Environment Monitoring for plants

Yupeng Chen:

ROLE(S)

Muhammad

Abdullah: ROLE(S)

DATE OF SUBMISSION: 11/12/2024

Executive Summary

Efficient environmental monitoring is essential for optimizing plant development, but typical techniques often fall short to adapt to dynamic environmental aspects liek temperature level, humidity, and light intensity. This job suggests a Smart Irrigation and Soil Monitoring System leveraging the Raspberry Pi 4 Version B, Sense HAT, and sensing units from the SunFounder Sensor Kit V2.0 to resolve this this constraint. The system gathers real-time environmental data and uses context-aware decision-making to automate watering based upon predetermined thresholds.

By imitating dirt problems via ecological criteria instead of relying only on direct dirt moisture readings, the system minimizes water wastage, boosts plant health and wellness, and offers a sustainable method to agriculture and horticulture. It features threshold-based AI formulas for decision-making and an instinctive web-based interface for real-time surveillance and hands-on watering control.

The recommended remedy lowers the dependency on inflexible watering timetables, resolving inadequacies in resource usage and guaranteeing plants get the ideal amount of water. This report details the system's style, execution, obstacles came across, and honest considerations, showcasing just how wise innovation can revolutionize resource management and lasting agriculture.

Table of Contents

Introduction

Background:

In agriculture and horticulture, achieving optimum plant development requires careful tracking of environmental factors such as temperature, humidity, and light intensity. These aspects significantly influence plant development and water demands. Typical irrigation systems often operate taken care of routines or fundamental timers, failing to adjust to real-time environmental modifications. This inflexibility can result in source waste, such as overwatering or underwatering, which not just hurts plant health yet also intensifies water scarcity problems in areas where conservation is crucial. The rise of clever innovations, consisting of IoT (Net of Things) and side computer, provides an opportunity to revolutionize irrigation systems by making them much more vibrant, context-aware, and resource-efficient (Phanindra et al. 2024).

Objectives:

This project intends to establish a Smart Watering and Soil Monitoring System that makes use of ecological data to automate and optimize watering processes. Unlike conventional systems, it does not rely only on straight dirt moisture analyses yet instead simulates soil conditions through the analysis of exterior ecological specifications (Yayan et al. 2022).

Using a Raspberry Pi 4 Version B, Feeling HAT, and sensors from the SunFounder Sensor Set V2.0, the system will collect real-time data on temperature, humidity, and light strength. These inputs will certainly educate context-aware formulas that decide when and how much to irrigate, enhancing water performance and sustaining lasting farming (Ozkan et al. 2021).

Extent:

The proposed system is designed for small gardening and farming usage, with potential scalability to larger applications. It incorporates hardware, software, and AI-driven decision-making to create a robust and adaptable remedy. Individuals can monitor real-time data and control watering setups with a web-based control panel. By automating irrigation processes, the system gets rid of inadequacies related to manual intervention, offering a more responsive and accurate technique to plant treatment (Buzura et al. 2022)

Relevance:

This task addresses important obstacles in traditional irrigation systems by incorporating modern-day side computing methods and IoT sensors to develop a sustainable and efficient solution. By reducing water waste and advertising much better source management, the system has the potential to dramatically improve farming efficiency while adding to worldwide water conservation efforts.

Additionally, the job acts as a model for incorporating wise innovations into daily applications, showcasing the transformative possibility of context-aware systems in resolving real-world difficulties (Mujiburrohman et al. 2023)

System Design

Overview of System Architecture

The Smart Irrigation and Dirt Monitoring System is designed around a dispersed architecture that integrates hardware, software program, and and context-aware decision-making to provide a receptive and sustainable irrigation solution. The central processing unit is a Raspberry Pi 4 Version B, which acts as the system's mind, interfacing with sensing units to gather real-time environmental data and make intelligent irrigation decisions (Helbet et al. 2021).

System Diagram:
The design contains five main parts:
1. Sensors: Environmental sensing units, consisting of a temperature level sensing unit, moisture sensing unit, and light sensing unit, accumulate real-time information to imitate soil problems. These sensors are attached to the Raspberry Pi through GPIO pins (Phanindra et al. 2024).

2. Processing Device: The Raspberry Pi 4 Model B processes sensing unit data making use of pre-configured threshold-based formulas, executing decision-making jobs locally via edge computer techniques. A relay module regulates a water pump that that provides water to plants when irrigation is caused based upon evaluated information. (Ozkan et al. 2021)

4. Interface: An online control panel, developed using Flask, permits users to keep track of environmental problems, sight watering logs, and manually readjust system settings(Buzura et al. 2022)

5. Power Supply: The entire system operates on a stable source of power to guarantee continual capability, with potential assimilation of a back-up battery for dependability. Environmental data is continually gathered by the sensing units and sent to the Raspberry Pi (Yayan et al. 2022).
The Raspberry Pi procedures this information locally using formulas that assess conditions like temperature level, moisture, and light intensity against preset limits. If the analysis indicates a need for watering, the Raspberry Pi triggers the relay module, which powers the water pump. The system logs ecological data and irrigation activities, which are presented on the user interface for real-time monitoring (Serhat et al. 2023).

Edge Handling and Context Recognition:
The system uses side processing to reduce latency and dependence on cloud-based

resources. By refining data locally on the Raspberry Pi, the system guarantees prompt reactions to transforming ecological problems. The unification of context-aware algorithms allows the system to make all natural decisions by incorporating several environmental factors, such as heats coupled with low humidity, to approximate optimal irrigation timing (Yayan et al. 2022).

System Reliability:

The style is creaeted for robustness, enabling the system to function separately also in low-connectivity atmospheres. By in your area keeping data and carryng out computations on the Raspberry Pi, the system makes certain integrity and continuity, even without net gain access to (Buzura et al. 2022)

Scalability and Future Enhancements:

The modular nature of the architecture makes it extremely scalable. Extra sensing units, such as soil pH or moisture sensing units, can be integrated to broaden the system's performance. Future improvements can consist of artificial intelligence designs to dynamically change limits based upon historical information, further enhancing watering effectiveness (Ozkan et al. 2021).

This style offers a comprehensive, effective, and easy to use remedy for handling irrigation systems in small agricultural or horticulture setups, promoting sustainability and resource conservation (Helbet et al. 2021)..

Sensors Integration

The Smart Irrigation and Soil Monitoring System relies on a network of sensing units to collect real-time ecological data important for making context-aware irrigation choices. These sensing units are meticulously picked for their ability to keep track of tempertaure level, moisture, and light strength, which are essential elements affecting plant delevopment and water requirements. The integration of these sensing units with the central processing unit, the Raspberry Pi 4 Design B, makes sure smooth data acquisition and handling (Mujiburrohman et al. 2023)

Listing of Sensors Utilized

1.Temperature level Sensor:

Steps ambient temperature level, which directly impacts evaporation prices and plant transpiration. High temperatures, when coupled with reduced humidity, can show the need for boosted watering frequency to avoid water tension on plants (Phanindra et al. 2024).

2.Humidity Sensing unit:

Functionality: Monitors air moisture degrees, a critical consider plant hydration and general health and wellness. High moisture decreases the demand for regular watering, assisting to conserve water sources by stopping unneeded watering cycles(Ozkan et al. 2021).

3.Light Sensing unit:
Steps the intensity of sunlight, which influences photosynthesis and soil drying out prices. Adjusts irrigation routines based on light exposure, making certain plants receive sufficient hydration during prolonged sunshine periods (Serhat et al. 2023).

All sensing units are incorporated with the Raspberry Pi 4 Version B using its GPIO (General Objective Input/Output) pins. The link process involves using jumper cables to make certain secure and exact data transmission. Each sensor undertakes a calibration process during system arrangement to guarantee exact information collection. Calibration regimens account for exterior variables such as environmental sound and sensor-specific mistakes, offering a standard for precise dimensions (Mujiburrohman et al. 2023).
Temperature Level Sensing Unit Calibration: Initial tests adjust the sensor's sensitivity to identify small adjustments in temperature level, guaranteeing precise readings in ever-changing problems. A controlled setting with predefined moisture degrees is used to verify sensor accuracy(Helbet et al. 2021).
Light Sensing Unit Calibration: Sensors are examined under numerous light intensities to figure out limits for low, medium, and high sunlight conditions (Ozkan et al. 2021).

Function in Context-Aware Decision Making
The sensing units collectively make it possible for the system to mimic soil problems indirectly by examining ecological specifications. As an example, a mix of heat, low humidity, and extreme sunshine can show the chance of completely dry soil, motivating watering. This approach minimizes dependency on direct dirt wetness readings, which can be affected by localized anomalies (Yayan et al. 2022).
Scalability and and Future Integration of the system's modular design allows for added sensors to be incorporated in the future, such as soil pH sensors, wetness sensors, or air pressure sensing units. These improvements could offer more granular data, boosting the accuracy of irrigation choices and increasing the system's applicability to varied farming environments (Mujiburrohman et al. 2023).
The Sensors Combination part guarantees that the system operates as a dependable, responsive, and efficient tool for wise watering, lowering water wastefulness and promoting lasting farming practices (Buzura et al. 2022)

AI and Edge Computing Techniques

The Smart Watering and Soil Keeping an eye on System incorporates AI formulas and side computer to enable real-time, context-aware decision-making for reliable watering monitoring. By refining ecological data in your area on a Raspberry Pi 4 Model B, the system ensures low latency, minimizes dependency on cloud services, and operates properly in low-connectivity settings. This section outlines the AI designs, side computer methods, and optimization methods employed in the system (Yayan et al. 2022).

AI Versions Applied
Helbet et al. (2021) A rule-based system contrasts sensor readings (e.g., temperature level, moisture, and light strength) versus predefined thresholds. Triggers irrigation

when ecological problems suggest that plants call for water. As an example, if the temperature surpasses 30 ° C and moisture goes down listed below 40%, the system triggers irrigation. Simple, computationally efficient, and very easy to carry out on resource-constrained side tools like the Raspberry Pi.

Context-Aware Algorithms these algorithms incorporate data from several sensing units to replicate indirect dirt conditions. For instance, high light strength integrated with reduced moisture signals raised soil drying, prompting watering even without straight dirt moisture readings (Buzura et al. 2022).

Supplies a more all natural understanding of ecological conditions to make certain exact irrigation choices. Improves precision by taking into consideration numerous variables, lowering the danger of over- or under-irrigation (Creț et al. 2021)

Side Processing on the Raspberry Pi
Helbet et al. (2021) talks about Side computing is main to the system's design, permitting information processing to take place in your area on the Raspberry Pi instead of depending on remote servers. This style uses several advantages:
1. Low Latency: Handling information locally guarantees immediate actions to altering ecological problems, important for applications like watering where hold-ups can influence plant wellness (Phanindra et al. 2024).
2. Bandwidth Efficiency: By minimizing the requirement to send raw data to the cloud, the system saves network data transfer and decreases functional prices.
3. Boosted Privacy and Protection: Delicate environmental information continues to be local, minimizing risks connected with information transmission and storage space in central servers (Nugroho et al. 2023).
The Raspberry Pi uses Python-based libraries, such as RPi.GPIO for hardware communciation and NumPy for data evaluation, to carry out edge handling. Real-time formulas review sensing unit inputs and make decisions within milliseconds, guaranteeing timely actuation of the irrigation system (Yayan et al. 2022).
Benefits of Combining AI and Edge Computing
Creț et al. 2021) Independence from Cloud Solutions: The system operates autonomously, making it ideal for remote areas with restricted internet connectivity.
Real-Time Decision-Making: By processing data in your area, the system makes certain prompt reactions to ecological modifications, essential for maximizing watering routines.
Lightweight AI designs and optimized algorithms reduce power intake, straightening with the system's sustainability objectives. The modular layout allows additional sensors or more advanced algorithms to be integrated without significantly influencing system performance (Ozkan et al. 2021).

Communication Between Instruments

Efficient communication in between tools is important for the smooth operation of the Smart Irrigation and Soil Keeping An Eye On System. The system utilizes multiple communication methods to allow smooth data excange in between sensing units,

the Raspberry Pi, and the user interface, making sure dependable and safe and secure transmission of information (Phanindra et al. 2024).

Networking Procedures
Wi-Fi Communication:

The Raspberry Pi 4 Design B uses its built-in Wi-Fi module to connect to a neighborhood network, helping with interaction between the system and the interface organized on an internet server. This connection allows customers to from another location monitor real-time environmental information and control irrigation settings via an internet control panel (Yayan et al. 2022).

GPIO Interaction:

Sensing units, including temperature level, humidity, and light sensing units, are directly linked to the Raspberry Pi with GPIO pins. Information is sent through electronic or analog signals, depending on the sensor type. Interaction between the Raspberry Pi and the relay module takes place via GPIO signals. The relay module analyzes these signals to control the water pump, enabling or disabling irrigation (Mujiburrohman et al. 2023)

Information Transmission

Helbet et al. (2021) Citizen Data Handling: Environmental data accumulated by sensors is refined locally on the Raspberry Pi, minimizing the need for constant information transmission to exterior web servers. The Flask-based internet dashboard gets updates by means of HTTP procedures, making it possible for customers to watch real-time information and adjust setups as required. Encryption protocols, such as HTTPS, are utilized to safeguard interaction in between the user's device and the web user interface, stopping unauthorized access to system controls or environmental data.

The communication structure of this system guarantees trustworthy, safe, and effective interactions between its components, creating the foundation of its real-time decision-making and easy to use user interface (Ozkan et al. 2021).

Individual Communication (UI/UX Style).

The Smart Watering and Dirt Monitoring System prioritizes an intuitive and obtainable individual experience by incorporating a properly designed user interface that allows individuals to interact flawlessly with the system. The online dashboard, developed using Flask, functions as the main touchpoint for users, supplying real-time information visualization and control capabilities. This section outlines the user interface layout, individual interaction techniques, and access features (Buzura et al. 2022)

User-Centered Layout Principles.
The UI/UX style follows these principles to make sure a positive individual experience. The interface reduces complexity, concentrating on the vital features

required for effective system operation. Uniform design components, such as font styles, colors, and switch designs, make sure a cohesive and expert look. Verification motivates and warnings help individuals prevent unexpected adjustments to essential settings (Mujiburrohman et al. 2023).

Future Enhancements.
To better enhance user communication, the complying with attributes could be included. Assimilation with voice assistants like Alexa or Google Assistant for hands-free procedure. AI-driven referrals for limit changes based upon historical information patterns. Interactive guides installed in the control panel to assist new users in navigating and setting up the system (Yayan et al. 2022).
The UI/UX style of this system bridges the gap between advanced watering innovation and user availability, guaranteeing that people with varied technical capabilities can effectively run and take advantage of the system (Mujiburrohman et al. 2023)

Stage 1: Equipment Configuration.
The primary step included putting together the system's hardware components, including the Raspberry Pi 4 Design B, Feeling HAT, and sensors from the SunFounder Sensing Unit Kit V2.0. Each sensing unit was connected to the Raspberry Pi by means of GPIO pins utilizing jumper cords (Nugroho et al. 2023).

Component Installment:

1. The temperature, humidity, and light sensing units were safely installed and attached to make certain steady data transmission. A relay module was mounted to manage the water pump, which would certaintly be activated based on refined sensor information (Buzura et al. 2022)

Sensor Calibration:

Each sensor was adjusted in controlled settings to make certain precision. For example, the moisture sensing unit was evaluated at different dampness levels, and its readings were adapted to decrease variances (Phanindra et al. 2024).

Power Administration:

A stable power supply was established to support constant operation. For prospective usage in remote areas, a backup battery system was explored to make sure reliability (Yayan et al. 2022).

Stage 2: Software Application Advancement.
Helbet et al. (2021) The software program was established using Python, leveraging its considerable collection community for hardware interaction and internet application growth. Python collections, such as RPi.GPIO, were utilized to check out data from the sensors and interact with the relay component. Real-time data

procurement was carried out to continually keep an eye on ecological specifications.

Decision-Making Formulas:

Threshold-based algorithms were developed to activate irrigation when environmental conditions satisfied preset standards, such as high temperature and low moisture. Context-aware algorithms incorporated information from multiple sensors to supply even more precise watering decisions (Yayan et al. 2022).

Web Dashboard:

1. A Flask-based internet application was created to enable users to keep track of real-time information, sight historical logs, and by hand manage the watering system. The control panel was maximized for mobile and desktop compatibility, guaranteeing a smooth customer experience across tools (Mujiburrohman et al. 2023)

Stage 3: System Combination.
After the hardware and software parts were independently established and tested, they were incorporated right into a natural system.

Checking and Debugging:

Initial screening entailed running the system in a regulated environment to confirm the functionality of sensing units, algorithms, and the web dashboard. Debugging concentrated on ensuring exact sensor analyses, reputable interaction in between parts, and smooth individual communication (Buzura et al. 2022).
Iterative Improvements:

Threshold values for irrigation were fine-tuned with iterative testing under differing environmental problems.The internet control panel user interface was fine-tuned based upon customer feedback to improve usability and accessibility. (Mujiburrohman et al. 2023)

Implementation and Final Testing:

The fully integrated system was released in a small-scale garden arrangement for last testing. This stage validated the system's capacity to operate autonomously, successully manage water use, and give precise environmental monitoring. (Nugroho et al. 2023).
Development Approach.
The growth followed a Nimble approach, damaging jobs into smaller iterations to make sure continual progress and versitlity. Regular screening and feedback loopholes permitted the identification and resolution of problems early in the process, resulting in a steady and reliable system. (Buzura et al. 2022).

The structured development process ensured that the Smart Watering and Dirt Surveillance System was built to meet its objectives of sustainability, reliability, and individual access. The phased technique facilitated effective fixing and repetitive enhancements, laying a solid foundation for future enhancements. (Creț et al. 2021).

Key Code Snippets and Algorithms.

In this section, we will certainly check out the key Python code bits and algorithms that were used in the Smart Watering and Soil Keeping Track Of System. The code serves multiple functions, including sensing unit data collection, threshold-based decision-making, and regulating the watering system with a relay component. Each code snippet is described carefully to demonstrate exactly how the system operates and how it ensures efficient and responsive irrigation (Mujiburrohman et al. 2023)

1. Sensor Data Collection.

The first vital function of the system is to accumulate real-time environmental data from the sensing units. The Raspberry Pi interfaces with the sensing units through its GPIO pins, and the adhering to Python code captures and refines the sensor information.

Code:

```
import RPi.GPIO as GPIO
import time
import Adafruit_DHT
import board
import busio
import adafruit_tcs34725

# Setup GPIO for relay
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

# DHT22 for Temperature and Humidity
DHT_SENSOR = Adafruit_DHT.DHT22
DHT_PIN = 4

# TCS34725 for Light sensor
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_tcs34725.TCS34725(i2c)

def read_sensors():
    # Read temperature and humidity from DHT22
    humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)

    # Read light intensity from TCS34725
    color = sensor.color
```

```
    light_intensity = (color[0] + color[1] + color[2]) / 3   # Average RGB for light intensity

    return temperature, humidity, light_intensity
```

Explanation:
GPIO Setup: The GPIO.setmode(GPIO.BCM) sets the GPIO pin numbering to the Broadcom (BCM) setting, and GPIO.setup(18, GPIO.OUT) sets GPIO pin 18 to manage the relay. The DHT22 sensor is used for gauging temperature level and moisture. It is attached to GPIO pin 4. The Adafruit_DHT. read(DHT_SENSOR, DHT_PIN) feature reads the sensing unit values. This sensing unit is used to gauge the light strength. It is connected with I2C communication. The RGB worths are fetched, and the average is computed to give a quote of the overall light strength.
This code collects temperature level, moisture, and light strength values, which are vital for the decision-making process. (Mujiburrohman et al. 2023)

Explanation:
Threshold Values: The variables TEMP_THRESHOLD, HUMIDITY_THRESHOLD, and LIGHT_THRESHOLD stand for the predefined conditions under which the system will trigger irrigation. For example, if the temperature exceeds 30 ° C and moisture is listed below 40%, watering is required .
Reasoning for Choice: The function check_irrigation() compares the real-time sensing unit values to these limits. If any of the problems suggest that irrigation is required (e.g., high temerature, reduced humidity, or high light intensity), the feature returns Real, signifying that irrigation needs to be triggered.

3. Triggering Irrigation (Relay Control).
When the choice to irrigate is made, the system turns on the water pump making use of a relay component. The relay is regulated through GPIO pins, and the adhering to code sets off the relay based upon the decision made by the AI formula.
Code:

```
def trigger_irrigation(irrigation_needed):
    if irrigation_needed:
        print("Irrigation triggered.")
        GPIO.output(18, GPIO.HIGH)    # Turn on water pump
        time.sleep(10)    # Water for 10 seconds
        GPIO.output(18, GPIO.LOW)      # Turn off water pump
    else:
        print("No irrigation needed.")
```

Explanation:
GPIO Control: The feature trigger_irrigation() uses GPIO pin 18 to regulate the relay component. When irrigation is needed (irrigation_needed == True), the code turns the relay on by establishing the GPIO pin to GPIO.HIGH (activating the water pump).
Water Pump Duration: The water pump competes 10 seconds, as defined by time.sleep(10), and then the relay is shut off with GPIO.LOW.
No Watering: If irrigation is not needed, the system publishes a message suggesting

that no action is taken.

4. Web-Based User Interface.
To enable individuals to communicate with the system, an internet control panel was created utilizing Flask, a Python web structure. This control panel provides customers with real-time data, system status, and handbook control options.
Code:

```
from flask import Flask, render_template

import threading

import paho.mqtt.client as mqtt

import os

from sense_hat import SenseHat

import requests



import time

import json

import queue



app = Flask(_name_)

sense = SenseHat()

mqtt_data_queue = queue.Queue()




# Weather API Configuration

API_KEY = "e4b29104f86ccc4035dc6810a5ac80fa"  # Get your free API key from OpenWeatherMap

CITY = "Carlow,IE"

API_URL = f"http://api.openweathermap.org/data/2.5/weather?q={CITY}&appid={API_KEY}&units=metric"
```

```
broker = "192.168.0.32"    # Replace with your Mosquitto broker's IP address
topic2 = "sensor/LightResistor"



def get_real_weather():
    """Fetch real-time temperature and humidity from OpenWeatherMap."""
    try:
        response = requests.get(API_URL)
        response.raise_for_status()    # Raise an exception for HTTP errors
        data = response.json()
        return {
            "temperature": data['main']['temp'],    # Current temperature in Celsius
            "humidity": data['main']['humidity']    # Current humidity in percentage
        }
    except Exception as e:
        print(f"Error fetching weather data: {e}")
        return None


def calibrate_readings():
    """Calibrate Sense HAT readings using real temperature and humidity."""
    # Read Sense HAT temperature and humidity
    sense_temperature = sense.get_temperature()
    sense_humidity = sense.get_humidity()
```

```
    # Fetch real weather data from the API
    real_weather = get_real_weather()
    if real_weather is None:
        print("Unable to fetch real weather data. Using raw Sense HAT readings.")
        return sense_temperature, sense_humidity

    # Calculate temperature offset and corrected value
    temperature_offset = sense_temperature - real_weather["temperature"]
    corrected_temperature = sense_temperature - temperature_offset

    # Calculate humidity offset and corrected value
    humidity_offset = sense_humidity - real_weather["humidity"]
    corrected_humidity = sense_humidity - humidity_offset

    return corrected_temperature, corrected_humidity

# Fetch and display corrected readings
corrected_temperature, corrected_humidity = calibrate_readings()


# Flask route for the web server
@app.route('/')
def index():
    try:
        #mqtt_data = mqtt_data_queue.get_nowait()
```

```
            #return f"Latest Temperature: {mqtt_data}"


            # Gather data from Sense HAT
            temperature = corrected_temperature
            humidity = corrected_humidity
            far_temperature = sense.get_temperature()
            far_humidity = sense.get_humidity()
            photoResistor=0
        # Pass the data to the HTML template
            sense.show_message(f"{temperature:.1f}C", scroll_speed=0.05) # sense temperature
            sense.show_message(f"{humidity: .2f}%")
            sense.show_message(f"{photoResistor: .2f} %")
            time.sleep(3)
            return render_template('index.html',
                                   temperature=round(temperature, 2),
                                   humidity=round(humidity, 2),
                                   ftemperature=round(far_temperature, 2),
                                   fphotoResistor=round(photoResistor,2),
                                   fhumidity=round(far_humidity, 2)    )
        except queue.Empty:
            # Gather data from Sense HAT
            temperature = corrected_temperature
            humidity = corrected_humidity
            far_temperature = sense.get_temperature()
            far_humidity = sense.get_humidity()
```

```
            photoResistor=0


           # Pass the data to the HTML template
            sense.show_message(f"{temperature:.1f}C", scroll_speed=0.05) # sense temperature
            sense.show_message(f"{humidity: .2f}%")
            sense.show_message(f"{photoResistor: .2f} %")
            time.sleep(3)
            return render_template('index.html',
                                   temperature=round(temperature, 2),
                                   humidity=round(humidity, 2),
                                   ftemperature=round(far_temperature, 2),
                                fphotoResistor=round(photoResistor,2),

                                   fhumidity=round(far_humidity, 2)    )



def on_message(client, userdata, message):
    mqtt_data = message.payload.decode()    # Decode byte data to string
    print(f"Received message: {mqtt_data} on topic {message.topic}")
    mqtt_data_queue.put(mqtt_data)    # Put the received message into the queue



# MQTT Subscriber function
def mqtt_subscriber():
    client = mqtt.Client()
```

```
    client.on_message = on_message    # Set the callback function

    # Connect to the broker
    client.connect(broker, 1883, 60)

    # Subscribe to the topic
    client.subscribe(topic)

    # Start the loop to process incoming messages
    client.loop_forever()

def mqtt_subscriber():
    client = mqtt.Client()
    client.on_message = on_message    # Set the callback function

    # Connect to the broker
    client.connect(broker, 1883, 60)

    # Subscribe to the topic
    client.subscribe(topic)

    # Start the loop to process incoming messages
    client.loop_forever()

# Function to start Flask server
```

```
def start_flask():
        app.run(debug=True, use_reloader=False, port=5001)
    # Set use_reloader=False to avoid conflicts with threading


if _name_ == "_main_":
        # Start Flask server in a separate thread
        flask_thread = threading.Thread(target=start_flask)
        flask_thread.start()


        # Start MQTT publisher in another thread
        mqtt_thread = threading.Thread(target=mqtt_publisher)
        mqtt_thread.start()
# Create MQTT client
#client = mqtt.Client()


# Assign callback
#client.on_message = on_message


# Connect to the broker
#client.connect(broker, 1883, 60)


# Subscribe to the topic
#client.subscribe(topic2)


# Start the MQTT loop
```

```
#print(f"Subscribed to topic {topic2}")

#client.loop_forever()
```

Explanation:

Flask Setup: The Flask web framework is booted up, and a path (@app. course('/')) is produced for the web page of the dashboard. The update_data() function constantly checks out sensing unit information, checks if irrigation is required, and causes irrigation when essential. This function runs in a separate thread to allow real-time updates without blocking the major program (Yayan et al. 2022)..

Making Control panel: The home() funtion provides the HTML web page (index.html) and displays the real-time worths of temperature, moisture, and light strength.

Threading: Since the sensor data requires to be continually upgraded, threading is used to run the information collection and irrigation process in the backgorund while maintaining the web server responsive (Ozkan et al. 2021).

5. Code Performance and Optimization.

To make sure the system runs smoothly on the Raspberry Pi, all algorithms were enhanced for side handling By refining the information locally on the Raspberry Pi and preventing constant data transmission to the cloud, the system conserves transmission capacity and makes certain low-latency decision-making. The use of straightforward threshold-based decision-making and marginal dependence on facility equipment finding out models makes certain the system remains receptive and effective, despite the limited computational resources of the Raspberry Pi (Nugroho et al. 2023).

Devices, Libraries, and Frameworks Made use of.

The Smart Watering and Dirt Checking System employs a range of tools, collections, and frameworks to make sure the system's functionality, performance, and scalability. These tools cover numerous elements of the job, including hardware combination, information processing, web growth, and edge computer. Below is a malfunction of the key elements made use of in the project (Yayan et al. 2022).

1. Setting Languages.

   Python:

   Python was picked as the main programs language because of its simpleness, readability, and vast ecosystem of libraries. It is particularly appropriate for tasks entailing hardware controll, sensing unit combination, and internet development. Python allows seamless integration with the Raspberry Pi 4 Version B, allowing straight control of the GPIO pins and reliable sensor information handling (Buzura et al. 2022).

   2. Collections and Structures

   .

   RPi.GPIO:

   RPi.GPIO is a library made use of to interface with the Raspberry Pi's GPIO (General Objective Input/Output) pins. This collection is necessary for

communicating with the sensors and relay module. It permits the system to review information from sensing units (such as the temperature, humidity, and light sensors) and control outside devices like the relay for irrigation (Aljelawy et al. 2022)

.

Functionality: Supplies functions for establishing input and result pins and controlling digital signals sent out to the relay module.

.

Flask:

Flask is a micro web framework for Python utilized to produce the web-based control panel for the system. It gives the interface whereby individuals can keep track of real-time environmental data and regulate the irrigation system. Flask is lightweight and suitable for constructing easy internet applications (Aljelawy et al. 2022)

.

Functionality: Takes care of HTTP demands, offers websites, and updates real-time data on the dashboard. It enables communication between the Raspberry Pi and the user's device, permitting remote monitoring and control of the irrigation system (Buzura et al. 2022).

NumPy:

NumPy is a collection for numerical computer in Python. It is utilized to do effective mathematical operations on the data collected from the sensing units. In this system, NumPy assists process light strength information from the TCS34725 sensing unit and calculate standards or change limits for decision-making. Supplies assistance for selection procedures and mathematical functions that maximize data dealing with and evaluatiom (Creț et al. 2021).

3. Equipment Elements

.

Raspberry Pi 4 Design B:

The Raspberry Pi 4 Version B serves as the central processing unit of the system. It integrates all parts, including sensors and actuators, and deals with real-time data handling and decision-making. The Raspberry Pi's GPIO pins are made use of to attach and regulate the sensing units and relay module. Serve as the primary controller, processing ecological data, making decisions, and communicating with the interface (Helbet et al. 2021)..

Feeling HAT:

The Sense HAT is an add-on board for the Raspberry Pi which contains a selection of sensing units, consisting of temperature level, humidity, and pressure sensing units. In this system, it is used to monitor environmental problems.

Functionality: Gives additional sensing units for keeping track of air pressure and temperature level, complementing various other sensing units to guarantee accurate environmental information collection(Buzura et al. 2022).

.

SunFounder Sensing Unit Package V2.0:

The SunFounder Sensor Package V2.0 has a variety of sensors that are useful in environmental monitoring. This set includes light sensors, temperature sensors, and relay modules, every one of which are necessary for the watering system's operation. Supplies the needed equipment to gather ecological information (light, temperature level, humidity) and control watering through the relay module (Helbet et al. 2021).

4. System Combination Devices

.

GitHub:

GitHub is utilized for variation control, collaboration, and sharing of the job's code. It ensures that all project members have accessibility to the most recent codebase and permits simple monitoring of adjustments and renovations. Helps with joint growth, ensures code stability, and tracks variation history. (Buzura et al. 2022).

Conclusion.

The tools, collections, and structures used in the Smart Watering and Dirt Keeping track of System are important to its functionality and efficiency. From Python and Flask for software program advancement to the Raspberry Pi and SunFounder Sensing Unit Kit V2.0 for hardware assimilation, each element plays an important function in producing a sustainable and straightforward system. By leveraging these technologies, the system has the ability to collect ecological data, make smart irrigation decisions, and provide customers with a responsive web user interface for tracking and control. These tools provide a solid foundation for the advancement of a scalable, reliable, and context-aware wise watering system (Aljelawy et al. 2022)

The growth and execution of the Smart Watering and Dirt Monitoring System ran into numerous obstacles, particularly in areas of sensing unit accuracy, computational constraints, and system optimization. Nonetheless, each obstacle was resolved with targeted services to ensure the system's reliability, efficiency, and scalability (Yayan et al. 2022).

Challenge:

Environmental sensing units, such as temperature level, humidity, and light sensing units, can in some cases give inaccurate information as a result of outside factors such as interference from weather, sensor placement, or perhaps aging of components. Incorrect sensor analyses might result in improper irrigation choices, either wasting water or falling short to deliver sufficient to the plants. To improve the accuracy of the sensing unit analyses, the following techniques were employed. Sensing units were adjusted before implementation to decrease mistakes. For instance, the moisture sensor was checked in different humidity problems to make certain that it provided trustworthy dimensions. Algorithms were implemented to smooth sensing unit information, straining spikes or fluctuations that are not rep of actual ecological conditions. In cases where accuracy was critical, repetitive sensing units (e.g., utilizing numerous light sensors) were added to validate readings and ensure uniformity. (Buzura et al. 2022).

Verdict and Reflection:

The Smart Watering and Soil Keeping an eye on System effectively addresses vital difficulties in traditional irrigation techniques by integrating environmental sensing units, edge computing, and context-aware decision-making algorithms. With real-time surveillance of temperature level, moisture, and light strength, the system maximizes watering, making sure that plants get the right amount of water while minimizing water waste. This technology not just improves plant health and wellness but additionally adds to lasting water usage, specifically in locations facing water shortage (Aljelawy et al. 2022).

Achievements:

Among the primary success of the task is the advancement of a wise, automated irrigation system that operates based on ecological problems instead of dealt with schedules. The integration of sensors such as the DHT22 (for temperature level and humidity), TCS34725 (for light intensity), and the relay module for water pump control enables a highly receptive system. Furthermore, the Raspberry Pi 4 Version B functions as an effective and economical central processing unit that allows real-time decision-making at the side. The application of Flask for the internet interface ensures that individuals can monitor the system's efficiency and control irrigation from another location, including benefit and boosting usability (Buzura et al. 2022)..

Limitation:

Throughout the project, several challenges were experienced, including guaranteeing sensing unit accuracy, taking care of the computational limitations of the Raspberry Pi, and adjust the threshold values for irrigation decisions. These were attended to via calibration, data filtering, optimization of formulas, and iterative screening. The assimilation of edge computing strategies made sure the system can make decisions in your area, decreasing latency and reducing dependence on cloud-based handling, which is important for applications needing instant actions like irrigation.

This project has several limitations and challenges, particularly due to a lack of funding and a lack of knowledge with both Raspberry Pis. It was quite difficult to create a Flask application that displayed sensor data on a webpage. Even with this element's successful implementation, we still lack a thorough understanding of sensor integration and data interpretation, particularly with regard to plant monitoring. Furthermore, because we lacked a genuine plant and specialized sensors like soil moisture monitors, we were unable to gather relevant data for plant care recommendations. In the future, the system may be able to employ machine learning and implicit knowledge to provide useful plant care suggestions by incorporating a soil sensor and evaluating plant health using complex algorithms.

Accurately showing data on humidity and temperature presented another difficulty. We took these readings using the Sense HAT, however the findings were distorted because the Raspberry Pi's operating heat affected the data. This was lessened by using the OpenWeatherMap API and applying a correcting formula to the Sense HAT signals in order to produce precise room temperature readings. On the website, the disparity between measured and corrected values was highlighted by displaying both

raw and modified Sense HAT data for transparency's sake.

A significant issue was discovered when the photoresistor light sensor data was linked to the Flask application via the MQTT protocol. Although the publisher and subscriber scripts performed as planned, transmitting and receiving data without any problems, the Flask application interface kept failing. These problems were most likely brought on by a lack of time to debug and optimize the MQTT implementation within the framework of the web application. Despite these limitations, the project demonstrates a solid foundation for future developments. With more time and funding, the system might be further enhanced to include robust sensor integration, enhanced data processing capabilities, and seamless MQTT-based communication for real-time data display.

Reflection:

Reviewing the project, it is evident that the mix of side computer and AI algorithms has the potential to change exactly how irrigation systems run. By making it possible for real-time, data-driven decisions, the system lowers human treatment and boosts performance. In addition, the job has actually provided useful insights right into the obstacles of integrating hardware and software, specifically when collaborating with restricted computational sources like the Raspberry Pi. (Buzura et al. 2022).

Among the most rewarding elements of this project has actually been its contribution to sustainability. The ability to conserve water by maximizing irrigation aligns with international goals for resource preservation and environmental protection. In future models, the system could be improved with artificial intelligence versions to anticipate irrigation requirements based on lasting information fads, further enhancing effectiveness and flexibility (Buzura et al. 2022).

Future Work:

Looking ahead, there are numerous chances to enhance and expand the system. First, the combination of dirt moisture sensing units might supply more exact data for irrigation choices. Additionally, increasing the system's capabilities to support anticipating analytics might allow it to change watering timetables based upon weather prediction and historic data. Future variations could likewise include higher cloud combination for data storage space and much more advanced maker learning algorithms, making it possible for smarter decision-making on a bigger range.

To conclude, the Smart Irrigation and Dirt Checking System not only addresses essential concerns in water monitoring yet additionally provides a sensible remedy that can be adjusted and scaled to fulfill the requirements of a selection of individuals, from metropolitan garden enthusiasts to massive farming operations. The success of this project highlights the possibility of side computing and AI in promoting sustainability and performance across different industries .

Reference:

Matha Vijaya Phanindra Kumar, Karthika R. (March 2024) “Traffic Sign Detection and Recognition with Deep CNN Using Raspberry Pi 4 in Real-time”, DOI: 10.1109/R10-HTC57504.2023.10461824

Robert Helbet, Vasile Monda, Andrei Cristian Bechet, Paul Bechet, (February 2021) “Low Cost System for Terrestrial Trunked Radio Signals Monitoring Based on Software Defined Radio Technology and Raspberry Pi 4”, DOI: 10.1109/EPE50722.2020.9305536

Zehra Ozkan, Erdem Bayhan, Mustafa Namdar, Arif Basgumus (November 2021)“Object Detection and Recognition of Unmanned Aerial Vehicles Using Raspberry Pi Platform”, DOI: 10.1109/ISMSIT52890.2021.9604698

Ömer Serhat Büyükçolak, Ramazan Yeniçeri, (June 2023)“Quadrotor Model Implementation on Raspberry Pi Zero and Pi 4 Boards using FreeRTOS”, DOI: 10.1109/MECO58584.2023.10154999

Yayan Li,Zhiyong Lin, Zhenxiong Huang, Zerong Cai, Litai Huang, Zongheng Wei, (November 2022) “A Channel Hopping LoRa Technology Based Emergency Communication System for Elderly People Living Alone” DOI: 10.1109/ISCIT55906.2022.9931253

Gabriel-Călin Creț,Laviniu Țepelea, Ioan Buciu, Ioan Gavriluț, Cristian Grava,(July 2021)“Using Block-Matching Methods to Help Navigating Visually Impaired People using Raspberry PI Platform”, DOI: 10.1109/EMES52337.2021.9484126

Loredana Buzura, Gabriel Groza, Radu Papara, Ramona Galatus, (January 2022)“Assisted OCT diagnosis embedded on Raspberry Pi 4”, DOI: 10.1109/SIITME53254.2021.9663686

Abdul Mujiburrohman Luthfi,Iswanto, Alfian Maarif, Gilang Nugraha Putu Pratama, (October 2023)“CNN-BiLSTM for Power Consumption Prediction using Raspberry Pi 4,” DOI: 10.1109/EECSI59885.2023.10295920

Qudes Mb Aljelawy, Tariq M Salman, (September 2022)“Detecting License Plate Number Using OCR Technique and Raspberry Pi 4 With Camera” DOI: 10.1109/ICMI55296.2022.9873776

Catur Adi Nugroho, Amiruddin Amiruddin, (October 2023) “A Low-cost Disk Imaging Device Using Raspberry Pi 4 for Digital Forensics”

DOI: 10.1109/ICoCICs58778.2023.10276963