Smart Irrigation System with Weather Sensing

Sayali Kudale
Computing and Software System
University of Washington
Bothell, WA, USA
sayalik@uw.edu

ABSTRACT

The importance of the plants in human life does not need to be proven. As plants provides tremendous benefits to our life; However, as plant parent many people failed to understand the needs of plant most importantly watering frequency. Plants give enormous advantages to our lives; nevertheless, many plant parents fail to comprehend the demands of their plants, particularly the frequency with which they should be watered. To address this issue, this project-based research is implemented using the NodeMCU ESP8266 chip-based microcontroller system and Internet of Things (IoT) technology. It is programmed in such a way that it can detect the moisture level of the soil and then automatically supply water to plants only, when necessary, without any human intervention. The main objective of this study is to develop an automated watering system that can control the water supply, optimize water usage by preventing overwatering in case of higher rain probability by weather forecast data. This system may be utilized for both indoor and outdoor plants, allowing individuals to enjoy growing plants without worrying about forgetfulness or vacations. The system has been thoroughly tested to ensure that it will run on its own. All of the sensors are in great working order: the moisture sensors precisely measured the moisture level.

KEYWORDS

Node MCU, Soil Moisture Sensor, automatic watering system, internet of things, Amazon Web Services, MQTT

1 INTRODUCTION

Plants, whether they be outdoor or indoor plants, are helpful to people. The favorable impact of plants to human mental and physical health is one of these advantages. Many individuals keep houseplants as a hobby or for the health advantages they provide. Moreover, Plants often be used to beautify both the interior and outside of our houses, as well as to celebrate special occasions such as weddings and parties. Indoor plants and house gardens, in general, contribute to cleaner, healthier air, which improves our safety and comfort. It also improves the quality of our surroundings. Indoor plants in offices have been recommended by researchers as a way to boost staff productivity [1].

People no longer have the time or energy to physically water their plants every day, from small gardens to large-scale farms; instead, they rely on irrigation systems and professional gardeners. On the other hand, the plant's care for some beginning gardeners might be daunting. It's easy for new gardeners to get confused and make mistakes while watering the plants. The most common mistake is overwatering. Plants growing in too-wet soil suffer from a lack of oxygen, which causes the roots to die and the plant to lose its vitality. Hence it is very important appropriately water the plants to keep them healthy and thriving.

The world's fast rise of information technology has made information more accessible than ever before. Every sector of life now has an increasing number of information technology applications. For monitoring and controlling systems, the internet, software, and hardware devices such as sensors are becoming more prevalent. Researchers began to apply these technologies to solve challenges associated with plant care by inventing an autonomous watering system.

This project's purpose is to find a solution to the aforementioned issues by using Internet of Things and sensors. An automated irrigation system will be developed by monitoring the soil moisture conditions. Weather data is also being used to optimize water consumption and conserve water by minimizing overwatering in the event of greater rain likelihood, as predicted by weather prediction data. People may also enjoy cultivating plants without having to worry about forgetting about them or going on vacation. This method ensures that the plants are never dehydrated by only providing water when it is needed.

The remainder of the paper is organized as follows: In Section 2, we will discuss the previous research and studies conducted for the automatic plant watering system. In Section 3, we will explore the system model problem statement and analysis. In Section 4, we discuss the design and implementation of the project. In Section 5, we will present the results and evaluations of the experiment. Finally, we will conclude in section 6 and discuss the future scope of the project.

2 RELATED WORKS

Researchers have discovered that by utilizing various technologies in the field of agriculture, we may boost productivity while reducing physical labor. Particularly, connecting physical resources on farms to the internet, enables the opportunity to remotely monitor agricultural conditions and infrastructure, freeing up time and effort and allowing individuals to focus on other things. As a result, significant attempts have been made to harness the advantages of IoT in the agricultural. This section discusses many researchers' attempts to automating irrigation in the agriculture industry, as well as indoor/outdoor gardening using sensors and IOT.

In [2], the researchers have developed automatic plant watering system using the Arduino ATmega 328p chip-based microcontroller system and Internet of Things (IoT) technology, but remote monitoring and controlling features were missing in their system. W. Difallah along with other researchers [2], proposed a solar power smart irrigation system using wireless sensor network as an IoT device, in this study they aimed to reduce the water consumption according to soil moisture and environmental conditions sensed by wireless sensor network devices.

A study by, S. Pawar and her colleagues [4], have highlighted importance of water conservation in their study and implemented a system using soil moisture sensor, DHT 11 sensor and raspberry pi to detect the change in moisture, humidity and temperature in the field and automate the irrigation. Along the same line C. Subramani et al [5], have implemented a web based Smart Irrigation system using Arduino Uno and Raspberry Pi. Although this system provides status of field parameters using html-based UI, they haven't used cloud resources and IoT technology.

In [6] S. Rawal, mainly focused on sensing soil moisture using YL-69 soil moisture sensor to prevent over irrigation or under irrigation of soil thereby avoiding crop damage. Also, monitoring was provided by regularly giving updates on a webpage using GSM-GPRS SIM900A modem and the sensor readings were transmitted to a Thing speak channel to generate graphs. Another study by R. Suresh [7], discussed adopting an automated microcontroller-based rain gun irrigation system, in which watering occurs only when there is a high demand for water, hence conserving a considerable amount of water. However, these systems only covered a small portion of agricultural area and were not commercially viable.

In Smart IoT based irrigation system with automated plant recognition [8], researchers developed an irrigation system, with the use of deep learning, which adjusts the amount of water for each type of plant through plants recognition. Even though their system was able to adjust the amount of water according to type of species of plant, it was only limited to 10-20 types. A. Gujar and his colleagues [9] also used deep learning technology to develop an Indoor plant monitoring system which gives timely alerts to the user about changing weather and soil moisture conditions along with chatbot support to understand the health condition of plant.

After researching all the aforementioned watering systems, this research proposes an IoT-based autonomous watering system for household use by using readily available and affordable resources. The majority of existing studies focus on automating the plant watering and providing remote monitoring to the user. However, several systems neglected other environmental factors like rain or relied on physical devices such as a DHT sensor to detect humidity surrounding the plant, potentially leading to overwatering if rain is expected in the future. The goal of this research is to use weather data to optimize water usage and preserve water by preventing overwatering in case of higher rain probability by weather forecast data. Moreover, People may enjoy growing plants without worrying about forgetting about them or heading on vacation. This technology can only provide water when it is required, ensuring that the plants are never dehydrated.

3 SYSTEM MODEL, PROBLEM STATEMENT, AND ANALYSIS

3.1 System Model

The system model consists of three entities, hardware components, a web client, and an IoT cloud resources. Each component communicates with the other via MQTT. A lightweight messaging service mainly uses a publish-subscribe communication pattern, which is mainly used for Machine to machine (M2M) communication and plays a crucial role on the Internet of things (IoT). MQTT is a great choice for wireless networks and requires very low bandwidth and power. Essentially, each entity uses MQTT to communicate with other entities within the project. Figure 1 demonstrates the general outline of this product.

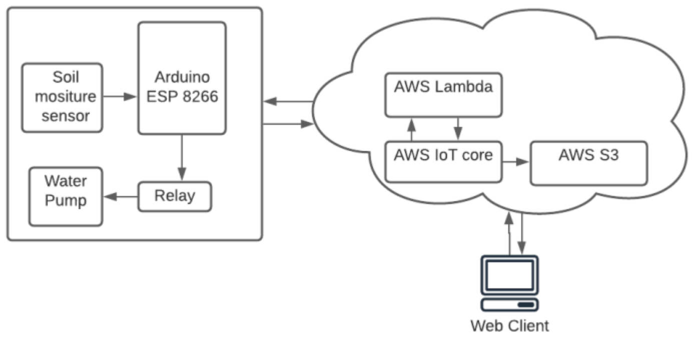


Figure 1: A high level system diagram of project

3.1.1 Hardware Components

i. NodeMCU

NodeMCU is an open-source Lua based firmware and development board specially targeted for IoT based Applications [10]. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The main reason to select this is because it is inexpensive and has an inbuilt Wi-Fi module [11]. It is like Arduino so can be coded using Arduino IDE software. It contains 10 General Purpose Input/output pins for interfacing with devices.

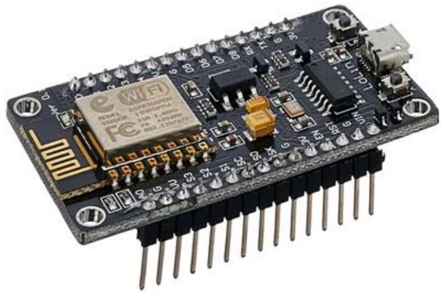


Figure 2: NodeMCU ESP8266 [10]

ii. Capacitive Soil Moisture Sensor

The capacitive soil moisture sensor module determines the amount of soil moisture by measuring changes in capacitance to determine the water content of soil. Capacitive soil sensors not only avoid corrosion of the probe but also gives a better reading of the moisture content of the soil as opposed to using a resistive soil moisture sensor. These sensors give analog signal output and requires 3.3 to 5.5V operating voltage. [12]

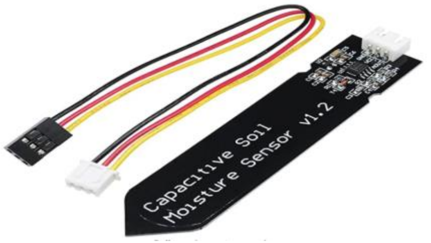


Figure 3: Capacitive soil sensors [11]

iii. Relay

A relay is an electrically operated switch. Relays are used where it is necessary to control a circuit by a separate low-power signal. A relay with calibrated operating characteristics and sometimes multiple operating coils is used to protect electrical circuits from overload.

As shown in Figure [5], the NodeMCU ESP8266 is connected to the motor via relay. Here relay can be operated as switch to on or off the motor.

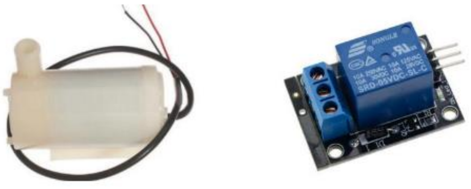


Figure 4: Mini water pump(left) and Relay(right)

3.1.2 Software Components

Arduino Integrated Development Environment (IDE) is used to write the code and upload it to NodeMCU board. It is a mix of hardware and software that functions as an open-source software which can support on various platforms like MAC OS X, Windows and Linux. The hardware consists of a circuit board with a programmable microcontroller, while the software consists of an IDE where we may develop and upload code. It supports the languages C and C++, which have special coding structure rules. The written code file then saved with the extension of .ino[13]. We have integrated the Arduino IDE with the NodeMCU board since the Arduino IDE is compatible with a wide range of circuit boards.

The web client is developed using HTML and Javascript. HTML stands for Hypertext Markup Language and is used to create webpages. HTML is a tag-based language. Javascript is a high-level, interpreted language that can easily be embedded with languages like HTML.

3.1.3 Cloud Resources

Cloud services will serve as the IoT project's backbone. Everything that doesn't need to be computed on the device itself will be done in the cloud. AWS will supply our backend for IoT services in this project via AWS IoT Core. Using their services allows you to leverage other non-IoT services that you'll need for a successful project.

This project's major hub is the AWS IoT core, which facilitates MQTT communication between the device and the cloud, as well as data processing and storage inside the cloud services. It has a user interface that is simple to use while still allowing for extensive functionality and data processing. The data processing is completed by the AWS Lambda function. The Lambda function is utilized in this implementation to compute the mean moisture level, which is then used to determine whether or not the plant should be watered. The Lambda function is also the service that makes data storage in the S3 bucket easier. AWS S3 is a basic storage service that will be used to store both raw and processed data.

3.2 Problem Statement and Analysis

Plants are beneficial to humans whether they are outdoor plants or indoor plants. People opt to have more plants in their house or yard in order to maintain a connection with nature. Many people, however, forget to water their plants throughout their daily activities, especially while they are away from home, making it difficult to maintain them healthy and growing. Beginner plant growers, on the other hand, continue to water their plants out of love until they die of root rot. Additionally, overwatering can occur in outdoor gardens when plants are watered without being aware of the probability of rain. Hence, the aim of this research project is to provide a solution by developing automated irrigation system using IoT sensors which will monitor the content of soil moisture of plants and automatically start the irrigation process only when watering is needed. In addition, this project will keep an eye on the weather forecast and refrain from watering if rain is expected.

This project assumes that the user is aware of the right soil threshold value required by the plant and can configure it using the web UI; also, if the plant is indoor, the user is responsible for deactivating advanced weather sensing in order to water the plant regardless of the weather outside. The implementation on the cloud resources for this project can be done within the Free Tier of AWS cloud services if not frequent email notifications are triggered to the user.

4 DESIGN AND IMPLEMENTATION

4.1 General Architecture of the System

The previously stated problem and solution outline were both critical in determining exactly what needed to be done to create a functional smart automatic irrigation system. The problems can be defined further once looking into the detailed implementation of three main components of the system: the cloud resources, NodeMCU device, web client. The overall design information can be seen in figure 2.

4.1.1. Cloud resources Implementation

Overall, this system uses AWS cloud as the central hub for the irrigation system. The services used includes: IoT Core, Lambda, Simple Storage Services (s3), and Simple Notification Service (ses). Additional services were used to evaluate the software such as cloud watch and other logging methods. IAM management roles were also created to help make this project work. The following section will outline all the steps done, lambda functions, and how the services interact and work.

The NodeMCU device is set up as a new thing in the IoT core, where all of the MQTT protocol certifications are generated. These certificate and key files will be stored on the physical device and used to link cloud resources to the NodeMCU. The policies for the NodeMCU's thing profile must be changed so that the device may both publish and subscribe to topics.

Topic	Publisher	Subscriber	Purpose
soilMoisture	NodeMCU	Web Client, Lambda	Sends current moisture values of plant
motorOn	Web Client, Lambda	NodeMCU, Web Client	Give signal to turn on motor
motorOff	Web Client, Lambda, NodeMCU	NodeMCU, Web Client	Give signal to turn off motor
updateConfig	WebClient	Lambda	User Configuration of Threshold and weather sensing settings in json format
weatherInfo	Lambda	WebClient	Current and Forecasted weather info in json format

Table 1: MQTT Topics used in the irrigation system

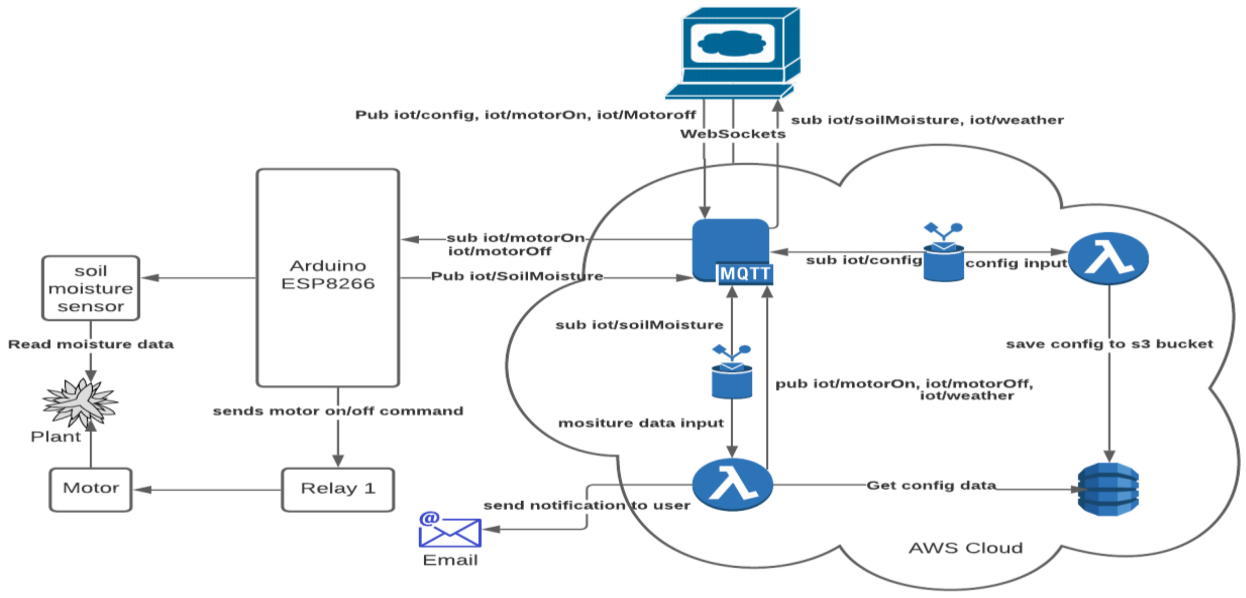


Figure 5: System design diagram

The AWS IoT console allows users to directly publish and subscribe to MQTT topics using a web interface. User can manually trigger the water pump, update the config settings, and view the weather data in json format. The MQTT topics used in this system along with their interaction is described in table 1.

Two rules will be created using AWS IoT to direct messages received to their intended destinations. Table 2 describes the rules which are created to trigger the lambda functions.

Rule	**Source Topic**	**Action**	**Purpose**
moisture_to_lambda	soilMoisture	Lambda Function - processIrrigationData	Trigger processIrrigationData lambda function to handle the soil moisture values published by NodeMCU
saveIrrigationConfigDataRule	update Config	Lambda Function - saveIrrigationConfigData	Trigger saveIrrigationConfigData lambda function to save the config settings published by the user

Table 2: AWS IoT rules used in the irrigation system

The AWS Lambda functions plays the vital role in the development of smart irrigation system. As seen in the overall system design diagram shown in Figure 5, two lambda functions were developed.

i. *SaveIrrigationConfigData:*
This lambda function is responsible for saving user configuration data to an S3 bucket. One thing to note is that the role attached to the lambda function must have the S3PutPolicy applied to it in order for the Lambda function to write to the S3 bucket. In addition, if maintaining within the limitations of AWS Free Tier, the frequency of writing to the S3 bucket must be managed by the user. As shown in the Table 2 updateConfig topic is the event source of this lambda function. The user can modify the settings from the web client and data is published in the MQTT updateConfig topic.

ii. *ProcessIrrigationData*: This lambda function performs the main algorithm of the system. As shown in the Table 2 SoilMoisture topic is the event source of this lambda function. To access other AWS resources such as AWS S3 this lambda function will need some permissions. These permissions can be handled by the execution role that is attached to the lambda function. By default, every execution role of lambda function has Amazon cloud watch logs policies, which allows lambda function to create the log group and put log streams in the created cloud watch log group. Table 3. describes the additional policies attached to the execution role of this lambda function.

Policy	**Action**	**Purpose**
S3_GetObject	s3:GetObject	Allows S3 bucket get access which enables lambda function to retrieve the files saved in the S3 bucket.
Iot-publish-all	iot:Publish	Allows IoT publish access which enables lambda function to publish the messages to MQTT topics.
Sns_sendEmail	ses:SendEmail ses:SendRawEmail	Allows SNS send email access which enables lambda function to send emails.

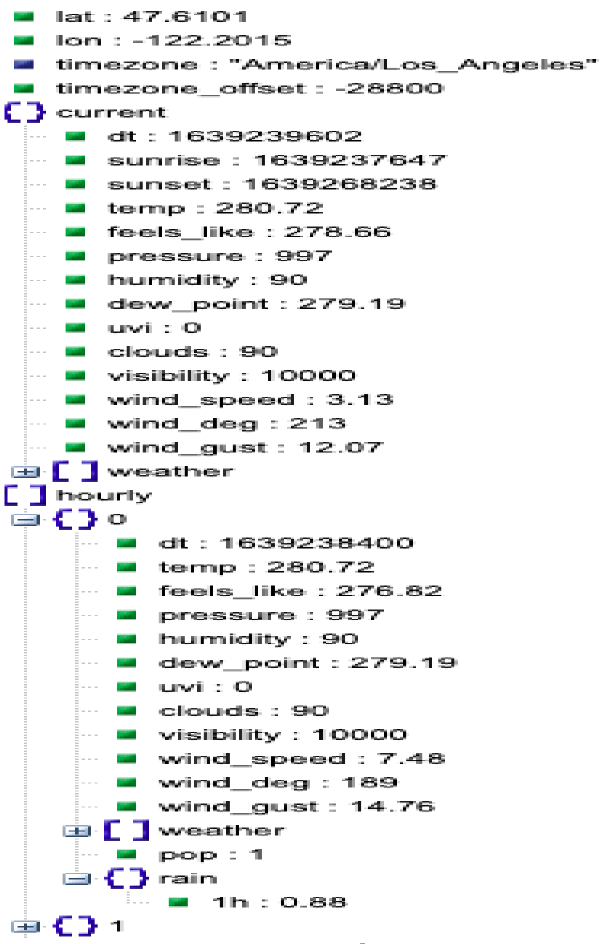
Table 3: ProcessIrrigationData specific policies

The ProcessIrrigationData lambda function works in two modes: when advance weather sensing setting activated and when it is deactivated. First, the config file is obtained from an AWS S3 bucket that contains json-formatted user settings. The weatherSensing parameter in this config file indicates if weather sensing is activated or deactivated by the user.

If the user disables weather sensing, the soil moisture value provided by the NodeMCU is compared to the threshold soil moisture value and a decision is made. The MQTT topic motorOn is published if the moisture value is greater than the threshold value, indicating that the soil is dry. The threshold value is found in the same config file that was retrieved from the S3 bucket.

When the user enables weather sensing, the system performs additional work to obtain weather prediction information and determine if rain is likely in the next 12 hours. If rain is a likely, the motor on command is not published even if the threshold is exceeded; otherwise, the command is published via MQTT motorOn topic.

For this project we have used the 12-hour window to check the rain probability but the same can be updated till 48 hours based on the location and normal weather condition around the plant. To get the weather information OpenWeatherMap api is being used. OpenWeatherMap is an online service, owned by OpenWeather Ltd, that provides global weather data via API, including current weather data, forecasts, nowcasts and historical weather data for any geographical location. The Free tier subscription is used for this project which allows up to 1,000,000 api calls per month [14]. The api requires latitude and longitude coordinates of the location as input and returns response in the json format which has current weather information and hourly forecast for the next 48 hours. The Json response given by the api is shown in figure 6.

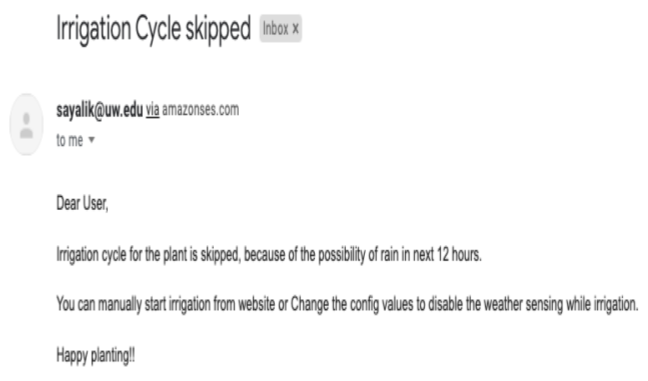


```
lat : 47.6101
lon : -122.2015
timezone : "America/Los_Angeles"
timezone_offset : -28800
{ } current
    dt : 1639239602
    sunrise : 1639237647
    sunset : 1639268238
    temp : 280.72
    feels_like : 278.66
    pressure : 997
    humidity : 90
    dew_point : 279.19
    uvi : 0
    clouds : 90
    visibility : 10000
    wind_speed : 3.13
    wind_deg : 213
    wind_gust : 12.07
  [ ] weather
[ ] hourly
  { } 0
      dt : 1639238400
      temp : 280.72
      feels_like : 276.82
      pressure : 997
      humidity : 90
      dew_point : 279.19
      uvi : 0
      clouds : 90
      visibility : 10000
      wind_speed : 7.48
      wind_deg : 189
      wind_gust : 14.76
    [ ] weather
      pop : 1
    { } rain
        1h : 0.88
  { } 1
```

Figure 6: OpenWeatherMap JSON response

The hourly weather parameter is parsed to check if rain parameter is present in the next 12-hour data. If that parameter is present, then *isRainPrediction* flag is marked as true and decision is made to not turn on the motor.

Whenever irrigation cycle is skipped, user is being notified via email. The AWS SES service is used to send the email notification. Figure 7 depicts the sample email notification.



Irrigation Cycle skipped Inbox ×

sayalik@uw.edu via amazonses.com
to me ▾

Dear User,

Irrigation cycle for the plant is skipped, because of the possibility of rain in next 12 hours.

You can manually start irrigation from website or Change the config values to disable the weather sensing while irrigation.

Happy planting!!

Figure 7: Sample email notification

4.1.2. NodeMCU device code implementation

The certification files and keys generated in the preceding step should be saved somewhere where the device-level application can access them. A secret.h header file was developed in this implementation to include all of the keys and other confidential information, such as Wi-Fi SSIDs and passwords.

The device-level program must include steps to enable connecting to AWS IoT, publishing a message, and processing a received message in order to connect the device to the cloud. The application must first connect to Wi-Fi using the Arduino Wi-Fi library and the SSID information in the secrets.h header file before connecting to AWS IoT. Next, using the PubSubClient library, develop a client to connect to the AWS IoT endpoint. The certification files and keys are used in this phase. The client will connect to the NodeMCU Thing and subscribe to the motorOn and motorOff topics once it is finished.

The device code reads the moisture value sent by sensor in the form of analog signals. The interval to read signal is taken as 1 hour. The device then publishes the moisture reading in the json format using pubSubClient library via the soilMoisture MQTT topic.

When a message to the motorOn and motorOff topics is received, the code to handle the message is activated.
If the motorOn Topic is received, the led bulletin and motor pin are given a high signal. When the program is started, the led bulletin and motor pin are set to output pins. We utilized motor pin GPIO 5 in this project, which is comparable to the d1. When the motorOff topic is received, the led bulletin and motor pin are given a low signal.

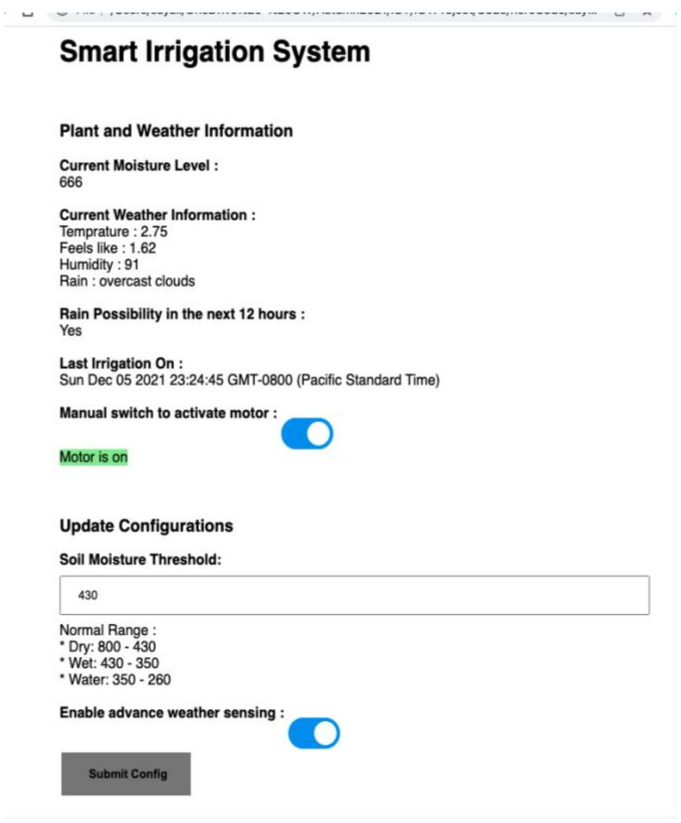
The handling of the interval setting will also be included in this code. When the motor is turned on, the interval is shortened to 5 seconds. This permits the sensor input to be read every 5 seconds, and if the moisture value reaches the threshold value, the motor is turned off by giving low signal to the motor pin, and the same MQTT message is published via the motorOff topic.

4.1.3. Web Client

An intuitive webpage is designed using HTML and Javascript. The basic layout of the web page is shown in Figure 5. To establish a websocket connection between the web client and the AWS IoT core, the Paho mqtt client library is utilized. Crypto Js is also used to construct the IoT endpoint, which includes all of the essential cryptographic security information. To begin, the aws account is used to gather security credentials and IoT endpoint information. The AWS access ID and access key are passed to the crypto Js library's SHA256 and HmacSHA256 methods, and a WebSocket URL is created with all of the security information needed to establish a successful websocket connection with AWS.

After successful connection with the AWS IoT core, Web Client subscribes to the weatherInfo, motorOn, motorOff MQTT topics. The callback handlers are initialized so that whenever

messages published on the subscribed topic callback method on the web client is triggered.
In the callback method, the payload from the MQTT message is parsed into Json object and updated in the HTML web page.


Smart Irrigation System

Plant and Weather Information

Current Moisture Level :
666

Current Weather Information :
Temprature : 2.75
Feels like : 1.62
Humidity : 91
Rain : overcast clouds

Rain Possibility in the next 12 hours :
Yes

Last Irrigation On :
Sun Dec 05 2021 23:24:45 GMT-0800 (Pacific Standard Time)

Manual switch to activate motor :

Motor is on

Update Configurations

Soil Moisture Threshold:

430

Normal Range :
* Dry: 800 - 430
* Wet: 430 - 350
* Water: 350 - 260

Enable advance weather sensing :

Figure 8: Web UI of Smart irrigation system

The web client is also utilized to communicate the information to AWS IoT core. User can activate the motor using the switch in web page and on click function *motorOn* and *motorOff* MQTT topics are published. Similarly, user can update the configuration such as soil moisture threshold value and enable/disable advance weather sensing. Whenever submit config button is clicked data is formatted into json format and published into the *updateConfig* topic.

4.2 Hardware Circuit Diagram

The hardware circuit schematic will be presented in this section as shown in Figure 9. The physical devices involved in this system are the NodeMCU ESP8266, Capacitive soil moisture sensor, mini-DC water pump and relay. The moisture sensor and the relay are both connected to the ESP8266 GPIO pins using jumper wires. The soil moisture sensor output is connected to analog pin A1. Relay input signal wire is connected to GPIO5 pin. Whenever device receives motor on or motor off command it gives high and low signals to the GPIO5 pin and as relay is connected to this pin same signal is sent to relay switch. The water pump utilized in this application had a voltage of 3-5V, making it a low-voltage choice. The positive end of the DC motor pump is connected to the normally open (NC) end of the relay and negative end is grounded.

The physical device will have a program for implementing the functionality of the microcontroller as well as the authentication information needed to access the cloud services. The setup method in the code on the physical device sets up the connection to AWS and sets the pin modes to its appropriate modes: input and output. The continuously running method that initiates the reading of the moisture level and triggers the water pump is uploaded to the NodeMCU ESP8266 using the Arduino IDE.

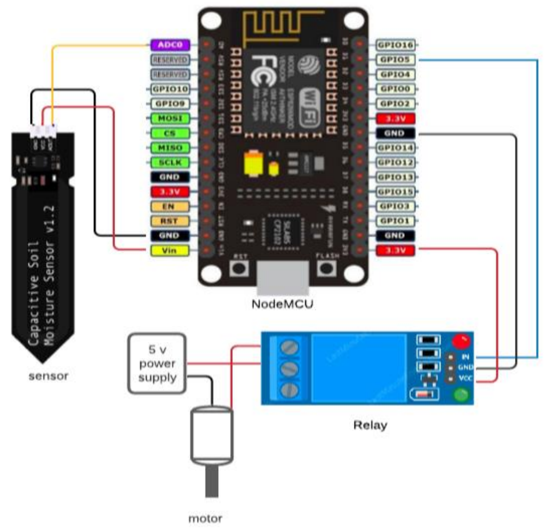


Figure 9: Hardware circuit diagram

5 Evaluation

The goal of this project's evaluation was to determine the automatic irrigation system's end-to-end latency and accuracy. The user experience and system performance would most certainly be influenced by these two qualities.

An experiment was conducted to determine the speed at which the dry soil moisture was read and when the message to start the motor was received from the cloud in order to assess the system's latency. Between these two procedures, the time elapsed was measured in milliseconds. This elapsed time had a mean value of 5002.93 milliseconds and a standard deviation of 0.4577 milliseconds. In Figure 10test, a graph depicts the distribution of these readings. This demonstrates that the elapsed time is relatively brief, and it can thus be concluded that the project's implementation delivers a quick and consistent response time in response to dry soil moisture readings.

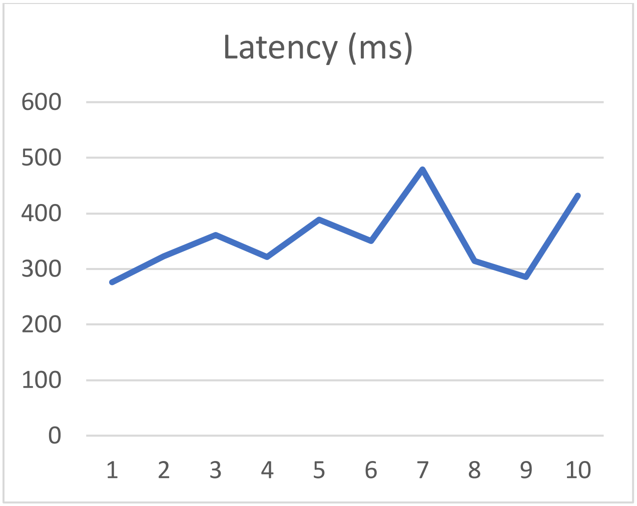


Figure 10: End to end latency of Smart irrigation System

An experiment was done on six small soil containers to test the accuracy of an autonomous watering system and measure the amount of water require by plants. The moisture content of the soil in each container ranged from 450-750. The greater the value of in the obtained reading data, the drier the soil is, and vice versa. Threshold value considered during this experiment was 430. Figure 11 shows the test results. In this experiment, when the soil moisture reading was more than the threshold, the system successfully started the motor. Furthermore, the system was able to successfully turn off the motor if the threshold of 430 was exceeded.

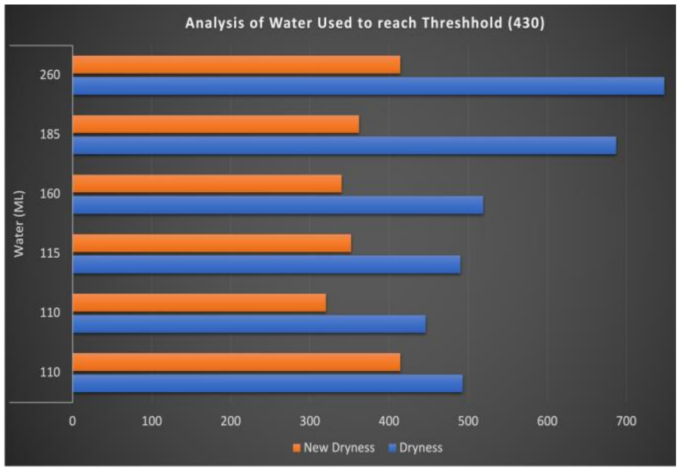


Figure 11: Analysis of water used to reach threshold

The same experiment data is utilized to estimate the water use in each irrigation cycle. Figure 12 shows the average amount of water utilized in each irrigation cycle in this experiment, whereas Figure 13 shows how long it took dry soil to achieve the below-threshold moisture value. The average amount of water utilized in this experiment was 150ml, and the average irrigation time was 6 seconds. This estimate may be used to assess the amount of water a plant requires and can be recommended to the user. When planning a trip, users may utilize this information to keep adequate water resources ready.

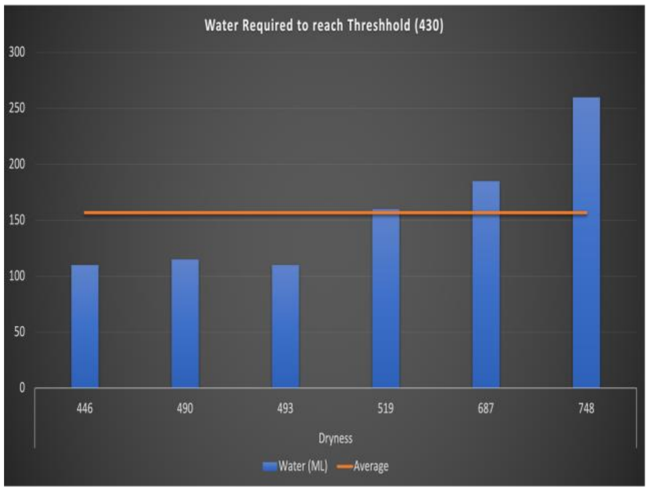


Figure 13: Water required to reach Threshold

Overall, tests suggest that the system had a reasonable latency, requiring just 300 milliseconds to complete all processes required to detect the moisture content, obtain weather data, and activate the motor. The latency of the system is suitable for most applications. Furthermore, the system has reached 100 percent accuracy in detecting soil dryness and making suitable watering decisions.

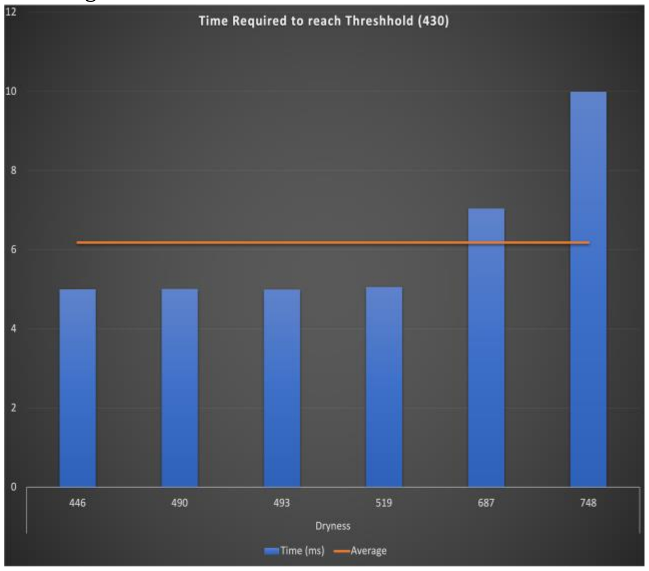


Figure 13: Time required to reach threshold

6 CONCLUSION AND FUTURE WORK

The goal of this project was to develop a system that would assist a plant owner with plant care by automatically watering plants based on their present soil moisture level and scheduling optimal irrigation cycles using weather data. This approach allows the plant owner to enjoy all of the benefits of owning a houseplant without having to worry about the proper watering care required for each individual plant. This was accomplished by reading soil moisture levels using a Wi-Fi equipped microcontroller, uploading the data to the cloud, assessing if the soil needed to be watered, and initiating a water pump to irrigate the plant. This project was a success, with further customization options that allowed users to select threshold values for determining soil moisture levels based on plant kind. In addition, the user had the option of enabling or disabling impact of the weather factors while deciding the irrigation schedule. This project creates a web interface that offers a detail about irrigation as well as weather information around the plant.

To improve the efficiency and effectiveness of the system, the following recommendations can be put into consideration. Firstly, water level sensors may be used to monitor the tank's water level, and if the level drops, the user will be notified. It would also relieve the user of the responsibility of monitoring the water level and determining when it should be refilled. The smart irrigation system will also provide recommendations on how to care for plants and identify the diseases they have using the AlexNet model, which has been trained on a dataset of plant leaves.

The scope of this project is limitless since, in today's fast-paced world, everyone will need a helping hand to care after the plant and offer status updates on its health, even if he or she is not physically present at the facility. This similar concept may be applied on a broad scale for agricultural purposes on vast acres of land, assisting farmers and reducing their workload.

REFERENCES

[1] Virginia I. Lohr, "What are the benefits of REDD + and why do they," October, vol. 881, no. October, pp. 675–682, 2010.

[2] R. J. Chakma, "An IoT based Energy Saving Automatic Watering System for Plants," vol. 9, no. 5, pp. 42–48, 2021.

[3] W. Difallah, K. Benahmed, B. Draoui, and F. Bounaama, "Design of a solar powered smart irrigation system (SPSIS) using WSN as an IoT device," ACM Int. Conf. Proceeding Ser., pp. 124–128, 2018, doi: 10.1145/3178461.3178482.

[4] M. Swapnali, B. Pawar, P. Rajput, and A. Shaikh, "Smart Irrigation System Using IOT And Raspberry Pi," Int. Res. J. Eng. Technol., p. 1163, 2008, [Online]. Available: www.irjet.net.

[5] C. Subramani et al., "IoT-Based Smart Irrigation System," Adv. Intell. Syst. Comput., vol. 1040, no. 2, pp. 357–363, 2020, doi: 10.1007/978-981-15-1451-7_39.

[6] S. Rawal, "IOT based Smart Irrigation System," Int. J. Comput. Appl., vol. 159, no. 8, pp. 7–11, 2017, doi: 10.5120/ijca2017913001.

[7] R.Suresh, S.Gopinath, K.Govindaraju, T.Devika, N.SuthanthiraVanitha, "GSM based Automated IrrigationControl using Raingun Irrigation System", InternationalJournal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 2, February 2014.

[8] J. Kwok and Y. Sun, "A smart IoT-based irrigation system with automated plant recognition using deep learning," ACM Int. Conf. Proceeding Ser., pp. 87–91, 2018, doi: 10.1145/3177457.3177506.

[9] A. Gujar, R. Joshi, A. Patil, and P. S. Aranjo, "Indoor Plant Monitoring System using NodeMCU and Deep Learning," pp. 1206–1212, 2020.

[10] "NodeMCU ESP8266 Pinout, Specifications, Features Datasheet." https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet (accessed Dec. 09, 2021).

[11] "Insight Into ESP8266 NodeMCU Features & Using It With Arduino IDE (Easy Steps)." https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/ (accessed Dec. 09, 2021).

[12] "Grove - Capacitive Soil Moisture Sensor (Corrosion Resistant) - Seeed Studio." https://www.seeedstudio.com/Grove-Capacitive-Moisture-Sensor-Corrosion-Resistant.html (accessed Dec. 09, 2021).

[13] "Getting Started with Arduino UNO | Arduino." https://www.arduino.cc/en/Guide/ArduinoUno (accessed Dec. 09, 2021).

[14] "Weather API - OpenWeatherMap." https://openweathermap.org/api (accessed Dec. 11, 2021).