APPLICATION DEVELOPMENT USING ANDROID STUDIO / WEB INTERFACE FOR AGRICULTURAL AUTOMATION

Submitted in fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
In
ELECTRONICS & COMMUNICATION ENGINEERING

By

Jotiprova Pal (16500319037)
Sandhikshyan Roy (16500319028)
Ayan Roy (16500319043)
Rageshri Sen (16500319024)
Deepsikha Gayen (16500319040)

Under the supervision of Prof. **Jhimlee Adhikary**

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CALCUTTA INSTITUTE OF ENGINEERING AND MANAGEMENT

24/1A Chandi Ghosh Road, Kolkata-700040

DECLARATION

This is our own work and we have not indulged in any form of plagiarism throughout the dissertation and also while writing this thesis. The work or ideas of other authors which are utilised in this report has been properly acknowledged and mentioned in the references.

Jotiprova Pal (16500319037)
Sandhikshyan Roy (16500319028)
Ayan Roy (16500319043)
Rageshri Sen (16500319024)
Deepsikha Gayen (16500319040)

CERTIFICATE

This is to certify that the project titled **Application Development using Android Studio/Web interface for Agricultural Automation** submitted by in the fulfilment of the requirements for the award of the degree of Bachelor of Technology, is a record of an original research work carried out by **Jotiprova Pal, Sandhikshyan Roy, Ayan Roy, Rageshri Sen, Deepsikha Gayen** under my supervision and guidance during the academic year 2022-2023. The results embodied in this thesis have not been submitted to any other University or Institute for any degree.

Jhimlee Adhikary
Assistant Professor
DEPARTMENT OF ELECTRONICS
AND COMMUNICATION ENGINEERING
CALCUTTA INSTITUTE OF ENGINEERING AND MANAGEMENT

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our supervisor Prof. Jhimlee Adhikary, for guiding us throughout this course and without whom, we would not be even able to start working in this area of research or shape our thesis. Her invaluable guidance led the project to its conclusion. Her patience and encouragement inspired us a lot.

We are thankful to the college authority, our respected Principal Prof. Dr. Sanjib Sil for providing the facilities for carrying out the project work.

Lastly, we thank our parents and friends for their constant encouragement, without them this project would not have been possible.

INDEX

ABSTRACT

Water is a scarce natural resource and a hugely important requirement in the agricultural sector. India first joined the league of water-deficient countries in 2011 with a per capita water availability of 1,428 kilolitres per year. Technical innovations like micro-irrigation play an essential role in efficient water management. Our team has developed an automated drip irrigation system that not only manages to efficiently use water but also automate the process depending on weather conditions and soil moisture levels without the interference of a person making the process very hassle free. In this project we’ve used a ESP8266 NodeMCU module that is connected to a soil moisture sensor and is Wi-Fi enabled which collects weather data from OpenWeatherMap, a hyperlocal precipitation forecast provider. A submersible motor is connected at the end to provide said water depending on both weather data and soil data.

INTRODUCTION

1.1 General

An embedded system is a microprocessor-based computer hardware system with software that is designed to perform a dedicated function, either as an independent system or as a part of a large system. At the core is an integrated circuit designed to carry out computation for real-time operations. Embedded system applications range from digital watches and microwaves to hybrid vehicles and avionics. As much as 98 percent of all microprocessors manufactured are used in embedded systems. Embedded system is otherwise called implanted PC framework, it is an uncommon type of an overall PC. Embedded systems can also be defined as a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today. So basically any Embedded system is a group/integrated circuit of other devices primarily containing a microcontroller/microprocessor, I/O ports, memory, communication port, oscillator etc which performs or can be programmed to do any electronic/electromechanical task. The NodeMCU (Node MicroController Unit) is such an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. The ESP8266, designed and manufactured by Espressif Systems, contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds. ESP8266 is also needed to be programmed in low-level machine instructions that can be interpreted by the chip hardware to work.

But, what about Arduino? The Arduino project created an open-source hardware design and software SDK for their versatile IoT controller. Similar to NodeMCU, the Arduino hardware is a microcontroller board with a USB connector, LED lights, and standard data pins. It also defines standard interfaces to interact with sensors or other boards. But unlike NodeMCU, the Arduino board can have different types of CPU chips (typically an ARM or Intel x86 chip) with memory chips, and a variety of programming environments. There is an Arduino reference design for the ESP8266 chip as well. However, the flexibility of Arduino also means significant variations across different vendors. For example, most Arduino boards do not have Wi-Fi capabilities, and some even have a serial data port instead of a USB port. Hence the NodeMCU was chosen, given it’s relatively low price and built in Wi-Fi capabilities.

1.2: Objective

In this context, we attempted to make a low-cost, efficient automated irrigation system using ESP8226. This would work without any user input and fully depend on data collected from weather forecast and soil moisture levels.

1.3 Outline of Methodology

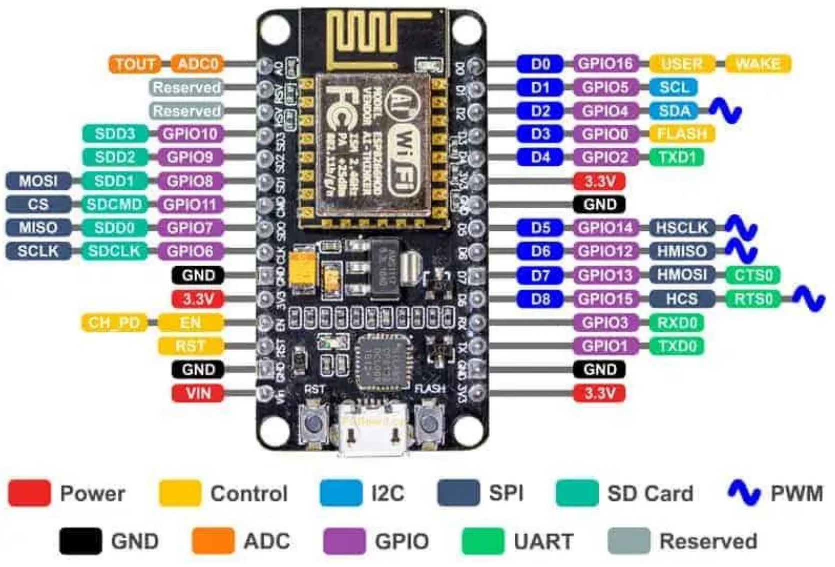
To make this prototype, we used a NodeMCU ESP8226 along with Arduino IDE and used WeatherAPI that can gather weather forecast data through an Application Programming Interface(API). This in turn is connected to a soil moisture sensor and a submersible pump. In the event of expected rainfall, it will postpone watering the crops leading to lower water usage, as well as eliminating human intervention altogether.

COMPONENTS USED

1.NodeMCU - ESP8266 Wifi Module

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). Strictly speaking, the term "NodeMCU" refers to the firmware rather than the associated development kits.

The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as Lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.



Power Pins : There are four power pins. VIN pin and three 3.3V pins.

- VIN can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on VIN is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the VIN pin.
- 3.3V pins are the output of the onboard voltage regulator and can be used to supply power to external components.

Ground: GND is the ground pin of NodeMCU.

I2C Pins: are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

GPIO Pins: NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

ADC Channel: The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins: NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

SPI Pins: NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO

SDIO Pins: NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

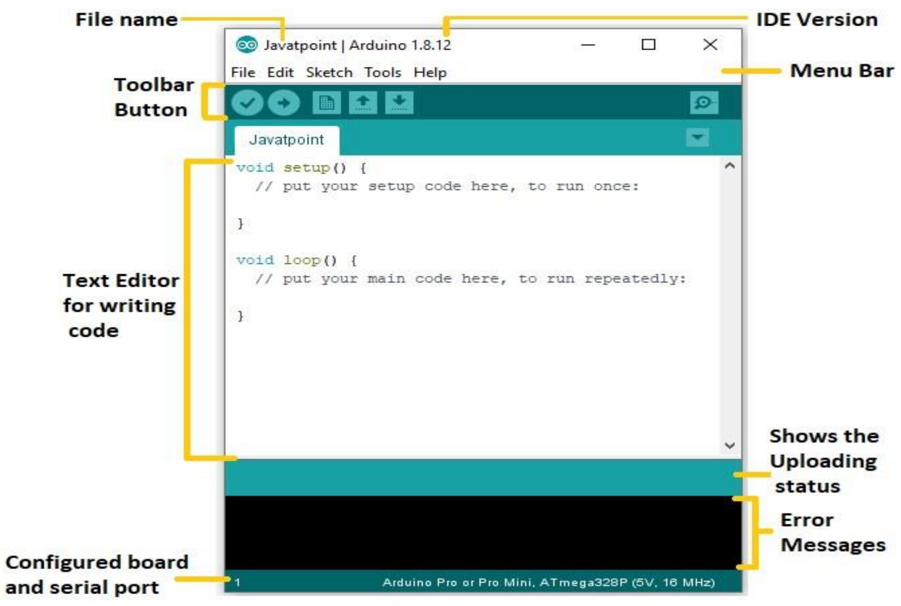
PWM Pins: The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz).

Control Pins: are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

- EN: The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
- RST: RST pin is used to reset the ESP8266 chip.
- WAKE: Wake pin is used to wake the chip from deep-sleep.

2. Arduino IDE

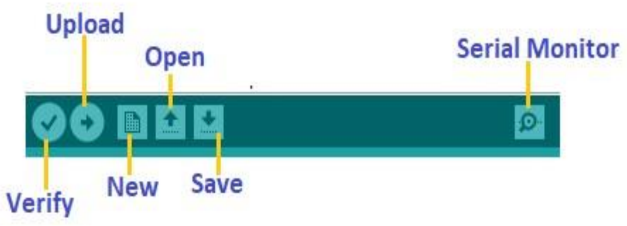
The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**. The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'


File name
IDE Version
Javatpoint | Arduino 1.8.12
File Edit Sketch Tools Help
Menu Bar
Toolbar Button
Javatpoint
Text Editor for writing code
Shows the Uploading status
Error Messages
Configured board and serial port
1
Arduino Pro or Pro Mini, ATmega328P (5V, 16 MHz)

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Toolbar Button

The icons displayed on the toolbar are **New, Open, Save, Upload,** and **Verify**. It is shown below:



Upload

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, we need to make sure that the correct board and ports are selected.

We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the Upload button present on the toolbar.

The latest Arduino boards can be reset automatically before beginning with Upload. In the older boards, we need to press the Reset button present on it. As soon as the uploading is done successfully, we can notice the blink of the Tx and Rx LED. If the uploading is failed, it will display the message in the error window. We do not require any additional hardware to upload our sketch using the Arduino Bootloader. A **Bootloader** is defined as a small program, which is loaded in the microcontroller present on the board. The LED will blink on PIN 13.

Open

The Open button is used to open the already created file. The selected file will be opened in the current window.

Save

The save button is used to save the current sketch or code.

New

It is used to create a new sketch or opens a new window.

Verify

The Verify button is used to check the compilation error of the sketch or the written code.

Serial Monitor

The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor.

3.Submersible Mini Water Pump - 3-6V DC

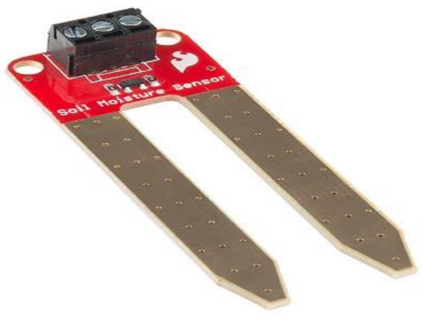
In our project, a Submersible water pump has been used, namely DC 3-6 V Mini Micro Submersible Water Pump. The submersible water pumps are built to push water without large particles or contaminants. They can be used to move water from one point to another, and they offer an efficient choice for drain systems, pools, utilities, and more.
A mini submersible water pump is a centrifugal water pump, which means that it uses a motor to power an impeller that is designed to rotate and push water outwards.

This DC 3-6 V Mini Micro Submersible Water Pump is a low-cost, small-size Submersible Pump Motor that can be operated from a 2.5 ~ 6V power supply. It can take up to 120 litres per hour with a very low current consumption of 220mA. The tube pipe is just needed to be connected to the motor outlet, and then it has to be submerged in water, and finally power it. It is extremely easy to use and great for building science projects, fire-extinguishers, fire fighting robots, mini fountains, and plant watering systems.

4.Soil Moisture Sensor

The soil moisture sensor is one kind of sensor used to gauge the volumetric content of water within the soil. As the straight gravimetric dimension of soil moisture needs eliminating, drying, as well as sample weighting. These sensors measure the volumetric water content not directly with the help of some other rules of soil like dielectric constant, electrical resistance, otherwise interaction with neutrons, and replacement of the moisture content.

The relation among the calculated property as well as moisture of soil should be adjusted & may change based on ecological factors like temperature, type of soil, otherwise electric conductivity. The microwave emission which is reflected can be influenced by the moisture of soil as well as mainly used in agriculture and remote sensing within hydrology. These sensors normally used to check volumetric water content, and another group of sensors calculates a new property of moisture within soils named water potential. Generally, these sensors are named as soil water potential sensors which include gypsum blocks and tensiometer.



Soil Moisture Sensor Pin Configuration. The FC-28 soil moisture sensor includes 4-pins:

1.VCC pin is used for power

2.A0 pin is an Analog output

3.D0 pin is a digital output

4.GND pin is a Ground

This module also includes a potentiometer that will fix the threshold value, & the value can be evaluated by the comparator-LM393. The LED will turn on/off based on the threshold value.

5. 5V Single-Channel Relay Module

Relay is one kind of electro-mechanical component that functions as a switch. The relay coil is energized by DC so that contact switches can be opened or closed. A single channel 5V relay module generally includes a coil, and two contacts like normally open (NO) and normally closed (NC).

Single-Channel Relay Module Specifications:

- Supply voltage – 3.75V to 6V
- Quiescent current: 2mA
- Current when the relay is active: ~70mA
- Relay maximum contact voltage – 250VAC or 30VDC
- Relay maximum current – 10A

5V Relay Module Pin Configuration

Normally Open (NO): This pin is normally open unless we provide a signal to the relay modules signal pin. So, the common contact pin smashes its link through the NC pin to make a connection through the NO pin.
Common Contact: This pin is used to connect through the load that we desire to switch by using the module.
Normally Closed (NC): This NC pin is connected through the COM pin to form a closed circuit. However, this NC connection will break once the relay is switched through providing an active high/low signal toward the signal pin from a microcontroller.
Signal Pin: The signal pin is mainly used for controlling the relay. This pin works in two cases like active low otherwise active high. So, in active low case, the relay activates once we provide an active low signal toward the signal pin, whereas, in an active high case, the relay will trigger once we provide a high signal toward the signal pin.
5V VCC: This pin needs 5V DC to work. So 5V DC power supply is provided to this pin.
Ground: This pin connects the GND terminal of the power supply.

6. Hi-Waote 9V Battery

Hi-Waote 9V Battery is the most commonly used and portable 9V battery. It is non-rechargeable and is a high capacity and low-cost solution for many electronic devices. It is based on Zinc Carbon Chemistry and can be used easily replaced if discharged just like any standard AA and AAA batteries.

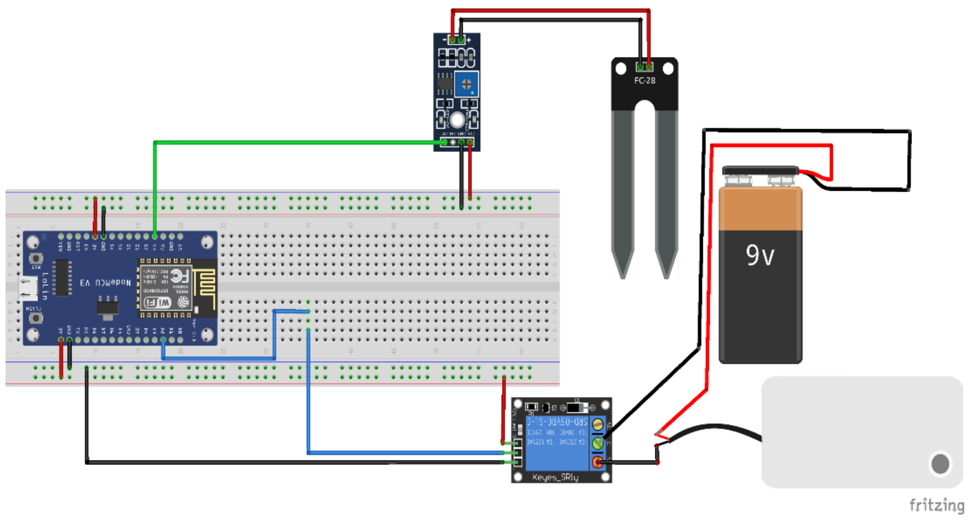
7. Weather API

WeatherAPI.com provides access to free weather and geo data via a JSON/XML restful API. It allows developers to create desktop, web and mobile applications using this data very easy.

It is a powerful fully managed weather and geolocation API provider that provides extensive APIs that range from the realtime and weather forecast, historical weather, Air Quality Data, Bulk Request, IP lookup, and astronomy through to sports, time zone, and geolocation.

Equipped with AI and machine learning this API delivers weather from global weather stations and high resolution weather models to deliver fast, reliable and accurate weather for a given location anywhere in the world. WeatherAPI.com makes it super easy to integrate our realtime, daily, hourly and 15 min interval weather forecast data, historical weather, marine weather, bulk request, air quality data, autocomplete, time zone, astronomy and sports data into your new or existing project.

CIRCUIT DIAGRAM



SOFTWARE, LANGUAGE AND LIBRARY REQUIRED FOR DEVICE AUTOMATION

SOFTWARE USED : Arduino IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

LANGUAGE USED : C++

C++ is a high-level, general-purpose programming language created by Danish computer scientist Bjarne Stroustrup. For our project we have used this language to implement our code. Being standard, C++ is portable and hence easily usable in Arduino IDE which does things in the background like, it #includes .h files when it thinks we need them.

LIBRARIES USED:

- **NTPClient:** To get time from a Network Time Protocol Server and keep it in sync, we need to use NTPClient to connect to a client server.

- **Arduino_JSON:** ArduinoJson is a JSON library for Arduino, IoT, and any embedded C++ project. It supports JSON serialization, JSON deserialization, MessagePack, streams, and fixed memory allocation. It has a simple API, it’s easy to use, and it’s trusted by thousands of clients.

- **Firebase Arduino Client Library:** The library supports Firebase products e.g. Realtime database, Cloud Firestore database, Firebase Storage and Google Cloud Storage, Cloud Functions for Firebase and Cloud Messaging. The library also supported other Arduino devices using Clients interfaces e.g. WiFiClient, EthernetClient, and GSMClient. Google Firebase Arduino Client Library for Espressif ESP8266 and ESP32.

- **ESP8266 Firebase:** A reliable low latency library to read, write, update and delete data from Firebase Realtime Database**.**

- **Firebase JSON:** Able to Parse, create and Edit the simple or complex (depth nested) JSON object as just specify the relative node/ element path. The easiest Arduino library JSON parser, builder and editor for ESP8266, ESP32, Teensy 3.x, Teensy 4.x and others MCUs.

PROGRAM CODE USED TO AUTOMATE THE IRRIGATION SYSTEM

```
#include <Arduino.h>
#if defined(ESP32)
#include <WiFi.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <Arduino_JSON.h>
#include <NTPClient.h>
#include <WiFiUdp.h>

// Provide the token generation process info.
#include "addons/TokenHelper.h"
// Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "name"
#define WIFI_PASSWORD "password"

// Insert Firebase project API Key
#define API_KEY "AIzaSyCDd_2Cnzmzt_zoHCELFNZqJzb7KTNF874"

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://autogator-782e9-default-rtdb.asia-southeast1.firebasedatabase.app/"

// Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

bool signupOK = false;

const long oneSecond = 1000;
const long oneMinute = oneSecond * 60;
const long oneHour = oneMinute * 60;
const int moistureSensorPin = A0;
const int moistureDigital = D2;
int moistureSensorValue = 0;
int moistureDigitalValue = 0;
const int pumpPin = D1;
const int ledPin = D0;
double checkInterval = 30;
int extreme_dryness = 885;
int dryness = 746;
int amountToPump_when_extreme_dryness = 4000;
int amountToPump_when_dryness = 2000;

const long utcOffsetInSeconds = 19800;
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOffsetInSeconds);

String openWeatherMapApiKey = "9b12704fd04d449084e102955232005";
String city = "Kolkata";
unsigned long lastTime = 0;
unsigned long timerDelay = 1000;
```

```
String jsonBuffer;

void setup()
{
    Serial.begin(9600);
    pinMode(ledPin, OUTPUT);
    pinMode(moistureSensorPin, INPUT);
    pinMode(moistureDigital, INPUT);
    pinMode(pumpPin, OUTPUT);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to Wi-Fi");
    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(300);
    }
    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();
    for (int i = 0; i <= 14; i++) // LED will blink fast to indicate that Nodemcu is connected to internet
    {
        digitalWrite(ledPin, HIGH);
        delay(100);
        digitalWrite(ledPin, LOW);
        delay(100);
    }

    /* Assign the api key (required) */
    config.api_key = API_KEY;

    /* Assign the RTDB URL (required) */
    config.database_url = DATABASE_URL;

    /* Sign up */
    if (Firebase.signUp(&config, &auth, "", ""))
    {
        Serial.println("ok");
        signupOK = true;
    }
    else
    {
        Serial.printf("%s\n", config.signer.signupError.message.c_str());
    }

    /* Assign the callback function for the long running token generation task */
    config.token_status_callback = tokenStatusCallback; // see addons/TokenHelper.h

    Firebase.begin(&config, &auth);
    Firebase.reconnectWiFi(true);
    timeClient.begin();
}

void loop()
{
    if ((Firebase.ready() && signupOK && (millis() - lastTime > timerDelay)))
    {
        if (WiFi.status() == WL_CONNECTED)
        {
            timeClient.update();
            Serial.println(timeClient.getFormattedTime());
            int H = timeClient.getHours();
            String serverPath = "http://api.weatherapi.com/v1/forecast.json?key=" + openWeatherMapApiKey + "&q=" + city +
"&day=1&hour=" + H;
```

```
jsonBuffer = httpGETRequest(serverPath.c_str());
JSONVar myObject = JSON.parse(jsonBuffer);
if (JSON.typeof(myObject) == "undefined")
{
  Serial.println("Parsing input failed!");
  return;
}
Serial.print("JSON object = ");
Serial.println(myObject);

Serial.print("Location:"); // Forecasted location
Serial.println(myObject["location"]["name"]);
// Write an Int number on the database path test/int
if (Firebase.RTDB.setString(&fbdo, "datab/city", city))
{
  Serial.println("PASSED");
  Serial.println("PATH: " + fbdo.dataPath());
  Serial.println("TYPE: " + fbdo.dataType());
}
else
{
  Serial.println("FAILED");
  Serial.println("REASON: " + fbdo.errorReason());
}

// Write an Float number on the database path test/float
Serial.print("Time: "); // Forecasted data time
Serial.println(timeClient.getFormattedTime());
if (Firebase.RTDB.setString(&fbdo, "datab/time", timeClient.getFormattedTime()))
{
  Serial.println("PASSED");
  Serial.println("PATH: " + fbdo.dataPath());
  Serial.println("TYPE: " + fbdo.dataType());
}
else
{
  Serial.println("FAILED");
  Serial.println("REASON: " + fbdo.errorReason());
}

double Temperature = myObject["current"]["temp_c"]; // current temperature value
Serial.print("Current Temperature = ");
Serial.println(Temperature);
if (Firebase.RTDB.setDouble(&fbdo, "datab/temp", Temperature))
{
  Serial.println("PASSED");
  Serial.println("PATH: " + fbdo.dataPath());
  Serial.println("TYPE: " + fbdo.dataType());
}
else
{
  Serial.println("FAILED");
  Serial.println("REASON: " + fbdo.errorReason());
}
double Windspeed = myObject["current"]["wind_kph"]; // current windspeed
Serial.print("Current Windspeed = ");
Serial.println(Windspeed);
if (Firebase.RTDB.setDouble(&fbdo, "datab/speed", Windspeed))
{
  Serial.println("PASSED");
  Serial.println("PATH: " + fbdo.dataPath());
  Serial.println("TYPE: " + fbdo.dataType());
}
else
```

```
{
   Serial.println("FAILED");
   Serial.println("REASON: " + fbdo.errorReason());
}

int Will_It_Rain = myObject["forecast"]["forecastday"][0]["hour"][0]["will_it_rain"];
Serial.print("Will It Rain: ");
if (Will_It_Rain == 1)
{
   Serial.println("YES");
   if (Firebase.RTDB.setString(&fbdo, "datab/willrain", "YES"))
   {
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
   }
   else
   {
      Serial.println("FAILED");
      Serial.println("REASON: " + fbdo.errorReason());
   }
}
else
{
   Serial.println("NO");
   if (Firebase.RTDB.setString(&fbdo, "datab/willrain", "NO"))
   {
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
   }
   else
   {
      Serial.println("FAILED");
      Serial.println("REASON: " + fbdo.errorReason());
   }
}
int Chance_Of_Rain = myObject["forecast"]["forecastday"][0]["hour"][0]["chance_of_rain"];
Serial.print("Chance of Rain: ");
Serial.println(Chance_Of_Rain);
if (Firebase.RTDB.setString(&fbdo, "datab/chancerain", "NO"))
   {
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
   }
   else
   {
      Serial.println("FAILED");
      Serial.println("REASON: " + fbdo.errorReason());
   }
moistureSensorValue = analogRead(moistureSensorPin); // read the value of the moisture Sensor
Serial.print("Moisture level sensor value: ");      // print it to the serial monitor
Serial.println(moistureSensorValue);
if (Firebase.RTDB.setInt(&fbdo, "datab/moisture", moistureSensorValue))
{
   Serial.println("PASSED");
   Serial.println("PATH: " + fbdo.dataPath());
   Serial.println("TYPE: " + fbdo.dataType());
}
else
{
   Serial.println("FAILED");
   Serial.println("REASON: " + fbdo.errorReason());
```

```
            }
            if (moistureSensorValue >= extreme_dryness)
            {
                digitalWrite(pumpPin, LOW);
                Serial.println("pump on for 4 seconds");
                delay(50000);
                digitalWrite(pumpPin, HIGH);
                Serial.println("pump off");
                delay(6 * oneSecond); // delay for 6 hours
            }
            else if (moistureSensorValue >= dryness)
            {
                if (Will_It_Rain == 1)
                {
                    delay(15 * oneSecond); // delay for 6 hours
                }
                else
                {
                    digitalWrite(pumpPin, LOW);
                    Serial.println("pump on for 2 seconds");
                    delay(amountToPump_when_dryness);
                    digitalWrite(pumpPin, HIGH);
                    Serial.println("pump off");
                    delay(15 * oneSecond); // delay for 6 hours
                }
            }
        }
        else
        {
            Serial.println("WiFi Disconnected");
        }
        lastTime = millis();
    }
    delay(15 * oneSecond); // delay for 6 hours
}

String httpGETRequest(const char *serverName)
{
    WiFiClient client;
    HTTPClient http;
    http.begin(client, serverName);    // Your IP address with path or Domain name with URL path
    int httpResponseCode = http.GET(); // Send HTTP POST request
    String payload = "{}";
    if (httpResponseCode > 0)
    {
        Serial.print("HTTP Response code: ");
        Serial.println(httpResponseCode);
        payload = http.getString();
    }
    else
    {
        Serial.print("Error code: ");
        Serial.println(httpResponseCode);
    }
    http.end();
    return payload;
}
```

LANGUAGE USED FOR WEB PAGE

Front-end Design: HTML, CSS

Hypertext Markup Language (HTML)

It is the standard markup language for creating web pages and web applications. HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[2] A form of HTML, known as HTML5, is used to display video and audio, primarily using the <**canvas**> element, together with JavaScript.

Cascading Style Sheets (CSS)

It is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. It is designed to enable the separation of presentation and content, including layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

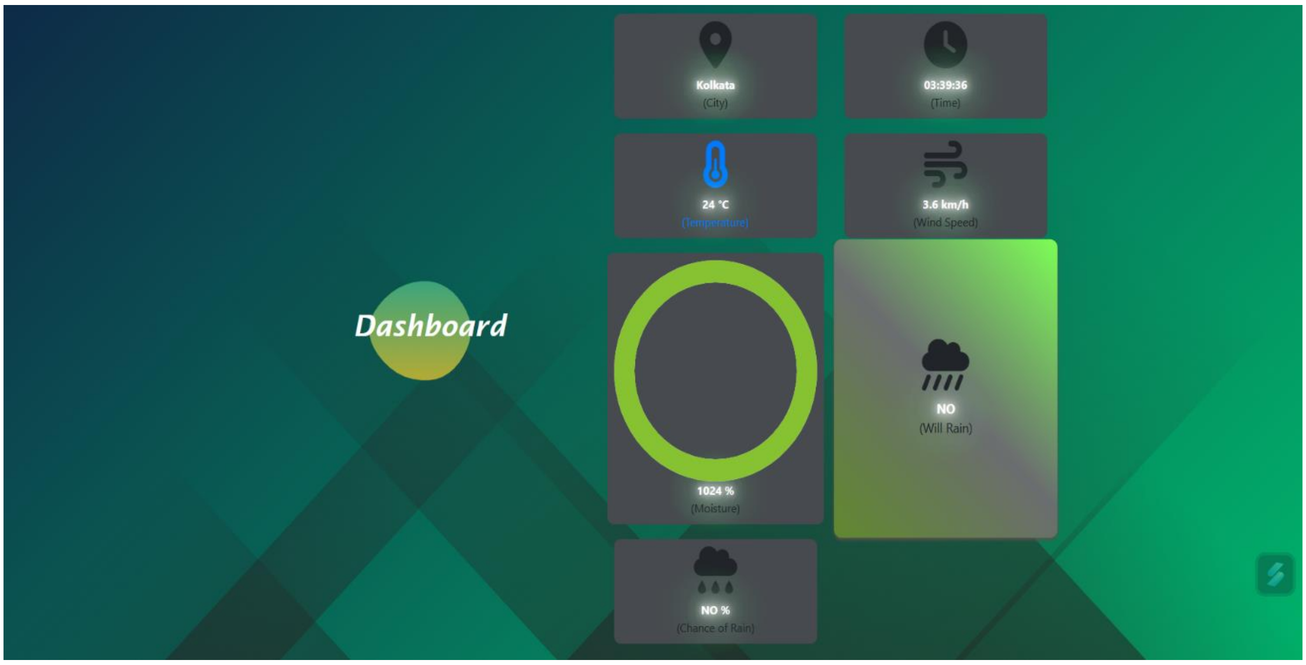
CSS has a simple syntax and uses a number of English keywords to specify the names of various style properties.

A style sheet consists of a list of rules. Each rule or rule-set consists of one or more selectors, and a declaration block.

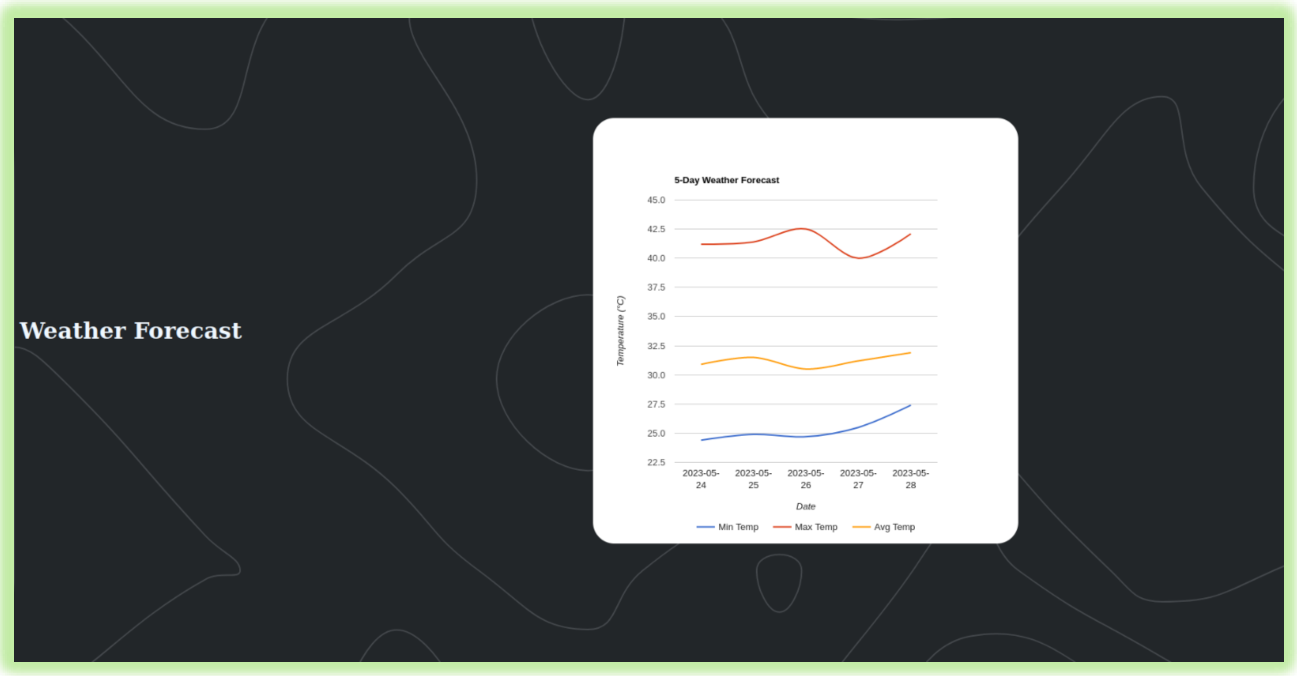
Back-end Design: JavaScript

JavaScript, often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js.

SCREENSHOT OF THE WEBPAGE

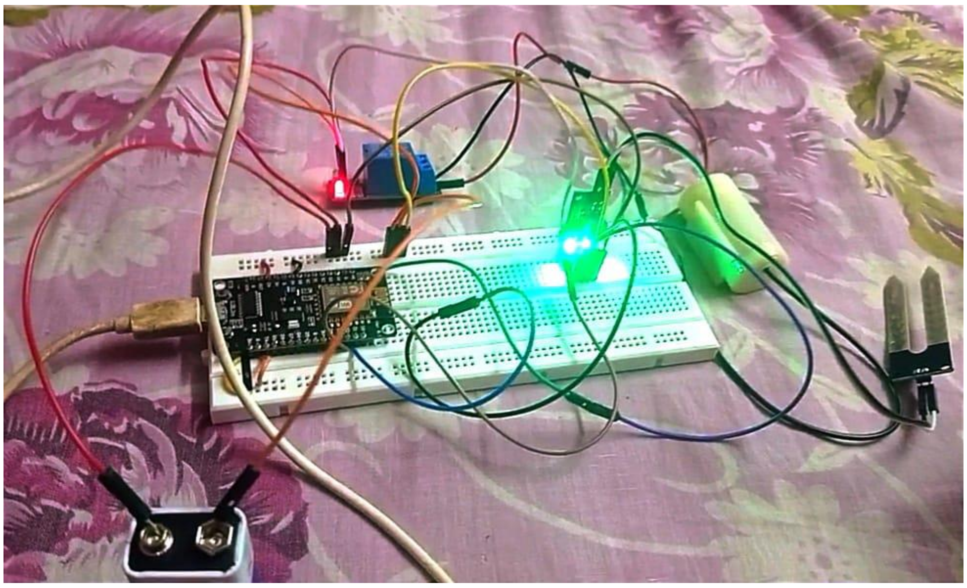


SCREENSHOT OF THE WEATHER DATA

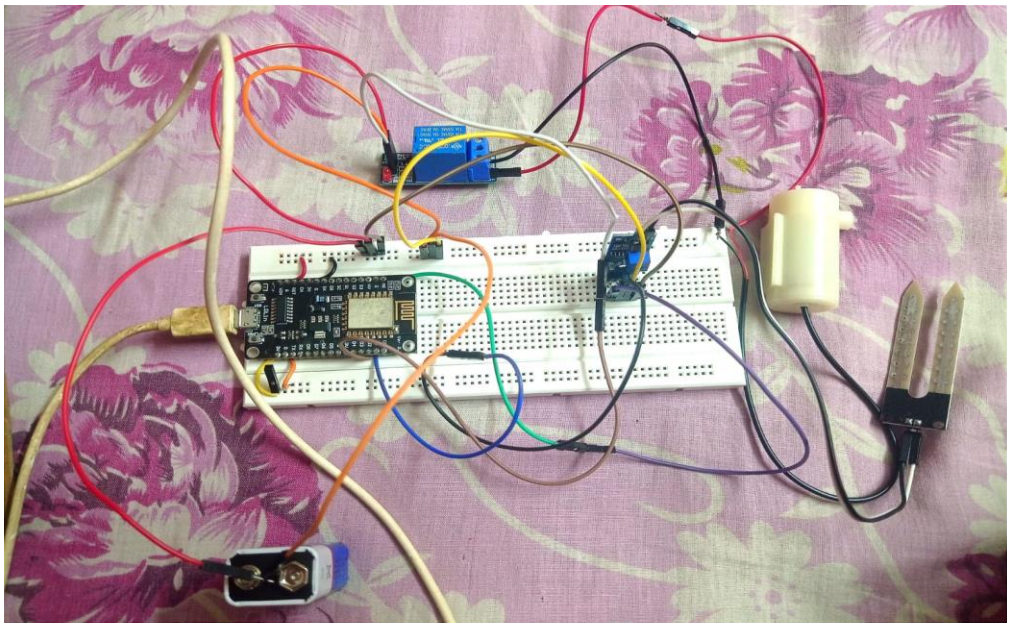


OUTCOME OF THE PROJECT

In ON state:



In OFF state:



CONCLUSION

In this project, we have tried to solve the problem of overuse and wasting of water due to lack of monitoring and human control at times of scarcity of water. Here we are learning the concept of automation in the field of agriculture with the help of a microcontroller in the form of Internet Of Things (IOT). So, for the purpose of implementation of our idea, we have used a soil moisture sensor to detect and measure the moisture level in the soil and we have also used WeatherAPI to collect information regarding the temperature, humidity and probability of rain in a certain area. We have discussed about the different components using NodeMCU we would require to implement our project i.e. we have given the hardware and software requirements in the project. Apart from that, it was a great experience to see the weather API working in such a synchronized way to give the right temperature at the right time resulting in the proper timing for the mini pump to start and stop. All together, this system worked very well as a device to help irrigate without any human intervention. The process which led us to the end of our major project, was really interesting.

BIBLIOGRAPHY

We have taken help from the following websites to complete this project:-

- www.weatherapi.com
- www.youtube.com
- www.hackster.io
- www.javapoint.com
- www.wikipedia.com