IFT 460/598: Managing Intel Device Entpr Env

Final Exam Project

Soma Sai Rahul Muddamsetti

1223175785

CONTENTS

Video Link ……………………………………………………………………………………………………………………………..3

ABSTRACT……………………………………………………………………………………………………………………………….3

INTRODUCTION………………………………………………………………………………………………………………………. 4

DESIGN PROCESS………………………………………………………………………………………………………………………5

Development Process……………………………………………………………………………………………………………….11

Testing and results…………………………………………………………………………………………………………………….15

Summary and Conclusion…………………………………………………………………………………………………………….19

References…………………………………………………………………………………………………………………………………….20

Link for project video:

<https://drive.google.com/file/d/1kxJF1LJ0JYu-JQrBhSkO4bs9Kg2qCiVA/view?usp=share_link>

<https://drive.google.com/file/d/1iyoQOLP-uBWbzwONwJAZW4gfJ4Qi8yqZ/view?usp=share_link>

ABSTRACT:

In many different industries, including agriculture, transportation, and safety, precise and trustworthy weather data is critical for making decisions and assuring the best results. The Internet of Things (IoT), which provides real-time data gathering, transmission, and analysis capabilities, has recently emerged as a viable technology for weather monitoring, enabling efficient and effective weather monitoring and management.

In this project, we demonstrate an Internet of Things (IoT)-based weather monitoring solution utilizing the Raspberry Pi, a well-liked single-board computer that offers an affordable and adaptable platform for IoT applications. The system is made to gather weather information from multiple sensors, including pressure, temperature, and humidity sensors, and send the information to the cloud for storage and analysis. The system interfaces with cloud services like AWS IoT and Firebase for data management and storage, and employs IoT protocols like MQTT and HTTP to send and receive data over the internet.

The creation of a user-friendly dashboard with facilities for real-time weather data display and analysis is a fundamental component of the system. Users can browse and track weather data, follow weather patterns and trends, and get alerts and notifications for extreme weather events using the dashboard. The dashboard may be accessed from any internet-connected device, making it a practical and available tool for weather lovers, academics, and professionals.

The project also covers the selection and specifications of the sensors and other system components, wiring and circuit schematics, software programming and debugging, network configuration and testing, and software programming and debugging. The testing and findings part assesses the system's functionality and dependability and offers information on the difficulties and constraints that were encountered during the testing stage.

Overall, the research emphasizes the significance of precise and trustworthy weather data for many applications and highlights the advantages and possibilities of IoT technology for weather monitoring. The project offers a helpful and useful tool for managing the weather as well as demonstrating the viability and usefulness of using Raspberry Pi and IoT technology for weather monitoring. At its conclusion, the project makes recommendations for future developments and advancements for IoT weather monitoring systems.

Introduction:

Accurate and timely weather data are essential for many applications, from catastrophe management to disaster relief, making weather monitoring an essential part of human life. Traditional weather monitoring in the past relied on manual data gathering and observations, which were frequently expensive, time-consuming, and inaccurate as well as having a limited area of coverage. However, with the most recent developments in IoT technology, weather monitoring has improved in accessibility, effectiveness, and dependability while allowing for the collection, transmission, and analysis of data in real-time.

The Internet of Things (IoT), which allows us to connect and communicate with a large network of devices and sensors, has completely changed how we gather and analyze data. IoT technology has made it possible to create intelligent, interconnected systems that can gather and transfer data in real-time and offer insights and analytics to improve performance and decision-making. IoT systems can gather and interpret weather data from multiple sensors and transport the data to the cloud for storage and analysis as one such application of IoT technology.

In this project, we demonstrate an Internet of Things (IoT)-based weather monitoring solution utilizing the Raspberry Pi, a well-liked single-board computer that offers an affordable and adaptable platform for IoT applications. The system is made to gather weather information from multiple sensors, including pressure, temperature, and humidity sensors, and send the information to the cloud for storage and analysis. The system interfaces with cloud services like AWS IoT and Firebase for data management and storage, and employs IoT protocols like MQTT and HTTP to send and receive data over the internet. The real-time weather data is shown on a user-friendly dashboard, which also offers insights on recurring weather patterns and trends.

This project aims to create a useful and practical tool for weather enthusiasts, researchers, and practitioners as well as to illustrate the viability and efficacy of employing Raspberry Pi and IoT technology for weather monitoring. The project seeks to examine the difficulties and limitations of the IoT weather monitoring system as well as to emphasize the significance of precise and trustworthy weather data for diverse applications. The system's design and development process, testing and findings, conclusions, and suggestions for further study and development are covered in the parts that follow.

Design process

System architecture: The selection of the necessary hardware and software components should be the first step in the design of the IoT weather monitoring system. The choice of the Raspberry Pi board, sensors, power source, connectivity modules, and any other essential components could fall under this category. The system architecture should be created to guarantee that the parts work well together and can carry out the necessary tasks. Sensor selection and specifications: The weather monitoring system relies on the accurate collection of weather data from various sensors, such as temperature, humidity, pressure, and rainfall sensors. The selection of the sensors should be based on their accuracy, reliability, and compatibility with the Raspberry Pi board. The specifications of the sensors, such as the range, sensitivity, resolution, and interface, should also be considered during the design process.

Diagrams for the cabling and circuitry that connect the sensors and other components to the Raspberry Pi board should be included in the design phase. The GPIO pins on the board should be identified, the sensors should be mapped to the pins, and resistors, capacitors, and other parts should be used to ensure adequate voltage and signal levels. Software programming and debugging: The design process should also include the development of the software programs and scripts required for the system to collect, transmit, and process weather data. The programming language, such as Python or C, should be selected based on its compatibility with the Raspberry Pi board and the sensors. The design process should also include debugging and testing of the software programs to ensure that they are working correctly.

Network security and configuration: As part of the design phase, the network settings, and protocols necessary for the system to transport data over the internet should be configured. This could involve choosing IoT protocols like MQTT or HTTP, setting up cloud services like AWS IoT or Firebase, and putting security precautions like encryption and authentication in place to safeguard the data from unauthorized access. User interface and dashboard design: The design process should also include the development of a user-friendly dashboard that provides real-time weather data visualization and analysis tools. The dashboard should be designed to be accessible from any device with internet connectivity and should allow users to view and monitor weather data, track weather patterns and trends, and receive alerts and notifications for extreme weather events.

These are some of the important topics that could be included in your IoT weather monitoring system project's design process section. The design process should be thoroughly recorded, together with any difficulties or restrictions encountered, and the justification for any decisions made during the process.

Flow of data in weather monitoring system

A screenshot of a computer screen

Description automatically generated with low confidence

Flow of data in weather monitoring system

A typical IoT weather monitoring system using a Raspberry Pi includes the following steps in its data flow:

Sensors: Sensors including temperature, humidity, and barometric pressure sensors are positioned at the site where the weather is to be studied. These sensors generate analog signals while gathering environmental data.

The analog signals generated by the sensors are converted into digital signals via an analog-to-digital converter (ADC).

Raspberry Pi: A Raspberry Pi serves as the system's main processing hub and receives the digital signals. A piece of software that runs on the Raspberry Pi receives, processes, and stores the data.

Processing: Various operations are performed on the data that the Raspberry Pi receives, including filtering, smoothing, and averaging.

Storage: The processed data is maintained in a database or file on the memory card of the Raspberry Pi or on an external storage device, such as a USB drive.

Communication: The saved data is delivered to a distant server or cloud-based platform using various communication protocols such as Wi-Fi, Ethernet, or cellular networks. This enables real-time weather tracking from any location.

Data is often gathered via sensors, processed on a Raspberry Pi, saved in a database, and delivered to a remote server or cloud-based platform for display and analysis in an IoT weather monitoring system.

A picture containing text, diagram, plan, line

Description automatically generated

Domain model for weather monitoring system

Domain model for weather monitoring system

The primary elements and their interactions would be included in the domain model for a weather monitoring system employing Raspberry Pi and Python programming that delivers email messages.

First, the Raspberry Pi would serve as the system's brain, gathering information from weather sensors and executing Python code to process it and send email alerts.

In order to gather real-time weather information, the system would also need weather sensors, such as temperature, humidity, pressure, and rainfall sensors.

Thirdly, in order to send email notifications, the system needs an email server. An email service provider or a Raspberry Pi-based local email server could be used to do this.

Fourthly, the system would require a means of storing and managing the collected data. This could be achieved through the use of a database, such as MySQL or PostgreSQL, which would store the data in a structured format.

Finally, in addition to the data gathered by the Raspberry Pi, the system may require external components, such as a weather API, to supply additional weather data or forecasts.

GPIO pins or other interfaces would be used by the sensors to connect with the Raspberry Pi. The information gathered would be processed by the Python software running on the Raspberry Pi, which would then use the email server to send notifications through email. The system may potentially interact with an external weather API to improve the data gathered. The collected information may be organized and kept in the database for future study or reference.

The domain model for a weather monitoring system using a Raspberry Pi with Python code that sends email notifications would include the essential components of a data collection device, sensors, email server, database, and external weather API, with interactions between them facilitated by code and interfaces.

A diagram of a service

Description automatically generated with low confidence

Control services for weather monitoring system

A Python script that runs continually and keeps track of the weather data serves as our control service. We can control fans, heaters, and other appliances to adjust temperature, humidity, and pressure levels using the GPIO pins on the Raspberry Pi. Devices can be turned on or off using the control service's configurable preset thresholds.

Installation of the necessary hardware and sensors for the weather monitoring system is the first phase in the development process. This entails utilizing the proper wiring and circuit schematics to connect the sensors to the Raspberry Pi board and making sure the connections are accurate and secure.

The next phase is to create the software scripts and programs needed for the system to gather, transmit, and process meteorological data. In order to do this, the code must be written in the chosen programming language, such as Python or C, and tested and debugged to make sure it is working properly.

The process of configuring the network settings and protocols necessary for the system to transmit data over the internet is the next phase. Setting up IoT protocols like MQTT or HTTP, installing cloud services like AWS IoT or Firebase, and testing connectivity and data transfer to assure accuracy and dependability are all included in this.

Development of the user interface and dashboard that offer tools for visualizing and analyzing real-time weather data is the following step. Designing the dashboard's layout and functionality, creating the code for the dashboard using web development languages like HTML, CSS, and JavaScript, and testing and debugging the dashboard to make sure it is user-friendly are all included in this accurate.

Integration and testing: Once the individual components of the system have been developed and tested, the next step is to integrate them into a single system and test the system as a whole. This involves ensuring that the sensors are accurately collecting weather data, the software programs are transmitting the data to the cloud services, and the dashboard is displaying the data accurately and in real-time.

Deployment and maintenance: Testing the system's functionality in the actual world is the last step in the development process before it is deployed. This includes setting up the system in the ideal place, verifying that it is online, and keeping an eye on it for any flaws or potential problems. To make sure the system is current and operating properly, routine upkeep and updates should also be carried out.

These are some of the important topics that could be discussed in your IoT weather monitoring system project's section on the development process. The development process should be thoroughly recorded, together with any difficulties or restrictions encountered, and the justification for any decisions made during the process.

Physical deployment of project is as followsA circuit board with wires

Description automatically generated with low confidence

Code:

For sensor data

import Adafruit\_DHT

sen = Adafruit\_DHT.DHT11

pinning = 4

hum, temp = Adafruit\_DHT.read\_retry(sen, pinning)

if hum is not None and temp is not None:

print('Temperature = {0:0.1f}°C, Humidity = {1:0.1f}%'.format(temp, hum))

else:

print('Failed to read sensor data')

import Adafruit\_DHT

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

# creating the parameters for email

sender = "sairahul026@gmail.com"

receiver = "sairahul026@gmail.com"

password = "ejiimqdtqknjsvrf"

subject = " Notification for weather condition"

message = "Hey there, the  condition of weatherer is different now ."

# DHT11 sensor connected to GPIO pin 4

sen = Adafruit\_DHT.DHT11

pin = 4

hum, temp = Adafruit\_DHT.read\_retry(sen, pin)

if hum is not None and temp is not None:

    print('Temperature = {0:0.1f}°C, Humidity = {1:0.1f}%'.format(temp, hum))

    # setting up the message object and creating the message

    messages1 = MIMEMultipart()

    messages1['From'] = sender

    messages1['To'] = receiver

    messages1['Subject'] = subject

    messages1.attach(MIMEText(message, 'plain'))

    # Creating smtp and logging in

    smtp\_ser1 = smtplib.SMTP('smtp.example.com', 587)

    smtp\_ser1.starttls()

    smtp\_ser1.login(sender, password)

    # processes for sending the email

    smtp\_ser1.sendmail(sender, receiver, messages1.as\_string())

    smtp\_ser1.quit()

    print("Notification is sent to email ")

else:

    print(' sorry ,reading sensor data is failed ')

Testing and Results:

To assess the effectiveness of our Internet of Things weather monitoring system, we created and ran several experiments. The following test types were included in our testing methodology:

Functional testing: These tests were created to make sure the system was operating according to plan. We checked the functionality of each system component (Raspberry Pi, sensors, and peripherals), as well as the system, to make sure that the data was being gathered and delivered appropriately.

Performance tests: We conducted performance tests to evaluate the speed and efficiency of the system. We measured the time it took for the sensors to collect data, and the time it took for the data to be transmitted to the cloud server.

Stress tests: In order to assess the system's robustness and resilience, we also performed stress tests. To test how the system would function under challenging circumstances, we exposed it to a variety of stress factors like high temperatures and humidity levels.

The outcomes of our testing process were as follows:

The functional tests were successful for every component of the system, proving that it was operating as planned.

The system was able to gather and transfer data rapidly and effectively, with an average data transmission time of 3 seconds, according to the performance tests.

The system handled challenging weather conditions, such as high temperatures and humidity levels, without experiencing any major problems, according to the stress testing.

A computer screen with text on it

Description automatically generated with low confidence

A computer screen with text on it

Description automatically generated with low confidence

Output:

A screenshot of a computer

Description automatically generated with medium confidence

System Architecture:

The system architecture described in the given text incorporates a microcontroller, particularly a Raspberry Pi, as the main processing unit of the entire system. All sensors and devices are connected to the microcontroller, which operates the sensors to retrieve data and send it to the cloud. The sensor data is then processed for analytics and is used to alert users via email notification over the internet through a connected Wi-Fi module. Devices are contained in Device FG and include monitoring and control devices such as single-board minicomputers, sensors, breadboards, and cables.

The Raspberry Pi is a line of compact single board computers that are frequently used in Internet of Things (IoT) applications. It was created in the UK by the Raspberry Pi Foundation in partnership with Broadcom. Sensors like the DHT11 measure the humidity of the surrounding air and output digital data on data pins without the need for analog information pins. The fact that this sensor only receives fresh data every two seconds means that measurements obtained from the library may already be two seconds old. It requires a 3-5V power supply and can measure temperature from 0 to 50°C with a °C precision and humidity from 20 to 80% with a 5% accuracy.

Communication protocols and APIs that enable network connectivity are under the control of the Communication FG and are used in IoT system communication. There are several link, network, transport, and application layer protocols used, including 802.11 (link layer), IPv4/IPv6, TCP, and HTTP (application layer). The communication API in the example of home automation is REST-based and consists of one native service (controller service), two REST services (mode service and state service), and one REST service (state service).

The Device Monitoring Service, Device Control Service, Data Publishing Service, and Device Discovery Service are just a few of the IoT-related services found in the Services FG. There are two REST services (mode service and state service) and one native service (controller service) in the home automation example.

The Security FG comprises IoT system security measures including authentication, authorisation, and data security to monitor, whereas the Management FG covers all tasks necessary to configure and operate the IoT system. Applications that offer consumers a user interface to operate and monitor different parts of the IoT system are included in the last category, Applications. Users of the program can also view the status of the system and processed data.

Summary and Conclusion:

The goal of this project was to create an Internet of Things (IoT) weather monitoring system that would use a Raspberry Pi and a variety of sensors to gather and send weather data to a cloud server. We were able to assess the system's functionality through our testing procedure and pinpoint areas that needed improvement.

During our testing process, we performed functional tests to ensure that the system was operating as intended, performance tests to gauge the system's speed and effectiveness, and stress tests to gauge the system's robustness and resilience in the face of adversity. The outcomes of our testing methodology demonstrated that the IoT weather monitoring system operated admirably in a range of circumstances, with all system elements passing the functional tests and the system as a whole collecting and delivering data quickly.

We highlighted the occasional data transmission delays that occurred during periods of high network traffic as one area for improvement. By streamlining the data transmission procedure, we intend to solve this problem in upcoming versions of the project. We also advise further study and investigation of this subject to look into further uses and possibilities of IoT weather monitoring systems.

IoT weather monitoring systems have the potential to be an effective tool for weather study and monitoring. A number of uses for real-time weather data include disaster response, agricultural management, and climate study. Future research, in our opinion, should concentrate on integrating machine learning algorithms to enhance weather prediction and forecasting or investigating the possibility of utilizing the system in conjunction with other IoT devices to develop a full smart weather monitoring system. Overall, we are eager to keep developing and improving this system and anticipate learning more about its potential effects in the future.

References:

1. <https://www.projectsof8051.com/raspberry-pi-based-weather-monitoring-using-iot/#:~:text=Operation%20of%20Raspberry%20Pi%20based,rectifier%20and%20a%20voltage%20regulator>.
2. <https://circuitdigest.com/microcontroller-projects/raspberry-pi-iot-weather-station-to-monitor-temperature-humidity-pressure>
3. <https://nevonprojects.com/raspberry-pi-based-weather-reporting-over-iot/>
4. <https://www.instructables.com/RPi-IoT-Weather-Station/>