Documentation for writing to InfluxDB from RaspberryPi 4 post receiving data from ESP32 via BLE

ESP32:

//Sending data from ESP32 to RPi4 - Air Quality monitoring sensors used

```
#include "BluetoothSerial.h"
#include<BLEDevice.h>
#include<DHT.h>
BluetoothSerial SerialBT;
#define DHTPIN 4
#define DHTTYPE DHT11
#define mq 34
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(115200);
dht.begin();
pinMode(mq,INPUT);
SerialBT.begin("ESP32-BT-Server"); // Name of your Bluetooth device
Serial.println("Bluetooth Started! Ready to pair...");
if (SerialBT.available()) {
String data = SerialBT.readStringUntil('\n'); // Read until newline
Serial.print("Received from Raspberry Pi:");
Serial.println(data);
}
}
void loop() {
float temp = dht.readTemperature();
float hum = dht.readHumidity();
float gas=analogRead(mq);

Serial.print("Temperature sensor:");
Serial.println(temp);

Serial.print("Humidity sensor:");
Serial.println(hum);
char data[64];
snprintf(data,sizeof(data),"%f,%f,%f",temp,hum,gas);

SerialBT.print(data);
Serial.print("Gas sensor:");
Serial.println(gas);
delay(2000);

}
```

RaspberryPi 4:

(i)Setting up InfluxDB:

```
//Installing and adding the InfluxDB repository
01 wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add-
02 echo "deb  https://repos.influxdata.com/debian  stable main" sudo
tee/etc/apt/sources.list.d/
//Update and install influxDB
03 sudo apt-get update
04 sudo apt-get install influxdb
//Start and enable influxdb
04 sudo systemctl start influxdb
05 sudo systemctl enable influxdb
//Verify if the influxdb is in active running state
06 sudo systemctl status influxdb
//Configuring influxdb
//open influxdb
07 influx
//Creating a database
08 CREATE DATABASE database_name
//Creating user for the above database
09 CREATE USER "my_user" WITH PASSWORD 'my_password' WITH ALL PRIVILEGES
//Grant write permissions to the user
10 GRANT WRITE ON database_name TO “my_user”
//exit the influxdb shell
11 exit
//Install the python libraries for influxdb-client
12 sudo pip3 install influxdb-client
```

Post executing these commands in the terminal, run your “xyz.py” script

(ii)RPi 4 Code:

```
import bluetooth
from influxdb import InfluxDBClient
import time
# InfluxDB 1.x Configuration
INFLUX_HOST = "localhost"   # Change if InfluxDB is on another system
INFLUX_PORT = 8086
```

```
INFLUX_DB = "database2"  # Your database name
MEASUREMENT = "sensor_data"  # Table name
# Initialize InfluxDB Client
client = InfluxDBClient(host=INFLUX_HOST, port=INFLUX_PORT,
database=INFLUX_DB)
# Find the Bluetooth address of ESP32
def find_esp32_address(target_name="ESP32-BT-Server"):
    print("Searching for ESP32 Bluetooth device...")
    nearby_devices = bluetooth.discover_devices(duration=8, lookup_names=True,
flush_cache=True, lookup_class=False)
    for addr, name in nearby_devices:
        print(f"Found: {name} - {addr}")
        if name == target_name:
            return addr
    return None


def main():
    esp32_address = find_esp32_address()
    if esp32_address is None:
        print("ESP32 not found. Ensure it is advertising.")
        return

    print(f"ESP32 address: {esp32_address}")

    try:
        # Create an RFCOMM socket
        sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
        sock.connect((esp32_address, 1))  # Port 1 is the default for Bluetooth Serial
        sock.settimeout(10)  # Set timeout to avoid blocking
        print("Connected to ESP32!")
```

```
while True:

    try:
        data = sock.recv(1024).decode('utf-8').strip()
        print(f"Received from ESP32: {data}")

        temp_str, hum_str, gas_str = data.split(',')
        temp = float(temp_str)
        humidity = float(hum_str)
        gas = float(gas_str)

        # Get timestamp
        timestamp = int(time.time() * 1e9)

        # Prepare JSON payload for InfluxDB
        json_body = [
            {
                "measurement": MEASUREMENT,
                "time": timestamp,
                "fields": {
                    "temperature": temp,
                    "humidity": humidity,
                    "gas": gas,
                }
            }
        ]
        client.write_points(json_body)
        print(f"Written: Timestamp={timestamp}")
        time.sleep(2)
    except bluetooth.btcommon.BluetoothError as e:
```

```
                print(f"Error receiving data: {e}")
                break

    except bluetooth.btcommon.BluetoothError as e:
        print(f"Bluetooth error: {e}")

    except KeyboardInterrupt:
        print("Exiting...")

    finally:
        if 'sock' in locals() and sock is not None:
            sock.close()
            print("Bluetooth connection closed.")

if __name__ == "__main__":
    main()
```

(iii) Reading from InfluxDB using SQL queries:

```
//Execute this code in RPi system
    from influxdb import InfluxDBClient
    # InfluxDB 1.x Configuration
    INFLUX_HOST = "localhost"   # Change if InfluxDB is on another system
    INFLUX_PORT = 8086
    INFLUX_DB = "database2"        # Your database name
    MEASUREMENT = "sensor_data" # Your measurement (table) name
    # Initialize InfluxDB Client
    client = InfluxDBClient(host=INFLUX_HOST, port=INFLUX_PORT,
    database=INFLUX_DB)
```

```
# Query data from InfluxDB

query = f'SELECT * FROM {MEASUREMENT} ORDER BY time DESC LIMIT 10'

# Fetch last 10 entries

result = client.query(query)

# Extract and print data

points = list(result.get_points())

if points:

    print("Latest Sensor Data:")

    for point in points:

        print(f"Time: {point['time']}, Temperature: {point['temperature']}, Humidity:
{point['humidity']}, MQ: {point['gas']}") # modified line

else:

    print("No data found in InfluxDB!")

# Close InfluxDB connection

client.close()
```