Microprocessor Based Systems

EMBEDDED AIR QUALITY SYSTEM
Assignment 2

Submitted By

Sai Teja Karanam - 30446182

Systems Engineering and Engineering Management
South Westphalia University of Applied Sciences - Soest

Submitted to **Prof. Dr. Dominik M. Aufderheide**

Date: 15 September 2024

Contents

1 Introduction

The main goal of the project is to develop an IoT system that monitors the Air Quality Index (AQI) by collecting data such as particulate matter (PM1, PM2.5, and PM10), temperature, and humidity using sensors. The collected data is transferred to a cloud service, ThingSpeak, for analysis and visualization. Additionally, key measurements such as AQI are displayed on a 7-segment display and LED matrix.

2 Hardware and Software

2.1 Hardware Components

- Raspberry Pi (Pi-Trokli)
- DHT11 Sensor for temperature and humidity
- PM Sensor for particulate matter measurement (PM1, PM2.5, PM10)
- 7-Segment Display
- LED Matrix Display

2.2 Software Libraries

- `requests` - for HTTP communication with ThingSpeak
- `json` - for processing data from ThingSpeak
- `time` - for time-related functions
- `matplotlib.pyplot` - for creating plots
- `numpy` - for numerical data handling
- `dht11` - for interfacing with the DHT11 sensor
- `RPi.GPIO` - for GPIO interaction on Raspberry Pi
- `thingspeak` - for communicating with ThingSpeak API

- `deque` from `collections` - for FIFO data buffering
- `Adafruit_LED_Backpack.SevenSegment` - for controlling the 7-segment display
- `luma.core.interface.serial.spi`, `luma.core.interface.serial.noop` - for SPI communication with LED matrix
- `luma.led_matrix.device.max7219` - for controlling the matrix LED display
- `luma.core.render.canvas` - for rendering visual elements on the matrix LED

2.3 Channel IDs and API Keys

- `channel_id = 2655368`
- `write_key = 'G5AQQ6LNS3SDOL76'`
- `read_key = 'POE3EBVUQYPJF2PC'`

3 Reading Sensor Cloud Data from ThingSpeak

3.1 Accessing the ThingSpeak Channel

The ThingSpeak cloud platform provides REST APIs for accessing data. Using the channel ID and API key, the latest PM measurements are fetched.

Listing 1: Fetching Data from ThingSpeak

```
import requests

def fetch_data_from_thingspeak(channel_id, api_key):
    url = f"https://api.thingspeak.com/channels/{channel_id}/fields/1.json?ap
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        return data
    else:
        print("Failed to retrieve data")
```

3.2 Decoding the Cloud Reading Using JSON

Once the data is fetched from ThingSpeak, it is decoded from JSON format to extract relevant PM values (PM1, PM2.5, and PM10).

Listing 2: Decoding JSON Data

```
def decode_data(json_data):
    entries = json_data['feeds']
    pm_values = []
    for entry in entries:
        pm_values.append({
            'PM1': entry['field1'],
            'PM2.5': entry['field2'],
            'PM10': entry['field3']
        })
    return pm_values
```

3.3 Visualizing PM Measurements and Mean Values

The collected PM data is visualized using the `matplotlib` library. For better accuracy, we compute the mean of the last 60 values.

Listing 3: Visualizing PM Data

```
import matplotlib.pyplot as plt

def plot_pm_data(pm_values):
    pm1 = [float(data['PM1']) for data in pm_values]
    pm25 = [float(data['PM2.5']) for data in pm_values]
    pm10 = [float(data['PM10']) for data in pm_values]

    plt.plot(pm1, label="PM1")
    plt.plot(pm25, label="PM2.5")
    plt.plot(pm10, label="PM10")
    plt.xlabel('Time')
    plt.ylabel('PM Concentration')
    plt.title('Particulate Matter Data')
    plt.legend()
```

```
plt.show()
```

4 Local Temperature and Humidity Measurements

4.1 Getting Temperature and Humidity Readings from Sensor

The DHT11 sensor provides temperature and humidity readings, which are processed and stored.

Listing 4: Reading from DHT11 Sensor

```
import dht11
import RPi.GPIO as GPIO

def get_temperature_humidity():
    instance = dht11.DHT11(pin=4)
    result = instance.read()
    if result.is_valid():
        return result.temperature, result.humidity
    else:
        print("Error reading from DHT11 sensor")
        return None, None
```

4.2 Implementation of FIFO Data Buffer

To smooth out fluctuations in the sensor readings, a FIFO buffer is implemented.

Listing 5: FIFO Data Buffer

```
from collections import deque

class FIFOBuffer:
    def __init__(self, size):
        self.buffer = deque(maxlen=size)

    def add(self, value):
        self.buffer.append(value)
```

```
    def get_average(self):
        return sum(self.buffer) / len(self.buffer)
```

5 AQI Calculation

5.1 Calculate the AQI for Each PM Measurement

The Air Quality Index (AQI) is calculated using pre-established breakpoints for PM2.5 and PM10.

Listing 6: AQI Calculation Function

```
def calculate_aqi(pm_value, breakpoints):
    for lower, upper, aqi_lower, aqi_upper in breakpoints:
        if lower <= pm_value <= upper:
            return (aqi_upper - aqi_lower) / (upper - lower) * (pm_value - lo
    return None
```

6 User Interface

6.1 Display Mean Value on the Seven Segment Display

The mean values of PM, temperature, and humidity are displayed on a 7-segment display.

Listing 7: Displaying on 7-Segment Display

```
from Adafruit_LED_Backpack import SevenSegment

def display_on_seven_segment(value):
    display = SevenSegment.SevenSegment(address=0x70)
    display.begin()
    display.print_number_str(str(value))
    display.write_display()
```

7 Conclusion

The IoT system for air quality monitoring efficiently collects and processes data from various sensors. This data is visualized in real-time on a 7-segment display and LED matrix while also being transmitted to the cloud for further analysis.

8 Reference

This project concept and code was co-created as a group with the following members: Karanam Sai Teja Muhammad Saad Majeed Umar Bin Ghayas Mohammad Imran