Fachhochschule Südwestfalen
University of Applied Sciences

Microprocessor Based Systems
Assignment 2

Assignment 2

Embedded Air Quality Measurement System

MSc. System Engineering and Engineering Management- SS 2023

Submitted to

Prof. Dr. Dominik Aufderheide

Submitted by

Deep Manishbhai Pathak

(30369250)

Microprocessor Based Systems
Assignment 2

Table of Contents:

SR NO.	Contents	Page no.
1	Declaration	3
2	**Task-1-Reading Sensor Cloud Data from Thingspeak**	4
2.1	Accessing the thingspeak channel	4
2.2	Decoding the cloud reading by using JSON	4
2.3	Visualizing PM measurements and mean values	5
3	**Task-2-Local Temperature and Humidity Measurement**	6
3.1	Getting Temperature and Humidity Readings from Sensor	7
3.2	Implementation of FIFO data buffer	7
3.3	Visualizing temperature and humidity measurements and mean values	8
4	**Task-3-AQI calculation**	10
4.1	Calculate the AQI for each PM measurement	10
4.2	Transfer AQI levels to a new cloud channel	11
5	**Task-4-User interface**	12
5.1	Selection of measurement value to be visualized	13
5.2	Display mean values and visualization	15
6	**Task-5-Overall program structure**	16
6.1	Updating channels every minute.	16
6.2	Taking sensor data every 10 seconds.	16
6.3	AQI calculation updated every minute.	16

Fachhochschule Südwestfalen
University of Applied Sciences

Microprocessor Based Systems
Assignment 2

Declaration

I hereby declare that this submission is my own original work and, to the best of my knowledge, contains no material previously published or written by another person, except where due acknowledgment has been made in the text. I have fully cited and referenced all material that is not original to this work. I am well aware of the fact that in accordance with German law, plagiarism is a punishable offense

Date:
31/08/2023

Microprocessor Based Systems
Assignment 2

Task-1-Reading Sensor Cloud Data from Thingspeak

2.1-Accessing the Thingspeak channel.

Firstly we have to take the data from the cloud for PM readings as well as the temperature and humidity readings this can be done by accessing the channel and decoding the json message and storing it in the array by creating different fields. arrays.

```
# Import necessary packages
import requests
import matplotlib.pyplot as plt
import numpy as np
from statistics import mean
import RPi.GPIO as RPi
import dht11
import time

# Initialize GPIO
RPi.setwarnings(False)
RPi.setmode(RPi.BCM)

# Import JSON data from Thingspeak
clouddata=requests.get('https://thingspeak.com/channels/343018/feed.json)
clouddata = str(clouddata.text)
```

Pseudo Code - For Accessing the Channel-343018

2.2-Decoding the cloud reading by using JSON

We will now load and decode the data using JSON and append the data values into the created fields by using the while loop, after which we will print the given data.

```
# Initialize empty arrays for each field
field_1, field_2, field_3, field_4, field_5, field_6, field_7 = [], [], [], [],
[], [], []
# Loop through the feeds and populate field arrays
for i = 0 to 99:
    field_1.append(feeds[i]['field1'])
    field_2.append(feeds[i]['field2'])
    field_3.append(feeds[i]['field3'])
    field_4.append(feeds[i]['field4'])
    field_5.append(feeds[i]['field5'])
    field_6.append(feeds[i]['field6'])
    field_7.append(feeds[i]['field7'])
```

Microprocessor Based Systems
Assignment 2

```
# Print values in each field
Print "Field 1:", field_1
Print "Field 2:", field_2
Print "Field 3:", field_3
Print "Field 4:", field_4
Print "Field 5:", field_5
Print "Field 6:", field_6
Print "Field 7:", field_7
```

Pseudo Code - For Appending the JSON data into field arrays

2.3-Visualizing PM measurements and mean values

To visualize the extracted data from the cloud, we need to calculate the mean values for fields 1, 2, and 3. The Pseudo code demonstrates how to do this by filtering out any invalid values using an exception if statement.

Once the data is filtered, the mean values for PM1, PM2.5, and PM10 will be determined. These values will be used to create three separate graphs using the matplotlib function. The first graph will represent the PM1 data, the second graph will show PM2.5 data, and the third graph will display PM10 data

```
# Calculate and print mean values for each field

for field_num from 1 to 3:
    filtered_field = []
    for val in globals()["field_" + field_num]:
        if val is not None:
            filtered_field.append(float(val))

    if filtered_field:
        mean_val = mean(filtered_field)
        Print("Mean of field", field_num, ":", mean_val)
    else:
        Print("No valid data in field", field_num)

# Visualize PM values and mean values
for field_num from 1 to 3:
    Create a subplot axs[field_num-1]
    Plot PM values using globals()["field_" + field_num]
    Plot a dashed line at the corresponding mean value
    Set labels and title
```

Pseudo Code - For calculating mean values and plotting the fields.

Microprocessor Based Systems
Assignment 2

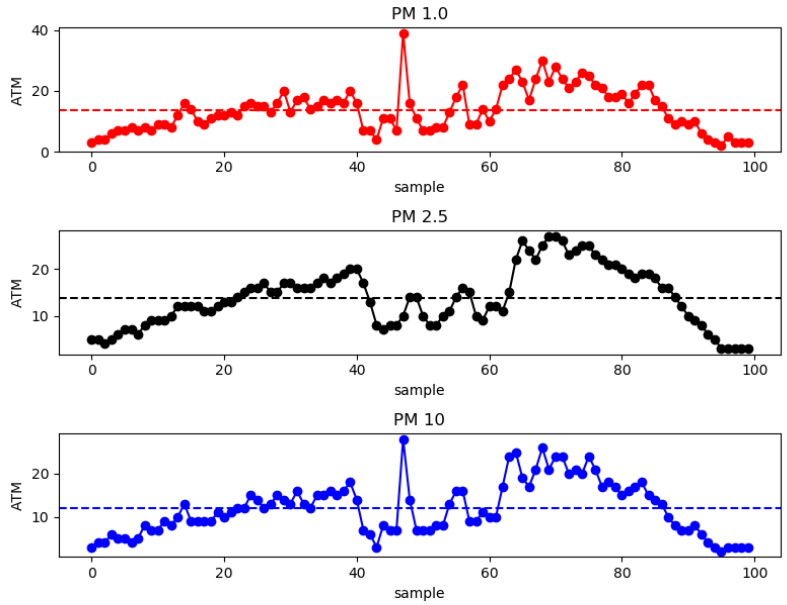

Figure-1.1-Visualized PM data along with the mean horizontal value.

3-Task-2-Local Temperature and Humidity Measurement

In this task, we have collect the local temperature and humidity readings from the DTH11 sensor and then append them to the cloud readings of temperature and humidity. The readings from the cloud has to be converted to the degree celsius which will be done by using a simple for loop.

```
# Calculate and print mean value for field6 (Temperature)
filtered_field_6 = []
for val in field_6:
    if val is not None:
        filtered_field_6.append(float(val))
if filtered_field_6:
    field6_mean = mean(filtered_field_6)
    Print("Mean of field6 value is", field6_mean)
else:
    Print("No valid data in field6")

# Convert Fahrenheit to Celsius for field6
new_field_6_celsius = []
for val in field_6:
    if val is not None:
        celsius_temp = (val - 32) * 0.555
        new_field_6_celsius.append(celsius_temp)
```

Microprocessor Based Systems
Assignment 2

```
# Calculate and print mean value for field7 (Humidity)
filtered_field_7 = []
for val in field_7:
    if val is not None:
        filtered_field_7.append(float(val))
if filtered_field_7:
    field7_mean = mean(filtered_field_7)
    Print("Mean of field7 value is", field7_mean)
else:
    Print("No valid data in field7")

# Assign humidity values
humidity = field_7

# Print converted temperature and humidity values
Print("The degree Celsius temperature is", new_field_6_celsius)
Print("The humidity is", humidity)
```

Pseudo Code to convert the cloud field data to degree celsius.

3.1-Getting Temperature and Humidity Readings from Sensor

This can be done by using the Rpi library and the DTH11 sensor to collect the temperature and humidity readings.

```
#local temperature and humidity values
while True:
    instance=dht11.DHT11(pin=4)
    result=instance.read()
    while not result.is_valid():
        result=instance.read()
    temp=result.temperature
    hum=result.humidity
    print("temperature is {0} and humidity is {1}".format(temp,hum))
```

Pseudo Code - Extracting the local temp and humidity values using DTH11 sensor

3.2-Implementation of FIFO data buffer.

```
#fifo buffer
    if len(new_field_6_celsius)>100:
        new_field_6_celsius.pop(0)
    if len(humidity)>100:
        humidity.pop(0)
```

Microprocessor Based Systems
Assignment 2

3.3-Visualizing temperature and humidity measurements and mean values

Previously, we converted the temperature data from the cloud into degrees Celsius. Following that, we collected 100 sets of temperature and humidity values using the DTH11 sensor. These values were then appended to their respective fields.

Now, we will visualize this data using the matplotlib library. By creating appropriate plots.

```
# Calculate average temperature
avg_temp = mean(new_field_6_celsius)

# Calculate average humidity
avg_hum = mean(humidity)

# Visualize temperature, humidity, and mean values
Create a subplot grid with 5 rows and 1 column, with a size of 8x10

# Subplot for temperature
Select the 4th subplot (axs[3])
Plot the Celsius temperature values from new_field_6_celsius using red ("ro") markers
Plot a dashed horizontal line at the average temperature using a red color
Set the y-axis label to "Celsius temperature"
Set the title to "Temperature"

# Subplot for humidity
Select the 5th subplot (axs[4])
Plot the humidity values from humidity using black ("ko") markers
Plot a dashed horizontal line at the average humidity using a black color
Set the y-axis label to "Humidity"
Set the title to "Humidity"

Display the plot
Close the plot

Exit the loop
```

Pseudo Code - Matplotlib vizualization code.

Microprocessor Based Systems
Assignment 2

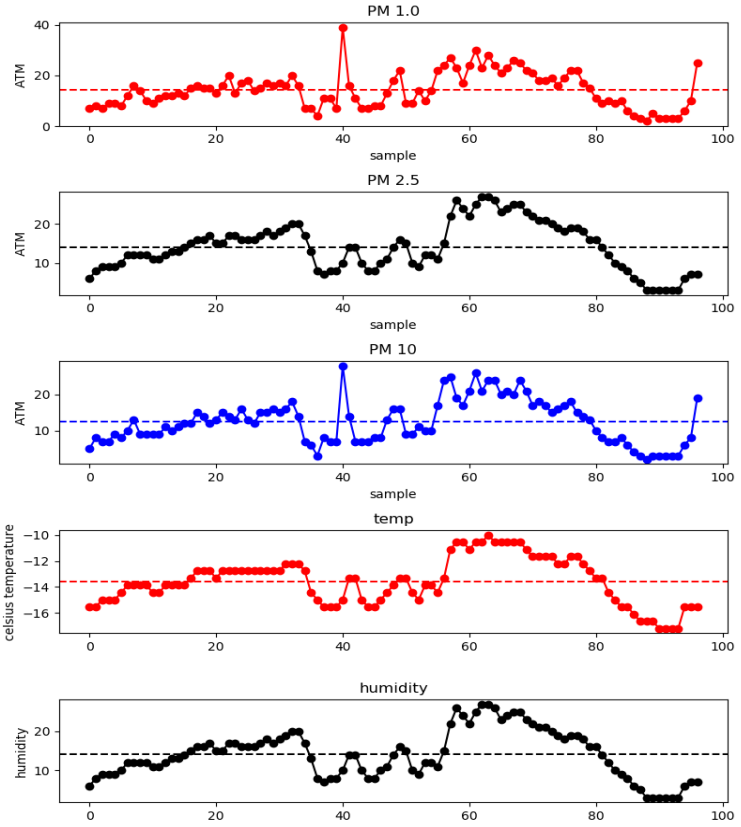

Figure-1.2-Output of the above code.

Microprocessor Based Systems
Assignment 2

Task-3-AQI calculation

4.1-Calculate the AQI for each PM measurement

As given in the assignment calculation of aqi can be easily done by the table which was giving the range of pm values with each corresponding AQI level.so we will look up into table and form a code which divides each pm values with respective AQI level.

```
# Import field_2 and field_3 from file1
Import field_2 from file1
Import field_3 from file1

# Import necessary libraries
Import thingspeak
Import time

# Define the new ThingSpeak channel details
channel_id = 2227426
write_key = '5RZ6KVVYHQLKAZ92'
channel = thingspeak.Channel(id=channel_id, api_key=write_key)
```

Pseudo code -Making new thingspeak channel.

```
Function AQI_pm25(x):
    If 0 <= x <= 12.0:
        AQI_Formula = (((50 - 0) * (x - 0)) / (12 - 0)) + 0
    Else If 12.1 <= x <= 35.4:
        AQI_Formula = (((100 - 51) * (x - 12.1)) / (35.4 - 12.1)) + 51
    Else If 35.5 <= x <= 55.4:
        AQI_Formula = (((150 - 101) * (x - 35.5)) / (55.5 - 35.5)) + 101
    Else If 55.5 <= x <= 150.4:
        AQI_Formula = (((200 - 151) * (x - 55.5)) / (150.4 - 55.5)) + 151
    Else If 150.5 <= x <= 250.4:
        AQI_Formula = (((300 - 201) * (x - 150.5)) / (250.4 - 150.5))+201
    Else If 250.5 <= x <= 350.4:
        AQI_Formula = (((400 - 301) * (x - 250.5)) / (350.4 - 250.5))+301
    Else If 350.5 <= x <= 500.4:
        AQI_Formula = (((500 - 401) * (x - 350.5)) / (500.4 - 350.5))+401
    AQI_Formula = Format AQI_Formula with two decimal places
    Return AQI_Formula
```

Pseudo code-Function for AQI-2.5 calculation based upon the table

Microprocessor Based Systems
Assignment 2

```
Function AQI_pm10(x):  # for PM10
    If 0 <= x <= 54.0:
        AQI_Formula = (((50 - 0) * (x - 0)) / (54 - 0)) + 0
    Else If 55 <= x <= 154:
        AQI_Formula = (((100 - 51) * (x - 55)) / (154 - 55)) + 51
    Else If 155 <= x <= 254:
        AQI_Formula = (((150 - 101) * (x - 155)) / (254 - 155)) + 101
    Else If 255 <= x <= 354:
        AQI_Formula = (((200 - 151) * (x - 255)) / (354 - 255)) + 151
    Else If 355 <= x <= 424:
        AQI_Formula = (((300 - 201) * (x - 355)) / (424 - 355)) + 201
    Else If 425 <= x <= 504:
        AQI_Formula = (((400 - 301) * (x - 425)) / (504 - 425)) + 301
    Else If 505 <= x <= 604:
        AQI_Formula = (((500 - 401) * (x - 505)) / (604 - 505)) + 401
    AQI_Formula = Format AQI_Formula with two decimal places
    Return AQI_Formula
```

Pseudo code -Function for AQI-10 calculation based upon the table

4.2-Transfer AQI levels to a new cloud channel

I have implemented the following code to determine the maximum AQI value between the AQI of 2.5 and 10. By comparing the AQI values of PM2.5 and PM10, we identify which one has a higher AQI level.Subsequently, i have uploaded all three fields (Maimum AQI, PM2.5, and PM10) to a newly created channel.

```
# Initialize a list to store maximum AQI values
Maximum_AQI = []

# Loop through the first 100 elements in aqi_pm25 and aqi_pm10
For z from 0 to 98:
    If aqi_pm25[z] < aqi_pm10[z]:  # If PM2.5 AQI is less than PM10 AQI
        v = aqi_pm10[z]  # Assign PM10 AQI value to v
    Else:  # If PM10 AQI is less than PM2.5 AQI
        v = aqi_pm25[z]  # Assign PM2.5 AQI value to v
    Append v to Maximum_AQI list
```

Microprocessor Based Systems
Assignment 2

```
    # Update cloud channel with AQI values for PM2.5, PM10, and Maximum
AQI
    Update channel with fields {'field1': aqi_pm25[z], 'field2':
aqi_pm10[z], 'field3': Maximum_AQI[z]}

    # Pause briefly to avoid excessive updates
    Sleep for 0.25 seconds
```

Pseudo code - For maximum AQI and updating the channel.

Task-4-User interface

Note : This part has been jointly done by me and my friend jigar shinde(30371835)

The user interface has to perform the following task:

1)- The first row of led matrix should show the current measurement on display.

A) – PM1.0

B) – PM2.5

C) – PM10.0

D) – Temperature

E) – Humidity

2)-Left and right buttons should be able to navigate through this.

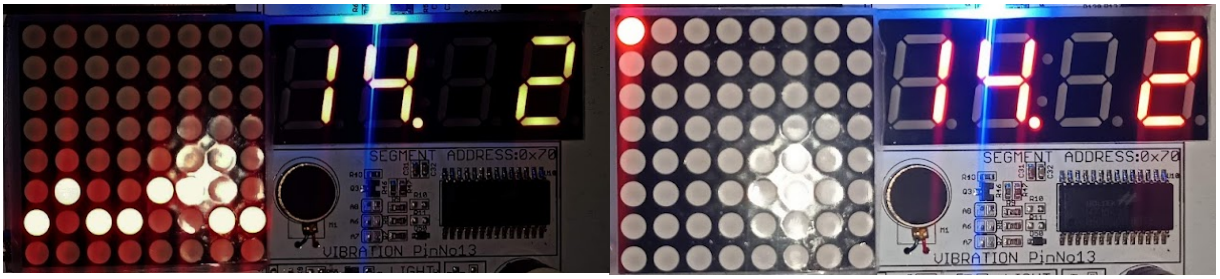

Figure-1.3-LED matrix and seven-segment representation.

Microprocessor Based Systems
Assignment 2

```
# Import necessary libraries and values from file1
Import RPi.GPIO as GPIO
Import spi from luma.core.interface.serial
Import max7219 from luma.led_matrix.device
Import canvas from luma.core.render
Import sleep from time
Import SevenSegment from Adafruit_LED_Backpack

# Import specific values from file1
Import field1_mean, humidity, field2_mean, field3_mean, avg_temp, avg_hum,
field_1, field_2, field_3, new_field_6_celsius from file1

# Initialize LED matrix and 7-segment display
Initialize serial using spi with port=0, device=1, gpio=noop()
Initialize device using max7219 with serial, cascaded=1,
block_orientation=90, rotate=0
Initialize segment_7SD using SevenSegment.SevenSegment with address=0x70
Call segment_7SD.begin()

# Initialize GPIO with RPi.GPIO
Assign lft_btn as 25
Assign rgt_btn as 19
Set GPIO mode to BCM
Setup lft_btn GPIO pin as input with pull-up off
Setup rgt_btn GPIO pin as input with pull-up off

Assign p as 0  # Pointer for iteration
Assign r as [0, 1, 2, 3, 4]  # Array for switching the values
```

Pseudo - Code for user interface

5.1-Selection of measurement value to be visualized

The 'p' pointer acts as a selector for displayed information. By pressing buttons, users navigate through predefined measurements like PM1, PM2.5, PM10, cloud average temperature, and humidity. Each 'p' value corresponds to a specific measurement type. LED matrices and the 7-segment display visually present these measurements, while bar graphs summarize trends visually.

Microprocessor Based Systems
Assignment 2

```
# Initialize 'p' for selection
p = 0

# Infinite loop for continuous updating and display
while True:
    Display("PM1 value - press button to change for 2 sec", p)

    VisualizeSelection(p)  # Visualize selection on LED Matrix Display
(MLD)
    Wait(0.5)  # Pause for a moment

    # Check button presses for navigation
    if RightButtonPressed():
        p = IncrementSelection(p)
        Print("right button is pressed!")
    elif LeftButtonPressed():
        p = DecrementSelection(p)
        Print("left button is pressed!")

    # Visualize and process based on 'p'
    if p == 0:
        DisplayPM1ValueAndBargraph()
        Wait(0.8)
    elif p == 1:
        DisplayPM2_5ValueAndBargraph()
        Wait(0.8)
```

Pseudo -Code for user interface

This code makes a screen that changes based on what you pick. It starts by showing a choice on the screen, like a highlighted option. Then, after a short pause, if you press the buttons, the choice changes. If you press right, the choice goes up from 0 to 4. If you press left, the choice goes down from 4 to 0.

After that, the code does different things based on the choice. When the choice is 0 or 1, it shows and calculates some data about PM1 and PM2.5. The numbers are shown on a small screen and as bars on the display. There's a little wait between each update.

This code lets you pick what you want to see on the screen, and it changes accordingly. It's a way to interact with the data and see it in different forms.

Microprocessor Based Systems
Assignment 2

```
if p == 2:
    mean_PM10 = CalculateMean(field3_mean)
    DisplayOn7Segment(mean_PM10)
    VisualizeBarGraph(field_3)

elif p == 3:
    mean_temp = CalculateMean(avg_temp)
    DisplayOn7Segment(mean_temp)
    VisualizeBarGraph(new_field_6_celsius)

elif p == 4:
    mean_hum = CalculateMean(avg_hum)
    DisplayOn7Segment(mean_hum)
    VisualizeBarGraph(humidity)

Sleep(0.8)
```

Pseudo code -Code for user interface

5.2-Display mean values and visualization

When 'p' is 2, the code displays and shows data for PM10 (Particulate Matter 10). It calculates the average PM10 value and shows it on a 7-segment display. Additionally, a bar graph of PM10 data is visualized using the LED Matrix Display (MLD). There's a brief delay between each update.

If 'p' is 3, the code presents data for cloud average temperature. It calculates and displays the average temperature in Celsius on the 7-segment display. The temperature data is also shown as a bar graph on the LED Matrix Display (MLD). A small pause is inserted between updates.

Finally, when 'p' is 4, the code handles data for cloud average humidity. It calculates and displays the average humidity value on the 7-segment display. Similar to previous cases, the humidity data is shown as a bar graph on the LED Matrix Display (MLD), with a short delay between updates.

Microprocessor Based Systems
Assignment 2

Task-5-Overall program structure

Note : This part has been jointly done by me and my friend jigar shinde(30371835).

For the overall program we have combined all the task above such that:

1. The reading and writing channels gets updated every minute.
2. The sensor data is taken every 10 seconds.
3. AQI Calculations are updated every minute

```
Initialize time counter (t) to 0

while True:
    if t % 60 == 0:
        Fetch cloud data from Thingspeak
        Extract field values for various measurements
        Calculate mean values for PM1, PM2.5, PM10, and cloud temperature
        Convert cloud temperature to Celsius
        Calculate and update AQI values for PM2.5 and PM10
        Update cloud channel with AQI values
        Increment time counter by 1

    if t % 10 == 0 or t == 1:
        Update local temperature and humidity data
        Visualize temperature, humidity, and mean values
        Calculate AQI values for PM2.5 and PM10 based on local data
        Update cloud channel with AQI values
        Increment time counter by 1
Increment time counter by 1
```

Pseudo code - Code for user interface

6.1-Updating channels every minute.

we formulated field arrays. Subsequent to this, we decoded JSON data and appended it to these arrays. After displaying the values and calculating their means, these mean values will be utilized in the visualization phase.

6.2-Taking sensor data every 10 seconds.

The initial step involves establishing a condition for the variable 't'. If this condition holds true, the sensor will proceed to gather the local temperature and humidity data. Subsequently, this acquired data will be stored within a FIFO buffer, followed by the visualization of the stored information.

6.3-AQI calculation updated every minute.

we collected the AQI 2.5 and AQI 10 arrays. We then proceeded to analyze their values, determining the highest value among them, which was subsequently included in the maximum array. With this completed, we updated the channel by integrating these three arrays.