**Documentation for Smart Waste Monitoring System**

# **Table of Contents**

1. Software Needed for Smart Waste Monitoring System 3

2. Necessary Libraries Required in Arduino IDE 3

3. Necessary Libraries Required in Visual Studio Code MQTT Publisher 3

4. Necessary Libraries Required in Visual Studio Code MQTT Subscriber 4

5. Connecting the wires from Arduino Uno Microcontroller to the sensors 5

6. Installation Steps for the Required Software 5

7. Software Needed for Smart Waste Monitoring System 8

# **1) Software Needed for Smart Waste Monitoring System:**

1. InfluxDB
2. Arduino IDE
3. Ubuntu LTS
4. Python
5. Visual Studio Code
6. MQTTX MQTT Client

# **2) Necessary libraries required in Arduino IDE:**

1. **Adafruit\_VL53L1X**
   1. Install via Arduino Library Manager (search for "Adafruit VL53L1X") or from GitHub:  
      <https://github.com/adafruit/Adafruit_VL53L1X>
   2. Also requires **Adafruit\_BusIO** dependency library.
2. **NeoGPS (NMEAGPS)**
   1. Install via Arduino Library Manager (search for "NeoGPS") or from GitHub:  
      <https://github.com/SlashDevin/NeoGPS>
3. **NeoSWSerial** 
   1. Install via Arduino Library Manager (search for "NeoSWSerial") or from GitHub:

<https://github.com/SlashDevin/NeoSWSerial>

1. **GPSport.h**
   1. This is typically part of the NeoGPS examples or a helper file. Ensure it's in your sketch folder or part of the NeoGPS library.

# **3) Necessary libraries required in Visual Studio Code MQTT publisher:**

1. **python-dotenv** – For loading environment variables from .env files.
   1. Install: pip install python-dotenv
2. **pyserial** – For serial communication (e.g., with Arduino).
   1. Install: pip install pyserial
3. **json** – Built into Python (Parses MQTT payloads).
4. **ssl** – Built into Python (used for secure connections).
5. **socket** – Built into Python (used for network communication).
6. **time** – Built into Python (for delays and timestamps).
7. **threading** – Built into Python (for multithreading).
8. **os** – Built into Python (for environment variables and system operations).
9. **datetime** – Built into Python (for handling dates and times).
10. **tzlocal** – For getting the system's local timezone.
    1. Install: pip install tzlocal
11. **paho-mqtt** – For MQTT communication (IoT messaging).
    1. Install: pip install paho-mqtt
12. **geopy** – For geocoding (converting addresses to coordinates).
    1. Install: pip install geopy
13. **smtplib** – Built into Python (for sending emails).
14. **email.mime** – Built into Python (for constructing email messages).

# **4) Necessary libraries required in Visual Studio Code MQTT subscriber:**

1. **python-dotenv** - For loading environment variables from .env files.
   1. Install: pip install python-dotenv
2. **json** - Built into Python (Parses MQTT payloads).
3. **ssl** - Built into Python (used for secure connections).
4. **socket** - Built into Python (used for network communication).
5. **csv** - Built into Python (Reads/writes CSV files (e.g., for data exports).
6. **pytz** - Timezone handling
   1. Install: pip install pytz
7. **os** - Built into Python (for environment variables and system operations).
8. **glob** - Built into Python (Finds files matching patterns (e.g., glob.glob("logs/\*.csv")).
9. **re** - Built into Python (Parses strings with regex (e.g., extracting bin IDs from filenames))
10. **datetime** - Built into Python (for handling dates and times).
11. **tzlocal** - For getting the system's local timezone.
    1. Install: pip install tzlocal
12. **paho-mqtt** - For MQTT communication (IoT messaging).
    1. Install: pip install paho-mqtt
13. **influxdb\_client** - Interacts with InfluxDB
    1. Install: pip install influxdb-client
14. **smtplib** - Built into Python (for sending emails).
15. **email.mime** - Built into Python (for constructing email messages).
16. **tensorflow** - Deep learning framework
    1. Install: pip install tensorflow
17. **numpy** - Numerical computing
    1. Install: pip install numpy
18. **pandas** -Data manipulation
    1. Install: pip install pandas
19. **matplotlib** -Data visualization
    1. Install: pip install matplotlib
20. **scikit-learn** -Machine learning metrics (e.g., MAE)
    1. Install: pip install scikit-learn
21. **Prophet** - Time series forecasting
    1. Install: pip install prophet

# **5) Connecting the wires from Arduino Uno Microcontroller to the sensors:**

**GPS NEO-M8 Modules:**

1. Connect a male to female Jumper wire from VCC pin of the GPS module to the 5V pin of the Arduino Uno.
2. Connect a male to female Jumper wire from GND pin of the GPS module to the GND pin of the Arduino Uno.
3. Connect a male to female Jumper wire from RX pin of the GPS module to the D3 pin of the Arduino Uno.
4. Connect a male to female Jumper wire from TX pin of the GPS module to the D4 pin of the Arduino Uno.

**Time-Of-Flight VL53L1X Sensors:**

1. Connect a male to female Jumper wire from VIN pin of the ToF sensor to the 3.3V pin of the Arduino Uno.
2. Connect a male to female Jumper wire from GND pin of the ToF sensor to the GND pin of the Arduino Uno.
3. Connect a male to female Jumper wire from SCL pin of the ToF sensor to the A5 pin of the Arduino Uno.
4. Connect a male to female Jumper wire from SDA pin of the ToF sensor to the A4 pin of the Arduino Uno.
5. Connect a male to female Jumper wire from GPIO1 pin of the ToF sensor to the D1 pin of the Arduino Uno.
6. Connect a male to female Jumper wire from XSHUT pin of the ToF sensor to the D2 pin of the Arduino Uno.

# **6) Installation Steps for the required Software:**

1. **InfluxDB (On MQTT Subscriber End)**

**Steps:**

Open the Ubuntu LTS and run the following commands:

**# Add InfluxDB repository**

wget -q https://repos.influxdata.com/influxdata-archive.key

sudo gpg --yes --dearmor -o /usr/share/keyrings/influxdata-archive-keyring.gpg ./influxdata-archive.key

echo "deb [signed-by=/usr/share/keyrings/influxdata-archive-keyring.gpg] https://repos.influxdata.com/$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/influxdb.list

**# Install InfluxDB**

sudo apt update

sudo apt install influxdb2

**# Start and enable the service**

sudo systemctl start influxdb

sudo systemctl enable influxdb

**Next Steps:**

* Open http://localhost:8086 in a browser.
* Create a bucket, generate an API token, and note:
  + Organization Name
  + Bucket Name
  + Token

1. **Arduino IDE (On MQTT Publisher End)**
   1. **Download** the latest Arduino IDE from the official site:  
      <https://www.arduino.cc/en/software>
      1. Choose **Windows Win 10 and newer** (or **Windows ZIP file** for portable install).
   2. **Run the installer** (.exe file) and follow the prompts.
      1. Check **"Install USB drivers"** (important for Arduino boards).
   3. **Launch Arduino IDE** after installation.

**Install Board Support (e.g., ESP32)**

1. Open **Arduino IDE** → **File → Preferences**.
2. In **Additional Boards Manager URLs**, add:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\_esp32\_index.json

1. Go to **Tools → Board → Boards Manager**, search for **ESP32**, and install.
2. Select your board:
   * **Tools → Board → ESP32 Arduino → "ESP32 Dev Module"**
3. **Visual Studio Code**
   1. **Download** VS Code for Windows:  
      <https://code.visualstudio.com/download>
   2. **Run the installer** (.exe file) and follow the prompts.
   3. **Launch VS Code** after installation.
   4. Install Extensions
      1. Open Extensions (Ctrl+Shift+X) and install:
         1. Python (for using Python scripts).
         2. Jupyter (for using Jupyter Notebook for running time series machine learning models).
4. **Python**

**Step 1: Download Python**

* + 1. Go to the official Python website:  
       <https://www.python.org/downloads/>
    2. Click **"Download Python 3.x.x"** (latest stable version).

**Step 2: Run the Installer**

1. Open the downloaded .exe file (e.g., python-3.11.5-amd64.exe).
2. **Check these boxes during installation:**
3. **"Add Python to PATH"** (critical for command-line usage).
4. **"Install launcher for all users"** (recommended).
5. Click **"Install Now"** (default settings are fine).

**Step 3: Verify Installation**

1. Open Command Prompt ( Win + R -> type cmd -> Enter)
2. Run: python –version
3. Expected Output: Python 3.x.x (Depending on the Python Version you’ve installed).

**5) MQTTX MQTT Client**

**Download & Install**

1. **Download** MQTTX for Windows:  
   <https://mqttx.app/download>
2. **Run the installer** (.exe file) and follow the prompts.
3. **Launch MQTTX** after installation.

**Test MQTT Connection**

1. Open MQTTX and click **"New Connection"**.
2. Enter:
   * **Name**: Real-Time Bin Monitoring
   * **Host**: broker.emqx.io
   * **Port**: 8883
3. Click **"Connect"**.
4. Click “**New Subscription**” and input “**sensor/data**” as the topic and select QoS as **1** → If successful, you can receive real-time data waste level monitoring information.

**6) Ubuntu LTS**

# **7) Instructions to run the Python scripts and launch the InfluxDB visualization tool:**

1) Connect the Arduino Microcontroller via USB into the MQTT publisher’s PC.

2) Open a new terminal and type python mqtt\_publish.py on 1 PC and python mqtt\_subscribe.py on another PC, then press Enter.

3) Launch Ubuntu <<version>> LTS

4) Type sudo systemctl start influxdb and Press Enter.

5) Input the password that you’ve set during installation of Ubuntu LTS and Press Enter.

6) Navigate to a web browser and type localhost:8086.

7) Input your username and password that you’ve set during installation of InfluxDB, and press Enter.

8) Navigate to Dashboards, click on create Dashboard and select new dashboard. Create a title for the Dashboard, i.e. Bin Level Monitoring <<month>> <<year>>, replace month and year with the actual month and year, and add the necessary cells as stated with the necessary visualization types and flux query in the waste\_monitoring.txt and click on the tick button at the top right hand corner of the window, and repeat this at the beginning of each month.

9) Open the Predictive Analytics Future Waste Levels Jupyter Notebook file and click on Run all at the top of the navigation bar of VS Code to perform predictive analytics to predict and forecast future waste levels.