



NITTE
(Deemed to be University)

**NMAM INSTITUTE
OF TECHNOLOGY**

Nitte (DU) established under Section 3 of UGC Act 1956 | Accredited with 'A+' Grade by NAAC

Department of Computer Science and Engineering

Report on Mini Project

Weather Detection System

Course Code: 20CSE21

Course Name: Internet of Things

Semester: VI SEM

Submitted To:

Dr. Venugopala P S
Professor

Department of Computer Science and Engineering

Submitted By:

Shailesh Kumar - 4NM20CS159
Pranav P Shetty - 4NM20CS133
Ryan D Souza - 4NM20CS148

Date of submission: 11 -May -2023

Signature of Course Instructor



NITTE
(Deemed to be University)

**NMAM INSTITUTE
OF TECHNOLOGY**

Nitte (DU) established under Section 3 of UGC Act 1956 | Accredited with 'A+' Grade by NAAC

CERTIFICATE

“Spam Mail Classification” is a bonafide work carried out by Shailesh Kumar (4NM20CS159), Pranav P Shetty(4NM20CS133), and Ryan D Souza(4NM20CS148) in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering prescribed by Visvesvaraya Technological University, Belagavi during the year 2022-2023.

It is certified that all corrections/suggestions indicated for the Internal Assessments have been incorporated into the report. The Mini project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of Guide

Signature of HOD

ABSTRACT

This IoT-based weather detection system project aims to develop a system that can monitor weather conditions and provide real-time data for analysis. The system uses various sensors to measure parameters such as temperature, humidity, atmospheric pressure, and rain. The collected data is then transmitted wirelessly to a cloud-based database service, where it is processed and analyzed. The system communicates with the sensors with the help of Arduino. The collected data is then uploaded to the cloud using raspberry pi.

The system also includes a user interface that displays real-time weather information locally. This is hosted on the raspberry pi device and can be accessed using the IP address of the raspberry pi by any other device on the same network. The project also consists of the remote application feature hosted on a remote hosting platform and access to every other device on the internet. This remote application provides the data of any data collection system by their id as the key. The cloud also runs services that allow one to view data and download the data using the custom Python code in any desired format.

The project utilizes modern technologies such as the Internet of Things (IoT), wireless communication, and cloud computing to create an efficient and reliable weather detection system that can be deployed in various locations. The proposed system can be used in various applications such as agriculture, aviation, transportation, and emergency response.

TABLE OF CONTENTS

Title Page	i
Certificate.....	ii
Abstract.....	iii
Table of Contents	iv
Introduction	4
Problem Statement.....	5
Objectives	6
Purpose and Requirement Specification	6-11
Process Specification	12
Domain Model Information	13-14
Information Model Specification	15
Service Specification	16
IoT Level Specification	17
Functional View Specification.....	18-29
Operational View Specification	30
Conclusion and Future Scope.....	31
References	32

INTRODUCTION

The integration of Internet of Things (IoT) technology has revolutionized many industries, and one such application is the development of weather detection systems. These systems leverage IoT devices and sensors to collect real-time weather data, enabling accurate monitoring and analysis of meteorological conditions. By combining IoT capabilities with weather sensors, data processing, and cloud connectivity, a robust and efficient weather detection system can be created. This essay explores the components and benefits of a weather detection system using IoT.

A weather detection system relies on a network of sensors strategically placed in various locations to capture meteorological parameters. These sensors include temperature, humidity, pressure, rainfall, wind speed, and direction sensors. Each sensor measures specific weather attributes and transmits the data wirelessly to a central hub or gateway. The IoT devices facilitate seamless connectivity and enable the sensors to communicate with each other, forming a comprehensive network for data collection.

Once the weather data is collected, it is processed and analyzed in real time. IoT platforms and cloud-based services play a vital role in managing and processing the massive amounts of data generated by the sensors. The collected data undergoes filtering, aggregation, and analysis to extract meaningful insights. Machine learning algorithms can be employed to detect patterns, trends, and anomalies in weather data, aiding in accurate forecasting and decision-making.

One of the key advantages of an IoT-based weather detection system is its ability to provide remote monitoring and real-time alerts. Meteorological data, including temperature changes, humidity levels, or sudden shifts in wind speed, can be monitored remotely through a web-based or mobile application. Users can receive timely notifications and alerts, allowing them to take appropriate actions or precautions in response to changing weather conditions. This feature is particularly useful for agriculture, transportation, and disaster management industries.

PROBLEM STATEMENT

Design a weather detection system using the concepts of IoT such that the data necessary for computing the weather is collected using the sensor processed on the edge and sent to the cloud for storage and visualizations.

OBJECTIVES

- The project should implement the concepts of IoT.
- The data required for computing the weather should be collected using various sensors like pressure sensors, humidity sensors rain sensors, etc.
- The sensors must be controlled by a microcontroller such as an Arduino board.
- The collected data must be sent to an edge device such as Raspberry Pi which is responsible for processing the data on the site.
- The weather needs to be computed from the collected data using any suitable methods such as statistical learning or machine learning.
- This computation needs to be done on the edge device.
- The edge must also host a local web application written in any language. This should display the collected data and should also refresh the contents as the new content is received from the controller.
- This site should be accessible on any device that is connected to the same network to which the edge device is connected.
- The edge device must be registered on any IoT service platform such as Watson IOT which is deployed on a remote cloud such as IBM Cloud.
- The device must be able to send the data to a remote database such as a Cloudbant database for storing the data.
- The database must be deployed as a service on the remote cloud.
- The communications must use standard protocols such as HTTP, MQTT, or rest-based communication
- The data stored in the database must be accessible to any other external applications for further analysis.
- The cloud must also include a service like data-studio for providing visualizations and the data stored in the database.
- This service must include options for storing the data in different file formats such as JSON, Excel, and CSV.
- The files must be stored in the object storage and must be downloadable using the URL string.

PURPOSE AND REQUIREMENT SPECIFICATION

Purpose:

The system should detect the weather by collecting Temperature, Humidity, Pressure, and Rain data from the environment and sending the data to the cloud. The data is then used for creating visualizations and rendering web applications.

Behavior:

The system should collect the data, process it locally, and send it to the cloud with the data sample rate of 1 sample per second.

System Management Requirement:

The System should be sending the data to Multiple applications using Parallel and Asynchronous threads which are created using the Python program.

The applications are:

- Local web app
- IBM Watson IOT Platform
- Cloudant database hosted in IBM cloud.



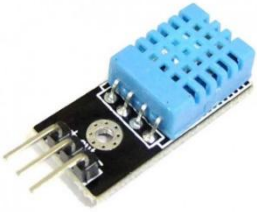
Application Management Requirement:

The application must operate on the edge all the time and send data to the local app and remote cloud for visualization and storage purposes. The application must be able to recover from any errors by implementing exception service routines and efficient thread management strategies for better hardware utilization.

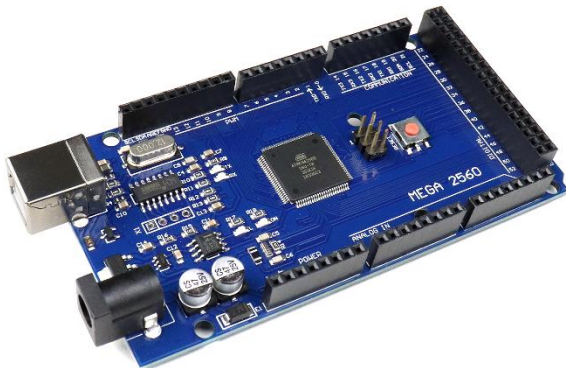
User Interface Requirement:

The collected data and computed weather must be displayed in a web application. Visualization of the data should be provided in different file formats like notebook, CSV, Excel, and JSON files.

HARDWARE SPECIFICATION

<p>Barometric Pressure Sensor</p> 	<p>The BMP180 is a digital barometric pressure and temperature sensor that is commonly used in a wide range of applications, including weather forecasting, altitude measurement, and navigation systems. To use the BMP180 sensor, you will need to connect it to a microcontroller or other digital circuit. The sensor has six pins: VCC, GND, SDA, SCL, EOC, and XCLR. VCC is connected to the positive supply voltage, GND is connected to the ground, and SDA and SCL are connected to the I2C data and clock lines. EOC (End of Conversion) is an output signal that indicates when the conversion of pressure and temperature is complete. XCLR (External Clear) is used to reset the sensor.</p>
<p>Rain Sensor</p> 	<p>A rain sensor, also known as a rain detector or rain switch, is a device designed to detect the presence of rain or moisture. It is commonly used in various applications, such as weather monitoring systems, automatic irrigation systems, and automotive rain-sensing windshield wipers. Conductive rain sensors are electronic devices that can detect the presence of rain by measuring the conductivity of a liquid. They work on the principle that water is a good conductor of electricity, and when rainwater encounters the sensor's electrodes, it changes the electrical resistance between them.</p>
<p>Temperature and Humidity Sensor</p> 	<p>The DHT11 is a digital temperature and humidity sensor that is commonly used in a variety of applications, such as environmental monitoring, weather stations, and home automation systems. To use the DHT11 sensor, you will need to connect it to a microcontroller or other digital circuit. The sensor has three pins: VCC, DATA, and GND. VCC is connected to the positive supply voltage, GND is connected to the ground, and DATA is connected to a digital input/output pin on the microcontroller. The sensor communicates with the microcontroller using a single-wire protocol, which means that only one data line is needed to transmit data.</p>

Arduino ATMEGA2560



The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 chip

Key Features:

- **Microcontroller:** Powered by the ATmega2560 microcontroller with 256 KB of flash memory, 8 KB of SRAM, and 4 KB of EEPROM.
- **I/O Pins:** Provides a total of 54 digital input/output pins, including 15 PWM outputs, and 16 analog inputs.
- **Clock Speed:** Operates at a clock speed of 16 MHz.
- **Memory:** Offers larger flash memory and SRAM compared to other Arduino boards
- **Communication:** Supports various communication interfaces such as USB, UART (serial), I2C, and SPI, facilitating easy interaction with other devices and sensors.
- **Power:** Can be powered via USB or an external power supply, accommodating a wide voltage range (typically 7V to 12V), with a 5V output for powering external devices.

Raspberry Pi 4



The Raspberry Pi 4 is a powerful single-board computer developed by the Raspberry Pi Foundation. It is the fourth generation in the Raspberry Pi series.

Key Features:

- **Processor:** The Raspberry Pi 4 features a Broadcom BCM2711 quad-core ARM Cortex-A72 CPU, running at 1.5 GHz.
- **RAM:** It is available with different RAM options, including 2GB, 4GB, and 8GB LPDDR4-3200 SDRAM, allowing for efficient multitasking and memory-intensive applications.
- **Connectivity:** The Raspberry Pi 4 offers dual-band 2.4 GHz and 5 GHz Wi-Fi 802.11ac, Gigabit Ethernet, Bluetooth 5.0, and two USB 3.0 ports, along with two USB 2.0 ports.
- **Video Output:** It supports dual-monitor output with resolutions up to 4K via two micro-HDMI ports.
- **Storage:** The Raspberry Pi 4 utilizes a microSD card slot for primary storage.
- **Operating System:** It is compatible with various operating systems, including the official Raspberry Pi OS (previously known as Raspbian), Ubuntu, and many other Linux distributions.

SOFTWARE SPECIFICATION

The following software is required to build and run the project.

1. **Python:**

Python is a popular, high-level programming language known for its simplicity and readability. It has a vast ecosystem of libraries and frameworks, making it versatile for various applications. Python's syntax and extensive community support make it an excellent choice for beginners and experienced developers alike.

To install Python on a Raspberry Pi:

1. Open the Terminal and update the package lists:

Run: `sudo apt update`

2. Install Python 3:

Run: `sudo apt install python3`

3. Verify the installation:

Run: `python3 --version`

4. (Optional) Set Python 3 as the default:

Run: `alias python=python3`

2. **Python project dependencies:**

The following dependencies are required to be installed on Raspberry Pi:

- `cloudant==2.15.0`:
Cloudant is a NoSQL database service that provides scalable and distributed data storage.
To install, use the command: `pip install cloudant==2.15.0`.
- `Flask==2.1.1`:
Flask is a popular web framework for building Python web applications.
To install, use the command: `pip install Flask==2.1.1`.
- `Flask_Cors==3.0.10`:
Flask-Cors is a Flask extension that enables Cross-Origin Resource Sharing (CORS) support.
To install, use the command: `pip install Flask_Cors==3.0.10`.
- `joblib==1.2.0`:
joblib is a library for lightweight pipelining and caching of Python functions.
To install, use the command: `pip install joblib==1.2.0`.
- `paho_mqtt==1.6.1`:
paho-mqtt is a Python MQTT client library for MQTT protocol communication.
To install, use the command: `pip install paho_mqtt==1.6.1`.
- `pyserial==3.5`:
pySerial is a Python library for serial communication with devices over a serial port.
To install, use the command: `pip install pyserial==3.5`.

3. Code editor such as Visual Studio Code:

Visual Studio Code is a lightweight yet powerful code editor developed by Microsoft. It offers a rich set of features, including intelligent code completion, debugging capabilities, Git integration, and an extensive library of extensions. It supports various programming languages and platforms, making it a popular choice among developers

To install Visual Studio Code on Raspberry Pi 4:

1. Open the Terminal and update the package lists:

Run: `sudo apt update`

2. Add the Microsoft GPG key:

Run: `wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > packages.microsoft.gpg`

`sudo install -o root -g root -m 644 packages.microsoft.gpg /etc/apt/trusted.gpg.d/packages.microsoft.gpg`

3. Add the Visual Studio Code repository:

Run: `echo "deb [arch=armhf] https://packages.microsoft.com/repos/code stable main" | sudo tee /etc/apt/sources.list.d/vscode.list`

4. Update the package lists again:

Run: `sudo apt update`

5. Install Visual Studio Code:

Run: `sudo apt install code`

4. Arduino IDE:

Arduino IDE (Integrated Development Environment) is a software tool specifically designed for programming and developing applications for Arduino boards. It provides a user-friendly interface, a code editor, and a range of built-in libraries to simplify Arduino programming.

To install Arduino IDE on Raspberry Pi, follow these steps:

1. Open a web browser on your Raspberry Pi and visit the official Arduino website at <https://www.arduino.cc/>.
2. Go to the "Software" section of the website.
3. Download the latest version of Arduino IDE for Linux ARM (32-bit) or Linux ARM 64-bit, depending on the architecture of your Raspberry Pi 4.
4. Once the download is complete, open the Terminal on your Raspberry Pi.
5. Navigate to the directory where the downloaded Arduino IDE package is located. For example, if the package is in the "Downloads" folder, you can use the following command:

`cd Downloads`

6. Extract the downloaded package by running the appropriate command based on the file extension. For example, if the package is a tarball (.tar.gz), you can use the following command:

`tar -xzf <package_name.tar.gz>`

7. Move into the extracted directory:

`cd <extracted_directory_name>`

8. Run the Arduino IDE installation script by executing the following command:

`./install.sh`

9. Follow the prompts and instructions provided by the installation script.

OPERATING SYSTEM SPECIFICATION

The operating system used in the raspberry pi is **Raspberry Pi OS (64-bit)**

The Raspberry Pi OS 64-bit version is an operating system specifically designed for the Raspberry Pi 4 and newer models, which feature a 64-bit ARM architecture. It provides several benefits compared to the 32-bit version, including improved performance and access to a larger address space.

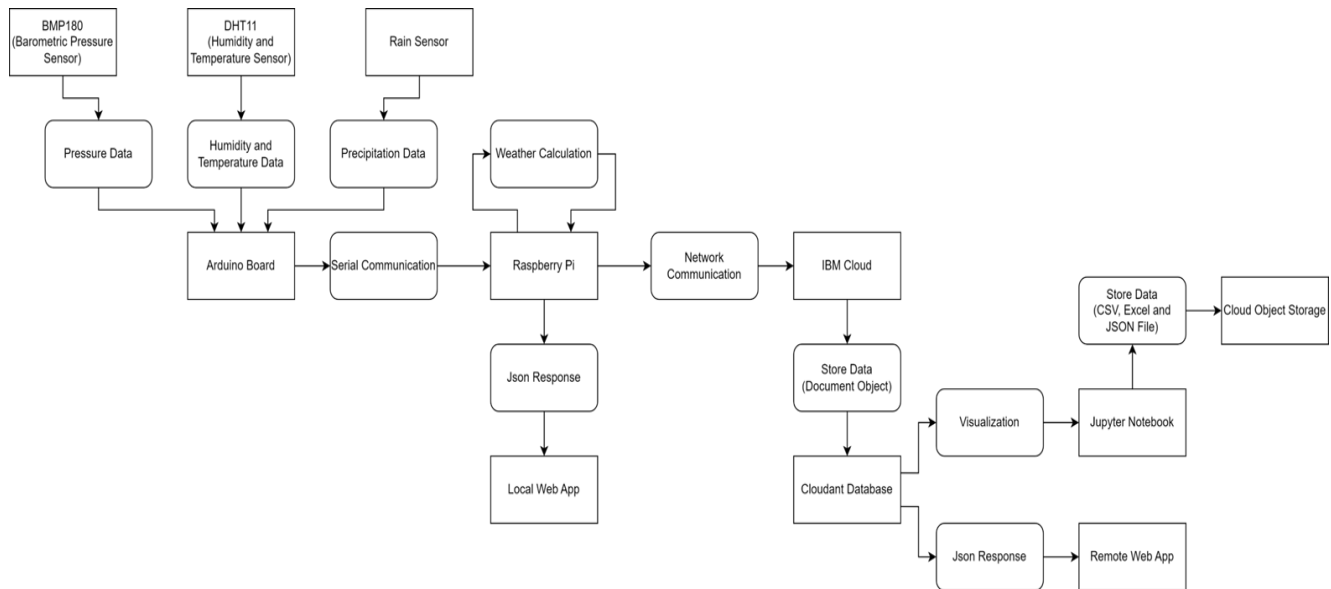
Key Features:

1. **Performance:** The 64-bit architecture allows Raspberry Pi to take full advantage of the hardware capabilities, resulting in improved overall performance and faster processing of tasks.
2. **Memory Access:** With a larger address space, the 64-bit version can support more RAM, enabling the use of larger memory-intensive applications and datasets.
3. **Software Compatibility:** Most of the software and libraries available for the Raspberry Pi are compatible with the 64-bit version. However, some older or specialized software may require modifications or specific 32-bit versions.
4. **Increased Security:** The 64-bit version provides enhanced security features, including stronger data protection mechanisms and improved support for modern encryption algorithms.
5. **Official Support:** The Raspberry Pi Foundation officially maintains and supports the 64-bit version of Raspberry Pi OS, ensuring regular updates and bug fixes.

Installation Steps:

1. Download the Raspberry Pi OS 64-bit image from the official Raspberry Pi website (<https://www.raspberrypi.org/software/operating-systems/>).
2. Prepare an SD card with sufficient capacity (16GB or larger) and format it using a tool like SD Card Formatter (<https://www.sdcard.org/downloads/formatter/>).
3. Use a tool like BalenaEtcher (<https://www.balena.io/etcher/>) to flash the downloaded Raspberry Pi OS 64-bit image onto the SD card.
4. Insert the SD card into your Raspberry Pi and power it on.
5. Follow the on-screen prompts to complete the initial setup of Raspberry Pi OS.
6. Once the setup is complete, you can start using the Raspberry Pi OS 64-bit version on your Raspberry Pi.

PROCESS SPECIFICATION



The entire process starts with the data collection phase where the data is collected from multiple sensors. This includes the data from the temperature, humidity, pressure, and rain sensors. This data is collected using an Arduino microcontroller which performs some unit conversions on the data. Then it appends the device id and encodes the data to JSON format and forwards it to the edge of the gateway device., the raspberry pi via serial communication

The raspberry pi will compute the weather based on the data received. It then appends the weather and the timestamp data to the original data and sends it to the remote cloud for storage using REST-based communication. the data is also sent to the local web application which uses the data for providing information to the user on the local network.

The data stored in the cloud is accessed by a remote web application using Ajax requests and REST-based API calls. This provides information about the weather to any user accessing the website via the internet.

The stored data is also used up by the different data analytics tools which are hosted on cloud platforms for analyzing the data and providing visualizations of the data. These tools can also store the data in different file formats such as JSON, CSV, and Excel. They store the data in IBM cloud object storage which is a nonpersistent storage and can hold up the data for 30 days. The data files hence stored can also be downloaded by setting the bucket as public and pasting the download link in the browser.

DOMAIN MODEL SPECIFICATION

Entities:

1. WeatherSensor:
Represents a physical weather sensor device that measures specific meteorological parameters such as temperature, humidity, pressure, and rainfall.
Attributes: Sensor ID, Sensor Type, Location
2. controller:
Acts as a central hub that receives data from multiple weather sensors.
Attributes: Controller ID, Location
3. Gateway:
Acts as a transponder that receives the data from the controller and sends it to multiple locations.
Attributes: Gateway ID, Location
4. WeatherData:
Captures the data collected by the weather sensors.
Attributes: Sensor ID, Timestamp, Temperature, Humidity, Pressure, Rainfall, Wind Speed, Wind Direction
5. User:
Represents an individual or entity that interacts with the weather detection system and accesses weather data.
Attributes: User ID, Name, Email
6. Cloud:
Stores the data and registers the device and runs the services for data visualizations and analytics.
attributes: service id, API keys, username, password, location

Relationships:

1. WeatherSensor - controller: A one-to-many relationship where multiple weather sensors are connected to a single controller.
2. Controller - gateway: A one-to-one relationship where a gateway receives data from a single controller
3. Gateway-cloud: A one-to-many relationship where multiple cloud services are connected to a single gateway.

behaviors:

1. CollectData:
The WeatherSensor collects weather data at regular intervals and sends it to the Gateway.
2. TransmitData:
The Gateway receives data from the WeatherSensors and transmits it to the cloud or central server for further processing.
3. StoreData:
The WeatherData entity stores the collected weather data, including temperature, humidity, pressure, rainfall, wind speed, and direction.
4. ProcessData:
The collected data is processed and analyzed in real-time to extract meaningful insights, detect patterns, and generate weather forecasts.
5. ProvideAccess:
The User can access the weather data and download the data for further use.
6. MonitorSystem:
The User can monitor the status of the weather detection system, including the connectivity of sensors and the gateway.

Constraints:

1. **Data Accuracy:** The weather sensors should be calibrated and maintained to ensure accurate measurement of meteorological parameters.
2. **Connectivity:** The weather sensors should have reliable wired connectivity to transmit data to the gateway.
3. **Data Security:** Measures should be implemented to ensure the security and privacy of weather data, especially during transmission and storage.
4. **Power Management:** Weather sensors and gateways should have efficient power management mechanisms to ensure continuous operation.

INFORMATION MODEL SPECIFICATION

The weather detection system deals with mainly 4 types of data:

- **Atmospheric Temperature:**

Atmospheric Temperature is determined using DHT11. It is a low-cost sensor that can measure temperature from 0 to 50 degrees Celsius with an accuracy of ± 2 degrees Celsius. The sensor uses a single-wire communication protocol to transmit data, which makes it easy to use with microcontrollers and other digital circuits. temperature is measured using the thermistor. The sensor outputs data in a 40-bit format, which includes 16 bits for temperature. The desirable range for the system range is from 20 to 45 degrees.

- **Atmospheric Humidity:**

Atmospheric Humidity is determined using DHT11. The sensor measures relative humidity from 20% to 90% with an accuracy of $\pm 5\%$. The sensor uses a single-wire communication protocol to transmit data, which makes it easy to use with microcontrollers and other digital circuits. The DHT11 sensor consists of a capacitive humidity sensor and a thermistor, both of which are integrated into a single chip. The sensor measures humidity by sensing changes in the capacitance of the humidity sensor, the sensor outputs data in a 40-bit format, which includes 16 bits for humidity. The desirable range is from 50 to 90%.

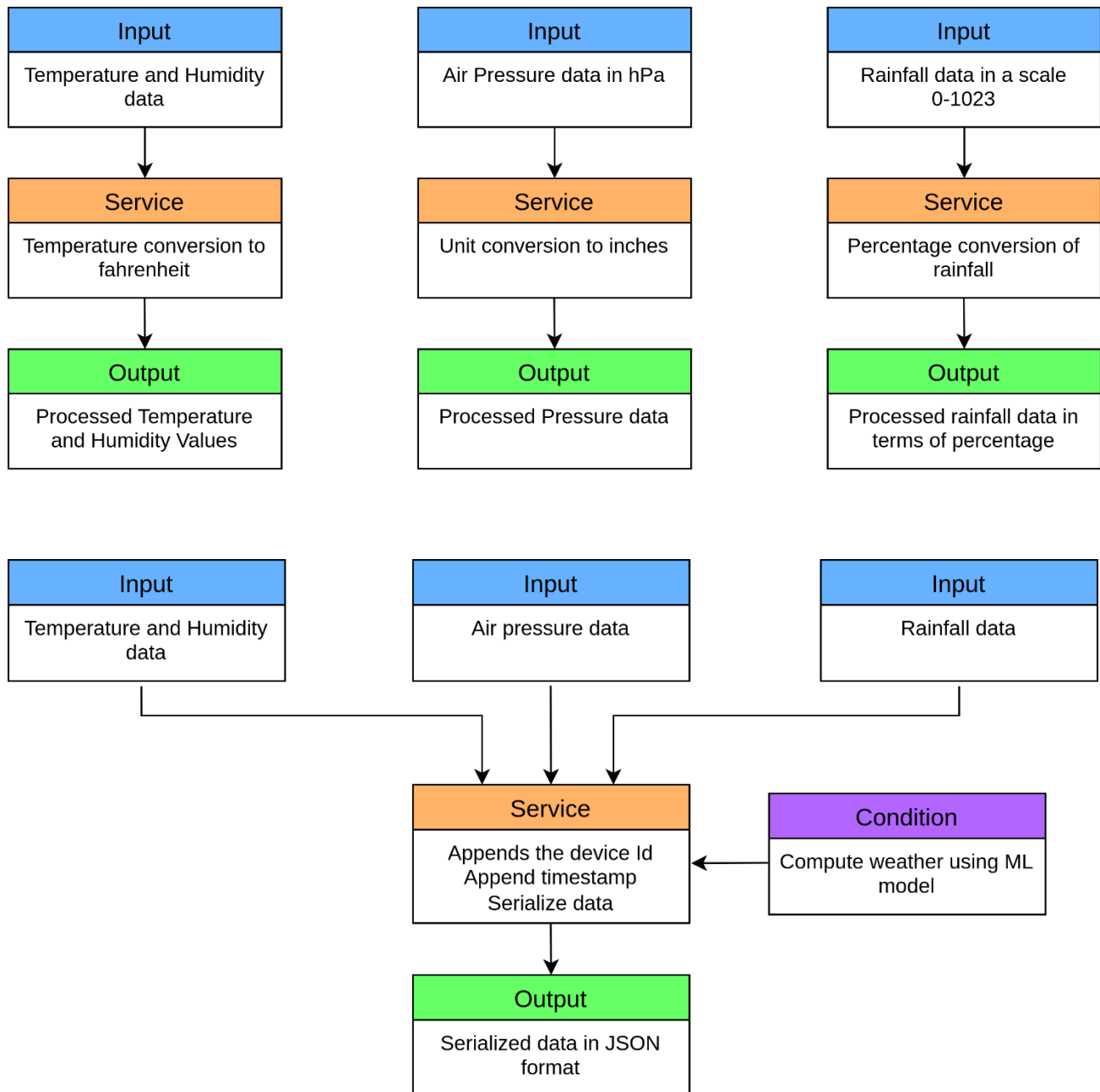
- **Barometric/Air Pressure:**

Air Pressure is determined using BMP180. It is a high-precision sensor that can measure atmospheric pressure from 300 to 1100 hPa with an accuracy of ± 1 hPa, and temperature from -40 to 85°C with an accuracy of $\pm 1^{\circ}\text{C}$. The BMP180 sensor uses an I2C communication protocol to transmit data, which makes it easy to use with microcontrollers and other digital circuits. The BMP180 sensor uses a MEMS (Micro-Electro-Mechanical System) pressure sensor and a temperature sensor to measure pressure and temperature, respectively. The pressure sensor uses a piezoresistive element that measures changes in pressure by sensing changes in the resistance of the element. The sensor outputs data in a 16-bit format, which includes 11 bits for pressure. The pressure in hPa is converted to inches. The desirable range is from 29. into 30.0in.

- **Rainfall:**

Rainfall is measured using the rain sensor. Conductive rain sensors rely on the fact that water is a conductor of electricity. When rain falls on the sensor's surface, the raindrops create a conductive path between the two electrodes. This conductive path reduces the resistance between the electrodes, resulting in a measurable change in conductivity. The sensor circuit measures this change and interprets it as the presence of rain. This gives the output in the form of analog signals having values ranging from 0 to 1023 which is mapped to 0 to 100%. The desirable range is from 0 to 40%.

SERVICE SPECIFICATION



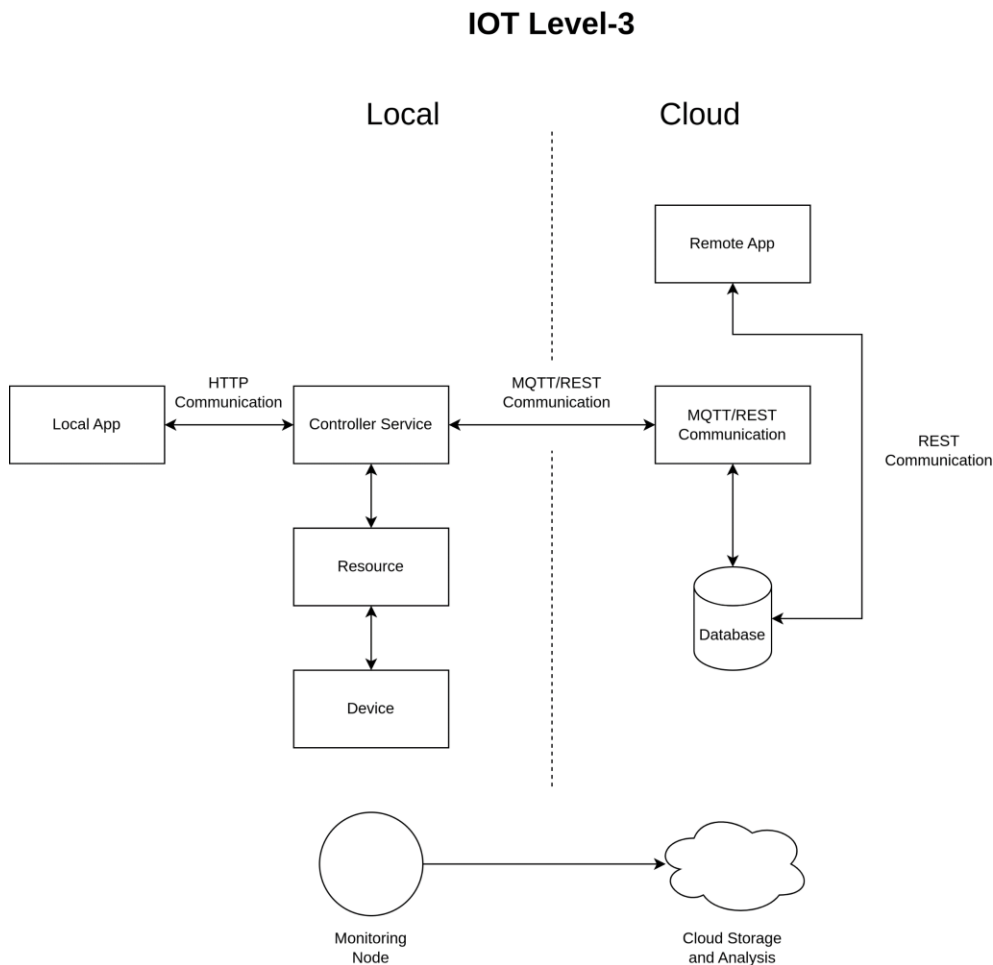
In this project, we are specifically dealing with 4 types of services. 3 of them are related to the sensors. The temperature and humidity sensor will collect the temperature data in Celsius and humidity data in percentage. the Celsius units will be converted to temperature In Fahrenheit.

The air pressure sensor will take the input in terms of hectopascal which is converted into pressure value in terms of inches.

The rain sensor collects the rain value in terms of analog voltage and then it is converted to a percentage range.

The 4th major service is related to the raspberry pi. This includes the computation of the weather using machine learning. Rearranging the data into the proper format and sending this data to local web applications and the remote database hosted in the cloud.

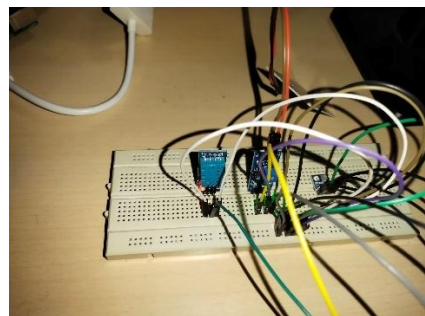
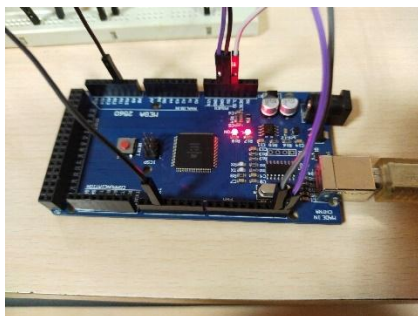
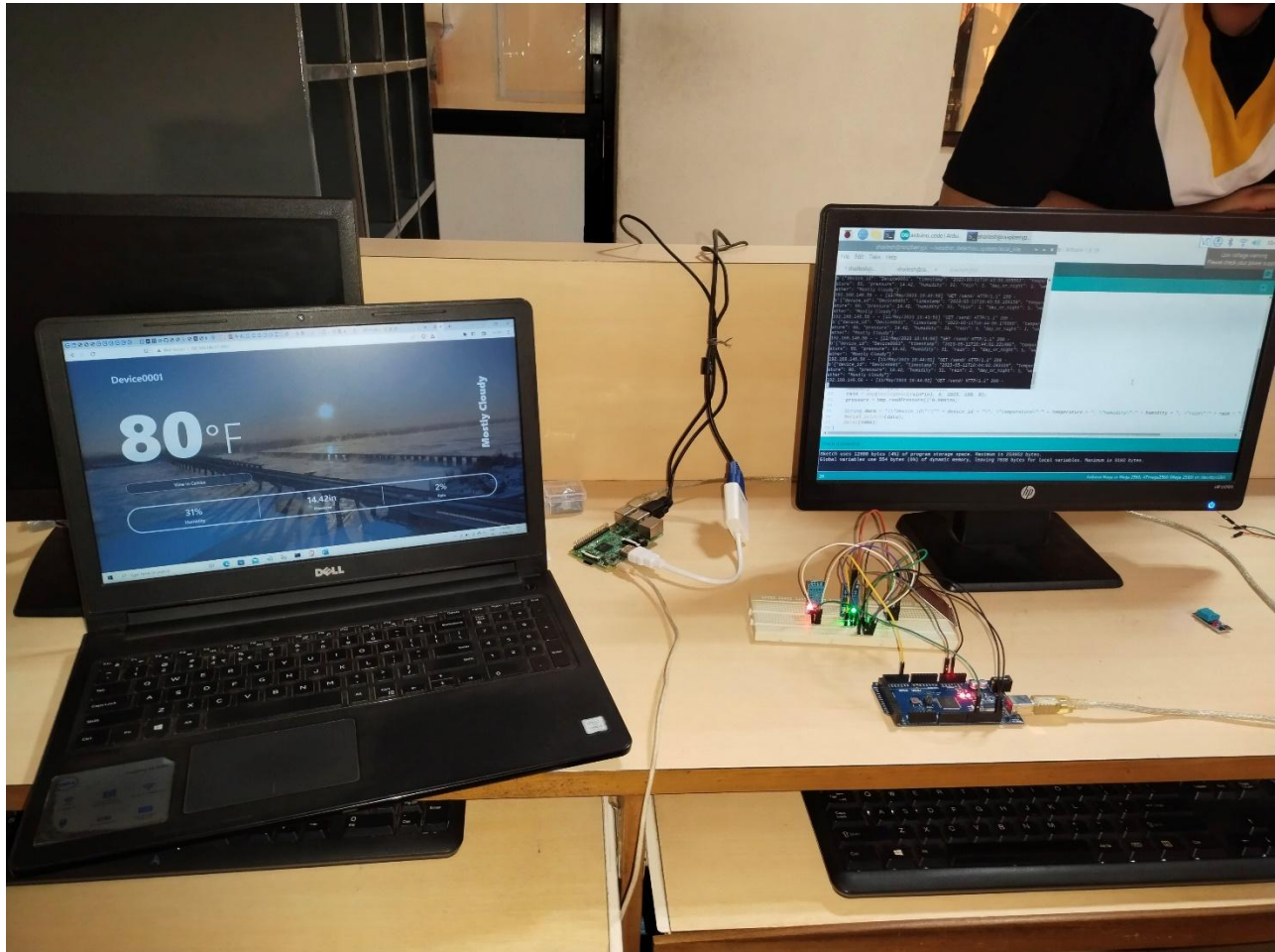
IOT LEVEL SPECIFICATION



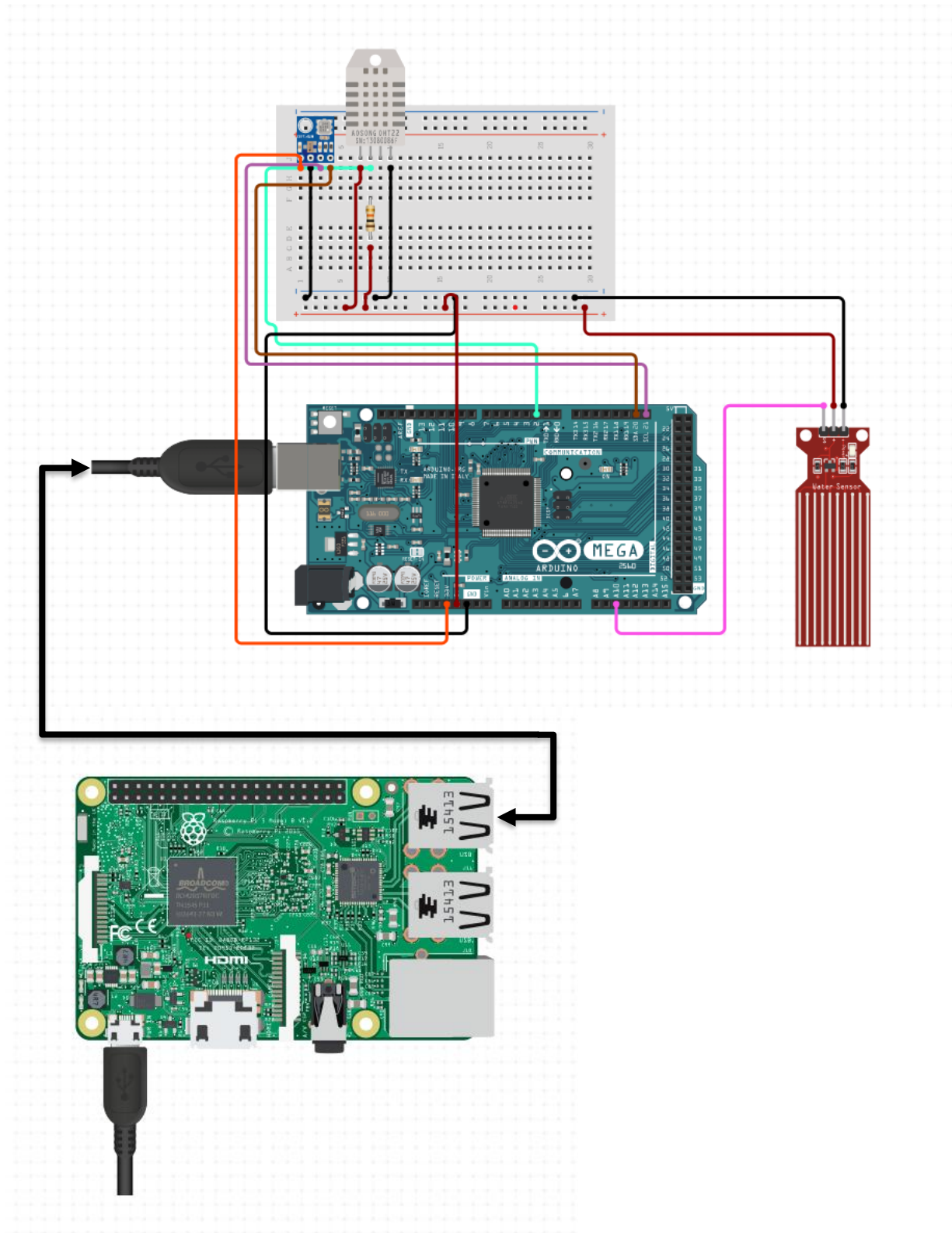
The Project we are going to work with is a level 3 IOT project. In this, a single device collects the data from the surroundings and performs some minor data processing locally. This data can be accessed by the local app running on the gateway device, which is raspberry pi. The controller service will ultimately upload this data to the cloud via Rest (Representational State Transfer) based communication provided by Cloudant REST API. The Cloudant database will be running as a service in the IBM Cloud. The device can also be registered in the Watson IOT Service hosted on the IBM cloud through MQTT communication. The data stored in the cloud will be accessed by the remote application and Watson data studio for further data analysis and other purposes.

FUNCTIONAL VIEW SPECIFICATION

Demonstration of the project setup:



Circuit connection diagram:



Python code running on a raspberry pi:

```
# -*- coding: utf-8 -*-
"""
```

Created on Sun Mar 12 18:44:08 2023

```
@author: user
"""
```

```
import paho.mqtt.client as mqtt
from cloudant.client import Cloudant
from datetime import datetime
import serial
import time
from socket import *
import json
import threading
from joblib import load
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
#=====
# Initializations and connection establishment
#=====
```

```
# Arduino board
ser = serial.Serial("dev/usb0", 115200, timeout=1.0)
time.sleep(3)
ser.reset_input_buffer()
print("Board ready")
```

```
# Model and Weather condition mapping table
model = load('model.joblib')
weather_map_table = {1: 'Mostly Cloudy', 2: 'Fair', 3: 'Partly Cloudy', 4: 'Cloudy', 5: 'Thunder', 6: 'Light Rain', 7: 'Light Rain Shower', 8: 'Fog', 9: 'Rain Shower', 10: 'Light Drizzle', 11: 'Rain'}
```

```
# IBM Watson IoT
device_type = "RaspberryPi"
device_id = "Device0001"
org_id = "1w95zu"
```

```
url = "{}.messaging.internetofthings.ibmcloud.com".format(org_id)
port = 8883
client_id = "d: {}: {}".format(org_id, device_type, device_id)
```

```
topic_name = "iot-2/evt/sensor_data/fmt/json"
```

```
username = "use-token-auth"
token = "Device0001"
```

```

def on_connect(client, userdata, flags, rc):
    print("Connecting to... " + url)
    print("Connection returned result: " + mqtt.connack_string(rc))

def on_disconnect(client, userdata, rc):
    print("Disconnected from... " + url)

def on_publish(client, userdata, mid):
    print("Published a message: " + str(mid))

def on_log(client, userdata, level, buf):
    print("LOG: ", buf)

client = mqtt.Client(client_id)
client.on_connect = on_connect
client.on_disconnect = on_disconnect
client.on_publish = on_publish

try:
    client.username_pw_set(username=username, password=token)
    client.tls_set()
    client.connect(url, port, 60)
except:
    print("Watson IOT Connection failed:")

# IBM Cloudant
URL = 'https://apikey-v2-
2porvk0ltjx250fch5349iijd7z20mmkvm2cgao48mkf:70d9628dc2173b2ea7f6749ff06b1575@b2cf2182-
108f-4d74-99d3-027d6846c573-bluemix.cloudantnosqldb.appdomain.cloud'
API_KEY = 'PQ79KqrbVKEFLkTorUdqxKun1CkU0fikbDI4KephQh1-'

try:
    cloudant_client = Cloudant.iam(None, API_KEY, url=URL, connect=True)
    db = cloudant_client['iot_data']
except:
    print("Cloudant Connection failed:")

# Local Website
serversocket=socket(AF_INET, SOCK_STREAM)
try:
    serversocket.bind(('127.0.0.1', 8000))
    serversocket.listen(5)
    print("serving from 127.0.0.1:8000")
except:
    print("Socket Connection Error")

#=====
# Data forwarding functions
#=====

```

```

def send_forward(data):
    data=json.loads(data)

    device_id = data['device_id']
    temperature = data['temperature']
    pressure = data['pressure']
    humidity = data['humidity']
    rain = data['rain']
    timestamp=datetime.now().strftime("%Y-%m-%dT%H:%M:%S.%f%z")
    if datetime.now().hour<18 and datetime.now().hour>6:
        day_or_night=1
    else:
        day_or_night=0

    # [temperature pressure humidity rain day_or_night]
    prediction = model.predict([[temperature, pressure, humidity, rain, day_or_night]])
    weather = weather_map_table[prediction[0]]

    transmit_data={'device_id': device_id, 'timestamp': timestamp, 'temperature': temperature, 'pressure':
    pressure, 'humidity': humidity, 'rain': rain, 'day_or_night': day_or_night, 'weather': weather}
    transmit_data=json.dumps(transmit_data)

    t1=threading.Thread(target=send_to_watson, args=(transmit_data, ))
    t2=threading.Thread(target=send_to_cloudant, args=(transmit_data, ))
    t3=threading.Thread(target=send_to_local, args=(transmit_data, ))
    t1.start()
    t2.start()
    t3.start()

def send_to_watson(transmit_data):
    (rc, mid) = client.publish(topic_name, payload=transmit_data)

def send_to_cloudant(transmit_data):
    db.create_document(json.loads(transmit_data))

def send_to_local(transmit_data):
    try:
        serversocket.settimeout(5.0)
        (clientsocket, address)=serversocket.accept()
        rawdata=clientsocket.recv(1024).decode()
        info=rawdata.split("\r\n")
        if len(info)>0:
            for i in info:
                print(i.strip('\r\n'))

        clientsocket.sendall(transmit_data.encode())
        clientsocket.shutdown(SHUT_WR)
    except:
        print("Socket Timeout")

```

```
#=====
# Main Context
#=====

try:
    while True:
        time.sleep(0.01)

        if ser.inWaiting()>0:
            line = ser.readline().decode().strip()
            print("Data from arduino:", line)
            t=threading.Thread(target=send_forward, args=(data))
            t.start()

except Exception as e:
    ser.close()
    client.disconnect()
    cloudant_client.disconnect()
    serversocket.close()
    print(e)
```


Arduino sketch running on Arduino IDE:

```
#include <Adafruit_I2CDevice.h>
#include <Adafruit_BMP085.h>
#include <DHT.h>
#define dhtPin 2
#define rainPin A10
#define dhtType DHT11
```

```
DHT dht(dhtPin, dhtType);
Adafruit_BMP085 bmp;
```

```
int temperature;
int humidity;
int rain;
float pressure;
String device_id = "Device0001";
```

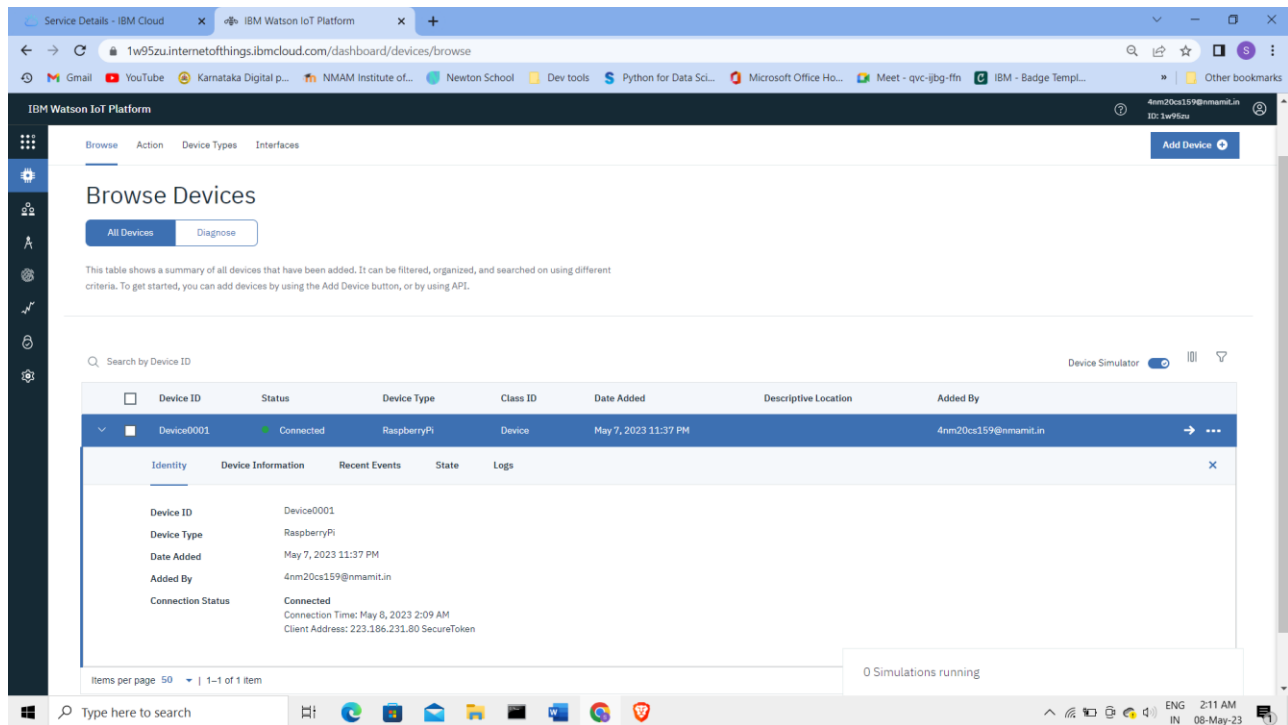
```
void setup() {
  //Put your setup code here, to run once:
  pinMode(dhtPin, INPUT);
  pinMode(rainPin, INPUT);

  Serial.begin(115200);
  dht.begin();
  bmp.begin();
}
```

```
void loop() {
  //Put your main code here, to run repeatedly:
  temperature = int((9 / 5.0) * dht.readTemperature() + 32);
  humidity = int(dht.readHumidity());
  rain = map(analogRead(rainPin), 0, 1023, 100, 0);
  pressure = bmp.readPressure() * 0.0002953;

  String data = "{" + "device_id:" + device_id + ", 'temperature':" + temperature + ", 'humidity':" + humidity
+ ", 'rain'" + rain + ", 'pressure':" + pressure + "}";
  Serial.println(data);
  delay(1000);
}
```

IBM Watson IoT platform screenshot:



Service Details - IBM Cloud x IBM Watson IoT Platform x +

1w95zu.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

Browse Action Device Types Interfaces

Add Device +

Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator ☒ ☐ ☐

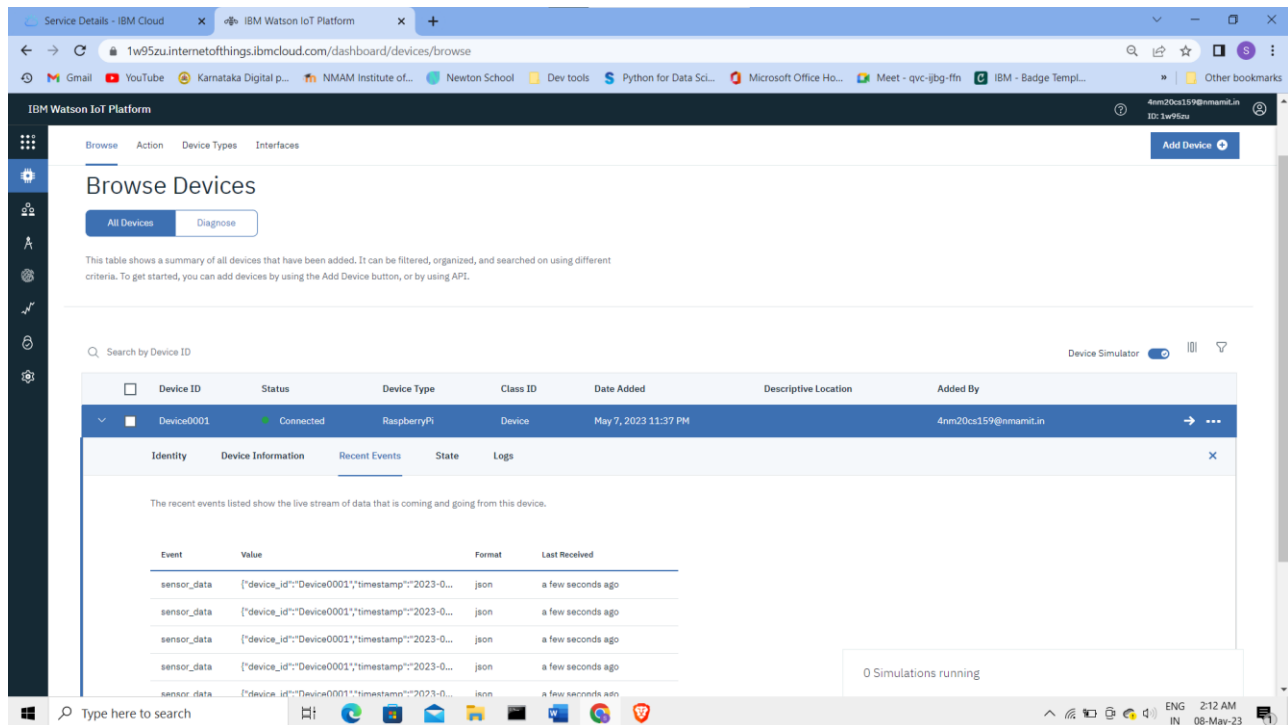
Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By
Device0001	Connected	RaspberryPi	Device	May 7, 2023 11:37 PM		4nm20cs159@nmamit.in

Identity Device Information Recent Events State Logs

Device ID: Device0001
 Device Type: RaspberryPi
 Date Added: May 7, 2023 11:37 PM
 Added By: 4nm20cs159@nmamit.in
 Connection Status: Connected
 Connection Time: May 8, 2023 2:09 AM
 Client Address: 223.186.231.80 SecureToken

Items per page: 50 | 1-1 of 1 item

0 Simulations running



Service Details - IBM Cloud x IBM Watson IoT Platform x +

1w95zu.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

Browse Action Device Types Interfaces

Add Device +

Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator ☒ ☐ ☐

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By
Device0001	Connected	RaspberryPi	Device	May 7, 2023 11:37 PM		4nm20cs159@nmamit.in

Identity Device Information Recent Events State Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
sensor_data	{"device_id":"Device0001","timestamp":"2023-0...	json	a few seconds ago
sensor_data	{"device_id":"Device0001","timestamp":"2023-0...	json	a few seconds ago
sensor_data	{"device_id":"Device0001","timestamp":"2023-0...	json	a few seconds ago
sensor_data	{"device_id":"Device0001","timestamp":"2023-0...	json	a few seconds ago
sensor_data	{"device_id":"Device0001","timestamp":"2023-0...	json	a few seconds ago

0 Simulations running

IBM Cloudant database screenshot

The screenshot shows the IBM Cloudant database interface for the 'iot_data' database. The left sidebar contains navigation options: All Documents, Query, Permissions, Changes, and Design Documents. The main area displays a table of documents with columns 'id', 'key', and 'value'. The table lists seven documents, each with a unique ID and a corresponding key and value. The 'value' column contains JSON objects representing weather data.

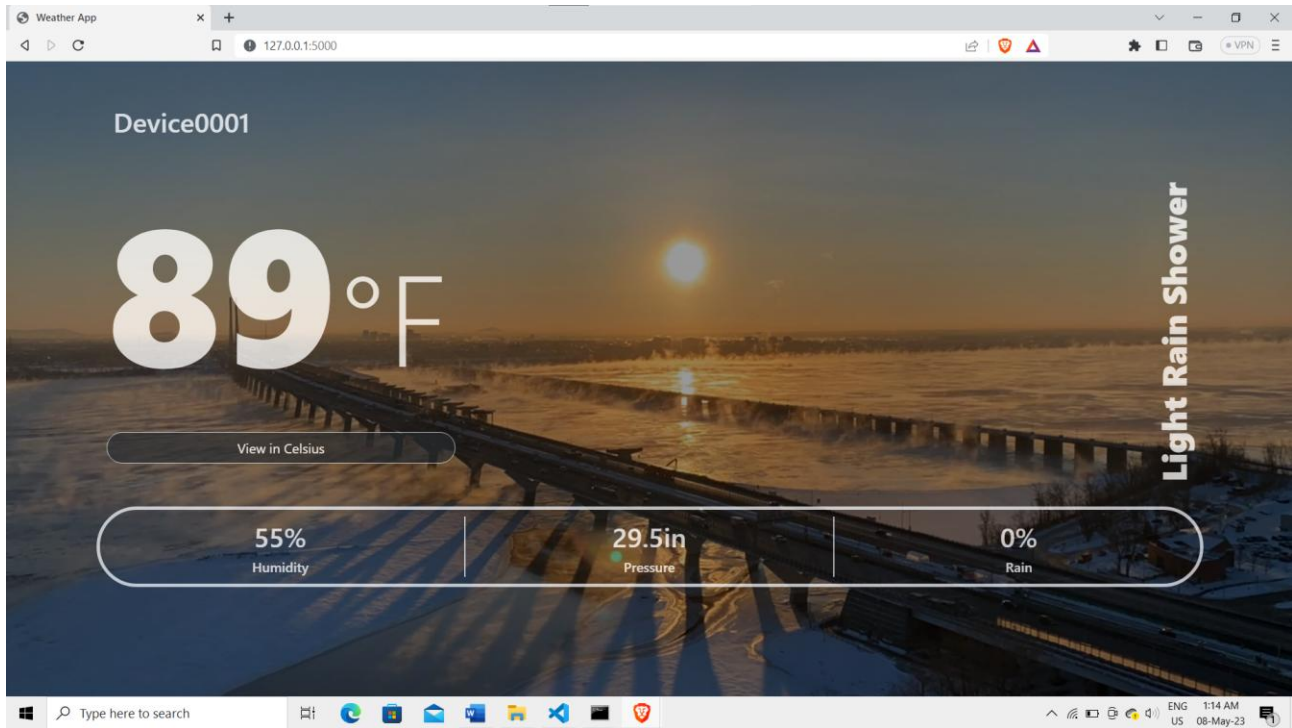
id	key	value
0c0a646d2bbc8ea689df4b9a6dc0b45	0c0a646d2bbc8ea689df4b9a6dc0b45	{ "rev": "1-04e7a1ed3d1e2dd24201daed4b2..." }
163bb430e752f3ad0cbcf0cc581ac4b6	163bb430e752f3ad0cbcf0cc581ac4b6	{ "rev": "1-8f77312c4ae1f221e7636844df1d..." }
18ecca1e1ca71864ee1a660f4fa7b2ce	18ecca1e1ca71864ee1a660f4fa7b2ce	{ "rev": "1-39a539740ea0bf27f753458f52c9f..." }
18ecca1e1ca71864ee1a660f4fa7b2ce	18ecca1e1ca71864ee1a660f4fa7b2ce	{ "rev": "1-bfc3bcc8647a720d1d79032c9e22..." }
2c55ccfeb9da420dc4906d705474fc82	2c55ccfeb9da420dc4906d705474fc82	{ "rev": "1-0b8c773ab39885a5785ccfb454b4..." }
8a1861a9cc98cd5ce8fdc3219ab48fd1	8a1861a9cc98cd5ce8fdc3219ab48fd1	{ "rev": "1-5e47d2bc0301392ceb58039d998..." }
8d95112d157f17b1ebfe0f1c0c413de8	8d95112d157f17b1ebfe0f1c0c413de8	{ "rev": "1-280135ea92ae7aee7d3b3baf6..." }

Showing document 1 - 7. Documents per page: 100

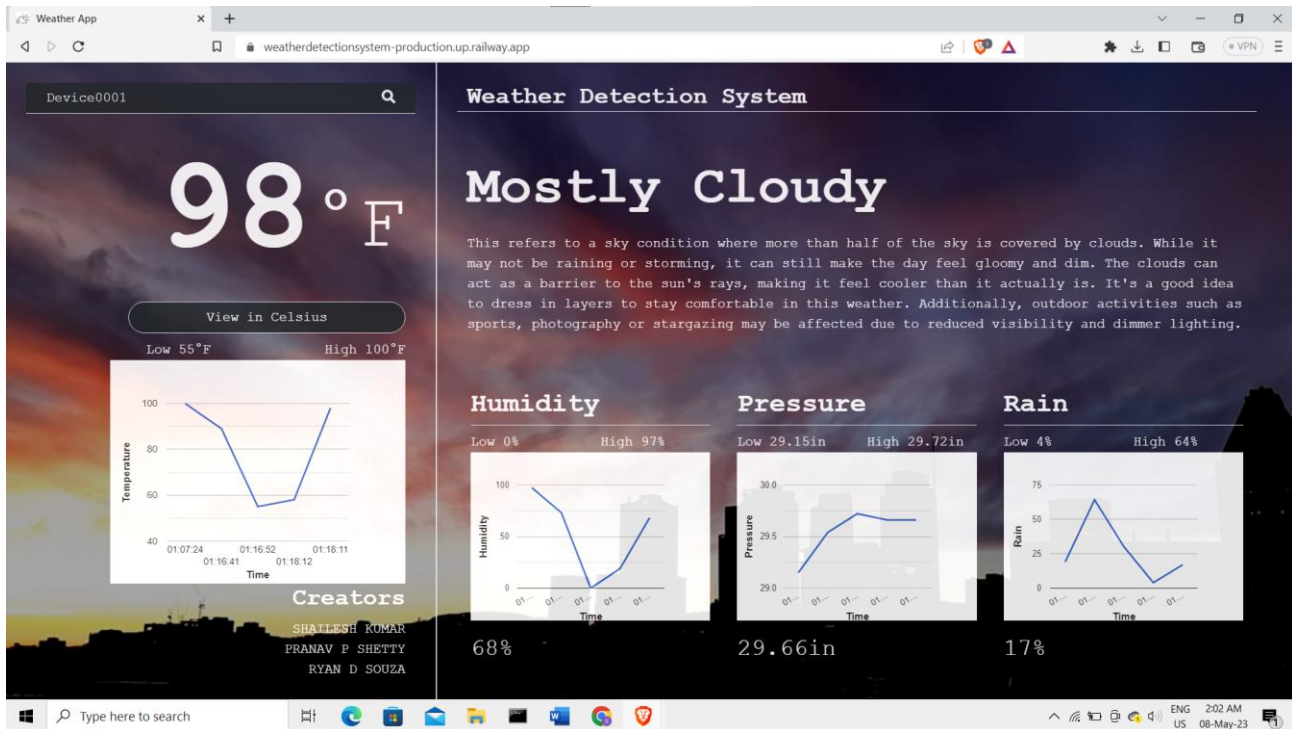
The screenshot shows the IBM Cloudant database interface for editing a document. The document ID is '0c0a646d2bbc8ea689df4b9a6dc0b45'. The document content is displayed in a text editor, showing a JSON object with various weather-related fields. The fields include '_id', '_rev', 'device_id', 'timestamp', 'temperature', 'pressure', 'humidity', 'rain', 'day_or_night', and 'weather'. The document is currently in a full-screen editing mode, with a 'Save Changes' button and a 'Press F11 to exit full screen' prompt.

```
1 {
2   "_id": "0c0a646d2bbc8ea689df4b9a6dc0b45",
3   "_rev": "1-04e7a1ed3d1e2dd24201daed4b20adaa",
4   "device_id": "Device0001",
5   "timestamp": "2023-04-11T13:16:54.268899",
6   "temperature": 57,
7   "pressure": 29.67185639175247,
8   "humidity": 19,
9   "rain": 58,
10  "day_or_night": 1,
11  "weather": "Cloudy"
12 }
```

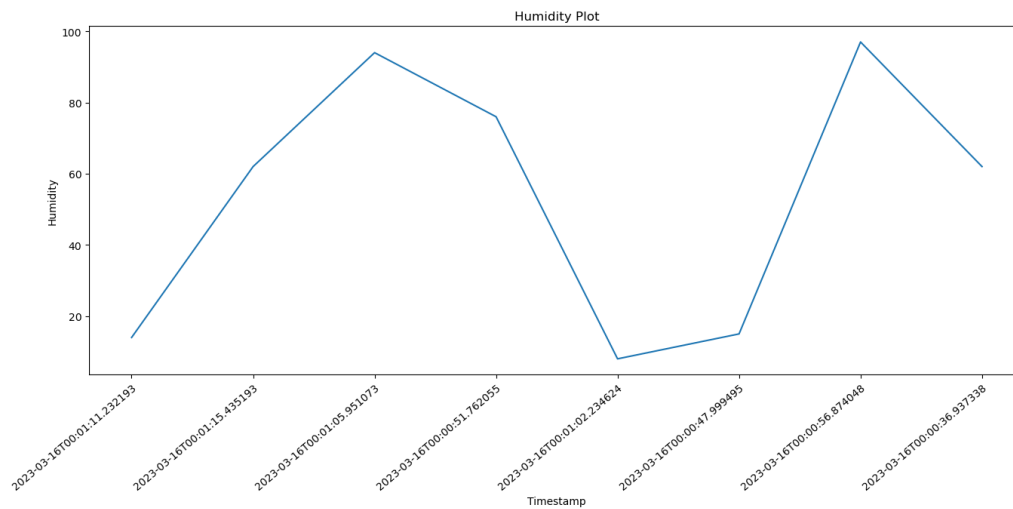
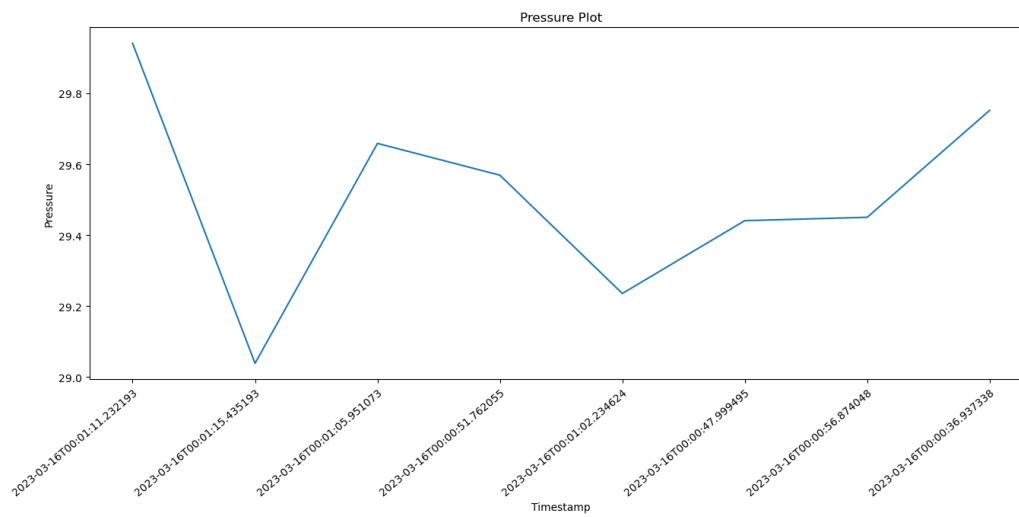
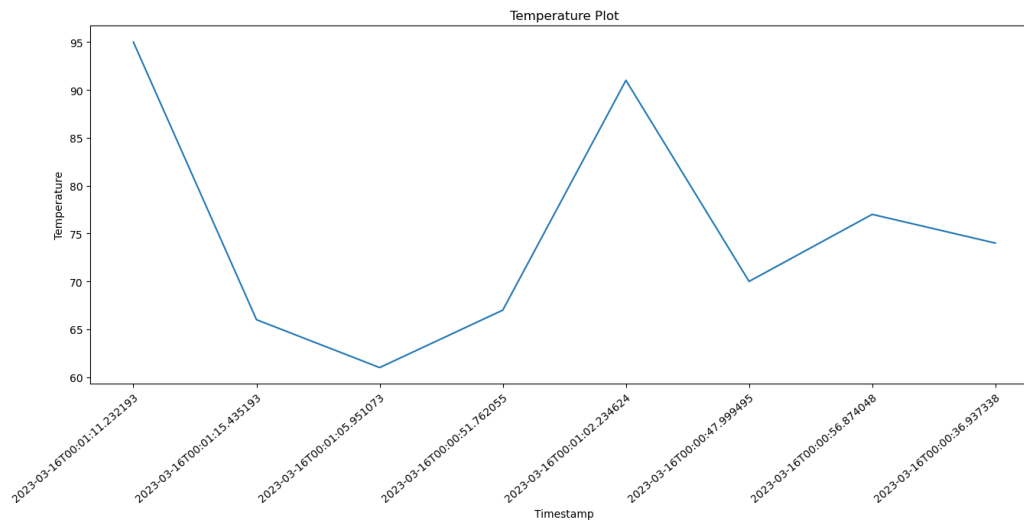
Local web app screenshot

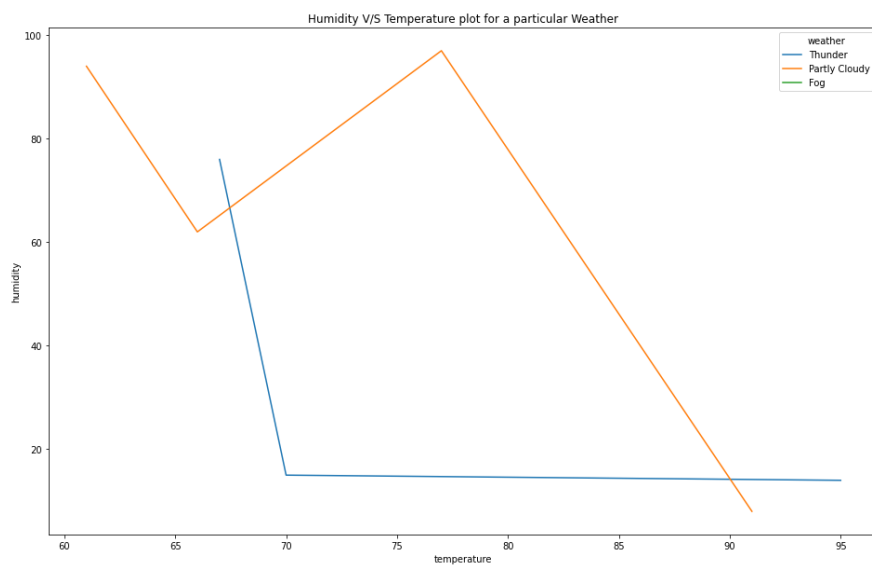
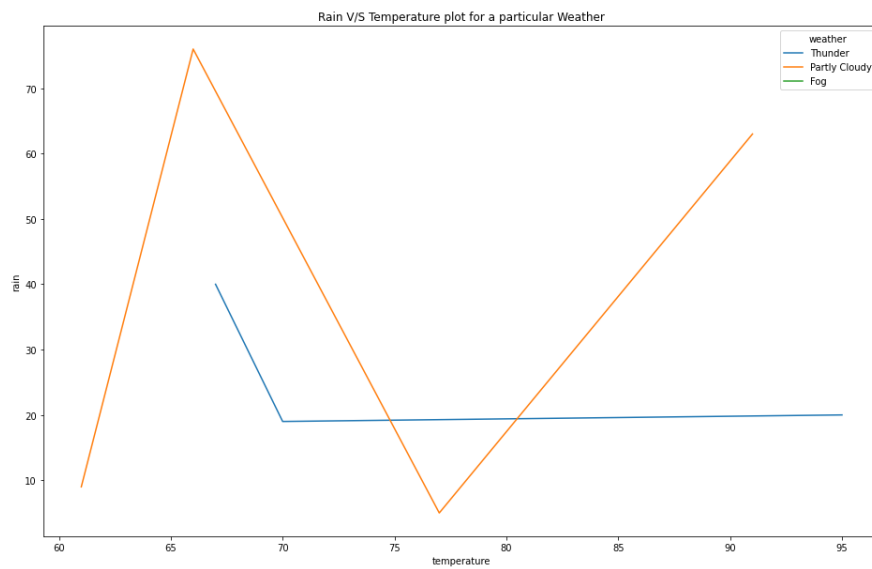
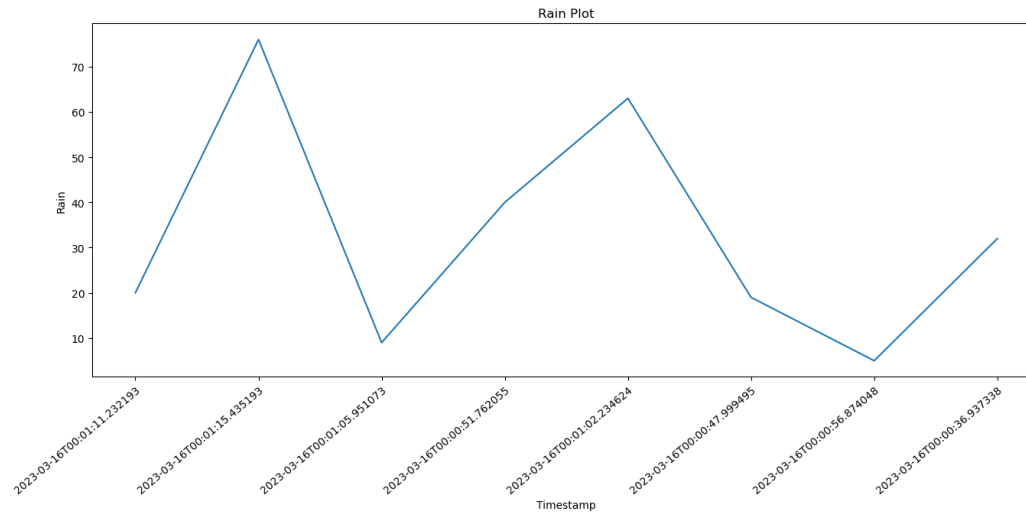


Remote webapp screenshot



Data visualization screenshot





OPERATIONAL VIEW SPECIFICATION

- Arduino and raspberry pi will be active all the time.
- The sensors connected to the Arduino will collect the data and send the data to the Arduino controller.
- The controller will perform the unit conversion and encoding.
- The controller will send the data to raspberry pi
- Raspberry Pi acts as the gateway and edge device.
- It computes the weather using machine learning algorithm and appends the timestamp to the data
- Raspberry Pi forwards the data to the local web app. Which can be accessed using the raspberry pi's local IP address: 192.168.1.105:5000
- Raspberry Pi also sends the data to the Cloudant database hosted in the IBM cloud using the Cloudant APIs
- The database is accessed by the IBM data-studio for performing analysis and visualization of the stored data.
- The data is also converted and stored in various file formats in object storage
- The data is also accessed by the remote web application hosted on the railway's platform, which is accessed using the IP address: <https://weatherdetectionsystem-production.up.railway.app/>

CONCLUSION AND FUTURE SCOPE

Conclusion

- Weather data is successfully collected using multiple sensors
- The collected data is sent to the raspberry pi
- Raspberry Pi computes the weather with an accuracy of 80%.
- The remote database successfully stores the data collected
- The web application can utilize the data stored in the database and enables the user to view the weather information.
- The data stored can be downloaded in multiple file formats to any user device.

Improvements

- Cheaper circuit components like Arduino Nano or esp23 can be used to reduce the project cost
- The model accuracy can be increased by using more relevant and large data.
- The interface design of the web application can be enhanced for providing a better user experience
- The data could be encrypted which prevents unauthorized users from accessing the data in transit.
- The download link can be made private enabling only the authenticated users to access the data.
- The website security can be enhanced by using the WSGI server in place of the development server.
- Paid tier could be used in the cloud which provides much better options for handling the data. Like direct upload of data to the database from the IOT platform, dashboards for visualizations, and direct download of data from the database rather than using object storage.
- Dedicated applications could be used for controlling the edge device rather than using SSH.

REFERENCES

- [1] <https://pypi.org/project/paho-mqtt/>
- [2] <https://developers-dot-devsite-v2-prod.appspot.com/chart/interactive/docs/gallery/linechart.html>
- [3] <https://www.arduino.cc/reference/en/libraries/dht-sensor-library/>
- [4] <https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/>
- [5] <https://flask.palletsprojects.com/en/1.1.x/quickstart/>
- [6] <https://python-cloudant.readthedocs.io/en/stable/query.html>
- [7] <https://0x58.medium.com/connecting-your-raspberry-pi-to-the-ibm-watson-iot-platform-d0d0734cefe4>
- [8] <https://cloud.ibm.com/docs>
- [9] <https://scikit-learn.org/stable/>