IoT Based Smart and Precise Autonomous Irrigation Management System

Sudha Arvind
Dept. of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, Telangana, India
sudharvind2024@gmail.com

P Jashwanth
Dept. of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, Telangana, India
jashwanth1025@gmail.com

Sri Balaji V
Dept. of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, Telangana, India
217r1a0454@cmrtc.ac.in

Yalala Ambica
Dept. of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, Telangana, India
217r1a0463@cmrtc.ac.in

N Umakanth
Dept. of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, Telangana, India
217r1a0439@cmrtc.ac.in

***Abstract*—Traditional irrigation methods are highly inefficient, labor intensive, and results in water wastage. Additionally unpredicted weather, unanticipated rainfalls could compromise the crop health. Addressing all the challenges the proposed IoT system automates the irrigation by using real-time data such as rainfall, soil-moisture, humidity, temperature and crop area and sends to a python-based engine, which calculates water requirement, controls the water pumps in the fields through microcontroller NodeMCU, provides the required water to the crop. By doing so, the system not only conserves the water but also reduces the labor costs, prevents under- or over-irrigation, increases the crop productivity, promotes the sustainable farming practices. The updates regarding the irrigation can be viewed by the web application controlled by the python-engine. The system is versatile, suitable for open fields, aeroponic towers, gardens, small farms and research purposes. The desired objective is to reduce the water consumption by 40% and labor cost by 90%, make the irrigation efficient and reduce cost.**

Keywords*—*IoT (Internet of Things), Smart Irrigation, Weather monitoring, Python-based engine, Moisture sensor, NodeMCU.

I. Introduction

In this ever-evolving world with current technology we can solve many problems by integrating modern technology to the solutions. As we know, agriculture is the main resource for the humanity to function, since it is the producer of the energy to the human civilization, most certainly from plant agriculture. By increasing the productivity in the farming sector, we can improve the quality of life. One of the major processes in the farming is irrigation. Recent researches and studies have claimed by introducing the IoT and building a smart irrigation system that enhances the irrigation methods, improves the productivity, and reduces the labor costs and time spent for the irrigation. Sometimes, overwatering occurs in case of unanticipated rainfall or change in weather conditions. Even though rainfall data is available at the local weather station, it might not be reaches to the farmers in time.

From the studies done on the irrigation for plants, overwatering will rot the roots, and underwatering halts the growth of plants [1],[2]. At a time to conserve water, and provide the accurate watering to the plants using IoT system can be done using sprinkler systems according to [3] this concept is beneficial for the water scarce areas. The idea of controlling motor through a microcontroller based on given inputs of soil moisture, temperature, humidity and water levels and creating an embedded system explored in Literature survey. In [4], there are clear definitions of and sensor related parameters were discussed for temperature, humidity, and very particularly soil moisture determines the measure of soil water content which defines its expression in terms of mass or volume of water content of the soil. On the basis of mass, soil water content is expressed in the gravimetric soil moisture content θg given by:

$$\theta_g = \frac{M_{water}}{M_{soil}} \tag{1}$$

where M_{water} is the mass of the water in the soil sample and M_{soil} is the mass of dry soil that is contained in the sample.

The volumetric soil moisture content of a soil θ_v, is defined by:

$$\theta_v = \frac{V_{water}}{V_{sample}} \tag{2}$$

Where, V_{water} is the volume of water content in the soil and V_{sample} is the total volume of dry soil + water + air present in the sample of field.

Both the values θ_g and θ_v are usually expressed in percent. The relationship between the gravimetric soil moisture content and volumetric soil moisture content is:

$$\theta_v = \theta_g \left(\frac{\rho_b}{\rho_w}\right) \tag{3}$$

Where, ρ_b is the dry soil bulk density and ρ_w is the soil water density.

The relation between the temperature and soil moisture were discussed in [5], regarding the heat capacity of soil and the water quantity in the soil are related and the equations can be given as follows:

The equation (4), calculates the amount of heat (Q) required to change the temperature of the soil by a specific amount and shows that soil moisture (W) is a crucial factor in determining how much heat the soil can absorb or release.

$$Q = 4.2 \times 10^3 \times V \times (0.2 + W) \times \Delta T \tag{4}$$

Where, V is the volume of the soil in m^3, ΔT is the temperature change in K (Kelvins), term $(0.2 + W)$ adjusts for both solid and water components.

Relative humidity concepts, which effect leaf growth, are clearly presented in [6], how the irrigation water influences the effective humidity in the air can also be viewed which is useful in order to calculate the water that to be provided to the crop. For the interface development, a web application can be created to allow farmers to monitor crop data from anywhere. This application will be independent of the software system used, and to build this platform, a clear understanding was taken from [7]. The step-by-step process to develop a secure platform for the current system is referred to in [8].

Even though all these parameters are considered, the unanticipated rainfall is the main problem that needs to be addressed, which is the main objective of the current study: to develop a system that will be taking the weather and rainfall that is going to be happening in the area is taken from the local weather stations. The required amount of water to be given to the field will vary according to the data based on the probability of the rainfall. The main motives of the study are to reduce the water usage, increase the productivity of the irrigation, reduce cost of labour, integrate technology and create a climate-resilient system, along with smart and adaptive agricultural practice to increase the sustainability.

The objectives and key contributions of this study are:

- To create an automatic IoT system using NodeMCU to improve and automate irrigation practices and increase productivity of farming.
- Integrate IoT-based sensors to monitor real-time data such as temperature, humidity, soil moisture and other parameters like crop area, motor capacity to create accurate results for precise irrigation.
- A Python-based server application accesses the data from the field, calculates, and controls motor based on the analysis of data.
- Apart from sensor data collection, use a Python engine to collect the data of the rainfall for particular area to include in the parameters which is main objective of study.
- To perform the practical operations using the developed system to study the efficiency, effectiveness and how scalable the system, and use it for future scope of challenges.
- Overall objective is to give a sustainable approach to improving irrigation and cost effectiveness compared to the previous methods.

II. Literature Survey

The literature review presents various IoT-based irrigation systems, ranging from simple sensor-based control to advanced AI-driven solutions. While these systems improve efficiency, they often lack adaptability, reliable communication, and scalability. Our proposed system builds on these insights by integrating real-time weather data, adaptive irrigation, and secure communication, addressing these limitations to offer a more efficient and scalable solution for sustainable irrigation.

A. *Smart Agriculture for Sustainability: The Implementation of Smart Irrigation Using Real-Time Embedded System Technology:*

This system, controlled by Arduino, collects data from sensors like temperature, humidity, and soil moisture, displaying output via a terminal or LCD. The algorithm in Arduino controls the motor based on these inputs, ensuring efficient irrigation based on real-time data. [9]

B. *A distributed system for supporting smart irrigation using Internet of Things technology:*

A Raspberry Pi-controlled distributed system collects inputs and stores data on a server. The system allows manual irrigation control via an Android app. It's cost-effective, automated, and alerts the farmer via text about motor activity. [10]

C. *An IoT Based Smart Irrigation System:*

This Arduino-based system collects soil moisture, temperature, and humidity data, presenting it on a display and allowing manual motor control. It requires user knowledge to operate but is not fully automatic. [11]

D. *Smart Irrigation system using IoT:*

This system uses a NodeMCU to take soil moisture readings, activating the water motor when moisture is below a threshold. It's a simple system, but prone to errors due to sensor inaccuracies. [12]

E. *Internet of Things Application for Implementation of Smart Agriculture System:*

A mobile robot powered by Raspberry Pi monitors parameters like soil moisture, pH, temperature, and humidity. Based on the collected data, the robot waters the crop area via a sprinkler or motor, providing accurate irrigation. [13]

F. *An IoT Based Soil Moisture Monitoring on Losant Platform:*

This system uses a soil moisture sensor to continuously collect data, which is sent to the Losant platform for real-time monitoring. If moisture levels fall below a set threshold, the system sends alerts to the user, ensuring timely intervention and water conservation. [14]

G. *Garden Watering System Based on Moisture Sensing:*

A system based on the Teensy 2.0 microcontroller uses soil moisture readings to determine the watering amount, ensuring efficient water usage and preventing over-watering. [15]

H. *Automatic Irrigation System Based on Computer Vision and an Artificial Intelligence Technique Using Raspberry Pi:*

This system utilizes computer vision to analyze soil images, deciding when to water plants based on processed datasets. Due to high power consumption and cost, it's more suited for smaller crops requiring precision irrigation. [16]

I. Smart Irrigation System Techniques using Artificial Intelligence and IoT:

This review discusses various irrigation systems leveraging Artificial Intelligence and IoT, with some systems fully automated using machine learning to manage irrigation efficiently. [17]

The reviewed literature provides a diverse range of IoT-based irrigation solutions that improve efficiency but still face challenges like adaptability and scalability. Building on these findings, our proposed system integrates advanced features such as real-time weather data, adaptive irrigation, and secure communication, addressing these existing gaps for a more robust and sustainable solution.

III. System Description and Tools Used

A. IoT Architecture of System:

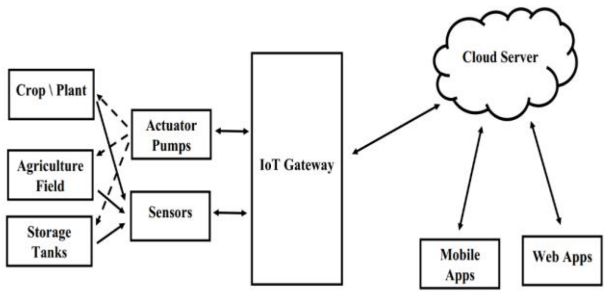


Fig. 1. Smart farming architecture[18]

The system is based on IoT architecture, it is given as shown in the Fig. 1 and the benefits of this architecture are represented in [18]. It is scalable, and a distributed system can be created, which aligns with the current system's goal. The current system is dependent, where the irrigation of 'N' number fields is automated and controlled by the single server. Based on the registered number of fields in the server, the irrigation is managed by the microcontrollers installed in the fields, which are connected to an IoT-based cloud server. The system is controlled by the outputs generated by a Python engine running on the server side.

B. Components used:

The hardware components for developing the prototype, which works on a small scale and can be easily implemented in larger regions, must be chosen carefully. The components for the system were selected to reduce costs and ensure that all the required features are available on board. The selection process involved studying previous research and reviews on Smart Irrigation systems, particularly those focusing on hardware. For a more detailed view, one can refer to [12] and [18] for insights of the different types of implementation hardware components available in market and suitable options based on specific requirements.

1) Esp8266 (NodeMCU):

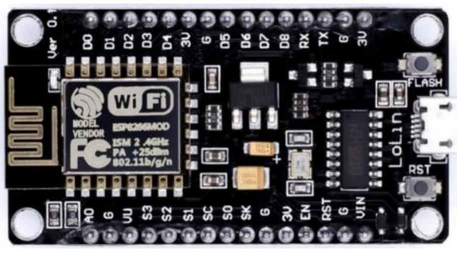


Fig. 2. ESP8266 microcontroller

The Fig. 2 shows the ESP8266 microcontroller has 20 accessible pins, including 1 Vin (5V input), 1 GND, 1 analog input pin (A0), and 16 GPIO pins (GPIO0-GPIO16) for digital I/O. It also features RX and TX pins for UART communication, providing versatile interfacing options. The ESP8266 was selected for its cost-effectiveness and built-in Wi-Fi capability, making it ideal for IoT applications. It features a built-in TCP/IP stack, GPIO pins, and supports various communication protocols meets all criteria to work as controller of irrigation process at the end-user side, exchanging of data with the Python-based application on the cloud server.

2) Soil Moisture Sensor (SMS):

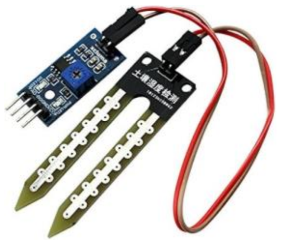


Fig. 3. SMS (Soil Moisture Sensor) with LM393 comparator

A soil moisture sensor measures the water content in soil to optimize irrigation. It includes an LM393 comparator with a potentiometer to adjust sensitivity. Two types exist: Frequency Domain Sensors, which measure soil's dielectric constant to assess moisture, and Neutron Moisture Gauges, which use neutron slowing to estimate water levels. Based on [2],[19] the soil moisture threshold value will vary for different soils, different crop, and the type of sensor used.

3) Digital Humidity and Temperature Sensor (DHT11):

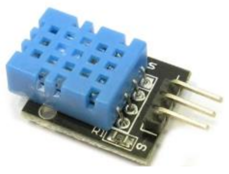


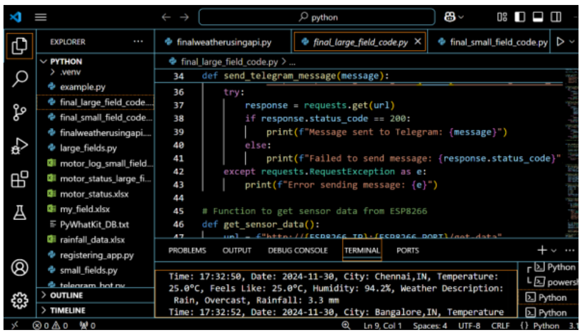
Fig. 4. DHT11 (digital humidity and temperature) sensor module

The humidity and temperature in air affect the soil moisture, which in turn impacts crop growth as discussed in [5] and [6]. The effects of temperature and humidity were considered to provide precise irrigation to the field.

C. Softwares and protocols used

The system primarily operates through a Python-based engine running on the server side, which is connected to the end user's microcontroller coded in C/C++. The microcontroller communicates with the Python engine to exchange information through various protocols such as HTTP over TCP/IP and Thread-based communications.

1) *Python application on server side:*



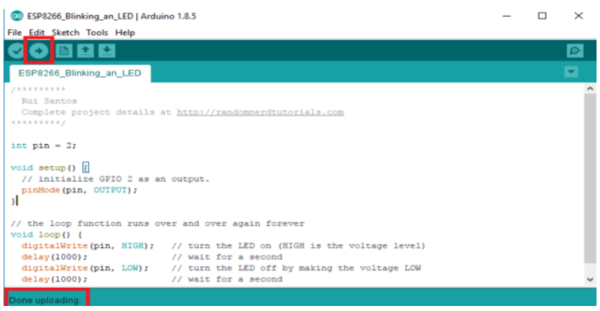
```python
def send_telegram_message(message):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            print(f"Message sent to Telegram: {message}")
        else:
            print(f"Failed to send message: {response.status_code}"
    except requests.RequestException as e:
        print(f"Error sending message: {e}")

# Function to get sensor data from ESP8266
def get_sensor_data():
```

```
Time: 17:32:50, Date: 2024-11-30, City: Chennai,IN, Temperature:
25.0°C, Feels Like: 25.0°C, Humidity: 94.2%, Weather Description:
 Rain, Overcast, Rainfall: 3.3 mm
Time: 17:32:52, Date: 2024-11-30, City: Bangalore,IN, Temperature
```

Fig. 5. VS Code is used in order to build the Python engine

The Python-based engine responsible for running a web interface over HTTP to collect inputs from the field data stored on the server, sensor data from the microcontroller, and rainfall data from a reliable online source. Additionally, all logs of motor activity are saved on the server and alerts to the farmer. The process of building a Python application can be explored in [7]. Methods to implement encrypted, end-to-end communication are discussed in [8]. As shown in the Fig. 5, the Python application can be easily developed, with libraries readily available for download through VS Code. Alternatively, PyCharm can be used, as it offers greater efficiency Python application development.

2) *Arduino IDE:*



```c
/*********
  Rui Santos
  Complete project details at http://randomnerdtutorials.com
*********/

int pin = 2;

void setup() {
  // initialize GPIO 2 as an output.
  pinMode(pin, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(pin, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(pin, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
```

```
Done uploading.
```

ESP8266_Blinking_an_LED | Arduino 1.8.5
File Edit Sketch Tools Help
ESP8266_Blinking_an_LED

Fig. 6. Arduino IDE with ESP8266 installations [20]

Arduino IDE can be used to set up the ESP8266 with the required installations for the ESP modules. The complete procedure for this setup can be referred to in [20].

3) *Libraries and Protocols:*

The system utilizes several libraries to perform different functions, including collecting data, creating a web interface, making requests, manipulating data, performing mathematical operations, providing accurate timing for irrigation and messaging, creating Excel files, and using libraries for JavaScript, HTML, and CSS purposes, as well as creating threads.

a) Flask library

Helps to create a lightweight web framework used to represent continuous real-time data from inputs in the web interface.

b) Requests library

This library simplifies the process of collecting sensor data by enabling easy HTTP requests to and from the ESP8266, sending control commands to manage devices like motors, and communicating with Telegram's Bot API for messaging alerts to user. This integration ensures efficient and reliable data collection and communication within the system.

c) Pandas library

Used for data manipulation and analysis. In the system, it is used to read data from the server to extract field data and rainfall, log motor status into the server, and process rainfall, motor runtime, and field data using DataFrames.

d) Numpy library

Provides support for array and mathematical operations for calculating probabilities and percentage values from the extracted inputs, providing accurate motor runtime. This has more prominence in the results section.

e) Datetime library

Handles date and time operations. In the system, it logs timestamps when the motor state changes and calculates elapsed motor runtime based on those timestamps.

f) Threading library

Used to create concurrent execution of multiple tasks, run the Flask server and the main motor control loop, and send messaging alerts while saving logs on the server.

g) OS library

Provides utilities for interacting with the operating system. It extracts existing log files from the computer system and creates a new log file if one doesn't exist.

h) Chart.js

Creates charts on the web interface, rendering lines and displaying values graphically in real time.

Since Python includes all the necessary libraries to support the system, it is the ideal choice for building a high-performance system capable of executing complex calculations and offering user-friendly features. The Python engine enables better integration and supports a wide range of functionalities required for the system's operation.

IV. Results And Discussion

In this study, the developed system is based on sensor data collected from the ESP8266. Rainfall data is gathered from the OpenWeatherAPI or, for more accurate results, from local weather forecasting stations and stored daily on the server. Registered field data is logged in Excel format by user application remains unchanged throughout the crop period. Using these data as inputs, the Python engine performs a mathematical analysis to determine the appropriate amount of water to given to each specific registered field. Output control signals are sent accordingly to ESP266 for motor control.

Based on the following formulas and mathematical analysis, the output of the irrigation is controlled:

For rainfall, the percentage change is calculated as follows:

$$\%\,Variation\ due\ to\ rain(P_R) = \left(\frac{T\times 100}{n} \div 2 + 50\right) \qquad (5)$$

Where P_R is the percentage change in expected rainfall, T is the total true incidents from past values given by T = $\sum_{i=1}^{n} t_i$, with t_i represents events where rainfall occurred (either 0 or 1) based on forecasted rainfall values. n is the total number of events considered. The value of P_R provides an estimate of the percentage of expected rainfall, which, when multiplied by the rainfall expected today, yields a more accurate measure of total rainfall. This adjusted value is then deducted from the irrigation water supplied to the crop. This parameter optimizes water management and reduces over-irrigation.

Based on the crop type, the soil moisture threshold value will change, as different crops require different soil moisture levels. For example, the ideal soil moisture level for fenugreek is low, whereas paddy requires a high-water content. Additionally, the soil moisture characteristics can vary, so a thorough analysis should be conducted to determine appropriate threshold values that will promote productive crop growth.

For temperature and humidity effects, soil moisture can vary by 5% to 10%, and these variations may differ based on temperature changes, as can be understand from equation (4).

The total percentage effect of water is given by:

The total percentage change (F_t) is given by,

$$F_t = \%SMV + \%TV - P_R - \%HV \qquad (6)$$

Where:

- %SMV is the percentage change in soil moisture value derived from the analog SMS sensor readings and calibrated according to equations (1)-(4),
- %TV is the percentage change that occurs due to temperature change,
- %HV is the percentage change that occurs due to humidity,
- P_R is the percentage change of rain that is expected to occur.

Based on the value of 'F_t' the total irrigation water is provide to the system is given by:

$$V_{i.w.} = F_t \times A_c \times V_{w.r.} \qquad (7)$$

Equation (7) represents the total irrigation water required for the crop, where: $V_{i.w.}$ is Total volume of irrigation water, in 'liters', needed for the crop, F_t is total percentage of water to be provided with respect to the normal value, A_c is a constant for irrigation water, representing the height of water needed for the crop, $V_{w.r.}$ is the Volume of water required per unit area for the crop/field.

Using Equation (7), the motor run time (M.R.T) is calculated by:

$$M.R.T = \frac{V_{i.w.}}{M_{w.c.}} \qquad (8)$$

Where, M.R.T represents the Motor Run Time, in minutes. $V_{i.w.}$ is Total volume of irrigation water, in liters, needed for the crop, and $M_{w.c.}$ is the mass of water content drawn by the motor in the field per minute.

The key result obtained from the mathematical analysis is the value of M.R.T., which determines how long the motor should remain ON. Based on the M.R.T. value, the control signal is sent from the Python engine to the ESP8266. These calculations are entirely performed by the Python engine on the server, enabling the autonomous irrigation of multiple crops using a single server. The calculations are designed to ensure accurate responses and avoid over-irrigation or under-irrigation of crops. Probabilistic theories and percentage values are used to achieve precision. Additionally, the parameter A_c changes based on the type of crop, soil, and irrigation practices (e.g., land-based, aeroponic, sprinklers) to make informed and accurate decisions about water management and irrigation.

A. Implementation

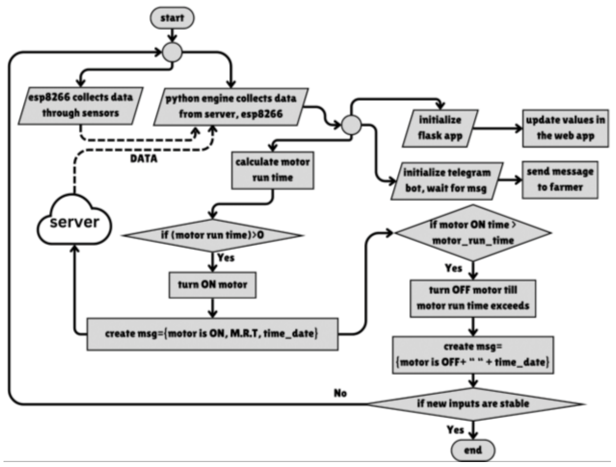

start
esp8266 collects data through sensors
python engine collects data from server, esp8266
DATA
server
calculate motor run time
initialize flask app
update values in the web app
initialize telegram bot, wait for msg
send message to farmer
if (motor run time)>0
Yes
turn ON motor
create msg={motor is ON, M.R.T, time_date}
if motor ON time > motor_run_time
Yes
turn OFF motor till motor run time exceeds
create msg= {motor is OFF+ " " + time_date}
No
if new inputs are stable
Yes
end

Fig. 7. Implementation flow diagram of the current system, showing how the system operates autonomously

The implementation of the system comprises hardware and software components. The hardware includes the main controller, ESP8266, connected to sensors and motors. It captures input values every 5 seconds during the set irrigation time for the crops and sends this data to the Python engine on the server. The Python engine performs calculations, fetches rainfall data, operates the web interface, sends alert messages, controls the motor by sending signals to the ESP8266, and logs motor activity on the server. The inputs are stored in three forms: (1) Field data (e.g., crop area, type, motor capacity, owner) is saved on the server and remains unchanged throughout the crop period; (2) Rainfall data, collected daily for the area from the OpenWeatherAPI, is stored in Excel format on the server which is later accessed for the analysis purposes as discussed earlier; and (3) Sensor values, captured by the ESP8266 from the sensors for every 5 seconds deployed in the crop area. All these inputs are required in the calculation of the irrigation water to be provided.

From Fig. 7, illustrates how the system operates: The ESP8266 is connected to the Python engine. In this basic setup, a crop area of approximately 1.5 m² is chosen, with sensors deployed to monitor the system's performance. Upon powering the microcontroller, it connects to the assigned network, allowing the Python engine to access it using a unique authentication token. Once the connection is established, the following tasks are performed concurrently by the Python engine:

- Retrieves rainfall data from reliable online sources.
- Displays real-time sensor data and motor status on the web page via the Flask application,

- Accesses stored data on the server regarding the specific field in operation and uses it for further processing.
- Initializes the Telegram bot to send alerts to the field owner whenever the motor state changes.
- Calculates the motor runtime using the equations for determining irrigation water requirements.
- Sends control signals to the microcontroller to turn the motor ON or OFF.
- Logs motor ON/OFF times on the server for future reference.

The system has three main outputs: motor control via Wi-Fi communication between a Python engine and the ESP8266, real-time Telegram alerts to the user about the motor's state, runtime, and rainfall data, and a web interface displaying real-time sensor data, motor status, and rainfall information, accessible through any web browser.

B. *Outputs and Results:*

1) Weather output collected by system:

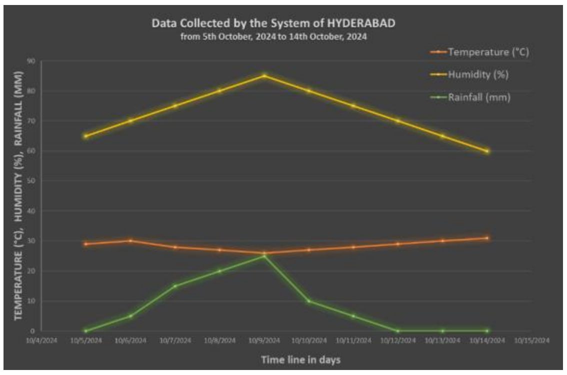
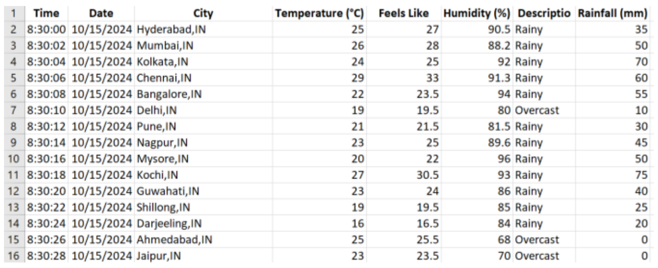


Fig. 8. Weather data of Hyderabad collected by the system, projected in a graph for clear understanding of the system's accuracy

The system collects rainfall, temperature, and humidity data from sources like OpenWeatherAPI, with local weather stations recommended for greater accurac.

Weather data for Hyderabad, as shown in Fig. 8, has been extracted and stored on the server. The graph displays time on the X-axis (from October 5 to October 14, 2024) and humidity (% in yellow), temperature (°C in orange), and rainfall (mm in green) on the Y-axis. This visualization aids in weather analysis and evaluates the accuracy of OpenWeatherAPI data, with a 70% correlation between the API data and actual conditions. The rainfall data is also stored in an Excel sheet, as shown in Fig. 9 these excel data is preserved for future use.



1	Time	Date	City	Temperature (°C)	Feels Like	Humidity (%)	Descriptio	Rainfall (mm)
2	8:30:00	10/15/2024	Hyderabad,IN	25	27	90.5	Rainy	35
3	8:30:02	10/15/2024	Mumbai,IN	26	28	88.2	Rainy	50
4	8:30:04	10/15/2024	Kolkata,IN	24	25	92	Rainy	70
5	8:30:06	10/15/2024	Chennai,IN	29	33	91.3	Rainy	60
6	8:30:08	10/15/2024	Bangalore,IN	22	23.5	94	Rainy	55
7	8:30:10	10/15/2024	Delhi,IN	19	19.5	80	Overcast	10
8	8:30:12	10/15/2024	Pune,IN	21	21.5	81.5	Rainy	30
9	8:30:14	10/15/2024	Nagpur,IN	23	25	89.6	Rainy	45
10	8:30:16	10/15/2024	Mysore,IN	20	22	96	Rainy	50
11	8:30:18	10/15/2024	Kochi,IN	27	30.5	93	Rainy	75
12	8:30:20	10/15/2024	Guwahati,IN	23	24	86	Rainy	40
13	8:30:22	10/15/2024	Shillong,IN	19	19.5	85	Rainy	25
14	8:30:24	10/15/2024	Darjeeling,IN	16	16.5	84	Rainy	20
15	8:30:26	10/15/2024	Ahmedabad,IN	25	25.5	68	Overcast	0
16	8:30:28	10/15/2024	Jaipur,IN	23	23.5	70	Overcast	0

Fig. 9. Weather data collected and logged in an Excel sheet by the Python engine.

This enhancement to the current system introduces an innovative concept: leveraging future weather predictions from OpenWeatherAPI to optimize irrigation planning. By utilizing probability-based calculations grounded in historical data, the system significantly reduces the risk of overwatering. Furthermore, the integration of real-time weather forecasts, sensor data, and probabilistic analysis ensures higher precision and reliability compared to traditional systems, making it a more accurate and efficient solution.

2) System functioning:

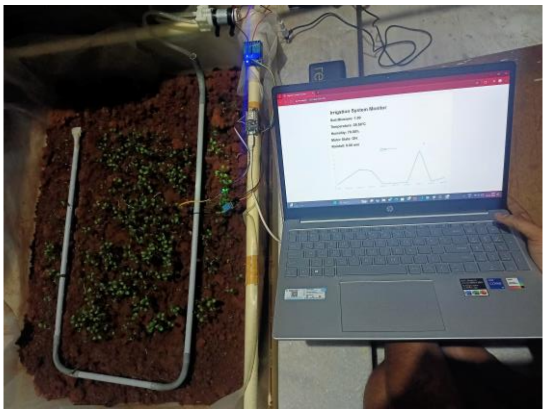


Fig. 10. System's output and implementation

As shown in Fig. 10, the system functions through Python and the ESP8266 exchanging information, which controls the motor based on the control signals sent to the ESP module from the Python engine. In Fig. 10, the motor in the crop field is connected to a drip irrigation system, and the crop being grown is fenugreek. The watering process is determined based on the area of the crop, type of soil, and threshold values assigned to the variables. Watering is scheduled at specific time intervals to ensure proper irrigation management.

The real-time sensor and rainfall data can be viewed through the web interface, which is OS-independent, allowing users to monitor the field's data from any device. Since the entire process is controlled by the IoT system, no human intervention is required for irrigation, which reduces labor costs and increases productivity.

a) Plants growth due to current system:

The system functions as designed, based on the provided values. Irrigation for the fenugreek crop, grown in a 1.5-square-meter area, was continuously monitored. As shown in Fig. 11, the crop was cultivated without any human intervention. The IoT system consistently monitored the field and executed its assigned tasks according to the calculations.

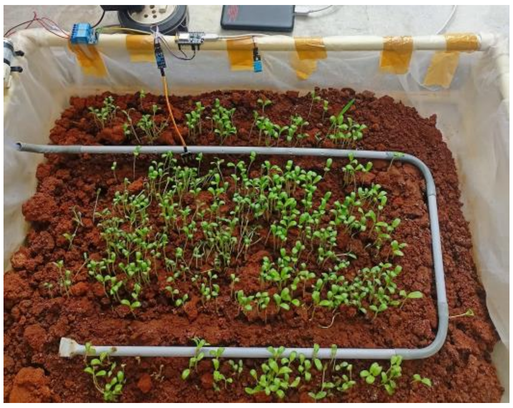
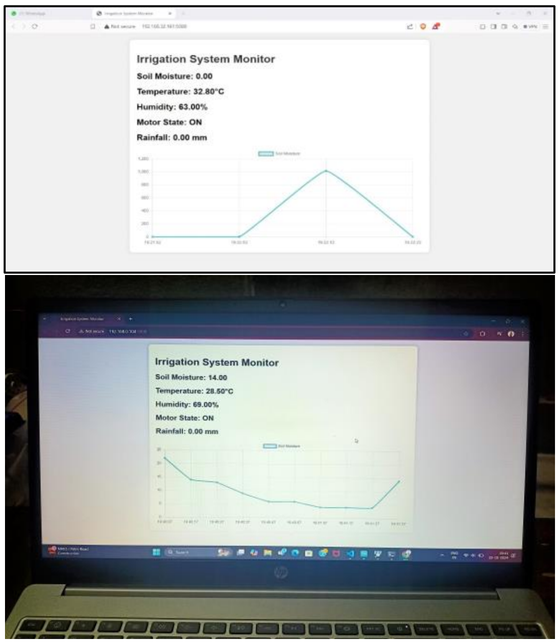


Fig. 11. Day-4 of fenugreek growth from the seed planting

b) Output of the web interface:

The system utilizes the Flask framework to create a web interface for real-time monitoring and control. This web interface is developed using HTML and CSS, ensuring a user-friendly design. A dynamic graph is generated on the interface to continuously display soil moisture levels, which play a crucial role in determining irrigation needs. In addition, key environmental data such as temperature, humidity, and rainfall (measured in millimeters) for the specified area is also presented on the web interface.

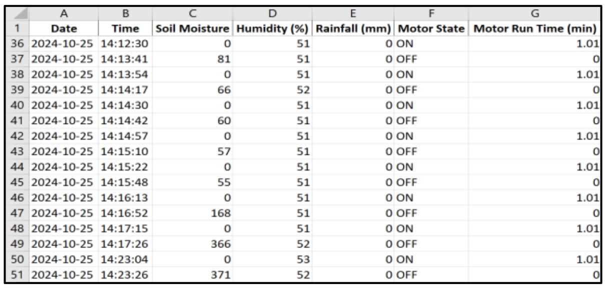

Irrigation System Monitor
Soil Moisture: 0.00
Temperature: 32.80°C
Humidity: 63.00%
Motor State: ON
Rainfall: 0.00 mm

Irrigation System Monitor
Soil Moisture: 14.00
Temperature: 28.50°C
Humidity: 69.00%
Motor State: ON
Rainfall: 0.00 mm

Fig. 12. Web interface of the System

As shown in Fig. 12, the web interface allows users to access and monitor this data in real time from any internet-connected device, ensuring flexibility and convenience. This seamless access to real-time information ensures the farmer stays informed about field conditions and irrigation schedules without the need for constant physical supervision, thereby improving efficiency and reducing manual effort.

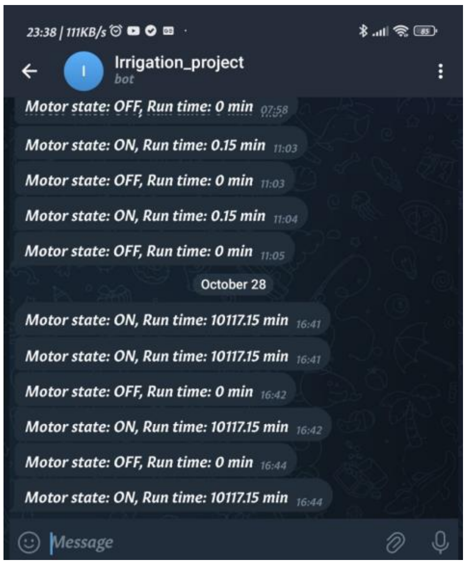
c) Logging and Text Alerts:



	A	B	C	D	E	F	G
1	Date	Time	Soil Moisture	Humidity (%)	Rainfall (mm)	Motor State	Motor Run Time (min)
36	2024-10-25	14:12:30	0	51	0	ON	1.01
37	2024-10-25	14:13:41	81	51	0	OFF	0
38	2024-10-25	14:13:54	0	51	0	ON	1.01
39	2024-10-25	14:14:17	66	52	0	OFF	0
40	2024-10-25	14:14:30	0	51	0	ON	1.01
41	2024-10-25	14:14:42	60	51	0	OFF	0
42	2024-10-25	14:14:57	0	51	0	ON	1.01
43	2024-10-25	14:15:10	57	51	0	OFF	0
44	2024-10-25	14:15:22	0	51	0	ON	1.01
45	2024-10-25	14:15:48	55	51	0	OFF	0
46	2024-10-25	14:16:13	0	51	0	ON	1.01
47	2024-10-25	14:16:52	168	51	0	OFF	0
48	2024-10-25	14:17:15	0	51	0	ON	1.01
49	2024-10-25	14:17:26	366	52	0	OFF	0
50	2024-10-25	14:23:04	0	53	0	ON	1.01
51	2024-10-25	14:23:26	371	52	0	OFF	0

Fig. 13. Motor Logs saved in Excel sheet format in server

The Fig. 13 provides an overview of how the Python engine logs motor ON/OFF statuses along with timestamps, enabling effective monitoring and analysis. The figure demonstrates continuous data logging performed during testing, which captures motor states and relevant parameters such as soil moisture, humidity, and rainfall. These logs are instrumental in verifying system performance and diagnosing issues, such as identifying faulty sensors. Additionally, they offer valuable insights for further research and optimization of the system.


23:38 | 111KB/s
Irrigation_project
bot

Motor state: OFF, Run time: 0 min 07:58
Motor state: ON, Run time: 0.15 min 11:03
Motor state: OFF, Run time: 0 min 11:03
Motor state: ON, Run time: 0.15 min 11:04
Motor state: OFF, Run time: 0 min 11:05
October 28
Motor state: ON, Run time: 10117.15 min 16:41
Motor state: ON, Run time: 10117.15 min 16:41
Motor state: OFF, Run time: 0 min 16:42
Motor state: ON, Run time: 10117.15 min 16:42
Motor state: OFF, Run time: 0 min 16:44
Motor state: ON, Run time: 10117.15 min 16:44

Message

Fig. 14. Message alerting system via Telegram controlled by Python-engine

As shown in Fig. 14, the Python engine also integrates a Telegram-based alert system to notify users of any changes in the motor state. Each time the motor is activated or deactivated, a real-time message is sent to the field owner, ensuring they remain informed about irrigation activities. Telegram was chosen over alternatives like email due to its open-source nature, speed, reliability, and user-friendly interface.

Together, these features ensure precise and automated irrigation, driven by predefined parameters and schedules. This design allows for flexibility, enabling adjustments based on crop-specific requirements, thereby enhancing the system's adaptability to diverse agricultural practices.

V. Conclusion And Perspective

In conclusion, this study presents an autonomous irrigation system that enhances irrigation practices by incorporating multiple parameters and calculations. The system is versatile, sustainable, and adaptable, capable of scaling based on additional input data and output types. By utilizing probabilistic theories for water supply calculations, as discussed in equations (5) to (8), the system improves water conservation by 40% and reduces labor costs by 90%, addressing water scarcity in arid regions. Its cost-effectiveness and scalability make it an ideal solution for both small and large-scale farmers. The Python-based engine supports various libraries and functions to efficiently manage data logging, user alerts, and real-time sensor data updates via a Flask-based web interface, with updates occurring every 10 seconds. A single server is responsible for managing a large number of fields, ensuring the system's scalability and adaptability.

The integration of IoT sensors, such as the DHT11 and soil moisture sensors, along with weather data from OpenWeatherAPI, ensures accurate and effective water management. Although Wi-Fi communication is currently used, the system could be upgraded to LoRa for higher reliability and security. The system's mathematical analysis of irrigation parameters, including rainfall data, guarantees precise water delivery to crops.

Additionally, the system is cost-effective, leveraging low-cost microcontrollers and readily available resources. The findings from this study have broad applicability, extending to various agricultural methods such as aeroponic towers, drip irrigation, and home gardening. Its user-friendly design requires minimal user knowledge, as irrigation is fully automated, with the primary challenge being the initial calibration process.

Overall, this study demonstrates the potential of technological advancements in irrigation, providing a practical solution for sustainable agriculture and laying the groundwork for future research and development in the field.

VI. Future Scope

For future improvements, the system could integrate LoRa communication to enhance reliability, speed, and long-range capabilities, especially for larger agricultural areas. Additionally, incorporating machine learning and AI could optimize irrigation practices automatically based on real-time data, further enhancing sustainability. However, the study has some limitations, such as not accounting for soil wilting capacity across various soil types and lacking large datasets for different crops. Future research will focus on expanding these datasets to support a wider range of crops and soil types. By adding these datasets to the server-side Python engine, the system can efficiently manage multiple fields. Integrating AI for predictive irrigation, expanding crop and soil datasets, and upgrading to LoRa communication will improve the system's scalability, reliability, and suitability for large-scale agricultural applications.

Acknowledgment

Our profound gratitude to Dr. Sudha Arvind, Professor in the Department of Electronics and Communication Engineering, for her invaluable guidance and unwavering support throughout the duration of this project. We also extend our sincere appreciation to Mr. S. Mallesh, Assistant Professor and Head of the Department of Electronics and Communication Engineering at CMR Technical Campus, and Dr. A. Raji Reddy, Director of CMR Technical Campus, for their generous provision of essential facilities and resources. Their collective support and mentorship have been instrumental in the successful completion of this work.

References

[1] J. Klett and D. Buelow, "Watering a home landscape during drought," *Colorado State University Extension*, 2013. [Online]. Available: https://extension.colostate.edu/topic-areas/yard-garden/watering-a-home-landscape-during-drought-7-240-2/. Accessed: Dec. 21, 2024.

[2] S. Datta, S. Taghvaeian, and J. Stivers, "Understanding soil water content and thresholds for irrigation management," *Oklahoma Cooperative Extension Service*, BAE-1537, June 2017. [Online]. Available: https://extension.okstate.edu/fact-sheets/understanding-soil-water-content-and-thresholds-for-irrigation-management.html. Accessed: Dec. 21, 2024.

[3] R. Johar, A. Bensenouci, and M.-A. Bensenouci, "IoT-based smart sprinkling system," in *Proc. of the 15th Learning and Technology Conf. (L&T), Jeddah, Saudi Arabia,* February 2018, pp. 1–7, doi: 10.1109/LT.2018.8368499.

[4] *Guide to Meteorological Instruments and Methods of Observation*, WMO No.-8, 7th ed., World Meteorological Organization, 2008. [Online]. Available: https://www.weather.gov/media/epz/mesonet/CWOP-WMO8.pdf. Accessed: Dec. 21, 2024.

[5] Z. Zhang, Z. Pan, F. Pan, J. Zhang, G. Han, N. Huang, J. Wang, Y. Pan, Z. Wang, and R. Peng, "The change characteristics and interactions of soil moisture and temperature in the farmland in Wuchuan County, Inner Mongolia, China," *Atmosphere*, vol. 11, no. 5, pp. 10–11, May 2020, doi: 10.3390/atmos11050503.

[6] Tamil Nadu Agricultural University, "Relative humidity and plant growth," *Agritech Portal*, 2021. [Online]. Available: https://agritech.tnau.ac.in/agriculture/agri_agrometeorology_relativehumidity.html. Accessed: Dec. 21, 2024.

[7] M. Breuss, "Python web applications: Deploy your script as a Flask app," *Real Python*. [Online]. Available: https://realpython.com/python-web-applications. Accessed: Dec. 21, 2024.

[8] V. Veeraiah, N. B. Rajaboina, G. N. Rao, S. Ahamad, A. Gupta, and C. S. Suri, "Securing online web application for IoT management," in *Proc. of the 2nd Int. Conf. on Advance Computing and Innovative Technologies in Engineering,* April 2022, pp. 1499–1503, doi: 10.1109/ICACITE53722.2022.9823733.

[9] A. Morchid, S. Ayache, M. Mansouri, M. Saadi, and F. Bouabdallah, "Smart agriculture for sustainability: The implementation of smart irrigation using real-time embedded system technology," in *Proc. of the 4th Int. Conf. on Innovative Research in Applied Science, Engineering and Technology*, March 2024, pp. 1–6, doi: 10.1109/IRASET60544.2024.10548972.

[10] A. A. Ahmed, S. Al Omari, R. Awal, A. Fares, and M. Chouikha, "A distributed system for supporting smart irrigation using Internet of Things technology," *Engineering Reports*, vol. 3, no. 1, pp. 4–10, January 2021, doi: 10.1002/eng2.12352.

[11] M. R. H. Naeem, S. Gawhar, M. B. H. Adib, S. A. Sakib, A. Ahmed, and N. A. Chisty, "An IoT-based smart irrigation system," in Proc. of the 2nd Int. Conf. on Robotics, Electrical and Signal Processing Techniques, January 2021, pp. 244–246, doi: 10.1109/ICREST51555.2021.9331092.

[12] T. Rathore, D. K. Gupta, and N. Kumar, "Smart irrigation system using IoT," in *Proc. of the 2023 3rd Int. Conf. on Secure Cyber Computing and Communication*, March 2023, pp. 607–609, doi: 10.1109/ICSCCC58608.2023.10176528.

[13] K. L. Krishna, O. Silver, W. F. Malende, and K. Anuradha, "Internet of Things application for implementation of smart agriculture system," in *Proc. of the Int. Conf. on IoT in Social, Mobile, Analytics, and Cloud*, Aug. 2017, pp. 54–59, doi: 10.1109/I-SMAC.2017.8058236.

[14] R. K. Kodali and A. Sahu, "An IoT-based soil moisture monitoring on Losant platform," in *Proc. of the 2nd Int. Conf. on Contemporary Computing and Informatics*, December 2016, pp. 764–768, doi: 10.1109/IC3I.2016.7918063.

[15] I. Al-Bahadly and J. Thompson, "Garden watering system based on moisture sensing," in *Proc. of the 9th Int. Conf. on Sensing Technology*, December 2015, pp. 263–268, doi: 10.1109/ICSensT.2015.7438404.

[16] M. Oudah, A. Al-Naji, T. Y. Al-Janabi, D. S. Namaa, and J. Chahl, "Automatic irrigation system based on computer vision and an artificial intelligence technique using Raspberry Pi," *Automation*, vol. 5, no. 2, pp. 92–104, May 2024, doi: 10.3390/automation5020007.

[17] A. Blessy J and A. Kumar, "Smart irrigation system techniques using artificial intelligence and IoT," in *Proc. of the 3rd Int. Conf. on Intelligent Communication Technologies and Virtual Mobile Networks*, February 2021, pp. 1355–1358, doi: 10.1109/ICICV50876.2021.9388444.

[18] A. Srivastava and D. K. Das, "A comprehensive review on the application of Internet of Things (IoT) in smart agriculture," *Wireless Personal Communications*, vol. 121, pp. 5–26, August 2021, doi: 10.1007/s11277-021-08970-7.

[19] Amitabh S., "Automatic plant watering and soil moisture sensing," *Instructables*. [Online]. Available: https://www.instructables.com/Automatic-Plant-Watering-and-Soil-Moisture-Sensing/. Accessed: Dec. 21, 2024.

[20] Random Nerd Tutorials, "Installing ESP8266 board in Arduino IDE (Windows, Mac OS X, Linux)," *Random Nerd Tutorials*. [Online]. Available: https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide. Accessed: Dec. 21, 2024.