



AQMD

Air Quality Monitoring Delivery



Problem statement

We know that the intensity of air pollution is increasing all over the places, yet we are ignoring this fact in an assumption that we are immune to it. It is found recently by scientists that how bad this could impact us. Health impacts extends way beyond the respiratory illness. Through research it is found that , foetal development ,mental health and metabolic syndrome are impacted due to increase in air pollution.

Based on Dr. Sarath Guttikunda article, "**A general understanding is that an ambient monitoring station can represent an area covering 2 km radius, which translates to 15 sq.km (rounded off).**" But if we consider Chennai Metropolitan area, it is spread across **1189 sq km** (planned to expand up to **8878 sq km**), whereas we have only **8** air quality monitoring stations as of 23rd Jan 2023. It is evident that we don't have enough stations to cover entire area.

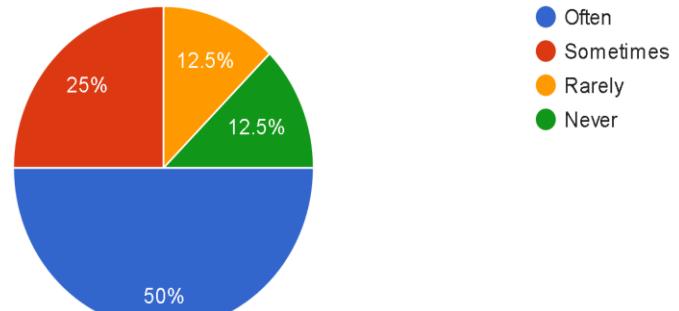
Proposed solution is to make use of BOV Garbage Collector, food delivery partners/ Cabs (Ola, Uber, Swiggy, Zomato ...etc.) to mount the air quality sensor that detects PM 2.5, 10 concentrations in the air and visualize it as a live heat map. It will be an effective solution than ambient monitoring station as it shows us exactly where the intensity of air pollution is higher on street basis.



Public survey

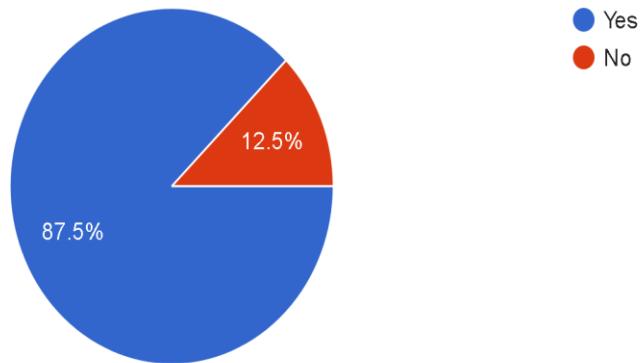
How frequently are you affected due to air pollution?

16 responses



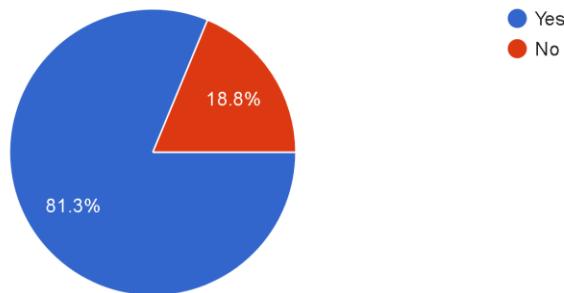
Provided we give you funds and installation, will you allow air quality monitoring device to be mounted in your vehicle?

16 responses

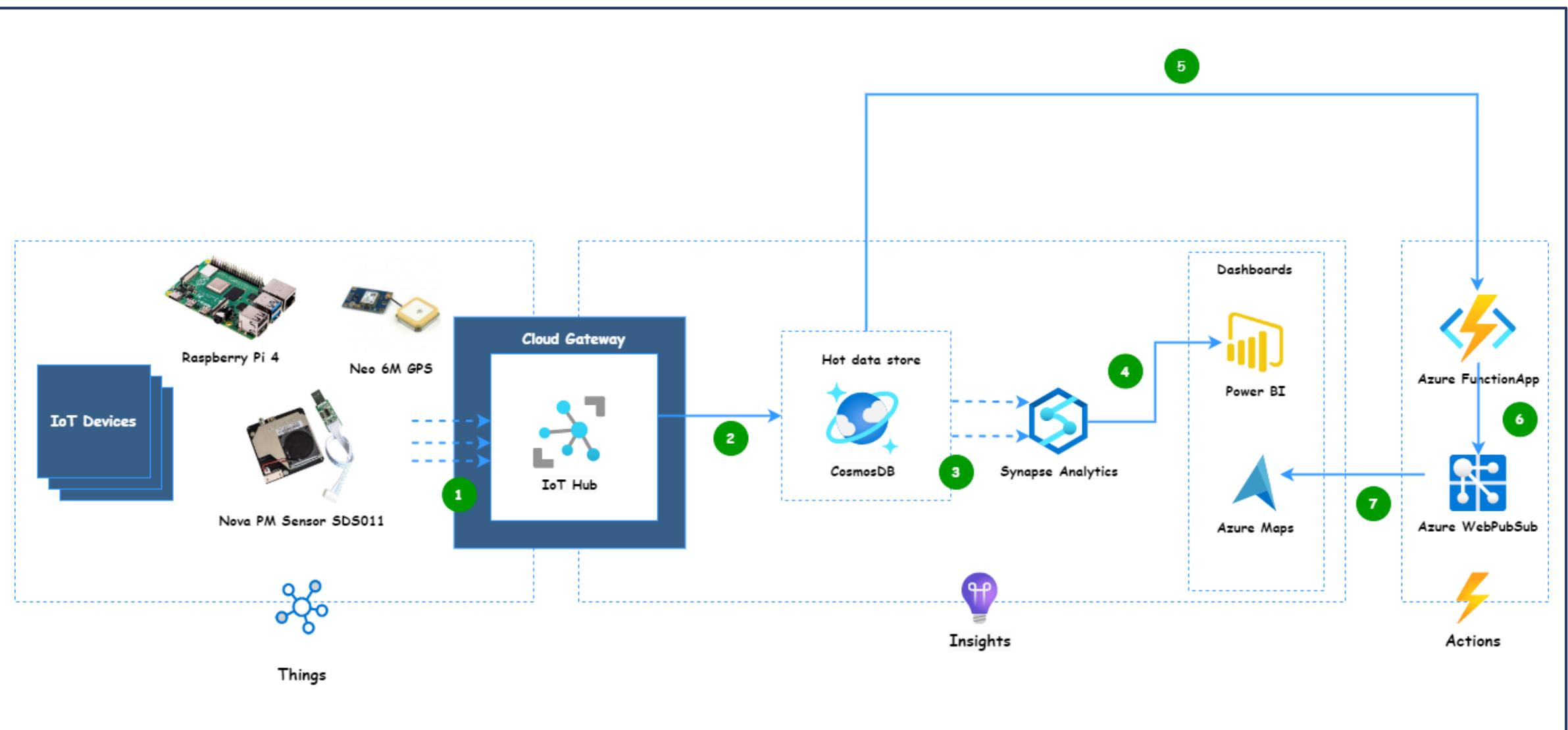


Do you wish to leave the world a better place for the next generation?

16 responses



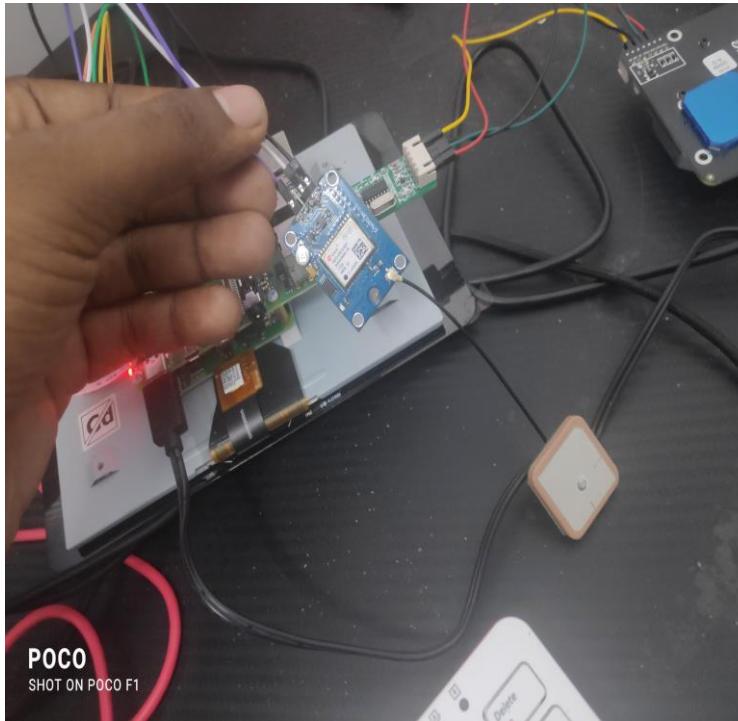
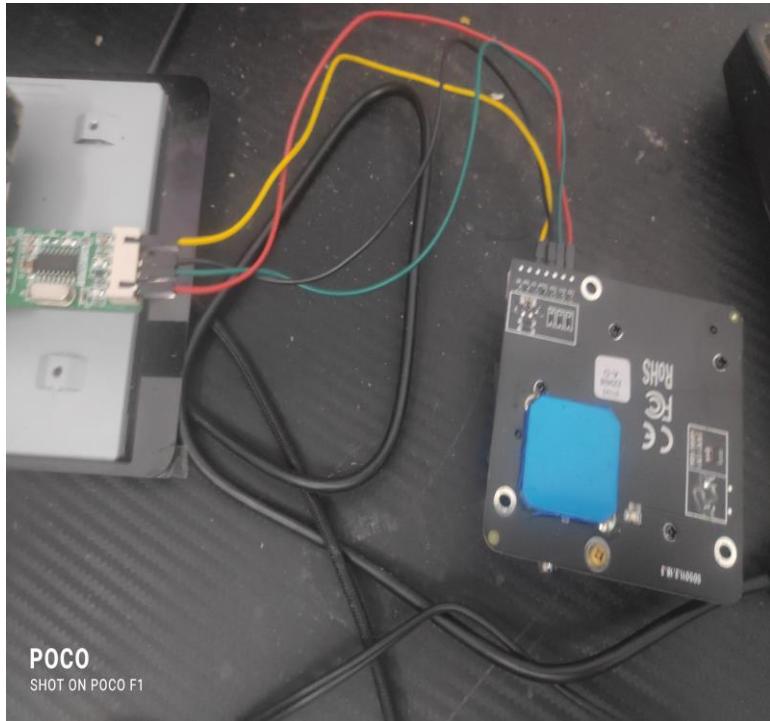
Proposed Architecture



Data Flow

1. IoT devices push the telemetry to cloud gateway (IoT Hub)
2. Message routing feature of IoT hub enables us to forward the telemetry to different store path. In our proposed architecture we have used Azure Cosmos DB as a hot store path to store data. Azure cosmos DB is ideal for IoT workloads as it is highly capable of ingesting high volume of data at low latency and high rates. IoT hub is also capable of filtering only specific messages to be pushed into Azure Cosmos DB.
3. Hybrid Transactional and Analytical Processing (HTAP) capability of Azure Synapse link for Azure Cosmos DB integrates data into analytical store which is highly efficient for analytical purposes.
4. Power BI helps us to visualize data in a Azure Map layer. Heat Map is used in our proposed architecture to better depict our scenario.
5. Azure Cosmos DB change feed triggers an Azure Functions
6. Change Feed trigger is used to publish the message to a Azure Web PubSub service
7. As a presentation layer we have a Azure Map integration in a web app and with the websockets we will have near real-time updates about the telemetry and could visualize it.

Device Setup



Nova PM Sensor SDS011 High Precision
Laser pm2.5 Air Quality Detection
Sensor



Ublox Neo-6M GPS Module with
EEPROM and Antenna



RaspberryPi 4 + PM Sensor SDS011 +
Neo-6M GPS Module



Connect PM sensor and GPS module with RaspberryPi

1. I have used USB 2.0 to TTL UART serial converter module for connecting PM sensor with Raspberry pi , which could be easily plugged into the USB port.
2. GPS module doesn't come with a soldered header pins. Hence we need to solder berg strip male connectors (4 pins) on the GPS module, prior to connecting with RaspberryPi
 - 2.a. For connecting Neo6M GPS module , jumper wires are required to connect with RaspberryPi. Follow the pin out diagram below for establishing connection between them.
 - 2.b. Optionally you can skip the jumper wire part connection to RaspberryPi module , if you have an extra USB 2.0 to TTL UART serial converter.

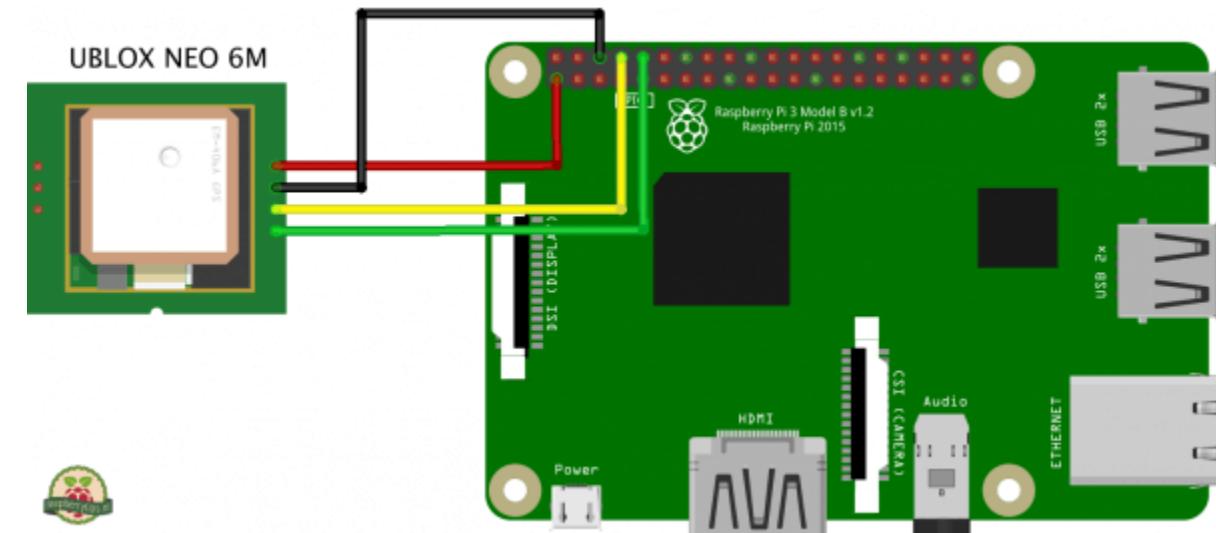


USB 2.0 to TTL UART Serial Connector



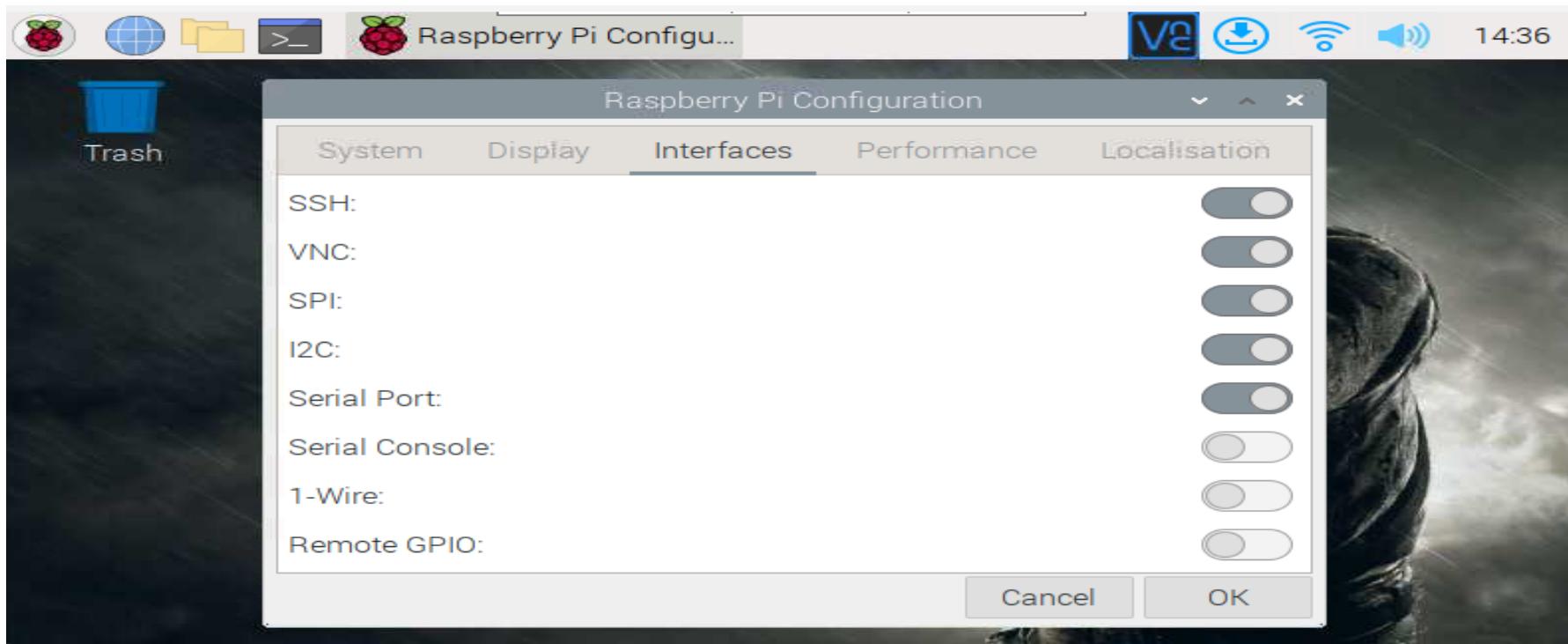
[BUY NOW](#)

Neo 6M GPS module with Jumper wires



RaspberryPi setup

1. Setting up RaspberryPi can be done by following the official documentation [here](#) .
2. After the setup is done , use command prompt to enter raspi-config or under Preferences->select RaspberryPi Configuration.
3. Enable Serial Port interface.



PM2.5 & PM10 safe levels

PM_{2.5} Particles

Based on the daily mean concentration for historical data, latest 24 hour running mean for the current day.

| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------------------|------|-------|-------|----------|----------|----------|-------|-------|-------|------------|
| Band | Low | Low | Low | Moderate | Moderate | Moderate | High | High | High | Very High |
| µg m ⁻³ | 0-11 | 12-23 | 24-35 | 36-41 | 42-47 | 48-53 | 54-58 | 59-64 | 65-70 | 71 or more |

PM₁₀ Particles

Based on the daily mean concentration for historical data, latest 24 hour running mean for the current day.

| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------------------|------|-------|-------|----------|----------|----------|-------|-------|--------|-------------|
| Band | Low | Low | Low | Moderate | Moderate | Moderate | High | High | High | Very High |
| µg/m ³ | 0-16 | 17-33 | 34-50 | 51-58 | 59-66 | 67-75 | 76-83 | 84-91 | 92-100 | 101 or more |

Create IoT hub

1. Azure subscription is required and you can create a free account [here](#).
2. Create an IoT Hub by clicking on **+ Create a resource** button or using the search bar.
3. Choose a subscription and create a **resource group** "rg-raspberrypi" if not created one earlier
4. Provide **IoT hub name** and choose the **region** closest to you
5. For the tier I have started with Standard tier , but **Free tier** should be more than enough for testing and evaluating AQMD.
6. Daily **message limit** can be updated based on the preferences for higher tier, however for the free tier we should go with the default one which is 8000 message per day quota.
7. Proceed to the next step **Networking** and accept the defaults for now and click on next step **Management** , you can optionally change the permission mode and assign yourself the contributor role.
8. Proceed to the final step and click on **Create** button.

Home > Create a resource > Marketplace >

IoT hub ...
Microsoft

Basics Networking Management Add-ons Tags Review + create

Create an IoT hub to help you connect, monitor, and manage billions of your IoT assets. [Learn more](#)

Project details
Choose the subscription you'll use to manage deployments and costs. Use resource groups like folders to help you organize and manage resources.

Subscription * rg-raspberrypi

Resource group *

Instance details
IoT hub name * Region * Tier *
Free trial explores the app with live data. Trials cannot scale or be upgraded later.

Daily message limit *

Review + create < Previous Next: Networking >

Home > Create a resource > Marketplace > IoT Hub >

IoT hub ...
Microsoft

Basics Networking Management Add-ons Tags Review + create

Pricing
IoT hub **₹0 INR** per month [Change basics](#)

Add-ons total [Change add-ons](#)

Basics
Subscription: Azure 360 subscription
Resource group: rg-raspberrypi
IoT hub name: raspberrypi-iot-hub
Region: Central India
Disaster recovery enabled: Yes
Tier: Free
Daily message limit: 8,000 (₹0/month)

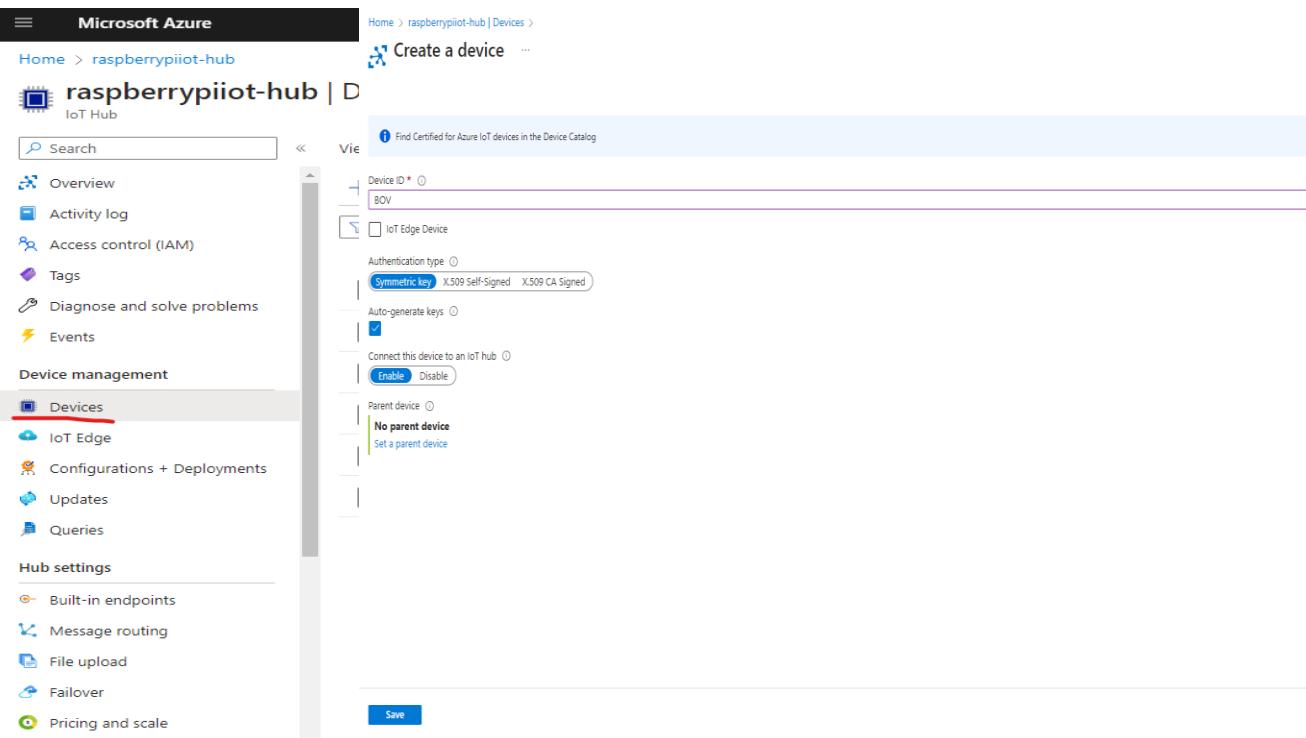
Networking
Connectivity configuration: Public access
Private endpoint connections: None
Allow public network access: Enabled

Management
Tier: F1
Number of F1 IoT hub units: 1
Device-to-cloud partitions: 2

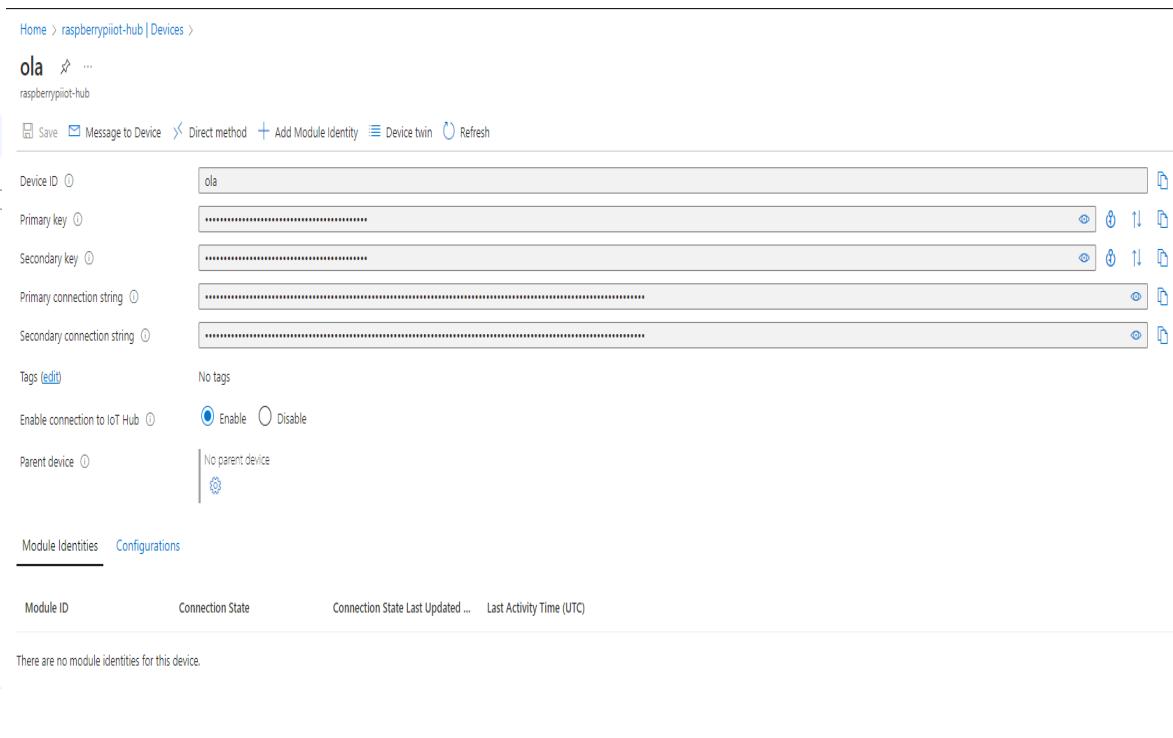
Create < Previous: Tags Next > Automation options

Register device

1. Under the device management in the left pane , click on **Devices**, then select **Add Devices** button
2. In the Create device pane , mention the device ID (Ola, Uber, Zepto, BOV, etc..) from which we will be collecting the telemetry.
3. Let's leave the default authentication type and let's use auto-generate keys
4. Once you click on **Save** button , device will be created along with autogenerated connection string for you to connect with.



The screenshot shows the Microsoft Azure IoT Hub 'Devices' blade. On the left, there is a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Device management (with 'Devices' selected), IoT Edge, Configurations + Deployments, Updates, and Queries. Below that is a Hub settings section with options for Built-in endpoints, Message routing, File upload, Failover, and Pricing and scale. The main area shows a 'Create a device' form. The 'Device ID' field contains 'BOV'. Under 'Authentication type', 'Symmetric key' is selected. Under 'Auto-generate keys', the 'checkbox' is checked. A note says 'Connect this device to an IoT hub' with 'Enable' and 'Disable' buttons. Under 'Parent device', it says 'No parent device' and 'Set a parent device'. At the bottom is a 'Save' button.



The screenshot shows the Microsoft Azure IoT Hub 'Devices' blade for the device 'ola'. The top navigation bar includes Home, raspberryiot-hub, Devices, Create a device, Save, Message to Device, Direct method, Add Module Identity, Device twin, Refresh, and a three-dot ellipsis. The main area displays the device configuration: Device ID (ola), Primary key (redacted), Secondary key (redacted), Primary connection string (redacted), Secondary connection string (redacted), Tags (No tags), Enable connection to IoT Hub (Enable selected), Parent device (No parent device), and Module Identities (Configurations tab selected). A note at the bottom states 'There are no module identities for this device.' At the very bottom are 'Module ID', 'Connection State', 'Connection State Last Updated ...', 'Last Activity Time (UTC)', and a 'Save' button.

IoT Hub Message routing

1. For the hot store path we will make use of Cosmos DB, prior to this step make sure to Create a Cosmos DB in a serverless mode. Follow the steps [here](#) to create Cosmos DB.
2. Create account name as **cosmos-raspberrypi** , database name as **airqualitymonitoringdelivery** and collection as **telemetry** with **synthetic_key** as partition key.
3. Now all the telemetries from IoT Hub should be ingested to this hot store path. In order to do that we should create a Message routing that routes all the data to Cosmos DB.
4. First step is to create Custom endpoints. In our case it is Cosmos Db (at the time of creating this blog , this feature is still in Preview)

The screenshot shows the 'Message routing' blade for the 'raspberrypiiot-hub' IoT hub. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Device management, Devices, IoT Edge, Configurations + Deployments, Updates, Queries, Hub settings, Built-in endpoints, and Message routing (which is currently selected). The main area displays a message stating 'You have created one or more custom routes which would stop messages from flowing to the built-in endpoint. Click here to add a route to the built-in endpoint if you would like to keep the data flowing there.' Below this is a section titled 'Send data from your devices to endpoints that you choose.' It includes tabs for 'Routes', 'Custom endpoints' (which is underlined in red), and 'Enrich messages'. A sub-section titled 'Choose which Azure services will receive your messages. You can add up to 10 endpoints to an IoT hub.' lists several options: 'Cosmos DB' (underlined in red), Event Hubs, Service Bus queue, Service Bus topic, and Storage. Each option has a '+' icon to add it.

Cosmos DB Custom endpoint

1. Provide a name for the Cosmos DB endpoint
2. Choose the Cosmos DB account and respective collection to which the data should be ingested into
3. Choose Synthetic partition for the message
4. Provide the partition key template
5. Leave the default authentication type
6. Click on create button

Home > raspberryiot-hub | Message routing >

Add a Cosmos DB endpoint (preview) ...

Route your telemetry and device messages to a Cosmos DB.

Endpoint name *

Choose an existing Cosmos DB account

Cosmos DB account *

Database *

Collection *

Generate a synthetic partition key for messages * Enable Disable

Partition key name *

Partition key template *

Authentication type

Choose the authentication type for this routing endpoint. [Learn more](#).

Key-based System-assigned User-assigned

System-assigned identity is switched off and cannot be used as an authentication type.

Create

IoT Hub Add routes

1. Click on Add routes
2. Provide a name for the route
3. Select the Cosmos DB endpoint created in the previous step
4. Choose data source to be device telemetry messages
5. Select Enable route
6. Optionally you can modify the routing query to push only specific messages to Cosmos Db , In our case it is set to true (push all messages to Cosmos Db)

Home > raspberrypi0-hub | Message routing >

Add a route ...

Name * ⓘ
iot-route ✓

Endpoint * ⓘ
cosmos-iot-route + Add endpoint

Data source * ⓘ
Device Telemetry Messages

Enable route * ⓘ
Enable Disable

Create a query to filter messages before data is routed to an endpoint. Learn more
Routing query ⓘ
1 true

Test

Save

Telemetry in Cosmos DB

The screenshot shows the Azure Cosmos DB Data Explorer interface for an account named "cosmos-raspberrypi". The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, Notifications, Data Explorer (which is selected and highlighted in red), Features, Default consistency, Backup & Restore, Networking, CORS, Dedicated Gateway, Keys, Advisor Recommendations, Microsoft Defender for Cloud, Identity, and Locks.

The main area displays the NOSQL API view for a database named "airqualitymonitoringdelivery". Under the "DATA" section, the "telemetry" container is selected, and its "Items" sub-section is highlighted with a red box. A table lists several items with their IDs and synthetic keys:

| id | /synthetic_key |
|-------------------------------|----------------|
| 1305d785-6b62-4d62-9df9-7... | ola-2023-02-25 |
| 3d64555d-18cc-4cd1-b0d5-6... | ola-2023-02-25 |
| eaf95eb1-af3e-4c31-b883-a5... | ola-2023-02-25 |
| 81e79eee-a30c-4540-ab13-b... | ola-2023-02-25 |
| efbaff7c-8806-47a2-a418-07... | ola-2023-02-25 |
| 93c1acb9-fe4c-46c4-9141-eb... | ola-2023-02-25 |
| dff2ed4-61ce-441f-9a87-d3... | ola-2023-02-25 |
| f8e9273b-d42c-442c-a1ea-8... | ola-2023-02-25 |
| cd41e8b1-a7ed-40f9-ac5b-9... | ola-2023-02-25 |
| 81114c48-dc3d-448a-9a5d-4... | ola-2023-02-25 |
| e593fe18-dd50-4b0e-a488-d... | ola-2023-02-25 |
| 6b3c3ee3-e075-4cf2-8efe-37... | ola-2023-02-25 |
| 66e69431-ecb3-427b-83b8-7... | ola-2023-02-25 |
| fd18aa2f-eed1-490d-8cc5-b1... | ola-2023-02-25 |
| 96af27af-72c3-4a93-a92b-21... | ola-2023-02-25 |
| 67b5e18d-6d8a-4fb1-981f-0... | ola-2023-02-25 |
| 5cfe9d75-3917-4c64-806a-b... | ola-2023-02-25 |

To the right of the table, a specific document is displayed in JSON format:

```
1   "id": "cd41e8b1-a7ed-40f9-ac5b-942975ba3be1",
2   "synthetic_key": "ola-2023-02-25",
3   "Properties": {},
4   "SystemProperties": {
5     "message-id": "b6612abb-37b0-473c-8ae0-3d04219a7959",
6     "iothub-connection-device-id": "ola",
7     "iothub-connection-auth-method": "{\"scope\":\"device\", \"type\":\"sas\", \"issuer\":\"iothub\", \"acceptingIpFilterRule\":null}",
8     "iothub-connection-auth-generation-id": "638125119070974689",
9     "iothub-content-type": "application/json",
10    "iothub-content-encoding": "utf-8",
11    "iothub-enqueuedtime": "2023-02-25T18:25:09.596Z",
12    "iothub-message-source": "Telemetry"
13  },
14  "iothub-name": "raspberrypi-iot-hub",
15  "Body": {
16    "pm_25": 21.4,
17    "pm_10": 23.3,
18    "lat": 13.136464666666667,
19    "lng": 80.16833
20  },
21  "_rid": "A5k3AJgj5EVeAAAAAAA==",
22  "_self": " dbs/A5k3AA==/colls/A5k3AJgj5EU=/docs/A5k3AJgj5EVeAAAAAAA==/",
23  "_etag": "\"f400b979-0000-2000-0000-63fa52850000\"",
24  "_attachments": "attachments/",
25  "_ts": 1677349509
26
27
```

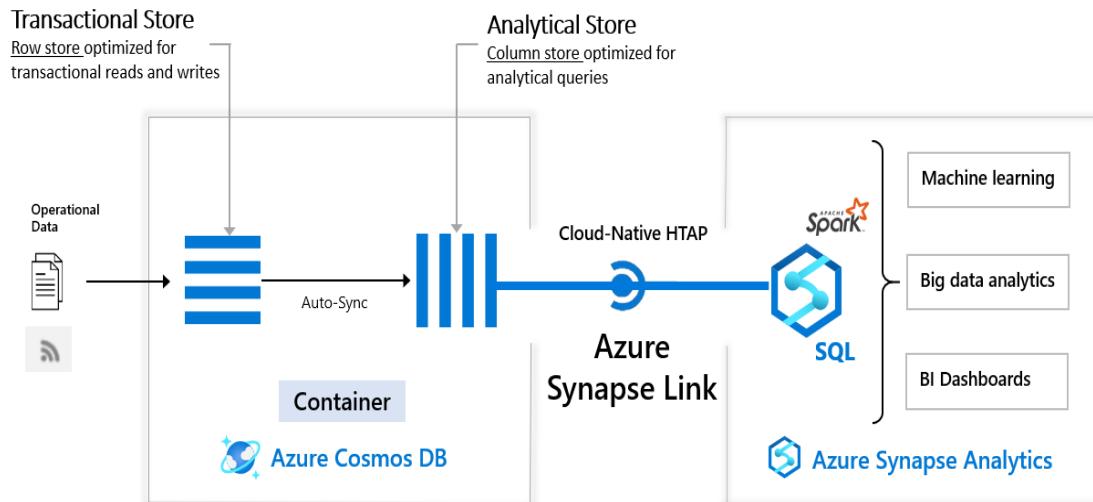
Sample telemetry

```
{  
    "id": "09a0341b-a981-4e1e-ae6b-834dd16faa07",  
    "synthetic_key": "ola-2023-03-21",  
    "Properties": {},  
    "SystemProperties": {  
        "message-id": "89bab089-1609-48d0-bebc-63e8c00de358",  
        "iothub-connection-device-id": "ola",  
        "iothub-connection-auth-method": "  
{\\"scope\\":\\"device\\",\\"type\\":\\"sas\\",\\"issuer\\":\\"iothub\\",\\"acceptingIpFilterRule\\":null},  
        "iothub-connection-auth-generation-id": "638125119018935233",  
        "iothub-content-type": "application/json",  
        "iothub-content-encoding": "utf-8",  
        "iothub-enqueuedtime": "2023-03-21T04:43:35.149Z",  
        "iothub-message-source": "Telemetry"  
    },  
    "iothub-name": "raspberrypi0t-hub",  
    "Body": {  
        "pm_25": 25.3,  
        "pm_10": 28.4,  
        "lat": 13.136398666666667,  
        "lng": 80.16815083333333  
    },  
    "_rid": "A5k3AJgj5EUkAQAAAAAAA==",  
    "_self": "dbs/A5k3AA==/colls/A5k3AJgj5EU=/docs/A5k3AJgj5EUkAQAAAAAAA==/",  
    "_etag": "\\"010076b4-0000-2000-0000-641935f70000\\\"",  
    "_attachments": "attachments/",  
    "_ts": 1679373815  
}
```

Azure Synapse flow

Enable Azure Synapse link for Azure Cosmos DB

1. Now that we have ingested data into Azure Cosmos DB , next step is to visualize and analyze the data to get meaningful insights from the telemetries
2. Azure Synapse link provides a cloud-native hybrid transactional and analytical processing (HTAP) capability that enables near real time analytics over operational data in Azure Cosmos DB.
3. Under Integrations in Azure Cosmos DB account , select Azure Synapse Link and click on Enable button
4. Once Azure synapse link is enabled, make sure to choose the collection if it is created already and create a workspace in the next screen.



The screenshot shows two Azure portal pages. The left page is titled 'cosmos-raspberrypi | Azure Synapse Link' and displays the 'Enable Azure Synapse Link' step, where the 'Account enabled' button is highlighted. The right page is titled 'workspace-synapselink' and shows the workspace configuration, including settings like resource group, location, and networking, along with sections for getting started and analytics pools.

cosmos-raspberrypi | Azure Synapse Link

Enable Azure Synapse Link

Account enabled

workspace-synapselink

Essentials

Resource group (move) : rg-raspberrypi

Status : Succeeded

Location : Central India

Subscription (move) : Azure 360 subscription

Subscription ID : 83343ae6-3277-475f-8be1-946e5d6d9359

Managed virtual network : No

Managed identity object id : dd90c2c3-529a-422f-937f-951766368cd4

Workspace web URL : <https://web.azuresynthesize.net/workspace=%2bsubscriptions%2f83343ae6-3277-475f-8be1-946e5d6d9359%2resourceGroups%2f>

Tags (edit) : CostCenter : prod

Networking

Primary ADLS Gen2 account : <https://dlsynapselink.dfs.core.windows.net>

Primary ADLS Gen2 file system : dlsynapselink

SQL admin username : divkar

SQL Active Directory ad. : live.com@divkar.divkar.k@gmail.com

Dedicated SQL endpoint : workspace-synapselink.sql.azuresynapse.net

Serverless SQL endpoint : workspace-synapselink-on-demand.sql.azuresynapse.net

Development endpoint : <https://workspace-synapselink.dev.azuresynapse.net>

Getting started

Open Synapse Studio

Read documentation

Analytics pools

SQL pools

Create SQL View in Azure Synapse analytics workspace

1. Enter into the Azure synapse analytics workspace
2. Select the database “**airqualitymonitorordelivery**” in the top pane and proceed to create a view
3. Create a view named “**aggregatetelemetry**” which aggregates the data by rounding off latitude and longitude to 3 points, to avoid any discrepancies while visualizing the data as a heatmap

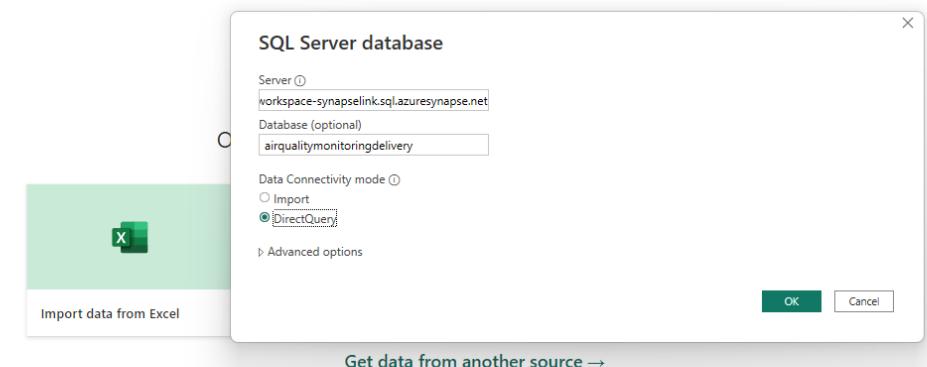
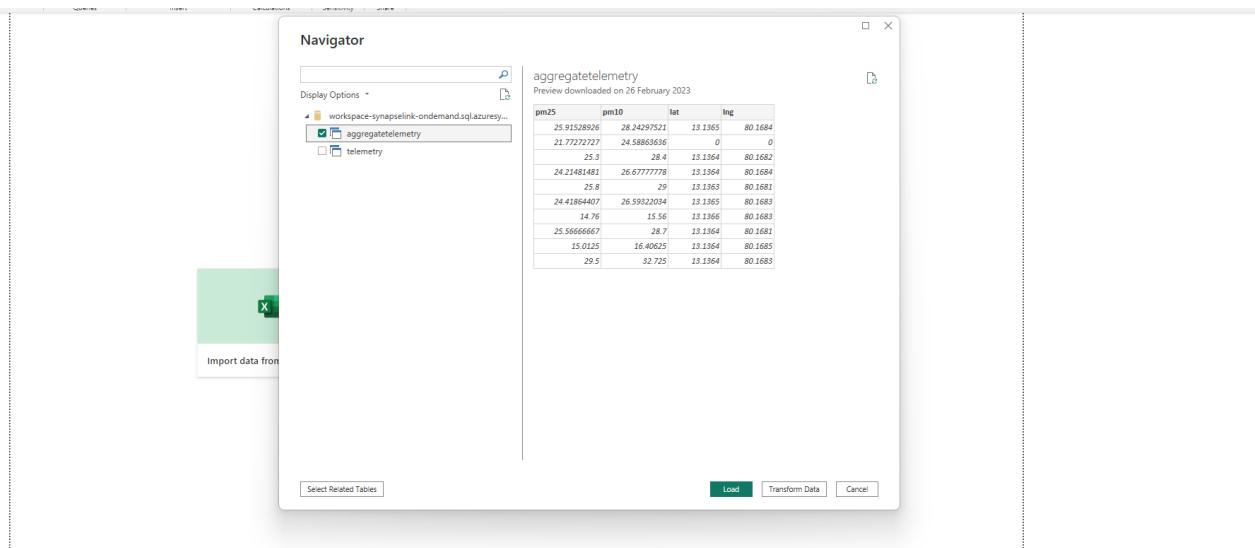
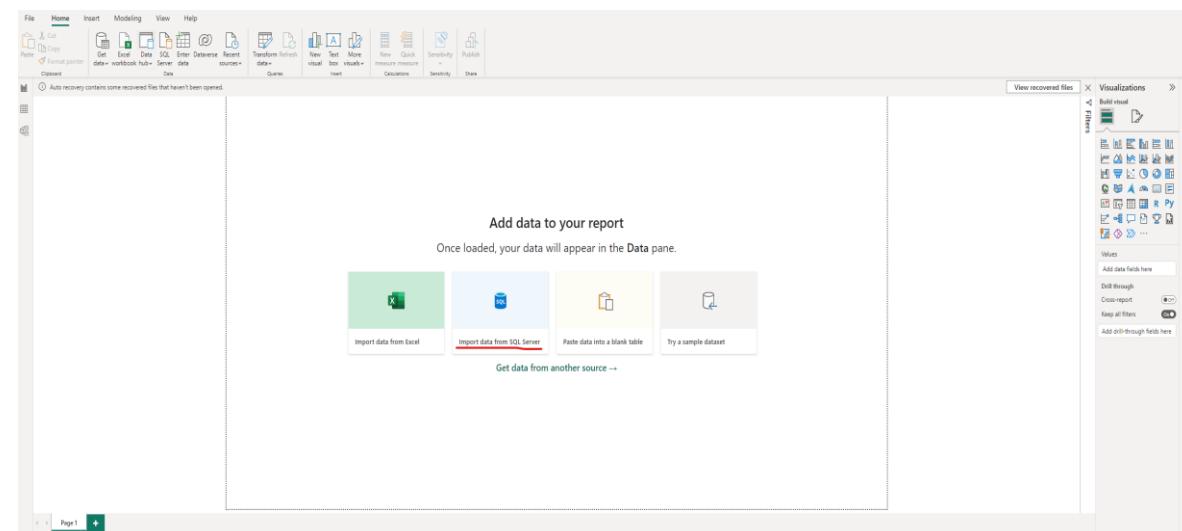
The screenshot shows the Azure Synapse Analytics workspace interface. On the left, the Data blade is open, displaying the workspace navigation bar with 'Synapse live', 'Validate all', and 'Publish all' buttons. Below this, the 'Workspace' tab is selected, showing a list of resources: 'SQL database' (with 'airqualitymonitoringdelivery' expanded), 'External tables', 'External resources', 'Views' (containing 'dbo.aggregatelemetry' and 'dbo.telemetry'), 'System views', 'Schemas', and 'Security'. A search bar 'Filter resources by name' is also present. On the right, a 'SQL script 1' tab is active in the main editor area. The script content is as follows:

```
1 Create view aggregatetelemetry as
2 SELECT avg(pm_25) as pm25,avg(pm_10) as pm10,ROUND(lat,3) as lat,ROUND(lng,3) as lng FROM
3 (SELECT avg(pm_25) as pm_25,avg(pm_10) as pm_10,ROUND(lat,3) as lat,ROUND(lng,3) as lng
4 FROM OPENROWSET(
5      'CosmosDB',
6      'Account=cosmos-raspberrypi;Database=airqualitymonitoringdelivery;Key=izvJr2SAW7RMYqUKE4Tsv4992Ld9PsuS2PqC0QXY7hbKCoQZKi2Y61Dnvej686DWxhpITSp1BwkOACDbA85nXg==',
7      telemetry)
8 WITH (
9      pm_25      float '$.Body.pm_25',
10     pm_10      float '$.Body.pm_10',
11     lat        varchar(100) '$.Body.lat',
12     lng        varchar(100) '$.Body.lng'
13 ) AS docs Group by lat,lng) as aqmd group by aqmd.lat,aqmd.lng;
14
15
16 drop view aggregatetelemetry
```

The status bar at the bottom right indicates 'General Information'.

Enable Power BI Integration

1. Sign up for a free Power BI account if you don't have one [here](#).
2. You can optionally download Power BI Desktop tool from [here](#) or you can use [app.powerbi.com](#)
3. Select Import data from SQL server.
4. Provide the connection details (Serverless SQL endpoint) in the server text box and choose **DirectQuery** as Data Connectivity mode
5. Click on OK button
6. In the next screen , sign in if prompts and then click on **connect** button.
7. Now Choose the view created in the earlier step and click on **Load** button



Python code – Initialize IoTHubDeviceClient

1. Import **IoTHubDeviceClient** and **Message** from **azure.iot.device**
2. Set the serial port to “**/dev/ttyUSB0**” for listening PM sensor data
3. Set the serial port to “**/dev/ttyAMA0**” for listening GPS module data
4. Initialize the IoTHubDeviceClient using the connection String copied from the Device registration screen

```
● ● ●

from azure.iot.device import IoTHubDeviceClient, Message

port = "/dev/ttyUSB0" # Set this to your serial port.
baudrate = 9600

gpsPort="/dev/ttyAMA0"
gpsBaudrate= 9600

device_client= IoTHubDeviceClient.create_from_connection_string("HostName=raspberrypi0t-hub.azure-devices.net;DeviceId=ola;SharedAccessKey=rbLJZLH3zB/khja5Ul++Xkh+nmvbScBg8z+i+9RoGUI=")
```

Python code – Form Message payload and send to IoT Hub

1. Form the schema for the message payload that will be sent to IoT Hub
MSG_PAYLOAD= '{"pm_25": {pm_25}, "pm_10": {pm_10}, "lat": {lat}, "lng": {lng}}'
2. Get the readings from GPS module to capture latitude and longitude and PM sensor to capture Pm_2.5 and Pm_10 readings
3. Populate the message payload
4. Send the Message payload to IoT Hub by calling
device_client.send_message(msg)

```
MSG_PAYLOAD = '{"pm_25": {pm_25}, "pm_10": {pm_10}, "lat": {lat}, "lng": {lng}}'
ser = Serial(port, baudrate=baudrate, bytesize=EIGHTBITS, parity=PARITY_NONE, stopbits=STOPBITS_ONE)
ser.flushInput()

HEADER_BYTE = b"\xAA"
COMMANDER_BYTE = b"\xC0"
TAIL_BYTE = b"\xAB"

byte, previousbyte = b"\x00", b"\x00"

while True:
    previousbyte = byte
    byte = ser.read(size=1)
    if previousbyte == HEADER_BYTE and byte == COMMANDER_BYTE:
        packet = ser.read(size=8) # Read 8 more bytes
        readings = struct.unpack('<Hcccc', packet)
        pm_25 = readings[0]/10.0
        pm_10 = readings[1]/10.0
        gpsSer= Serial(gpsPort,9600,timeout=0.5)
        dataout= pynmea2.NMEAStreamReader()
        newdata= gpsSer.readline().decode('utf-8',errors='ignore')
        lat=0
        lng=0
        id = packet[4:6]
        checksum = readings[4][0]
        calculated_checksum = sum(packet[:6]) & 0xFF
        checksum_verified = (calculated_checksum == checksum)
        tail = readings[5]
        if tail == TAIL_BYTE and checksum_verified:
            if newdata[0:6] == "$GPRMC":
                newmsg= pynmea2.parse(newdata)
                lat= newmsg.latitude
                lng=newmsg.longitude
                payload= MSG_PAYLOAD.format(pm_25=pm_25, pm_10=pm_10, lat=lat, lng=lng)
                msg= Message(payload)
                msg.message_id= uuid.uuid4()
                msg.content_encoding= "utf-8"
                msg.custom_properties["deviceId"]= "ola"
                msg.content_type= "application/json"
                print("sending message: {}".format(msg))
                device_client.send_message(msg)
                time.sleep(2)
                print("PM 2.5:", pm_25, "µg/m³ PM 10:", pm_10, "µg/m³ ID:", bytes(id).hex())
```

Real-time Azure Maps flow

Create Azure Web PubSub

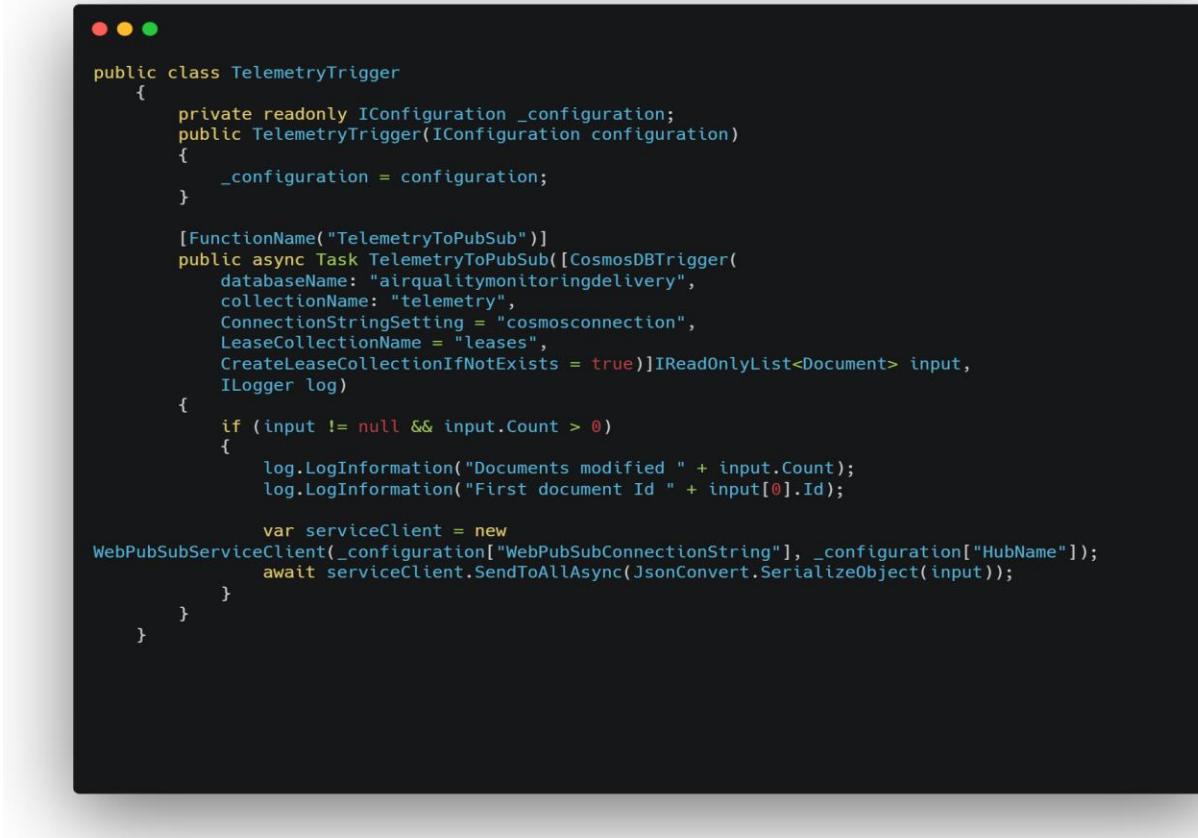
1. Azure subscription is required and you can create a free account [here](#).
2. Create a WebPubSub instance by clicking on **+ Create a resource** button or using the search bar.
3. Choose a subscription and create a **resource group** "rg-raspberrypi" if not created one earlier
4. Provide **Resource name** and choose the **region**
5. For the tier I have chosen **Free tier**, that should be more than enough for testing and evaluating AQMD.
6. Proceed to the final step and click on **Create** button.
7. After resource is created add a hub (**AQMD**) to the instance
8. Copy the ConnectionString from the key session

The screenshot shows the Azure portal interface for creating a Web PubSub service. At the top, the navigation bar includes Home > Create a resource > Marketplace > Web PubSub Service. The main title is "Web PubSub Service" with a "Create new" button. Below the title, there are tabs for Basics, Networking, Tags, and Review + create. The Basics tab is selected. A sub-header says "Deploy fully managed WebPubSub Service at scale. Learn more about WebPubSub".
Project Details:
Subscription: Azure 360 subscription
Resource group: rg-raspberrypi (highlighted in purple)
Service Details:
Resource Name: pubsub-raspberrypi
Region: West US
Pricing tier: Free (Up to 20 connections, 40,000 KB messages per day included)
At the bottom, there are buttons for Review + create, Next : Networking >, and Download a template for automation.

The screenshot shows the "Keys" section of a Web PubSub service named "pubsub-raspberrypi". The left sidebar has a "Settings" menu with "Keys" selected. The main area displays two access keys: Primary and Secondary. Each key has a "Key" field containing a long string of characters, a "Connection string" field with the endpoint and access key, and a "Regenerate [Primary/Secondary] Key" button. The "Primary" key's connection string is: "Endpoint=https://pubsub-raspberrypi.webpubsub.azure.com;AccessKey=OiooejPZOnvQkCqFRB\$4Ofs6ERSvJlwCzubyOu8QV0=:Version=1.0;"

C# code – Telemetry trigger

1. Create Azure Function
2. Create Cosmos DB Trigger for the **telemetry** collection in **airqualitymonitoringdelivery** database.
3. Use Azure WebPubSub client to publish the changes to the **AQMD** hub created in the previous step



```
public class TelemetryTrigger
{
    private readonly IConfiguration _configuration;
    public TelemetryTrigger(IConfiguration configuration)
    {
        _configuration = configuration;
    }

    [FunctionName("TelemetryToPubSub")]
    public async Task TelemetryToPubSub([CosmosDBTrigger(
        databaseName: "airqualitymonitoringdelivery",
        collectionName: "telemetry",
        ConnectionStringSetting = "cosmosconnection",
        LeaseCollectionName = "leases",
        CreateLeaseCollectionIfNotExists = true)] IReadOnlyList<Document> input,
        ILogger log)
    {
        if (input != null && input.Count > 0)
        {
            log.LogInformation("Documents modified " + input.Count);
            log.LogInformation("First document Id " + input[0].Id);

            var serviceClient = new
WebPubSubServiceClient(_configuration["WebPubSubConnectionString"], _configuration["HubName"]);
            await serviceClient.SendToAllAsync(JsonConvert.SerializeObject(input));
        }
    }
}
```

C# code – Get Connection URL for Web PubSub

1. In order to establish connection with Azure Web PubSub instance created over the previous step in the frontend , we need a ConnectionURL
2. HTTP trigger with Azure WebPubSub input bindings is used to generate ConnectionURL for a given user.

```
● ● ●

public class Connection
{
    private readonly ILogger<Connection> _logger;

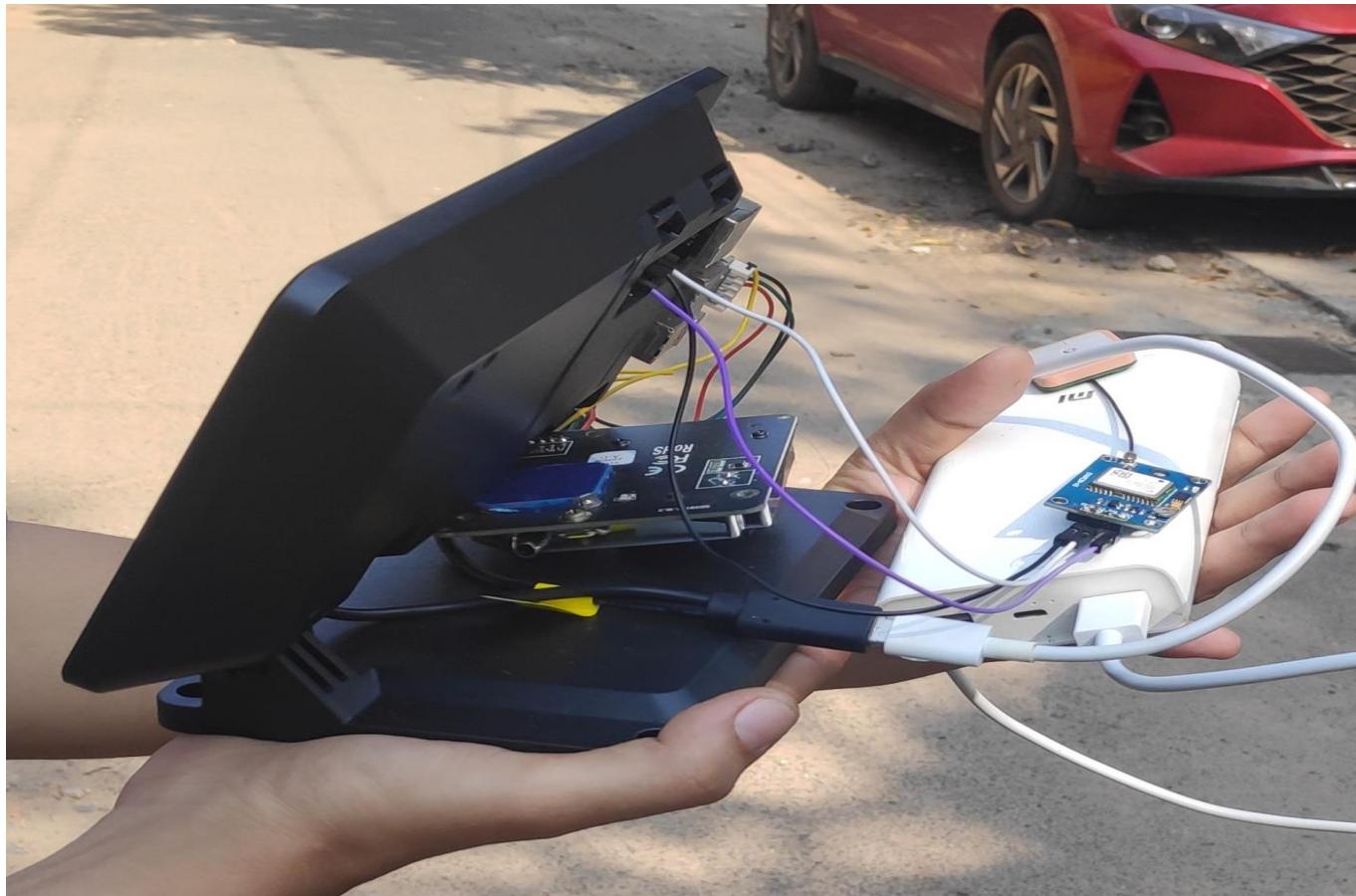
    public Connection(ILogger<Connection> log)
    {
        _logger = log;
    }

    [FunctionName("GetConnectionString")]
    [OpenApiOperation(operationId: "Run", tags: new[] { "name" })]
    [OpenApiParameter(name: "name", In = ParameterLocation.Query, Required = true, Type =
typeof(string), Description = "The **Name** parameter")]
    [OpenApiResponseWithBody(statusCode: HttpStatusCode.OK, contentType: "text/plain", bodyType:
typeof(string), Description = "The OK response")]
    public async Task<WebPubSubConnection> GetConnectionString(
        [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route = "negotiate")] HttpRequest
req, [WebPubSubConnection(Hub = "AQMD", UserId = "{query.userid}")] WebPubSubConnection connection)
    {
        Console.WriteLine("login");
        return await Task.FromResult(connection);
    }
}
```

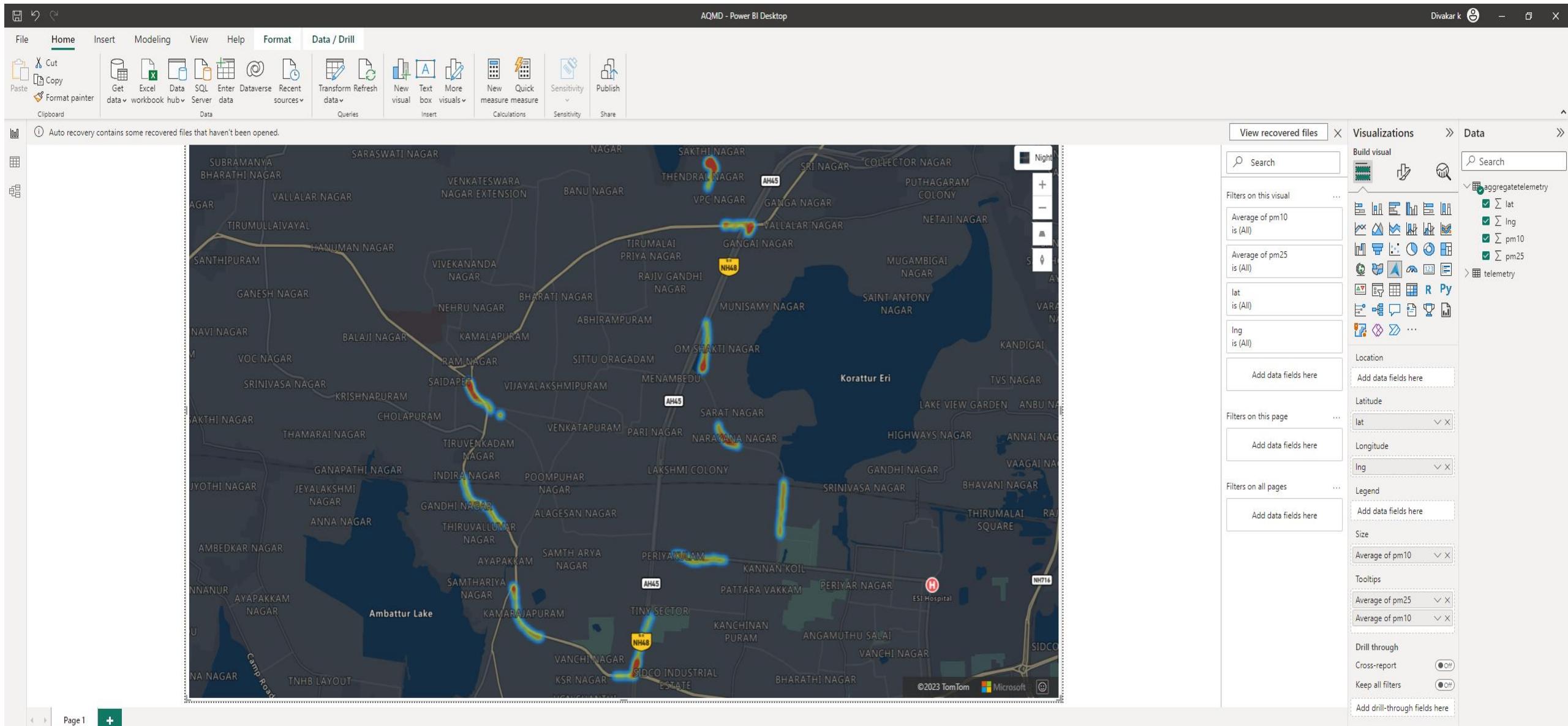
Demo

It's time to ride!

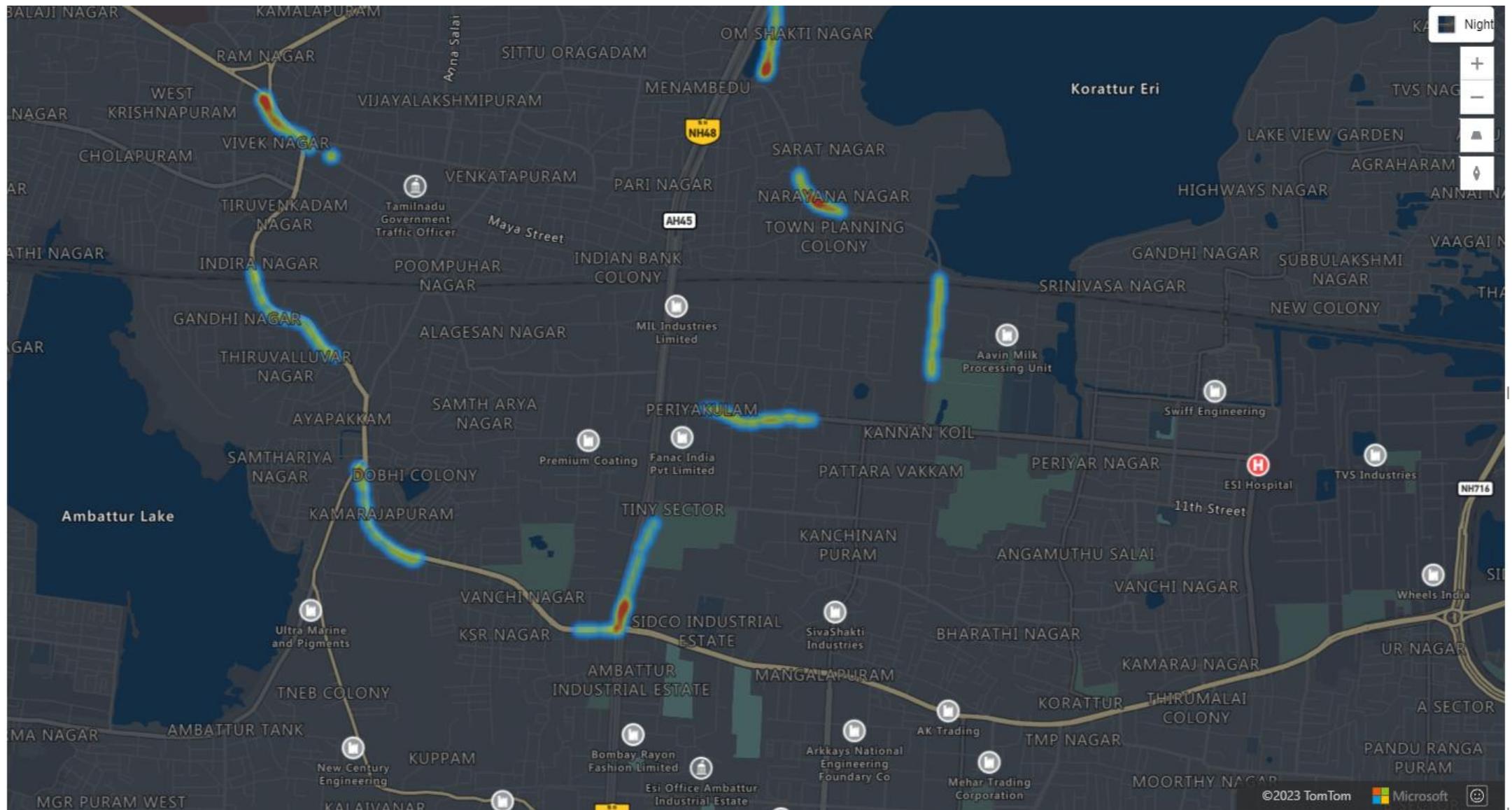
It is optional to buy a case for raspberry pi model . In my case I have bought a case to hold my touch screen + raspberry Pi. Myself and my wife decided to take it for a ride in and around our residence (**Ambattur estate**). Outcome of the ride can be seen in the next slide, which feels more accurate as we felt the same before while crossing those streets!



Visualize telemetry as heatmap in Power BI



Heatmap closer look



Challenges

Sensor maintenance

Data management

Privacy concerns

Vehicles can experience harsh conditions that can impact the performance of air quality sensors.

Data management can be challenging as it involves collecting, storing and analyzing data from millions of vehicles.

Data collected by vehicle-based air quality sensors can contain sensitive information about individuals and their driving habits.

How overcome challenges:

Sensor maintenance:

- Encouraging widespread adaption of air quality sensors in vehicles requires significant effort and investment in awareness and education.
 - Timely notification to drivers about the calibration of devices.
-

Data management:

In our implementation we have used Azure IoT hub, that is capable of ingesting data from millions of devices and for the warm path store, Azure Cosmos DB is used that provides us the benefits of ingesting device telemetry data at high rates and return indexed queries with low latency and high availability.

- TTL for Hot store path is set to 90 days
- TTL for Azure Synapse analytics is set to infinite to look back historical data

Business Benefits

Wider Coverage:

Data can be collected from a wider range of locations, including areas where fixed monitoring stations may not exist

Real-time data:

The Sensors in vehicles can provide real-time data on air quality, allowing for more immediate responses to changing conditions.

Able to visualize data in a heatmap street by street basis.

Dynamic monitoring:

Vehicles can travel to different locations and provide data on air quality in a variety of places, such as urban, suburban and rural areas.

Exposure information:

Individuals can receive information about their personal exposure to air pollution, which can inform decisions about daily activities and travel

Source Code

<https://github.com/Cloud-Jas/AQMD>

References

| | Link |
|---------------------------------|--|
| Air quality monitoring Insights | Dr. Sarath Guttikunda study on Ambient monitoring system |
| | Experts say five air quality monitoring stations not enough, seek 3 more |
| | Interview with Dr. Sarath Guttikunda who wins AGU Award |
| | Live Air Quality monitoring stations |
| | Chennai Metropolitan Area to be trifurcated, will cover 5,904 sqkm |
| | PM2.5, PM10 safe levels breached in Delhi throughout summer |
| Raspberry Pi | Link |
| | Setting up your Raspberry Pi |