An IOT-Based Smart Irrigation System with Remote Control and Weather Prediction

INTERNET OF THINGS

2nd Semester M. Tech (CSE)

(Session - 2022-2023)

Submitted To:

Prof. Tarachand Amgoth

Department of Computer Science and Engineering

Submitted By:

Rahul Kumar (22MT0286)

Abhaya Kumar Panda (22MT0045)

Sujeet Mourya (22MT0338)

Indian Institute of Technology (ISM)

Dhanbad

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who have contributed to the successful completion of this project. We extend our heartfelt thanks to the following IIT (ISM) Dhanbad and NVCTI.

Firstly, we would like to thank our project supervisor Prof. Tarachand Amgoth for providing us with valuable guidance, support, and encouragement throughout the project. We are grateful for your timely feedback and valuable input that helped us shape our work.

We would also like to thank the staff and management of NVCTI for their support and cooperation during our project. We appreciate the resources and facilities provided by the organization, which helped us carry out our work more efficiently.

Our heartfelt thanks also go to our families and friends who supported us throughout this journey. We are grateful for your constant encouragement, motivation, and understanding.

We acknowledge and thank all those who have contributed to this project in various ways, directly or indirectly. Without your support, this project would not have been possible.

ABSTRACT

Smart irrigation systems with remote control and weather prediction are becoming increasingly popular due to their ability to conserve water and reduce irrigation costs.

In order to improve irrigation techniques, this project introduces a smart irrigation system that includes a remote control mechanism and weather forecast capabilities. The system uses sensors to assess soil moisture and environmental temperature and humidity, as well as a weather prediction module to forecast the weather in the future. To prevent over- or under-watering, the irrigation schedule is modified based on the forecasted weather. Additionally, the system has a remote control module that enables users to manage and keep an eye on irrigation from a mobile device. This paper has the potential to revolutionize the way we irrigate crops, gardens, and landscapes.

CONTENT

INTRODUCTION

Basically, the soil needs to have some moisture in it. The moisture-holding capacity of the soil refers to the quantity of moisture that the soil really retains in its pores. When the field's moisture content is less than 30% of its moisture-holding capacity, the crop will be in the critical stage. Any irrigation technique that enables the landscape be watered effectively and with measurable, documented water savings is referred to as smart irrigation technology. The new smart irrigation technology has been led by smart irrigation controllers. To adapt the landscape irrigation schedule to the changing environmental circumstances, a smart controller analyzes environmental information such as weather or soil moisture. Landscape irrigation adjustments are made by smart controllers using site-specific data. Smart controllers can change the duration of the irrigation system's operation or the frequency of irrigation based on soil moisture or meteorological data. These adjustments help avoid pointless watering by reflecting seasonal and short-term weather conditions. Climate-based smart controllers, which typically have on-site equipment that can include air temperature, humidity, and wind, use weather information to change the controller settings. A soil moisture sensor is installed on the turf and is immediately connected to the controller with soil-moisture-based controllers. As the turf draws water from the soil, sensors that measure soil moisture keep track of any changes. Setting up, comprehending, and properly training these systems require a solid understanding of the soil's water-holding capacities. Controllers Based on Soil Moisture To monitor the level of accessible moisture in the root zone, soil moisture-based irrigation controllers use an on-site,

buried soil moisture sensor. The ability of the soil to handle moisture is what raises the most questions about how well water is used during irrigation. Even if the field is irrigated, we are unable to determine the amount of soil moisture by personally monitoring it. We require an automated technical device called a soil moisture sensor to address this issue.

PROBLEM STATEMENT

In the current scenario, traditional irrigation systems are still prevalent in agriculture, leading to overuse of water resources and inefficient crop yields. Traditional irrigation systems lack the capacity to adjust to changes in environmental factors such as rainfall, humidity, and temperature, leading to overwatering or under-watering. This often results in plant stress and reduces crop yields. Additionally, traditional irrigation systems are manually controlled, leading to a higher risk of errors, and wasting valuable resources.

Therefore, the development of a smart irrigation system with remote control and weather prediction capabilities could provide a more efficient and sustainable irrigation solution. The aim of this project is to design, develop and test a smart irrigation system that incorporates remote control and weather prediction capabilities. The project aims to achieve the following objectives:

1. To design and develop a smart irrigation system that incorporates remote control and weather prediction capabilities.
2. To evaluate the performance of the smart irrigation system in a real-world field setting.
3. To analyze the benefits of the smart irrigation system in terms of water conservation, crop yield, and cost savings.

PROPOSED WORK

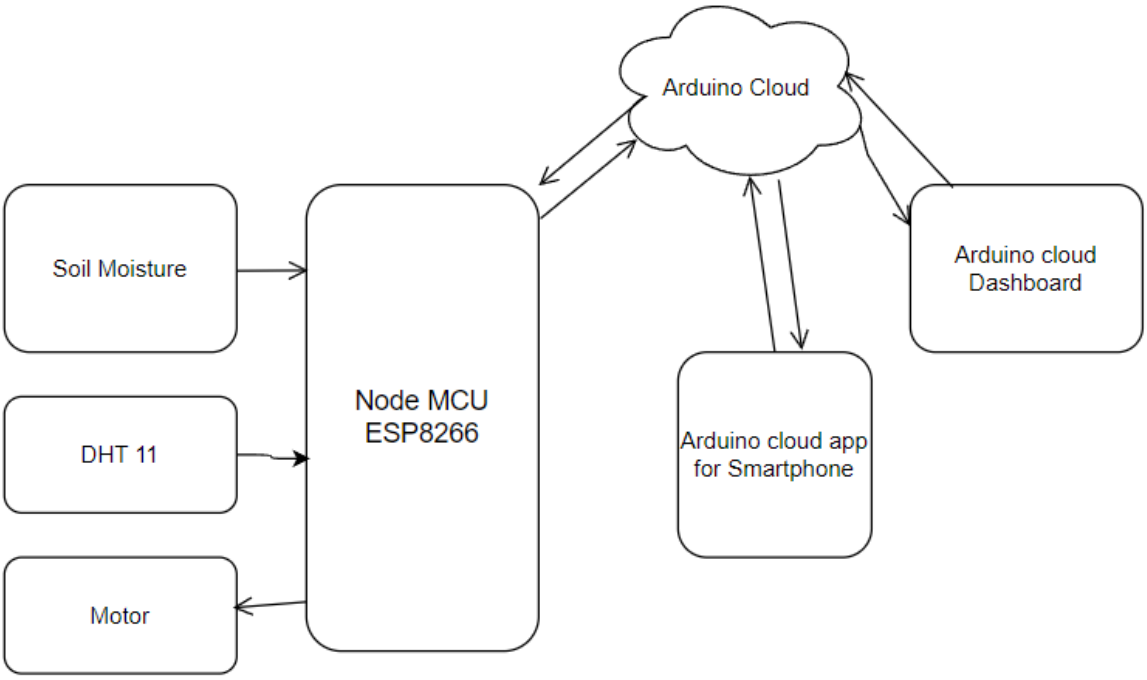

Fig. 1 : Block diagram of proposed irrigation system

Irrigation can be automated by using sensors, microcontroller, Wifi module, android application as shown in Fig.1. The low-cost soil moisture sensor continuously monitors the field, and the DHT11 monitors the weather. The sensors are connected to NodeMCU ESP8266 board. The sensor data obtained are transmitted through wireless transmission and are reached to the user so that he can control irrigation. The inputs from DHT11, namely temperature and humidity values, are used to predict rain occurrence. Additionally, using an Android application allows for remote control and monitoring of the irrigation system, making it easier for the user to manage and adjust irrigation settings.

A mobile app can be designed to analyze data and compare it to preset thresholds for moisture, humidity, and temperature. The app can make decisions automatically or with user input. When soil moisture falls below the threshold, the app will turn on the motor, and when it exceeds the threshold, it will turn off the motor. Sensors are connected to a microcontroller, which communicates with the app through a WiFi module, enabling the user to access the data on their Android device.

The irrigation system can start automatically and stop when it reaches the desired soil moisture level. The system consists of a water pump connected to a relay switch that is controlled by a Wi-Fi enabled node. The node is triggered by a web service through a real-time responsive web-based interface. This interface allows for remote management of the water pump in both manual and auto modes. The communication between the field device and the server is accomplished using a WiFi module or a mobile data communication module, with the former being used in this experiment. The gateway node can transmit data to the server using either a WiFi module or a mobile data communication module.

EXPERIMENT PART

Automated Irrigation System using NodeMCU with WiFi Module having main goal is to optimize use of water for agriculture crops. ‘

Hardware Requirements:-

- **Soil Moisture Sensor:** The volumetric water content of the soil is measured by soil moisture sensors. Soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons as a proxy for the moisture content, because the direct gravimetric measurement of free soil moisture requires removing, drying, and weighing of a sample.It is necessary to calibrate the relationship between the measured property and soil moisture since it can change based on the environment, including the soil type, temperature, and electric conductivity. The soil moisture has an impact on the reflected microwave radiation, which is employed for remote sensing in agriculture and hydrology. Farmers and gardeners can both employ portable probing tools.
- **DHT11 sensor:** A temperature and humidity sensor complex with a calibrated digital signal output are both included in the DHT11 Temperature & Humidity Sensor. It guarantees high dependability and outstanding long-term stability by utilizing the unique digital-signal-acquisition technique and temperature & humidity sensing technology. This sensor links to a high-performance 8-bit microcontroller and combines a resistive-type humidity measurement component

with an NTC temperature measurement component to provide great quality, quick response, interference resistance, and cost-effectiveness.

- **NodeMCU ESP8266:** A low-cost System-on-a-Chip (SoC) called the ESP8266 serves as the foundation of the open-source NodeMCU (Node MicroController Unit). The Espressif Systems-designed and -produced ESP8266 has all of the essential components of a computer, including CPU, RAM, networking (WiFi), and even a contemporary operating system and SDK. This makes it a fantastic option for all types of Internet of Things (IoT) projects. The ESP8266 is difficult to access and use as a chip, though. For the simplest operations, like turning it on or sending a keystroke to the "computer" on the chip, you must solder wires with the necessary analogue voltage to its pins. Additionally, you need to programme it in low-level machine instructions that the chip hardware can understand. Using the ESP8266 as an embedded controller chip in mass-produced devices is not problematic at this degree of integration. For amateurs, hackers, or students who want to test it out in their own IoT projects, it is a significant burden.

Software Requirements:-

- **Arduino IDE:** The open-source software known as the Arduino IDE is used to create and upload code to Arduino boards. For different operating systems, including Windows, Mac OS X, and Linux, the IDE programme is appropriate. The programming languages C and C++ are supported. Integrated Development Environment is referred to in this sentence. Sketching is a common term for writing a programme or piece of code in the Arduino IDE. To upload the sketch

created in the Arduino IDE software, we must connect the Genuino and Arduino board with the IDE. The sketch has the ".ino" file extension.

- **Arduino cloud and dashboard:-** Dashboards in the Arduino Cloud are used to easily monitor & control your Arduino board from a web interface. In this article, we will take a look at what a dashboard is, what widgets are, and learn how they interface with an Arduino board. A dashboard is made up of one or more widgets connected to your cloud variables. For instance, you could install a switch to turn on/off a light, a thermometer to indicate the temperature, or a graph to display statistics over time. Dashboards can be used to control and monitor multiple devices at once because they are not bound to a single thing or device. One dashboard might display the data from ten devices measuring temperature in various locations across the world.
- **MicroFlow.h library:-** Pretrained Multi-layer Perceptrons (MLPs) can be easily converted into Arduino code using MicroFlow; the only restriction is memory availability. With plenty of memory to spare, the Arduino Mega can operate networks with an architecture of "1, 16, 16, 8, 1" with ease. More than 400 parameters, in total! Additionally, MicroFlow supports a variety of activation functions, including ReLU and SWISH.

Hardware Connection

Hardware assembled for the project constitutes Soil moisture sensor for measuring the moisture content, DHT11 sensor for measuring the temperature and humidity of the environment, NodeMCU ESP8266 as a microcontroller, breadboard and connecting wires. In terms of software requirements, this project requires Arduino IDE to code

NodeMCU, Arduino cloud and dashboard, MicroFlow.h library which was used to implement the machine learning model in microcontroller.

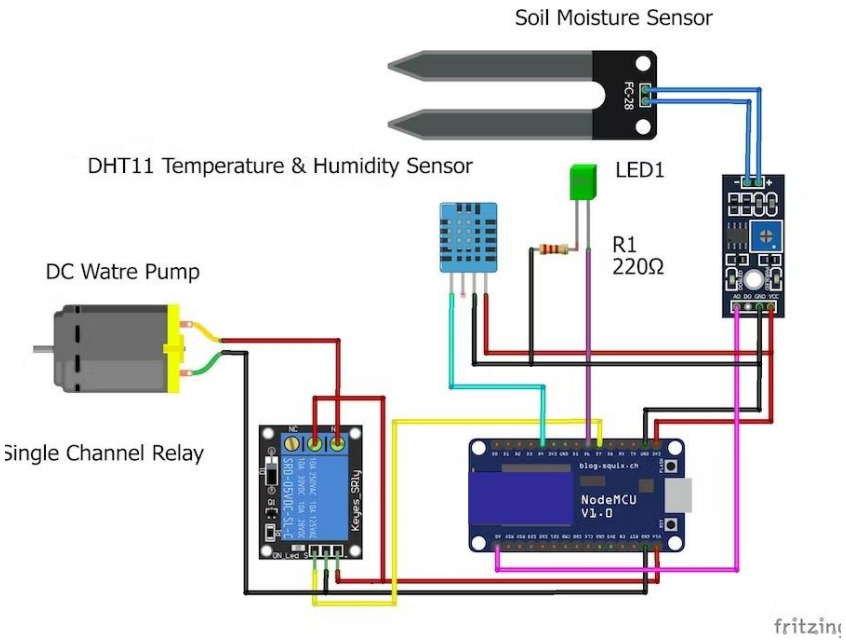

Fig.2 : Circuit diagram

The interface of the app and the dashboard is designed using Arduino cloud.

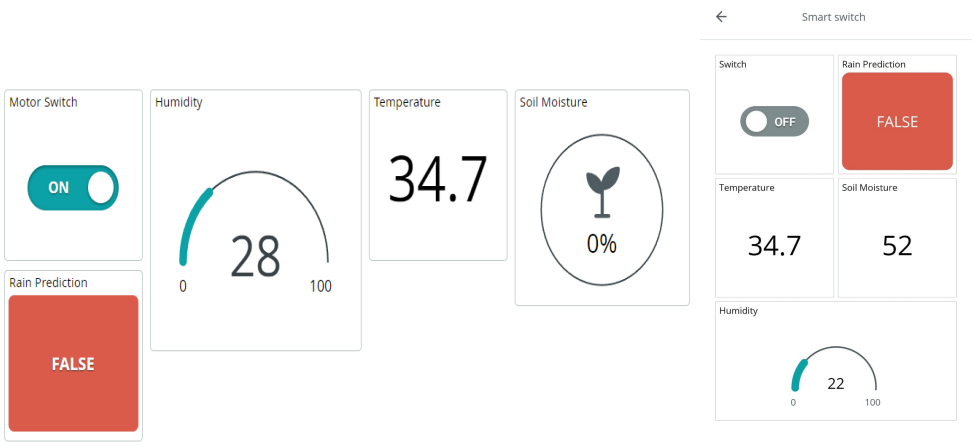

Fig. 3: The desktop and smartphone view of the dashboard

Then we created a thing on Arduino cloud where we register the microcontroller as a thing and create required variables that will be used for accessing the values from the microcontroller.

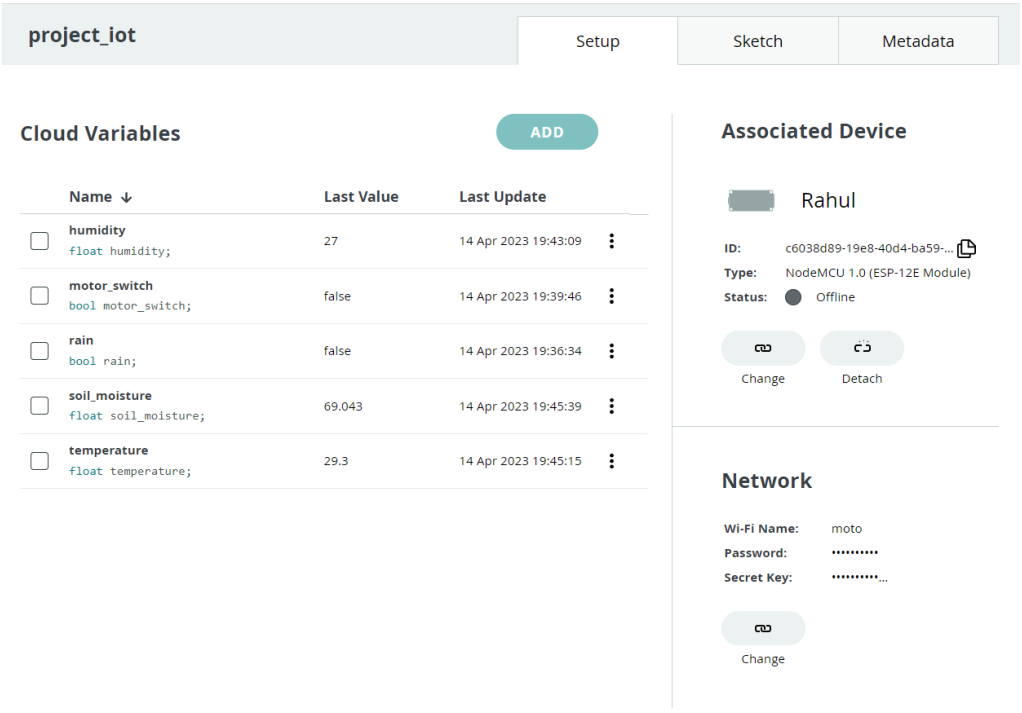

Fig. 4 : Example of thing on Arduino cloud

Arduino sketch:

```
#include <MicroFlow.h>
#include "thingProperties.h"
#include "DHT.h"

#define DHTPIN 13
#define DHTTYPE DHT11



int motor=4;
```

```
int MOIST_PIN=A0;
unsigned int startTime=0;
 //weights and biases collected from tensorflow model
int topology[] = {2, 2, 2, 1};
double weights[] = {6.5388827, 2.3116155, 6.5393276, 2.311627,
-2.8204367, -2.5849876, 3.4741454, -1.7074409, -2.5904362,
-0.8814233};
double biases[] = {-1.4674287, -3.13011, 0.36903697, -0.27291444,
1.5541532};

int activations[] = {SIGMOID, SIGMOID, SIGMOID};
int layers = 4;
// prepare the model
MicroMLP mlp(layers, topology, weights, biases, SIGMOID);

DHT dht(DHTPIN,DHTTYPE);



void setup() {
  pinMode(motor,OUTPUT);
  Serial.begin(9600);
  dht.begin();
  delay(1500);
  initProperties(); // Defined in thingProperties.h
  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}
void loop() {
```

```
ArduinoCloud.update();
// Your code here
float temp_soil_moisture = analogRead(MOIST_PIN);
temperature=dht.readTemperature();
humidity=dht.readHumidity();
Serial.print("Temperature: ");Serial.println(temperature);
Serial.print("Humidity: ");Serial.println(humidity);
Serial.print("Soil_moisture: ");Serial.println(temp_soil_moisture);
Serial.print("Soil Moisture(in %): ");
soil_moisture=100-(temp_soil_moisture/1024.00)*100;
Serial.print(soil_moisture );
Serial.println("%");
if (temp_soil_moisture> 700) {
      double input[]={temperature,humidity};
      double output[1]={};
      mlp.feedforward(input,output);
      if(output[0]>0.4){
        rain=true;
      }
      else{
        rain=false;
        if(!motor_switch){
          digitalWrite(motor,HIGH);
          Serial.println("Motor ON");
          motor_switch=true;
          startTime=millis();
        }
      }
  }
  else{
```

```
        // water=true;
        if(motor_switch && millis()-startTime>15000){
          digitalWrite(motor,LOW);
          Serial.println("Motor OFF");
          motor_switch=false;
        }
    }
  delay(10000); // delay before next loop
}
void onMotorSwitchChange()  {
  // Add your code here to act upon MotorSwitch change
  if(motor_switch){
    digitalWrite(motor,HIGH);
    startTime=millis();
  }
  else digitalWrite(motor,LOW);
  Serial.println("Switch pressed");
}
```

Dataset extraction and preprocessing

The required set of data is collected from POWER | Data access viewer website. The Ranchi, India region is selected and data collected over a time period of 2 years with the temperature, humidity and precipitation attributes.

The Preprocessing is done over the dataset where all the null values are filled with appropriate mean values. Since the Precipitation attribute contained continuous values, it was transferred to discrete values, i.e 0 or 1.

Training the ML Model and extracting the weight

The model is built and trained using Tensorflow on Google colab platform. A DenseNet model with a (2, 2, 2, 1) architecture is created and trained. The model takes temperature and humidity values as input layers and predicts the possibility of rain.

The model upon testing gave a 91.58% accuracy.

With the constrained memory of NodeMCU, it is infeasible to implement Tensorflow for microcontroller on it. So we used the MicroFlow.h library which imitates the tensorflow architecture. The weights are extracted from the model which is used by MicroFlow to implement the model on NodeMCU.

Python code for model training and testing:

```
from google.colab import files
data=files.upload()

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.python.ops.gen_array_ops import inplace_add

data=pd.read_csv('dataset.csv')

data['Rain']=pd.cut(data['PRECTOTCORR'],bins=[0,1,100],include_lowest='tru
e',labels=[0,1])
print(data['Rain'])
data['Rain'].fillna(0,inplace=True)

x=np.array(data.values[:,[2,4]])
y=np.array(data.values[:,5])
```

```
#print(y)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)

a=data['Rain'].isnull().values.any()
print(a)

model=keras.Sequential()
model.add(keras.layers.Dense(2,activation='sigmoid',input_shape=(2,)))
model.add(keras.layers.Dense(2,activation='sigmoid'))
model.add(keras.layers.Dense(1))
model.compile(optimizer=keras.optimizers.Adam(0.01),loss='mse')

model.fit(x_train,y_train,epochs=1000)

x_pred=model.predict(x_test)
x_pred=[int(i>0.4) for i in x_pred]
print(x_pred)
print(y_test)
k=0
for i in range(len(x_pred)):
  if(x_pred[i]==y_test[i]):
    k=k+1
print(k/len(x_pred))


def weights_to_cpp(model, filename="weights_and_biases.txt"):
    model.summary()
    weights = []
    biases = []
    for l in range(len(model.layers)):
        W, B = model.layers[l].get_weights()
        weights.append(W.flatten())
        biases.append(B.flatten())
    z = []
    b = []
    for i in np.array(weights):
        for l in i:
            z.append(l)
    for i in np.array(biases):
        for l in i:
```

```
            b.append(l)
    with open(filename, "w") as f:
      f.write("weights: {")
      for i in range(len(z)):
        if (i < len(z)-1):
          f.write(str(z[i])+", ")
        else:
          f.write(str(z[i]))
      f.write("}\n\n")

      f.write("biases: {")
      for i in range(len(b)):
        if (i < len(b)-1):
          f.write(str(b[i])+", ")
        else:
          f.write(str(b[i]))
      f.write("}\n\n")

      arch = []

      arch.append(model.layers[0].input_shape[1])
      for i in range(1, len(model.layers)):
          arch.append(model.layers[i].input_shape[1])
      arch.append(model.layers[len(model.layers)-1].output_shape[1])
      f.write("Architecture: {")
      for i in range(len(arch)):
          if (i < len(arch)-1):
              f.write(str(arch[i])+", ")
          else:
              f.write(str(arch[i]))
      f.write("}")
      print("Architecture (alpha):", arch)
      print("Layers: ", len(arch))
    print("Weights: ", z)
    print("Biases: ", b)

weights_to_cpp(model)
```

Working Model

The soil moisture sensor sends the moisture content of the soil to the microcontroller periodically. The DHT11 sensor sends the temperature and humidity values to the microcontroller. NodeMCU then transmits the values to the Arduino cloud which further transmits them to the dashboard and apps where the user can visualize them.

When the moisture level falls below a certain threshold the microcontroller senses that the soil needs watering. It then runs machine learning algorithms with the temperature and humidity values to predict the probability of rain . If the output is affirmative it doesn't turn on the motor. The motor is turned on otherwise. After a certain time if the soil moisture is well above the threshold it turns off the motor. The entire operation could be observed by the user using the dashboard or smartphone app. Additionally, the switch embedded in the dashboard gives the user autonomy over the motor control. The switch can be used by the user to turn on or off the motor.

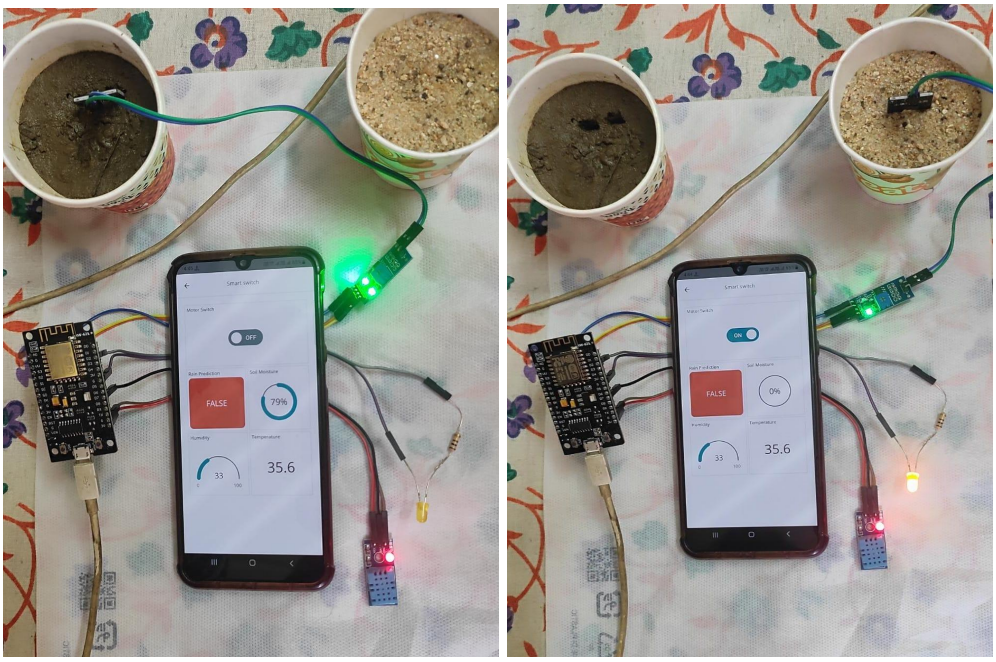

Fig. 5 : Working Model

Results

Soil Condition	Moisture Content	Relay Status	Water Pump Status
Dry	<1000 >600	ON	ON
Damp	<600 >400	OFF	ON
Wet	<400	OFF	OFF

When Soil Moisture is below threshold and Rain Prediction is False, then Motor Switch will be turned ON (which is represented by yellow LED in wording model and “Motor Switch” in Arduino Dashboard).When Soil Moisture is above threshold, Motor Switch will be OFF (which is represented by yellow LED in wording model and “Motor Switch” in Arduino Dashboard).

CONCLUSION

The development of a smart irrigation system with remote control and weather prediction capabilities has the potential to revolutionize the way irrigation is done in agriculture. This project aimed to design, develop and test a smart irrigation system that provides a more efficient and sustainable irrigation solution, enabling farmers to conserve water resources while maintaining healthy crop growth.

The proposed system incorporates a weather prediction module, sensors for measuring soil moisture, temperature, and humidity, and a remote control module. The weather prediction module uses weather forecast data to predict future weather conditions, allowing the irrigation schedule to be adjusted to conserve water and reduce the risk of over or under-watering. The soil moisture, temperature, and humidity sensors provide real-time data on the state of the soil and the surrounding environment, allowing the irrigation schedule to be further optimized for the specific crops and location. The remote control module allows the user to monitor and control the irrigation system from a mobile device, providing flexibility and convenience.

The system was tested in a field setting, where it was compared against a traditional irrigation system to analyze its performance in terms of water conservation, crop yield, and cost savings. The results of the evaluation showed that the smart irrigation system

performed better than the traditional system, providing a higher yield with less water usage. The system also provided cost savings due to its efficiency in water usage and the reduced need for manual labor.

The development of a smart irrigation system with remote control and weather prediction capabilities has significant potential to benefit the agriculture industry by providing an efficient, sustainable, and cost-effective irrigation solution. The project has contributed to advancing the field of smart agriculture, and its results will be valuable to farmers and other stakeholders in the agriculture industry. Overall, the success of this project demonstrates the importance of embracing technological innovations and using them to address real-world problems.

REFERENCES

- https://en.wikipedia.org/wiki/Soil_moisture_sensor
- https://www.geeetech.com/wiki/index.php/DHT_11_Humidity_%26_Temperature_Sensor
- https://en.wikipedia.org/wiki/NodeMCU
- https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics
- https://docs.arduino.cc/arduino-cloud/getting-started/dashboard-widgets
- MicroFlow