SMART IRRIGATION SYSTEM

PROJECT REPORT

Submitted for CAL in B.Tech Internet of Things (CSE3009)

By

1.PIYUSH BAMEL	**16BCE1221**
2.KAUSTUBH NAGAR	**16BCE1338**
3.KUNAL DILIP CHANDIRAMANI	**16BCE1396**

Slot: F2+TF2

Name of faculty: Prof. Rekha D

(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING)

CERTIFICATE

This is to certify that the Project work entitled "***Smart Irrigation System***" that is being submitted by **Piyush Bamel, Kaustubh Nagar and Kunal Dilip Chandiramani** for CAL in B.Tech. **Internet of Things (CSE3009)** is a record of bonafide work done under my supervision. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Place : VIT University, Chennai Campus

Signature of Students:

PIYUSH BAMEL	**16BCE1221**
KAUSTUBH NAGAR -	**16BCE1338**
KUNAL CHANDIRAMANI	**16BCE1396**

Signature of Faculty:

PROF. REKHA D

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, Prof. Rekha D, School of Computer Science and Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We express our thanks to our Programme Chair Dr. Rajesh Kanna B (for B.Tech CSE) for his/her support throughout the course of this project. We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course. We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

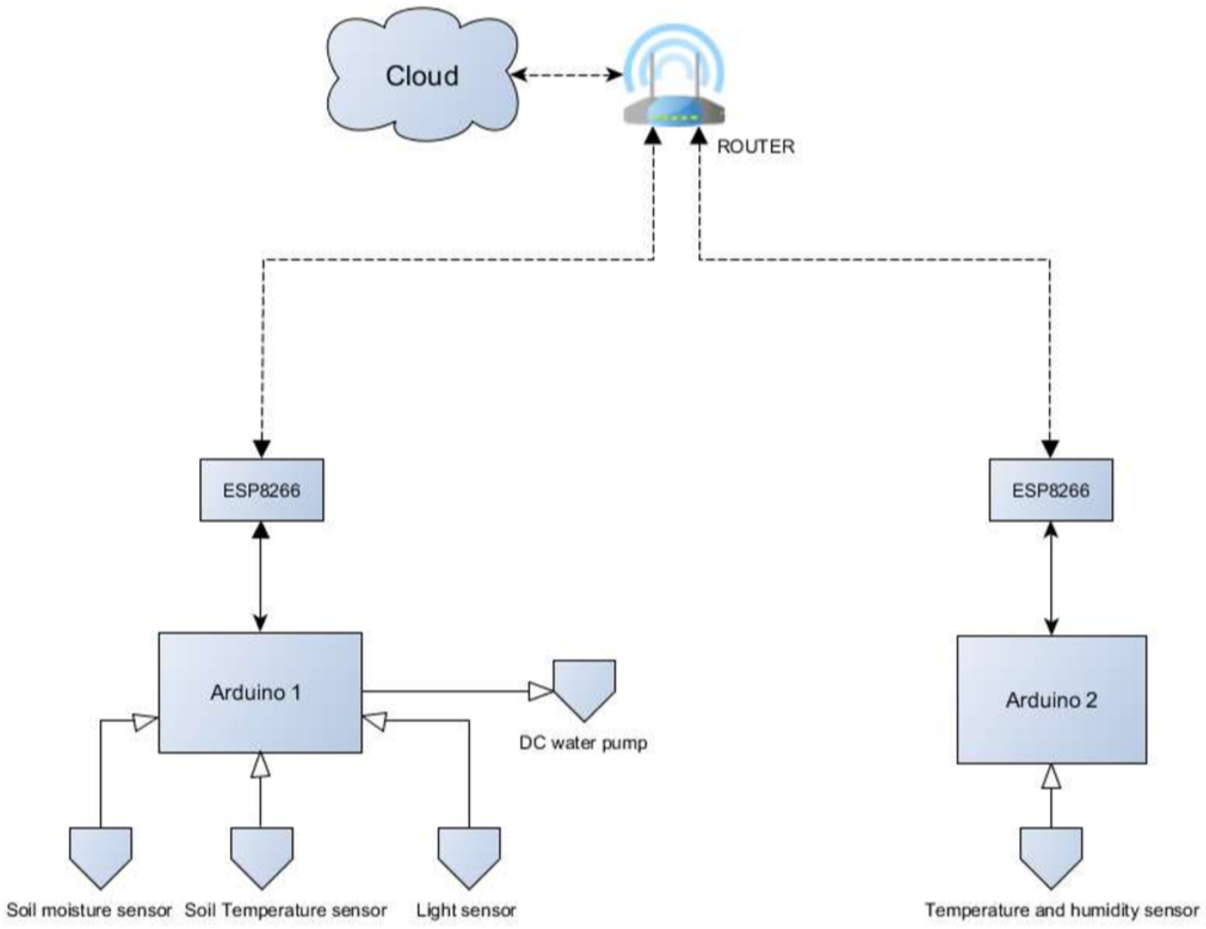
INTRODUCTION

India is the country of village and agriculture plays an important role for development of country. In our country, agriculture depends on the monsoons which has insufficient source of water. So the irrigation is used in agriculture field. In Irrigation system, depending upon the soil type, water is provided to plant. In agriculture, two things are very important, first to get information of about the fertility of soil and second to measure moisture content in soil. Nowadays, for irrigation, different techniques are available which are used to reduce the dependency of rain. And mostly this technique is driven by electrical power and on/off scheduling. In this technique, water level indicator placed in water reservoir and soil moisture sensors are placed root zone of plant and near the module and gateway unit handles the sensor information and transmit data to the controller which in turns the control the flow of water through the valves.

Smart Irrigation System can be used for controlling the watering or irrigation of plants. It controls the irrigation of plants where the need of human intervention can be reduced. This mainly focused on wastage of water, which is a major concern of modern era. It also aids time saving, cost effectiveness, environmental protection, low maintenance and operating cost and efficient irrigation service. Arduino is used in the design of the prototype model in making the system compact and sustainable. The system has sensor which measures the moisture of the soil and switches on watering pump according to the requirement.

LEVEL OF DEPLOYMENT

Level 4 : Data storage and Analysis on cloud with multiple nodes.
A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud-based. Smart Irrigation System consists of multiple node placed in different locations for monitoring weather and plant in an area. Node 1 consists of temperature, humidity and dew point sensor which constantly monitors surrounding and sent data to cloud for further processing. Processing done over it in cloud is prediction of rainfall using minimum,maximum and average of temperature, humidity and dew point to predict category of rainfall.
Node 2 consists of plant monitoring system with records and updates soil moisture as well status of motor. If a certain condition is met, it actuates the motor to water the plants.



IMPLEMENTATION AND WORKING

The system is realized using multiple components that work in synchronization. It consists of 2 Arduino Uno, one for calculating rain prediction and other for measuring the soil moisture in the soil and accordingly supplying water to the soil using water pump connected to it. To store the data and make rainfall prediction using machine learning algorithm the system uses Thingspeak cloud which is connected to the arduino using ESP8266 module. A dashboard was also developed for monitoring and controlling the system.

REQUIRED HARDWARE COMPONENTS:

- *FOR SOIL MONITORING*

1. Evana soil moisture/ humidity sensor
2. Arduino uno
3. REES52 serial wireless transceiver module for IOT ESP8266
4. Electomanai Micro DC 3-6V Micro submersible mini water pump
5. 9V Battery
6. Jumper wires

ESTIMATED COST : **Rs 1577**

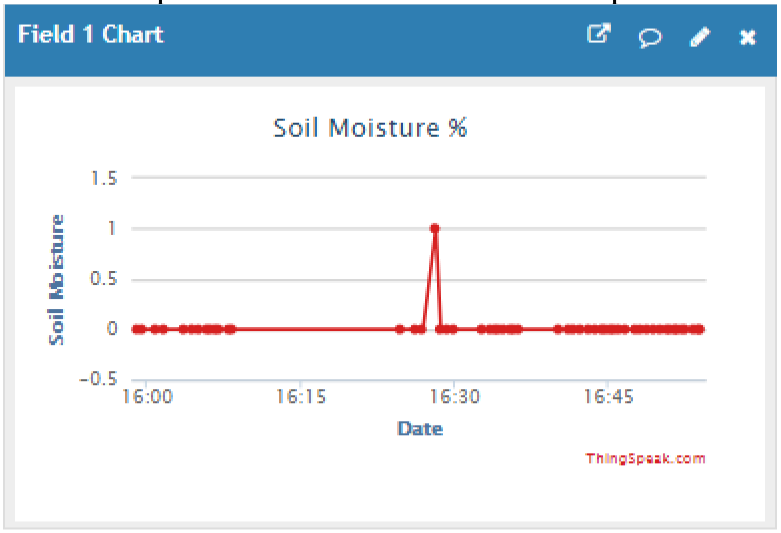
- *FOR WEATHER STATION*

1. Robodo electronics DHT11 temperature and humidity sensor
2. Arduino uno
3. REES52 serial wireless transceiver module for IOT ESP8266
4. 9v battery
5. Jumper wires

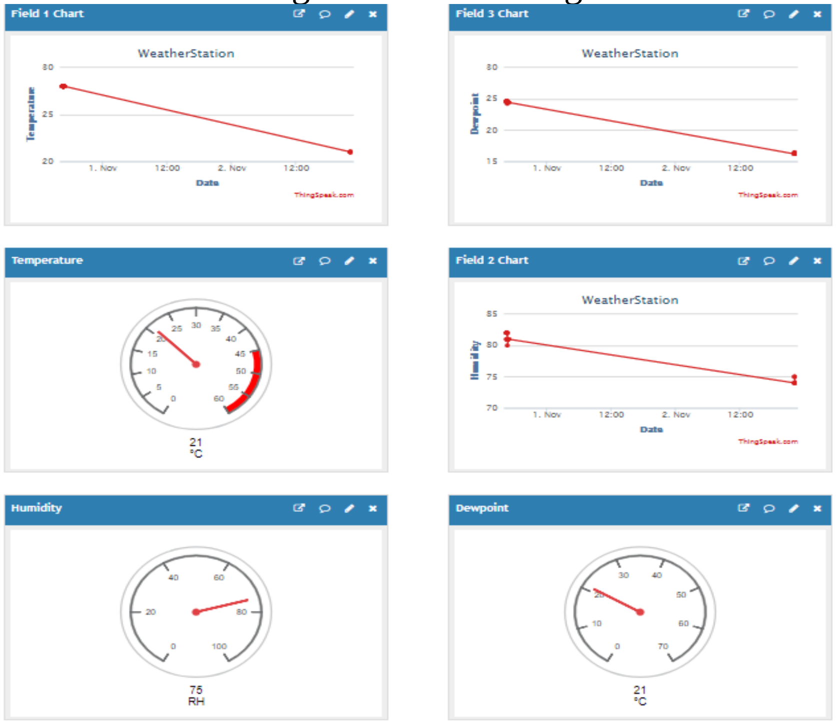
ESTIMATED COST : **Rs 996**

The working and steps involved in smart irrigation system are as follows:

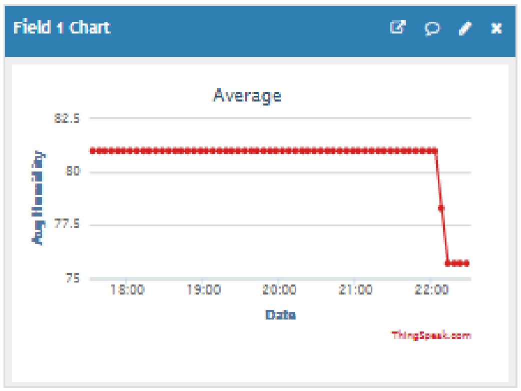
- Initially node 1 (Soil monitoring node) checks for soil moisture using enava soil moisture and humidity sensor connected to it. If the soil moisture reading is less than 20 percent than it checks for rain prediction category.

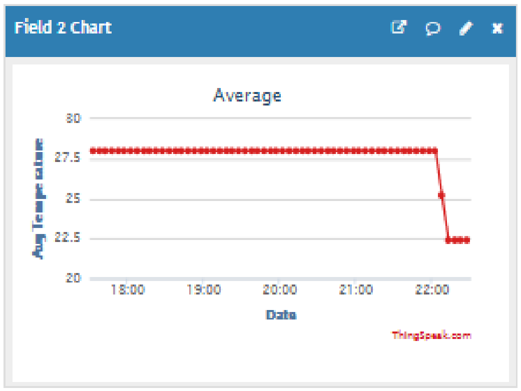


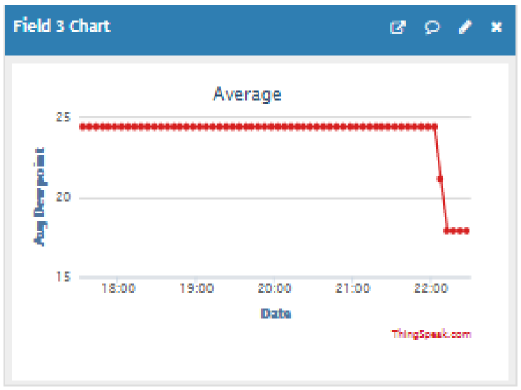
- Parallely node 2 (weather station node) takes the humidity and temperature reading from the environment and stores the data into the thingspeak cloud using ESP8266 module.Once humidity and temperature readings are uploaded on the cloud using these two reading we calculate the dew point.



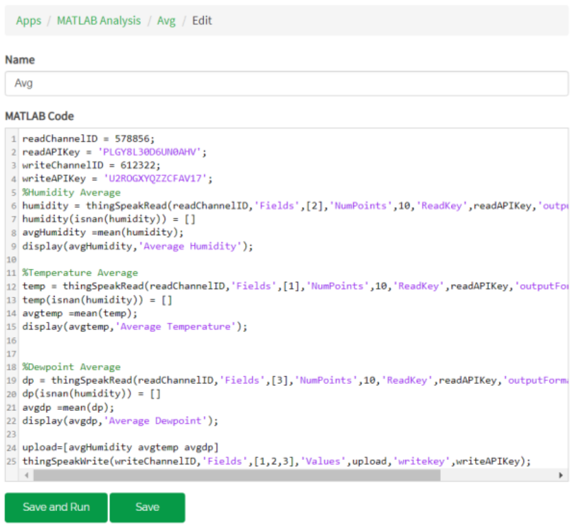
- In the cloud using the temperature, humidity and dew point readings, we also calculate the average temperature, average humidity and average dew point taking the last 10 readings. Which will be used as a feature for rainfall prediction.These are calculated with help of MATLAB Analysis with TimeControl (features provided by Thingspeak)







MATLAB ANALYSIS



Name

Avg

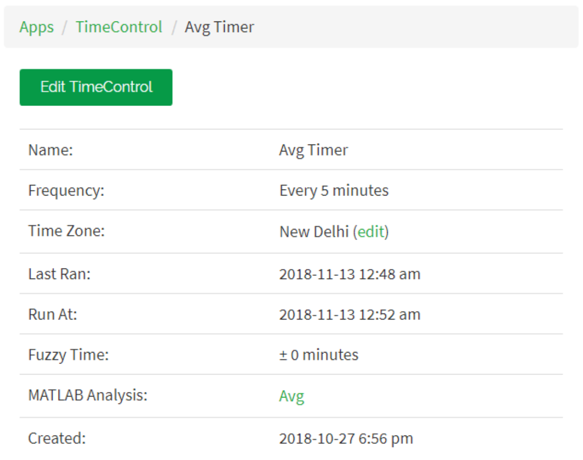
MATLAB Code

```
readChannelID = 578856;
readAPIKey = 'PLGY8L30D6UN0AHV';
writeChannelID = 612322;
writeAPIKey = 'U2ROGXYQZZCFAV17';
%Humidity Average
humidity = thingSpeakRead(readChannelID,'Fields',[2],'NumPoints',10,'ReadKey',readAPIKey,'outpu
humidity(isnan(humidity)) = []
avgHumidity =mean(humidity);
display(avgHumidity,'Average Humidity');

%Temperature Average
temp = thingSpeakRead(readChannelID,'Fields',[1],'NumPoints',10,'ReadKey',readAPIKey,'outputFo
temp(isnan(humidity)) = []
avgtemp =mean(temp);
display(avgtemp,'Average Temperature');


%Dewpoint Average
dp = thingSpeakRead(readChannelID,'Fields',[3],'NumPoints',10,'ReadKey',readAPIKey,'outputForm
dp(isnan(humidity)) = []
avgdp =mean(dp);
display(avgdp,'Average Dewpoint');

upload=[avgHumidity avgtemp avgdp]
thingSpeakWrite(writeChannelID,'Fields',[1,2,3],'Values',upload,'writekey',writeAPIKey);
```

Code

```
        readChannelID =;
        readAPIKey = ;
        writeChannelID =;
        writeAPIKey =;
        %Humidity Average
        humidity =
thingSpeakRead(readChannelID,'Fields',[2],'NumPoints',10,'ReadKey',readAPIKey
,'outputFormat','matrix');
        humidity(isnan(humidity)) = []
        avgHumidity =mean(humidity);
        display(avgHumidity,'Average Humidity');
        %Temperature Average
        temp =
thingSpeakRead(readChannelID,'Fields',[1],'NumPoints',10,'ReadKey',readAPIKey
,'outputFormat','matrix');
        temp(isnan(humidity)) = []
        avgtemp =mean(temp);
        display(avgtemp,'Average Temperature');
        %Dewpoint Average
        dp =
thingSpeakRead(readChannelID,'Fields',[3],'NumPoints',10,'ReadKey',readAPIKey
,'outputFormat','matrix');
        dp(isnan(humidity)) = []
        avgdp =mean(dp);
        display(avgdp,'Average Dewpoint');
        upload=[avgHumidity avgtemp avgdp]
        thingSpeakWrite(writeChannelID,'Fields',[1,2,3],'Values',upload,'writekey',
writeAPIKey);
```

TimeControl

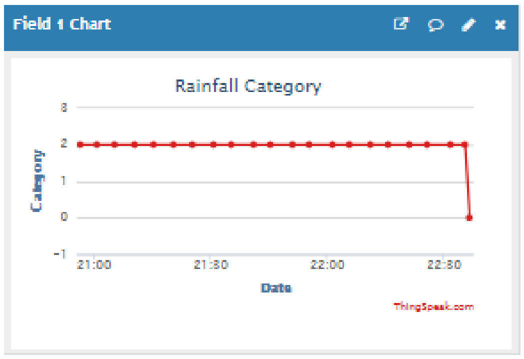


Apps / TimeControl / Avg Timer

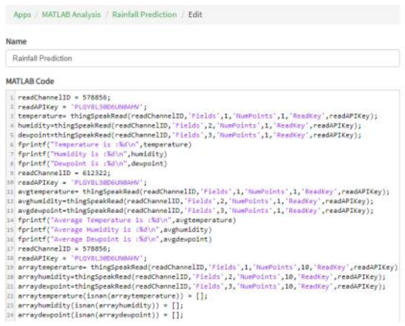
Edit TimeControl

Name:	Avg Timer
Frequency:	Every 5 minutes
Time Zone:	New Delhi (edit)
Last Ran:	2018-11-13 12:48 am
Run At:	2018-11-13 12:52 am
Fuzzy Time:	± 0 minutes
MATLAB Analysis:	Avg
Created:	2018-10-27 6:56 pm

- Using year, month, day, max humidity , minimum humidity, average humidity, max temperature, minimum temperature, average temperature, max dew point , minimum dew point, average dew point as features we apply linear regression with gradient descent to predict the rainfall category. We have divided rainfall into four categories, the representation of these categories is 0 - no rainfall, 1 - light rainfall, 2 - medium rainfall, 3 - heavy rainfall. These are calculated with help of MATLAB Analysis with TimeControl (features provided by Thingspeak)



MATLAB ANALYSIS



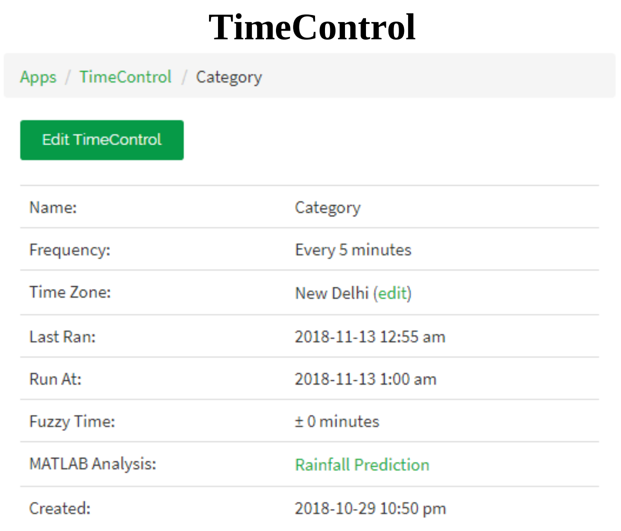
Rainfall Prediction

```
readChannelID = 578856;
readAPIKey = 'PLGY8L30D6UN0AHV';
temperature= thingSpeakRead(readChannelID,'Fields',1,'NumPoints',1,'ReadKey',readAPIKey);
humidity=thingSpeakRead(readChannelID,'Fields',2,'NumPoints',1,'ReadKey',readAPIKey);
dewpoint=thingSpeakRead(readChannelID,'Fields',3,'NumPoints',1,'ReadKey',readAPIKey);
fprintf("Temperature is :%d\n",temperature)
fprintf("Humidity is :%d\n",humidity)
fprintf("Dewpoint is :%d\n",dewpoint)
readChannelID = 612322;
readAPIKey = 'PLGY8L30D6UN0AHV';
avgtemperature= thingSpeakRead(readChannelID,'Fields',1,'NumPoints',1,'ReadKey',readAPIKey);
avghumidity=thingSpeakRead(readChannelID,'Fields',2,'NumPoints',1,'ReadKey',readAPIKey);
avgdewpoint=thingSpeakRead(readChannelID,'Fields',3,'NumPoints',1,'ReadKey',readAPIKey);
fprintf("Average Temperature is :%d\n",avgtemperature)
fprintf("Average Humidity is :%d\n",avghumidity)
fprintf("Average Dewpoint is :%d\n",avgdewpoint)
readChannelID = 578856;
readAPIKey = 'PLGY8L30D6UN0AHV';
arraytemperature= thingSpeakRead(readChannelID,'Fields',1,'NumPoints',10,'ReadKey',readAPIKey)
arrayhumidity=thingSpeakRead(readChannelID,'Fields',2,'NumPoints',10,'ReadKey',readAPIKey);
arraydewpoint=thingSpeakRead(readChannelID,'Fields',3,'NumPoints',10,'ReadKey',readAPIKey);
arraytemperature(isnan(arraytemperature)) = [];
arrayhumidity(isnan(arrayhumidity)) = [];
arraydewpoint(isnan(arraydewpoint)) = [];
```

Code

```
readChannelID =;
readAPIKey =;
temperature= thingSpeakRead(readChannelID,'Fields',1,'NumPoints',1,'ReadKey',readAPIKey);
humidity=thingSpeakRead(readChannelID,'Fields',2,'NumPoints',1,'ReadKey',readAPIKey);
dewpoint=thingSpeakRead(readChannelID,'Fields',3,'NumPoints',1,'ReadKey',readAPIKey);
fprintf("Temperature is :%d\n",temperature)
fprintf("Humidity is :%d\n",humidity)
fprintf("Dewpoint is :%d\n",dewpoint)
readChannelID =;
readAPIKey =;
avgtemperature= thingSpeakRead(readChannelID,'Fields',1,'NumPoints',1,'ReadKey',readAPIKey);
avghumidity=thingSpeakRead(readChannelID,'Fields',2,'NumPoints',1,'ReadKey',readAPIKey);
avgdewpoint=thingSpeakRead(readChannelID,'Fields',3,'NumPoints',1,'ReadKey',readAPIKey);
fprintf("Average Temperature is :%d\n",avgtemperature)
fprintf("Average Humidity is :%d\n",avghumidity)
fprintf("Average Dewpoint is :%d\n",avgdewpoint)
readChannelID =;
readAPIKey =;
arraytemperature= thingSpeakRead(readChannelID,'Fields',1,'NumPoints',10,'ReadKey',readAPIKey);
arrayhumidity=thingSpeakRead(readChannelID,'Fields',2,'NumPoints',10,'ReadKey',readAPIKey);
arraydewpoint=thingSpeakRead(readChannelID,'Fields',3,'NumPoints',10,'ReadKey',readAPIKey);
arraytemperature(isnan(arraytemperature)) = [];
```

```
arrayhumidity(isnan(arrayhumidity)) = [];
arraydewpoint(isnan(arraydewpoint)) = [];
maxtemperature=max(arraytemperature);
maxhumidity=max(arrayhumidity);
maxdewpoint=max(arraydewpoint);
mintemperature=min(arraytemperature);
minhumidity=min(arrayhumidity);
mindewpoint=min(arraydewpoint);
d = datetime('today');
[year,month,date] = ymd(d);
theta = [ 0.122021 0.042260 -0.013015 -0.041535 0.146186 -0.071218 -0.052697 -
0.034748 0.074159 0.128699]
X=[1 maxtemperature avgtemperature mintemperature maxdewpoint avgdewpoint
mindewpoint maxhumidity avghumidity minhumidity]
%use test set for prediction - using mean & stdev of entire set
y_hat = X*theta';
fprintf("y_hat is %d\n",y_hat);
if y_hat<=1
y_hat = 0;
elseif y_hat>1 && y_hat<=5
y_hat = 1;
elseif y_hat>5 && y_hat<10
y_hat = 2;
else
y_hat = 3;
end
fprintf(' y = %.3f \n',y_hat);
writeChannelID =;
writeAPIKey =;
thingSpeakWrite(writeChannelID,'Fields',1,'Values',y_hat,'writekey',writeAPIKey)
;
```



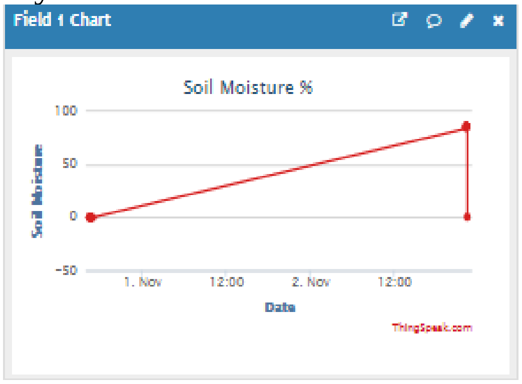
TimeControl

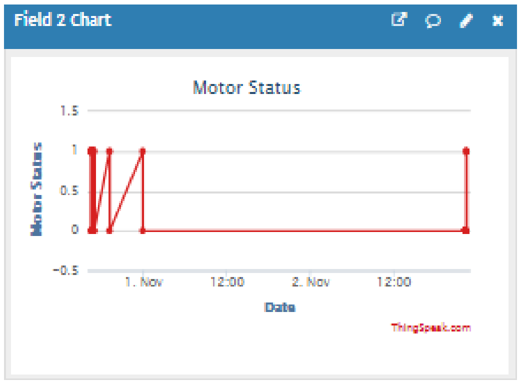
Apps / TimeControl / Category

Edit TimeControl

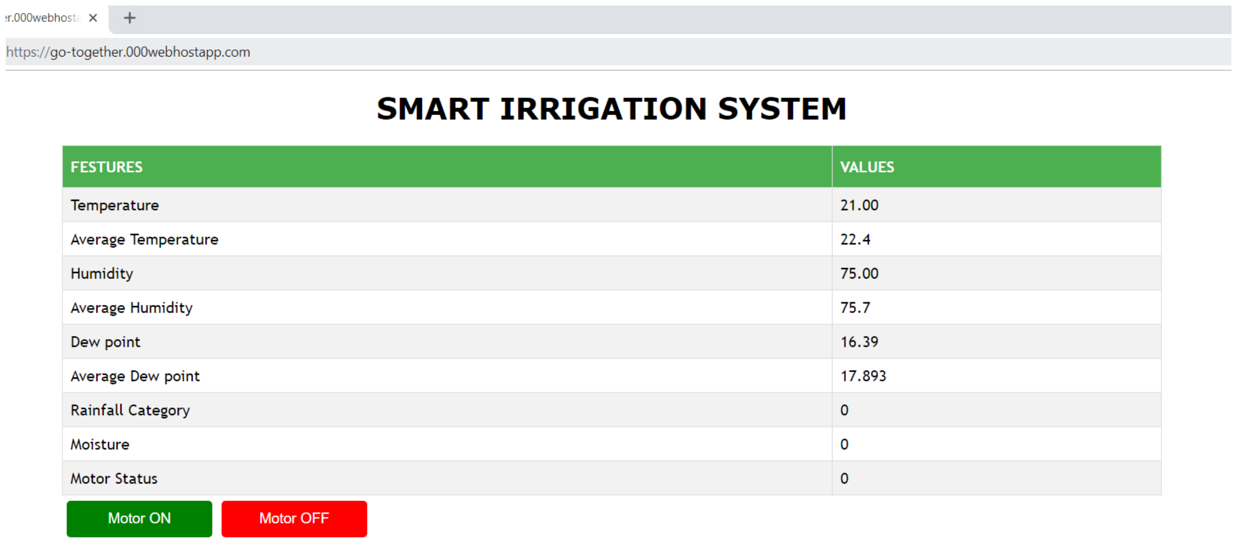
Name:	Category
Frequency:	Every 5 minutes
Time Zone:	New Delhi (edit)
Last Ran:	2018-11-13 12:55 am
Run At:	2018-11-13 1:00 am
Fuzzy Time:	± 0 minutes
MATLAB Analysis:	Rainfall Prediction
Created:	2018-10-29 10:50 pm

- Once the rainfall category is predicted it send the back the category of rainfall predicted by the machine learning algorithm to node 1 (Soil Monitoring node) which depending on the rainfall category switches ON the motor (that pumps water). Like in our case if rainfall category is 0 - motor is switched ON for 10 seconds, 1 - motor is switched ON for 6 seconds, 2 - motor is switched ON for 3 seconds and for 3 - motor is switched On for only 1 second.





- Also we have developed a dashboard for the user using which user can monitor the status at any point of time, also dashboard can be used to manually switch ON/OFF the motor according to the use of user. The link for dashboard is http://go-together.000webhostapp.com. System operates in 2 modes manual and automatic. When manually switched off from dashboard is goes to automatic mode.


https://go-together.000webhostapp.com

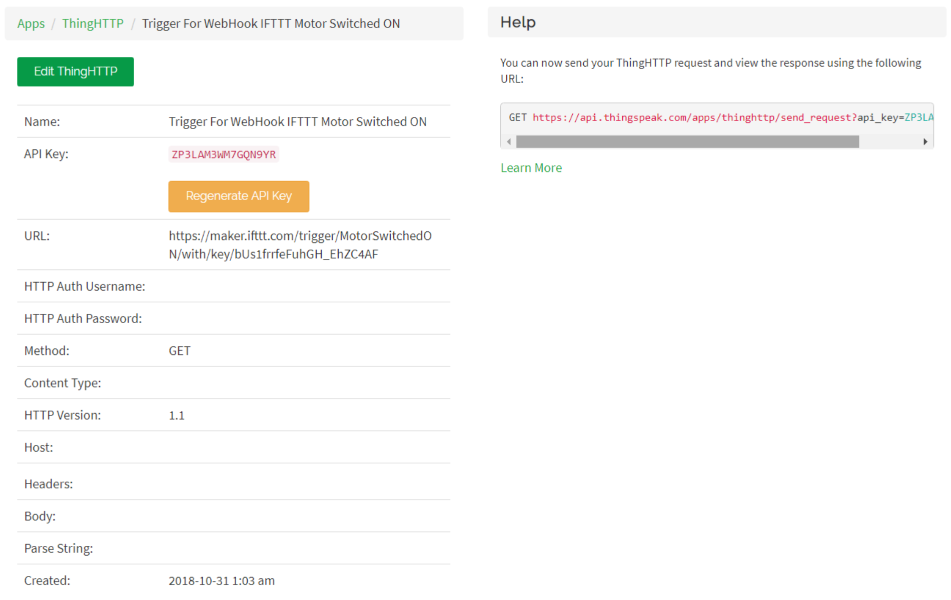
SMART IRRIGATION SYSTEM

FESTURES	VALUES
Temperature	21.00
Average Temperature	22.4
Humidity	75.00
Average Humidity	75.7
Dew point	16.39
Average Dew point	17.893
Rainfall Category	0
Moisture	0
Motor Status	0

- Using webhooks in IFTTT platform http requests are called whenever the motor status changes . ThingHTTP is run with the help of React feature provided by Thingspeak. User gets notified on his/her mobile phone every time the motor switches ON/OFF.

<u>ThingHTTP</u>

- **Motor ON**


Apps / ThingHTTP / Trigger For WebHook IFTTT Motor Switched ON

Name:	Trigger For WebHook IFTTT Motor Switched ON
API Key:	ZP3LAM3WM7GQN9YR
URL:	https://maker.ifttt.com/trigger/MotorSwitchedON/with/key/bUs1frrfeFuhGH_EhZC4AF
HTTP Auth Username:	
HTTP Auth Password:	
Method:	GET
Content Type:	
HTTP Version:	1.1
Host:	
Headers:	
Body:	
Parse String:	
Created:	2018-10-31 1:03 am

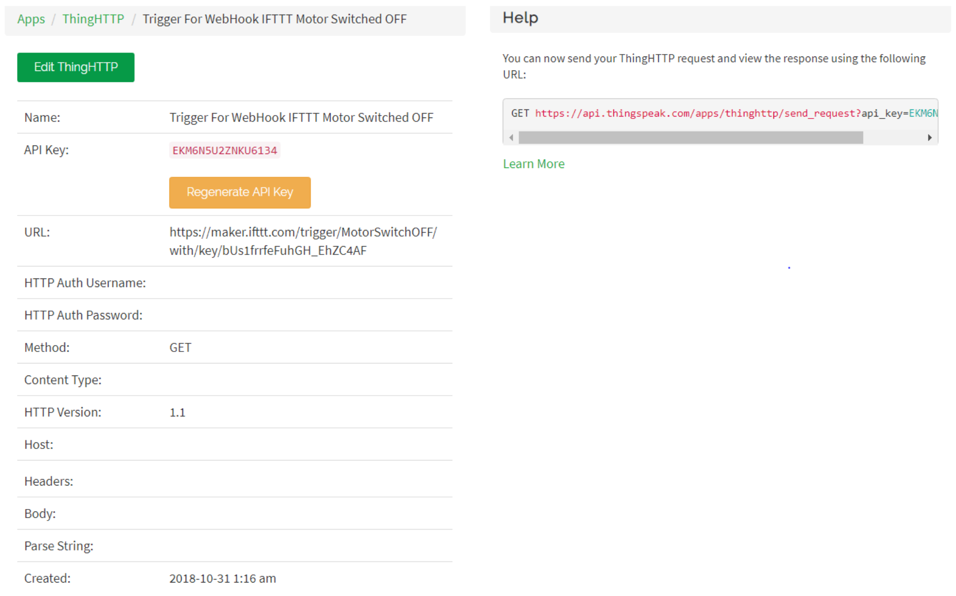
Help

You can now send your ThingHTTP request and view the response using the following URL:

```
GET https://api.thingspeak.com/apps/thinghttp/send_request?api_key=ZP3LA
```

Learn More

- **Motor OFF**



Apps / ThingHTTP / Trigger For WebHook IFTTT Motor Switched OFF

Edit ThingHTTP

Name:	Trigger For WebHook IFTTT Motor Switched OFF
API Key:	EKM6N5U2ZNKU6134
URL:	https://maker.ifttt.com/trigger/MotorSwitchOFF/with/key/bUs1frrfeFuhGH_EhZC4AF
HTTP Auth Username:	
HTTP Auth Password:	
Method:	GET
Content Type:	
HTTP Version:	1.1
Host:	
Headers:	
Body:	
Parse String:	
Created:	2018-10-31 1:16 am

Help

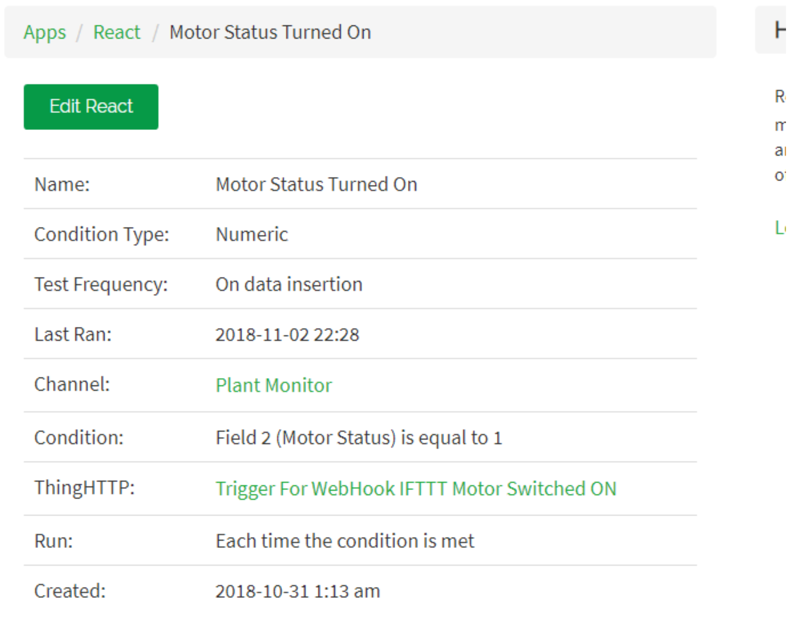
You can now send your ThingHTTP request and view the response using the following URL:

```
GET https://api.thingspeak.com/apps/thinghttp/send_request?api_key=EKM6N
```

Learn More

React

- **Motor ON**

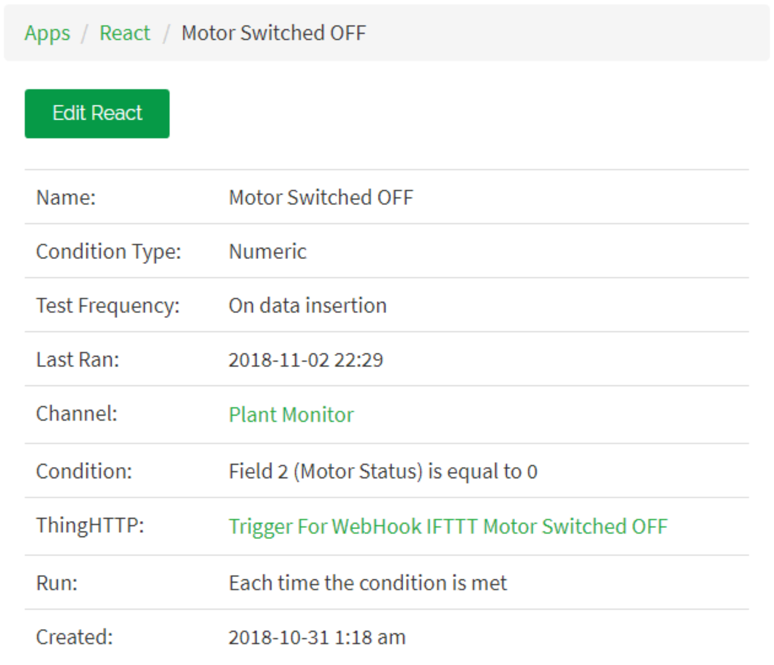


Apps / React / Motor Status Turned On

Edit React

Name:	Motor Status Turned On
Condition Type:	Numeric
Test Frequency:	On data insertion
Last Ran:	2018-11-02 22:28
Channel:	Plant Monitor
Condition:	Field 2 (Motor Status) is equal to 1
ThingHTTP:	Trigger For WebHook IFTTT Motor Switched ON
Run:	Each time the condition is met
Created:	2018-10-31 1:13 am

- **Motor OFF**

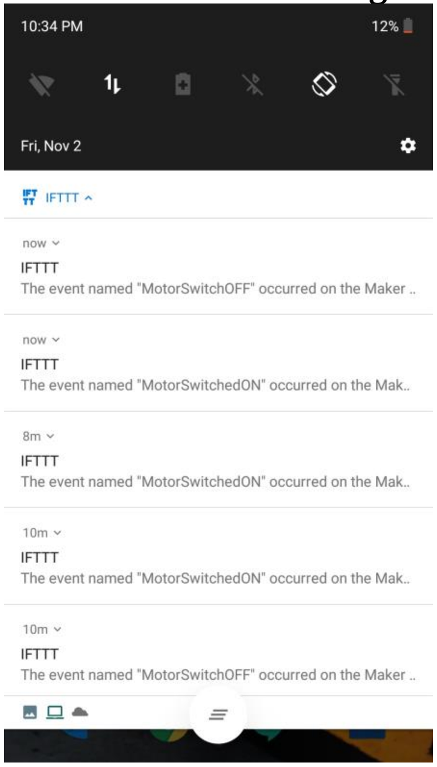


Apps / React / Motor Switched OFF

Edit React

Name:	Motor Switched OFF
Condition Type:	Numeric
Test Frequency:	On data insertion
Last Ran:	2018-11-02 22:29
Channel:	Plant Monitor
Condition:	Field 2 (Motor Status) is equal to 0
ThingHTTP:	Trigger For WebHook IFTTT Motor Switched OFF
Run:	Each time the condition is met
Created:	2018-10-31 1:18 am

Notification Generated on configured device



10:34 PM 12%

Fri, Nov 2

IFTTT

now
IFTTT
The event named "MotorSwitchOFF" occurred on the Maker ..

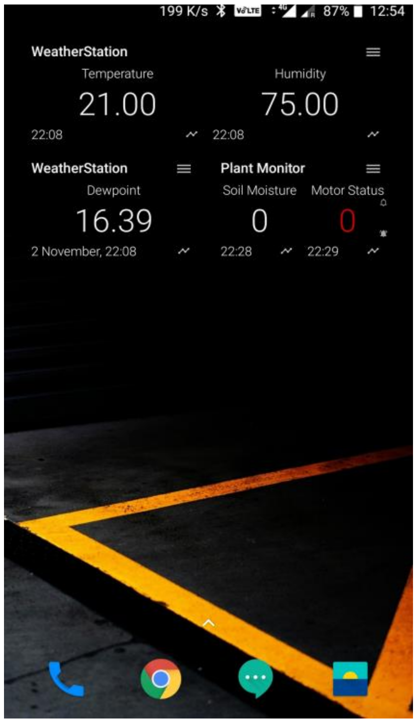
now
IFTTT
The event named "MotorSwitchedON" occurred on the Mak..

8m
IFTTT
The event named "MotorSwitchedON" occurred on the Mak..

10m
IFTTT
The event named "MotorSwitchedON" occurred on the Mak..

10m
IFTTT
The event named "MotorSwitchOFF" occurred on the Maker ..

Widgets for Smartphone for easier monitoring



WeatherStation

Temperature 21.00 22:08

Humidity 75.00 22:08

WeatherStation

Dewpoint 16.39 2 November, 22:08

Plant Monitor

Soil Moisture 0 22:28

Motor Status 0 22:29

PROTOCOL STACK

As Arduino Uno doesn’t have communication capabilities to internet, we require a ESP8266 module with provides it with TCP/IP stack.

COMMUNICATION PROTOCOL USED BY ARDUINO UNO ON PHYSICAL LAYER :

UART (Universal Asynchronous Receiver/Transmitter) : UART is a form of serial communication because data is transmitted as sequential bits (we’ll get to this in a bit). The wiring involved with setting up UART communication is very simple: one line for transmitting data (TX) and one line for receiving data (RX).

SPI (Serial Peripheral Interface) : SPI is different from UART in several key ways: Synchronous. Follows a master-slave model, where there is one master device and multiple slave devices. More than two lines required for implementation

PROTOCOL STACK FOR COMMUNICATION BETWEEN ESP8266 AND CLOUD

Layer	**Protocols**
Application	HTTP
Transport	TCP
Network	IPV4/IPV6
Data Link	WIFI 802.11
Physical	UART,SPI

REST API is used for this constraint environment with uses HTTP over TCP. It works with client-server model. It is used as we only require latest message according when we have necessity of a service. Also for making operations energy efficient, arduino frequently goes in sleep state to save energy.

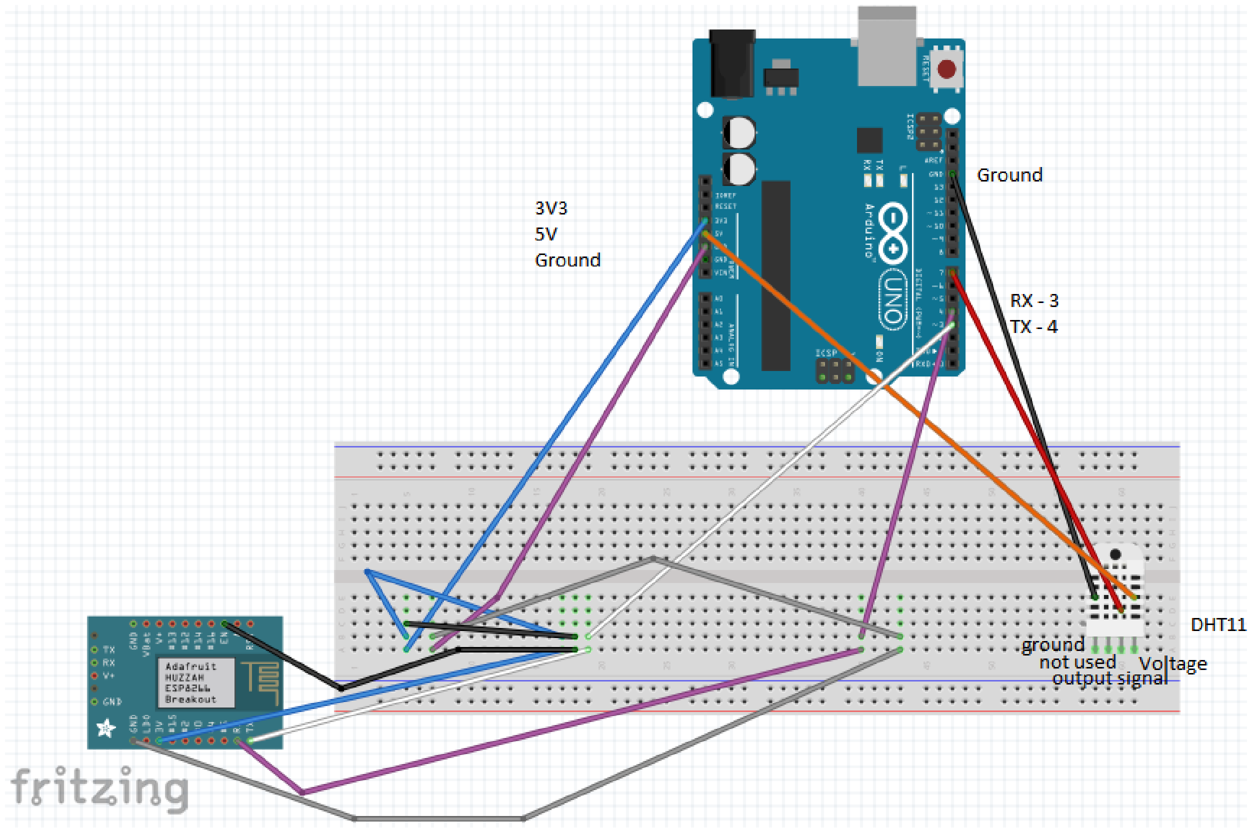
CHALLENGES/CONSTRAINTS

- To get better predictions more features are required like Sunlight intensity, cloud cover and wind speeds. If these features are used with pre-existing once, we would certainly see jump in accuracy of algorithm.
- Area specific meteorological data is required to make prediction more personalised and accurate.
- Keeping the equipment compact is also a challenge, as it constitutes in better usability and portability.
- Keeping cost low at prototyping is also a challenge. Mass production can drive the cost down significantly so it can be affordable to consumers.
- Plant specific information can also be useful for better usability of the product. It can be manually calibrated with ease during set-up time.

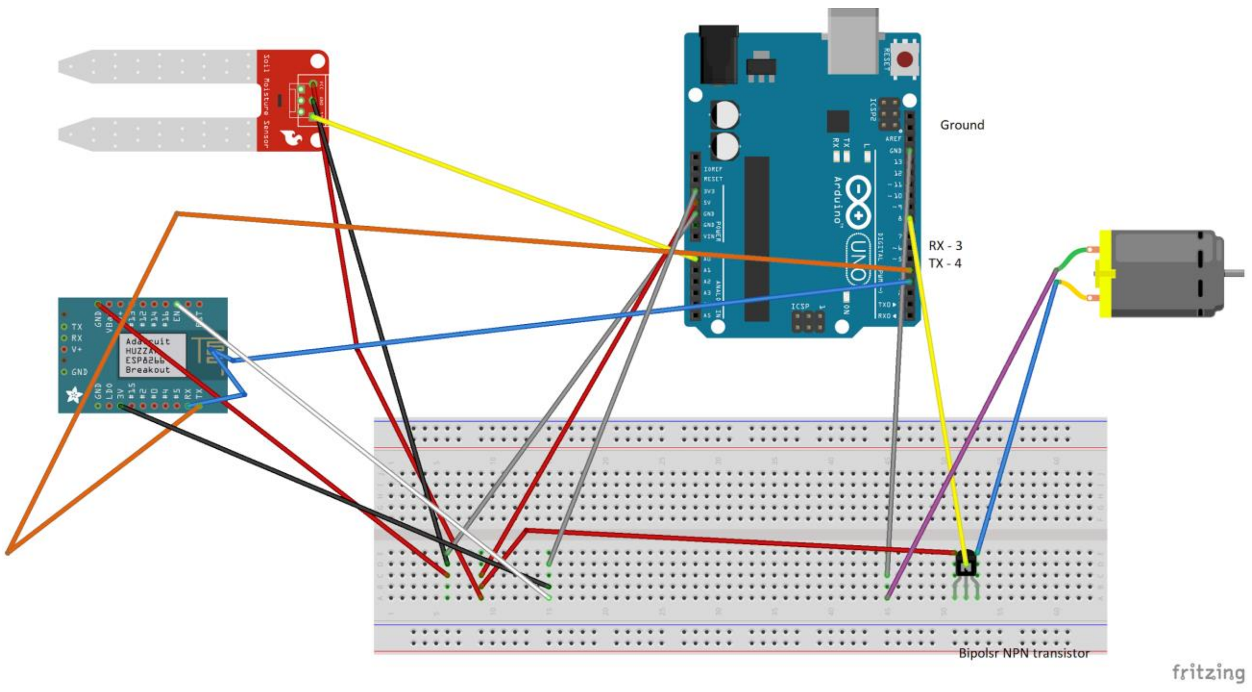
CIRCUIT DIAGRAMS

Circuits diagrams with pin configuration are made using a free software called fritzing http://fritzing.org/home/. Fritzing is an open-source hardware initiative that make electronics accessible as a creative material for anyone.

- Weather Station



- Plant Monitoring



ARDUINO CODE

1. **Weather_Station.ino**

```
#include<SoftwareSerial.h> //to interface esp8266 with arduino through serial
communication
#include <dht.h>

dht DHT; //DHT sensor object
#define DHT11_PIN 7
SoftwareSerial esp8266(3,4);//set the software serial pins RX=3,TX=4
#define SID"piyush"
#define PASS"okaytry12"

void sendAT(String command,const int timeout){
  String response="";
  esp8266.print(command);
  long int time=millis();
  while((time+timeout)>millis()){
    while(esp8266.available()){
      char c = esp8266.read();
      response +=c;
    }
  }
  Serial.println(command);

}

void connectwifi(){
  delay(1000);
  sendAT("AT\r\n",2000);
  sendAT("AT+CWMODE=1\r\n",2000);//call sendAT function to set esp8266 to
station mode
  sendAT("AT+CWJAP=\""SID"\",\""PASS"\"\r\n",2000);//AT command to
connect with wifi network
  while(!esp8266.find("OK")){
    //wait for connection
  }
  Serial.println("WIFI CONNECTED");
  sendAT("AT+CIFSR\r\n",2000);//AT command to print IP address on serial
monitor
  sendAT("AT+CIPMUX=0\r\n",2000);//AT command to set esp8266 to single
connection
}
```

```
void setup() {
  Serial.begin(115200);
  esp8266.begin(115200);//begin the software serial communication with baud rate
115200
  sendAT("AT+RST\r\n",2000);//call sendAT function to send reset AT command
  connectwifi();
}

void loop() {
  int chk = DHT.read11(DHT11_PIN);
  double gamma = log(DHT.humidity / 100) + ((17.62 * DHT.temperature) / (243.5
+ DHT.temperature));
  double dp = 243.5 * gamma / (17.62 - gamma);
  //int converted to string so that values can be passed to cloud
  String Temperature=String(DHT.temperature);//convert integer to string data type
  String Humidity=String(DHT.humidity);
  String Dewpoint=String(dp);
  Serial.print("Temperature: ");
  Serial.println(Temperature);
  Serial.print("Humidity: ");
  Serial.println(Humidity);
  Serial.print("Dewpoint: ");
  Serial.println(Dewpoint);
  updateThinkspeak(Temperature,Humidity,Dewpoint);//call the function to update
Thingspeak channel
  delay(10000);
}
void updateThinkspeak(String Temperature,String Humidity,String Dewpoint){
   sendAT("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n",2000);//tcp
connection with domain name and port no.
   delay(2000);
   String cmdlen;
   String cmd = "GET
/update?api_key=&field1="+Temperature+"&field2="+Humidity+"&field3="+De
wpoint+"\r\n";//update the temperature,humidity and dewpoint values to
thingspeak channel
   cmdlen=cmd.length();
   sendAT("AT+CIPSEND="+cmdlen+"\r\n",2000);
   esp8266.print(cmd);
   sendAT("AT+CIPCLOSE\r\n",2000);
   delay(1000);
}
```

2. Plant_Monitor.ino

```
#include<SoftwareSerial.h> //to interface esp8266 with arduino through serial
communication
SoftwareSerial esp8266(3,4);//set the software serial pins RX=3,TX=4
#define SID"piyush"
#define PASS"okaytry12"
String response="";
int sensor_pin = A0;

int output_value ;
int a;
void sendAT(String command,const int timeout){
  response="";
  int copy=0;
  esp8266.print(command);
  long int time=millis();
  while((time+timeout)>millis()){
    while(esp8266.available()){
      char c = esp8266.read();
      if(isAlpha(c))
        copy=0;
      if(copy)
        response +=c;
      if(c==':')
        copy=1;

    }
  }
  Serial.println(command);
}

void connectwifi(){
  delay(1000);
  sendAT("AT\r\n",2000);
  sendAT("AT+CWMODE=1\r\n",2000);//call sendAT function to set esp8266 to
station mode
  sendAT("AT+CWJAP=\""SID"\",\""PASS"\"\r\n",2000);//AT command to
connect with wifi network
  while(!esp8266.find("OK")){
    //wait for connection
  }
  Serial.println("WIFI Connected");
  sendAT("AT+CIFSR\r\n",2000);//AT command to print IP address on serial
monitor
  sendAT("AT+CIPMUX=0\r\n",2000);//AT command to set esp8266 to single
connection
}


void setup() {

   Serial.begin(115200);
   pinMode(8, OUTPUT);
   int a=0;
   esp8266.begin(115200);//begin the software serial communication with baud rate
115200
   sendAT("AT+RST\r\n",2000);//call sendAT function to send reset AT command
   connectwifi();
   delay(2000);

   }

void loop() {

   output_value= analogRead(sensor_pin);

   output_value = map(output_value,1023,73,0,100);
   Serial.print("Mositure : ");

   Serial.println(output_value);
   String moisture=String(output_value);
   String motor;
      a=0;
      int auth=getData2();
      if(auth==1)
      {
        Serial.println("Admin forced motor to start");
        a=1;
        motor=String(a);
        updateThinkspeak(moisture,motor);
        digitalWrite(8, HIGH);
      }
      else{
        if(output_value<20)
        {
            digitalWrite(8, LOW);
```

```
    int category=getData();
  if(category==0)
  {
    a=1;
    motor=String(a);
    updateThinkspeak(moisture,motor);
    Serial.println("Start Motor for 10 secs");
    digitalWrite(8, HIGH);
    delay(10000);
    digitalWrite(8,LOW);
    Serial.println("Motor Off");
    output_value= analogRead(sensor_pin);
    output_value = map(output_value,1023,73,0,100);
    String moisture=String(output_value);
    a=0;
}
if(category==1)
{
  a=1;
  motor=String(a);
  updateThinkspeak(moisture,motor);
  Serial.println("Start Motor for 6 secs");
  digitalWrite(8, HIGH);
  delay(6000);
  digitalWrite(8,LOW);
  Serial.println("Motor Off");
  output_value= analogRead(sensor_pin);
  output_value = map(output_value,1023,73,0,100);
  String moisture=String(output_value);
  a=0;
}
if(category==2)
{
  a=1;
  motor=String(a);
  updateThinkspeak(moisture,motor);
  Serial.println("Start Motor for 2 secs");
  digitalWrite(8, HIGH);
  delay(2000);
  digitalWrite(8,LOW);
  Serial.println("Motor Off");
  output_value= analogRead(sensor_pin);
  output_value = map(output_value,1023,73,0,100);
  String moisture=String(output_value);
```

```
            a=0;
          }
          if(category==3)
          {
            a=1;
            motor=String(a);
            updateThinkspeak(moisture,motor);
            Serial.println("Start Motor for 1 sec for maintain threshold moisture");
            digitalWrite(8, HIGH);
            delay(1000);
            digitalWrite(8,LOW);
            Serial.println("Motor Off");
            output_value= analogRead(sensor_pin);
            output_value = map(output_value,1023,73,0,100);
            String moisture=String(output_value);
            a=0;
            Serial.println("Heavy Rainfall Prediction");
          }
          a=0;
          motor=String(a);
          updateThinkspeak(moisture,motor);


     }
     else
     {
          Serial.println("Enough moisture");
          a=0;
          motor=String(a);
          updateThinkspeak(moisture,motor);
     }



    }


     delay(4000);


  }
```

```
void updateThinkspeak(String Moisture,String Motor){

   sendAT("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n",2000);//tcp
connection with domain name and port no.
   delay(2000);
   String cmdlen;
   String cmd = "GET /update?api_key=
&field1="+Moisture+"&field2="+Motor+"\r\n";//update the temperature,humidity
and dewpoint values to thingspeak channel
   cmdlen=cmd.length();
   Serial.println(cmd);
   sendAT("AT+CIPSEND="+cmdlen+"\r\n",2000);
   esp8266.print(cmd);
   sendAT("AT+CIPCLOSE\r\n",2000);
   delay(1000);

}
int getData(){
  sendAT("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n",2000);
  delay(2000);
  String cmdlen;
  String fieldname="field1";
  String cmd="GET /channels/channel_id/field/"+fieldname+"/last"+"\r\n";
  ///channels/channel_id/fields/1.csv?api_key="+API+"&results=1"+"\r\n"
  cmdlen=cmd.length();
  sendAT("AT+CIPSEND="+cmdlen+"\r\n",2000);
  esp8266.print(cmd);
  sendAT("AT+CIPCLOSE\r\n",2000);
  Serial.print("Rainfall Category Predicted: ");
  int category = response.toInt();
  Serial.println(category);
  return(category);
}
int getData2(){
  sendAT("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n",2000);
  delay(2000);
  String cmdlen;
  String fieldname="field1";
  String cmd="GET /channels/channel_id/field/"+fieldname+"/last"+"\r\n";
  cmdlen=cmd.length();
  sendAT("AT+CIPSEND="+cmdlen+"\r\n",2000);
  esp8266.print(cmd);
  sendAT("AT+CIPCLOSE\r\n",2000);
  Serial.print("Admin Mode: ");
  int category = response.toInt();
  Serial.println(category);
  return(category);
  delay(5000);


}
```

MACHINE LEARNING ALGORITHM

In the algorithm we have used linear regression with gradient descent to predict the rainfall. We have used gradient descent for error optimization with learning rate of 0.01 and make it loop for 4000 iterations. Initially before training the algorithm we have applied regularization for better results . Also we have divided our dataset into train and test, so that we can test our model and check its mean square error. We see that our model has performed very well with training error of 0.082 and testing error of 0.015.

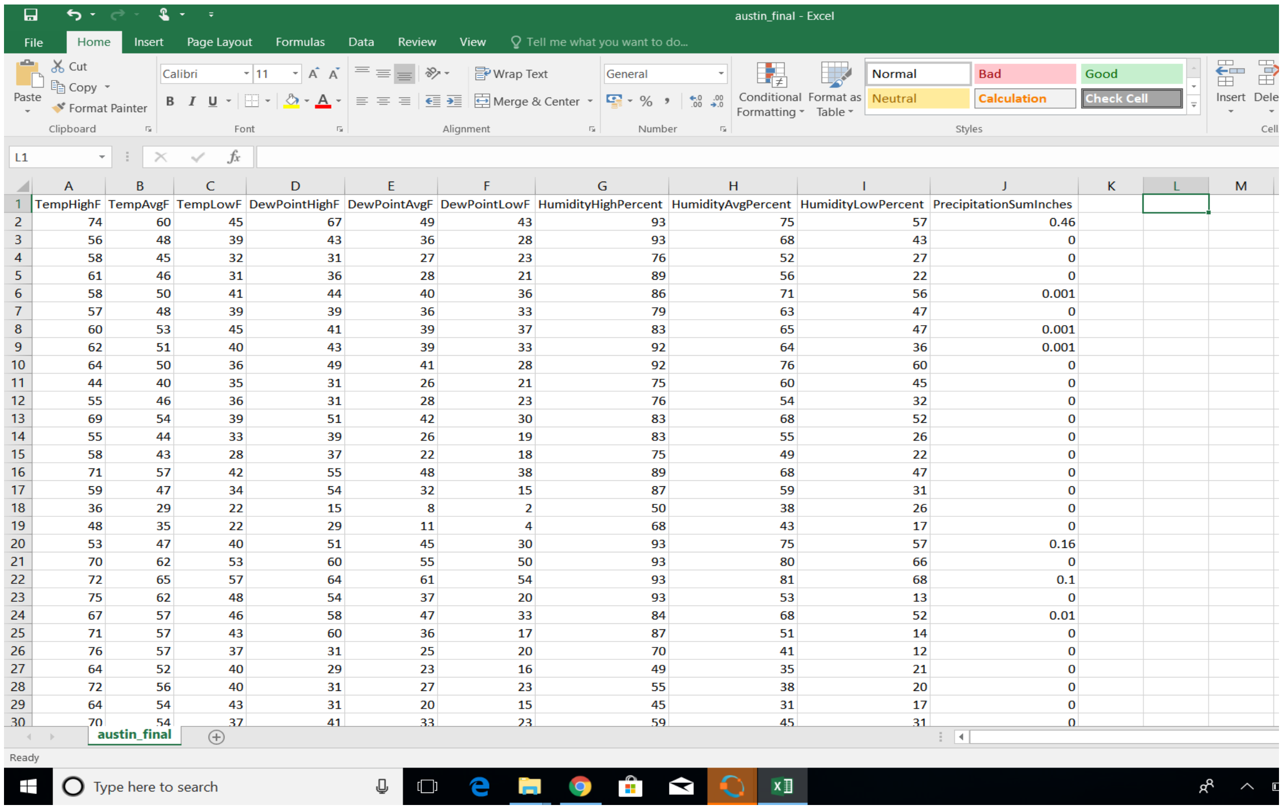
CODE:

```
%Linear Regression with Gradient Descent

%% Clear and Close Figures
clear ; close all; clc

%Input must contain feature columns followed by dependent variable column at
end
data = csvread('austin_final.csv');

%Gradient Descent parameters
alpha = 0.01; num_iters = 4000;
%percentage of data to use for training
train_perc = .95;

%extract columns to use
X = data(:,1:end-1);
X_orig = X;
y = data(:,end);

% Print out some data points
fprintf('First 3 examples from the dataset: \n');
fprintf(' x = [%.0f %.0f %.0f %.0f %.0f %.0f %.0f %.0f %.0f %.0f %.0f %.0f %.0f
%.0f %.0f %.0f %.0f %.0f %.0f], y = %.0f \n', [X(1:3,:) y(1:3,:)]');
fprintf('Program paused. Press enter to continue.\n');
pause;

%split into training and test sets:
test_rows = round(size(X,1)*(1-train_perc)); %number of rows to use in test set
X_test = X(1:test_rows,:); y_test = y(1:test_rows,:);%this is the test set
X = X(test_rows+1:end,:); y = y(test_rows+1:end,:);%this is the training set
```

```
%Compute mean and standard deviation, normalize X
mu = mean(X); sigma = std(X);
X = (X-repmat(mu,[size(X,1) 1]))./repmat(sigma,[size(X,1) 1]);
X_test = (X_test-repmat(mu,[size(X_test,1) 1]))./repmat(sigma,[size(X_test,1) 1]);


%Add intercept term to X
X = [ones(size(X,1), 1) X];
X_test = [ones(size(X_test,1), 1) X_test];

% Init Theta and Run Gradient Descent
theta = zeros(size(X,2), 1);
[theta, J_history] = gradientDescentMulti(X, y, theta, alpha, num_iters);

% Plot the convergence graph
figure; plot(1:numel(J_history), J_history);
xlabel('Number of iterations');ylabel('Cost J');

% Display gradient descent result
fprintf('Theta computed from gradient descent: \n');
fprintf(' %f \n', theta);

% Display cost function start and end values
fprintf('Cost function start: %g\n',J_history(1));
fprintf('Cost function end: %g\n',J_history(end));

%use test set for prediction - using mean & stdev of entire set
y_hat = X_test*theta;

%Compute training set error
J_train = computeCostMulti(X, y, theta);

%Compute test set error
J_test = computeCostMulti(X_test,y_test,theta);

fprintf('Program paused. Press enter to continue.\n');
pause;

for idx = 1:numel(y_hat)
  if y_hat(idx)<0.2
    y_hat(idx) = 0;
  endif
end
```

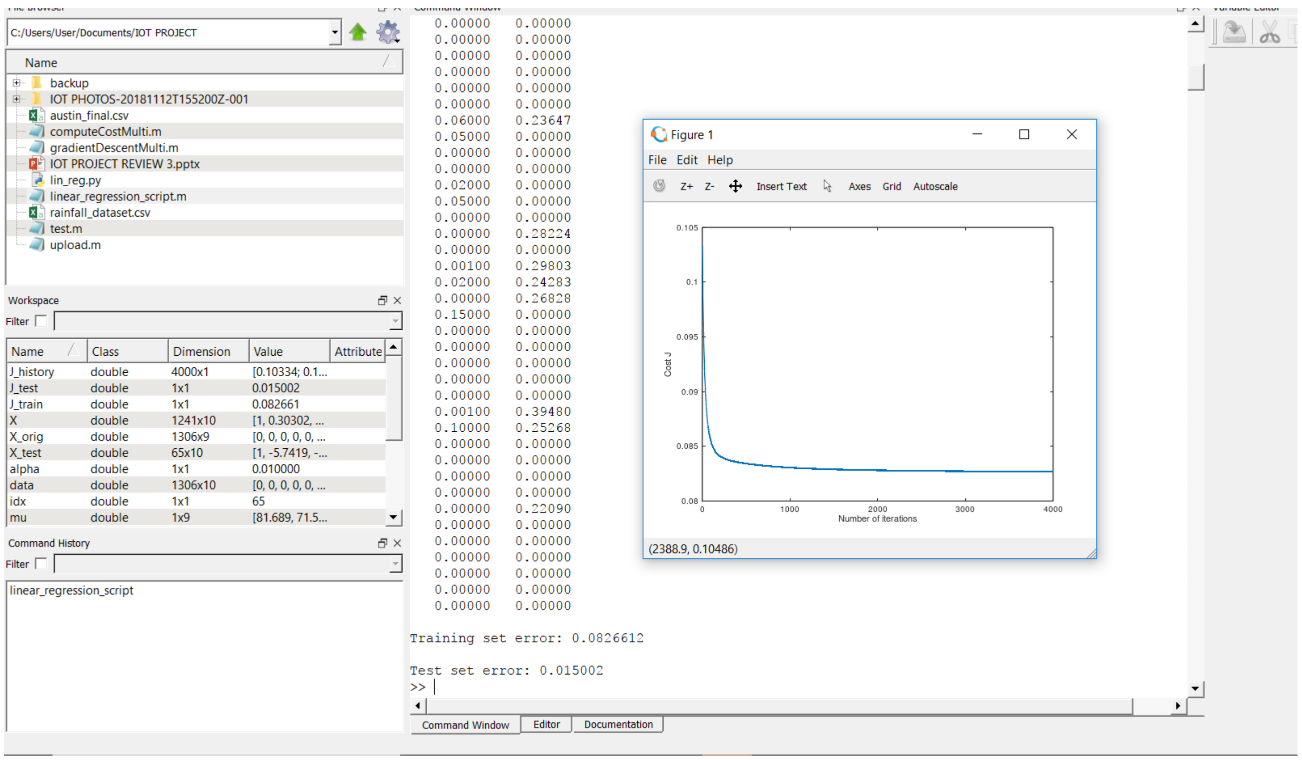
```
%print test set variable - actual and prediction
fprintf('\nLinear fit:\n\n')
fprintf('\ty\t\ty_hat\n');
disp([y_test,y_hat]);
fprintf('\nTraining set error: %g\n',J_train);
fprintf('\nTest set error: %g\n',J_test);
%EOF
```

DATASET SCREENSHOT:



	A	B	C	D	E	F	G	H	I	J
1	TempHighF	TempAvgF	TempLowF	DewPointHighF	DewPointAvgF	DewPointLowF	HumidityHighPercent	HumidityAvgPercent	HumidityLowPercent	PrecipitationSumInches
2	74	60	45	67	49	43	93	75	57	0.46
3	56	48	39	43	36	28	93	68	43	0
4	58	45	32	31	27	23	76	52	27	0
5	61	46	31	36	28	21	89	56	22	0
6	58	50	41	44	40	36	86	71	56	0.001
7	57	48	39	39	36	33	79	63	47	0
8	60	53	45	41	39	37	83	65	47	0.001
9	62	51	40	43	39	33	92	64	36	0.001
10	64	50	36	49	41	28	92	76	60	0
11	44	40	35	31	26	21	75	60	45	0
12	55	46	36	31	28	23	76	54	32	0
13	69	54	39	51	42	30	83	68	52	0
14	55	44	33	39	26	19	83	55	26	0
15	58	43	28	37	22	18	75	49	22	0
16	71	57	42	55	48	38	89	68	47	0
17	59	47	34	54	32	15	87	59	31	0
18	36	29	22	15	8	2	50	38	26	0
19	48	35	22	29	11	4	68	43	17	0
20	53	47	40	51	45	30	93	75	57	0.16
21	70	62	53	60	55	50	93	80	66	0
22	72	65	57	64	61	54	93	81	68	0.1
23	75	62	48	54	37	20	93	53	13	0
24	67	57	46	58	47	33	84	68	52	0.01
25	71	57	43	60	36	17	87	51	14	0
26	76	57	37	31	25	20	70	41	12	0
27	64	52	40	29	23	16	49	35	21	0
28	72	56	40	31	27	23	55	38	20	0
29	64	54	43	31	20	15	45	31	17	0
30	70	54	37	41	33	23	59	45	31	0

OUTPUT:



C:/Users/User/Documents/IOT PROJECT

Name
backup
IOT PHOTOS-20181112T155200Z-001
austin_final.csv
computeCostMulti.m
gradientDescentMulti.m
IOT PROJECT REVIEW 3.pptx
lin_reg.py
linear_regression_script.m
rainfall_dataset.csv
test.m
upload.m

Workspace

Filter

Name	Class	Dimension	Value	Attribute
J_history	double	4000x1	[0.10334; 0.1...	
J_test	double	1x1	0.015002	
J_train	double	1x1	0.082661	
X	double	1241x10	[1, 0.30302, ...	
X_orig	double	1306x9	[0, 0, 0, 0, 0, ...	
X_test	double	65x10	[1, -5.7419, -...	
alpha	double	1x1	0.010000	
data	double	1306x10	[0, 0, 0, 0, 0, ...	
idx	double	1x1	65	
mu	double	1x9	[81.689, 71.5...	

Command History

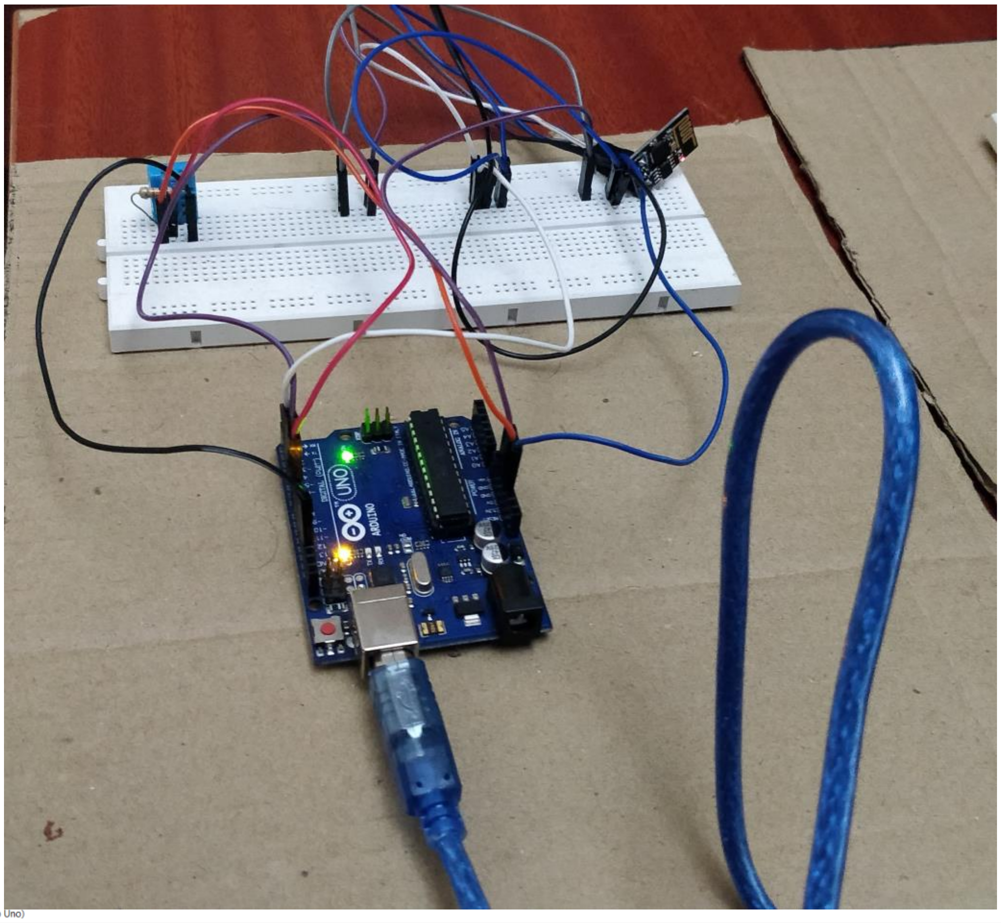
Filter

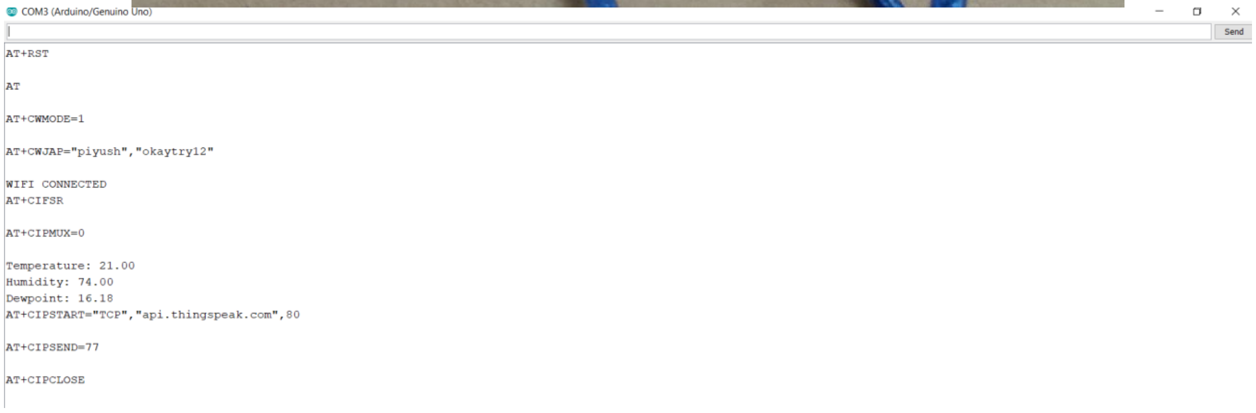
```
linear_regression_script
```

```
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.06000   0.23647
   0.05000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.02000   0.00000
   0.05000   0.00000
   0.00000   0.00000
   0.00000   0.28224
   0.00000   0.00000
   0.00100   0.29803
   0.02000   0.24283
   0.00000   0.26828
   0.15000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00100   0.39480
   0.10000   0.25268
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.22090
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000
   0.00000   0.00000

Training set error: 0.0826612

Test set error: 0.015002
>> |
```

IMPLEMENTATION IMAGES

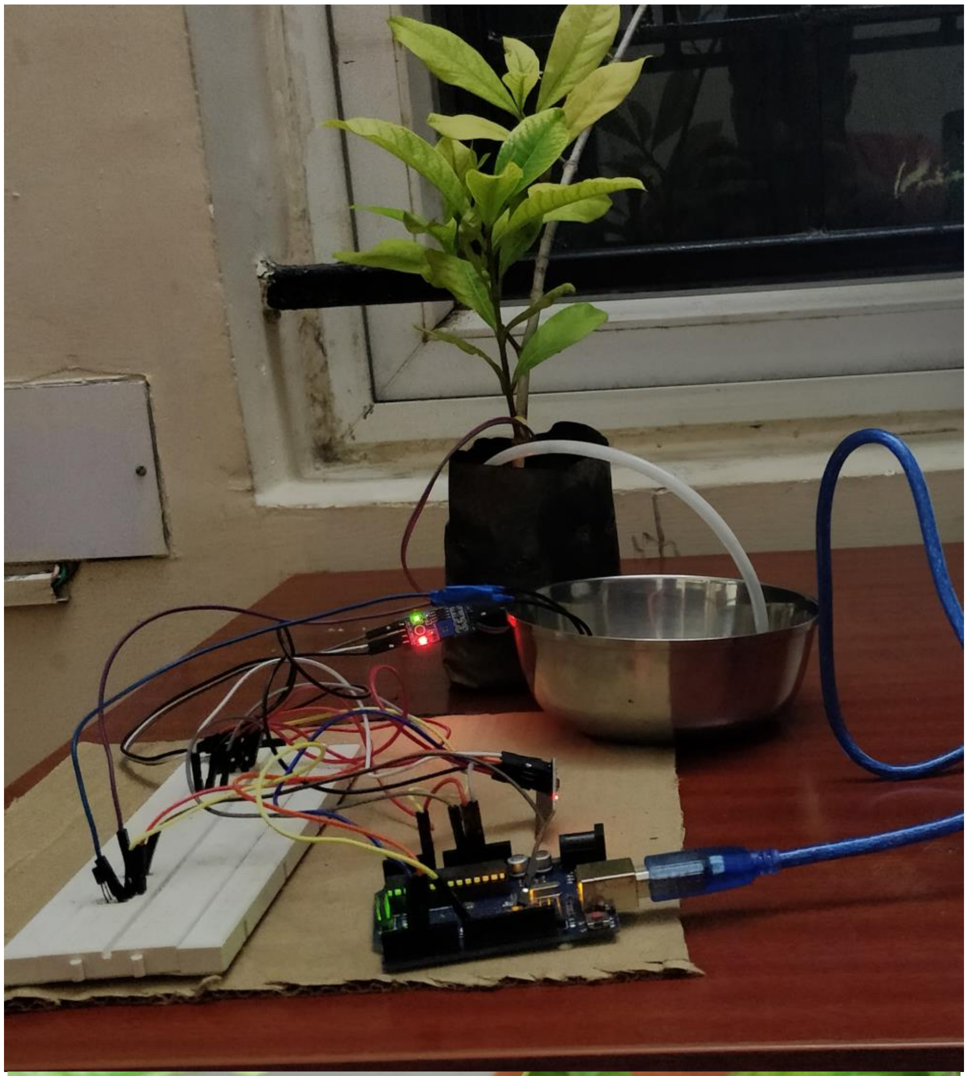
- Weather Station



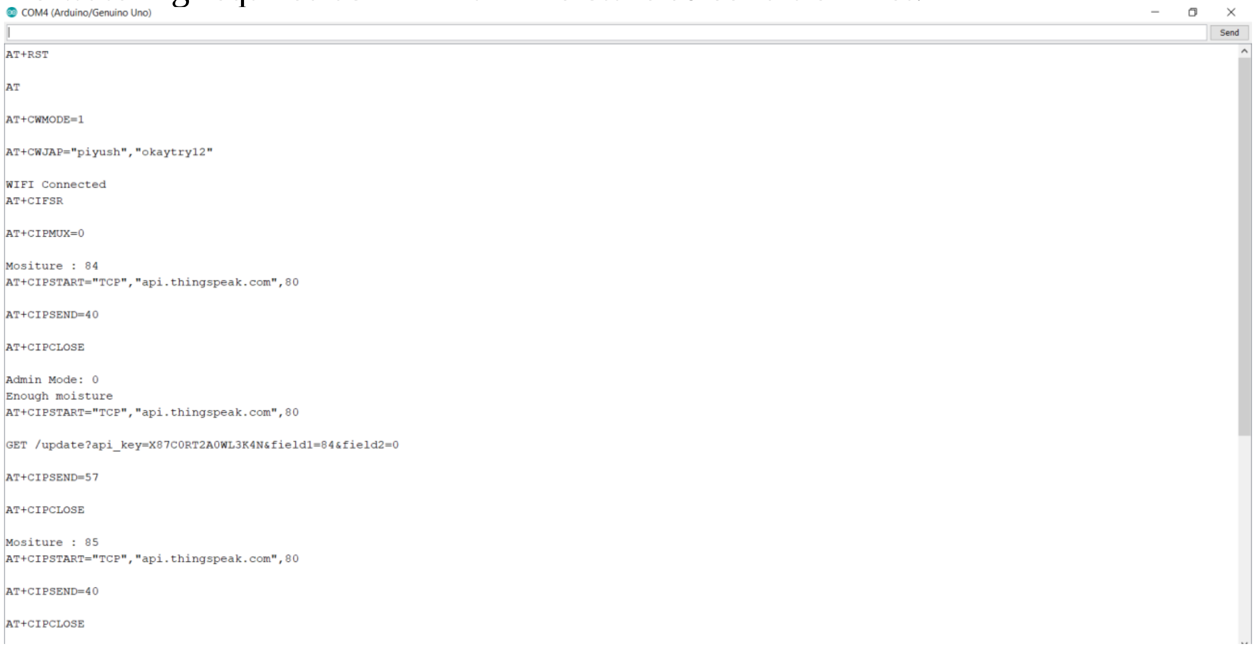


```
AT+RST

AT

AT+CWMODE=1

AT+CWJAP="piyush","okaytry12"

WIFI CONNECTED
AT+CIFSR

AT+CIPMUX=0

Temperature: 21.00
Humidity: 74.00
Dewpoint: 16.18
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=77

AT+CIPCLOSE
```

- Plant Monitoring

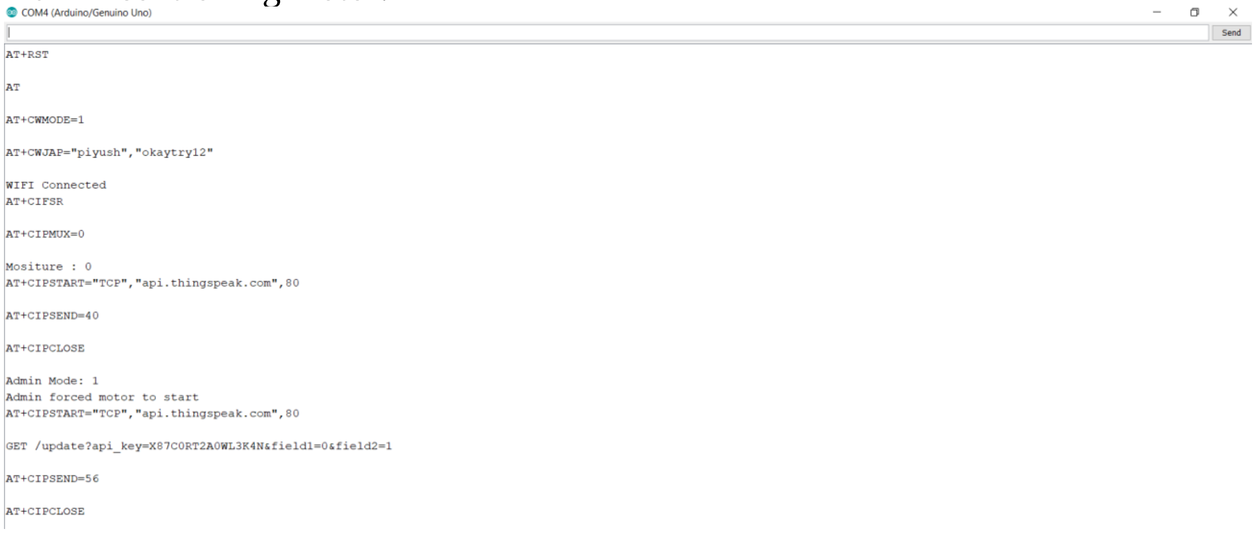


No watering required as minimum moisture % condition met:



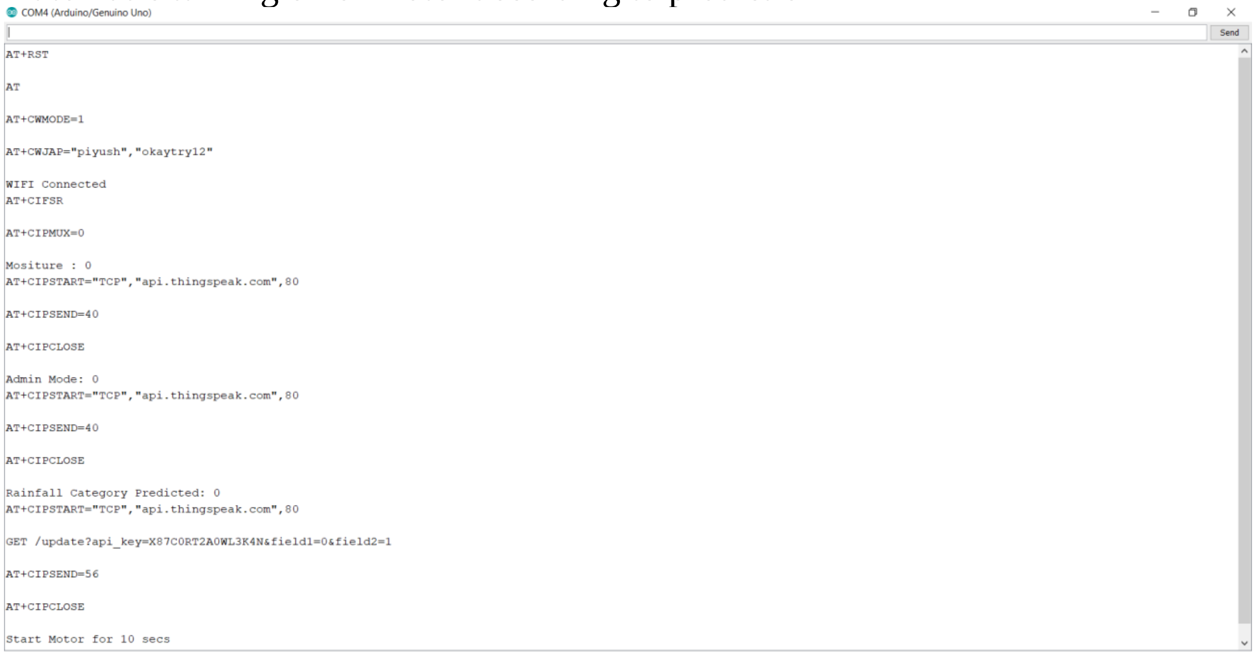
```
AT+RST

AT

AT+CWMODE=1

AT+CWJAP="piyush","okaytry12"

WIFI Connected
AT+CIFSR

AT+CIPMUX=0

Mositure : 84
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=40

AT+CIPCLOSE

Admin Mode: 0
Enough moisture
AT+CIPSTART="TCP","api.thingspeak.com",80

GET /update?api_key=X87C0RT2A0WL3K4N&field1=84&field2=0

AT+CIPSEND=57

AT+CIPCLOSE

Mositure : 85
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=40

AT+CIPCLOSE
```

Admin controlling motor:



```
AT+RST

AT

AT+CWMODE=1

AT+CWJAP="piyush","okaytry12"

WIFI Connected
AT+CIFSR

AT+CIPMUX=0

Mositure : 0
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=40

AT+CIPCLOSE

Admin Mode: 1
Admin forced motor to start
AT+CIPSTART="TCP","api.thingspeak.com",80

GET /update?api_key=X87C0RT2A0WL3K4N&field1=0&field2=1

AT+CIPSEND=56

AT+CIPCLOSE
```

Automatic turning on of motor according to prediction



```
AT+RST

AT

AT+CWMODE=1

AT+CWJAP="piyush","okaytry12"

WIFI Connected
AT+CIFSR

AT+CIPMUX=0

Mositure : 0
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=40

AT+CIPCLOSE

Admin Mode: 0
AT+CIPSTART="TCP","api.thingspeak.com",80

AT+CIPSEND=40

AT+CIPCLOSE

Rainfall Category Predicted: 0
AT+CIPSTART="TCP","api.thingspeak.com",80

GET /update?api_key=X87C0RT2A0WL3K4N&field1=0&field2=1

AT+CIPSEND=56

AT+CIPCLOSE

Start Motor for 10 secs
```