VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590014.

Internship Project

On

"Smart Weather Forecasting"

Submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF ENGINEERING

By

1.ABHINAYA C H	**(1AP21CS002)**
2. ANKITHA V P	**(1AP21IS003)**
3. SINCHANA B R	**(1AP21CS043)**

Under the guidance of:

SAI CHARAN TEJA

2024 - 2025

A P S COLLEGE OF ENGINEERING

Anantha Gnana Gangothri,

NH-209, Kanakapura Road, Somanahalli, Bengaluru-560116.

TABLE OF CONTENTS

Chapter 1

Abstract

This project will focus on developing a weather forecasting system that integrates both hardware and software components. The system will use a DHT11 sensor for monitoring local temperature and humidity, while weather data for any specific city will be fetched using the Open cloud. The forecast data, including temperature, humidity, rain, and wind speed, will be displayed on a 16x2 LCD display. A web interface developed using Flask will allow users to input the city name and visualize the weather forecast for the next 24 hours. Additionally, an LED indicator will turn ON when the local temperature surpasses a predefined threshold. This project will serve as a comprehensive example of integrating IoT components with real-time data and web technologies.

Chapter 2

Introduction

The Weather Forecasting Application aims to provide real-time weather updates while integrating IoT components such as the DHT11 sensor, Raspberry Pi, and a 16x2 LCD screen. Weather forecasting plays a critical role in decision-making for industries such as agriculture, logistics, and home automation. This project will combine hardware for local weather monitoring and a cloud-based API to gather forecast data for specific cities. A Flask-based web interface will allow users to interact with the system, visualize the weather data, and analyse

Chapter 2.1

Objectives and Problem Statement

Objectives :

This project involves the development of a real-time weather forecasting system that integrates both hardware and software components. The system will use the Open cloud to fetch and display live weather data, including temperature, humidity, wind speed, and rain levels, for any user-specified city. Locally, a DHT11 sensor will monitor temperature and humidity, with the data displayed on an LCD screen for immediate viewing. To enhance functionality, an LED indicator will be triggered when the local temperature exceeds a pre-defined threshold, providing a visual alert. Additionally, the system will feature an interactive web interface, allowing users to input city names and access detailed weather information, including a 24-hour forecast. This combination of real-time sensor data, external API integration, and user-friendly interaction will provide a comprehensive weather monitoring and forecasting solution.

Problem Statement:

1. Lack of accurate weather information in remote areas can lead to ineffective decision-making.

2. Local weather monitoring systems often lack integration with real-time cloud-based data.

3. Limited knowledge of IoT components, such as sensors and web interfaces, inhibits students and enthusiasts from building practical projects.

4. There is a need for a user-friendly, cost-effective weather monitoring system that bridges local and global weather data.

Chapter 3

Applications

The project has a variety of real-world applications, such as:

1. Agriculture

- **Importance**: Farmers depend on accurate weather forecasts to plan agricultural activities like sowing, irrigation, fertilization, and harvesting.
- **Applications**:
  - Predicting rainfall helps determine the best time for sowing crops.
  - Temperature forecasts assist in protecting crops from extreme weather (e.g., frost, heatwaves).
  - Humidity predictions help prevent crop diseases caused by excessive moisture.
  - Reducing water wastage by scheduling irrigation during dry conditions.
- **Example**: Farmers can receive early warnings about storms or drought conditions to take protective measures.

2. Disaster Management

- **Importance**: Early weather forecasts help in disaster preparedness and minimizing damages.
- **Applications**:
  - Predicting **cyclones, hurricanes, storms**, or heavy rainfall helps authorities evacuate affected areas early.
  - Helps in identifying flood-prone zones and preparing resources for rescue operations.
  - Accurate data supports managing wildfires, landslides, or droughts.
- **Example**: Forecast systems alert communities before a cyclone, allowing time for evacuation and resource distribution.

3. Aviation and Transportation

- **Importance**: Weather conditions significantly affect air, sea, and land travel.
- **Applications**:
  - **Aviation**: Airlines depend on forecasts to plan safe flight paths and avoid turbulence, storms, or poor visibility.
  - **Maritime**: Ships rely on weather forecasts to avoid storms, rough seas, and navigate safely.
  - **Road Transport**: Accurate weather predictions help drivers prepare for fog, snow, heavy rain, or icy conditions.
- **Example**: Airports delay or reroute flights during severe thunderstorms or hurricanes to ensure passenger safety.

4. Power Generation and Renewable Energy

- **Importance**: Weather forecasts are vital for energy production, especially in renewable energy sectors like solar and wind.
- **Applications**:
  - Predicting sunlight hours helps optimize **solar energy generation**.
  - Wind forecasts help plan wind turbine operations and reduce wear during storms.
  - Hydropower plants rely on rainfall forecasts to manage water levels in reservoirs.
- **Example**: Energy companies use forecasts to balance electricity supply and demand.

5. Construction and Infrastructure

- **Importance**: Weather conditions play a significant role in construction activities.
- **Applications**:
  - Forecasting rain and storms helps schedule construction work during dry weather.
  - Reducing risks of structural damage caused by extreme weather.
  - Ensuring worker safety during adverse weather conditions.

- **Example**: Construction projects in flood-prone areas rely on weather systems to avoid delays.

6. Tourism and Outdoor Events

- **Importance**: Tourists and event organizers depend on weather forecasts to plan outdoor activities.
- **Applications**:
    - Tourists check forecasts for clear skies to plan vacations, hiking, and adventure sports.
    - Large outdoor events like concerts, weddings, and sports rely on accurate predictions to avoid rain interruptions.
- **Example**: Organizers reschedule cricket matches if heavy rainfall is predicted.

7. Smart Cities and IoT Systems

- **Importance**: Integrating weather data into smart systems ensures optimized city operations.
- **Applications**:
    - Managing **traffic systems** during heavy rains or fog.
    - Automating **smart irrigation** systems based on weather forecasts.
    - Enhancing **air quality monitoring** and pollution prediction.
- **Example**: Smart cities use weather predictions to automate water supply systems during droughts.

8. Personal Use

- **Importance**: Weather forecasts help individuals plan their daily activities.
- **Applications**:
    - Planning outdoor activities like travel, exercise, and picnics.
    - Preparing for clothing and gear based on the forecast (e.g., carrying umbrellas during rainy seasons).
    - Ensuring safety during severe weather (e.g., avoiding travel during storms).

- **Example**: Mobile weather apps notify individuals about rain or heatwaves for better daily planning.

9. Military Operations

- **Importance**: Military missions rely heavily on weather conditions for operational success.
- **Applications**:
  - Planning airstrikes, ground movements, or naval missions during clear weather.
  - Predicting sandstorms or fog that could impact visibility.
  - Using weather data for rescue and relief operations.
- **Example**: Military drones or aircraft are deployed when favorable weather is forecasted.

10. Environmental Monitoring

- **Importance**: Weather systems monitor changing environmental conditions over time.
- **Applications**:
  - Tracking climate changes like global warming and melting glaciers.
  - Monitoring air quality, pollution levels, and atmospheric conditions.
  - Supporting environmental research to predict future weather trends.
- **Example**: Researchers use weather data to study rising sea levels caused by temperature changes.

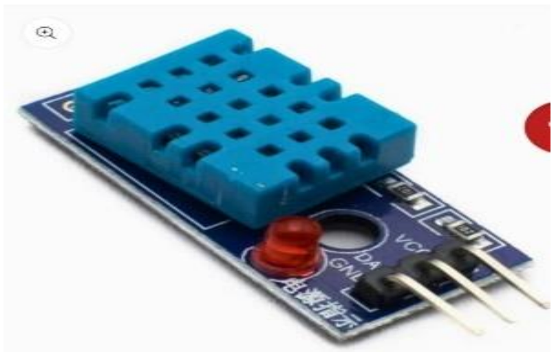
Chapter 4

Components Used

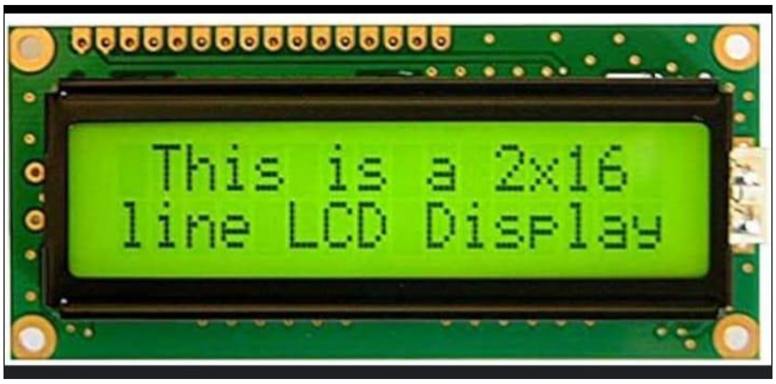
Hardware Components:

1. Raspberry Pi Acts as the main controller to interface with sensors, display, and APIs

2. DHT11 Sensor: Measures local temperature and humidity.



3. 16x2 LCD Display: Displays weather details such as temperature, humidity, rain, and wind speed.



4. LED: Indicator that turns ON when the temperature exceeds the set threshold.

5. Jumper Wires: Used to connect various hardware components.

Software Components:

1. Python: Programming language used for controlling the hardware and interfacing with APIs.

2. Flask: Web framework used to build the user interface for weather input and display.

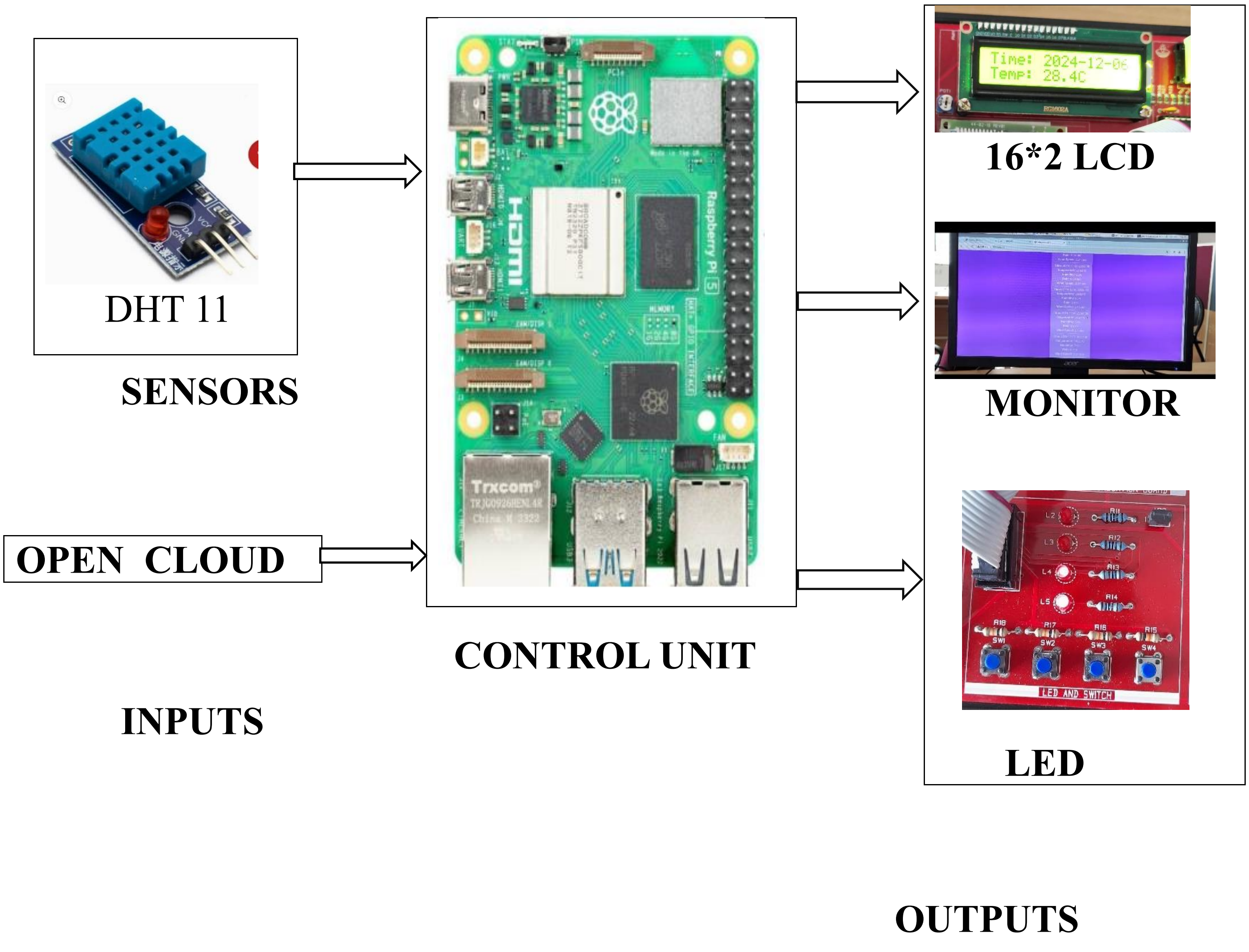
3. Open cloud : Provides real-time weather data for specific locations.

4. Adafruit_DHT Library: Reads data from the DHT11 sensor.

5. HTML & CSS: Used to design the web interface for the Flask application.

Chapter 5

Flow Chart



1. Sensors (DHT11)

- DHT11 is a temperature and humidity sensor.
- It collects real-time environmental data like temperature and humidity.
- The data from the DHT11 sensor is fed to the control unit (Raspberry Pi) for processing.

2. Open Cloud

- The Open cloud provides weather information like temperature, humidity, and other weather conditions from external cloud servers.
- This cloud fetches real-time weather updates (like current temperature and humidity) for comparison or display.
- The control unit retrieves this data via internet connectivity.

3. Control Unit (Raspberry Pi)

- The Raspberry Pi acts as the central processing unit of the system.
- It collects data from:
  - DHT11 Sensor (real-time temperature and humidity data).
  - Open Weather API (external weather updates).
- The control unit processes the inputs and sends the data to the outputs.

4. Outputs

The processed data is displayed or used in the following components:

a) 16x2 LCD Display

- A 16x2 LCD screen displays data like:
    - Current temperature and humidity.
    - Real-time weather information fetched from the sensor or Open Weather API.

b) Monitor

- The monitor displays the weather data on a larger screen.
- This could include:
    - Raw sensor data.
    - Real-time updates from the Open Weather API.
    - Any visualizations or system logs generated by the Raspberry Pi.

c) LED

- LEDs can be used as indicators for:
    - Threshold alerts (e.g., high temperature or humidity).
    - System status (e.g., normal operation, errors).
- The LEDs light up based on the conditions defined in the Raspberry Pi program.

Chapter 6

Conclusion

The Weather Forecasting Application was successfully implemented to monitor and display real-time weather data. The system fetched weather data from the OpenWeatherMap API and displayed it on both the 16x2 LCD display and the web interface. Additionally, the DHT11 sensor effectively measured local temperature and humidity, and an LED indicator was triggered when the temperature exceeded the threshold. The integration of hardware and software components provided a user-friendly and interactive solution for weather monitoring.

Chapter 7

Future Work

The following improvements can be implemented in future versions of the project:

1. Mobile Application Integration: Develop an Android or iOS app for remote weather monitoring.

2. Data Logging: Store weather data over time to analyze historical trends and patterns.

3. SMS/Email Alerts: Send automated alerts for extreme weather conditions.

4. Multiple Sensors: Integrate additional sensors to measure air quality, pressure, and other environmental parameters.

5. Improved Interface: Enhance the user interface to include graphs, charts, and interactive elements.

Chapter 8

Appendix

A. Hardware Connections:

- DHT11 Sensor: Connected to Raspberry Pi GPIO Pin 4.
- 16x2 LCD Display: Pins connected as follows:
  - RS: GPIO 15
  - E: GPIO 12
  - D4-D7: GPIO 23, 21, 19, 24
- LED: Connected to GPIO Pin 12.

B. Code Snippets:

1. Initialize GPIO and LCD Display:

```
python
def init_gpio():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(led2, GPIO.OUT, initial=0)
    lcd_init()
```

2. Fetch Weather Data:

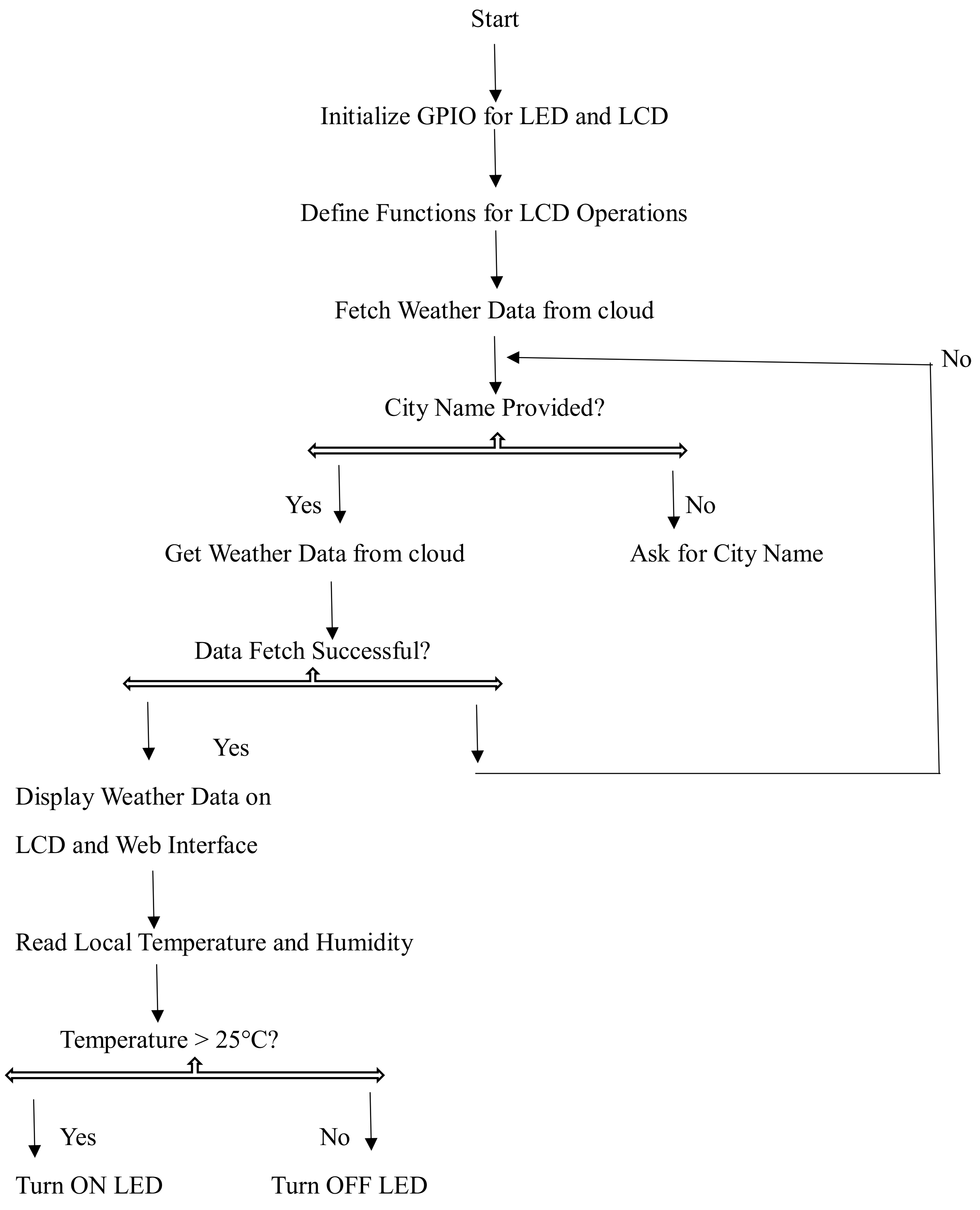
```
python
def get_weather_data(city_name):
    url = f"{BASE_URL}?q={city_name}&appid={API_KEY}&units=metric"
    response = requests.get(url)
 return response.json()
```

3. Web Interface (HTML):

```
html

<form action="/" method="POST">

   <label for="city">Enter City Name:</label>

   <input type="text" id="city" name="city" required>

   <button type="submit">Get Weather</button>

</form>
```

Chapter 8.1

PSEUDO CODE :



$$\begin{tikzcd}
{} \arrow[d] \\
\text{Display Temperature on LCD} \arrow[d] \\
\text{Main Program Running on Flask (Port 5000)} \arrow[d] \\
\text{Repeat Monitoring Local}
\end{tikzcd}$$

References

1. OpenWeatherMap API Documentation: https://openweathermap.org/api

2. Raspberry Pi GPIO Library Documentation: https://gpiozero.readthedocs.io

3. Flask Web Framework: https://flask.palletsprojects.com/