers on the output of the model
4- Train the new layers on your dataset

3.1.7 Training the model

Using Article developed By Indian AI Production Access the directory where mounted project the script to run the training of the model "train.py".

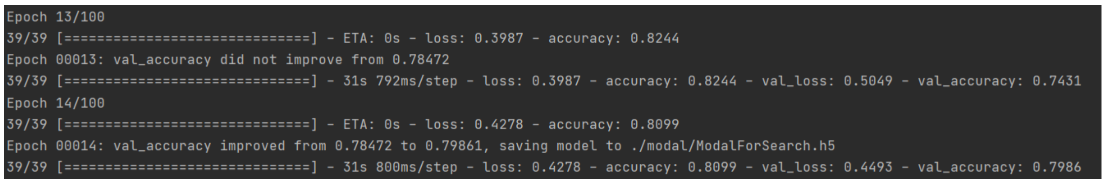
3.1.7.1 Training of Neural Network:

All experiments in this section for the training of neural networks were run on Intel Core i3 11th generation, with 8GB RAM.

3.1.7.2 Train Run

A-Train run was done on the model with the model run with an initial batch size of 32 meaning that 32 image samples are used to train the network each time. 100 epochs were set for the initial run and a learning rate of 0.0001 was configured.

The results of the first run are as shown below in Figure 3-7 and the accuracy at the end of the 14th epoch was 79%.

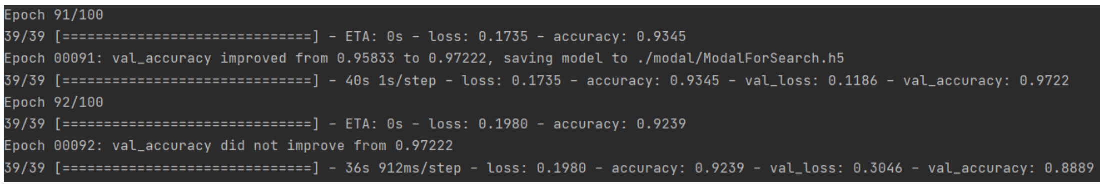


```
Epoch 13/100
39/39 [==============================] - ETA: 0s - loss: 0.3987 - accuracy: 0.8244
Epoch 00013: val_accuracy did not improve from 0.78472
39/39 [==============================] - 31s 792ms/step - loss: 0.3987 - accuracy: 0.8244 - val_loss: 0.5049 - val_accuracy: 0.7431
Epoch 14/100
39/39 [==============================] - ETA: 0s - loss: 0.4278 - accuracy: 0.8099
Epoch 00014: val_accuracy improved from 0.78472 to 0.79861, saving model to ./modal/ModalForSearch.h5
39/39 [==============================] - 31s 800ms/step - loss: 0.4278 - accuracy: 0.8099 - val_loss: 0.4493 - val_accuracy: 0.7986
```

Figure 3-7: 14th epoch

An accuracy of 97.2% was realized at the end of the 91th epoch.

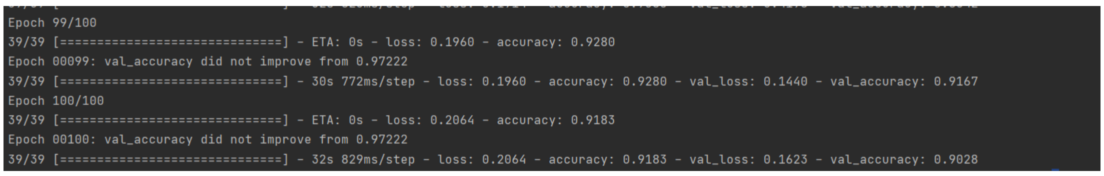
Figure 3-8: shows the last accuracy improve.



```
Epoch 91/100
39/39 [==============================] - ETA: 0s - loss: 0.1735 - accuracy: 0.9345
Epoch 00091: val_accuracy improved from 0.95833 to 0.97222, saving model to ./modal/ModalForSearch.h5
39/39 [==============================] - 40s 1s/step - loss: 0.1735 - accuracy: 0.9345 - val_loss: 0.1186 - val_accuracy: 0.9722
Epoch 92/100
39/39 [==============================] - ETA: 0s - loss: 0.1980 - accuracy: 0.9239
Epoch 00092: val_accuracy did not improve from 0.97222
39/39 [==============================] - 36s 912ms/step - loss: 0.1980 - accuracy: 0.9239 - val_loss: 0.3046 - val_accuracy: 0.8889
```

Figure 3-8: 91th epoch

The accuracy didn't increase after that in the 100th epoch as shown in Figure 3-9.



```
Epoch 99/100
39/39 [==============================] - ETA: 0s - loss: 0.1960 - accuracy: 0.9280
Epoch 00099: val_accuracy did not improve from 0.97222
39/39 [==============================] - 30s 772ms/step - loss: 0.1960 - accuracy: 0.9280 - val_loss: 0.1440 - val_accuracy: 0.9167
Epoch 100/100
39/39 [==============================] - ETA: 0s - loss: 0.2064 - accuracy: 0.9183
Epoch 00100: val_accuracy did not improve from 0.97222
39/39 [==============================] - 32s 829ms/step - loss: 0.2064 - accuracy: 0.9183 - val_loss: 0.1623 - val_accuracy: 0.9028
```

Figure 3-9: 100th epoch

Figure 3-10: shows the accuracy and loss result in training.

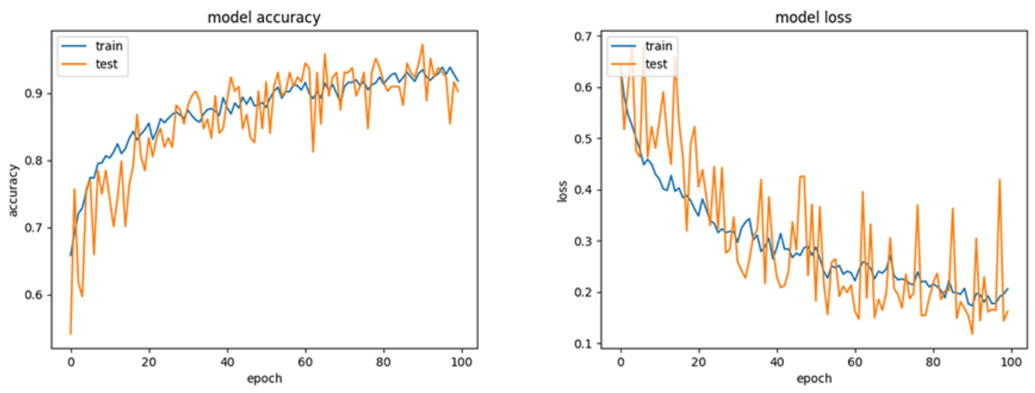


Figure 3-10: accuracy and loss

3.1.8 Interface:

A simple web interface for displaying the results of the image processing process. It was designed on the ionic framework. It contains two pages, a page for uploading the images to be analyzed, and a page for displaying the results of the analysis., Figure 3-11 show the flowchart of the full system from the start point to combine processed image to map in their locations.

Figure 3-11: shows the interface flowchart.

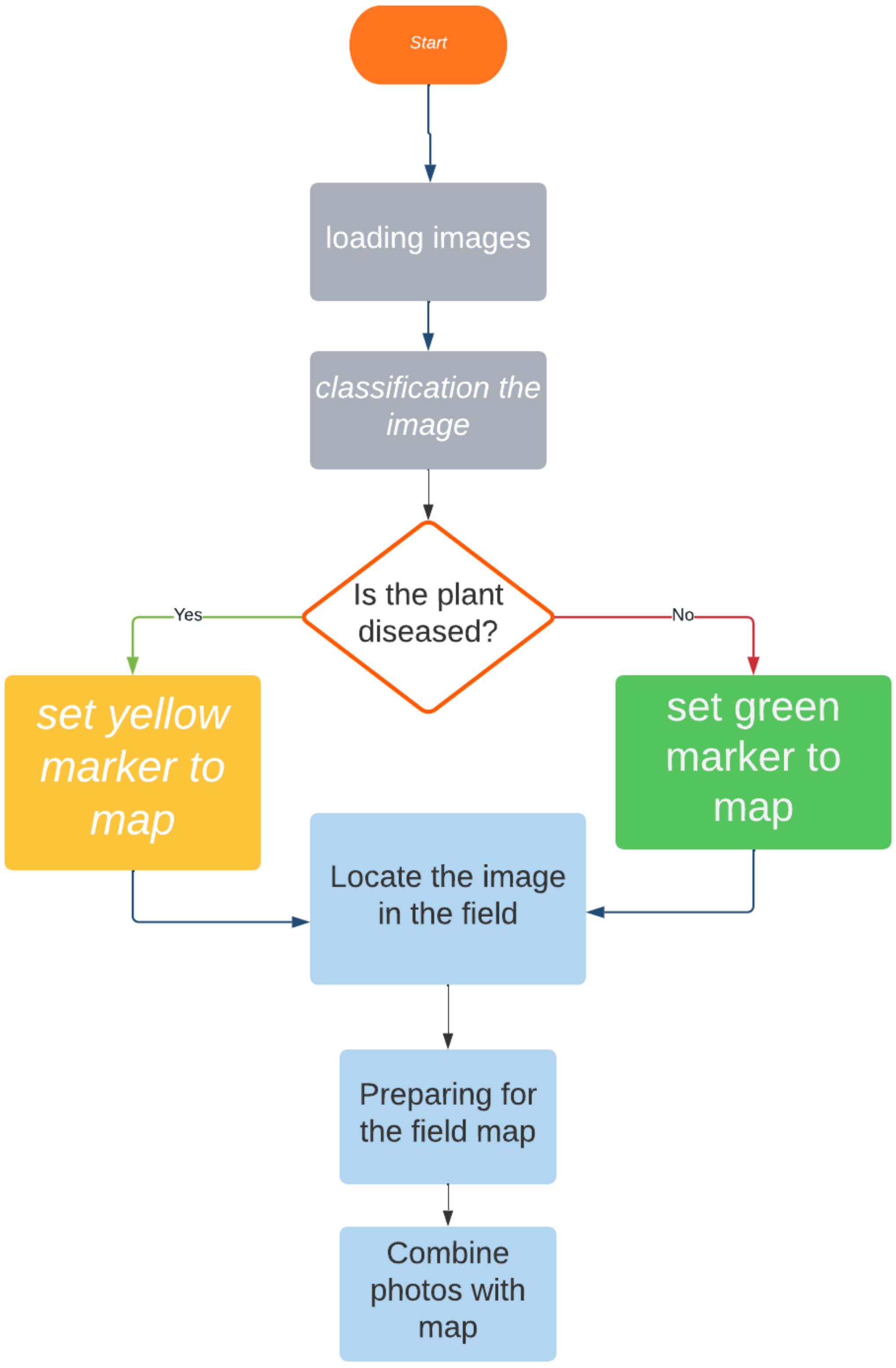


Figure 3-11: flowchart predicting

The main page of the interface: Figure 3-12 at the bottom display the home page layout of the interface

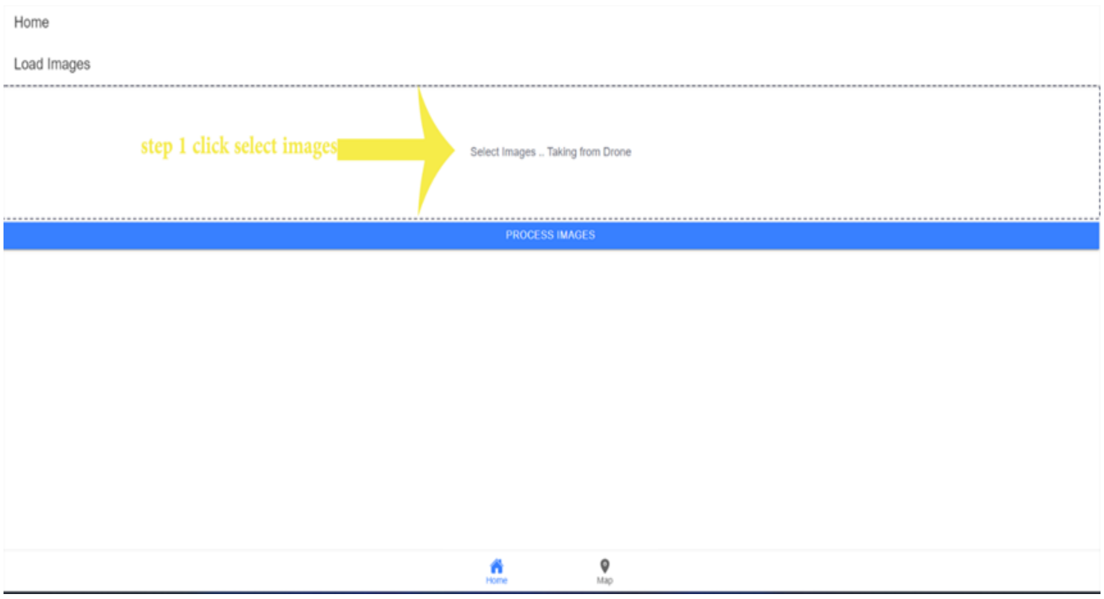
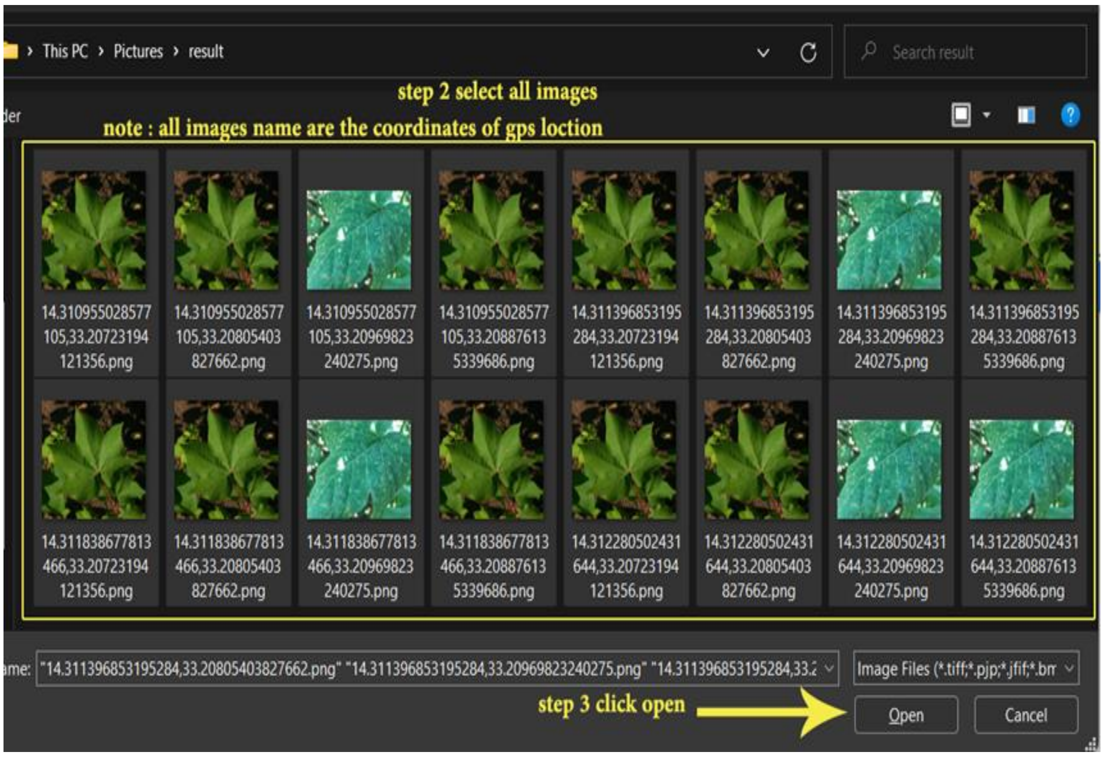

Home
Load Images
step 1 click select images
Select Images .. Taking from Drone
PROCESS IMAGES
Home
Map

Figure 3-12: home page layout

After selecting the images to be analyzed, they are sent to the CNN model code that was, Figure 3-13 at the bottom display the steps to upload images.


This PC › Pictures › result
Search result
step 2 select all images
note : all images name are the coordinates of gps loction
step 3 click open
Open
Cancel

14.310955028577105,33.20723194121356.png
14.310955028577105,33.20805403827662.png
14.310955028577105,33.20969823240275.png
14.310955028577105,33.208876135339686.png
14.311396853195284,33.20723194121356.png
14.311396853195284,33.20805403827662.png
14.311396853195284,33.20969823240275.png
14.311396853195284,33.208876135339686.png
14.311838677813466,33.20723194121356.png
14.311838677813466,33.20805403827662.png
14.311838677813466,33.20969823240275.png
14.311838677813466,33.208876135339686.png
14.312280502431644,33.20723194121356.png
14.312280502431644,33.20805403827662.png
14.312280502431644,33.20969823240275.png
14.312280502431644,33.208876135339686.png

ame: "14.311396853195284,33.20805403827662.png" "14.311396853195284,33.20969823240275.png" "14.311396853195284,33.2

Image Files (*.tiff;*.pjp;*.jfif;*.bm

Figure 3-13: selecting images for predicting

3.1.8.1 main function of this web page

The main function of this web page is to classify cotton diseases, so a JavaScript function was developed to send images to our model developed in the previous section (python code)

Appendix B 2, the flowing code is JavaScript code contain function send images

```
processImages () {//called we click on process button
  this.files.forEach ( (file, i) => {//loop in all images selected
   this.postfile(file, 'predict','', async (result)=> {//send the image to CNN model to predict
     let img = await this.getBase64(file)//convert image encode to base64
     this.markers.data.push({//preparing map markers in place of each image
       image: img, //set image (base64)
       location: this.locations[i], //set coordinates to marker
       status: result.status//the status is the result coming from CNN model is the either healthy or disease
     })
     if (i=== this.files.length -1){//check if this last image
       alert('Processing End Open Map to Show Result') //alert user to go map to see the result
}})})}
```

3.1.8.2 main function back end

is the code for receiving image analysis from the interface

```
def predict():
 if request.method == 'POST':# form request method
     file = request.files['image']  # fet input
     filename = file.filename # saving file name
     file_path = os.path.join('static/user uploaded', filename) # save the image in uploaded files
     file.save(file_path) # save image
     pred, output_result pred_cot_dieas(cott_plant=file_path)#fuction pred_cot_dieas will fine out in appendix
     return {'status': output_result, 'pred': pred} # response to interface with predicted status of image
```

3.1.9 processing Image

predictions that the model has made for the test data, we can use the predict function. The predict function will give an array with 2 numbers. The array index with the highest number represents the model prediction. The sum of the array equals 1 (since each number is a probability). find out the code in the appendix.

3.1.10 Locate the image in the field

In this step, the images are collected with the map at the location they were taken from the drones stored in the locations located in the file name in formula "lat, log.png"

3.1.11 Preparing for the field map

- We declare the application as HTML5 using the HTML declaration.
- We create a div element named "map" to hold the map.
- We define a JavaScript function that creates a map in the div.
- We load the Maps JavaScript API using a script tag [39]

3.2 Hardware design

3.3 Design of quadcopter

The system model is separated into two parts: the electrical part which consists of Ardupilot, electronic speed controller, Lithium Polymer battery, brushless DC motor, propellers, transmitter and receiver, Arduino Uno, and ESP 32 camera.

And the mechanical parts consist of frame, arms and propellers Figure 3-14: shows the block diagram of the system model

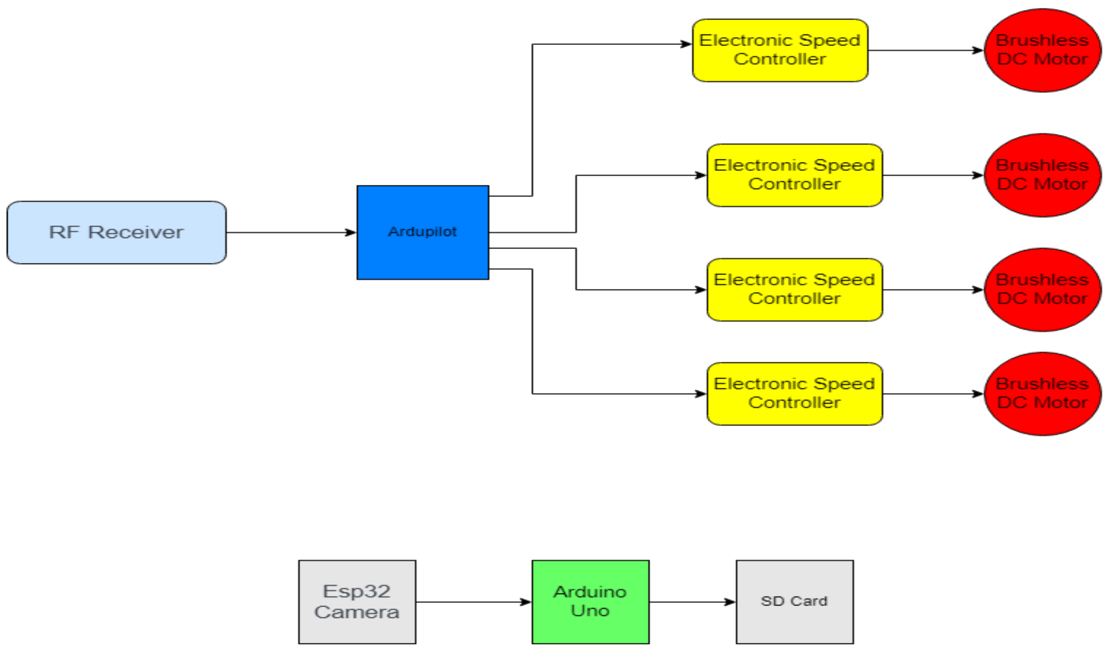


Figure 3-14: hardware system block diagram

3.3.1.1 Drone assembly

The first step of assembling a drone is building a frame or choosing the proper type of manufactured carbon fiber frame as mentioned earlier because it can hold much weight more than plastic, and the carbon fiber substance of itself has lightweight.

The second step is to connect both the positive and negative of the motor speed controller with the positive and negative of the PDB and the same process is done on the other ESC.

Figure 3-15: shows connecting the electronic speed controller to the power distribution

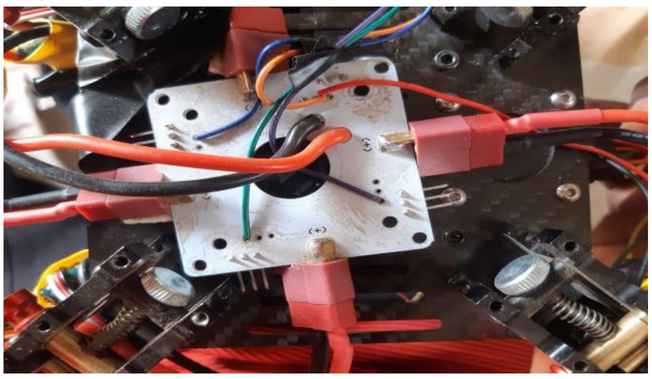


Figure 3-15: ESCs and PDB

ECSs connections to the motors:

For Clockwise two motors:

- Connect the VCC of ESC with the VCC of the motor.
- Connect the GND of ESC with the GND of the motor.
- Connect PWM/Signal of ESC with Signal/PWM of the motor.

For Counter-Clock wise two motors just swap any two wires:

- Connect the VCC of ESC with GND of motor
- Connect the GND of ESC with the VCC of the motor.
- Connect PWM/Signal of ESC with Signal/PWM of motor

Quadcopter controller (Ardupilot) attached on the main base of the drone with Ardupilot input pins (1-5) are connected with the Fly Sky FSiA6 receiver channel input pins (CH1-CH5) via jumper wires (Female to Female).

Now ESC connections with Ardupilot flight controller as follows:

Figure 3-16: shows the Ardupilot mega pins.

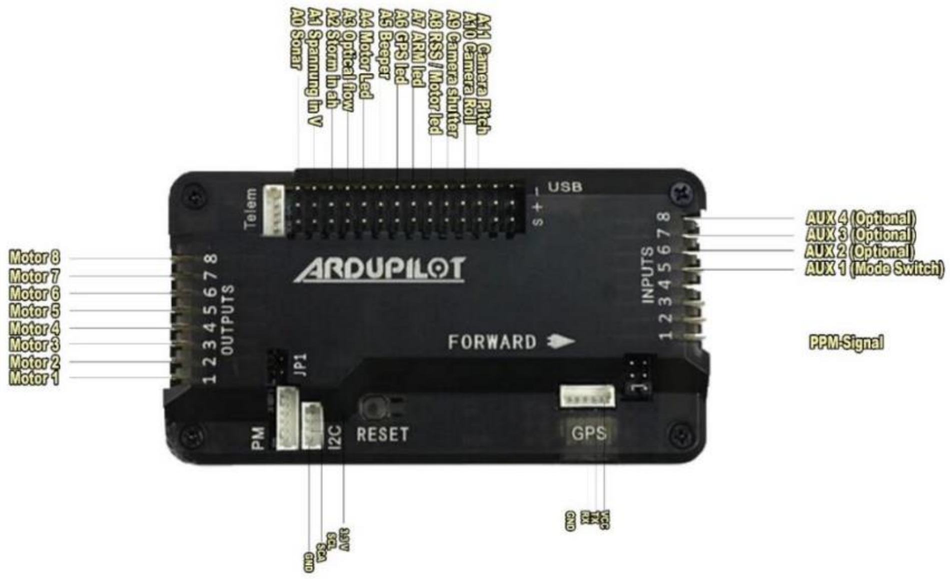


Figure 3-16: APM 2.8 Pins Configurations

- Connect the ESC-1 Pin with Ardupilot Output Pin - 1.

- Connect the ESC-2 Pin with Ardupilot Output Pin - 2.
- Connect the ESC-3 Pin with Ardupilot Output Pin - 3.
- Connect the ESC-4 Pin with Ardupilot Output Pin - 4.

Figure 3-17: shows ESCs and Receiver connect to the ardupilot.

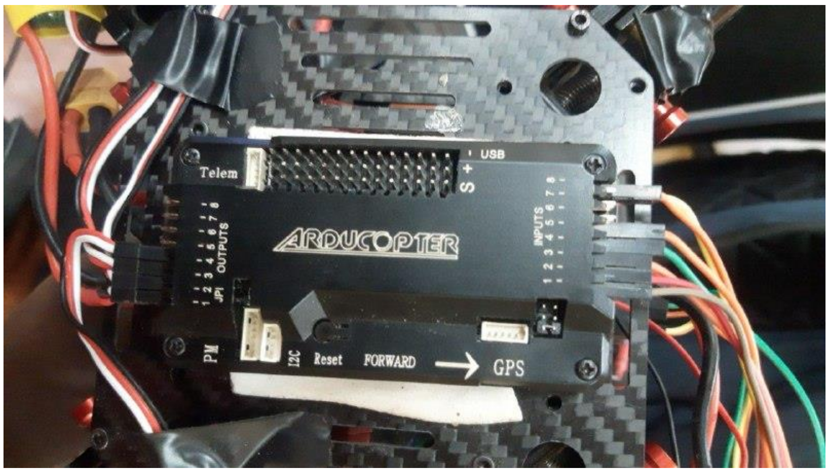


Figure 3-17: ESCs connection to Ardupilot Output and Receiver Channel Input

3.3.1.2 Programming Flight Controller

Mission Planner is a ground control station for Plane, Copter, and Rover. It is compatible with Windows only. Mission Planner can be used as a configuration utility or as a dynamic control supplement for autonomous vehicles.

Mission Planner contains many features as:

- Load the firmware (the software) into the autopilot board that controls the vehicle.
- Setup, configure, and tune vehicles for optimum performance.

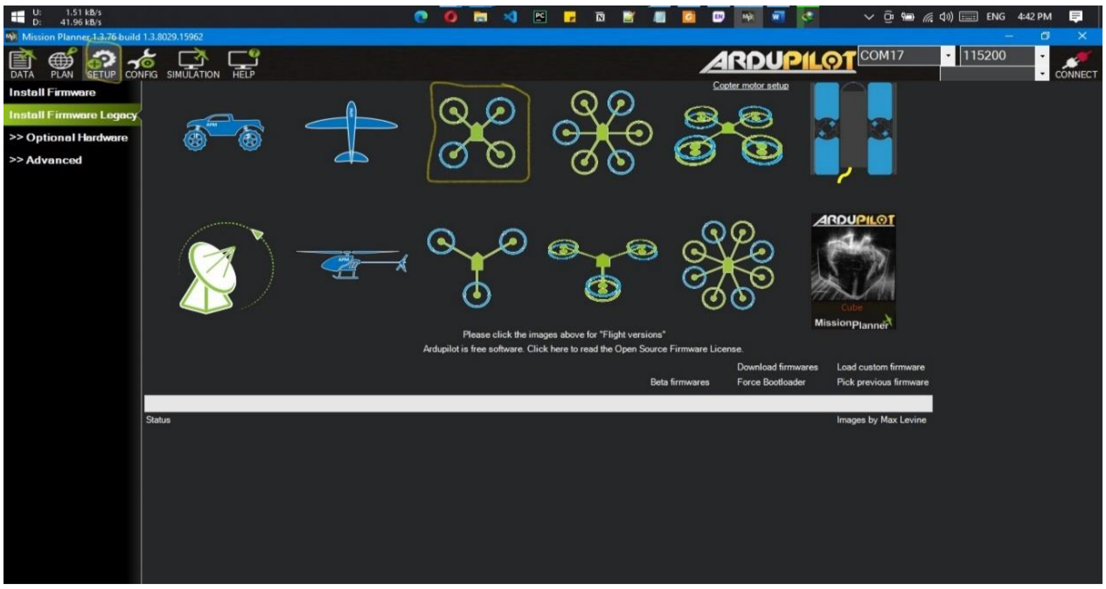
3.3.1.3 Installing and Configuration

Once the ground station is installed on the computer, the autopilot is being connected using the micro-USB cable. Windows should automatically detect and install the correct driver software.

The COM port drop-down on the upper-right corner of the screen AUTO option is selected or a specific port for the type of the connected board. The Baud rate is set to 115200 as shown.

On the Mission Planner's SETUP | Install Firmware screen, an appropriate icon that matches the quadcopter frame is selected Next it will detect the board type that is being used. After all, goes well the firmware will be successfully uploaded to the board.

Figure 3-18: shows the mission planner setup.


Mission Planner 1.3.76 build 1.3.8029.15962

Please click the images above for "Flight versions"

Ardupilot is free software. Click here to read the Open Source Firmware License.

Figure 3-18: Mission Planner Setup Page

Radio Control Calibration involves capturing each RC input channel's minimum, maximum, and "trim" values so that Ardupilot can correctly interpret the input. To setup transmitter:

- Ensure the battery is disconnected
- Connect RC transmitter to the Ardupilot
- Turn on RC transmitter
- Connect the autopilot to the PC using a USB cable
- On the Mission Planner press the "Connect" button and open Mission

- Planner’s INITIAL SETUP | Mandatory Hardware | Radio Calibration screen
- Some green bars should appear showing the Ardupilot is receiving input from the Transmitter/Receiver

  Figure 3-19: shows the radio calibration.

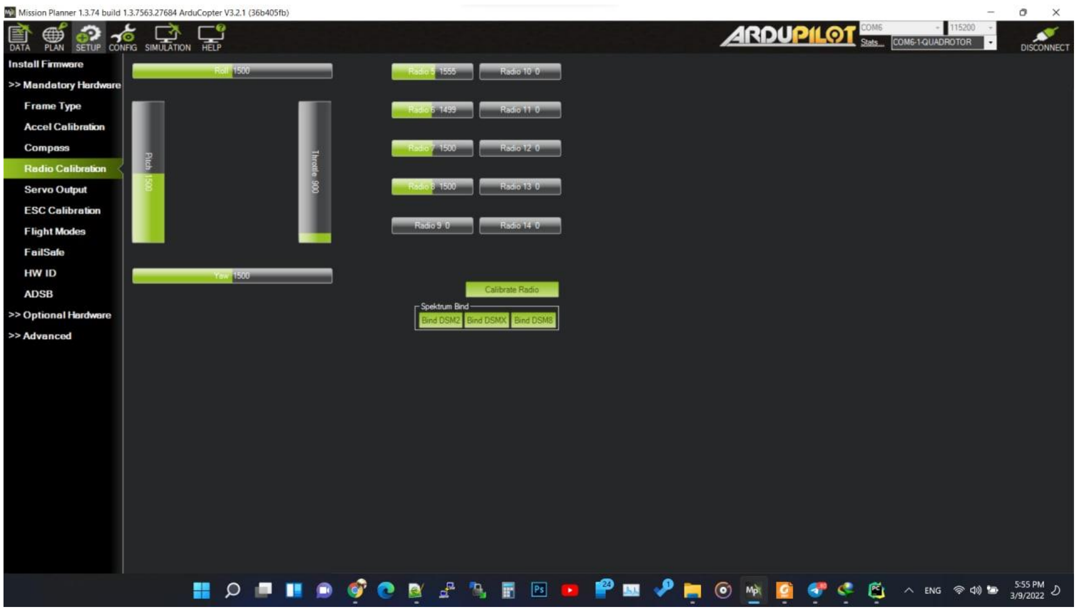


Figure 3-19: Radio Calibration Tab

- Calibration:
- Open Mission Planner’s INITIAL SETUP | Mandatory Hardware | Radio Calibration screen
- Click on the green “Calibrate Radio” button on the bottom right
- 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 are seen so far.

Figure 3-20: shows the transmitter calibration.

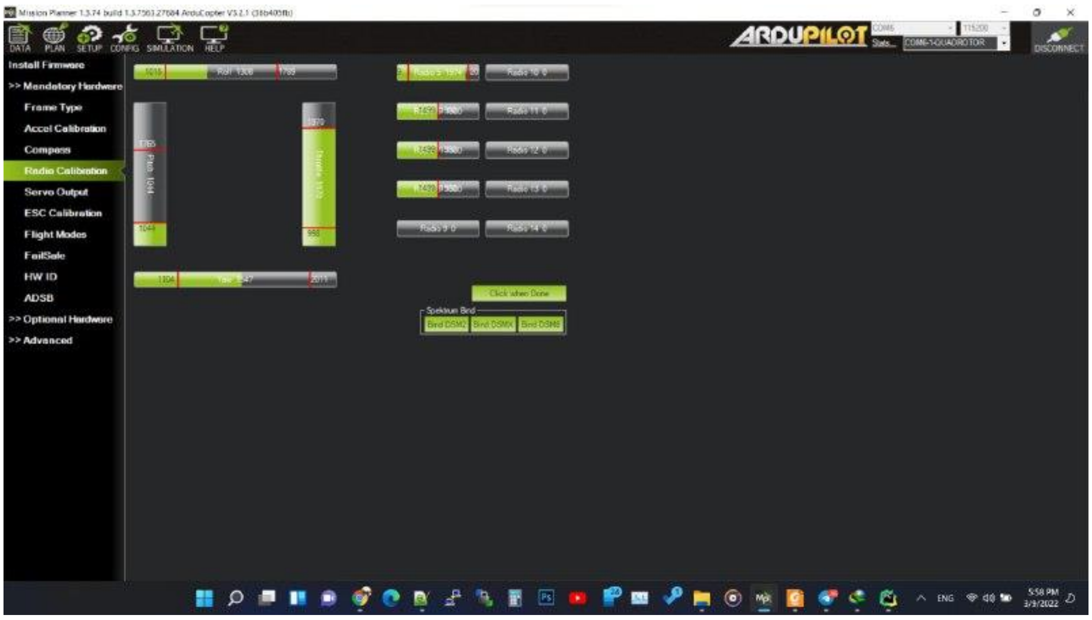
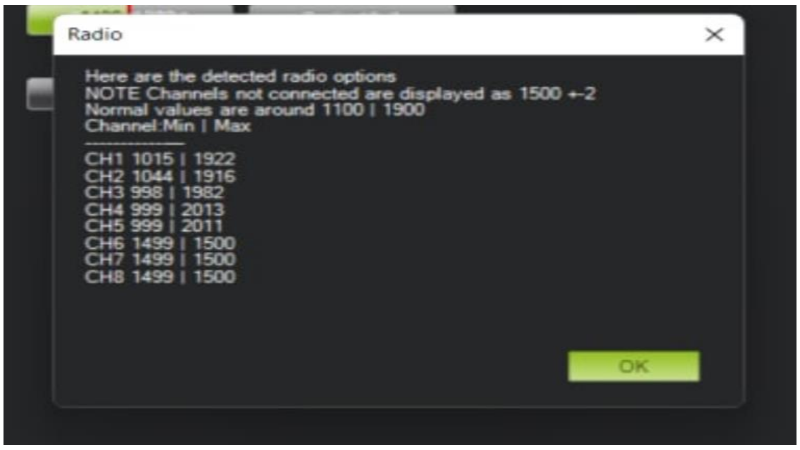


Figure 3-20: Transmitter Calibration

- Select Click when Done.
- A window will appear with the prompt, "Ensure all your sticks are
- centered and the throttle is down and clicks 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 1015 for minimums and 2013 for maximums

  Figure 3-21: shows transmitter values.



Radio

```
Here are the detected radio options
NOTE Channels not connected are displayed as 1500 +-2
Normal values are around 1100 | 1900
Channel:Min | Max
--------------
CH1 1015 | 1922
CH2 1044 | 1916
CH3 998 | 1982
CH4 999 | 2013
CH5 999 | 2011
CH6 1499 | 1500
CH7 1499 | 1500
CH8 1499 | 1500
```

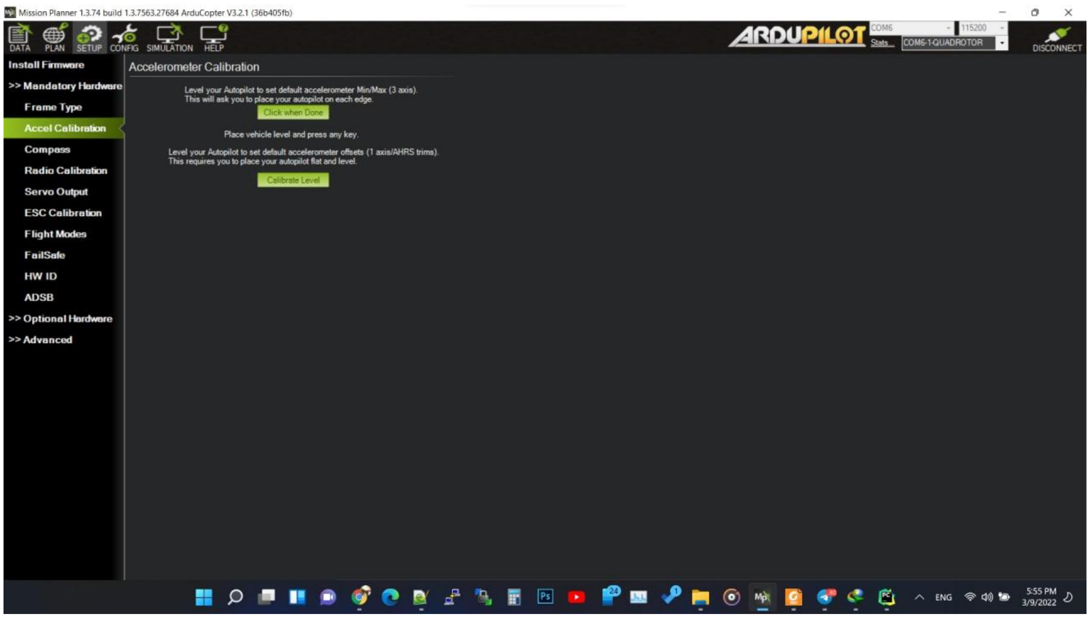
OK

Figure 3-21: Transmitter Calibration Data

For accelerometer calibration:

- Under Setup | Mandatory Hardware, select Accel Calibration from the left side menu

  Figure 3-22: shows the accel calibration process.


Mission Planner 1.3.74 build 1.3.7563.27684 ArduCopter V3.2.1 (36b405fb)

Accelerometer Calibration

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

Place vehicle level and press any key.

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

Figure 3-22: Accel Calibration

- Click Calibrate Accel to start the calibration
- Mission Planner will prompt 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 completed the calibration process, Mission Planner will display “Calibration Successful!” as shown below.

3.3.1.4 Installing the camera

The ESP32-CAM board already contains the camera module, and microSD card slot in addition to this, we used a microSD card to save the video

connect the ESP32 to the Arduino:

- connect the ground of the Arduino with the reset pin in Arduino.
- connect 5v pin in Arduino with the 5v pin in ESP32.
- connect the ground in the Arduino with the ground in the Esp32.
- connect the RX pin in the Arduino with the U0R in The ESP32.
- connect the TX pin in the Arduino with the U0T in The ESP32.
- connect GIPO pins and ground pins in the ESP32.

  Figure 3-23: shows Connecting the camera to the Arduino.

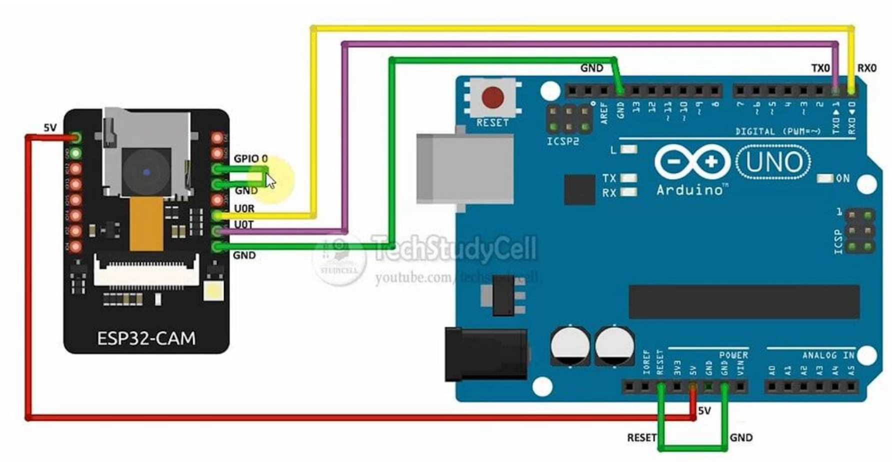


Figure 3-23: connect the ESP32 to the Arduino

CHAPTER FOUR
RESULT AND DISCUSSION

This section presents all the results of our dissertation of all the work carried out in the previous section.

4.1 Model training results

The results presented in this section are related to the training of our deep learning model with the collected image dataset. As mentioned in the previous chapter, we developed a cotton crop disease identification model based on transfer learning. " ndianaiproduction" was used as a feature extractor and a change, a new Dense and Output layer to result in 2 classes were changed or the requirements of our research problem.

Our dataset was split into three subsets namely training set, validation set, and test set. During the training process, our model was periodically evaluated using the validation set.

The model autotunes some of the parameters based on the periodic evaluation results on the validation set. The final evaluation of the model after the training phase has been completed was carried out using the training set. This is the most important step to get the working accuracy and generalizability of our model. Matplotlib in Keras was used to plot the training and validation losses vs epochs and training and validation accuracies vs epochs after the training process was completed.

4.2 Loss Graphs

The figure 4-1: shows the loss graphs of the training process.

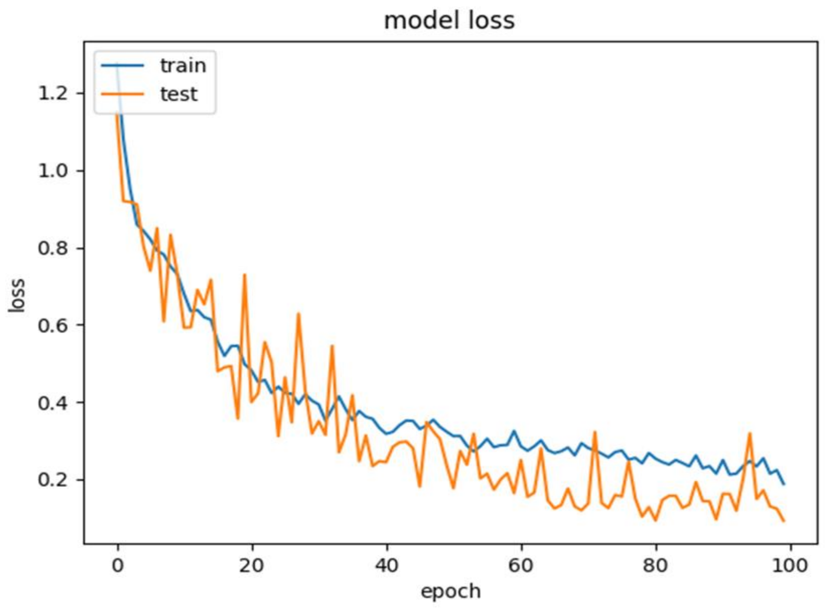


Figure 4-1: we can see that we obtained a good fitting curve up to around 90 epochs which

Is identified by a training and validation loss that decreases to a point of stability with a minimal generalization gap between the two values. The performance of the model on the validation dataset began to degrade after 90 epochs, so the training process was stopped. Before 90 epochs, the model has low variance and generalizes the data well. Further training from this point increased the variance of the model which means the model is no longer learning but overfitting or memorizing the data.

4.3 Accuracy graph

After fine-tuning the parameters of the model and several training iterations, an average overall accuracy of about 97.2% was achieved as observed from Figure 4-2: Overall, this model was able to generalize our data. The accuracy increased gradually until it converged at an average of 97.2%.

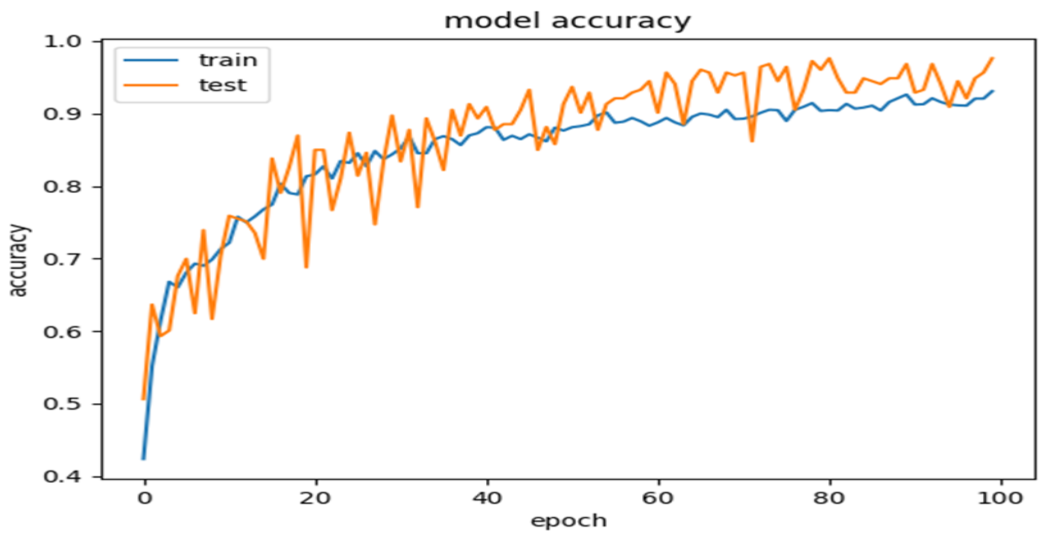


Figure 4-2: model accuracy

4.4 Simple test example

A random test was carried out on the model using an image that the model had not seen before. Figure 4-3 shows how the model was able to correctly predict the disease.

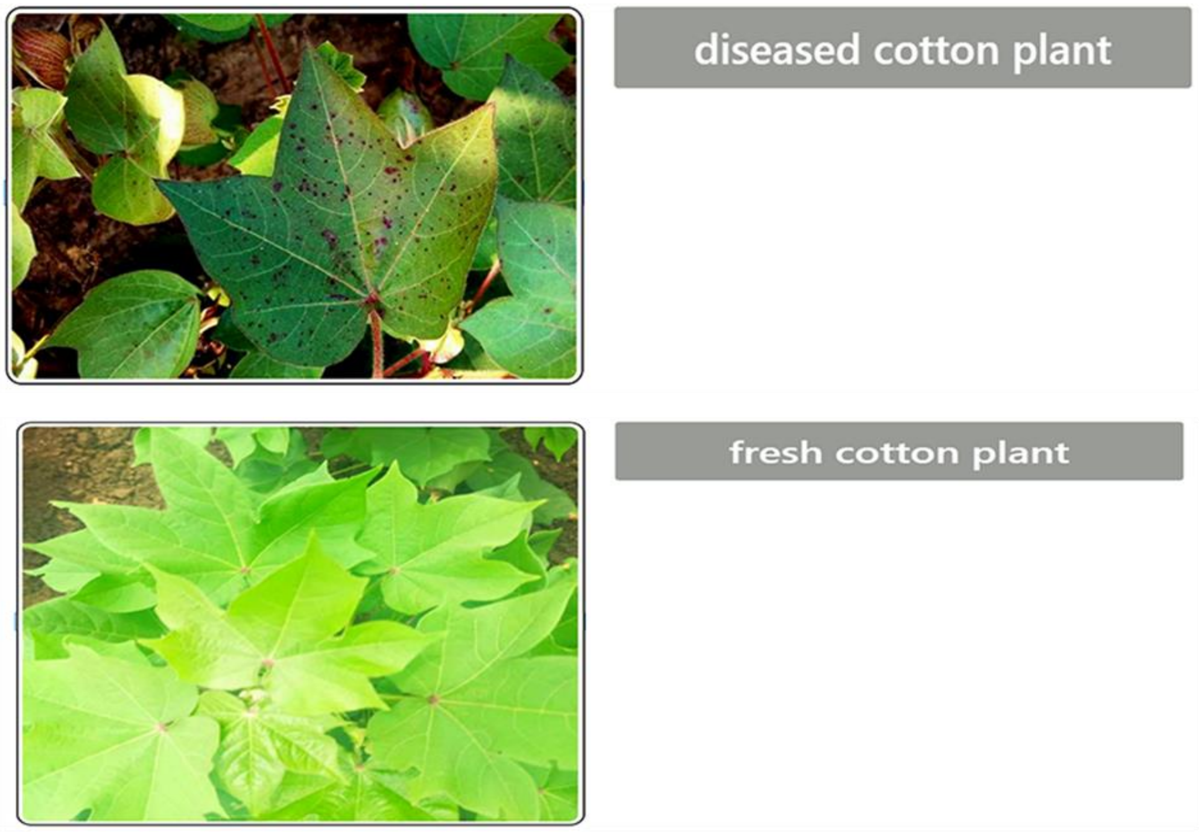


Figure 4-3: Simple test example

4.5 Result of the quadcopter

In this part, we will talk about the result of the operation of the quadcopter, which is the result of a failed take-off and a problem with the elements. Having assembled the components of the quadcopter, as mentioned in the previous chapter, in position X, we came to the following: the inability of the quadcopter to take off was the weight of the hull. During our experiment with fans, we found that the fans are larger than the frame can handle, the frame size is 380mm, the maximum it can handle is 9 inches, our fans are 10 inches, we cut the fans with our own hands to solve this problem. While attempting to take off, the quadcopter flipped over and one of the fans broke. We addressed this issue with a poster, and after the last experience and references to all sanitary settings and proper installation of propellers and motors, we came to the following: Maintenance of the quadcopter is balancing on.

Figure 4-4: Shows the result of the quadcopter.

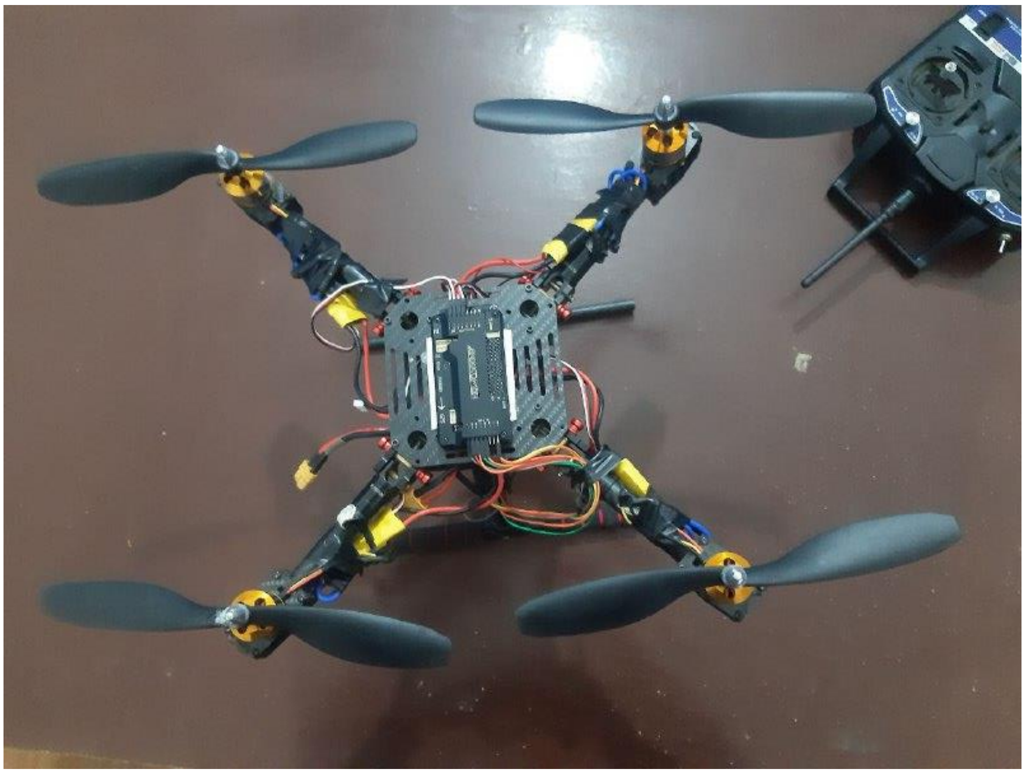


Figure 4-4: assemble the quadcopter

4.6 CNN model result in real data

After training CNN, it was used to analyze standard images) Test image from the dataset (To fully experiment with the system, the images used in Figure 4-5 appear, the coordinates of the site to be transferred in the name of the image file are placed in the name of the image file for use in determining the location of the images in the map, the coordinates taken from the task plan to determine the field area to be examined. The images are a mixture of healthy and sick plants to illustrate how the CNN model works. The images were marked by yellow icons indicating the affected plant and green icons indicating the healthy plant.

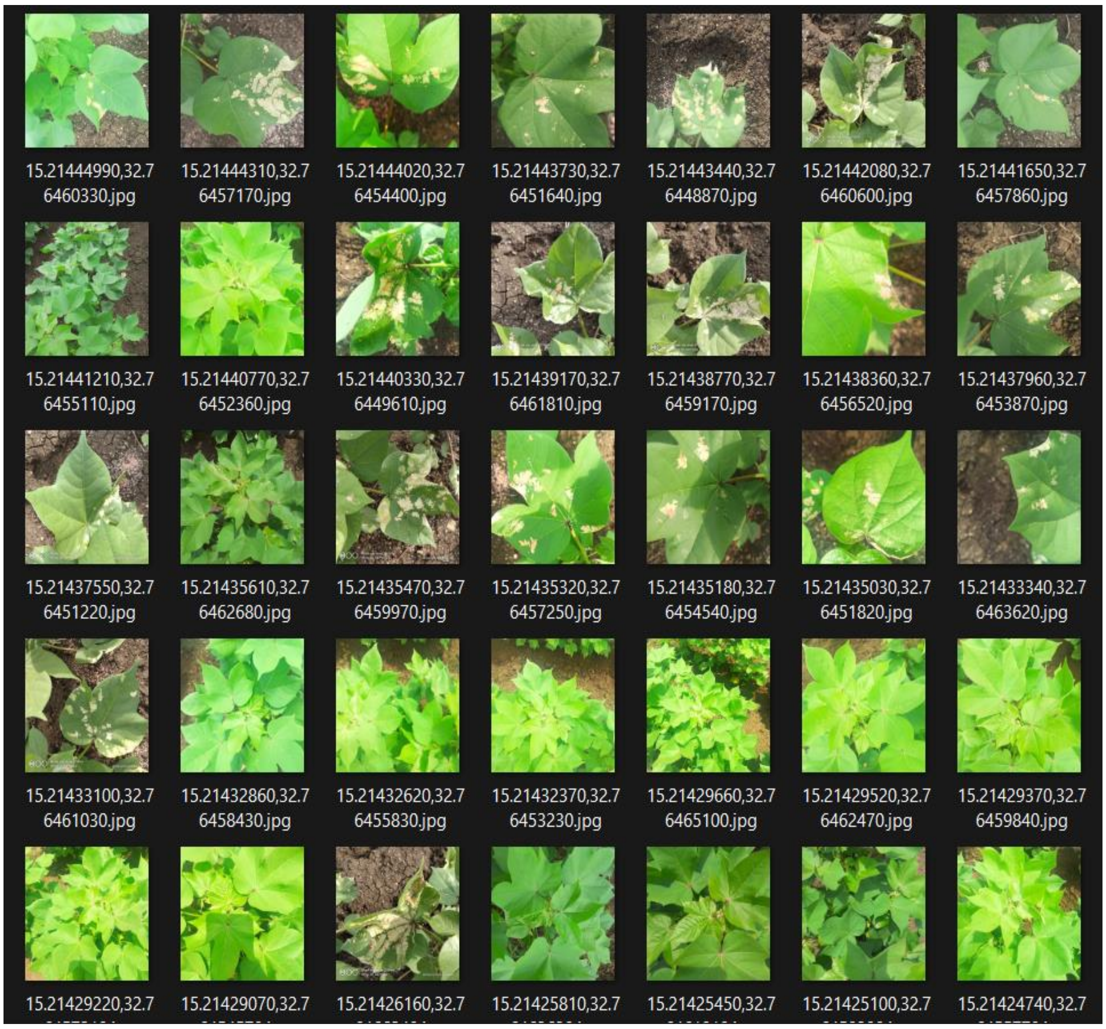
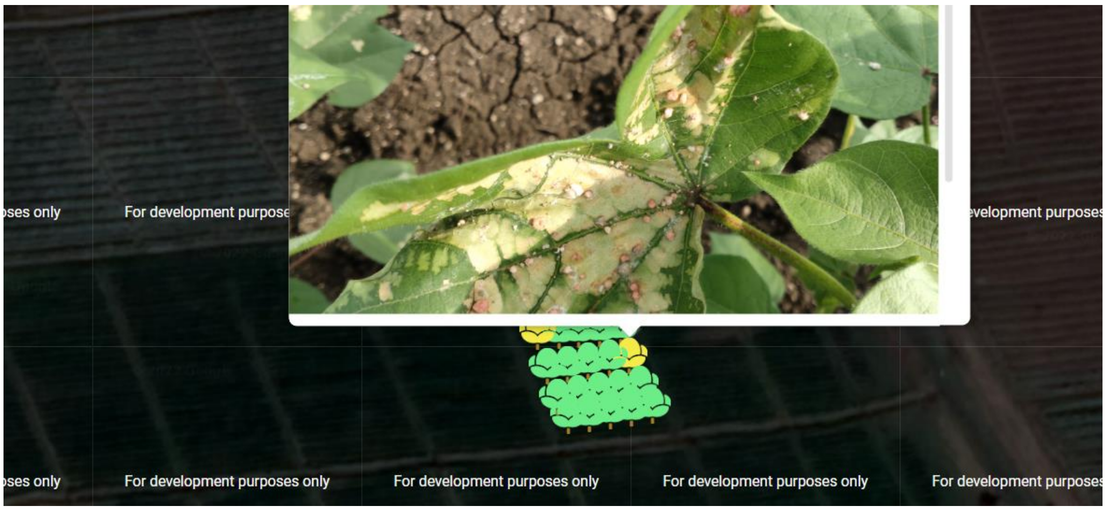

15.21444990,32.7 6460330.jpg
15.21444310,32.7 6457170.jpg
15.21444020,32.7 6454400.jpg
15.21443730,32.7 6451640.jpg
15.21443440,32.7 6448870.jpg
15.21442080,32.7 6460600.jpg
15.21441650,32.7 6457860.jpg
15.21441210,32.7 6455110.jpg
15.21440770,32.7 6452360.jpg
15.21440330,32.7 6449610.jpg
15.21439170,32.7 6461810.jpg
15.21438770,32.7 6459170.jpg
15.21438360,32.7 6456520.jpg
15.21437960,32.7 6453870.jpg
15.21437550,32.7 6451220.jpg
15.21435610,32.7 6462680.jpg
15.21435470,32.7 6459970.jpg
15.21435320,32.7 6457250.jpg
15.21435180,32.7 6454540.jpg
15.21435030,32.7 6451820.jpg
15.21433340,32.7 6463620.jpg
15.21433100,32.7 6461030.jpg
15.21432860,32.7 6458430.jpg
15.21432620,32.7 6455830.jpg
15.21432370,32.7 6453230.jpg
15.21429660,32.7 6465100.jpg
15.21429520,32.7 6462470.jpg
15.21429370,32.7 6459840.jpg
15.21429220,32.7
15.21429070,32.7
15.21426160,32.7
15.21425810,32.7
15.21425450,32.7
15.21425100,32.7
15.21424740,32.7

Figure 4-5: testing images

When clicking on the icon, the real images taken by Arduino

Figure 4-6 shows: Real image from Arduino



For development purposes only
For development purposes only
For development purposes only
For development purposes only

Figure 4-6: show disease image steed in map

Figure 4-7: shows the images were distinguished by yellow icons indicating the affected plant and green icons indicating the healthy plant. The icon site represents the coordinates of the plant with GPS

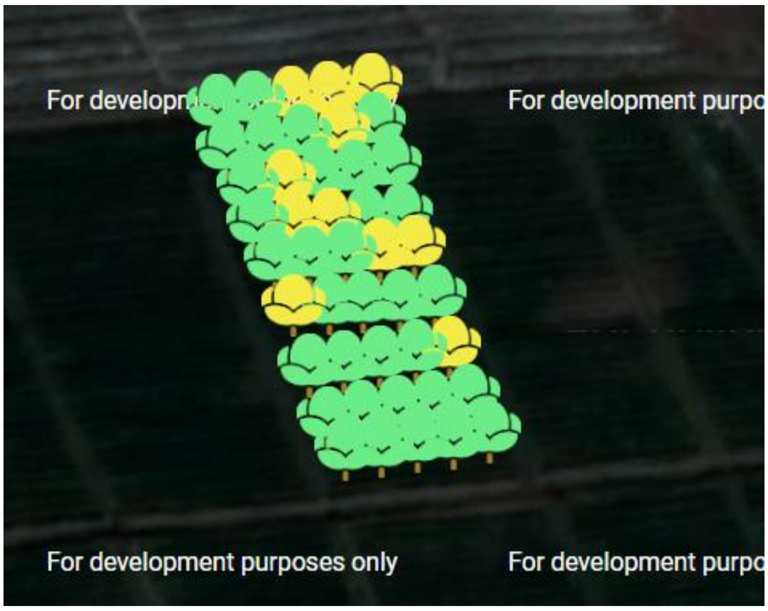


Figure 4-7: final result

CHAPTER FIVE
CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

We designed a system that develops an identification model for one of the cotton paper diseases using drones and digital image processing technology.

With this work, we will have reduced the losses in cotton productivity resulting from bacterial blight to the least possible, and we will have developed modern agriculture with new techniques and raised the level of Sudanese agriculture to keep pace with global agriculture.

5.2 RECOMMENDATION

Drones provide updated high-resolution data and images for many different purposes. These features can be taken advantage of by developing our research, as it can discover more than one disease by adding evidence for the diseases to be discovered, or by making drones help in treating sick plants by carrying pollen, or adding to the drone the task of knowing the terrain before Agriculture and whether the land is decertified, or the drone can be working autonomously.

Further, this can be done in a much more enhanced way by combining classification/detection systems within the drone control systems using a microprocessor. Which will lead to reducing effort and cost.

REFERENCE

[1] t. w. t. s. https://www.sas.com/en_us/insights/analytics/computer-vision.html#:~:text=Computer%20vision%20is%20a%20field, 4/10/2021.

[2] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A guide to convolutional neural networks for computer vision," *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, pp. 1-207, 2018.

[3] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in plant science*, vol. 7, p. 1419, 2016.

[4] Y. Bengio, Y. Lecun, and G. Hinton, "Deep learning for AI," *Communications of the ACM*, vol. 64, no. 7, pp. 58-65, 2021.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[6] I. G. Maglogiannis, *Emerging artificial intelligence applications in computer engineering: real word ai systems with applications in ehealth, hci, information retrieval and pervasive technologies*. Ios Press, 2007.

[7] C. M. Bishop, "Pattern recognition," *Machine learning*, vol. 128, no. 9, 2006.

[8] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85-117, 2015.

[9] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27-48, 2016.

[10] F. Chollet, *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929-1958, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[13] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154-171, 2013.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580-587.

[15] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440-1448.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91-99, 2015.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.

[18] https://www.arduino.cc/en/software, 4/2/2022.

[19] https://www.optimusdigital.ro/index.php?controller=attachment&id_attachment=451, 4/2/2022.

[20] https://www.rhydolabz.com/documents/26/BLDC_A2212_13T.pdf, 1/3/2022.

[21] https://dronenodes.com/pdb-power-distribution-board/, 6/3/2022.

[22] https://tamiyabase.com/articles/55-reviews/190-review-flysky-fs-ct6b-fs-r6b-2-4ghz-stick-radio-combo, 9/302022.
[23] https://www.u-blox.com/en/docs/UBX-15031086, 7/3/2022.
[24] https://www.ardupilot.co.uk, 4/2/2022.
[25] https://www.olimex.com/Products/IoT/ESP32/ESP32-CAM/, 4/2/2022.
[26] https://www.arduino.cc/en/software, 6/3/2022.
[27] https://ardupilot.org/planner/docs/mission-planner-overview.html, 6/3/2022.
[28] https://www.jetbrains.com/pycharm/, 6/3/2022.
[29] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Epfl, 2007.
[30] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in *2007 IEEE/RSJ international conference on intelligent robots and systems*, 2007: Ieee, pp. 153-158.
[31] M. B. T. Khalid, M. M. E.-M. A. Abbas, M. E. E.-d. Mohammed, S. A. S. M. Ahmed, and G. M. Superviser, "Farm Mapping Using Quadcopter," Sudan University Of Science & Technology, 2020.
[32] G. Ipate, G. Voicu, and I. Dinu, "Research on the use of drones in precision agriculture," *University Politehnica of Bucharest Bulletin Series,* vol. 77, no. 4, pp. 1-12, 2015.
[33] P. Chaudhary, "Ionic Framework," *International Research Journal of Engineering and Technology,* vol. 5, no. 05, pp. 3181-3185, 2018.
[34] https://capacitorjs.com/docs, 9/3/2022.
[35] M. SIMRAN M, "Leaf-Disease-Detection using Python (Open CV)," 2020.
[36] A. Sharma, B. Kochar, N. Joshi, and V. Kumar, "Breast Cancer Detection Using Deep Learning and Machine Learning: A Comparative Analysis," in *International Conference on Innovative Computing and Communications*, 2021: Springer, pp. 503-514.
[37] F. Chollet, "Keras: The python deep learning library, Keras," *IoKeras. io,* 2015.
[38] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," ed, 2015.
[39] https://developers.google.com/maps/documentation/javascript/overview, 9/3/2022.

APPENDIX

Appendix A Webpage

Appendix A 1 Result.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpClient, HttpHeaders , HttpRequest } from '@angular/common/http';
import {DataMarkers} from "../../providers/dataMarkers";
let satedImages = false;
interface location{
  lat: any;
  log: any;
}
@Component({
  selector: 'page-result',
  templateUrl: 'result.html',
  styleUrls: ['./result.scss'],
})
export class ResultPage implements OnInit {
  // Gets a reference to the list element
  segment = '0';
  files: File[] = [];
  rows: any[] = [];
  cols: any[] = [];
  locations: location[] = [];

  constructor(
    public http: HttpClient ,
    public markers: DataMarkers
  ) { }

  ngOnInit() {
}


getBase64(file): Promise<string> {
  return new Promise((resolve) => {
    var reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = function () {
      resolve(reader.result as string);
    };
    reader.onerror = function (error) {
      console.log('Error: ', error);
    };
  })
}
setSegemtnt(activeIndex: Promise<number>) {
  activeIndex.then((idnex) => {
    this.segment = idnex + '';
    if (idnex === 1){
      (document.getElementsByClassName('map-Result')[0] as HTMLElement).focus();
    }
  });
}
num(segment: string): number {
  return parseFloat(segment);
}
onSelect(event): void {
  console.log(event);
  if (event.addedFiles.length > 0){
    satedImages = true;
    this.files.push(...event.addedFiles);
    this.files.forEach((file, i) => {//loop in images i is the index of the image
      //locations array contains coordinates for every image
```

```
   this.locations.push( //add location coordinates
    {
     //The file name contains the coordinates, latitude and longitude, taken from the GPS
     lat: file.name.split(',',2) [0],// set latitude
     log: file.name.split(',',2) [1],// set longitude
    },)
  })

  this.files.forEach((file,i) => {
   //locations array contains coordinates for avery image
   this.locations.push(
    {
     //The file name contains the coordinates, length and width, taken from the GPS
     lat: file.name.split(',',2)[0],
     log: file.name.split(',',2)[1],
    },
   )
  })
 }

}
processImages(){
 this.files.forEach(  (file,i) => {
  this.postFile(file,'predict','',async (result) => {
   let img = await this.getBase64(file)
   this.markers.data.push({
    image: img,
    location: this.locations[i],
    status: result.status
   })
   if (i=== this.files.length -1){
    alert('Processing End Open Map To Show Result')
   }
```

```
        })
      })
    }

    postFile(file: File,urlSuffix: string, name: string, callback): void{
      const endpoint = urlSuffix;
      const formData: FormData = new FormData();
      formData.append("image", file);
      this.makeAPICallForFileUpload(endpoint , formData, callback);
    }
    makeAPICallForFileUpload(urlSuffix, params, callback: (response) => void): any {

      let URL_API = "http://127.0.0.1:5000/" + urlSuffix;
      return this.http.post(URL_API, params).subscribe(
        response => {
          callback(response);
        },
      );
    }
    onRemove(event): void {
      console.log(event);
      this.files.splice(this.files.indexOf(event), 1);
      if (this.files.length === 0){
        satedImages = false;
      }
    }
  }
```

Appendix A 2 Result.html

```
<ion-header translucent="true">

  <ion-toolbar>

    <ion-title>Home</ion-title>

  </ion-toolbar>

  <ion-toolbar>

      <ion-title>Load Images</ion-title>

  </ion-toolbar>

</ion-header>

<ion-content fullscreen="true">

  <div>

    <ngx-dropzone name="files[]" id="Images" aria-describedby="Images" aria-label="Images" aria-
labelledby="Images" accept="image/*" [expandable]="true" [maxFileSize]="1024*1024*10"
(change)="onSelect($event)">

      <ngx-dropzone-label>Select Images .. Taking from Drone  </ngx-dropzone-label>

      <ngx-dropzone-image-preview [removable]="true" (removed)="onRemove(f)" ngProjectAs="ngx-dropzone-
preview" *ngFor="let f of files" [file]="f"></ngx-dropzone-image-preview>

    </ngx-dropzone>

    <ion-button expand="full" (ionClick)="processImages()">Process Images</ion-button>

  </div>

</ion-content>
```

Appendix A 3 Map.ts

```
import { Component, ElementRef, Inject, ViewChild, AfterViewInit } from '@angular/core';
import { ConferenceData } from '../../providers/conference-data';
import { Platform } from '@ionic/angular';
import [20] from '@angular/common';
import { darkStyle } from './map-dark-style';
import {DataMarkers} from "../../providers/dataMarkers";
@Component({
 selector: 'page-map',
 templateUrl: 'map.html',
 styleUrls: ['./map.scss']
})
export class MapPage implements AfterViewInit {
 @ViewChild('mapCanvas', { static: true }) mapElement: ElementRef;

 constructor(
  @Inject(DOCUMENT) private doc: Document,
  public confData: ConferenceData,
  public platform: Platform,
  public markers: DataMarkers
 ) {}
 async ngAfterViewInit() {
  const appEl = this.doc.querySelector('ion-app');
  let isDark = false;
  let style = [];
  if (appEl.classList.contains('dark-theme')) {
   style = darkStyle;
  }
  const googleMaps = await getGoogleMaps(
   'AIzaSyBXdJogECiKximzAZjSpVoFGyNVVFz2yCM'
  );

```

```
let map;

this.confData.getMap().subscribe((mapData: any) => {
  const mapEle = this.mapElement.nativeElement;

  map = new googleMaps.Map(mapEle, {
    lat: 14.3118986,
    lng: 33.2081465,
    center: mapData.find((d: any) => d.center),
    zoom: 100,
    mapTypeId: "satellite",
    mapTypeControlOptions: {
      mapTypeIds: ["satellite"],
    },
    styles: style
  });
  const drawingManager = new googleMaps.drawing.DrawingManager({
    drawingMode: googleMaps.drawing.OverlayType.MARKER,
    drawingControl: true,
    drawingControlOptions: {
      position: googleMaps.ControlPosition.TOP_CENTER,
      drawingModes: [
        // googleMaps.drawing.OverlayType.MARKER,
        // googleMaps.drawing.OverlayType.CIRCLE,
        googleMaps.drawing.OverlayType.POLYGON,
        // googleMaps.drawing.OverlayType.POLYLINE,
        googleMaps.drawing.OverlayType.RECTANGLE,
      ],
    },
    markerOptions: {
      icon: "https://developers.google.com/maps/documentation/javascript/examples/full/images/beachflag.png",
    },
    circleOptions: {
```

```
    fillColor: "#ffff00",
    fillOpacity: 1,
    strokeWeight: 5,
    clickable: false,
    editable: true,
    zIndex: 1,
  },
});

drawingManager.setMap(map);

googleMaps.event.addListener(drawingManager, 'overlaycomplete', function(event) {
  console.log(event);
  if (event.type == 'rectangle') {
    let bounds = event.overlay.getBounds();
    let start = bounds.getNorthEast();
    let end = bounds.getSouthWest();
    let center = bounds.getCenter();
    let dis = haversine_distance(start,end)
    dvideRectuangle(bounds,googleMaps,map);
  }else if (event.type === 'polygon'){
    let locations = event.overlay.getPath().getArray();
    console.log(locations);
    locations.forEach((loc,i) => {
      console.log('point '+ (i + 1) +' lat = ' + loc.lat() + ' lng = ' + loc.lng())
      let dis;
      if (i === 0){
        dis = haversine_distance(locations[i],locations[locations.length - 1])
      }else {
        dis = haversine_distance(locations[i],locations[i-1])
      }
      console.log(dis)
    })
```

```
      let area = googleMaps.geometry.spherical.computeArea(event.overlay.getPath());
      console.log('area = ' + area)
    }
  });

  mapData.forEach((markerData: any) => {
    if(this.markers.data.length > 0){
      this.markers.data.forEach((marker,i) => {
        const infoWindow = new googleMaps.InfoWindow({
          content: `<h5 class="`+marker.status+`">${marker.status}</h5><img class="image-map" src="`+ marker.image+`" alt=""
        });
        addMarker({
          name: marker.status,
          lat: parseFloat(marker.location.lat),
          lng: parseFloat(marker.location.log),
          center: i===0
        },googleMaps,map,infoWindow)
      })
    }
  });

  googleMaps.event.addListenerOnce(map, 'idle', () => {
    mapEle.classList.add('show-map');
  });
});

const observer = new MutationObserver((mutations) => {
  mutations.forEach((mutation) => {
    if (mutation.attributeName === 'class') {
      const el = mutation.target as HTMLElement;
      isDark = el.classList.contains('dark-theme');
      if (map && isDark) {
        map.setOptions({styles: darkStyle});
```

```
          } else if (map) {
            map.setOptions({styles: []});
          }
        }
      });
    });
    observer.observe(appEl, {
      attributes: true
    });

  }
}

function getGoogleMaps(apiKey: string): Promise<any> {
  const win = window as any;
  const googleModule = win.google;
  if (googleModule && googleModule.maps) {
    return Promise.resolve(googleModule.maps);
  }

  return new Promise((resolve, reject) => {
    const script = document.createElement('script');
    script.src = `https://maps.googleapis.com/maps/api/js?key=${apiKey}&libraries=drawing&v=3.31`;
    script.async = true;
    script.defer = true;
    document.body.appendChild(script);
    script.onload = () => {
      const googleModule2 = win.google;
      if (googleModule2 && googleModule2.maps) {
        resolve(googleModule2.maps);
      } else {
        reject('google maps not available');
      }
    };
  });
}

function haversine_distance(mk1, mk2) {
  console.log(mk1,mk2)
  var R = 3958.8; // Radius of the Earth in miles
  var rlat1 = mk1.lat() * (Math.PI/180); // Convert degrees to radians
  var rlat2 = mk2.lat() * (Math.PI/180); // Convert degrees to radians
  var difflat = rlat2-rlat1; // Radian difference (latitudes)
  var difflon = (mk2.lng()-mk1.lng()) * (Math.PI/180); // Radian difference (longitudes)

  var d = 2 * R * Math.asin(Math.sqrt(Math.sin(difflat/2)*Math.sin(difflat/2)+Math.cos(rlat1)*Math.cos(rlat2)*Math.sin(difflon/2
  return d;
}

function dvideRectuangle(bounds,googleMaps,map){
  var southWest = bounds.getSouthWest();
  var northEast = bounds.getNorthEast();
  console.log(bounds);
  var numberOfParts = 4;

  var tileWidth = (northEast.lng() - southWest.lng()) / numberOfParts;
  var tileHeight = (northEast.lat() - southWest.lat()) / numberOfParts;
  for (var x = 0; x < numberOfParts; x++) {
    for (var y = 0; y < numberOfParts; y++) {
      var areaBounds = {
        north: southWest.lat() + (tileHeight * (y+1)),
        south: southWest.lat() + (tileHeight * y),
        east: southWest.lng() + (tileWidth * (x+1)),
        west: southWest.lng() + (tileWidth * x)
      };
      // console.log("Point : " + x + " ,  "  + y + " = "+ "\n"
      //   + "north:" + southWest.lat() + "+" + (tileHeight * (y+1)) +  " " + areaBounds.north  + "\n"
      //  + "south:" + southWest.lat() + "+" + (tileHeight * y) + " " + areaBounds.south + "\n"
      //   +  "east:" + southWest.lng() + "+" + (tileWidth * (x+1)) + " " + areaBounds.east + "\n"
      //   + "west:" + southWest.lng() + "+" + (tileWidth * x)  + " " + areaBounds.west)
      var area = new googleMaps.Rectangle({
        strokeColor: '#42a5a5',
        strokeWeight: 2,
        map: map,
        bounds: areaBounds

      });
      let Center = area.bounds.getCenter()
      console.log(Center.lat() + "," + Center.lng() + ".png")
    }
  }
}
function addMarker(markerData: marker,googleMaps, map,infoWindow) {
  const marker = new googleMaps.Marker({
    position: markerData,
    map,
    title: markerData.name,
    icon: markerData.name === 'diseased'?'assets/img/deisae.png':'assets/img/helthy.png'
  });
  marker.addListener('click', () => {
    infoWindow.open(map, marker);
  });
}
interface marker{
  "name": string,
  "lat": number,
  "lng": number,
  "center": boolean
}
```

Appendix A 4 Map.html

```
<ion-header>

  <ion-toolbar>

    <ion-buttons slot="start">

      <ion-menu-button></ion-menu-button>

    </ion-buttons>

    <ion-title>Map</ion-title>

  </ion-toolbar>

</ion-header>


<ion-content>

  <div #mapCanvas class="map-canvas"></div>

</ion-content>
```

# Appendix B	Python CNN model

## Appendix B 1	Train.py

```python
# import libraries
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt

print(keras.__version__)

train_data_path = "./dataset/training_set"
validation_data_path = "./dataset/val"
learning_rate = 0.0001
batch_size = 32
epochs = 100


def plotImages(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(20, 20))
    axes = axes.flatten()
    for img, ax in zip(images_arr, axes):
        ax.imshow(img)
    plt.tight_layout()
    plt.show()


# this is the augmentation configuration we will use for training
# It generate more images using below parameters
training_datagen = ImageDataGenerator(rescale=1. / 255,
                                      rotation_range=40,
                              width_shift_range=0.2,
                              height_shift_range=0.2,
                              shear_range=0.2,
                              zoom_range=0.2,
                              horizontal_flip=True,
                              fill_mode='nearest')

# this is a generator that will read pictures found in
# at train_data_path, and indefinitely generate
# batches of augmented image data
training_data = training_datagen.flow_from_directory(train_data_path,  # this is the target directory
                                      target_size=(150, 150),  # all images will be resized to 150x150
                                      batch_size=batch_size,
                                      class_mode='binary')  # since we use binary_crossentropy loss, we need binary labels
print(len(training_data))
print(training_data.class_indices)

# this is the augmentation configuration we will use for validation:
# only rescaling
valid_datagen = ImageDataGenerator(rescale=1. / 255)

# this is a similar generator, for validation data
valid_data = valid_datagen.flow_from_directory(validation_data_path,
                                  target_size=(150, 150),
                                  batch_size=batch_size,
                                  class_mode='binary')
print(len(valid_data))
images = [training_data[0][0][0] for i in range(5)]
plotImages(images)

model_path = './modal/ModalForSearch.h5'
checkpoint = ModelCheckpoint(model_path, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
```

```
callbacks_list = [checkpoint]

# Building cnn model
cnn_model = keras.models.Sequential([
  keras.layers.Conv2D(filters=32, kernel_size=3, input_shape=[150, 150, 3]),
  keras.layers.MaxPooling2D(pool_size=(2, 2)),
  keras.layers.Conv2D(filters=64, kernel_size=3),
  keras.layers.MaxPooling2D(pool_size=(2, 2)),
  keras.layers.Conv2D(filters=128, kernel_size=3),
  keras.layers.MaxPooling2D(pool_size=(2, 2)),
  keras.layers.Conv2D(filters=256, kernel_size=3),
  keras.layers.MaxPooling2D(pool_size=(2, 2)),

  keras.layers.Dropout(0.5),
  keras.layers.Flatten(),  # neural network beulding
  keras.layers.Dense(units=128, activation='relu'),  # input layers
  keras.layers.Dropout(0.1),
  keras.layers.Dense(units=256, activation='relu'),
  keras.layers.Dropout(0.25),
  keras.layers.Dense(units=2, activation='softmax')  # output layer
])

# compile cnn model
cnn_model.compile(optimizer=Adam(lr=learning_rate), loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
cnn_model.summary()
keras.utils.plot_model(cnn_model, to_file="my_model.png", show_shapes=True)
# train cnn model
trainedModel = cnn_model.fit(training_data,
                      epochs=100,
                      verbose=1,
                      validation_data=valid_data,
                      callbacks=callbacks_list)  # time start 16.06

```

```
from keras.models import save_model

model = save_model(cnn_model, model_path)  # saving model to './modal/ModalForSearch.h5'
# # summarize history for accuracy
plt.plot(trainedModel.history['accuracy'])
plt.plot(trainedModel.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(trainedModel.history['loss'])
plt.plot(trainedModel.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
#
print(trainedModel.history)
```

Appendix B 2 App.py

```
# Import necessary libraries
from flask import Flask, render_template, request
from flask_cors import CORS, cross_origin
import numpy as np
import os
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model
from keras.utils.vis_utils import plot_model

# load model
# from werkzeug.datastructures import FileStorage

model = load_model("model/ModalForSearch.h5")
plot_model(model, to_file='model_plot.png',  show_layer_names=True)

def pred_cot_dieas(cott_plant):
  img = load_img(cott_plant, target_size=(150, 150))
  img = img_to_array(img) / 255
  img = np.expand_dims(img, axis=0)
  result = model.predict(img).round(3)
  pred = np.argmax(result)  # get the index of max value
  # return pred
  # # {'diseased cotton plant': 0,  'fresh cotton plant': 1}
  if pred == 0:
    return " diseased cotton plant", 'disease_plant.html'  # if index 0 burned leaf
  else:
    return "healthy cotton plant", 'healthy_plant.html'  # if index 3


# ------------>>pred_cot_dieas<<--end
```

```python


# Create flask instance
app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'

# render index.html page
@app.route("/", methods=['GET', 'POST'])
def home():
    return render_template('index.html')

    # Python3 code here creating class


class predResult:
    def __init__(self, file_path, pred):
        self.file_path = file_path
        self.pred = pred


# get input image from client then predict class and render respective .html page for solution
@app.route("/predict", methods=['GET', 'POST'])
@cross_origin()
def predict():
    file_Results = []
    if request.method == 'POST':
        print('started')
        file = request.files['files[]']  # fet input
        filename = file.filename
        print("@@ Input posted = ", filename)

        file_path = os.path.join('static/user uploaded', filename)
        file.save(file_path)

        print("@@ Predicting class......")
        pred, output_page = pred_cot_dieas(cott_plant=file_path)
        return render_template(output_page, pred_output=pred, user_image=file_path)


# For local system & cloud
if __name__ == "__main__":
    app.run(threaded=False, )
```