**Exp No :**

**Date: /06/2022**

**Smart Irrigation System using IoT**

**Aim:**

To build a prototype of an IoT system using Thingspeak cloud and Raspberry Pi 4B and develop a smart irrigation system which automates the irrigation and monitors the health of soil of the agricultural field using appropriate sensors.

**Hardware requirements:**

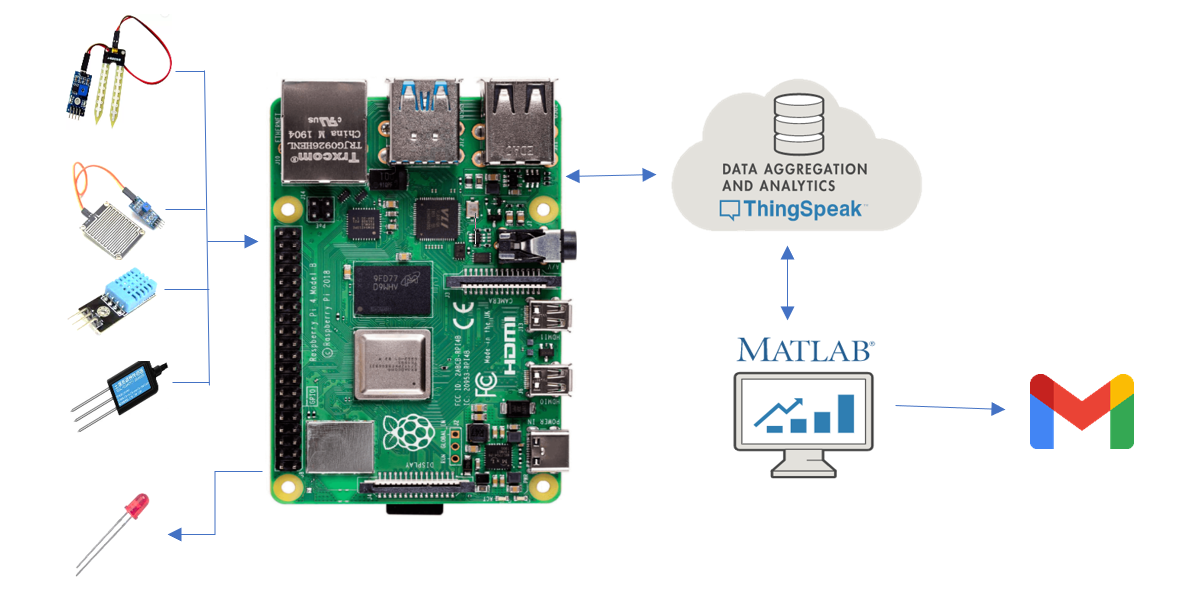
Raspberry Pi 4B, LED ,resistors ,npk , rain , DHT11,moisture sensors

**Software requirements:**

Raspbian OS , Python IDLE v3.7 , Thingspeak cloud.

**Theory:**

Smart irrigation systems can improve crop yields while saving water. Smart irrigation systems use IoT devices with soil moisture ,rain and temperature/humidity sensors to determine the amount of moisture in the soil and air, hotness in the environment and release the flow of water through the irrigation pipes only when the moisture levels/temperature go below/beyond a predefined threshold. Further the soil composition is recorded and analysed so that the plant growth and yield is maximized . The block diagram of the smart irrigation system is shown in Figure 1.

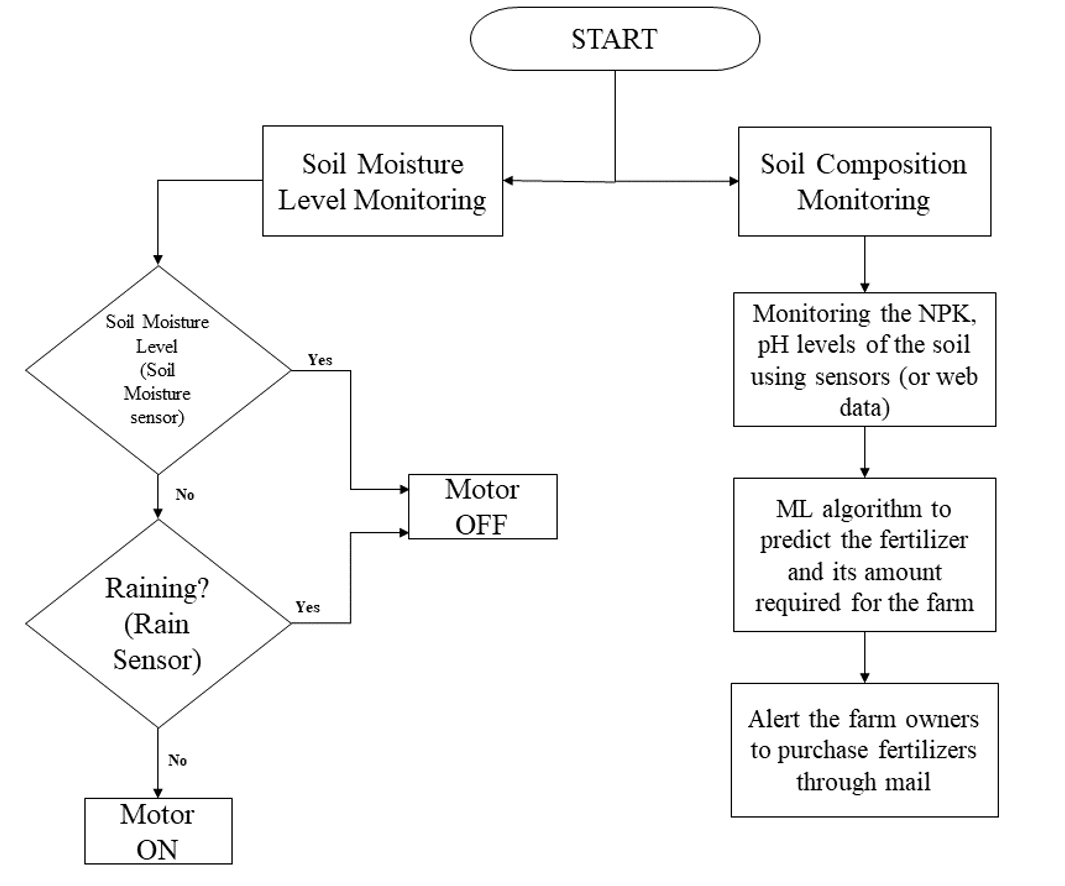


**Figure 1. Smart irrigation system Block diagram**

**Algorithm:**

1. Power up raspberry pi module and Connections are made as per the depicted layout diagram.
2. The soil moisture sensor and rain sensor are placed and the digital output is used here.
3. Raspberry Pi is coded to sense the temperature, humidity ,rain ,npk and soil moisture readings at regular intervals.
4. The sensed values are uploaded in the respective fields of the ThingSpeak channel.
5. When the soil moisture and rain sensor value is high , the LED/motor is turned ON. On all other conditions , the LED/motor is tured OFF.
6. At regular time intervals, the npk values are checked and if they go below a particular threshold, the values are analysed and the required fertilizer is predicted.
7. An email alert is sent to the user regarding the soil composition and fertilizer.

**FlowChart:**

****

**Figure 2. Flow chart for smart irrigation system**

**RPi – code:**

import adafruit\_dht

import requests

import random

#GPIO SETUP

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

channel = 15 #moisture

channel1 = 14 #led

channel2 = 3 #rain

#pin=D17

GPIO.setup(channel, GPIO.IN) #soilmoisture

GPIO.setup(channel2, GPIO.IN) #rain

GPIO.setup(channel1, GPIO.OUT, initial=GPIO.LOW) #led

i=0;

warr=[]

parr=[]

# Enter Your API key here

myAPI = 'P5EFLMNP9GSISCQJ' #write API key

# URL where we will send the data, Don't change it

baseURL = 'https://api.thingspeak.com/update?api\_key=%s' % myAPI

#m= adafruit\_dht.DHT11(pin, use\_pulseio=False)

def rain():

if GPIO.input(channel):

w=0;

print ('No Water Detected!')

sleep(1)

else:

w=1;

print ('Water Detected!')

sleep(1)

return w

def moisture():

if GPIO.input(channel2):

p=0;

print('No rain')

sleep(1)

else:

p=1;

print('raining')

sleep(1)

return p

while True :

try:

temp=random.randrange(35,40)

humi=random.randrange(90,95)

w=rain()

p=moisture()

warr.append(w)

parr.append(p)

# Sending the data to thingspeak

conn = urlopen(baseURL + '&field1=%s&field2=%s&field3=%s&field4=%s'%(temp,humi,w,p))

conn.close()

if (warr[i] == 0) & (parr[i]== 0):

GPIO.output(channel1, GPIO.HIGH)

print('led on')

i=i+1

sleep(5)

GPIO.output(channel1,GPIO.LOW)

else:

GPIO.output(channel1,GPIO.LOW)

print('led off')

sleep(3)

i=i+1

except Exception as e:

print(str(e))

break

**Matlab Analysis :**

ChannelID = 1741844;

%RANDOM DATA GENERATION AND WRITING IN CHANNEL

dataField5 = randi([4 42],1,1);% randi([range],no.of data points,dimension of matrix)

dataField6 = randi([0 42],1,1);

dataField7 = randi([0 19],1,1);

writeKey = 'P5EFLMNP9GSISCQJ'; % Write API Key

thingSpeakWrite(channelID,[dataField5,dataField6,dataField7],'Fields',[5,6,7],'WriteKey',writeKey)

%MAIL ACTUATION

% Provide the ThingSpeak alerts API key. All alerts API keys start with TAK.

alertApiKey = 'TAK1bpuRPQfTKdkbH0f';

% Set the address for the HTTTP call

alertUrl="https://api.thingspeak.com/alerts/send";

% webwrite uses weboptions to add required headers. Alerts needs a ThingSpeak-Alerts-API-Key header.

options = weboptions("HeaderFields", ["ThingSpeak-Alerts-API-Key", alertApiKey ]);

% Set the email subject.

alertSubject = sprintf("Plant soil information");

% Read the recent data.

n= thingSpeakRead(channelID,'Fields',5)

p= thingSpeakRead(channelID,'Fields',6)

k= thingSpeakRead(channelID,'Fields',7)

n\_threshold=40;

p\_threshold=40;

k\_threshold=15;

% Check to make sure the data was read correctly from the channel.

if isempty(n)

alertBody = ' No data read from plant. ';

else

% Set the outgoing message

if ((n<n\_threshold) | (p<p\_threshold) | (k<k\_threshold))

alertBody = ' Soil is depleted of nutrients....Add 28-28 Fertilizer to soil ';

end

end

% Catch errors so the MATLAB code does not disable a TimeControl if it fails

try

webwrite(alertUrl , "body", alertBody, "subject", alertSubject, options);

catch someException

fprintf("Failed to send alert: %s\n", someException.message);

end

**ML Analysis :** (Using Random Forest Classifier)

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import pickle

import warnings

warnings.filterwarnings('ignore')

Data\_path = "/content/drive/MyDrive/Iot mini project/Fertilizer Prediction.csv"

df = pd.read\_csv(Data\_path)

df.head()

# Statistical info

df.describe()

df.isnull().sum()

X = df.drop(columns = ['Fertilizer Name', 'Potassium', 'Temparature'], axis=1)

y = df['Fertilizer Name']

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [2,3])], remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X[1])

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators= 100, criterion = 'gini' , random\_state= 42)

classifier.fit(X\_train, y\_train)

# Predicting the test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix and Calculating the Accuracy

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

acc5 = accuracy\_score(y\_test, y\_pred)

print("Random Forest Classification's Accuracy:", acc5)

y\_random\_test = {'Temparature':[35], 'Humidity ':[64], 'Moisture':[30], 'Soil Type': ['Red'],'Crop Type':['Tobacco'],'Nitrogen':[39],'Potassium':[1], 'Phosphorous':[16] }

test\_df = pd.DataFrame.from\_dict(y\_random\_test)

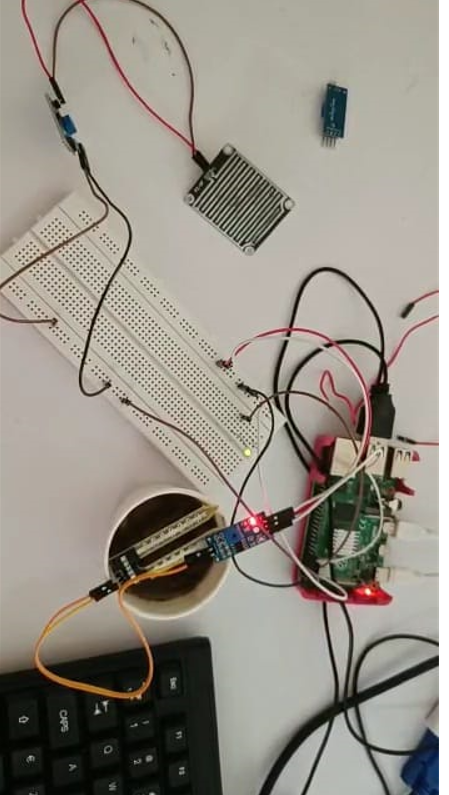
X\_test = test\_df.drop(columns = [ 'Potassium', 'Temparature'], axis=1)

X\_test = np.array(ct.transform(X\_test))

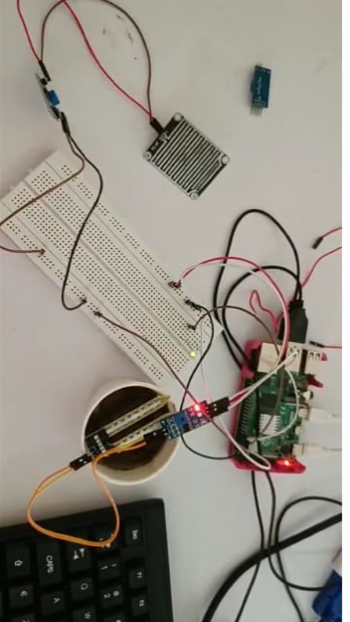
y\_pred = classifier.predict(X\_test)

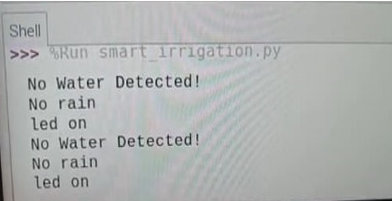
print(y\_pred)

**Output and Discussions:**

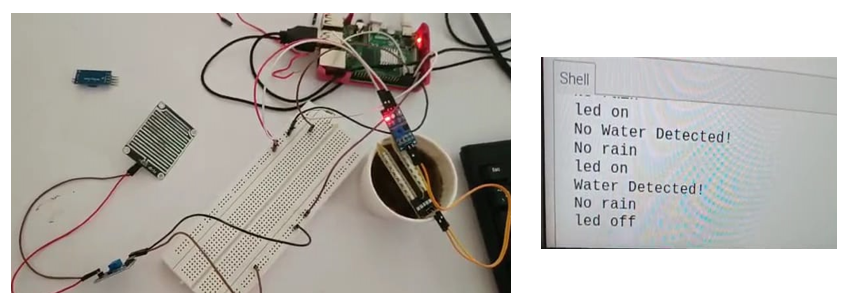
****The real time design of the prototype of smart irrigation system is shown in figure 3. The temperature, humidity ,rain and soil moisture readings are stored in the Cloud as shown in Figure 6. The LED/motor actuation output is shown in Figure 4&5 .At the regular time intervals, using matlab analysis the npk values are got from the channels and sent to ML analysis and the required fertilizer is predicted as shown in Figure 7. And an email is sent to the registered user as shown in Figure 8.

**Figure 3. Connection setup**

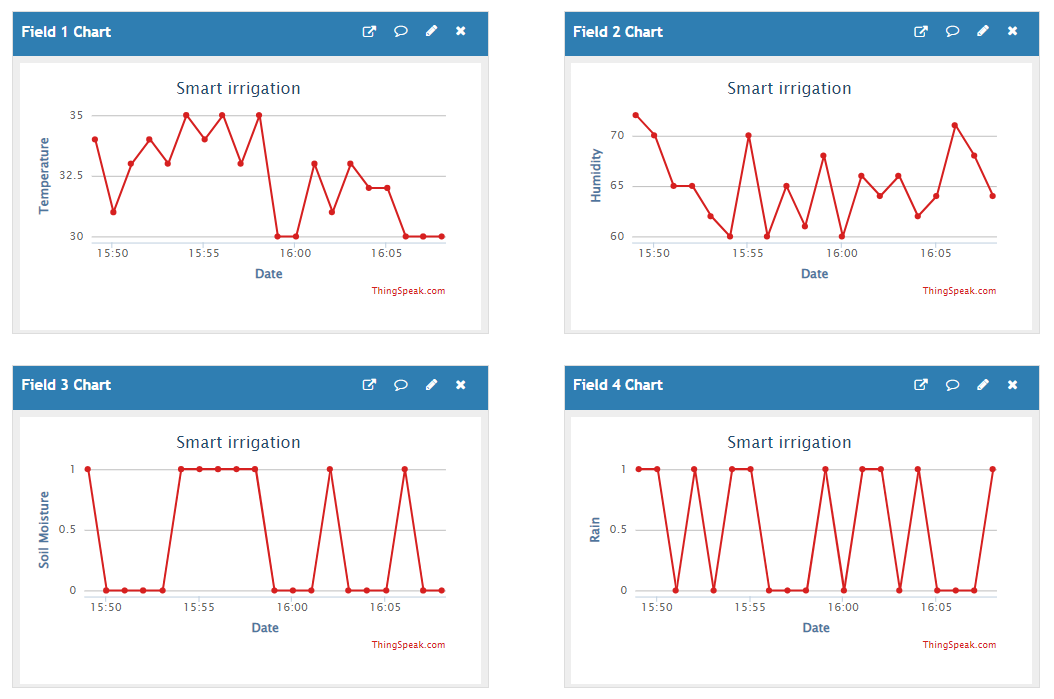
****

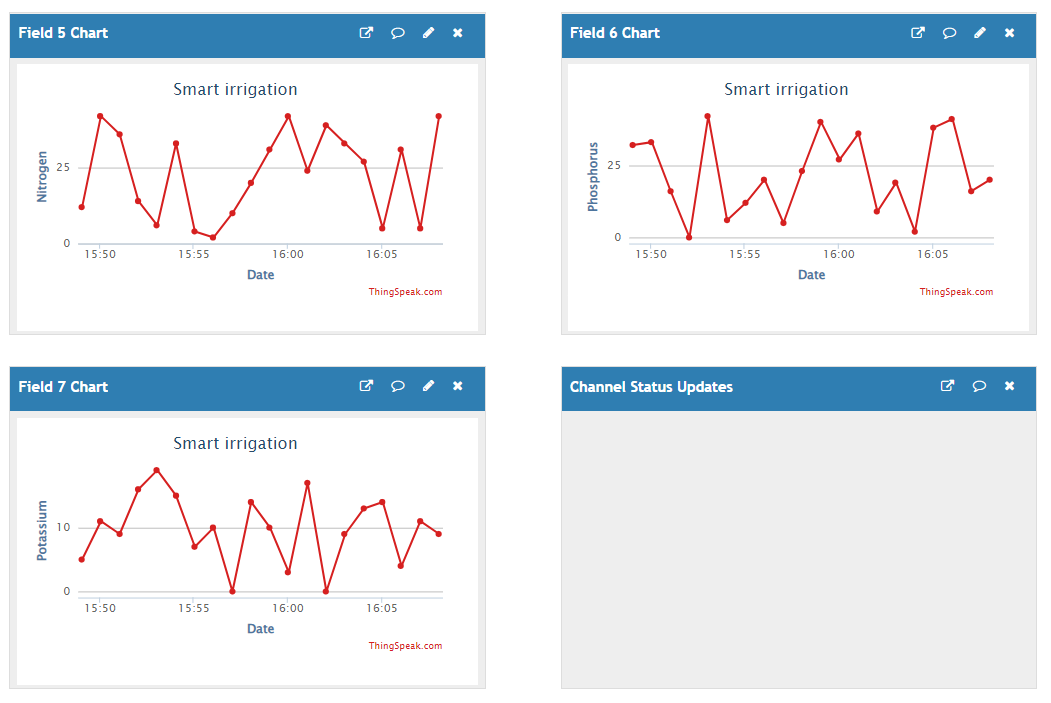
****

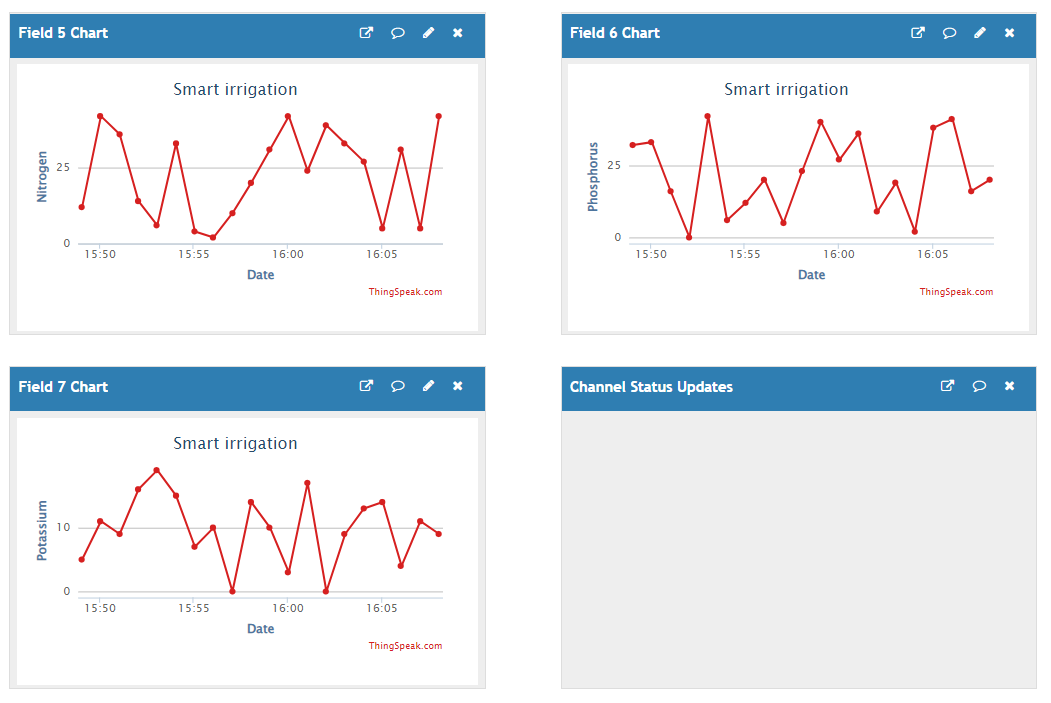
**Figure 4. Output when rain and soil moisture sensor value is low**

****

**Figure 5. Output for other conditions**

****

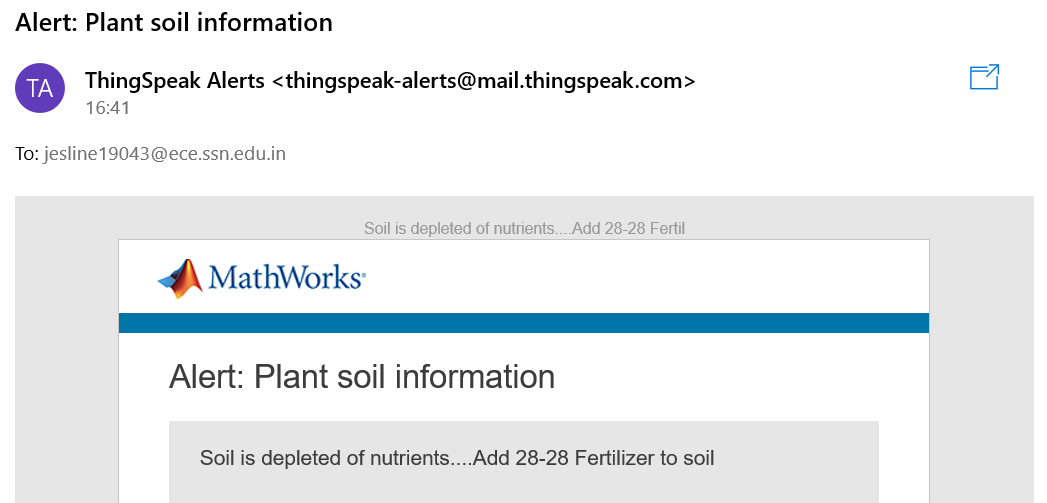
****

****

**Figure 6. ThingSpeak cloud**

****

**Figure 7. ML analysis output**

****

**Figure 8. Email alert**

**Result:**

The deployed model uses the advantages of the ThingSpeak cloud and accurately activates the irrigation system. The soil moisture level, raining pattern ,temperature and humidity is measured and using cloud analysis the level is maintained at an optimum range. An email alert is also sent to the registered user. As a future scope this system can be made as an intelligent system, where in the system predicts user actions, rainfall pattern, time to harvest so that agricultural system can be made independent of human operation .