Team: Renukanthan, Akil & Vishwaa

Mini Project
IoT-Based Weather Station Using Raspberry Pi

AIM:

To build a prototype of an IoT-based weather station using a Raspberry Pi that collects, processes, and visualizes real-time weather data, including temperature, humidity, pressure, and rain detection, enabling remote monitoring.

HARDWARE REQUIREMENTS:

- Raspberry Pi (e.g., Raspberry Pi 3, 4, or Zero W)
- DHT11 or DHT22 Temperature and Humidity Sensor
- BMP180 or BMP280 Barometric Pressure Sensor
- Rain Gauge or Rain Detection Module
- MicroSD Card for Raspberry Pi OS installation
- Optional: Anemometer, Wind Vane, OLED/LCD Display
- Power Supply (5V 2.5A or higher)
- Breadboard and Jumper Wires

SOFTWARE REQUIREMENTS:

- Raspberry Pi OS (Raspbian)
- Python 3 (pre-installed on Raspberry Pi OS)
- Adafruit Python Libraries (e.g., Adafruit_DHT)
- SQLite or Cloud-based Database
- Optional: Flask or Django for web interface

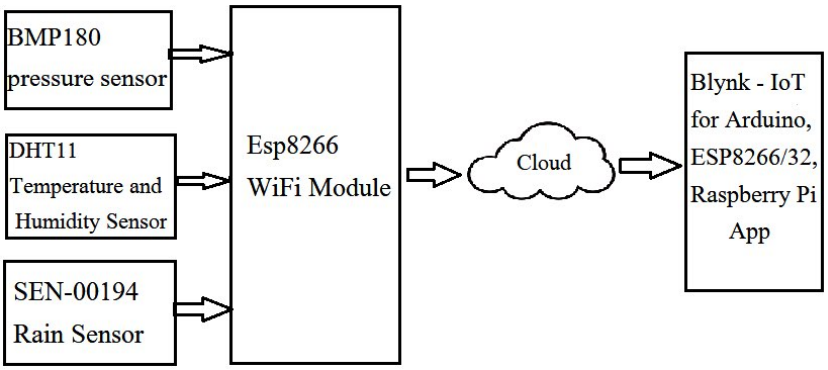
THEORY:

An IoT-based weather station is designed to monitor weather conditions in real-time. This system uses Raspberry Pi as the primary computational device, allowing it to collect and analyze data from various sensors, including temperature, humidity,

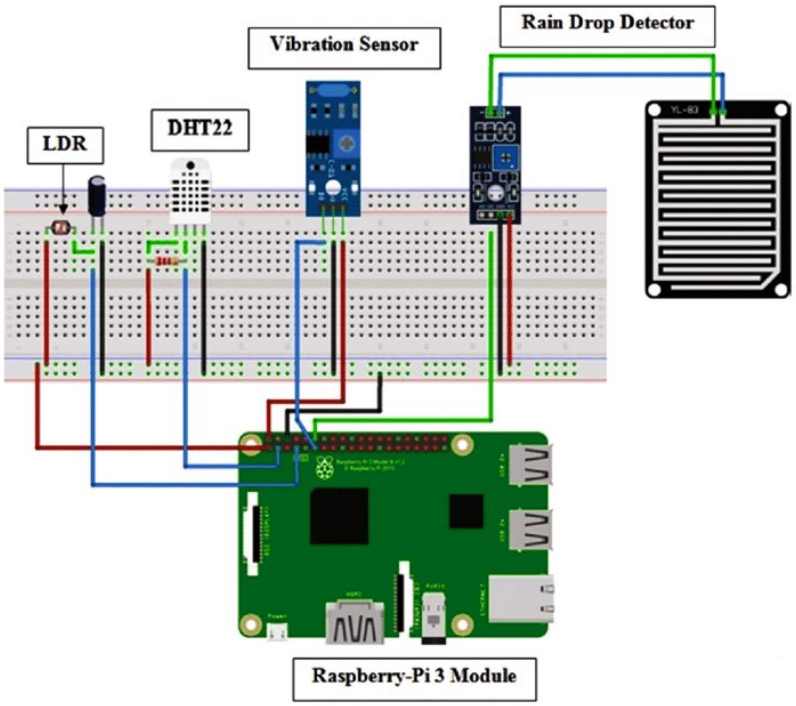
barometric pressure, and rain detection. This data is then stored locally using SQLite or uploaded to a cloud platform like ThingSpeak for remote access. Advanced visualization tools can be built using Python libraries, and optional web interfaces can be developed using frameworks like Flask or Django for remote monitoring.

The project involves three main components: hardware (sensors and Raspberry Pi), software (Python for programming and data storage), and the cloud (for data analysis and remote access). The complete workflow consists of sensor data collection, local/cloud storage, and optional data visualization via a web interface.

BLOCK DIAGRAM:



LAYOUT DIAGRAM:



PROTOTYPE DESIGN:

The prototype consists of three parts:

1. Hardware: This includes connecting sensors like DHT11/DHT22 for temperature and humidity, BMP180/BMP280 for barometric pressure, and a rain gauge module to the Raspberry Pi's GPIO pins. Optional sensors for wind speed and direction can also be connected.

2. Software: Python scripts are used to read sensor data and store it locally using SQLite or upload it to the cloud using ThingSpeak. Data is processed for visualization and analysis.

3. Control: The Raspberry Pi automates data collection and transmits it to the designated database, making the data accessible remotely or locally.

CODE:

```
from machine import Pin, ADC
import network
import time
import dht
from umqtt.simple import MQTTClient

# Blynk credentials
BLYNK_AUTH_TOKEN = "fO5Z_uUhGTAonkMwCZKYsi1J3CPMmce1"
BLYNK_SERVER = "blynk.cloud"
BLYNK_PORT = 1883

# WiFi credentials
SSID = "Airtel_Renukanthan"
PASSWORD = "Renu@0987"
```

```
# Pins configuration
DHT_PIN = 4
SOIL_PIN = 32
RAIN_PIN = 5
WATER_PIN = 18

# Initialize sensors
dht_sensor = dht.DHT11(Pin(DHT_PIN))
soil_moisture = ADC(Pin(SOIL_PIN))
soil_moisture.atten(ADC.ATTN_11DB)  # Full range for ESP32 ADC
rain_sensor = Pin(RAIN_PIN, Pin.IN)
water_level_sensor = Pin(WATER_PIN, Pin.IN)

# Connect to WiFi
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(SSID, PASSWORD)
    while not wlan.isconnected():
        print("Connecting to WiFi...")
        time.sleep(1)
    print("Connected to WiFi:", wlan.ifconfig())

# Function to send data to Blynk
def blynk_write(vpin, value):
    topic = f"v1/{BLYNK_AUTH_TOKEN}/update/{vpin}"
    msg = f"[{value}]"
    client.publish(topic, msg)

# Function to read sensors and send data
def read_sensors():
    try:
```

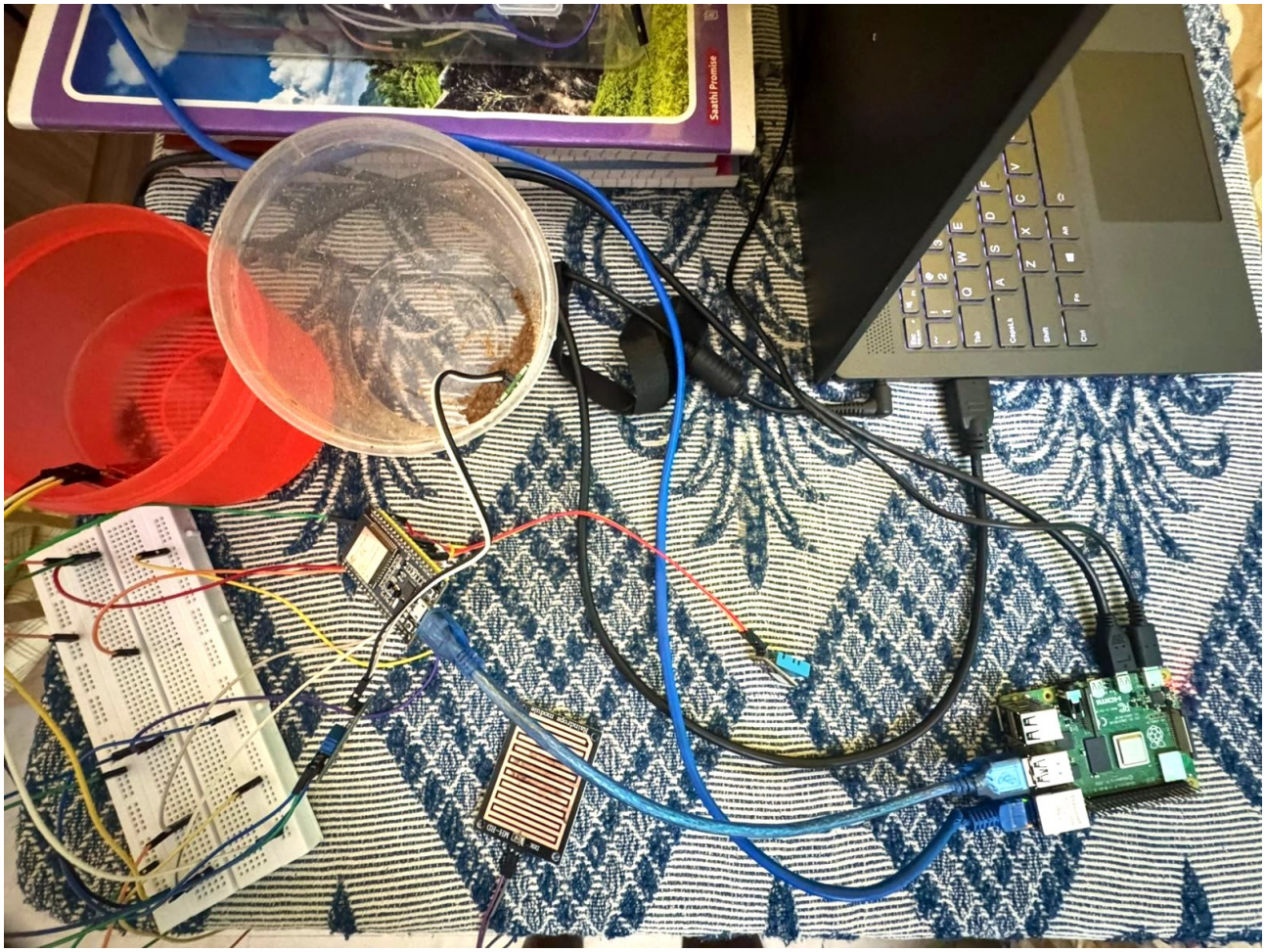
```
        # Temperature and Humidity from DHT11
        dht_sensor.measure()
        temperature = dht_sensor.temperature()
        humidity = dht_sensor.humidity()
        blynk_write(0, temperature)
        blynk_write(1, humidity)

        # Soil Moisture Sensor (Convert to percentage)
        soil_value = soil_moisture.read()
        soil_percentage = int((soil_value / 4095) * 100)
        blynk_write(2, soil_percentage)

        # Rain Sensor
        rain_status = rain_sensor.value()
        blynk_write(3, rain_status)

        # Water Level Sensor
        water_level = water_level_sensor.value()
        blynk_write(4, water_level)

        # Print values to the console for debugging
            print(f"Temperature: {temperature}°C, Humidity: {humidity}%, Soil Moisture: {soil_percentage}%, Rain: {rain_status}, Water Level: {water_level}")

    except Exception as e:
        print("Error reading sensors:", e)

# Setup MQTT client for Blynk
client = MQTTClient("esp32_client", BLYNK_SERVER, port=BLYNK_PORT, keepalive=60)
client.connect()

# Main program
```

```
connect_wifi()
while True:
    read_sensors()
```

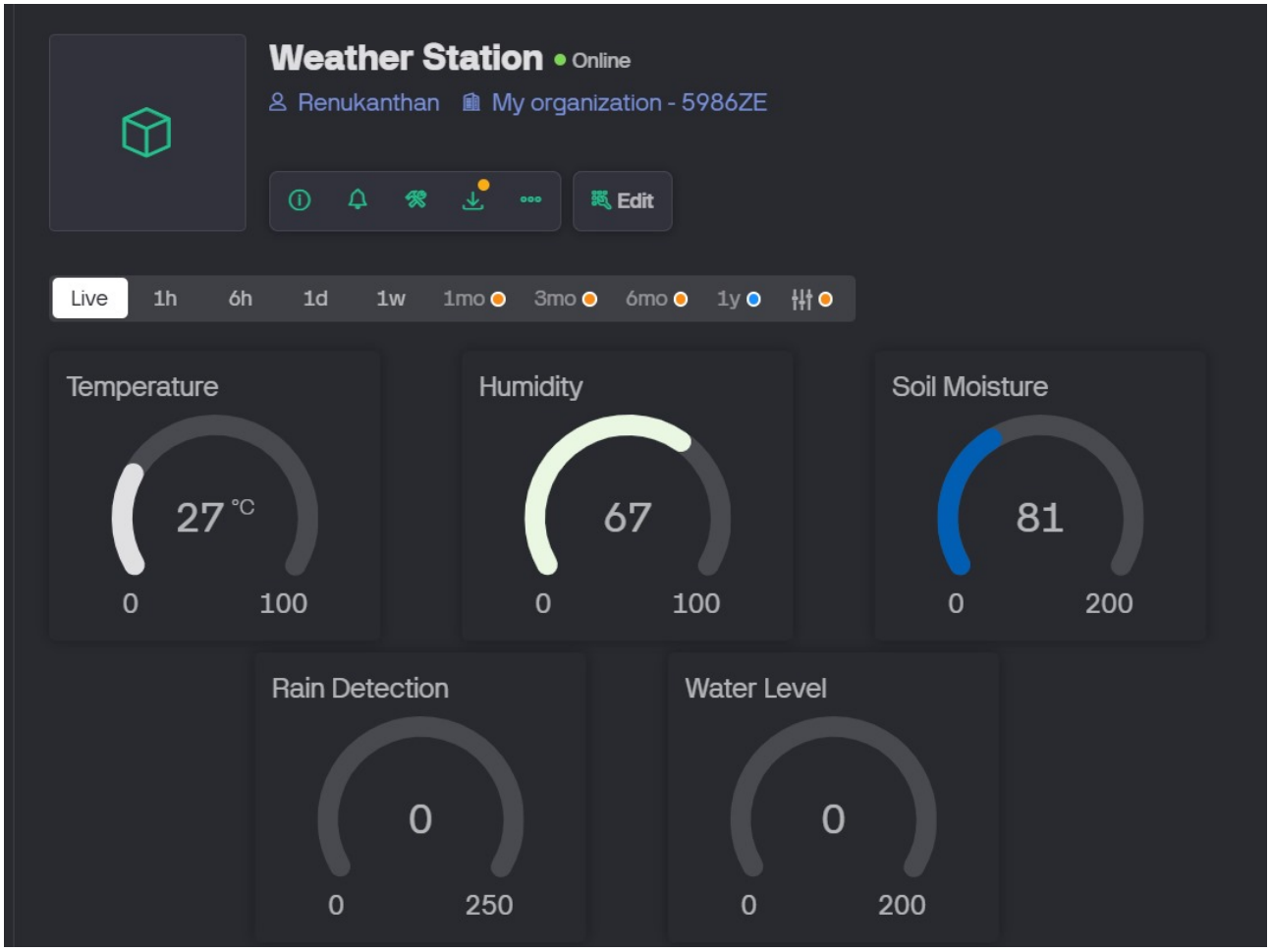
RESULTS AND DISCUSSION:

The developed IoT-based weather station prototype collects real-time weather data, including temperature, humidity, and barometric pressure. Data is stored locally in SQLite and optionally sent to the cloud (e.g., ThingSpeak) for remote monitoring. Real-time graphs are generated to visualize the data. The system allows for advanced analytics and remote monitoring, making it suitable for various weather monitoring applications.

CONNECTION SETUP:



OUTPUT:



RESULT:

The IoT-based weather station provides accurate real-time data collection, storage, and analysis. By using Raspberry Pi's computational power and cloud-based connectivity, the system can deliver advanced weather monitoring capabilities suitable for personal or commercial applications. Future developments can include more sensor nodes and advanced data analysis techniques for broader coverage.