BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 416 (January 2022) **A1**
Microprocessor and Embedded Systems Laboratory

Final Project Report

IoT Based Smart Agricultural System

Evaluation Form:

STEP	DESCRIPTION	MAX	SCORE
1	Report (Format, Reference)	10	
2	Design Method and Complete Design (Hardware Implementation)	15	
3	Video Demonstration	10	
4	Novelty of Design	15	
5	Project Management and Cost Analysis	10	
6	Considerations to Public Health and Safety, Environment and Cultural and Societal Needs	10	
7	Assessment of Societal, Health, Safety, Legal and Cultural issues relevant to the solution	10	
8	Evaluation of the sustainability and impact of designed solution in societal and environmental contexts	10	
9	Individual Contribution (Viva)	20	
10	Team work and Diversity	10	
	TOTAL	120	

Signature of Evaluator: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and Write your name in your own handwriting. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Full Name: Sheikh Munim Hussain **Student ID: 1706008**	**Full Name: Shafin Shadman Ahmed** **Student ID: 1706020**
Full Name: Md. Jahidul Hoq Emon **Student ID: 1706017**	**Full Name: Azazul Islam** **Student ID: 1706033**

Table of Contents

1 Abstract

The main objective of this project was to design a smart agricultural system based on the Internet of Things. This system's purpose is to give necessary information to farmers so that they can act accordingly. In this system, different sensors were used to detect changes in temperature, pressure, humidity, and water levels, and to detect if plants are diseased or not. Data were presented in such a way that farmers can get an idea of what to do in certain circumstances and take necessary steps, and comfortably continue farming if there is no danger. This project aims to further ease farming so that farmers can easily take decisions based on obtained data. Farmers can be helped to get access to the Internet and digitization, and loss of crops due to diseases can be prevented.

2 Introduction

The importance of agriculture in the history of mankind cannot be ignored. Agriculture can be considered a keystone of human lives. Major civilizations arose based on agriculture, and even in modern times, it is agriculture that effectively runs the world. This is because agriculture provides for the most basic need of people, food. Like many countries of the world, agriculture is one of the most vital sectors of our country too. It goes by saying that agriculture is the backbone of Bangladesh's economy.

Since ancient times, there have been tremendous advances in agricultural technologies. Human labor in agriculture was a must in older days, at present human involvement is less direct than before. The advent of modern technology has facilitated farming in various ways and eased agriculture to a great extent. However, the trend of technological development in the agricultural sector of Bangladesh has always been minimal. It is evident that our agriculture is still strongly dependent on older, traditional ways of farming. Some countries which have far less arable land than ours have invested in technological advancement in Bangladesh, and as a result their agricultural output is far greater compared to ours. But there is still much scope to develop our agriculture in the technological side. Keeping such things in mind, this project of ours has been conducted.

Smart Farming, based on the Internet of Things (IoT), aims at further easing the human involvement in farming. Farmers will be able to get an idea of the current state of their lands and their crops from home. Depending on the data they receive, they can take further steps to address if there are any problems. In this system, farmers don't have to inspect their lands on their own, which reduces their physical labor to a great extent. Also, in case of emergencies when it is impossible for farmers to go to their lands, such a system can come into handy. This system will also detect whether the crops are diseased or not. If the diseased plants can be detected earlier, it will be easier for farmers to take preventive steps quickly and either prevent or reduce losses.

3 Design

3.1 Design Method

Total implementation has five segments

1. Data Monitoring
2. Application of Monitoring (Irrigation System)
3. Real-time Weather Monitoring
4. Security System
5. Leaf Image Analysis using Machine Learning and Automation

3.1.1 Data Monitoring

In data monitoring, the following data related to farming are taken and uploaded to web servers. The sensors used to collect the data are also listed:
1. Humidity (DHT11 Sensor)
2. Soil Moisture (HL69/YL69 Soil Moisture Sensor)
3. Temperature (DHT11 Sensor)
4. Soil Temperature (DS18B20 Sensor Probe)
In this project, a firmware named ESP8266 based on NodeMCU IoT platform has been used to collect all these data from these sensors. Thereafter, these data are sent to Adafruit IO for visualization. Both instant and periodic changes can be visualized live using Adafruit IO. A free version of Adafruit IO has been used for this purpose.

3.1.2 Application of Monitoring (Irrigation System)

One application of monitoring deserves its special description, and that is the irrigation system. Irrigation is performed using the following apparatuses:
1. Water pump (Motor)
2. Soil Moisture (HL69/YL69 Sensor)
The irrigation system has been operated on threshold percentage of soil moisture sensor result. If the soil moisture level is below the threshold level, the sensor will send data to the server and the motor will be turned on. If the soil moisture reaches the threshold level, the pump will turn off automatically, but there will be a little water flow since there is a short lag between sending the command of shutting down the motor and its execution.

3.1.3 Real-time Weather Monitoring

Another application of monitoring is the real-time weather monitoring system, which is performed using the following apparatuses:
1. Rain (Rain Sensor)
2. Absolute Pressure (BMP180 Sensor)
Data for absolute pressure, sea-level pressure and altitude of a particular area are collected from the BMP180 barometric Sensor. The current location's altitude is given to the sensor as

an input. A rain sensor is used to detect rainfall. The rain sensor is sensitive enough to detect even a slight rainfall. Signal is sent to the farmer even if there is some drizzle. The data for real-time weather monitoring are updated and visualized on Blynk server. Blynk allows data to be visualized live.

3.1.4 Security System

In addition to data monitoring, some security measures have also been taken, these include:
1. MQ135 gas sensor has been used to detect toxic and harmful gases in the environment. Even if there is a fire, the gas sensor will detect the smoke and send data to the server, which will be then sent to the farmer.
2. ESP32 camera module with a camera has been used to get a live-stream view of the farm. The live-stream system works like a close-circuit camera; it can provide a baseline for surveillance and security of the whole agricultural system.
3. The captured video can be used to undergo through image processing. For this, machine learning or artificial intelligence can be used to visualize a part of the plant and decide if the said plant is healthy or not. Predictions can also be made from these captures.

3.1.5 Leaf Image Analysis using Machine Learning and Automation

In previous section, ESP32 cam was used for implementation of a low budget surveillance system. Apart from that, availability of live stream creates an opportunity to analysis the output of the stream. A whole range of possibilities appear and one of them could be capturing the video frame and use it as an input for image processing or machine learning model.
In fact, working on the leaf image as a tool for detection of different disease like late blight, early blight or leaf spot is a matter of interest for a while and advancement of machine learning opens a new scope of opportunity here. But most of the available examples deal with this problem by taking an image of a leaf at first and then making a prediction through the model. Also, the available dataset provides only single leaf image which makes it difficult for the model to work on surveillance camera feed of multiple images. Lack of automation could result in a matter of inconvenience for rural farmers. These issues were also addressed here and a lookout for solutions was tried also.

3.1.5.1 Dataset

Training a machine learning model requires enough data so that the model can learns the feature. In this project, two different datasets were used for this purpose:
i) Plantvillage Dataset
ii) Cassava leaf image Dataset
Both datasets are available in Kaggle and used in several competitions and academic research purposes.

Plantvillage Dataset

This dataset contains 54,035 images of 14 plant species under 38 classes. Due to lack of hardware resources, only 5 classes of a single plant species (Tomato) were used. These five classes and number of images for train and test set are:

Class	Train	Test
Early blight	800	200
Late Blight	1527	382
Septoria Leaf Spot	1417	354

Two Spotted Spider Mite	1341	336
Healthy	1273	318
Total	6358	1590

Cassava Leaf Image Dataset

As traits of healthy leaf image is almost like different species of healthy leaf image from cassava leaf image dataset was integrated to train the model for detecting healthy leaf from multiple leaves image. As there is no available dataset that contains multiple leaves image with disease, this remains as a limitation for the model.

3.1.5.2 Model

A pre-trained convolutional neural network architecture VGG16 model was used to extract features from the image.

Architecture of the model is given below:

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 256, 256, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 256, 256, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 128, 128, 64)      0
_________________________________________________________________
```

```
block2_conv1 (Conv2D)        (None, 128, 128, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 128, 128, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 64, 64, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 64, 64, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 64, 64, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 64, 64, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 32, 32, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 32, 32, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 16, 16, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 16, 16, 512)       2359808
```

```
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 8, 8, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
```

Later these features were passed as an input to two different machine learning algorithms.

i) Random Forest Classifier

ii) Xgboost Classifier

The one with the best outcome was used later.

3.1.5.3 Method

After collecting the image and setting up models, images were passed through a pre-processing stage. A (256,256) dimension was used for the purpose. Then extracted features from these images using VGG16 model was used as training medium for machine learning algorithm. Prepared model was later used for prediction and automation. Also, to keep the farmers constantly updated about the health status, outcome of the model was continuously updated at Adafruit IO server.

3.1.5.4 Code

Preparation of Model

```
# importing library


from sklearn.ensemble import RandomForestClassifier
import numpy as np
import glob
import seaborn as sns
import os
import random
import cv2
from sklearn import preprocessing
from tensorflow.keras.applications.vgg16 import VGG16
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import xgboost as xgb
import joblib
import pickle


# Creating static and local variables


SIZE = 256
SEED_TRAINING = 121
SEED_TESTING = 197
SEED_VALIDATION = 164
CHANNELS = 3
n_classes = 5
EPOCHS = 50
BATCH_SIZE = 16
input_shape = (SIZE, SIZE, CHANNELS)


#Training


def training(path) :


    train_images = []
    train_labels = []


    path = path + '\*'
    for directory_path in glob.glob(path) :
        label = directory_path.split('\\')[-1]


        for img_path in
glob.glob(os.path.join(directory_path, '*.JPG')) :
            img = cv2.imread(img_path,
cv2.IMREAD_COLOR)
            img = cv2.resize(img, (SIZE, SIZE))


            cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            train_images.append(img)
            train_labels.append(label)



    train_data = list(zip(train_images, train_labels))
    random.seed(SEED_TRAINING)
    random.shuffle(train_data)
    train_images, train_labels = zip(*train_data)


    train_images = np.array(train_images)
    train_labels = np.array(train_labels)


    train_images = train_images / 255.0
    return train_images, train_labels


#Testing


def testing(path) :
    test_images = []
    test_labels = []


    path = path + '\*'
    for directory_path in glob.glob(path) :
        labels = directory_path.split('\\')[-1]
        for img_path in
glob.glob(os.path.join(directory_path, '*.JPG')) :
            img = cv2.imread(img_path,
cv2.IMREAD_COLOR)
            img = cv2.resize(img, (SIZE, SIZE))
```

```
# random forest


rfc = RandomForestClassifier()
rfc.fit(X_train_features, y_train)
rfc_pred = rfc.predict(X_test_features)



# inversing le transforms
rfc_pred = le.inverse_transform(rfc_pred)


# accuracy
print('Accuracy of Random Forest : ',
metrics.accuracy_score(y_test_labels, rfc_pred))


# xgboost
model = xgb.XGBClassifier(use_label_encoder = False)
model.fit(X_train_features, y_train)
xgb_pred = model.predict(X_test_features)


# inversing le transforms
xgb_pred = le.inverse_transform(xgb_pred)


# accuracy
print('Accuracy of XGBOOST : ',
metrics.accuracy_score(y_test_labels, xgb_pred))


# confusion matrix
cm = confusion_matrix(y_test_labels, rfc_pred)
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
sns.heatmap(cm, annot = True).set_title('Random Forest
Preformance')


cm = confusion_matrix(y_test_labels, xgb_pred)
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
sns.heatmap(cm, annot = True).set_title('XGB
Preformance')


# classification report
print('Random Forest Report : ')
print(classification_report(y_test_labels, rfc_pred))


print('XGB Report : ')
print(classification_report(y_test_labels, xgb_pred))


# save the model
pickle.dump(model, open("model1.pkl", "wb"))


# load the model
working_model = pickle.load(open("model1.pkl", "rb"))


# processing image
def pred_img_proc(img_path):
    p_img = cv2.imread(img_path,cv2.IMREAD_COLOR)


    if p_img is None:
        print('Wrong path:', img_path)
    else:
        p_img = cv2.resize(p_img, dsize=(256,256))
    p_img=p_img[np.newaxis,:, :, :]
    print(np.shape(p_img))
    p_img=p_img/255.0
    p_feature_extractor_test = vgg_model.predict(p_img)
    print(np.shape(p_feature_extractor_test))
    p_features_test =
p_feature_extractor_test.reshape(p_feature_extractor_te
st.shape[0], -1)
    print(np.shape(p_features_test))
    return p_features_test


pimg1=pred_img_proc(path)


# label
def show_label(v):
        if v==0:
            return 'Tomato___Early_blight'
        if v==1:
```

```
            test_images.append(img)
            test_labels.append(labels)


    test_data = list(zip(test_images, test_labels))
    random.seed(SEED_TESTING)
    random.shuffle(test_data)
    test_images, test_labels = zip(*test_data)
    test_images = np.array(test_images)
    test_labels = np.array(test_labels)


    test_images = test_images / 255.0
    return test_images, test_labels


# preprocessing training and testing images


X_test, y_test_labels = training(r'test_path)
X_train, y_train_labels = training(r'train_path')


# encoding labels from text to integer


le = preprocessing.LabelEncoder()
le.fit(y_train_labels)
train_label_encoded = le.transform(y_train_labels)
le.fit(y_test_labels)
test_label_encoded = le.transform(y_test_labels)


# extracting original labels
labels = dict(zip(le.classes_,range(len(le.classes_))))


# aliasing
y_train, y_test = train_label_encoded,
test_label_encoded


# loading vgg16


vgg_model = VGG16(weights = 'imagenet',  include_top =
False, input_shape = (SIZE, SIZE, 3))
for layer in vgg_model.layers :
    layer.trainable = False
vgg_model.summary()


# feature extraction
feature_extractor = vgg_model.predict(X_train)


# perform operation on train and test set


feature_extractor_test = vgg_model.predict(X_test)
features_test =
feature_extractor_test.reshape(feature_extractor_test.s
hape[0], -1)
X_test_features = features_test


features =
feature_extractor.reshape(feature_extractor.shape[0], -
1)
X_train_features = features
```

```
            return 'Tomato___Late_blight'
        if v==2:
            return 'Tomato___Septoria_leaf_spot'
        if v==3:
            return 'Tomato___Spider_mites Two-
spotted_spider_mite'
        if v==4:
            return 'Tomato___healthy'


# prediction
wm_pred = working_model.predict(pimg1)
print(wm_pred)
print(show_label(wm_pred))


# adafruit update
from Adafruit_IO import RequestError, Client, Feed
username= __
io_key= __


aio=Client(username,io_key)
test=aio.feeds('disease')



# pre processing


def pred_img_proc1(img):
    #p_img = cv2.imread(img_path,cv2.IMREAD_COLOR)


    #if p_img is None:
     #   print('Wrong path:', img_path)
    #else:
    p_img = cv2.resize(img, dsize=(256,256))
    p_img=p_img[np.newaxis,:, :, :]
    print(np.shape(p_img))
    p_img=p_img/255.0
    p_feature_extractor_test = vgg_model.predict(p_img)
    print(np.shape(p_feature_extractor_test))
    p_features_test =
p_feature_extractor_test.reshape(p_feature_extractor_te
st.shape[0], -1)
    print(np.shape(p_features_test))
    return p_features_test


# frame capture and automation
url = r'http://192.168.31.195/capture'


while True:
    img_resp = urlopen(url)
    imgnp = np.asarray(bytearray(img_resp.read()),
dtype="uint8")
    img = cv2.imdecode(imgnp, -1)
    cv2.imshow("Camera", img)
    pimg1=pred_img_proc1(img)
    wm_pred = working_model.predict(pimg1)
    print(wm_pred)
    print(show_label(wm_pred))
    aio.send_data(test.key,show_label(wm_pred))


    if cv2.waitKey(10000) & 0xFF == ord('q'):
        break
```

Table: Code for Model Training and Detection via Camera

3.2 Circuit Diagram

Here we used two boards for the whole system. Circuit diagram of board 1 is given below:

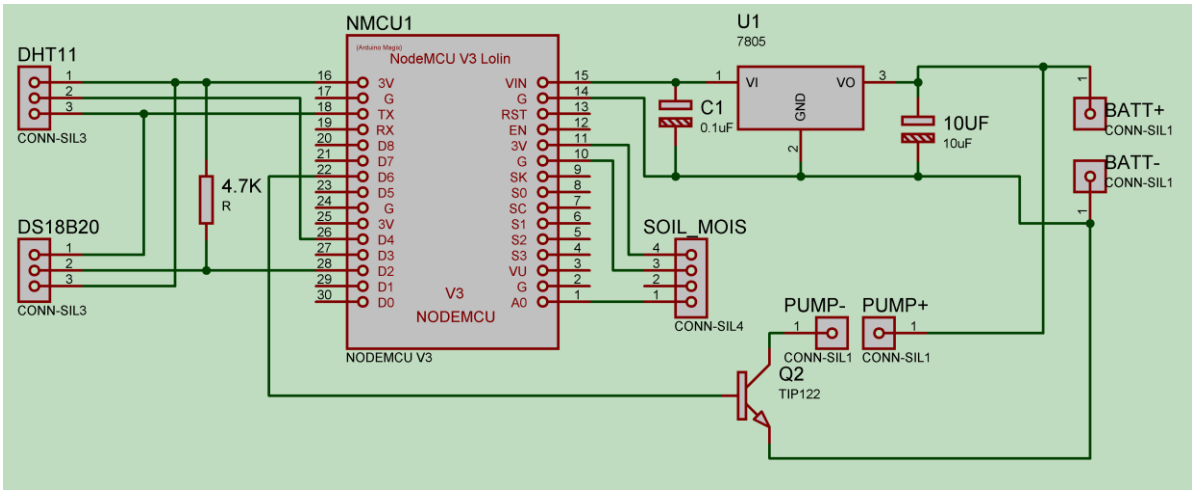

Figure 1: Circuit Design for Board 1

It consists of Soil Moisture Sensor (Soil Moisture), DHT 11 (Temperature and Humidity), DS18B20 (Soil Temperature sensor). It also drives water pump.
Next, we will see the circuit diagram of board 2:

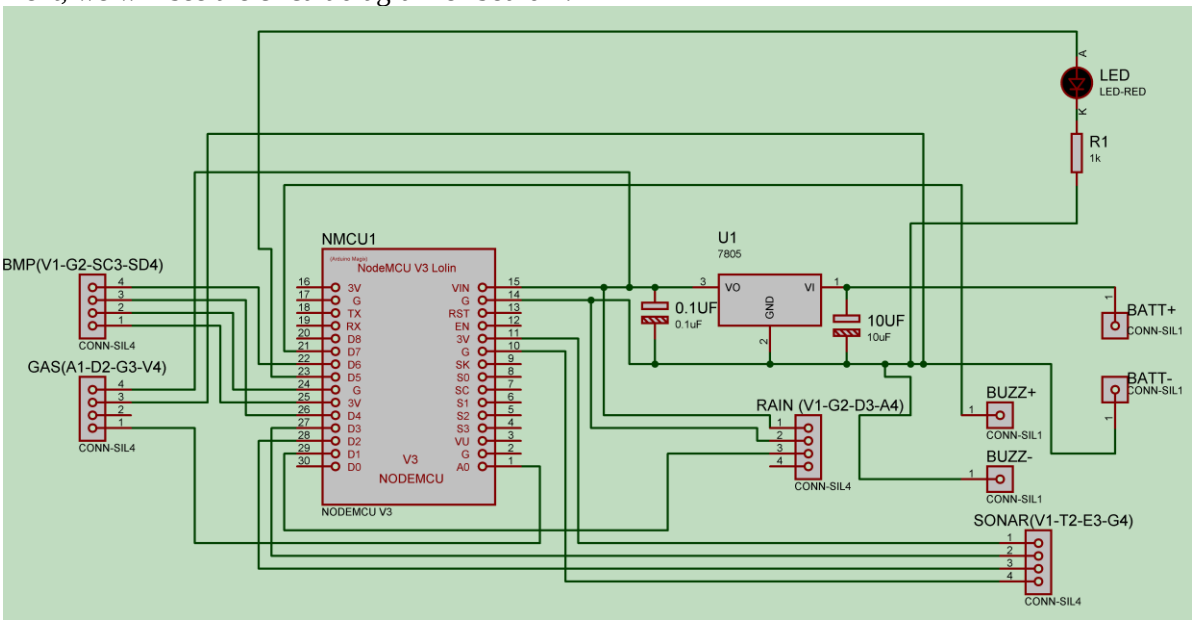

Figure 2: Circuit Diagram for Board 2

This board has BMP 180 (Pressure and Altitude Sensor), HC-SR04 SONAR (as Water Level Detector in this project), MQ-138 (Gas Sensor) and Rain Alarm sensor. It also has LED to show safe working environment and buzzer to convey alarm.

3.3 Full Source Code of Firmware

```
#include "esp_camera.h"
#include <WiFi.h>

#define CAMERA_AI_THINKER // Has PSRAM


#include "camera_pins.h"

const char* ssid = "Stop Pinching";
const char* password = "roomkey8";

void Camera_Server();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;


  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#if defined(CAMERA_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif
```

```
// camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x",
err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();

  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just
a bit
    s->set_saturation(s, -2); // lower the saturation
  }

  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_WIDE) || defined(CAMERA_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Camera_Server();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {

  delay(10000);
}
```

Table: Source Code for the Camera Live Feeding

```
#define BLYNK_TEMPLATE_ID "TMPLeXlMitUT"
#define BLYNK_DEVICE_NAME "Rain and Gas"
#define BLYNK_AUTH_TOKEN
"lEXq6LLFAD2ARb_SB2K8uPRYXh2aCaho"

char ssid[] = "Stop Pinching";
char pass[] = "roomkey8";

#define MQ2_SENSOR    A0 //A0
#define RAIN_SENSOR   5  //D1
#define GREEN_LED     14 //D5
#define RED_LED       13 //D7
#define WIFI_LED      16 //D0
#define TRIGGERPIN D3
#define ECHOPIN    D2
#define I2C_SCL D4
#define I2C_SDA D6

#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>
#include <Adafruit_Sensor.h>
Adafruit_BMP085 bmp;

float dst,bt,bp,ba;
char dstmp[20],btmp[20],bprs[20],balt[20];
bool bmp085_present=true;

int MQ2_SENSOR_Value = 0;
int RAIN_SENSOR_Value = 0;
bool isconnected = false;
char auth[] = BLYNK_AUTH_TOKEN;

#define VPIN_BUTTON_1    V1
#define VPIN_BUTTON_2    V2

long tankDepth = 11;

BlynkTimer timer;

void checkBlynkStatus() { // called every 2 seconds by
SimpleTimer
  getSensorData();
  isconnected = Blynk.connected();
  if (isconnected == true) {
    digitalWrite(WIFI_LED, LOW);
    sendSensorData();
  }
  else{
    digitalWrite(WIFI_LED, HIGH);
    Serial.println("Blynk Not Connected");
  }
}

void getSensorData()
{
  MQ2_SENSOR_Value = map(analogRead(MQ2_SENSOR), 0,
1024, 0, 100);
  RAIN_SENSOR_Value = digitalRead(RAIN_SENSOR);
  if (MQ2_SENSOR_Value > 50 ){
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(RED_LED, HIGH);
  }
  else if (RAIN_SENSOR_Value == 0 ){
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(RED_LED, HIGH);
  }
  else{
    digitalWrite(GREEN_LED, HIGH);
    digitalWrite(RED_LED, LOW);
  }

}
void sendSensorData()
{
  Blynk.virtualWrite(VPIN_BUTTON_1, MQ2_SENSOR_Value);
  if (MQ2_SENSOR_Value > 50 )
  {
    Blynk.logEvent("gas", "Gas Detected!");

  Blynk.run();
  timer.run();
}
  if (RAIN_SENSOR_Value == 0 )
  {
    Blynk.logEvent("rain", "Water Detected!");
    Blynk.virtualWrite(VPIN_BUTTON_2, "Water
Detected!");
  }
  else if (RAIN_SENSOR_Value == 1 )
  {
    Blynk.virtualWrite(VPIN_BUTTON_2, "No Water
Detected.");
  }
    if (!bmp.begin())
  {
        Serial.println("Could not find a valid BMP085
sensor, check wiring!");
        while (1) {}
        }


  float bp =  bmp.readPressure()/100;
  float ba =  bmp.readAltitude();
  float bt =  bmp.readTemperature();
  float dst = bmp.readSealevelPressure()/100;


Blynk.virtualWrite(V5, bp);
Blynk.virtualWrite(V4, ba);
Blynk.virtualWrite(V3, bt);
Blynk.virtualWrite(V6, dst);
Serial.print("Pressure: ");
Serial.print(bp);
Serial.print("Temperature: ");
Serial.print(bt);
Serial.print("Altitude: ");
Serial.print(ba);
Serial.print("Sea: ");
Serial.print(dst);
}

void setup()
{
  Serial.begin(9600);

  pinMode(TRIGGERPIN, OUTPUT);
  pinMode(ECHOPIN, INPUT);
  pinMode(MQ2_SENSOR, INPUT);
  pinMode(RAIN_SENSOR, INPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(WIFI_LED, OUTPUT);

  digitalWrite(GREEN_LED, LOW);
  digitalWrite(RED_LED, LOW);
  digitalWrite(WIFI_LED, HIGH);

  WiFi.begin(ssid, pass);
  Wire.begin(I2C_SDA, I2C_SCL);
  timer.setInterval(2000L, checkBlynkStatus); // check
if Blynk server is connected every 2 seconds
  Blynk.config(auth);
  delay(1000);
}
void loop()
{
  long duration, distance;
  digitalWrite(TRIGGERPIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIGGERPIN, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIGGERPIN, LOW);
  duration = pulseIn(ECHOPIN, HIGH);
  distance = (duration/2) / 29.1;
  Serial.print(distance);
  Serial.println("Cm");
  double level = tankDepth - distance;
  if (level > 0) {
    long percentage = ((level / tankDepth)) * 100;
    Blynk.virtualWrite(V0, percentage);
  }
  else Blynk.virtualWrite(V0, 0);  delay(500);
}
```

Table: Source Code for the Rain, Gas, SONAR, BMP sensor

```
#include <ESP8266WiFi.h>
#include <DallasTemperature.h>
#include <OneWire.h>
#include "DHT.h"
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <ArduinoJson.h>

const char *ssid =  "Stop Pinching";     // Enter your
WiFi Name
const char *pass =  "roomkey8"; // Enter your WiFi
Password
WiFiClient client;

#define SERVER "io.adafruit.com"
#define PORT 1883
#define NAME "SZubeen" // Your Adafruit IO Username
#define PASSWORD "5fa661ffd59840029fd6c8909150e27a" //
Adafruit IO AIO key


unsigned long Last_Connected_Time = 10 * 60 * 1000;
// last time you connected to the server, in
milliseconds
const unsigned long Interval_Posting = 10 * 60 * 1000;
// posting interval of 10 minutes  (10L * 1000L; 10
seconds delay for testing)
const int MOISTURE_PIN = A0;  // moisteure sensor pin
const int MOTOR_PIN = D0;
float moisturePercentage;              //moisture
reading
int temperature, humidity, soil_temperature;

#define ONE_WIRE_BUS 4   //D2 pin of nodemcu
#define DHTTYPE DHT11   // DHT 11
#define dht_dpin D4

DHT dht(dht_dpin, DHTTYPE);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

const unsigned long Interval = 50000;
unsigned long Previous_Time = 0;

//Set up the feed you're publishing to

Adafruit_MQTT_Client mqtt(&client, SERVER, PORT, NAME,
PASSWORD);
Adafruit_MQTT_Publish Moisture =
Adafruit_MQTT_Publish(&mqtt,NAME "/f/Moisture");  //
Moisture is the feed name where you will publish your
data
Adafruit_MQTT_Publish Temperature =
Adafruit_MQTT_Publish(&mqtt,NAME "/f/Temperature");
Adafruit_MQTT_Publish Humidity =
Adafruit_MQTT_Publish(&mqtt,NAME "/f/Humidity");
Adafruit_MQTT_Publish soil_temperature =
Adafruit_MQTT_Publish(&mqtt,NAME
"/f/soil_temperature");

//Set up the feed you're subscribing to

Adafruit_MQTT_Subscribe Pump =
Adafruit_MQTT_Subscribe(&mqtt, NAME "/f/Pump");

void setup()
{
  Serial.begin(9600);
  delay(10);
  dht.begin();
  sensors.begin();
  mqtt.subscribe(&Pump);
  pinMode(MOTOR_PIN, OUTPUT);
  digitalWrite(MOTOR_PIN, LOW); // keep motor off
initally
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");              // print ... till
not connected
  }
if (moisturePercentage > 50) {
  digitalWrite(MOTOR_PIN, LOW);          // turn off
mottor
 }

 temperature = dht.readTemperature();
 humidity = dht.readHumidity();
 Serial.print("Temperature: ");
 Serial.print(temperature);
 Serial.println();
 Serial.print("Humidity: ");
 Serial.print(humidity);
 Serial.println();
 sensors.requestTemperatures();
 soil_temperature = sensors.getTempCByIndex(0);
 Serial.println("Soil Temperature: ");
 Serial.println(soil_temperature);

 if (Current_Time - Previous_Time >= Interval) {
    if (! Moisture.publish(moisturePercentage)) //This
condition is used to publish the Variable
(moisturePercentage) on adafruit IO. Change thevariable
according to yours.
         {
           }
    if (! Temperature.publish(temperature))
         {
           }
    if (! Humidity.publish(humidity))
         {
         //delay(30000);
          }
    if (! soil_temperature.publish(soil_temperature))
        {
          }
    if (! WeatherData.publish(icon))
        {
          }
    Previous_Time = Current_Time;
}

Adafruit_MQTT_Subscribe * subscription;

while ((subscription = mqtt.readSubscription(5000)))
//Dont use this one until you are conrolling something
or getting data from Adafruit IO.
     {
    if (subscription == &Pump)
      {
      //Print the new value to the serial monitor
      Serial.println((char*) Pump.lastread);
      if (!strcmp((char*) Pump.lastread, "ON"))
       {
        digitalWrite(MOTOR_PIN, HIGH);
       }
     if (!strcmp((char*) Pump.lastread, "OFF"))
       {
        digitalWrite(MOTOR_PIN, LOW);
       }
     }
    }

  delay(9000);
 // client.publish(WeatherData, icon)
}

void MQTT_connect()
{
  int8_t ret;
  // Stop if already connected.
  if (mqtt.connected())
  {
    return;
  }

  uint8_t retries = 3;

  while ((ret = mqtt.connect()) != 0) // connect will
return 0 for connected
  {
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
       if (retries == 0)
       {
         // basically die and wait for WDT to reset me
```

```
  Serial.println("");
  Serial.println("WiFi connected");
}

void loop()
{
 unsigned long Current_Time = millis();
 MQTT_connect();

 if (millis() - Last_Connected_Time > Interval_Posting)
{
    // note the time that the connection was made:
    Last_Connected_Time = millis();
    HTTP_Request();
  }

//}


  moisturePercentage = ( 100.00 -
( (analogRead(MOISTURE_PIN) / 1023.00) * 100.00 ) );
  Serial.print("Soil Moisture is  = ");
  Serial.print(moisturePercentage);
  Serial.println("%");

 if (moisturePercentage < 40) {
  digitalWrite(MOTOR_PIN, HIGH);         // tun on
motor
 }
```

```
        while (1);
      }
  }
}
```

Table: Source Code for the Soil and Temperature related sensors

4 Implementation

4.1 Description

4.1.1 PCB Description

The PCB design and implementation of design for board 1 is shown below:

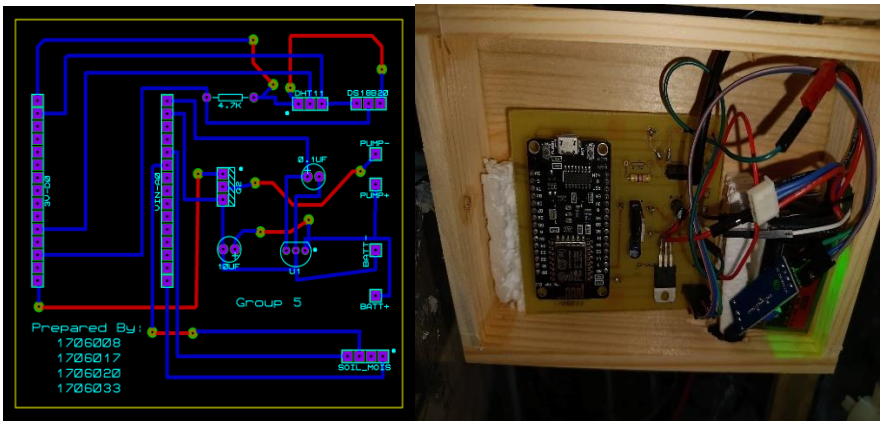

Figure 3: (Left) PCB Layout and (Right) Implementation of Design for Board 1

We showed the implementation of PCB in the box where we set the PCB.
Now we will show the PCB layout and implementation for Board 2:

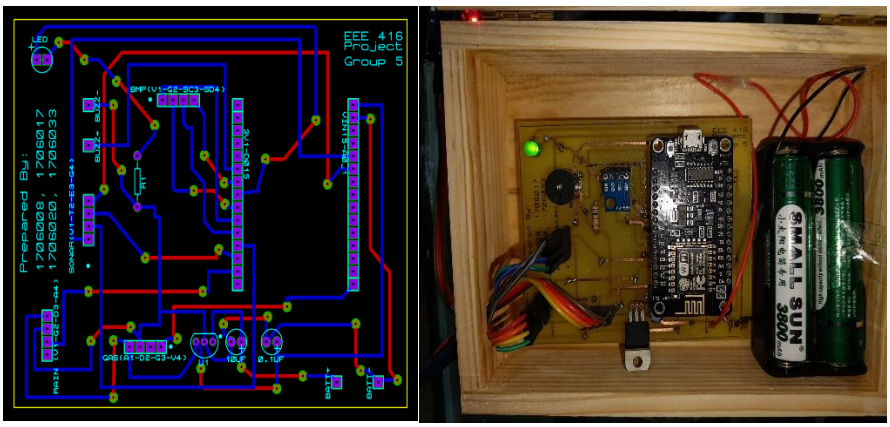

Figure 4: (Left) PCB Layout and (Right) Implementation of Design for Board 2

4.1.2 Sensors Description

Adafruit IO: Data Monitoring

Adafruit Industries is an American open-source hardware company which designs, manufactures and sells electronic products, tools and components, and specializes in data monitoring interfaces like their own Adafruit IO. Adafruit IO helps a user to visualize the changes in data by means of widgets like gauges, vertical and horizontal bars, graphs etc. Making a desired interface and interpreting data from the interface is quite easy in Adafruit IO.

In this experiment, Adafruit IO has been used to monitor the changes in data related to soil temperature, environment temperature, humidity and soil moisture. The instantly obtained data are shown in gauges and bars, and data for a timespan of one month can be accessed from graphs. From all these data, an idea about the soil temperature, environment temperature, humidity and soil moisture of a particular day can be obtained. The farmer can use these data to predict the conditions for the upcoming days and be prepared accordingly.

Sensors

DHT11 Temperature and Humidity Sensor: This sensor determines both temperature and humidity of a certain place. Due to such features, this sensor has a large demand for projects.

Figure 5: DHT11 Sensor

We used a DHT11 sensor in our project. The temperature and humidity values obtained were visualized using Adafruit IO. No preset values were used here because different crops have different requirements for optimum temperature and humidity. If the conditions are not met, the farmer must think of some other ideas. Also, the graph generated from Adafruit IO interface can help a farmer predict what may happen in the upcoming days.

HCSR04 SONAR/Ultrasonic Sensor: HCSR04 SONAR or Ultrasonic Sensor is a widely used

ultrasonic sensor, which is mainly used to determine the distance of a certain object or place with respect to the sensor itself. To run this sensor, an Arduino code is needed. Necessary formulae are given in the code to determine the distance from the readings obtained from the sensor itself.

Figure 6: SONAR (HCSR04) Sensor

In this project, such a sensor has been used to determine the water level of a container. The increase and decrease of water level can be seen on the interface made in Blynk. A preset water level is used in our project; the sensor measures the distance of the current water level from itself, then subtracts the obtained distance from the total distance, then the difference is divided by the total distance to give an idea of the water level of the container. Based on the information obtained about water level, the farmer can refill the container if the percentage is too low. Also, this work will be of great benefit for farmers who cannot use shallow pumps.

BMP180 Barometric Pressure/Temperature/Altitude Sensor: The BMP180 Sensor is mainly used for detecting the pressures and altitudes of a particular area. The pressures may be either relative or absolute, and the altitude may be either from sea level or other references.

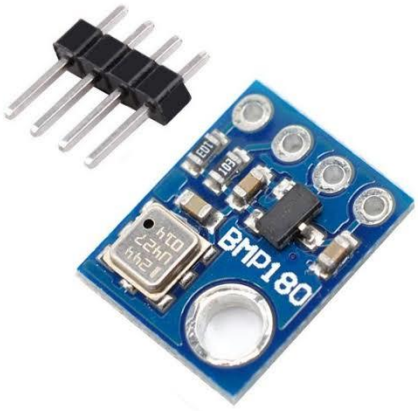

Figure 7: BMP180 Sensor

In this project, a BMP180 sensor has been used to detect such pressures and altitudes. In case of low pressures, the pressure values will obviously change, so the farmer can take necessary steps just in time to save his crops.

DS18B20 Temperature Sensor: The DS18B20 Temperature Sensor is a sensor which is useful for determining soil temperature. There are two types of such sensors, the elongated one is more useful to get soil temperature.

Figure 8: DS18B20 Sensor

One such sensor was used in our project. The purpose of this sensor was to determine the soil temperature and upload it to the Adafruit IO interface. The data obtained from the sensor is read in temperature units. So, if the farmer can understand that the soil temperature is bad for his crops, then he can take necessary steps. In this case, it is to be noted that a preset temperature was not set, because the same field can be used for harvesting different corps at different times of a year and depending on crops, critical temperatures will be different.

HL69/YL69 Soil Moisture Sensor: The HL69/YL69 Soil Moisture Sensor is a sensor that detects the moisture level of the soil of a particular area. The soil moisture level data obtained from the sensor can be read as a percentage value on data monitoring interfaces.

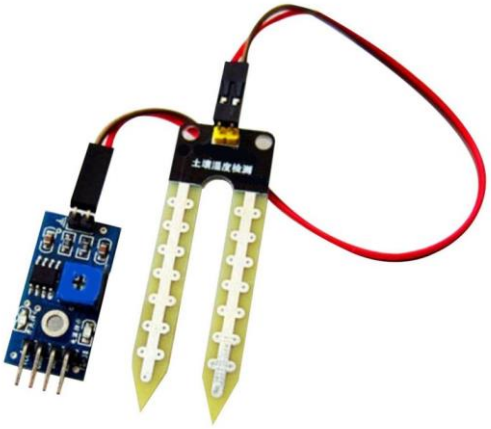

Figure 9: Soil Moisture Sensor

In this project, one such sensor has been used to determine the soil moisture. The data obtained is sent to ESP8266 NodeMCU, whence a signal is given to turn the water pump motor on or off using an Arduino code. If the soil moisture is below 40%, the motor will be turned on, and if the moisture is above 50%, the motor will turn off.

MQ138 Gas Sensor: MQ138 is a gas sensor use to determine gases which may be harmful for a system. If the sensor detects certain types of gases, it sends signals to the microcontroller, from which codes can be executed or other necessary steps can be taken.

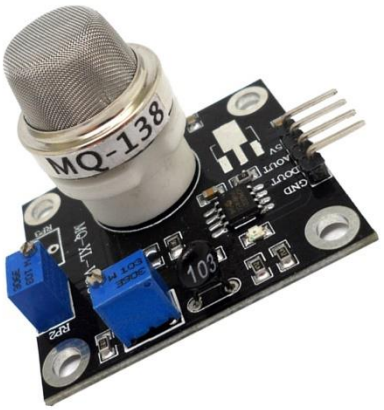

Figure 10: Gas Sensor

One such gas sensor has been used in this project. The purpose of this gas sensor in this experiment is to detect smoke, CO_2 and gas particles which may harm the crops. The data obtained is uploaded in the Blynk interface as a string indicating whether any gas has been detected or not. If such gases are found by the sensor, a siren is also activated. This sensor is also useful in cases where there is fire nearby, and by taking quick steps after being alerted by the sensor, crops can be saved.

Rain Sensor: Rain Sensor is one of the most popular rain-detecting sensors. The popularity is high because it can effortlessly detect whether it is raining or not. If rainwater falls on the sensor surface, a signal is sent to the microcontroller indicating that there is at least some rainfall. From the signals received, necessary steps can be taken.

Figure 11: Rain Sensor

In our project, a rain sensor was used. If rainwater falls on the sensor surface, a siren is activated, and a message is uploaded on Blynk interface indicating that there is rainfall.

If both the rain sensor and the gas sensor give outputs of no rain and no gas respectively, then a farmer can be free of thought and can concentrate in other activities of his daily life.

4.2 Results

Random forest classifier and Xgboost classifier was used to train on features received from VGG16 model. Their classification report is given below:

Random Forest Report :

	precision	recall	f1-score	support
Tomato___Early_blight	0.47	0.39	0.43	200
Tomato___Late_blight	0.73	0.76	0.74	382
Tomato___Septoria_leaf_spot	0.72	0.74	0.73	354
Tomato___Spider_mites Two-spotted_spider_mite	0.87	0.86	0.86	335
Tomato___healthy	0.88	0.92	0.90	318
accuracy			0.76	1589
macro avg	0.73	0.73	0.73	1589
weighted avg	0.76	0.76	0.76	1589

Figure 12: Classification Report of Random Forest

XGB Report :

	precision	recall	f1-score	support
Tomato___Early_blight	0.85	0.75	0.80	200
Tomato___Late_blight	0.92	0.91	0.92	382
Tomato___Septoria_leaf_spot	0.92	0.98	0.95	354
Tomato___Spider_mites Two-spotted_spider_mite	0.96	0.98	0.97	335
Tomato___healthy	0.98	0.98	0.98	318
accuracy			0.93	1589
macro avg	0.93	0.92	0.92	1589
weighted avg	0.93	0.93	0.93	1589

Figure 13: Classification Report of Xgboost

Based on overall accuracy and other performance, xgboost classifier appears as the clear choice between the two.

Prediction on Image

Primarily the model was trained on single leaf image on each class. After completion of train set, the model was tested on single leaf image and multiple leaves image. Though the model has performed satisfactory in single leaf image, it fails to predict in later cases. Some sample prediction output on input images are given below:

Single Leaf Image

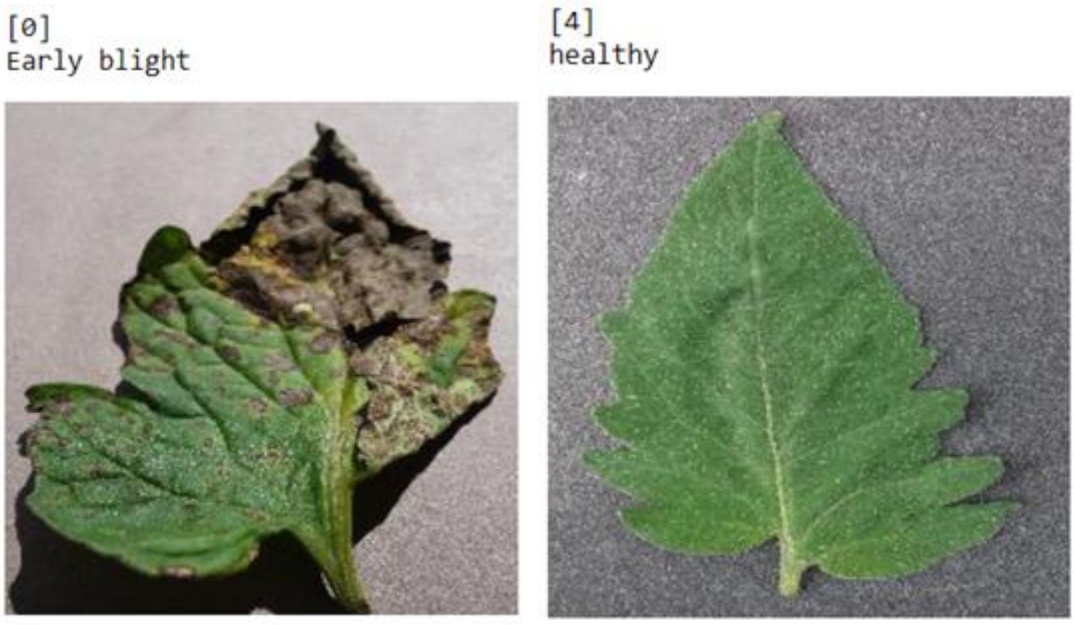

Figure 14: Correct Prediction on Single Leaf Image

As we can see from above examples model successfully detects the condition from single leaf image.

Multiple Leaves Image

Figure 15: Wrong Prediction on Healthy Leaf Image

Figure 16: Wrong Prediction on Early Blight Leaf Image

As ESP32 cam will stream multiple leaves image, this issue needs to be addressed so that model can detect condition from the captured frame of ESP32 cam.
To improve model behaviour, some little tweaks were made on the dataset. As healthy leaf

image of most of the plants show similar traits, multiple healthy leaves images for Cassava Dataset was used. Some sample predictions images on improved model is shown below:

Figure 17: Correct Prediction on healthy Leaf Image

Figure 18: Wrong Prediction for Early Blight

From above it's obvious due to scarcity of dataset that contains labelled multiple leaves image with disease, model couldn't be improved in this side. A further collection of images like that can overcome this problem in future.

Prediction on Captured Frame of ESP32 cam

As mentioned above, model can predict healthy leaf from single leaf and multiple leaves image while disease can only be predicted for image with single leaf. Outcome from ESP32 cam input was more or less same though in some cases medium quality images and lack of light may lead to wrong prediction. Some sample predictions are given below:

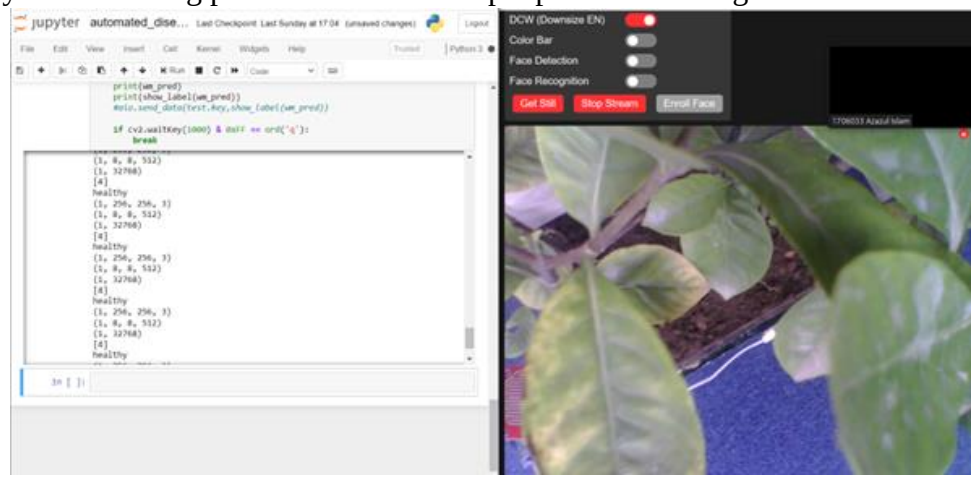

Figure 19: Healthy Leaf Image Detection

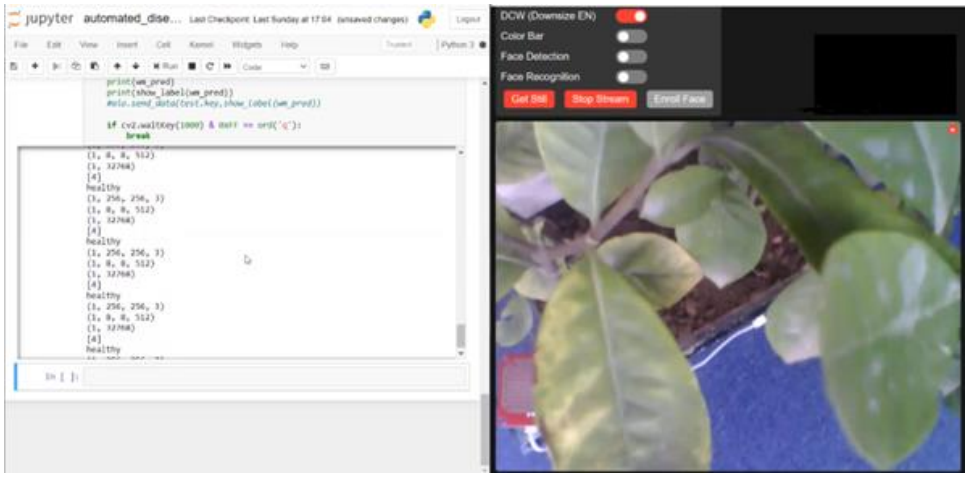

Figure 20: Failed to Detect the Leaf with Yellow Spot

Later additional code was attached to update the condition in Adafruit IO server.

4.3 GitHub Link

https://github.com/munim-sah75/IoT-Based-Smart-Agriculture-System.git

4.4 YouTube Link

https://youtu.be/v6FSAP1Xo8k

5 Design Analysis and Evaluation

5.1 Novelty

In this project, different IoT sensors have been integrated together i.e BMP180 as pressure sensor, HCSR04 as sonar sensor, DHT11 as temperature and humidity sensor, DS18B20 as temperature sensor etc. All the sensors' outputs have been sent to the cloud server using IoT. A pump has been used to water the plants in suitable situations. Also, a surveillance camera has been implemented to observe the plants. A machine learning model using OpenCV, and Python has been implemented to detect healthy and rotten leaves from the surveillance camera successfully. The surveillance camera captures and sends images to the server, whence the machine learning model detects whether the plant is healthy or not. Analysis, execution and detection have all been done in this project simultaneously.

5.2 Project Management and Cost Analysis

5.2.1 Bill of Materials

Name of equipment	No. of units bought	Unit Price (In BDT)	Total Price (In BDT)
DHT11 Sensor	1	180	180
BMP180 Sensor	1	170	170

HCSR04 Sensor	1	96	96
MQ-138 Sensor	1	190	190
Soil Moisture Sensor	1	165	165
DS18B20 Sensor	1	130	130
Rain Sensor	1	130	130
Esp8266 board	2	470	940
6V DC Water Pump	1	196	196
Tube for DC Pump	1	30	30
TIP122 NPN Transistor	2	16.59	33
Esp32 camera module	1	890	890
PCB	2	354.5	709
Plants	2	85	170
Aquarium + Whole Setup	1	1200	1200
4.7kΩ Resistor	1 Pack(20)	13	13
0.1uF Capacitor (275V)	1	8.70	8.70
0.1uF Capacitor (400V)	1	1.5	1.5
1kΩ Resistor	1	2.4	2.4
Buzzer	1	18.17	18.17
9V Battery with Connector	1 Each	66	66
Jumper Wire (Male to Male)	1 Bundle	95	95
Female to Female Wire	10	3	30
Female to Male Wire	10	3	30
Male to Male Wire	10	3	30
FTDI USB 3V3 to 5V	1	545	545
Large Breadboard	1	155	155
7805 Voltage Regulator	2	15	30

Table: Bill of Materials Used

5.2.2 Calculation of Per Unit Cost of Prototype

Per unit cost of the prototype is BDT 6254 calculated from the upper table.

5.2.3 Calculation of Per Unit Cost of Mass-Produced Unit

For mass production, we don't need any aquarium to implement. It will be implemented on farm/physical land. We also don't need any pump as we only design the control elements. We planted tree here for demonstration. However, in real world field farmer will plant their required seeds. We won't need the plant cost here. So, without that cost, we need BDT 4700 to implement it for mass-produced unit.

However, if the farmer needs to save his data and get more features through online, he may buy his own domain and the premium services of online platforms. Its expenditure depends on the type of security and services he wants. So, this type of cost is not fixed rather, it depends on the farmer himself.

5.2.4 Timeline of Project Implementation

We started doing the work of the project from 1 July 2022. All the works completed between 30 August 2022. So, in total 2 months are needed to implement the project.

5.3 Practical Considerations of the Design to Address Public Health and Safety, Environment, Cultural, and Societal Needs

5.3.1 Considerations to public health and safety

No equipment has been used in this project which can harm the safety and health of mass people. Rather, the outcome of this project will ensure better harvest from lands. Here, the power source was to power up the circuits for just some Volts which is in no way harmful. So, it is sufficient to say that

public safety and health are not harmed by this project.

5.3.2 Considerations to environment

The main purpose of the project was to monitor the environmental factors such as temperature, moisture, humidity etc. of plants and lands. There is no way that this project will have a detrimental effect on the environment. Environment safety has been ensured while doing this project.

5.3.3 Considerations to cultural and societal needs

Although farmers of Bangladesh are not yet smart enough with digital systems, this project has the potential to create milestones in the harvesting system of the country. As Bangladesh is mostly dependent on its agricultural system, digitization in this field will create more positive impacts on the lives of farmers. This project will also ensure that there are no rotten crops in the field which will make sure healthy crops are being harvested. So, it can be said without doubt that our society will see a cultural shift by implementing this project on a large scale.

5.4 Assessment of the Impact of the Project on Societal, Health, Safety, Legal and Cultural Issues

5.4.1 Assessment of Societal Issues

Societal issues mainly indicate the social acceptance of a modern technology-based project like ours. If farmers are interested in advanced technology related to farming, it will lead to improved and easier farming systems. Such changes can increase the harvest amount pr annum, which can be a big boon for the agriculture-dependent population of ours. The farming societies will also get improved over time.

5.4.2 Assessment of Health and Safety Issues

Health and safety issues are important when it comes to implementing any project. In this project of ours, the main safety issues may come from not taking enough safety measures. An unsafe setup will obviously lead to accidents, so care has been taken while implementing the project so that the system maintains utmost safety.

5.4.3 Assessment of Legal Issues

In this project, nothing was done which would directly go against the laws of the country. So, it can be safely assumed that there would be no legal issues.

5.4.4 Assessment of Cultural Issues

Our farming culture is traditionally based on manual labor. This project of ours is more based on technology and less based on traditional methods. This may not suit well for certain farming societies who are more inclined towards traditions. But if long-term use is considered, a technological approach will certainly bring better outputs and will help farmers a lot in the long run.

5.5 Evaluation of the Sustainability and Impact of the Designed Solution in the Societal and Environmental Contexts

5.5.1 Evaluation of Sustainability

If sustainability is considered, this project is quite sustainable on its own. The only caveat is the recharging of the power source after a certain time. That means, a rechargeable power source should be charged at regular intervals. As an alternative, solar cells can be used to keep the batteries charged during the entire course of the day.

5.5.2 Evaluation of Impact of Design in Societal Context

Although many farmers are still not familiar with modern technology, this project is made to be quite user-friendly to the farmers. Such a user-friendly project can also make people in general more interested in technology.

5.5.3 Evaluation of Impact of Design in Environmental Context

Lithium-ion batteries, if not disposed of properly, will harm the environment in more ways than one. Also, lithium mining is nowadays debatable due to issues of environmental safety and child labor. From this perspective, renewable energies like solar energy and hydro-energy can be good alternatives. Solar panels can be used to harness solar energy.

6 Reflection on Individual and Teamwork

6.1 Individual Contribution of Each Member

All four members of the project group have worked heart and soul for the implementation of this project. Although the project was a team effort, individual efforts on certain works have made the combined efforts easier. Here is a detailed description of the individual contributions of each member of this project group:

- Sheikh Munim Hussain: Gas Sensor, Rain Sensor, PCB.
- Md. Jahidul Hoq Emon: Surveillance System, Adafruit IO
- Shafin Shadman Ahmed: Irrigation, SONAR Sensor
- Azazul Islam: BMP180 Sensor, Machine Learning Model Training.

With all these people's individual and combined efforts, a successful project demonstration has been made possible.

6.2 Mode of Teamwork

For a successful project implementation, giving enough time is a crucial factor. It has been ensured by all the members of the team that ample time is given to the project. After class time, the team worked together to implement this project. Hall room and central library have been used for this purpose. Buying different equipment, ordering them etc. have been done together. Also, when any member got stuck at a certain point, the whole team sat together and gave their own thoughts about solving the problem. Thus, the whole team's participation has been ensured.

6.3 Diversity Statement of Team

Our team consists of people from different districts of the country. Munim is from Satkhira, Emon is from Brahmanbaria, Shafin is from Kushtia and Azazul is from Dhaka. Also, two of our members are hall residents (Munim and Emon), whereas the other two are attached. Munim is gravitated towards cricket and new smartphones, Emon is more inclined to video games, Shafin has interests in vehicles and Razon is an international affairs enthusiast.
Also, when the team got stuck at a problem, different ideas came from everyone which helped to reach to the solution more efficiently and compare their impacts.

6.4 Logbook of Project Implementation

Date	Milestone Achieved	Individual Role	Team Role	Comments
09-07-2022	SONAR sensor worked	Emon & Munim coded the sonar code and implemented the logic.	Shafin & Azazul bought the sensor and generated the logic behind the codes.	Our sensor could work successfully to measure a certain distance.
13-07-2022	DHT11 Sensor worked	Munim coded the DHT11 sensor to calculate humidity and temperature.	Shafin, Emon & Azazul collected muds to use in the process.	Accurate calculation of temperature and humidity has been done.
20-07-2022	BMP180 Sensor Worked	Azazul coded the BMP180 and calculated pressure.	Real time data was monitored to match with calculation by Emon & Munim.	Practical data matched with experimental data.
27-07-2022	Adafruit IO setup.	Emon & Shafin created a dashboard using Adafruit IO.	Proper code was done to send data to server using Adafruit by the team.	Real time data monitoring was achieved.
03-08-2022	Rain Sensor worked	Munim set up the rain sensor and it pressed buzzer if it could detect rain.	Rest of the team members worked to get data to the server.	Real time monitoring of weather reached another milestone.
08-08-2022	Esp32 camera module	Emon & Azazul worked to setup esp32 camera.	Other members worked to get the data to server.	Live streaming made possible to a local host using the camera module.
15-08-2022	Machine Learning model training and detection.	Azazul trained an ML model and predicted the leaves' condition.	Other members collected photos for the database and did preprocessing.	Model successfully trained with very nice accuracy. Detection was done using esp32 camera photos.
22-08-2022	Water Pump for irrigation.	Shafin & Munim setup the water pump for irrigation purpose.	Value of threshold moisture and other criteria were calculated by Azazul & Emon.	It pumped water according to moisture of the soil. Also, the pump could be manually controlled by Adafruit.
28-08-2022	Plants bought and aquarium setup.	Emon & Shafin collected plants from nursery for demonstration purpose	Munim & Azazul setup the plants in the aquarium and integrated the sensors inside.	An overall setup of the project has been done.
30-08-2022	Project video, presentation and	All worked equally here.	The whole team worked together	Everything merged and it

	demonstration.		to merge all the efforts and make the project come to life.	worked!

Table: Logbook of the Project

7 References

- Plantvillage Dataset: https://www.kaggle.com/datasets/emmarex/plantdisease
- Cassava Leaf Image Dataset: https://www.kaggle.com/competitions/cassava-leaf-disease-classification/data
- VGG16 Architecture Prediction: https://www.kaggle.com/code/meetnagadia/tomato-leaf-disease-prediction-using-vgg16-and-cnn
- Esp32 camera: ESP32-CAM, Camera Module Based on ESP32, OV2640 Camera Included (waveshare.com)