AN PRESENTATION ON AIR QUALITY MONITORING

SUBMITTED BY

K.Badri Prasath

P.Ganapathi

K.Anjali

P.Indhumathi

K.Karthika

Introduction to Python

Definition:

Python is a high-level, versatile, and widely-used programming language known for its simplicity, readability, and ease of learning.

It was created by Guido van Rossum and first released in 1991. Python is dynamically typed and supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Key Features:

1.Readability:

- Python emphasizes clean and readable code, making it easier for programmers to express concepts in fewer lines of code compared to languages like C++ or Java.

2.Interpreted Language:

- Python is an interpreted language, meaning it does not require compilation before running. This allows for quicker development and testing cycles.

3. Dynamic Typing:

- Python is dynamically typed, which means you don't need to specify variable types. The interpreter infers types at runtime, providing flexibility but requiring careful attention to variable assignments.

4.Rich Standard Library:

- Python comes with a vast standard library that provides modules and packages for a wide range of tasks, from file I/O to web scraping, making it a powerful tool for various applications.

5.Platform Independence:

- Python is a cross-platform language, meaning code written in Python can run on different operating systems, including Windows, macOS, Linux, and more.

6. Object-Oriented:

- Python supports object-oriented programming (OOP) principles, allowing for the creation of reusable and organized code through classes and objects.

7.Community and Ecosystem:

- Python has a large and active community of developers who contribute to the language's development, provide support, and create third-party libraries and frameworks.

Common Use Cases:

1. Web Development:

- Python is used for both back-end and front-end development. Popular web frameworks like Django and Flask simplify building web applications.

2. Data Science and Analytics:

- Python is a go-to language for data analysis and manipulation. Libraries like NumPy, Pandas, Matplotlib, and TensorFlow are widely used in data

science projects.

3.Artificial Intelligence and Machine Learning:

- Python's simplicity and powerful libraries, along with frameworks like TensorFlow and PyTorch, make it a preferred choice for AI and ML applications.

4.Automation and Scripting:

- Python's ease of use and extensive standard library make it excellent for automating repetitive tasks and writing scripts.

5.Scientific Computing:

- Python is used in scientific research and engineering applications for tasks like simulations, numerical analysis, and complex calculations.

6.Networking and Systems Administration:

- Python's robust libraries and modules like `socket` and `paramiko` make it well-suited for network programming and systems administration

tasks.

Python's versatility, combined with its supportive community and extensive ecosystem, has contributed to its widespread adoption across various domains of software development and beyond. It remains one of the most popular and versatile programming languages in the world.

AIR QUALITY MONITORING USING RASPBERRY PI AND PYTHON

INTRODUCTION

Air is the very stuff we breathe. It ' s about 78% nitrogen, 21% oxygen, and 1% argon, and then there' s the assorted 'other' bits and pieces – many of which have been spewed out by humans and our related machinery.

Carbon dioxide is obviously an important polluter for climate change, but there are other bits we

should be concerned about for our health, including particulate matter.

This is just really small bits of stuff, like soot and smog. They're grouped together based on their size – the most important, from a health perspective, are those that are smaller than 2.5 microns in width (known as PM2.5), and PM10, which are between 10 and 2.5 microns in width. This pollution is linked with respiratory illness, heart disease, and lung cancer.

Creating a full-fledged IoT project with complete source code and output is beyond the scope of a single response. However, I can guide you through the process and provide a simplified example code for a basic air quality monitoring system using a Raspberry Pi and a PM2.5 sensor.

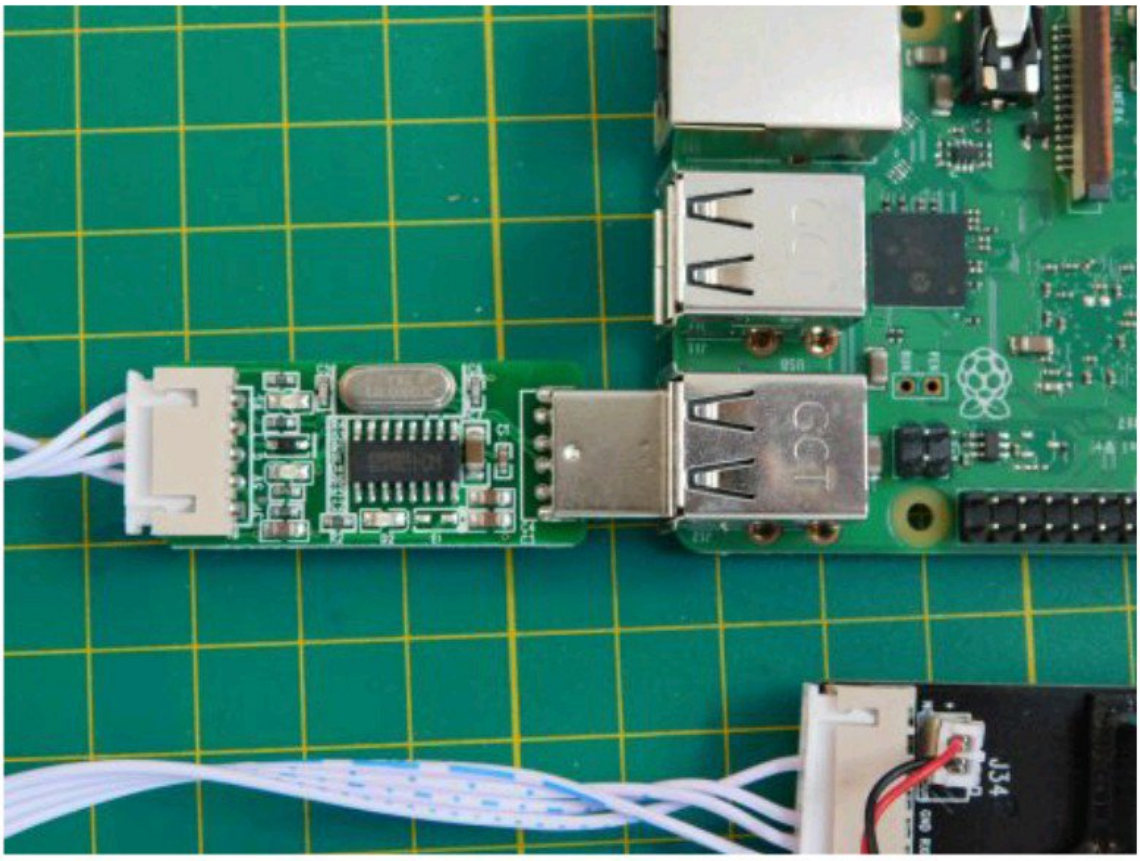
Components Needed

- Raspberry Pi
- PM2.5 sensor (e.g., SDS011)
- Jumper wires
- Internet connection

Getting started

We picked the SDS011 sensor for our project (see ‘Picking a sensor’ below for details on why). This sends output via a binary data format on a serial port.

You can read this serial connection directly if you’re using a controller with a UART, but the sensors also usually come with a USB-to-serial connector, allowing you to plug it into any modern computer and read the data.



The USB-to-serial connector makes it easy to

connect the sensor to a computer

First, you'll need a Raspberry Pi (any version) that's set up with the latest version of Raspbian, connected to your local network, and ideally with SSH enabled.

The wiring for this project is just about the simplest we'll ever do: connect the SDS011 to the Raspberry Pi with the serial adapter, then plug the Raspberry Pi into a power source.

Before getting started on the code, we also need to set up a data repository. You can store your data wherever you like – on the SD card, or upload it to some cloud service. We've opted to upload it to Adafruit IO, an online service for storing data and making dashboards.

We'll use Python 3 for our code, and we need two modules – one to read the data from the SDS011 and one to push it to Adafruit IO. You can install this by entering the following commands in a terminal:

pip3 install pyserial adafruit-io

You'll now need to open a text editor and enter the following code:

```
import serial, time
from Adafruit_IO import Client
aio = Client('your-adafruit-username', 'your- adafruit-key')

ser = serial.Serial('/dev/ttyUSB0')

while True:
    data = []
    for index in range(0,10):
        datum = ser.read()
        data.append(datum)

    pmtwofive = int.from_bytes(b''.join(data[2:4]), byteorder='little') / 10
    aio.send('kingswoodtwofive', pmtwofive)
    pmten = int.from_bytes(b''.join(data[4:6]), byteorder='little') / 10
    aio.send('kingswoodten', pmten)
    time.sleep(10)
```

We're interested in bytes 2 and 3 for PM2.5 and 4 and 5 for PM10. We convert these from bytes to integer numbers with the slightly confusing line:

pmtwofive = int.from_bytes(b''.join(data[2:4]), byteorder='little') / 10

from_byte command takes a string of bytes and converts them into an integer. However, we don't have a string of bytes, we have a list of two bytes, so we first need to convert this into a string.

The b'' creates an empty string of bytes. We then

use the join method of this which takes a list and joins it together using this empty string as a separator.

As the empty string contains nothing, this returns a byte string that just contains our two numbers. The byte_order flag is used to denote which way around the command should read the string.

We divide the result by ten, because the SDS011 returns data in units of tens of grams per metre cubed and we want the result in that format aio.send is used to push data to Adafruit IO.

The first command is the feed value you want the data to go to. We used kingswoodtwofive and kingswoodten, as the sensor is based in Kingswood. You might want to choose a more geographically relevant name. You can now run your sensor with:

python3 airquality.py

...assuming you called the Python file airquality.py and it's saved in the same directory the terminal's in.

At this point, everything should work and you can set about running your sensor, but as one final point, let's set it up to start automatically when you turn the Raspberry Pi on. Enter the command:

crontab -e

...and add this line to the file:

@reboot python3 /home/pi/airquality.py

With the code and electronic setup working, your sensor will need somewhere to live. If you want it outside, it'll need a waterproof case (but include

some way for air to get in). We used a Tupperware box with a hole cut in the bottom mounted on the wall, with a USB cable carrying power out via a window. How you do it, though, is up to you.

Now let's democratise air quality data so we can make better decisions about the places we live.

Picking a sensor

There are a variety of particulate sensors on the market. We picked the SDS011 for a couple of reasons. Firstly, it's cheap enough for many makers to be able to buy and build with.

Secondly, it's been reasonably well studied for accuracy. Both the hackAIR and InfluencAir projects have compared the readings from these sensors with more expensive, better-tested sensors, and the results have come back favourably.

Safe levels

Once you're monitoring your PM2.5 data, what

should you look out for? The World Health Organisation air quality guideline stipulates that PM2.5 not exceed 10 µg/m3 annual mean, or 25 µg/m3 24-hour mean; and that PM10 not exceed 20 µg/m3 annual mean, or 50 µg/m3 24-hour mean. However, even these might not be safe. In 2013, a large survey published in The Lancet "found a 7% increase in mortality with each 5 micrograms per cubic metre increase in particulate matter with a diameter of 2.5 micrometres (PM2.5).

Where to locate your sensor

Standard advice for locating your sensor is that it should be outside and four metres above ground level. That's good advice for general environmental monitoring; however, we're not necessarily interested in general environmental monitoring – we're interested in knowing what we're breathing in.

Locating your monitor near your workbench will give you an idea of what you're actually inhaling – useless for any environmental study, but useful if you spend a lot of time in there. We found, for

example, that the glue gun produced huge amounts of PM2.5, and we'll be far more careful with ventilation when using this tool in the future.

Adafruit IO

You can use any data platform you like. We chose Adafruit IO because it's easy to use, lets you share visualisations (in the form of dashboards) with others, and connects with IFTTT to perform actions based on values (ours tweets when the air pollution is above legal limits).

One thing to be aware of is that Adafruit IO only holds data for 30 days (on the free tier at least). If you want historical data, you'll need to sign up for the Plus option (which stores data for 60 days), or use an alternative storage method. You can use multiple data stores if you like.

Hardware Setup

1. Connect the SDS011 sensor to the Raspberry Pi using jumper wires.

2. Make sure your Raspberry Pi is connected to the internet.

Software Setup

1. Install Raspbian OS on your Raspberry Pi.
2. Open a terminal and update your system:

```
sudo apt-get update
sudo apt-get upgrade
```

3. Install required Python packages for serial communication and data processing:

```
sudo pip install pyserial
```

Python Script

Create a Python script (e.g., `air_quality_monitor.py`) with the following code:

CODE

```python
import serial
import time

ser = serial.Serial('/dev/ttyUSB0')    # Adjust port if
necessary

def read_sensor_data():
    while True:
        data = ser.read(10)
        if data[0] == 66 and data[1] == 77:
            pm25 = (data[3] * 256 + data[2]) / 10
            pm10 = (data[5] * 256 + data[4]) / 10
            return pm25, pm10

if __name__ == '__main__':
    try:
        while True:
            pm25, pm10 = read_sensor_data()
```

```
            print(f'PM2.5: {pm25} µg/m³, PM10: {pm10} µg/m³')
            time.sleep(10)  # Adjust interval as needed

    except KeyboardInterrupt:
        ser.close()
```

Run the Script

In the terminal, navigate to the directory containing your Python script and run it:

```
python air_quality_monitor.py
```

Output

The script will continuously read data from the PM2.5 sensor and print the PM2.5 and PM10 readings in micrograms per cubic meter (µg/m³) every 10 seconds.

Please note that this is a basic example. A complete project would involve additional features like data logging, web interface, alerts, and more sophisticated sensor handling.

Also, ensure you have the correct driver for your specific PM2.5 sensor and adjust the serial port accordingly.

For a comprehensive project, you may consider using a microcontroller with built-in Wi-Fi capabilities (like an ESP32) for easier IoT integration and data transmission.

CONCLUSION

The proposed system provides low cost, low power, compact and highly accurate system for monitoring the environment with the dedicated sensors remotely from any place in this world.

A perfect tradeoff between accuracy and cost is achieved by making use of single board minicomputer Raspberry pi and appropriate sensors leading to a wellgrounded system. Datasheets

available on the dashboard of IBM Bluemix account will help in framing good policies against the increasing level of pollution to ensure healthful environment.

Air quality monitoring system can be more advantageous if pollutants like Sulfur dioxide, nitrogen dioxide, ground level ozone etc. are also monitored.

Introduction to Cloud

Definition:

Cloud computing refers to the delivery of computing services, including servers, storage, databases, networking, analytics, and software, over the internet. Instead of owning and maintaining physical hardware and software, users can access these resources through cloud providers on a pay-as-you-go basis.

Key Concepts:

1.On-Demand Availability:

- Cloud resources are available to users on-demand. They can be provisioned and de-provisioned quickly, allowing for flexibility and scalability.

2.Resource Pooling:

- Cloud providers maintain a large pool of computing resources that are shared among multiple users. Resources are dynamically allocated and reassigned based on demand.

3.Elasticity:

- Cloud services can be scaled up or down based on workload requirements. This allows businesses to easily adapt to changing needs without significant upfront investment.

4.Self-Service:

- Users can provision and manage resources

through a web-based dashboard or API without the need for human intervention from the service provider.

5.Metered Service:

- Cloud computing resources are typically billed based on usage. This pay-as-you-go model can lead to cost savings compared to traditional on-premises infrastructure.

Service Models;

1.Infrastructure as a Service (IaaS):

- Provides virtualized computing resources over the internet. Users can rent virtual machines, storage, and networking components. Examples: Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure.

2.Platform as a Service (PaaS):

- Offers a platform allowing developers to build, deploy, and manage applications without worrying

about the underlying infrastructure. Examples: Google App Engine, Heroku, Microsoft Azure App Service.

3.Software as a Service (SaaS):

- Delivers software applications over the internet on a subscription basis. Users can access the software via a web browser. Examples: Gmail, Salesforce, Microsoft Office 365.

Deployment Models:

1. Public Cloud:

- Cloud resources are owned and operated by a third-party cloud service provider and are made available to the general public. Example providers: AWS, GCP, Azure.

2.Private Cloud:

- Resources are used exclusively by a single organization. It may be hosted on-premises or by a third-party provider. Offers more control over

security and customization.

3.Hybrid Cloud:

- Combines elements of both public and private clouds, allowing data and applications to be shared between them. Provides greater flexibility and optimization of existing infrastructure.

4.Multi-Cloud:

- Involves using services from multiple cloud providers. Offers redundancy, cost optimization, and the ability to choose the best services from different providers.

Cloud computing has revolutionized the way businesses and individuals access and manage their computing resources, enabling greater flexibility, efficiency, and cost-effectiveness.
It plays a crucial role in modern IT infrastructure and continues to drive innovation in technology.


THANK
YOU