Real time/societal research Project Lab report on

"FLORASENSE (PLANT MONITORING SYSTEM)"

Submitted in CMR Institute of Technology, Hyderabad in partial fulfillment of the requirements for the award of the Laboratory of

REAL TIME SOCIETAL RESEARCH PROJECT

Of

II-B.Tech. II-Semester

In

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

K.GAYATHRI	**22R01A7337**
M.DHANUSH	**22R01A7342**
N.VINUTHNA	**22R01A7345**
G.SWAPNA	**23R05A7306**

Under the Guidance of

Mr.MANTESH PATIL

(Professor, CSE (AI & ML) Department)

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

(Approved by AICTE, Permanent Affiliation to JNTUH, Accredited by NBA and NAAC)

Kandlakoya(V),Medchal Road,Hyderabad

2023-2024

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad)

Kandlakoya(V), Medchal Road, Hyderabad

CERTIFICATE

This is to certify that a Major project entitled

"FLORASENSE(THE PLANT MONITORING SYSTEM)" Is

being Submitted by:

K.GAYATHRI	**(22R01A7337)**
M.DHANUSH	**(22R01A7342)**
N.VINUTHNA	**(22R01A7345)**
G.SWAPNA	**(23R05A7306)**

In partial fulfillment of the requirement for award of the Real Time Societal Research Project of II-B. Tech II-Semester in AIML towards a record of a bonafide work carried out under our guidance and supervision.

Signature of Guide	Signature of Project Coordinator	Signature of HOD
Mr.Mantesh Patil	**Dr.Y.Nagesh**	**Prof.P.Pavan Kumar**

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M. Janga Reddy**, Director, **Dr. G.Madhusudhana Rao**, Principal and **Prof. P. Pavan Kumar** Head of Department, Dept of Computer Science and Engineering(AI&ML),CMR Institute of Technology for their Inspiration and valuable guidance during entire duration.

We are extremely thankful to **Dr. Y. Nagesh**, Major project Coordinator and Internal Guide **Mr.Mantesh Patil**, Dept of Artificial Intelligence and Machine Learning (AI & ML), CMR Institute of Technology for their constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for every step towards success.

K.GAYATHRI (22R01A7337)
M.DHANUSH (22R01A7342)
N.VINUTHNA (22R01A7345)
G.SWAPNA (23R05A7306)

ABSTRACT

Our busy social lives have led to the neglect of homegrown plants, which often die from a lack of water and care. Recognizing the need to monitor plant health effectively, we developed Flora Sense—a device that senses soil moisture levels and displays a plant's condition on a webpage with messages like “I Am Happy” or “I Am Sad,” along with corresponding emojis. By providing farmers with precise soil insights, personalized crop recommendations, and early disease detection, Flora Sense enhances agricultural efficiency, reduces resource wastage, and promotes sustainable practices. Utilizing Python, Visual Studio Code, and Arduino technology, this innovative project offers a seamless and intuitive user experience, revolutionizing our interaction with nature and technology and pushing the boundaries of environmental conservation and technological innovation. The “FLORASENSE” (plant monitoring device) is not only convenient but also eco-friendly. By eliminating the need for a device that senses the moisture level or a device that continuously waters the plant in this product there is no need for continuous checking of plant health condition. Data displayed on a web page in real-time enables convenient remote monitoring No more worrying about plant condition with this we can automatically do so. As technology continues to shape the future of agriculture, this project stands as a beacon of progress, addressing challenges and fostering a more resilient and productivefarming landscape.

INDEX

LIST OF FIGURES

LIST OF SCREENSHOTS

1.INTRODUCTION

1.1 PROJECT INTRODUCTION:

"Empowering plant care with real-time insights, bridging technology and nature."

The "Plant Monitoring System Using Soil Moisture Sensor with Webpage Display" is a cutting-edge solution designed to revolutionize the management and care of plants. This innovative system combines the precision of soil moisture measurement presenting real-time data through a user-friendly webpage interface.

Traditional plant care often relies on periodic manual inspections, which can result in delayed detection of critical issues affecting plant health. This project addresses this challenge by creating an integrated system that continuously monitors key indicators of plant well-being: soil moisture levels and visual cues from plant imagery.

The system employs a soil moisture sensor embedded in the plant's soil to track moisture content. Simultaneously, These data streams are collected and processed by a central unit, enabling the creation of a dynamic webpage interface accessible to users.

Through this webpage, users gain access to real-time information about their plants' moisture levels of their condition. The graphical representations and intuitive interface empower users to make informed decisions regarding irrigation schedules, plant health interventions, and overall care practices, all remotely and in real time.

1.2 PROJECT OBJECTIVES

- Create a system to constantly check plant health indicators like soil moisture and visual appearance.
- Gather data from soil moisture sensors and a camera to track plant conditions.
- It should be easy to use.
- Display real-time plant data on a user-friendly webpage for easy access.

1.3 PROBLEM STATEMENT

Traditional plant care relies heavily on periodic manual inspections to assess soil moisture and plant health. This approach, though simple, often leads to delayed detection of critical issues such as under-watering or over-watering, which can adversely affect plant growth and vitality. Manual inspections are not only labour-intensive but also prone to human error and oversight. The lack of timely and accurate data can result in inadequate plant care, leading to suboptimal growth conditions and potential plant loss.

2.SYSTEM ANALYSIS

2.1 EXISTING SYSTEM:

- **Soil Moisture Sensor**: When we normally keep a Soil Moisture Sensor in plant. It senses the Soil Content where we can say that If the Soil Content is less than 500, it is Wet. If the Soil Content is more than 500, It is Dry.
- **Device that automatically waters the plant:** There is a device that automatically waters the plant. Using a Microcontroller, We can control the plant soil content and water the plant.

2.1.1 DISADVANTAGES:

- Automatic systems may overwater plants if not correctly calibrated or if they fail to accurately assess soil moisture levels, leading to root rot and other water-related plant diseases.
- Automatic watering systems may cause damage to plants or the setup if they malfunction, such as continuous watering during sensor failure or blockages, potentially leading to water wastage or plant stress.
- Even though the soil moisture sensor gives the output immediately, Owner cannot go and check every time.

2.2 PROPOSED SYSTEM:

A Device that Senses plant needs and displays on a Webpage that “I Am Happy”, and “I am sad”, with an Emoji. It Senses the Soil Moisture Range of plants.

The web application will provide real-time updates on various parameters crucial for plant health, such as soil moisture level, ambient temperature, and light exposure. Users can access this information anytime, anywhere.

Users can set personalized alerts based on predefined thresholds for each parameter. For example, they can receive notifications when the soil moisture level drops below a certain point, indicating the need for watering.

The application will feature an interactive dashboard that displays the status of all monitored plants in a user-friendly interface. Users can easily navigate between different plants and view historical data trends.

The web application will store historical data for each plant, allowing users to track changes in plant status over time. This data can also be used for analysis and insights into optimal plant care practices.

Our design includes a very simple mechanism. Firstly, the Soil Moisture Sensor senses the Soil Content. A web page is created using VS Code, IDLE(Python), and Command Prompt. These all are Controlled by a Device called a Microcontroller (named Arduino UNO). The values Sensed by the Soil Moisture Sensor are Displayed in the Created Webpage.

The Plant Status Indicator Web Application represents an innovative solution to help users effectively monitor and care for their plants. By leveraging the power of web technologies, real-time data, and personalized alerts, we aim to empower plant enthusiasts with the tools they need to ensure the health and vitality of their green companions.

2.2.1 ADVANTAGES:

- **Real-Time Monitoring :** Provides immediate updates on soil moisture levels and other environmental parameters, allowing users to respond promptly to their plants' needs.
- **User-Friendly Indicators** : Utilizes intuitive messages like “I Am Happy” or “I Am Sad” with corresponding emojis to indicate the plant’s status, making it easy for users to understand their plants' needs at a glance without interpreting complex data.
- **Web-Based Access** : Offers a web application that can be accessed from any internet-enabled device, providing users with the flexibility to monitor their plants remotely from anywhere at any time.

2.3 LITERATURE SURVEY :

The "Florasense" project aims to investigate the relationship between soil moisture levels and plant health by leveraging plant fluorescence as an indicator, using Arduino-based systems for real-time monitoring. Soil moisture is a crucial factor influencing plant health, with optimal levels necessary for nutrient uptake, photosynthesis, and growth. Water stress, whether from drought or waterlogging, adversely affects plants, leading to physiological changes detectable through fluorescence, particularly chlorophyll fluorescence, which indicates photosynthetic efficiency. Previous versions of the project utilized traditional soil moisture and fluorescence measurement techniques, which, while effective, were often limited by higher costs and complexity. The current Arduino-based version offers a cost-effective and versatile solution, integrating soil moisture sensors and fluorescence sensors to provide continuous, real-time data. This integration enables precision agriculture by optimizing irrigation schedules, offering early stress detection, and supporting research into drought-resistant plant varieties and improved water management practices. Significant advancements have been made with Arduino systems in real-time monitoring and stress detection protocols. However, challenges remain in sensor accuracy, data integration, and scalability. Future research should enhance sensor technology, refine data analysis methods, and explore comprehensive plant health assessments.

3. REQUIREMENT SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS

- **Processor** : Intel CORE i5
- **RAM** : 16 GB
- **Hard Disk** : 512 GB
- **Key Board** : Standard Windows Keyboard
- **Mouse** : Two or Three Button Mouse
- **Monitor** : SVGA

3.2 SOFTWARE REQUIREMENTS

- **Operating System** : Windows 10
- **Coding Language** : Python
- **Front-end** : Python
- **Designing** : HTML,CSS

3.1. 1ARDUINO UNO

FIG.1 ARDUINO UNO

The Arduino Uno is a popular microcontroller board renowned for its versatility and

ease of use in the realm of electronics and prototyping. It is a foundation for countless projects, offering a wide range of inputs and outputs, and facilitating connections to sensors, motors, and various components. Its user-friendly interface, powered by a simplified integrated development environment (IDE), allows beginners and experts alike to write and upload code effortlessly.

The Uno, based on the ATmega328P microcontroller, boasts multiple digital and analog pins, making it adaptable for diverse applications, from basic LED blinking experiments to complex IoT (Internet of Things) projects. Its open-source nature encourages a vast community of developers and enthusiasts to share projects, codes, and resources, fostering continuous innovation and learning in the world of electronics and programming.

It has 2 KB of SRAM (Static Random-Access Memory), used for storing variables and runtime data. The operating temperature range for the Arduino Uno is typically between -40°C to 85°C.

3.1.2 SOIL MOISTURE SENSOR

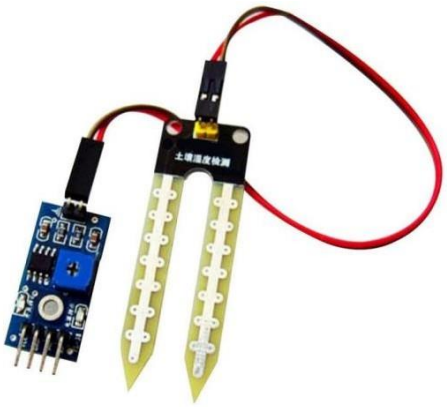


FIG.2 SOIL MOISTURE SENSOR

A soil moisture sensor is an integral component of modern plant monitoring systems, designed to measure the water content in soil. This sensor utilizes probes inserted into the soil to detect the moisture level, providing crucial insights into the plant's hydration needs. Operating on principles like conductivity or capacitance, these sensors translate soil moisture into electrical signals, which are then processed and interpreted by a microcontroller or similar unit. By continuously monitoring moisture levels, these sensors enable users to gauge the plant's

hydration status in real-time. This data is pivotal in determining the optimal watering schedule, preventing overwatering or underwatering scenarios, and ultimately promoting healthier plant growth. Integrating this sensor into the system facilitates proactive and precise plant care, aligning watering practices with the plant's actual needs, thus contributing significantly to efficient and sustainable cultivation.

3.1.3 FTDI MODULE

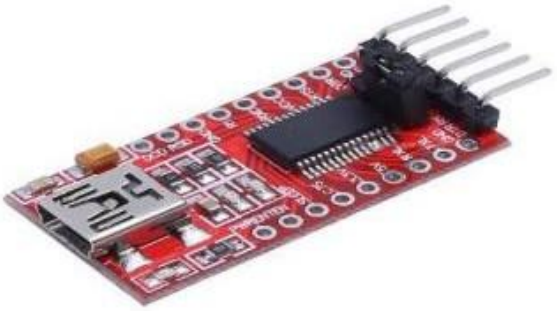


FIG.3 FTDI MODULE

The FTDI module is a crucial component in electronics, facilitating communication between devices like microcontrollers and computers. Known for its USB-to-serial conversion capabilities, the FTDI module acts as a bridge, enabling seamless data transfer between these devices. Its versatility lies in supporting various protocols, making it widely used in programming, debugging, and interfacing tasks.

This module simplifies the complexity of communication protocols, allowing developers to focus on their applications' functionalities without dealing with intricate hardware-level intricacies. Its reliability, ease of integration, and compatibility across different platforms make the FTDI module a go-to choice in numerous electronic projects requiring efficient and stable data exchange between devices.

3.1.4 JUMPER WIRES

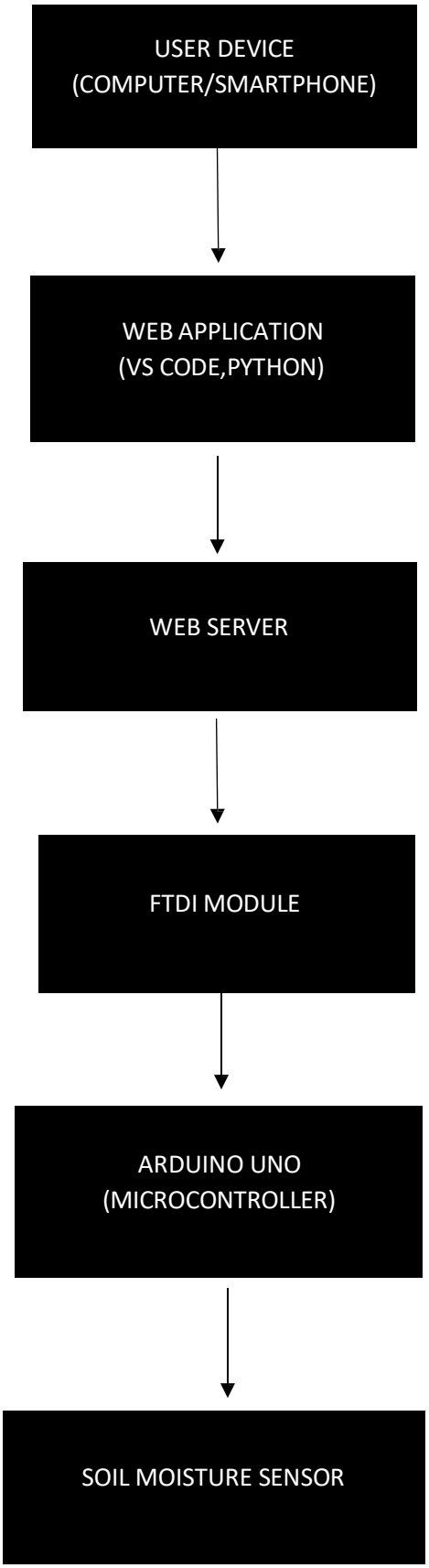
FIG.4 JUMPER WIRES

Jumper wires play a pivotal role in electronics by serving as connectors between various components on a breadboard or within a circuit. These wires, typically made of insulated conductive material, facilitate the flow of electricity between different points on a circuit board. Their flexibility and ease of use make them invaluable for prototyping and experimenting, allowing rapid adjustments or reconfigurations in electronic designs.

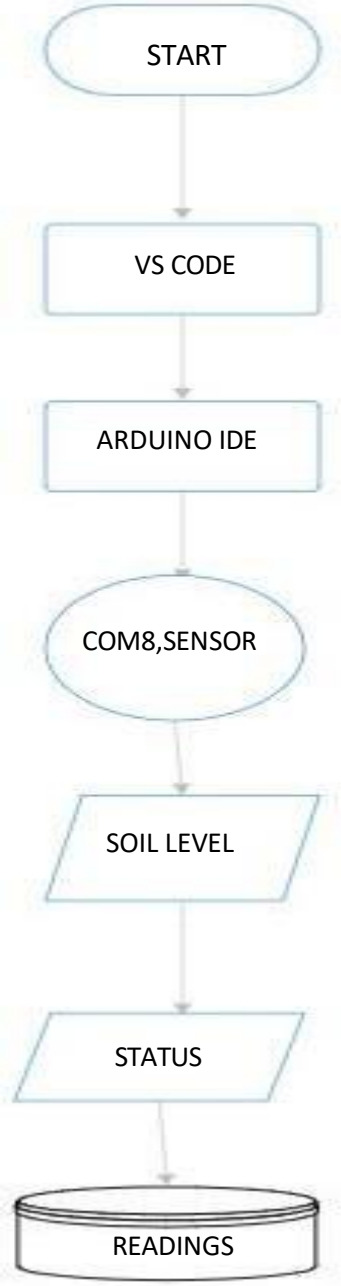
Jumper wires come in various lengths, colors, and connection types, enabling precise and organized connections between components such as sensors, microcontrollers, LEDs, and more.

4.SYSTEM ARCHITECTURE

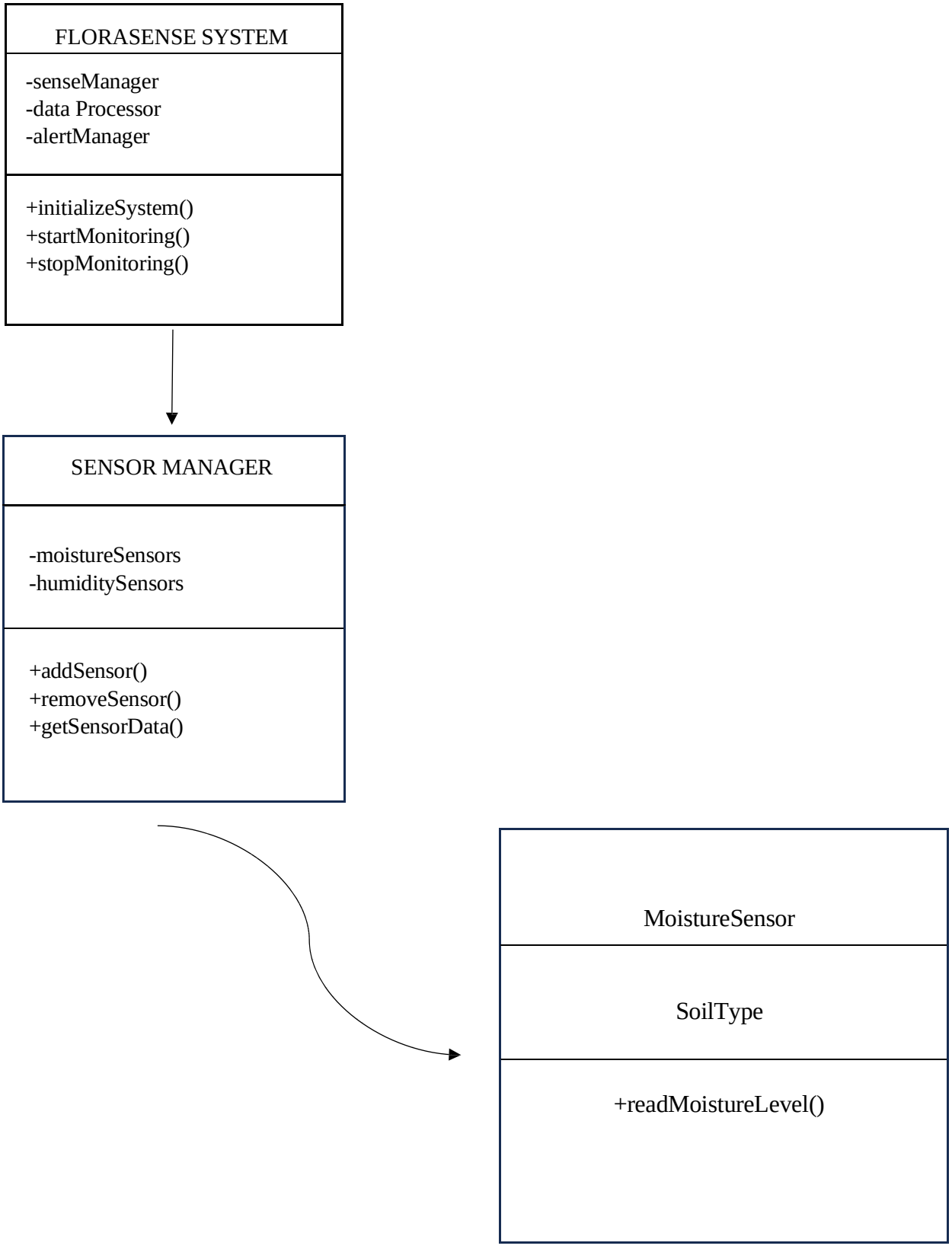
4.1 ARCHITECTURE :



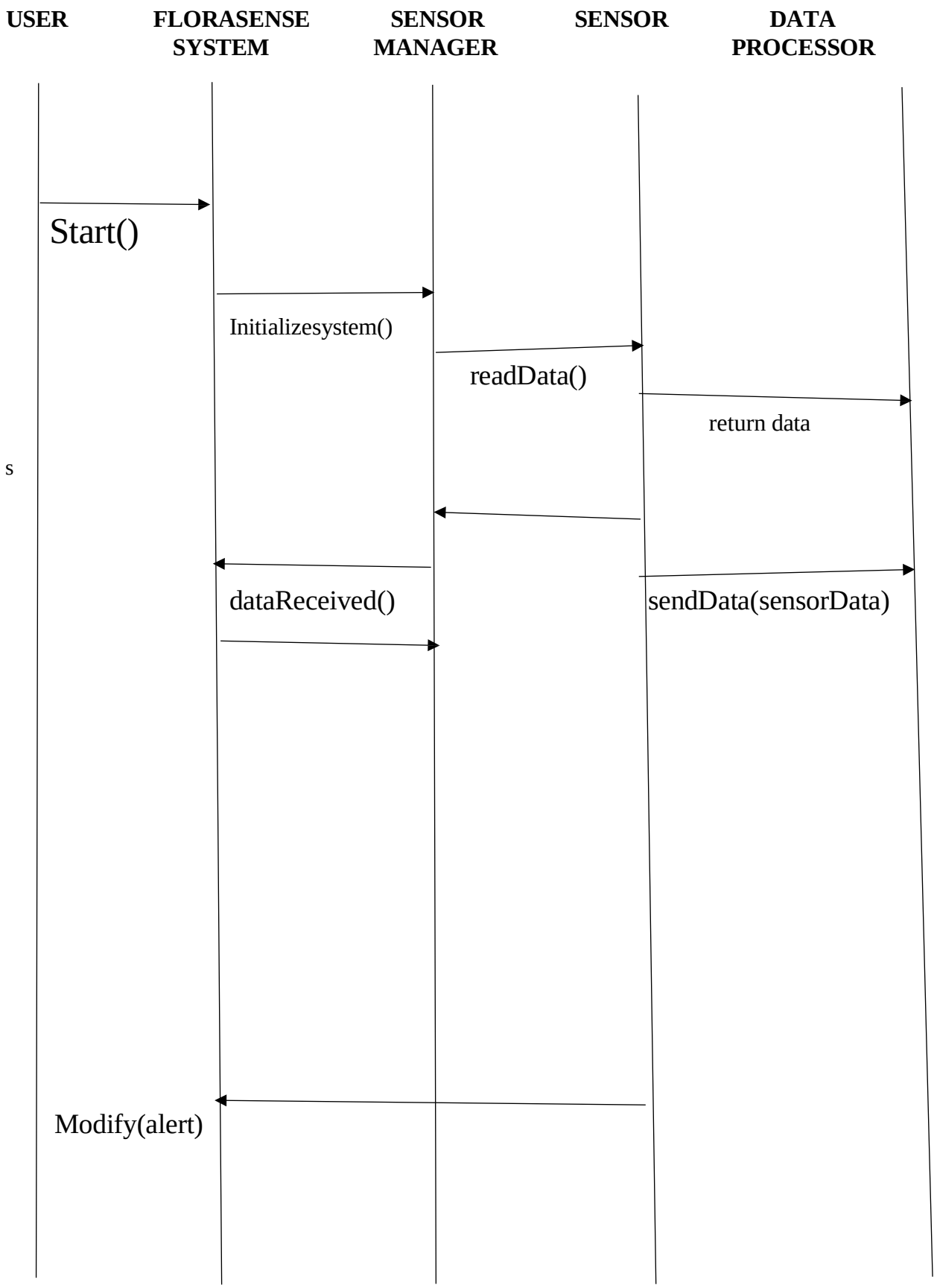
4.2 FLOW DIAGRAM :



4.3 CLASS DIAGRAM :



4.4 SEQUENCE DIAGRAM :



5.IMPLEMENTATION

5.1 MODULES

- User
- Automated Control
- Web Application
- Accuracy and Calibration

5.2 MODULES DESCRIPTION

5.2.1 User

User module is a fundamental component that manages user interactions and data within the system. It includes features for user authentication, access control, and profile management. The module ensures secure and personalized user experiences by handling identity verification, preferences, and account settings. It plays a pivotal role in creating a seamless interface for users to engage with the project, maintaining user-specific data integrity, and supporting personalized interactions. The user module is essential for establishing a structured framework that enhances overall system usability, security, and user satisfaction.

5.2.2 Automated Control

Integrating Actuators to Automate Watering Based on Sensor Readings and Predefined Thresholds. Automating the process of watering plants can greatly enhance efficiency, conserve water, and ensure optimal plant growth. By integrating actuators with sensor readings and predefined thresholds, an automated control system can be designed to manage the watering process seamlessly. Here's an extended approach to implementing such a system

5.2.3 Web Application

Florasense is a web application designed to keep your plants healthy and thriving. It displays real-time data on soil moisture levels, ensuring you are always informed about your plants' needs. The intuitive dashboard provides a comprehensive status of each plant, alerting you when they need watering or other care. With detailed historical data and trend analysis, Florasense helps you optimize watering schedules and improve plant health. Easy-to-set alerts and notifications ensure you never miss a critical update, making plant care effortless and efficient.

5.2.4 Accuracy and Calibration

Ensuring accurate sensor readings is crucial for reliable data in an automated watering system. This involves properly calibrating each sensor to account for specific environmental conditions and sensor drift over time. Regular calibration checks should be scheduled to maintain sensor precision. Calibration can be achieved by comparing sensor outputs with known reference standards or using calibration solutions. It's essential to follow manufacturer guidelines for each sensor type. Additionally, implementing self-calibration routines and error-checking algorithms can enhance overall system reliability and data integrity. Proper sensor maintenance and recalibration ensure the system's long-term effectiveness and accuracy.

5.3 TOOLS IMPLEMENTED:

SQLAlchemy:

SQLAlchemy is a popular SQL toolkit and Object Relational Mapper. It is written in Python and gives full power and flexibility of SQL to an application developer. It is an open source and **cross-**platform software released under MIT license.

SQLAlchemy is famous for its object-relational mapper (ORM), using which, classes can be mapped to the database, thereby allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.

As size and performance of SQL databases start to matter, they behave less like object collections. On the other hand, as abstraction in object collections starts to matter, they behave less like tables and rows. SQLAlchemy aims to accommodate both of these principles.

For this reason, it has adopted the data mapper pattern (like Hibernate) rather than the active record pattern used by a number of other ORM**s**. Databases and SQL will be viewed in a different perspective using SQLAlchemy.

Michael Bayer is the original author of SQLAlchemy. Its initial version was released in February 2006. Latest version is numbered as 1.2.7, released as recently as in April 2018.

SQLAlchemy - Environment setup

Any version of Python higher than 2.7 is necessary to install SQLAlchemy. The easiest way to install is by using Python Package Manager, **pip**. This utility is bundled with standard distribution of Python.

- pip install sqlalchemy

Using the above command, we can download the **latest released version** of SQLAlchemy from python.org and install it to your system.
In case of anaconda distribution of Python, SQLAlchemy can be installed from **conda terminal** using the below command –

- conda install -c anaconda sqlalchemy

It is also possible to install SQLAlchemy from below source code –

- python setup.py install

SQLAlchemy is designed to operate with a DBAPI implementation built for a particular database. It uses dialect system to communicate with various types of DBAPI implementations and databases. All dialects require that an appropriate DBAPI driver is installed.

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.'

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.

6.TECHNOLOGIES

6.1 PYTHON INTRODUCTION:

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. Created by Guido van Rossum and first released in 1991, Python has since gained immense popularity and is widely used across various domains including web development, data science, artificial intelligence, scientific computing, and automation. One of Python's defining features is its clear and concise syntax, which makes it easy to learn and understand, even for beginners. Its emphasis on readability is evident in its use of indentation to define code blocks, eliminating the need for explicit block delimiters like braces. Python's extensive standard library provides a rich set of modules and packages for tasks ranging from file I/O and networking to web development and GUI programming. Additionally, Python's vibrant ecosystem is bolstered by a vast collection of third-party libraries and frameworks, such as NumPy, pandas, TensorFlow, Django, Flask, and scikit-learn, which further extend its capabilities. Python's versatility stems from its support for multiple programming paradigms, including procedural, object-oriented, and functional programming. This flexibility allows developers to choose the most suitable approach for their projects. The language's dynamic typing and automatic memory management through garbage collection contribute to rapid development cycles and increased productivity. Python's cross-platform compatibility ensures that code written on one platform can run seamlessly on others, including Windows, macOS, and various Unix-like operating systems. Its open-source nature encourages collaboration and community- driven development, leading to frequent updates and improvements. In recent years, Python's dominance in fields like data science and machine learning has surged, thanks to libraries like NumPy, pandas, scikit-learn, and TensorFlow, which provide robust tools for data manipulation,analysis,and Overall, Python's simplicity, versatility, and extensive ecosystem make it a preferred choice for developers across industries, driving innovation and empowering individuals and organizations to solve complex problems efficiently. One of Python's key strengths lies in its extensive standard library, which provides modules and packages for a wide range of tasks, from web development and networking to database management and GUI programming. Moreover, Python's vibrant ecosystem boasts a plethora of third-party libraries and frameworks tailored to diverse domains, including scientific computing, artificial intelligence, data analysis, and web application development.The language's versatility extends to its support for multiple programming paradigms, including procedural, object-oriented, and functional programming, enabling developers to adopt the most appropriate approach for their projects.

6.2 HTML (HyperText Markup Language) :

HTML, or HyperText Markup Language, stands as the cornerstone of the World Wide Web. Since its inception in 1991 by Tim Berners-Lee, HTML has revolutionized the way we interact with digital content. It is the standard markup language used to create and design web pages and applications. As a markup language, HTML's primary role is to structure content on the internet, making it accessible and comprehensible for web browsers and users alike. Unlike programming languages that focus on logic and data manipulation, HTML's strength lies in its ability to define the structure and layout of web content.

HTML uses a series of elements and tags, which are enclosed in angle brackets, to mark up text and other content. These elements can include headings (<h1> to <h6>), paragraphs (<p>), links (<a>), images (<img>), and many other types of content. Each element has a specific purpose and function, contributing to the overall structure and presentation of a web page. For instance, the <a> tag is used to create hyperlinks, allowing users to navigate from one page to another, thereby interlinking the vast web of documents on the internet. The <img> tag, on the other hand, enables the embedding of images, enhancing the visual appeal and informational value of web pages.

A typical HTML document starts with a <!DOCTYPE html> declaration, which tells the browser that the document is written in HTML5, the latest version of the HTML standard. Following this, the document is enclosed within <html> tags. Inside the <html> tag, the content is divided into two main sections: the <head> and the <body>. The <head> contains meta-information about the document, such as its title (<title>), character set (<meta charset="UTF-8">), and links to external resources like CSS stylesheets and JavaScript files. The <body> section houses the content that is visible on the web page, including text, images, and other multimedia elements.

HTML's versatility and simplicity have made it an enduring technology. It supports a wide range of elements that enable the embedding of multimedia, such as audio (<audio>) and video (<video>) tags, as well as interactive elements like forms (<form>) that allow user input. This has allowed developers to create rich, interactive web experiences that go beyond static text and images. The integration of HTML with other web technologies like CSS (Cascading Style Sheets) and JavaScript further enhances its capabilities. CSS is used to control the appearance and layout of HTML elements, enabling the creation of visually appealing and responsive web designs. JavaScript adds interactivity and dynamic behaviour, allowing for the creation of complex web applications that can respond to user actions and update content in real-time.

6.3 CSS (CASCADING STYLE SHEETS) :

Cascading Style Sheets (CSS) is an indispensable technology in modern web development, providing the essential tools for defining the visual presentation and layout of web pages. Introduced in the mid-1990s by Håkon Wium Lie and Bert Bos, CSS revolutionized web design by enabling the separation of content and presentation, a principle that has profoundly impacted the way websites are built and maintained. CSS's primary role is to enhance HTML by allowing developers to apply styles to HTML elements, thus transforming plain text and unstyled content into visually appealing web pages. By specifying how HTML elements should be rendered in browsers, CSS brings the design and layout capabilities necessary to create rich, interactive, and responsive user experiences.

CSS operates through a system of selectors, properties, and values. Selectors are used to target specific HTML elements, which can then be styled using various properties such as color, font, margin, and padding. For example, a CSS rule like p { color: blue; } applies a blue color to all paragraph text (<p>) in a document. The flexibility of CSS selectors allows for precise targeting of elements through classes, IDs, attribute selectors, and pseudo-classes. This flexibility makes CSS a powerful tool for applying consistent styles across large websites while maintaining the ability to override general styles with more specific ones, thanks to its cascading nature. This cascade, along with inheritance and specificity, enables a structured and logical approach to applying styles, where more specific rules take precedence over more general ones, and child elements can inherit styles from their parent elements.

One of the core strengths of CSS is its ability to manage layout. Traditional techniques like floats and positioning have given way to more modern layout models such as Flexbox and CSS Grid. Flexbox simplifies the design of complex layouts that are responsive and flexible, allowing items within a container to be dynamically adjusted according to the available space. CSS Grid provides a more advanced layout system that can handle both rows and columns, offering precise control over the placement of items on a grid. These layout models enable developers to design web pages that adapt seamlessly to different screen sizes and devices, crucial for creating responsive designs in an era dominated by mobile browsing.

CSS also supports a range of visual enhancements beyond layout. Properties such as border-radius, box-shadow, and background-image allow developers to create sophisticated visual effects and animations without relying on images or external graphics. CSS transitions and animations enable smooth changes to properties over time, such as animating the color of a button on hover or transitioning the position of an element when a user interacts with it. This capability enhances user experience by providing visual feedback and creating dynamic, engaging interactions.

7.SOURCE CODE

app.py

```
from flask import Flask, render_template
from flask_sqlalchemy import SQLAlchemy
import serial

app = Flask(_name_)
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///soil_moisture.db"
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
db = SQLAlchemy(app)

class Reading(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    moisture_value = db.Column(db.Integer)
    timestamp = db.Column(db.DateTime, default=db.func.current_timestamp())

    def _init_(self, moisture_value):
        self.moisture_value = moisture_value

# Function to read data from serial port
def read_serial_data():
    try:
```

```
        with serial.Serial('COM8', 9600) as arduino:
            sensor_value = arduino.readline().decode().strip()
        try:
            moisture_value = int(sensor_value)
        except ValueError:
            moisture_value = 0
        return moisture_value
    except serial.SerialException as e:
        print(f"Failed to establish serial connection: {e}")
        return None

@app.route('/')
def index():
    moisture_value = read_serial_data()

    if moisture_value is not None:
        if moisture_value < 200:
            moisture_status = "I Am Dead ☠"
        elif moisture_value < 500:
            moisture_status = "I Am Happy 😇"
        else:
            moisture_status = "I Am Sad 😢"
```

```
            reading = Reading(moisture_value)
            db.session.add(reading)
            db.session.commit()
        else:
            moisture_value = 0
            moisture_status = "Serial connection error"

    return render_template('index.html', status=moisture_status,
moisture_value=moisture_value)

@app.route('/readings')
def readings():
    readings = Reading.query.all()
    return render_template('readings.html', readings=readings)

if _name_ == '_main_':
    with app.app_context():
        db.create_all()  # Ensure the database and table are created
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Soil Moisture Status</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Rubik+Doodle+Shadow&display=swap
');

        body {
            font-family: "Rubik Doodle Shadow", sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
            background: url('../static/images/background-image.jpg') no-repeat center
center fixed;
            background-size: cover;
```

```
}

.container {
    text-align: center;
    background: rgba(0,0, 0, 0.6);
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 0 10px #fff(0, 0, 0, 0.1);
    color: aquamarine;
}
.container h1 {
    color: #fff;
    padding-bottom: 10px;
}
.status {
    text-decoration: dotted underline;
    font-size: 24px;
    color: #fff;
}

.moisture-reading {
    font-size: 18px;
    color: #fff;
```

```
        }

        .live-status {
            margin-top: 20px;
            font-size: 22px;
        }

        .button {
            display: inline-block;
            margin-top: 10px;
            padding: 10px 20px;
            text-decoration: none;
            background: rgb(114, 235, 116);
            color: #fff;
            border-radius: 6px;
            transition: 0.2s;
        }

        .button:hover {
            box-shadow: 0 0 0 4px #fff, 0 0 0 8px rgb(114, 235, 116);
        }
    </style>
</head>
```

```
<body>
    <div class="container">
        <h1>Soil Moisture Status:</h1>
        <p class="status">{{ status }}</p>
        <p class="moisture-reading">Moisture Reading: {{ moisture_value }}</p>

    </div>
</body>
</html>
```

readings.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Soil Moisture Readings</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            background: linear-gradient(to right, #a8e063, #56ab2f); /* Green gradient */
        }
        h1 {
            color: #ffffff;
            text-shadow: 1px 1px 2px #000000;
            text-align: center;
            text-decoration: wavy;
        }
        ul {
            list-style-type: decimalsss;
            padding: 0;
        }
        li {
            background: rgba(255, 255, 255, 0.8); /* Semi-transparent white */
```

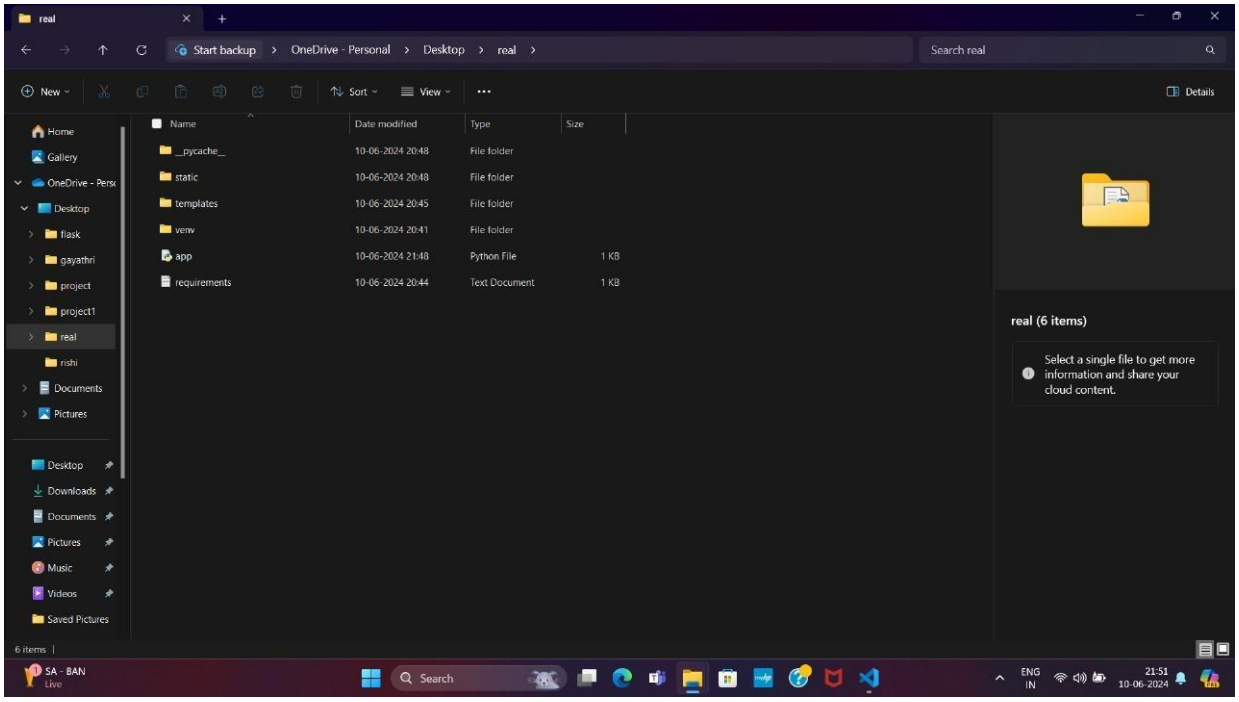
```
            margin: 5px;
            padding: 5px;
            border: 1px solid #ddd;
            border-radius: 4px;
        }
        .moisture-box {
            width: 5%; /* Decrease the width */
            margin: 10px auto;
            padding: 10px;
            background-color: #e6ffe6;
            border: 1px solid #ccc;
        }

    </style>
</head>
<body>
    <h1>Soil Moisture Readings</h1>
    <ul>
        {% for reading in readings %}
            <li>Moisture Value: {{ reading.moisture_value }} ({{ reading.timestamp }})</li>
        {% endfor %}
    </ul>
```

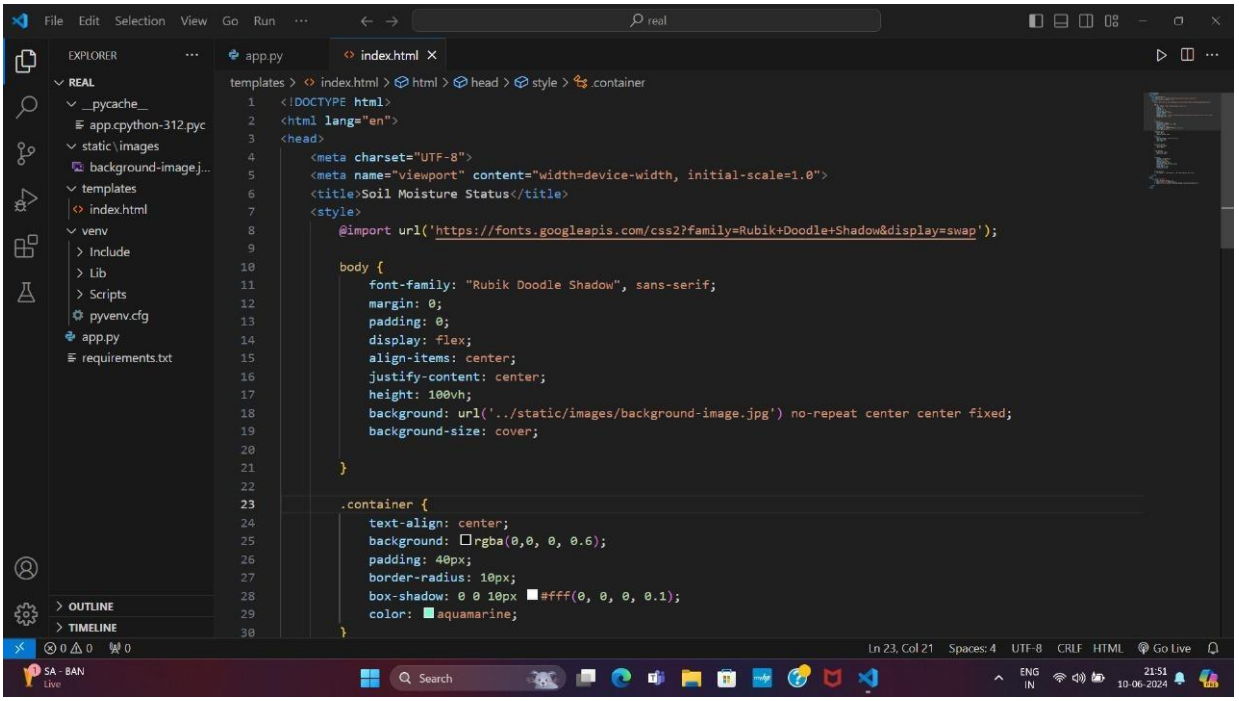
```
</body>    </html>
```

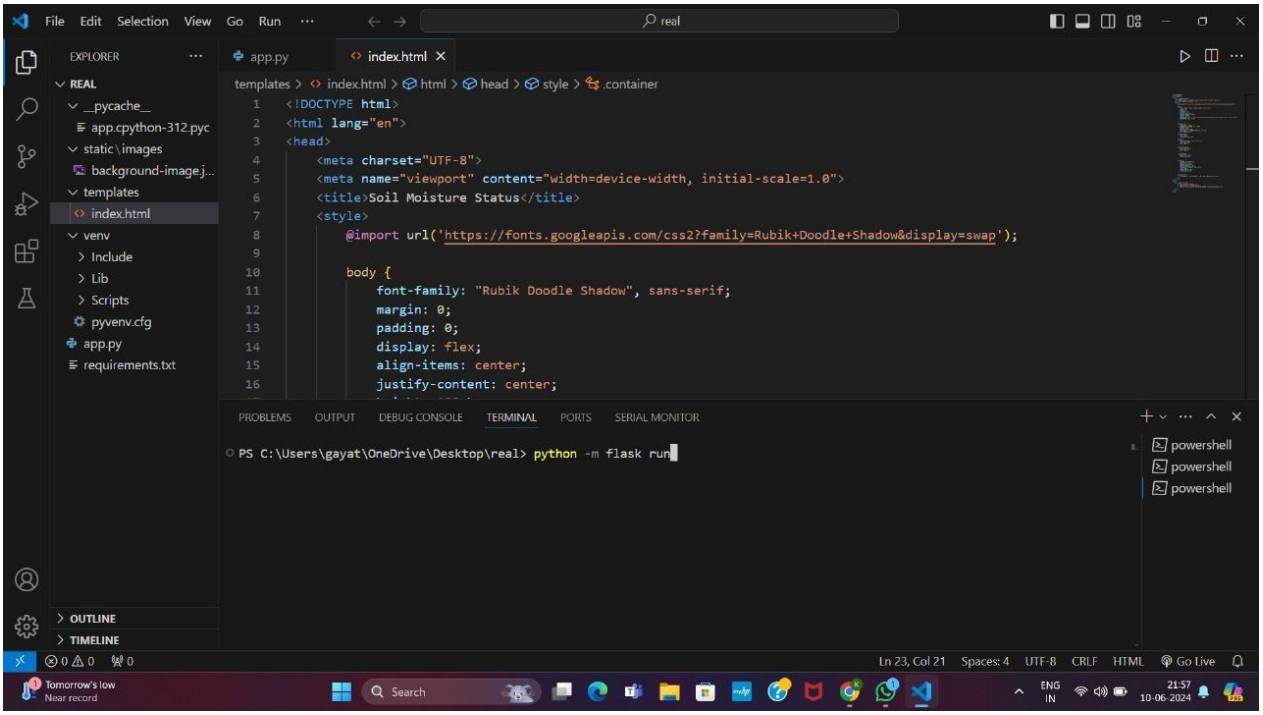
arduino.ino

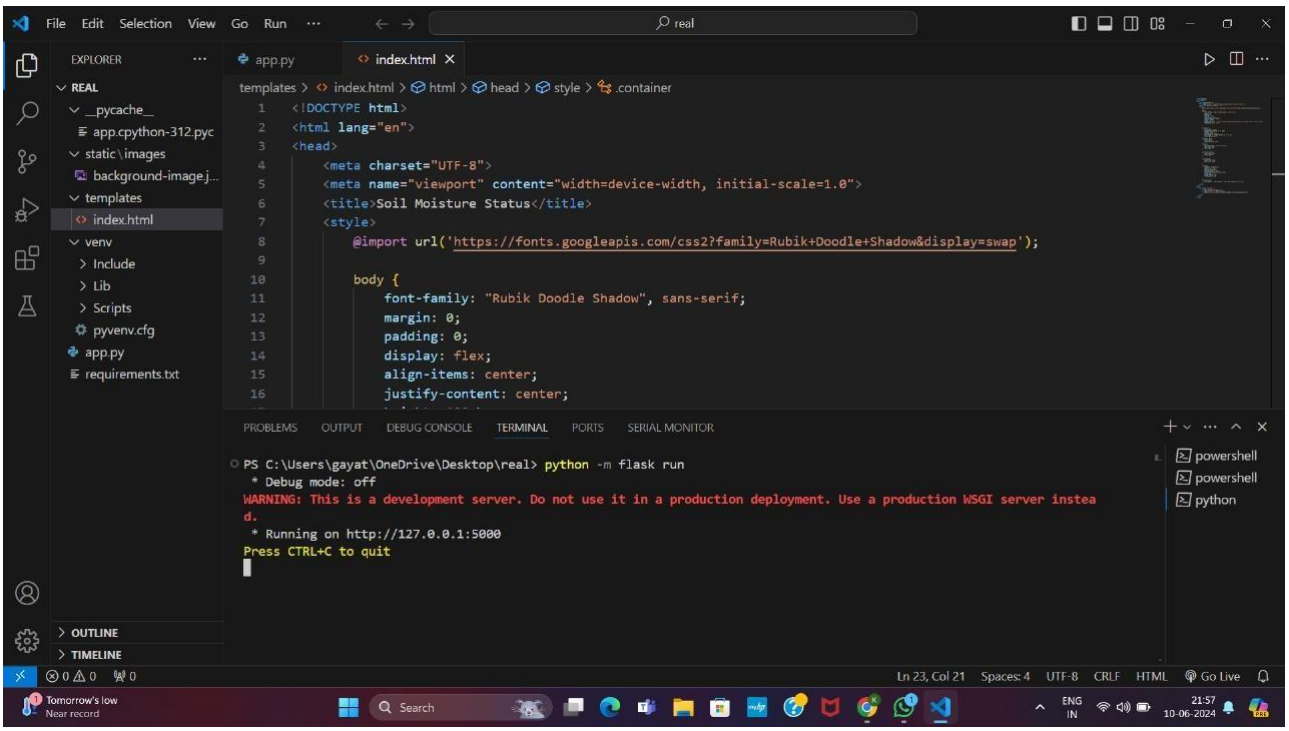
```
const int moistureSensorPin = A0; // Analog pin connected to the sensor

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(moistureSensorPin);
  Serial.println(sensorValue); // Send sensor value to serial monitor


  delay(1000); // Delay before next reading
}
```

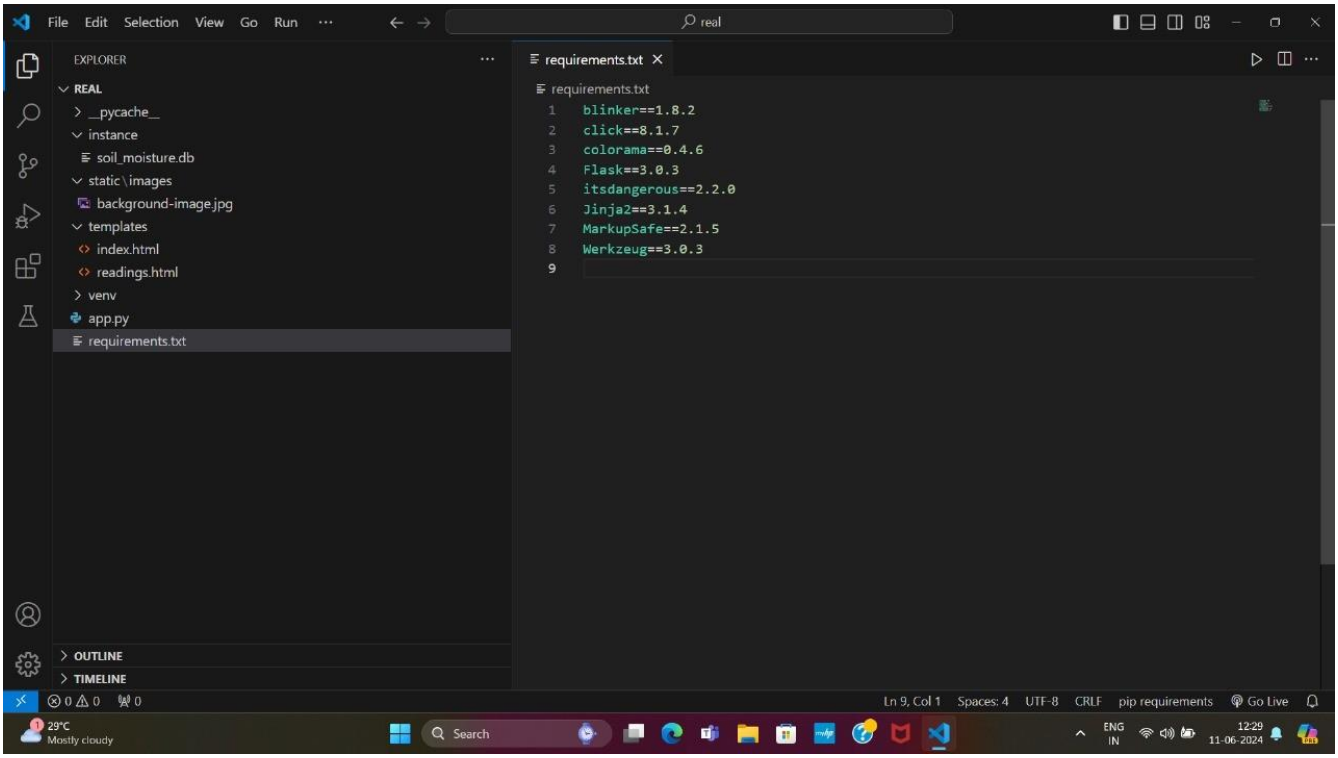
8.RESULTS

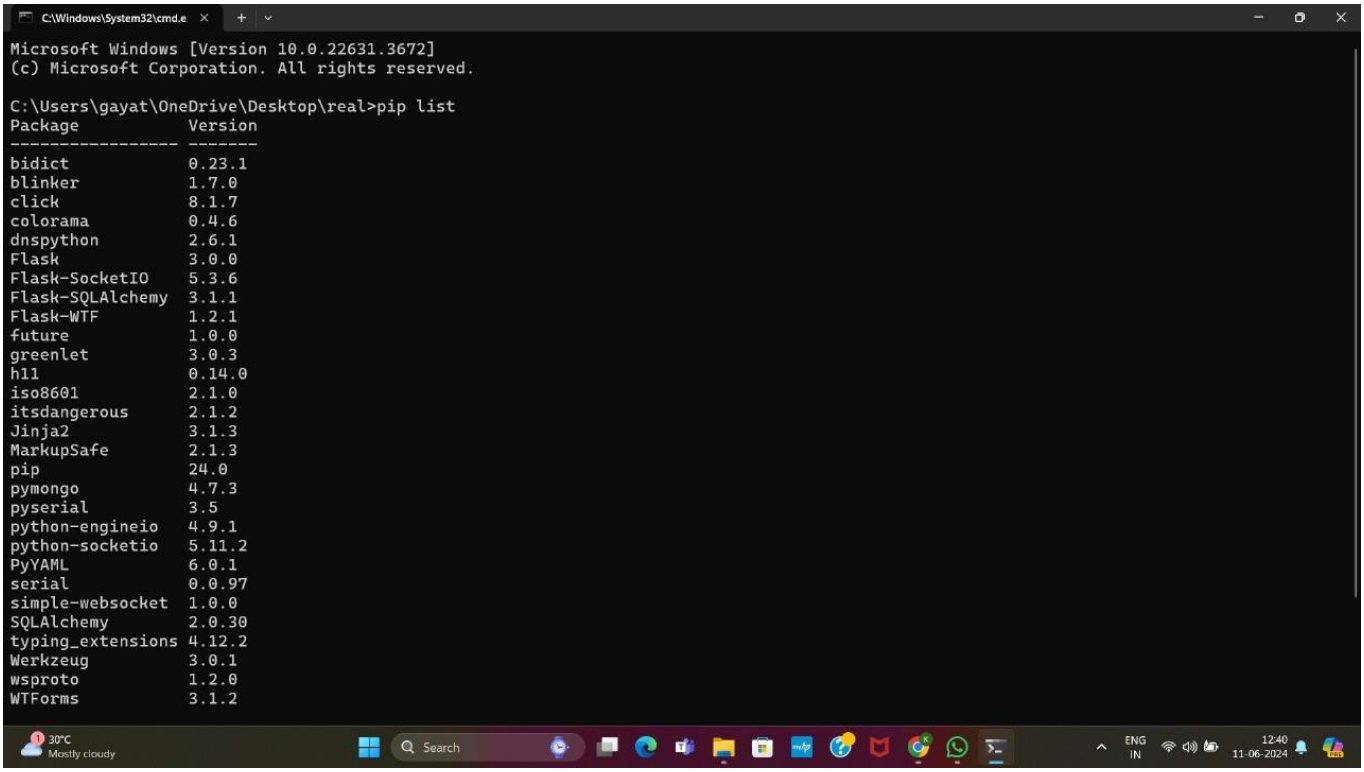

real
Start backup
OneDrive - Personal
Desktop
real
Search real
New
Sort
View
Details
Home
Gallery
OneDrive - Persc
Desktop
flask
gayathri
project
project1
real
rishi
Documents
Pictures
Desktop
Downloads
Documents
Pictures
Music
Videos
Saved Pictures
Name
Date modified
Type
Size
__pycache__ 10-06-2024 20:48 File folder
static 10-06-2024 20:48 File folder
templates 10-06-2024 20:45 File folder
venv 10-06-2024 20:41 File folder
app 10-06-2024 21:48 Python File 1 KB
requirements 10-06-2024 20:44 Text Document 1 KB
real (6 items)
Select a single file to get more information and share your cloud content.
6 items
SA - BAN
Live
Search
ENG
IN
21:51
10-06-2024

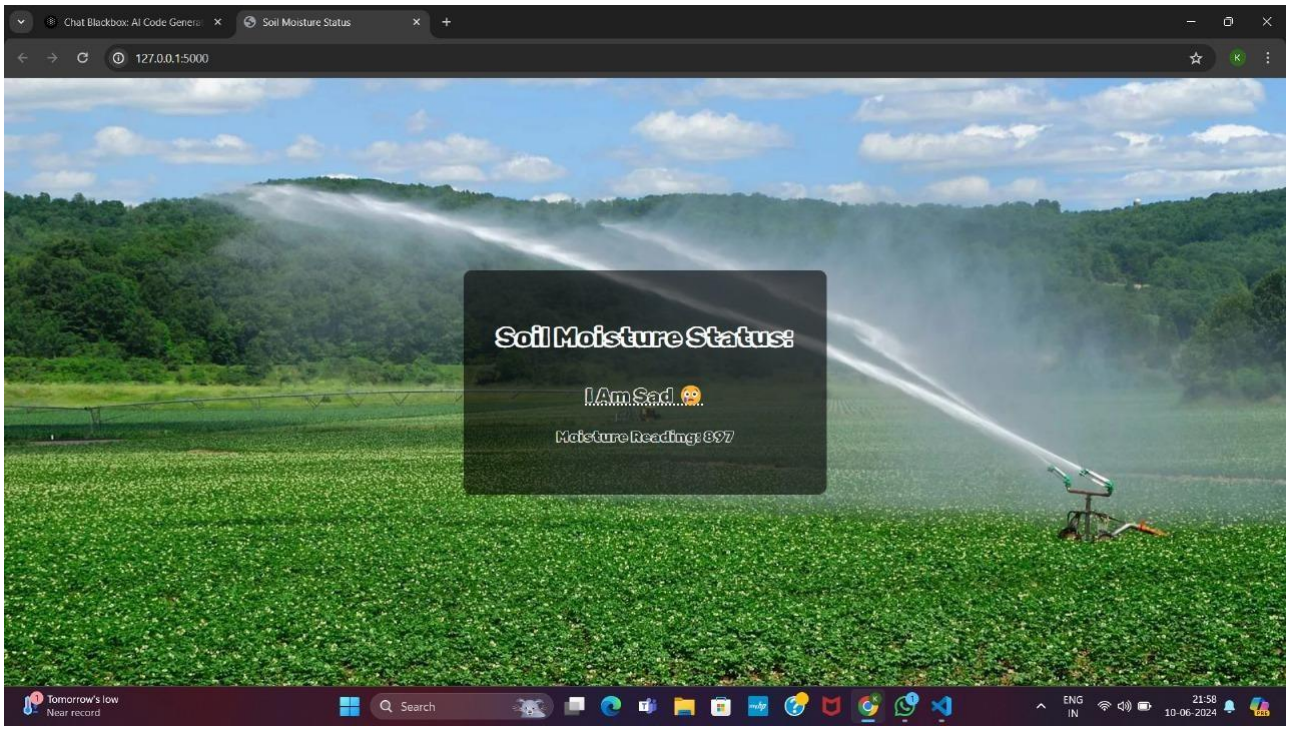

File Edit Selection View Go Run
real
EXPLORER
REAL
__pycache__
app.cpython-312.pyc
static\images
background-image.j...
templates
index.html
venv
Include
Lib
Scripts
pyvenv.cfg
app.py
requirements.txt
app.py
index.html
templates > index.html > html > head > style > .container
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Soil Moisture Status</title>
<style>
@import url('https://fonts.googleapis.com/css2?family=Rubik+Doodle+Shadow&display=swap');
body {
font-family: "Rubik Doodle Shadow", sans-serif;
margin: 0;
padding: 0;
display: flex;
align-items: center;
justify-content: center;
height: 100vh;
background: url('../static/images/background-image.jpg') no-repeat center center fixed;
background-size: cover;
}
.container {
text-align: center;
background: rgba(0,0, 0, 0.6);
padding: 40px;
border-radius: 10px;
box-shadow: 0 0 10px #fff(0, 0, 0, 0.1);
color: aquamarine;
}
OUTLINE
TIMELINE
Ln 23, Col 21
Spaces: 4
UTF-8
CRLF
HTML
Go Live
SA - BAN
Live
Search
ENG
IN
21:51
10-06-2024

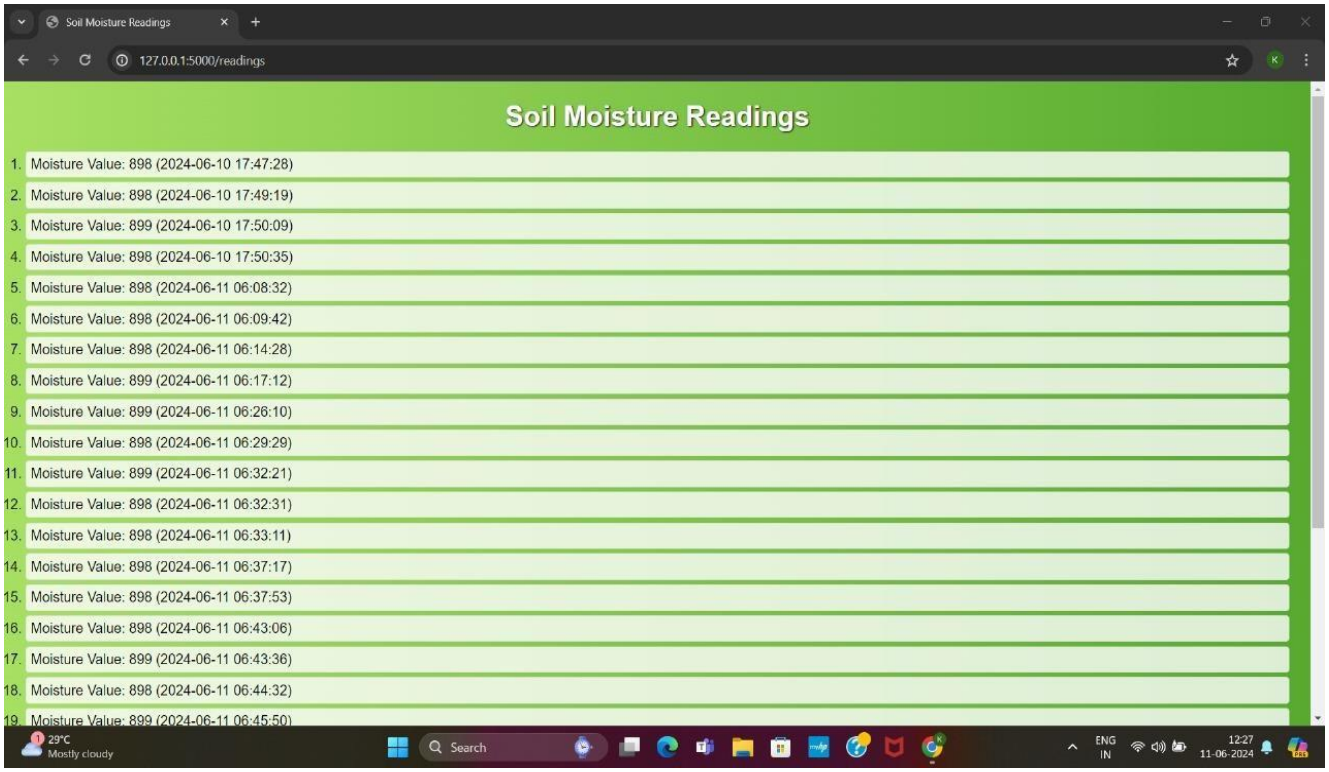

File Edit Selection View Go Run
real
EXPLORER
app.py
index.html
templates > index.html > html > head > style > .container
REAL
__pycache__
app.cpython-312.pyc
static\images
background-image.j...
templates
index.html
venv
Include
Lib
Scripts
pyvenv.cfg
app.py
requirements.txt
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Soil Moisture Status</title>
<style>
@import url('https://fonts.googleapis.com/css2?family=Rubik+Doodle+Shadow&display=swap');
body {
font-family: "Rubik Doodle Shadow", sans-serif;
margin: 0;
padding: 0;
display: flex;
align-items: center;
justify-content: center;
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR
PS C:\Users\gayat\OneDrive\Desktop\real> python -m flask run
powershell
powershell
powershell
OUTLINE
TIMELINE
Ln 23, Col 21 Spaces: 4 UTF-8 CRLF HTML Go Live
Tomorrow's low
Near record
Search
ENG
IN
21:57
10-06-2024


File Edit Selection View Go Run
real
EXPLORER
app.py
index.html
templates > index.html > html > head > style > .container
REAL
__pycache__
app.cpython-312.pyc
static\images
background-image.j...
templates
index.html
venv
Include
Lib
Scripts
pyvenv.cfg
app.py
requirements.txt
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Soil Moisture Status</title>
<style>
@import url('https://fonts.googleapis.com/css2?family=Rubik+Doodle+Shadow&display=swap');
body {
font-family: "Rubik Doodle Shadow", sans-serif;
margin: 0;
padding: 0;
display: flex;
align-items: center;
justify-content: center;
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR
PS C:\Users\gayat\OneDrive\Desktop\real> python -m flask run
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instea
d.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
powershell
powershell
python
OUTLINE
TIMELINE
Ln 23, Col 21 Spaces: 4 UTF-8 CRLF HTML Go Live
Tomorrow's low
Near record
Search
ENG
IN
21:57
10-06-2024


File Edit Selection View Go Run
real
EXPLORER
REAL
__pycache__
instance
soil_moisture.db
static \ images
background-image.jpg
templates
index.html
readings.html
venv
app.py
requirements.txt
requirements.txt
blinker==1.8.2
click==8.1.7
colorama==0.4.6
Flask==3.0.3
itsdangerous==2.2.0
Jinja2==3.1.4
MarkupSafe==2.1.5
Werkzeug==3.0.3
OUTLINE
TIMELINE
Ln 9, Col 1 Spaces: 4 UTF-8 CRLF pip requirements Go Live
29°C
Mostly cloudy
Search
ENG
IN
12:29
11-06-2024


C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.3672]
(c) Microsoft Corporation. All rights reserved.
C:\Users\gayat\OneDrive\Desktop\real>pip list
Package Version
bidict 0.23.1
blinker 1.7.0
click 8.1.7
colorama 0.4.6
dnspython 2.6.1
Flask 3.0.0
Flask-SocketIO 5.3.6
Flask-SQLAlchemy 3.1.1
Flask-WTF 1.2.1
future 1.0.0
greenlet 3.0.3
h11 0.14.0
iso8601 2.1.0
itsdangerous 2.1.2
Jinja2 3.1.3
MarkupSafe 2.1.3
pip 24.0
pymongo 4.7.3
pyserial 3.5
python-engineio 4.9.1
python-socketio 5.11.2
PyYAML 6.0.1
serial 0.0.97
simple-websocket 1.0.0
SQLAlchemy 2.0.30
typing_extensions 4.12.2
Werkzeug 3.0.1
wsproto 1.2.0
WTForms 3.1.2
30°C
Mostly cloudy
Search
ENG
IN
12:40
11-06-2024


Chat Blackbox: AI Code Genera
Soil Moisture Status
127.0.0.1:5000
Soil Moisture Status:
I Am Sad
Moisture Reading: 897
Tomorrow's low
Near record
Search
ENG
IN
21:58
10-06-2024


Soil Moisture Readings
127.0.0.1:5000/readings
Soil Moisture Readings
1. Moisture Value: 898 (2024-06-10 17:47:28)
2. Moisture Value: 898 (2024-06-10 17:49:19)
3. Moisture Value: 899 (2024-06-10 17:50:09)
4. Moisture Value: 898 (2024-06-10 17:50:35)
5. Moisture Value: 898 (2024-06-11 06:08:32)
6. Moisture Value: 898 (2024-06-11 06:09:42)
7. Moisture Value: 898 (2024-06-11 06:14:28)
8. Moisture Value: 899 (2024-06-11 06:17:12)
9. Moisture Value: 899 (2024-06-11 06:26:10)
10. Moisture Value: 898 (2024-06-11 06:29:29)
11. Moisture Value: 899 (2024-06-11 06:32:21)
12. Moisture Value: 898 (2024-06-11 06:32:31)
13. Moisture Value: 898 (2024-06-11 06:33:11)
14. Moisture Value: 898 (2024-06-11 06:37:17)
15. Moisture Value: 898 (2024-06-11 06:37:53)
16. Moisture Value: 898 (2024-06-11 06:43:06)
17. Moisture Value: 899 (2024-06-11 06:43:36)
18. Moisture Value: 898 (2024-06-11 06:44:32)
19. Moisture Value: 899 (2024-06-11 06:45:50)
29°C
Mostly cloudy
Search
ENG
IN
12:27
11-06-2024


MALABAR
GOLD & DIAMONDS
CELEBRATE THE BEAUTY OF LIFE

9.CONCLUSION

This project successfully integrates hardware and software components to create a comprehensive system for monitoring and displaying soil moisture and plant status in real-time. By combining the capabilities of an Arduino microcontroller with the versatility of Python programming, we have developed a functional and practical IoT application that demonstrates the seamless interaction between sensors, data processing, and user interface technologies. At the core of this system is the Arduino microcontroller, a popular choice for prototyping and implementing IoT solutions due to its ease of use and wide range of compatible sensors. Soil moisture sensors connected to the Arduino provide critical data on the moisture levels in the soil, which is essential for maintaining optimal plant health. The Arduino reads the analog or digital signals from these sensors, processes the data, and prepares it for transmission to a computer.

10.FUTURE ENHANCEMENT

1. Advanced Sensor Integration

1. **Nutrient Level Monitoring:**
    - **Integration:** Add sensors that can monitor soil nutrients such as nitrogen, phosphorus, and potassium levels.
    - **Benefit:** Provides more comprehensive insights into soil health, allowing for tailored fertilization and improved plant care.
2. **Environmental Sensors:**
    - **Integration:** Incorporate sensors for temperature, humidity, and light intensity.
    - **Benefit:** Allows monitoring of environmental conditions that affect plant growth, enabling better adjustment of plant care routines.

2. Enhanced Data Analytics and AI Integration

1. **Predictive Analytics:**
    - **Integration:** Implement machine learning models to predict watering needs based on historical data and weather forecasts.
    - **Benefit:** Optimizes watering schedules, reduces water usage, and prevents overwatering.
2. **Health Diagnosis:**
    - **Integration:** Use AI to analyse plant images for disease detection or pest identification.
    - **Benefit:** Early detection of issues can lead to quicker interventions, improving plant survival and health.
3. **Personalized Recommendations:**
    - **Integration:** Develop AI algorithms to provide customized care recommendations based on specific plant types and conditions.
    - **Benefit:** Offers users actionable insights tailored to their plants' needs.

3. Expanded Web Application Features

1. **Mobile Application:**
    - **Development:** Create a mobile app for iOS and Android for on-the-go monitoring.
    - **Benefit:** Provides users with flexibility to monitor and manage plant care from anywhere.
2. **Historical Data Visualization:**
    - **Feature:** Add graphical tools for visualizing historical trends in soil moisture and other metrics.
    - **Benefit:** Helps users understand long-term patterns and make informed decisions.
3. **User Customization:**
    - **Feature:** Allow users to customize alert thresholds, notification settings, and interface themes.
    - **Benefit:** Enhances user experience by providing more control over the application.

4. Automation and IoT Integration

1. **Automated Watering Systems:**
    - **Integration:** Connect with smart irrigation systems that can be controlled based on sensor data.
    - **Benefit:** Automates watering, ensuring plants receive the right amount of water without manual intervention.
2. **Integration with Home Assistants:**
    - **Feature:** Enable compatibility with home automation systems like Google Home or Amazon Alexa.
    - **Benefit:** Provides voice control and integration with broader smart home systems.
3. **Wireless Communication:**
    - **Integration:** Implement Wi-Fi or Bluetooth for wireless sensor data transmission.
    - **Benefit:** Simplifies setup and increases flexibility in sensor placement.

11.REFERENCES

https://www.google.com/url?sa=i&url=https%3A%2F%2Fin.rsdelivers.com%2Fproduct%2Fkitronik%2F4110%2F4110-200mm-jumper-wire-breadboard-jumper-wire-in%2F2048239&psig=AOvVaw1QnW2HNlX7Hz8cR9NLv6-R&ust=1704702463428000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCKixp9ntyoMDFQAAAAAdAAAAABAD

https://www.google.com/url?sa=i&url=https%3A%2F%2Fm.indiamart.com%2Fproddetail%2Farduino-uno-smd-21626891533.html&psig=AOvVaw0zBAZvb_u5vm2t6AacGNdg&ust=1704704229945000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCLiUiqL0yoMDFQAAAAAdAAAAABAI

https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6488907

https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=7361

https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=7755

https://www.w3schools.com/html/

https://playground.arduino.cc/Main/CapacitiveSensor/