07-05-2024
Exp. No.:7

Karthick D
3122 21 3002 041

IoT BASED WEATHER REPORTING SYSTEM

AIM:

- The aim of this project is to develop an Internet of Things (IoT) weather monitoring system using Raspberry Pi equipped with sensors for temperature, humidity (DHT11), and atmospheric pressure (BMP180). Additionally, the project aims to integrate and visualize the data collected from these sensors with Blynk, a platform for building IoT applications, and create a machine learning (ML) based web application for temperature and humidity prediction using logistic regression.

HARDWARE REQUIREMENTS:

- Raspberry Pi
- DHT11 Sensor (for temperature and humidity)
- BMP180 Sensor (for atmospheric pressure)

SOFTWARE REQUIREMENTS:

- Blynk Platform
- Jupyter Notebook
- Python v3.7
- HTML/CSS for Web Application
- Machine Learning Libraries (e.g., Scikit-learn)
- Flask Framework (for Web Application)

THEORY:

- Weather forecasting is crucial for public safety, agriculture, transportation, energy production, and public health. It enables early warnings and preparedness for severe weather events, optimizing crop yields, managing transportation disruptions, ensuring energy grid reliability, and protecting public health from weather-related risks. In disaster preparedness, aviation, maritime navigation, renewable energy planning, and tourism, accurate forecasts aid in decision-making, resource allocation, and risk mitigation. By providing timely and reliable weather information, forecasting contributes to resilience, efficiency, and safety across various sectors, supporting economic stability and societal well-being.

Karthick D
3122 21 3002 041

1. **DHT11:**

The DHT11 sensor is a digital temperature and humidity sensor. It contains an NTC (Negative Temperature Coefficient) Temperature Sensor, a humidity-sensing component, and an 8-bit MCU. The NTC Temperature Sensor is a negative temperature coefficient resistor whose resistance varies inversely with the atmospheric temperature. The 8-bit MCU uses this property to determine the atmospheric temperature. The humidity-sensing component is made up of two electrodes that sandwich a moisture-holding substrate. The conductivity between the two electrodes varies when the moisture between the electrodes varies and the 8-bit MCU uses this to determine the relative humidity of the atmosphere. The 8-bit MCU also handles the transmission of the data with the Raspberry Pi.

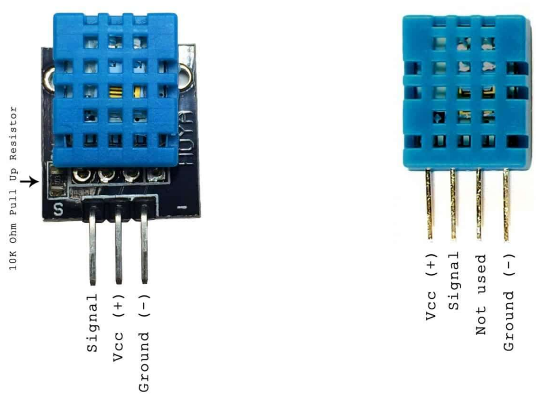
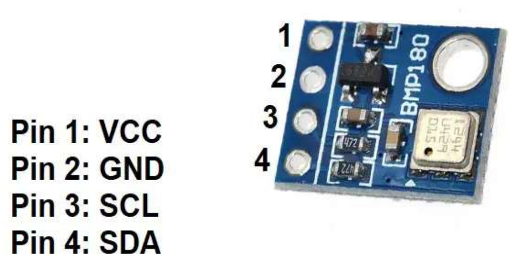


Fig 7.1: DHT 11 Sensor

2. **BMP180**:

The BMP180 sensor is a digital barometric pressure sensor capable of measuring atmospheric pressure and temperature. Utilizing a piezo-resistive sensor, it delivers accurate pressure readings suitable for precise environmental monitoring applications. Additionally, it calculates altitude based on pressure and temperature data, providing valuable insights into changes in elevation. With its digital output interface, the BMP180 seamlessly integrates with microcontrollers for efficient data collection and analysis in weather monitoring systems.



Karthick D
3122 21 3002 041

METHODOLOGY:

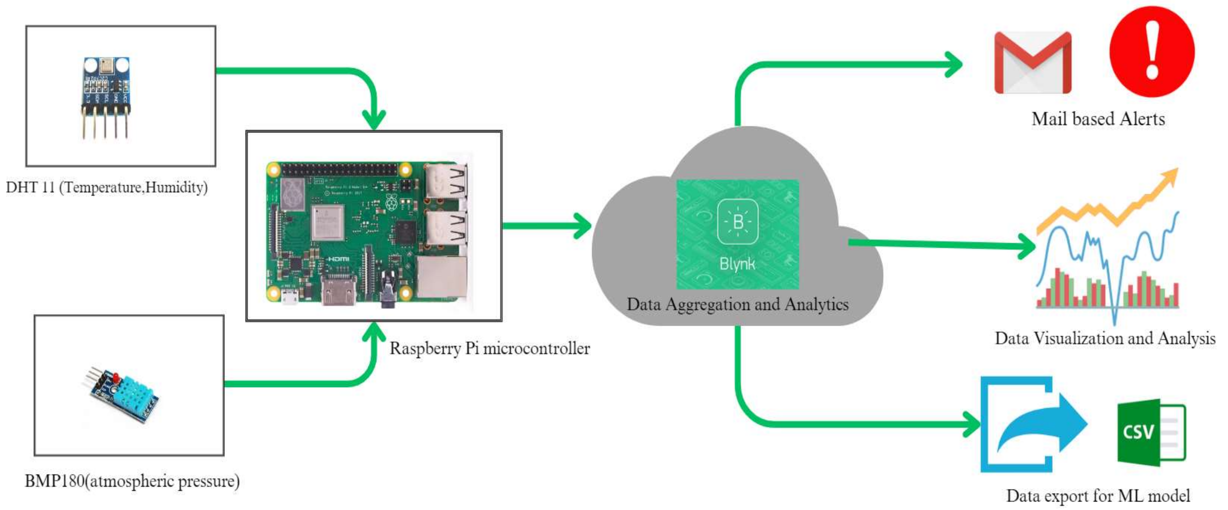


Figure 7.3: Block Diagram

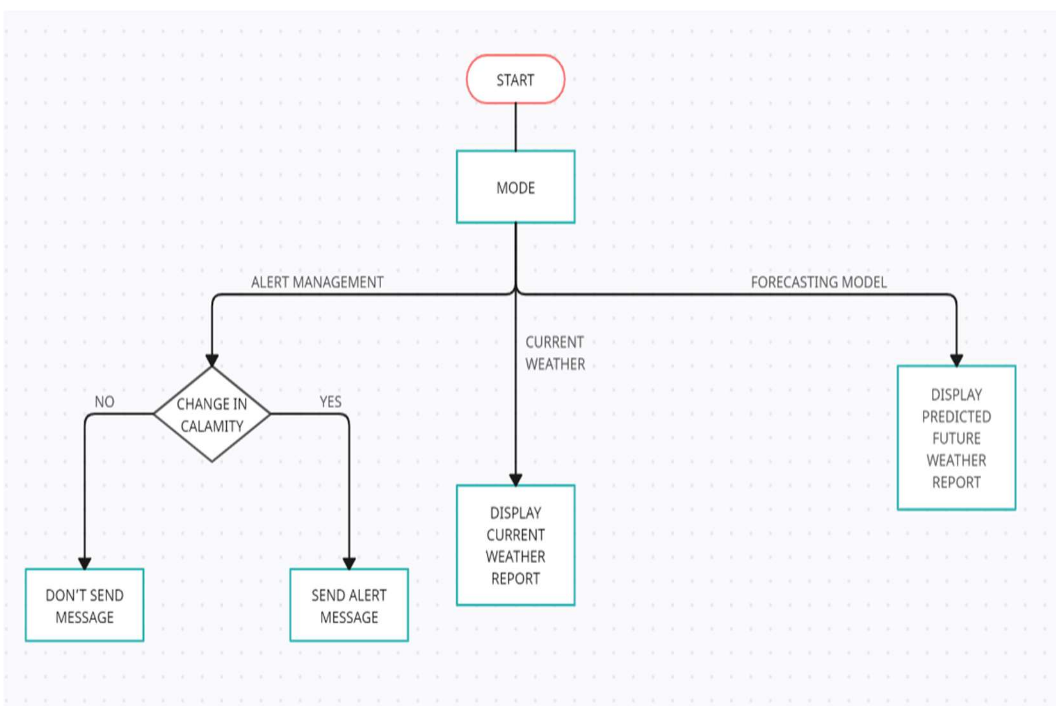


Fig 7.4: Flowchart

Karthick D
3122 21 3002 041

1. . **Hardware Setup:**
    - Install and configure Raspberry Pi as the central hub for data collection and processing.
    - Connect BMP180 and DHT11 sensors to Raspberry Pi according to their respective pin configurations.
    - Ensure proper wiring and connections for accurate sensor readings.
2. **Software Installation and Configuration:**
    - Install necessary Python libraries for sensor data acquisition, such as Adafruit_BMP for BMP180 and Adafruit_DHT for DHT11.
    - Set up Blynk IoT platform for remote monitoring and control of the weather reporting system.
    - Configure Blynk project settings to enable email notifications.
3. **Data Acquisition and Transmission:**
    - Write Python scripts to continuously read data from BMP180 and DHT11 sensors at predefined intervals.
    - Utilize Blynk APIs to send sensor data to the Blynk server for remote access and visualization.
4. **Alert System Setup:**
    - Within the Blynk app, navigate to the project settings and locate the "Notifications" section.
    - Select "Email" as the notification type.
    - Define alert conditions based on sensor readings or specific events, such as thresholds for temperature and humidity.
    - Enter the email address(es) where alerts should be sent and configure the subject and content of the email to provide meaningful information about the alert.
    - Customize the email alerts to include additional information such as timestamp, location, or any other relevant details.
    - Test the alert system to ensure that emails are being sent correctly and that recipients receive them promptly.
5. **Machine Learning Model Development:**
    - Collect historical weather data, including temperature, humidity, and atmospheric pressure.
    - Develop a logistic regression model using Python libraries like Scikit-learn to predict temperature and humidity based on historical data.
6. **Web Application Development:**
    - Use Flask framework for building the backend of the web application.
    - Implement HTML/CSS for the frontend design and user interface.
    - Integrate the logistic regression model into the Flask backend to provide real-time temperature and humidity predictions.

Karthick D
3122 21 3002 041

i. RPi CODE:

```
import time
import board
import adafruit_dht
import psutil
from BlynkLib import Blynk
import bmpsensor

# Initialize Blynk
BLYNK_AUTH_TOKEN = 'Zt_Zs0sJOvKPx5iOjktQh19h_RqqlBuA'
blynk = Blynk(BLYNK_AUTH_TOKEN)

# We first check if a libgpiod process is running. If yes, we kill it!
for proc in psutil.process_iter():
    if proc.name() == 'libgpiod_pulsein' or proc.name() == 'libgpiod_pulsei':
        proc.kill()

# Initialize DHT11 sensor on pin D26
dht_sensor = adafruit_dht.DHT11(board.D5)

@blynk.on("connected")
def blynk_connected():
    print("Connected to Blynk IoT services")

while True:
    try:
        # Read temperature and humidity from DHT11 sensor
        temp = dht_sensor.temperature
        humidity = dht_sensor.humidity
        print("DHT11 Sensor - Temperature: {}*C   Humidity: {}% ".format(temp, humidity))

        # Read data from BMP180 sensor
        bmp_temp, pressure, altitude = bmpsensor.readBmp180()
        print("BMP180 Sensor - Temperature: {}*C     Pressure: {} hPa     Altitude: {} 
meters".format(bmp_temp, pressure, altitude))

        # Send DHT11 data to Blynk using virtual pins
        blynk.virtual_write(1, temp)      # Virtual pin 1 for DHT11 temperature
        blynk.virtual_write(0, humidity)   # Virtual pin 0 for DHT11 humidity

        # Send BMP180 data to Blynk using virtual pins
```

```
    blynk.virtual_write(2, pressure)   # Virtual pin 3 for BMP180 pressure

except RuntimeError as error:
    print(error.args[0])
    time.sleep(2.0)
    continue
except Exception as error:
    dht_sensor.exit()
    raise error

time.sleep(2.0)
```

ii. ML MODEL FOR TEMPERATURE AND HUMIDITY PREDICTION USING LOGISTIC REGRESSION:

```
import pandas as pd
from datetime import datetime, timedelta
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import joblib

# Load the data and preprocess as before (assuming you have already done this)
data = pd.read_csv("dataset.csv", encoding='latin1')
data['Hour'] = pd.to_datetime(data['Time'], format='%I:%M %p').dt.hour
data['Minute'] = pd.to_datetime(data['Time'], format='%I:%M %p').dt.minute
data['Temperature'] = data['Temperature'].str.replace('°C', '').astype(float)
data['Humidity'] = data['Humidity'].str.replace('%', '').astype(float)

# Define time intervals for the next day
time_intervals = ['6:00 PM', '6:30 PM', '7:00 PM', '7:30 PM', '8:00 PM', '8:30 PM', '9:00
PM', '9:30 PM', '10:00 PM', '10:30 PM', '11:00 PM', '12:30 AM', '1:00 AM', '1:30 AM',
'2:00 AM','2:30 AM', '3:00 AM', '3:30 AM', '4:00 AM', '4:30 AM', '5:00 AM', '5:30
AM', '6:00 AM', '6:30 AM', '7:00 AM', '7:30 AM', '8:00 AM', '8:30 AM', '9:00 AM',
'9:30 AM', '10:00 AM', '10:30 AM', '11:00 AM', '11:30 AM', '12:00 PM', '12:30 PM',
'1:00 PM', '1:30 PM', '2:00 PM', '2:30 PM', '3:00 PM', '3:30 PM', '4:00 PM', '4:30 PM',
'5:00 PM', '5:30 PM']

# Generate the time points for the next day
next_day_times = [datetime.strptime(time, '%I:%M %p') + timedelta(days=1) for time
in time_intervals]

# Train linear regression models for temperature and humidity prediction
temp_model = LinearRegression()
humidity_model = LinearRegression()
```

```
X_temp = data[['Hour', 'Minute']]
y_temp = data['Temperature']
X_humidity = data[['Hour', 'Minute']]
y_humidity = data['Humidity']

temp_model.fit(X_temp, y_temp)
humidity_model.fit(X_humidity, y_humidity)

# Save the trained models
joblib.dump(temp_model, 'temp_model.pkl')
joblib.dump(humidity_model, 'humidity_model.pkl')

# Function to make predictions on new data and generate graphs
def predict_and_plot_temperature_humidity(new_data):
    # Load the trained models
    temp_model = joblib.load('temp_model.pkl')
    humidity_model = joblib.load('humidity_model.pkl')

    # Preprocess the new data
    new_data['Hour'] = pd.to_datetime(new_data['Time'], format='%I:%M %p').dt.hour
    new_data['Minute'] = pd.to_datetime(new_data['Time'], format='%I:%M
%p').dt.minute

    # Predict temperature and humidity for each time point
    next_day_predictions = []
    actual_temperatures = []
    actual_humidity = []
    for time_point in next_day_times:
        X_next_day = pd.DataFrame({'Hour': [time_point.hour], 'Minute':
[time_point.minute]})
        next_day_temp_pred = temp_model.predict(X_next_day)
        next_day_humidity_pred = humidity_model.predict(X_next_day)
        next_day_predictions.append({'Time': time_point.strftime('%I:%M %p'),
                                     'Temperature': next_day_temp_pred[0],
                                     'Humidity': next_day_humidity_pred[0]})

        # Get actual values from the dataset
        actual_temp = data[(data['Hour'] == time_point.hour) & (data['Minute'] ==
time_point.minute)]['Temperature'].values
        actual_humid = data[(data['Hour'] == time_point.hour) & (data['Minute'] ==
time_point.minute)]['Humidity'].values
        if len(actual_temp) > 0 and len(actual_humid) > 0:  # Ensure data is available for
this time point
```

```
            actual_temperatures.append(actual_temp[0])
            actual_humidity.append(actual_humid[0])
        else:
            actual_temperatures.append(None)
            actual_humidity.append(None)

    # Convert predictions into DataFrame
    next_day_predictions_df = pd.DataFrame(next_day_predictions)

    # Plot bar graph for Temperature
    plt.figure(figsize=(10, 5))
    plt.bar(range(len(next_day_times)), actual_temperatures, width=0.4, align='center',
label='Actual Temperature')
    plt.bar([x + 0.4 for x in range(len(next_day_times))],
next_day_predictions_df['Temperature'], width=0.4, align='edge', label='Predicted
Temperature')
    plt.xlabel('Time Intervals')
    plt.ylabel('Temperature (°C)')
    plt.title('Actual vs Predicted Temperature for Next Day')
    plt.xticks(range(len(next_day_times)), [time.strftime('%I:%M %p') for time in
next_day_times], rotation=45)
    plt.legend()
    plt.tight_layout()
    plt.show()

    # Plot bar graph for Humidity
    plt.figure(figsize=(10, 5))
    plt.bar(range(len(next_day_times)), actual_humidity, width=0.4, align='center',
label='Actual Humidity')
    plt.bar([x + 0.4 for x in range(len(next_day_times))],
next_day_predictions_df['Humidity'], width=0.4, align='edge', label='Predicted
Humidity')
    plt.xlabel('Time Intervals')
    plt.ylabel('Humidity (%)')
    plt.title('Actual vs Predicted Humidity for Next Day')
    plt.xticks(range(len(next_day_times)), [time.strftime('%I:%M %p') for time in
next_day_times], rotation=45)
    plt.legend()
    plt.tight_layout()
    plt.show()

    return next_day_predictions_df
```

Karthick D
3122 21 3002 041

RESULTS AND DISCUSSION:

- Through the integration of Raspberry Pi, BMP180, DHT11 sensors, Blynk IoT platform, and machine learning-based web app development, a comprehensive IoT weather reporting system was developed. This system enables real-time monitoring of weather conditions and provides predictive insights through machine learning. By configuring email alerts within Blynk, users can receive timely notifications based on predefined conditions. The successful deployment and testing of the system demonstrate its potential for enhancing weather monitoring capabilities in various applications.

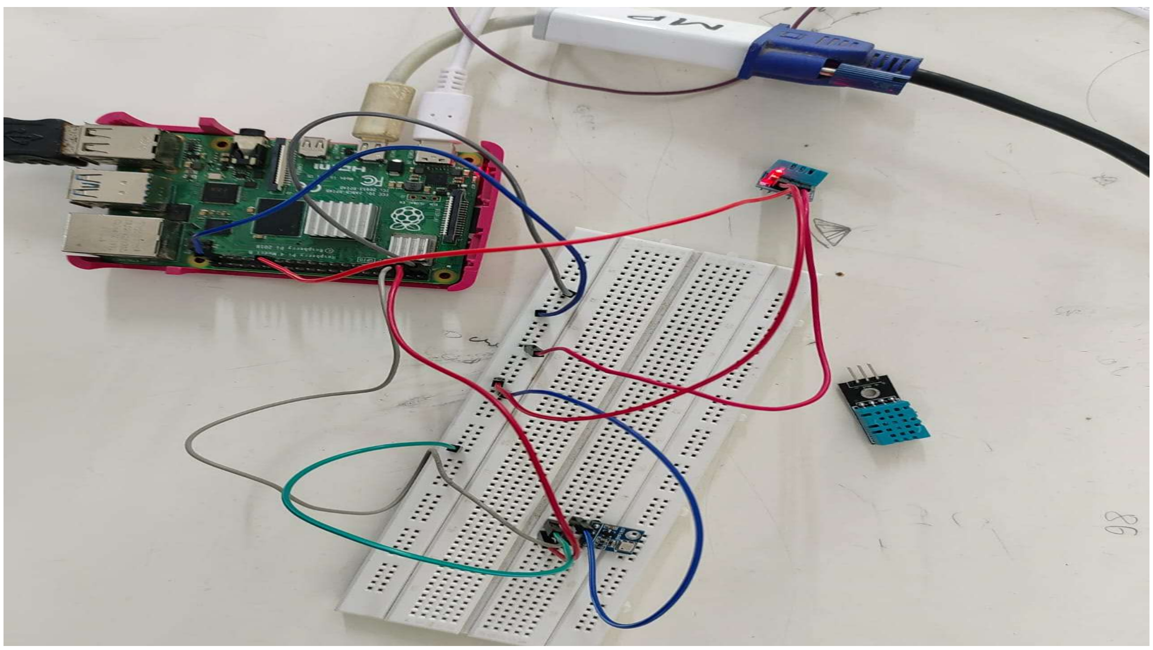


Figure 7.5: Hardware Implementation

Karthick D
3122 21 3002 041

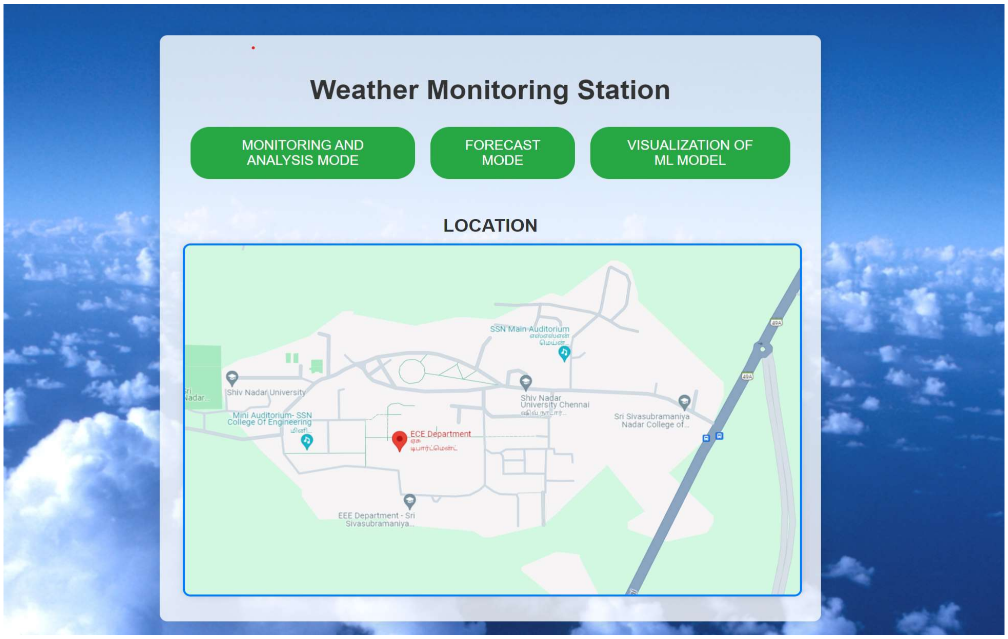


Figure 7.6: Deployment for web app

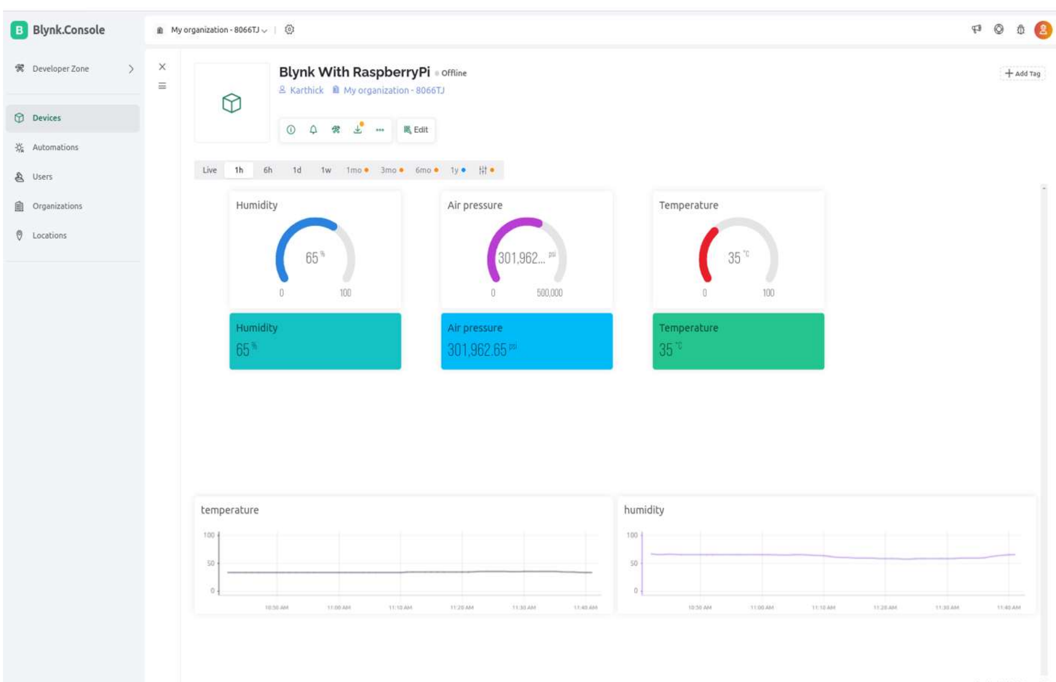


Figure 7.7: Monitoring and Analysis using Blynk IOT

Figure 7.8: Temperature and Humidity Prediction

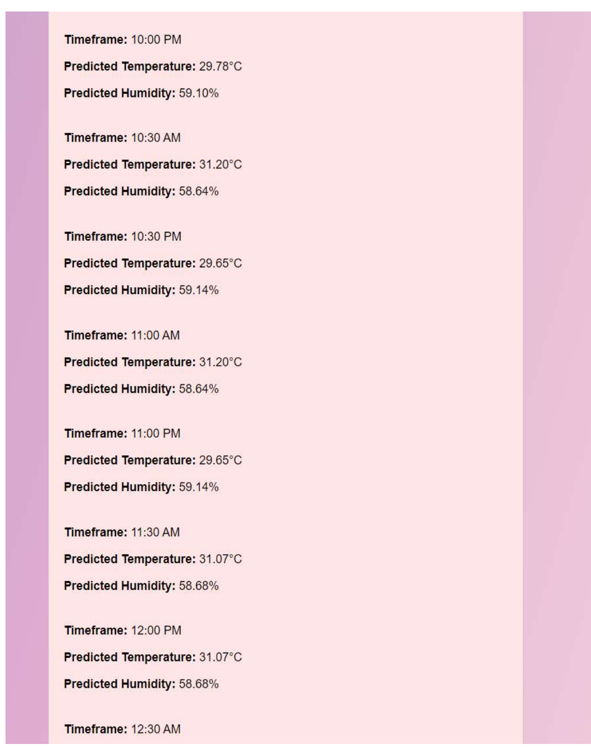


Figure 7.9: Humidity and Temperature Prediction using ML model

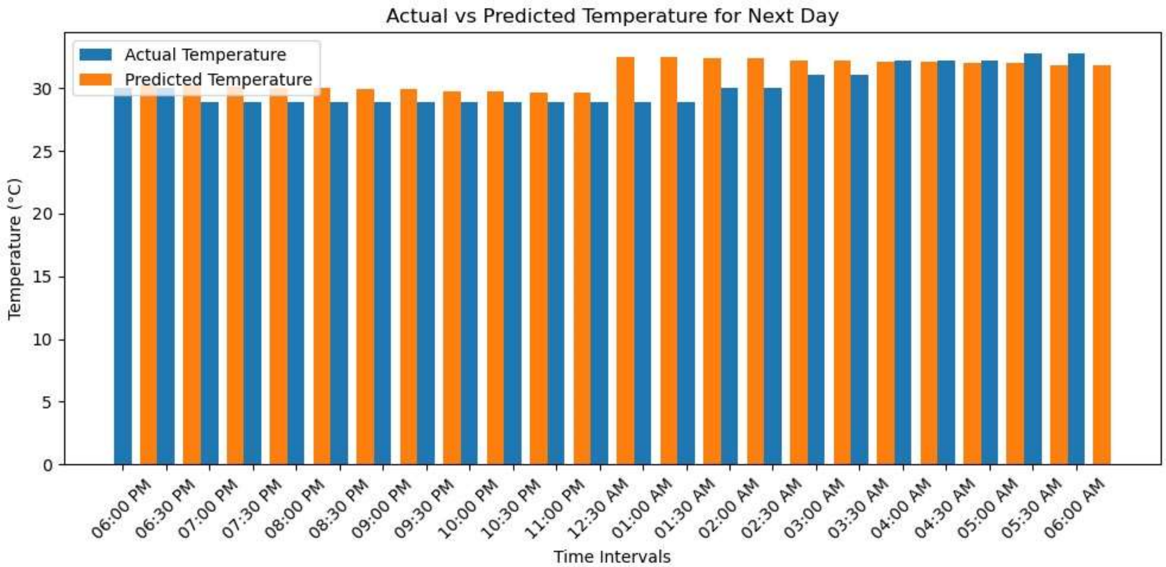
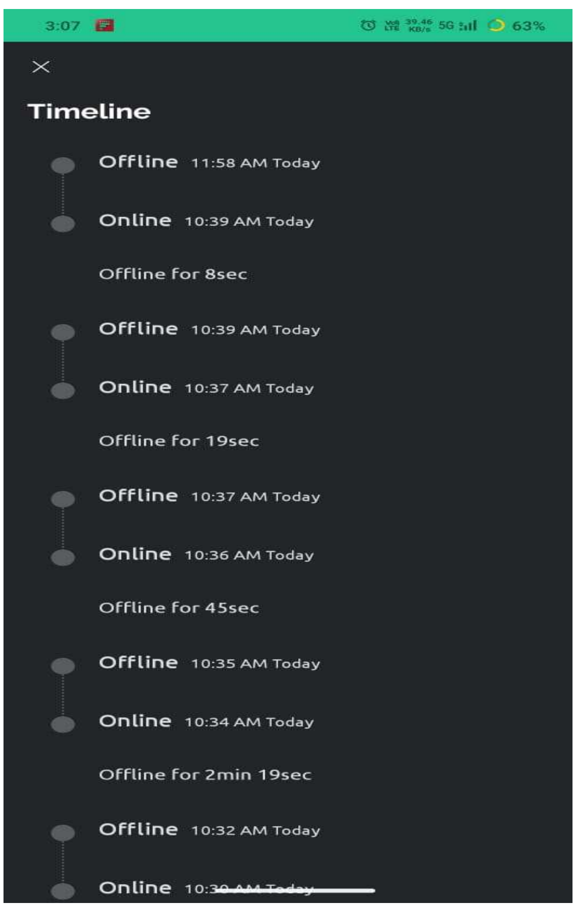


Figure 7.10: ML model Accuracy visualization



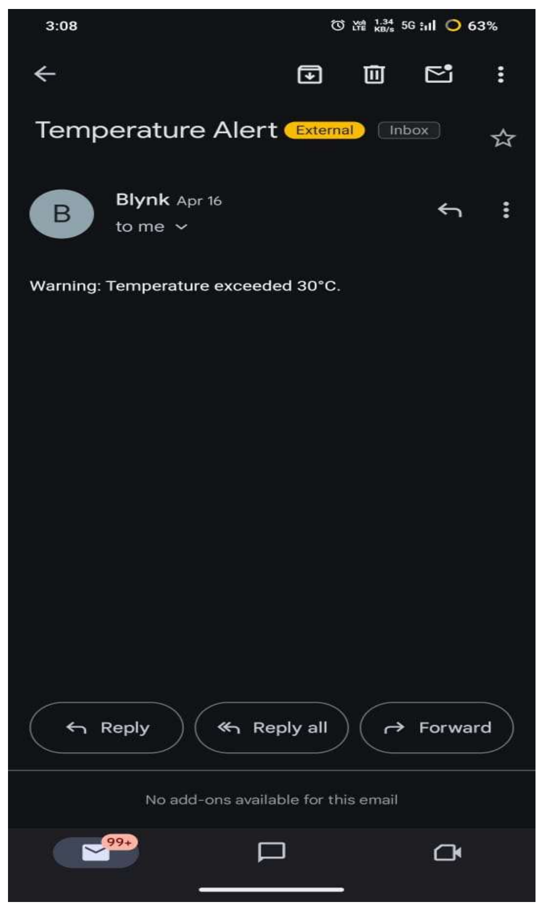


Figure 7.11: Alert using Blynk IOT Console

Karthick D
3122 21 3002 041

RESULT:

The deployed weather monitoring system leverages the advantages of the Blynk platform, providing real-time data collection and analysis. It effectively measures temperature, humidity, and atmospheric pressure, allowing for accurate weather monitoring. Through Blynk's interface, users can remotely access and visualize the collected data. Alert messages and notifications are sent to registered users based on predefined conditions, ensuring timely response to changing weather conditions. As a future scope, the system can be expanded to include additional sensors or integrated with machine learning algorithms for advanced weather prediction capabilities.

	6	**6**	**6**	**6**	**6**	**Total Marks**	**Faculty Signature**
Assessment 1	**5**	**6**	**4**	**4**	**5**	**24**	
Assessment 2	**4**	**3**	**3**	**2**	**3**	**15**	
Assessment 3	**6**	**5**	**6**	**5**	**6**	**28**	
Continuous Evaluation	**23**						