Setting Up MQTT Server and MQTT-MongoDB Connection

Prerequisites

Ensure the following guidelines have been completed:

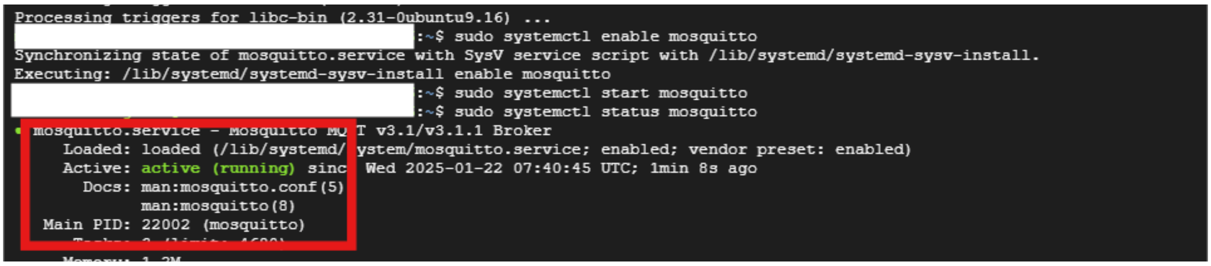
1. Setting Up Arduino IDE for ESP32.
2. Setting Up a Virtual Machine on Google Cloud Platform (GCP).
3. Setting Up MongoDB and MongoDB Charts.

Step 1: Install Mosquitto and Mosquitto Clients

1. Launch a new SSH terminal and install Mosquitto and its clients:
   - `sudo apt-get install mosquitto`
   - `sudo apt-get install mosquitto-clients`

Step 2: Test the Mosquitto Service

1. Enable and start the Mosquitto service:
   - `sudo systemctl enable mosquitto`
   - `sudo systemctl start mosquitto`
   - `sudo systemctl status mosquitto`



```
Processing triggers for libc-bin (2.31-0ubuntu9.16) ...
[redacted]:~$ sudo systemctl enable mosquitto
Synchronizing state of mosquitto.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
[redacted]:~$ sudo systemctl start mosquitto
:~$ sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker
     Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2025-01-22 07:40:45 UTC; 1min 8s ago
       Docs: man:mosquitto.conf(5)
             man:mosquitto(8)
   Main PID: 22002 (mosquitto)
      Tasks: 2 (limit: 4680)
     Memory: 1.2M
```

Step 3: Install Python and Required Libraries

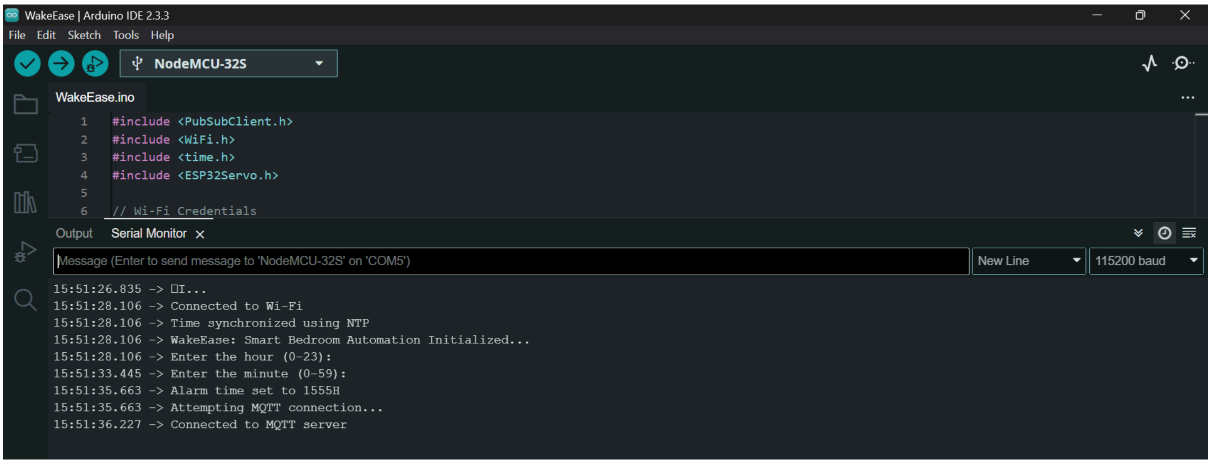
1. Install Python's package manager (pip) and necessary libraries:
   - `sudo apt install python3-pip`
   - `pip install paho-mqtt`
   - `pip install pymongo`
   - `pip install pytz`

Ensure MQTT Broker and IoT System Are Working

These steps ensure that your IoT system is successfully publishing data to the MQTT broker and that the broker is correctly distributing messages to subscribed clients.

Step 1: Connect ESP32 to Laptop and Observe Serial Monitor

1. Connect the ESP32 to your laptop via USB.
2. Open the Serial Monitor in the Arduino IDE.
3. Set the baud rate to 115200 to match the sketch.
4. Observe the Serial Monitor for output messages to verify the ESP32's functionality, such as Wi-Fi connection and data published to the MQTT broker.


WakeEase | Arduino IDE 2.3.3
File Edit Sketch Tools Help
NodeMCU-32S
WakeEase.ino
Output
Serial Monitor
Message (Enter to send message to 'NodeMCU-32S' on 'COM5')
New Line
115200 baud

```
#include <PubSubClient.h>
#include <WiFi.h>
#include <time.h>
#include <ESP32Servo.h>

// Wi-Fi Credentials
```

```
15:51:26.835 -> □I...
15:51:28.106 -> Connected to Wi-Fi
15:51:28.106 -> Time synchronized using NTP
15:51:28.106 -> WakeEase: Smart Bedroom Automation Initialized...
15:51:28.106 -> Enter the hour (0-23):
15:51:33.445 -> Enter the minute (0-59):
15:51:35.663 -> Alarm time set to 1555H
15:51:35.663 -> Attempting MQTT connection...
15:51:36.227 -> Connected to MQTT server
```

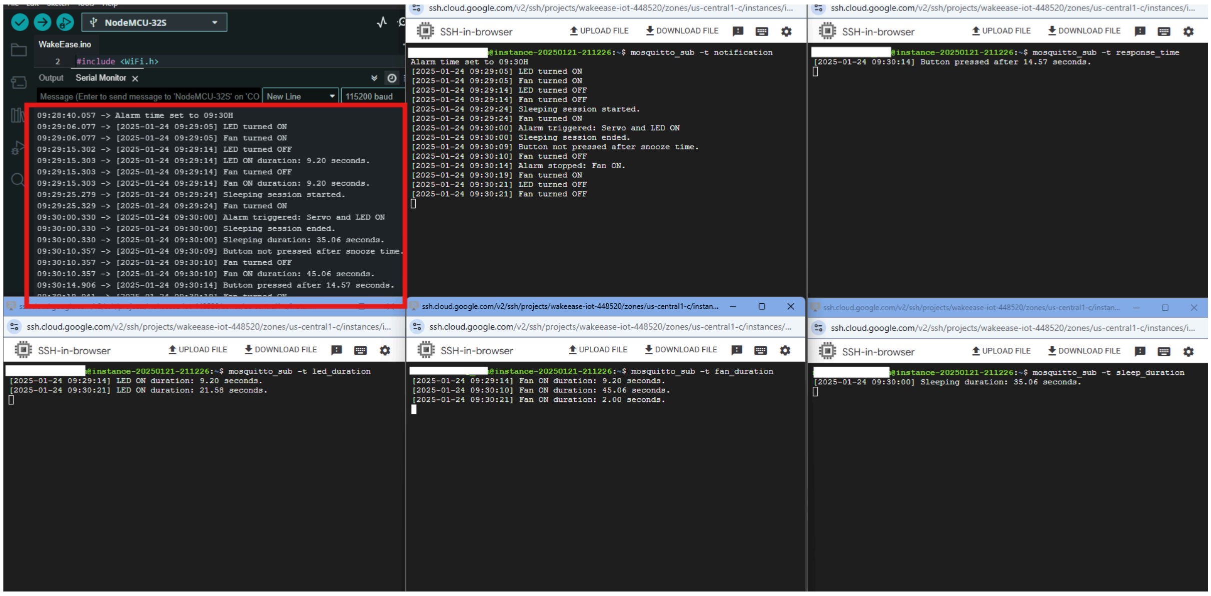
Step 2: Open Five SSH Terminals:

1. In each terminal, subscribe to one of the following MQTT topics:
   - `mosquitto_sub -t notification`
   - `mosquitto_sub -t led_duration`
   - `mosquitto_sub -t fan_duration`
   - `mosquitto_sub -t sleep_duration`
   - `mosquitto_sub -t response_time`

Step 3: Observe the Terminals:

1. Check all terminals for incoming sensor data published by the IoT system.
2. If data appears, this confirms that:
   - ➢ The IoT system is publishing data to the MQTT broker.
   - ➢ The MQTT broker is distributing messages to subscribed clients.

➢ Subscribed clients are successfully receiving data.


NodeMCU-32S
WakeEase.ino
Output
Serial Monitor
Message (Enter to send message to 'NodeMCU-32S' on 'CO
New Line
115200 baud
ssh.cloud.google.com/v2/ssh/projects/wakeease-iot-448520/zones/us-central1-c/instances/i...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

```
2   #include <WiFi.h>
```

```
09:28:40.057 -> Alarm time set to 09:30H
09:29:06.077 -> [2025-01-24 09:29:05] LED turned ON
09:29:06.077 -> [2025-01-24 09:29:05] Fan turned ON
09:29:15.302 -> [2025-01-24 09:29:14] LED turned OFF
09:29:15.303 -> [2025-01-24 09:29:14] LED ON duration: 9.20 seconds.
09:29:15.303 -> [2025-01-24 09:29:14] Fan turned OFF
09:29:15.303 -> [2025-01-24 09:29:14] Fan ON duration: 9.20 seconds.
09:29:25.279 -> [2025-01-24 09:29:24] Sleeping session started.
09:29:25.329 -> [2025-01-24 09:29:24] Fan turned ON
09:30:00.330 -> [2025-01-24 09:30:00] Alarm triggered: Servo and LED ON
09:30:00.330 -> [2025-01-24 09:30:00] Sleeping session ended.
09:30:00.330 -> [2025-01-24 09:30:00] Sleeping duration: 35.06 seconds.
09:30:10.357 -> [2025-01-24 09:30:09] Button not pressed after snooze time.
09:30:10.357 -> [2025-01-24 09:30:10] Fan turned OFF
09:30:10.357 -> [2025-01-24 09:30:10] Fan ON duration: 45.06 seconds.
09:30:14.906 -> [2025-01-24 09:30:14] Button pressed after 14.57 seconds.
09:30:19.941 -> [2025-01-24 09:30:19] Fan turned ON
```

```
[redacted]@instance-20250121-211226:~$ mosquitto_sub -t notification
Alarm time set to 09:30H
[2025-01-24 09:29:05] LED turned ON
[2025-01-24 09:29:05] Fan turned ON
[2025-01-24 09:29:14] LED turned OFF
[2025-01-24 09:29:14] Fan turned OFF
[2025-01-24 09:29:24] Sleeping session started.
[2025-01-24 09:29:24] Fan turned ON
[2025-01-24 09:30:00] Alarm triggered: Servo and LED ON
[2025-01-24 09:30:00] Sleeping session ended.
[2025-01-24 09:30:09] Button not pressed after snooze time.
[2025-01-24 09:30:10] Fan turned OFF
[2025-01-24 09:30:14] Alarm stopped: Fan ON.
[2025-01-24 09:30:19] Fan turned ON
[2025-01-24 09:30:21] LED turned OFF
[2025-01-24 09:30:21] Fan turned OFF
```

```
[redacted]@instance-20250121-211226:~$ mosquitto_sub -t response_time
[2025-01-24 09:30:14] Button pressed after 14.57 seconds.
```

```
[redacted]@instance-20250121-211226:~$ mosquitto_sub -t led_duration
[2025-01-24 09:29:14] LED ON duration: 9.20 seconds.
[2025-01-24 09:30:21] LED ON duration: 21.58 seconds.
```

```
[redacted]@instance-20250121-211226:~$ mosquitto_sub -t fan_duration
[2025-01-24 09:29:14] Fan ON duration: 9.20 seconds.
[2025-01-24 09:30:10] Fan ON duration: 45.06 seconds.
[2025-01-24 09:30:21] Fan ON duration: 2.00 seconds.
```

```
[redacted]@instance-20250121-211226:~$ mosquitto_sub -t sleep_duration
[2025-01-24 09:30:00] Sleeping duration: 35.06 seconds.
```

Troubleshooting: If No Data Appears

❖ **Check MQTT Topics**:

- Ensure that the topics subscribed in the terminals (notification, duration, and response_time) match the topics defined in WakeEase.ino.

❖ **Verify the External IP**:

- Confirm that the external IP address in WakeEase.ino (line 11) matches the external IP of the current VM instance.

❖ **Check Network Connectivity**:

- The Wi-Fi settings in WakeEase.ino (SSID and password) are correct.
- The IoT system is connected to the configured Wi-Fi.
- The IoT system is within the Wi-Fi range.

❖ **Verify Mosquitto Service Status**:

- Ensure the Mosquitto broker is running:

  -
    ```
    sudo systemctl status mosquitto
    ```

❖ **Restart Mosquitto Broker**:

- Restart the Mosquitto service and try again:

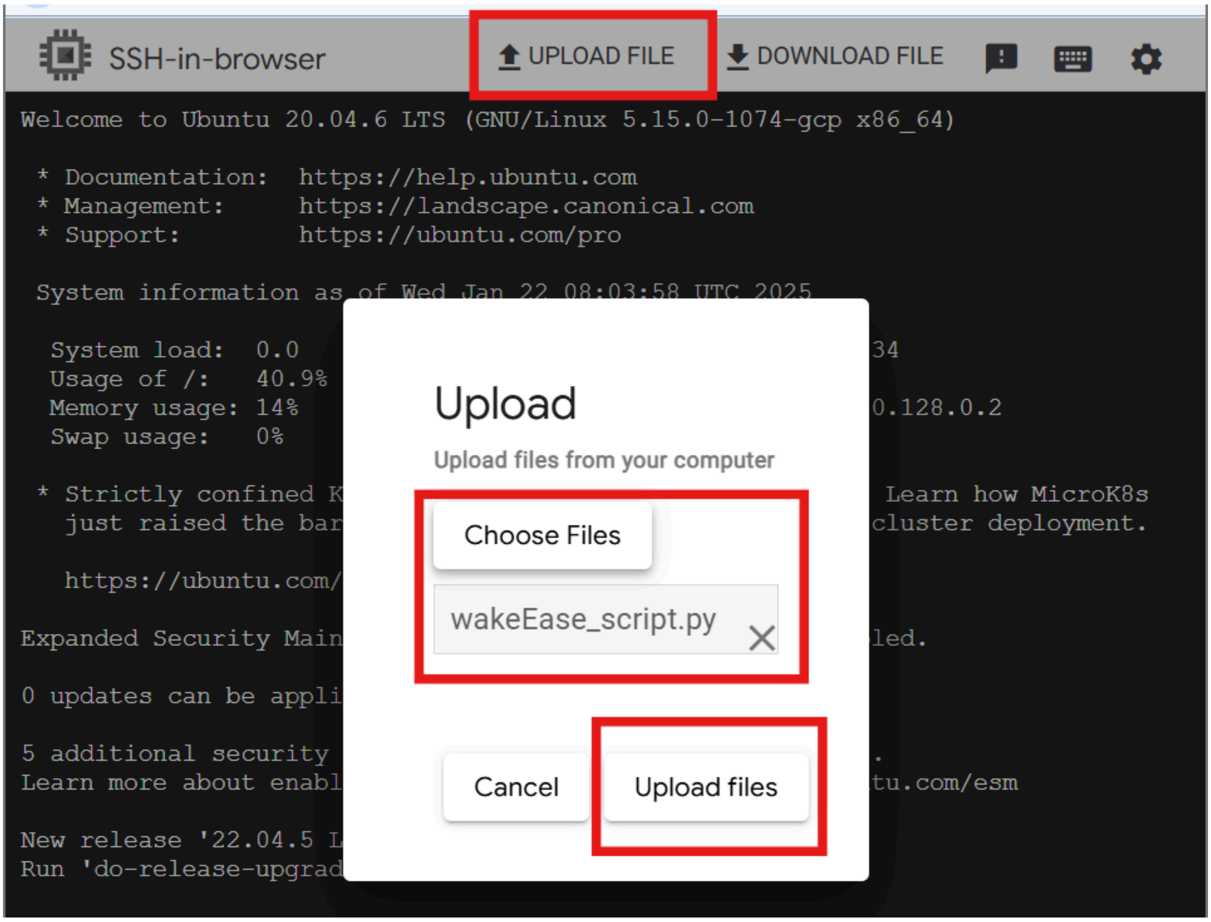
  -
    ```
    sudo systemctl restart mosquitto
    ```

Verify MQTT-MongoDB Connection

These steps ensure that data received by the MQTT broker is correctly ingested and stored in MongoDB.

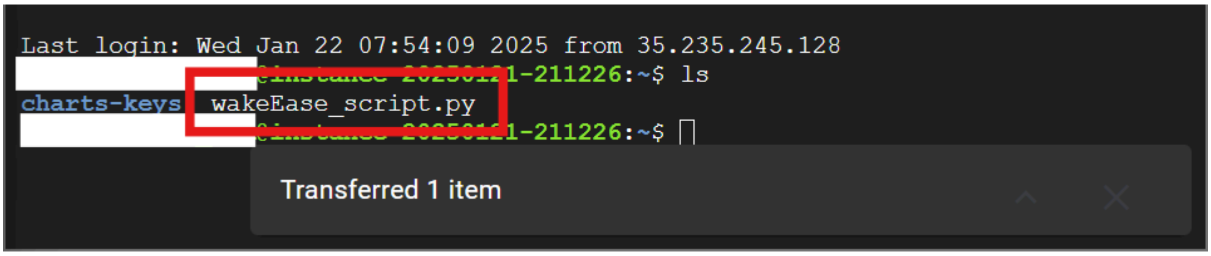
Step 1: Upload Python Script

1. Upload the wakeEase_script.py file to the VM using the **Upload File** button.



```
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1074-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Jan 22 08:03:58 UTC 2025

  System load:  0.0                                  34
  Usage of /:   40.9%
  Memory usage: 14%                                  0.128.0.2
  Swap usage:   0%

 * Strictly confined K                                Learn how MicroK8s
   just raised the bar                               cluster deployment.

   https://ubuntu.com/

Expanded Security Main                               led.

0 updates can be appli

5 additional security                                .
Learn more about enabl                               tu.com/esm

New release '22.04.5 L
Run 'do-release-upgrad
```

2. Confirm the file upload by listing the contents of the directory:
   - `ls`

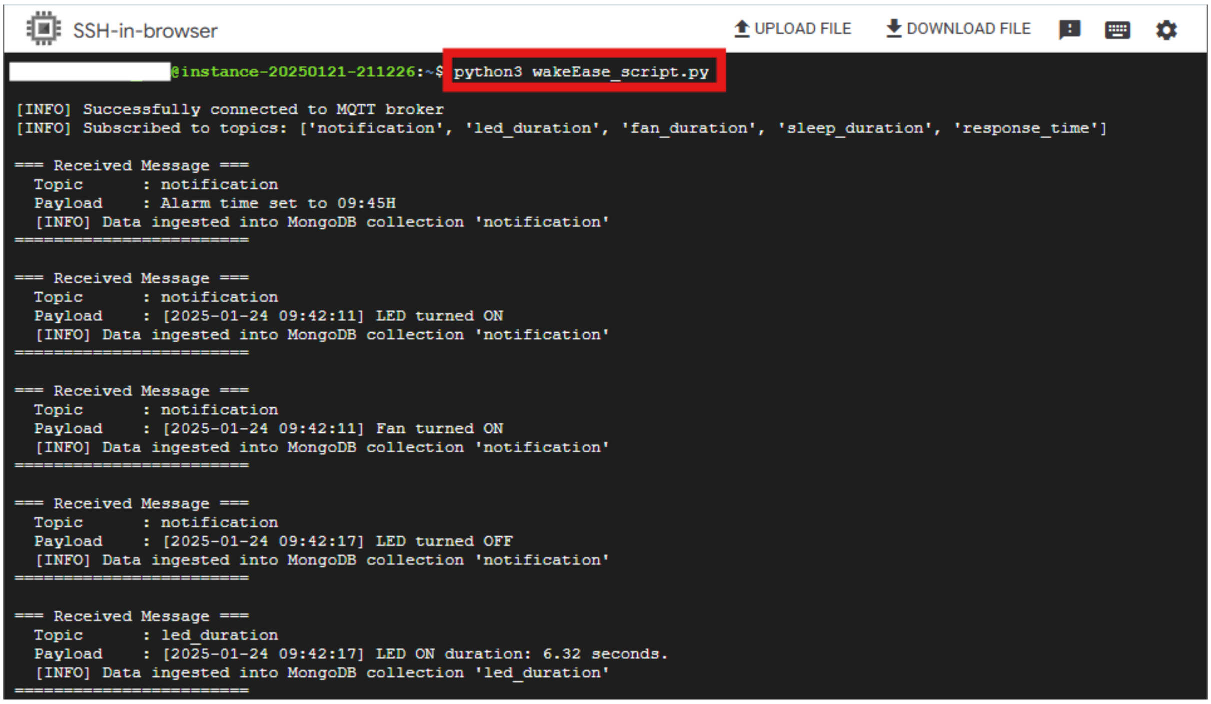


```
Last login: Wed Jan 22 07:54:09 2025 from 35.235.245.128
[redacted]@instance-20250121-211226:~$ ls
charts-keys  wakeEase_script.py
[redacted]@instance-20250121-211226:~$
```

Transferred 1 item

Step 2: Run the Python Script:

1. Execute the script to process MQTT data and store it in MongoDB:
   - `python3 wakeEase_script.py`
2. Observe the terminal for logs indicating successful data ingestion.

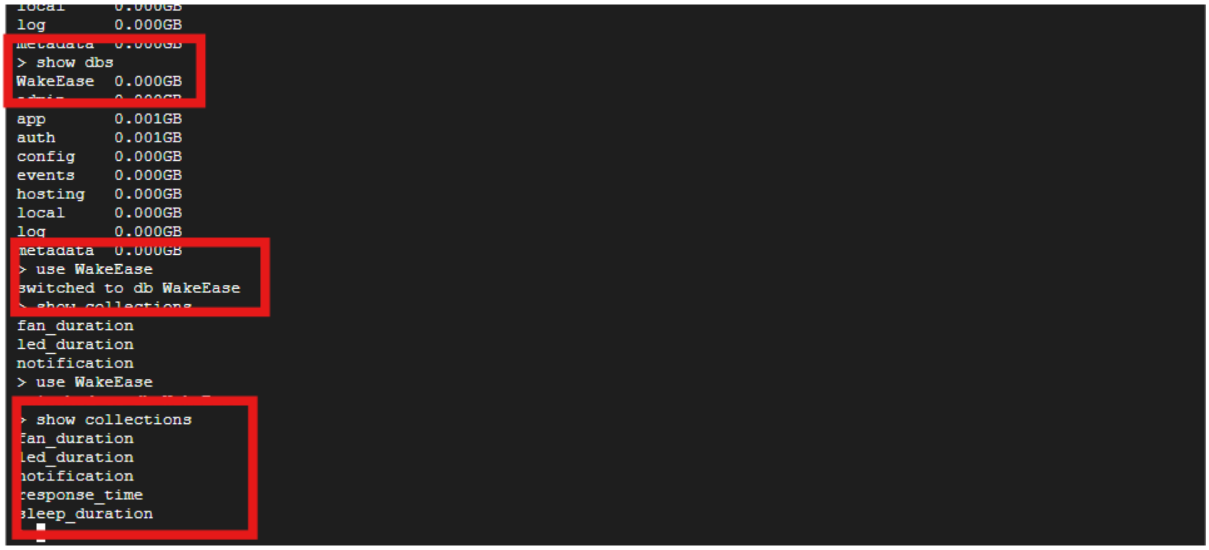


```
[redacted]@instance-20250121-211226:~$ python3 wakeEase_script.py

[INFO] Successfully connected to MQTT broker
[INFO] Subscribed to topics: ['notification', 'led_duration', 'fan_duration', 'sleep_duration', 'response_time']

=== Received Message ===
  Topic      : notification
  Payload    : Alarm time set to 09:45H
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : notification
  Payload    : [2025-01-24 09:42:11] LED turned ON
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : notification
  Payload    : [2025-01-24 09:42:11] Fan turned ON
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : notification
  Payload    : [2025-01-24 09:42:17] LED turned OFF
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : led_duration
  Payload    : [2025-01-24 09:42:17] LED ON duration: 6.32 seconds.
  [INFO] Data ingested into MongoDB collection 'led_duration'
========================
```

Step 3: Verify Data in MongoDB:

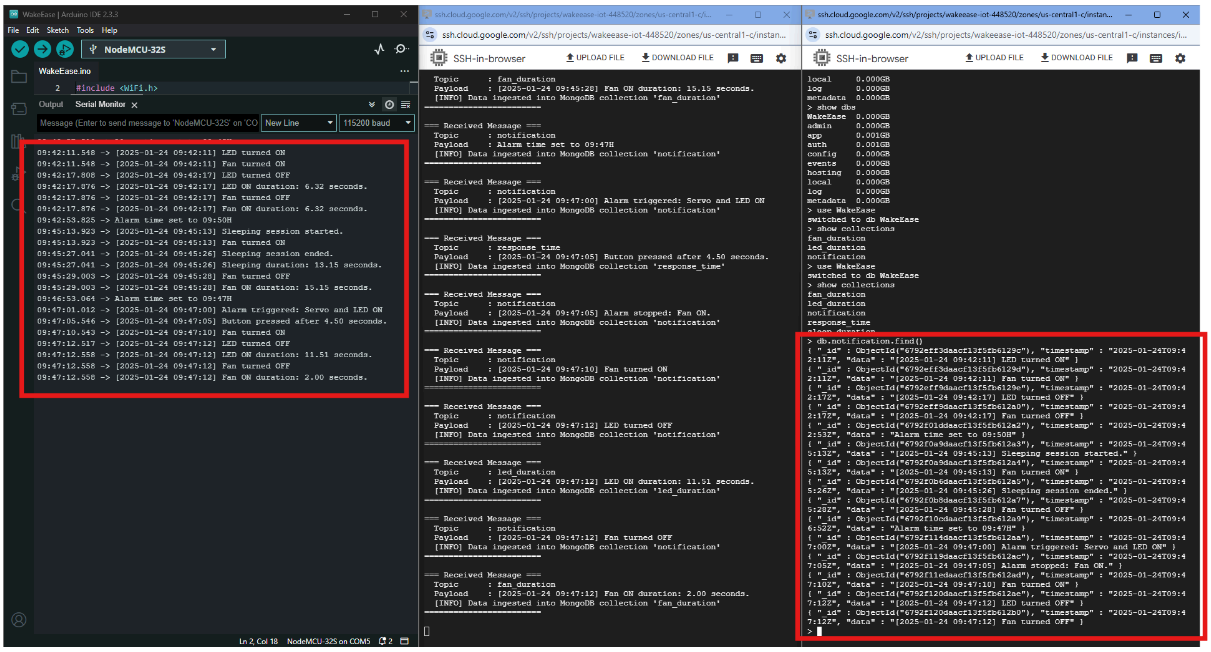
Note: You must send some readings from the IoT system first. Otherwise, the "WakeEase" database and collections (notification, duration, and response_time) will not be created or available in MongoDB.

1. Launch the MongoDB shell:
   - `mongo`
2. Show all available databases in MongoDB:
   - `show dbs`
3. In mongoDB shell, switch to the **WakeEase** database:
   - `use WakeEase`
4. Show collections in **WakeEase** database:
   - `show collections`



```
local      0.000GB
log        0.000GB
metadata   0.000GB
> show dbs
WakeEase   0.000GB
admin      0.000GB
app        0.001GB
auth       0.001GB
config     0.000GB
events     0.000GB
hosting    0.000GB
local      0.000GB
log        0.000GB
metadata   0.000GB
> use WakeEase
switched to db WakeEase
> show collections
fan_duration
led_duration
notification
> use WakeEase
switched to db WakeEase
> show collections
fan_duration
led_duration
notification
response_time
sleep_duration
```

5. Check the collections for stored data [IoT topics]:
   - db.notification.find()
   - db.led_duration.find()
   - db.fan_duration.find()
   - db.sleep_duration.find()
   - db.response_time.find()
6. If data is present, the MQTT-MongoDB connection is functioning correctly.



WakeEase | Arduino IDE 2.3.3
File Edit Sketch Tools Help
NodeMCU-32S
WakeEase.ino
Output Serial Monitor
Message (Enter to send message to 'NodeMCU-32S' on 'CO
New Line
115200 baud
Ln 2, Col 18 NodeMCU-32S on COM5
ssh.cloud.google.com/v2/ssh/projects/wakeease-iot-448520/zones/us-central1-c/instan...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

```
2    #include <WiFi.h>
```

```
09:42:11.548 -> [2025-01-24 09:42:11] LED turned ON
09:42:11.548 -> [2025-01-24 09:42:11] Fan turned ON
09:42:17.808 -> [2025-01-24 09:42:17] LED turned OFF
09:42:17.876 -> [2025-01-24 09:42:17] LED ON duration: 6.32 seconds.
09:42:17.876 -> [2025-01-24 09:42:17] Fan turned OFF
09:42:17.876 -> [2025-01-24 09:42:17] Fan ON duration: 6.32 seconds.
09:42:53.825 -> Alarm time set to 09:50H
09:45:13.923 -> [2025-01-24 09:45:13] Sleeping session started.
09:45:13.923 -> [2025-01-24 09:45:13] Fan turned ON
09:45:27.041 -> [2025-01-24 09:45:26] Sleeping session ended.
09:45:27.041 -> [2025-01-24 09:45:26] Sleeping duration: 13.15 seconds.
09:45:29.003 -> [2025-01-24 09:45:28] Fan turned OFF
09:45:29.003 -> [2025-01-24 09:45:28] Fan ON duration: 15.15 seconds.
09:46:53.064 -> Alarm time set to 09:47H
09:47:01.012 -> [2025-01-24 09:47:00] Alarm triggered: Servo and LED ON
09:47:05.546 -> [2025-01-24 09:47:05] Button pressed after 4.50 seconds.
09:47:10.543 -> [2025-01-24 09:47:10] Fan turned ON
09:47:12.517 -> [2025-01-24 09:47:12] LED turned OFF
09:47:12.558 -> [2025-01-24 09:47:12] LED ON duration: 11.51 seconds.
09:47:12.558 -> [2025-01-24 09:47:12] Fan turned OFF
09:47:12.558 -> [2025-01-24 09:47:12] Fan ON duration: 2.00 seconds.
```

```
  Topic      : fan_duration
  Payload    : [2025-01-24 09:45:28] Fan ON duration: 15.15 seconds.
  [INFO] Data ingested into MongoDB collection 'fan_duration'
========================

=== Received Message ===
  Topic      : notification
  Payload    : Alarm time set to 09:47H
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : notification
  Payload    : [2025-01-24 09:47:00] Alarm triggered: Servo and LED ON
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : response_time
  Payload    : [2025-01-24 09:47:05] Button pressed after 4.50 seconds.
  [INFO] Data ingested into MongoDB collection 'response_time'
========================

=== Received Message ===
  Topic      : notification
  Payload    : [2025-01-24 09:47:05] Alarm stopped: Fan ON.
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : notification
  Payload    : [2025-01-24 09:47:10] Fan turned ON
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : notification
  Payload    : [2025-01-24 09:47:12] LED turned OFF
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : led_duration
  Payload    : [2025-01-24 09:47:12] LED ON duration: 11.51 seconds.
  [INFO] Data ingested into MongoDB collection 'led_duration'
========================

=== Received Message ===
  Topic      : notification
  Payload    : [2025-01-24 09:47:12] Fan turned OFF
  [INFO] Data ingested into MongoDB collection 'notification'
========================

=== Received Message ===
  Topic      : fan_duration
  Payload    : [2025-01-24 09:47:12] Fan ON duration: 2.00 seconds.
  [INFO] Data ingested into MongoDB collection 'fan_duration'
========================
```

```
local     0.000GB
log       0.000GB
metadata  0.000GB
> show dbs
WakeEase  0.000GB
admin     0.000GB
app       0.001GB
auth      0.001GB
config    0.000GB
events    0.000GB
hosting   0.000GB
local     0.000GB
log       0.000GB
metadata  0.000GB
> use WakeEase
switched to db WakeEase
> show collections
fan_duration
led_duration
notification
> use WakeEase
switched to db WakeEase
> show collections
fan_duration
led_duration
notification
response_time
sleep_duration
> db.notification.find()
{ "_id" : ObjectId("6792eff3daacf13f5fb6129c"), "timestamp" : "2025-01-24T09:4
2:11Z", "data" : "[2025-01-24 09:42:11] LED turned ON" }
{ "_id" : ObjectId("6792eff3daacf13f5fb6129d"), "timestamp" : "2025-01-24T09:4
2:11Z", "data" : "[2025-01-24 09:42:11] Fan turned ON" }
{ "_id" : ObjectId("6792eff9daacf13f5fb6129e"), "timestamp" : "2025-01-24T09:4
2:17Z", "data" : "[2025-01-24 09:42:17] LED turned OFF" }
{ "_id" : ObjectId("6792eff9daacf13f5fb612a0"), "timestamp" : "2025-01-24T09:4
2:17Z", "data" : "[2025-01-24 09:42:17] Fan turned OFF" }
{ "_id" : ObjectId("6792f01ddaacf13f5fb612a2"), "timestamp" : "2025-01-24T09:4
2:53Z", "data" : "Alarm time set to 09:50H" }
{ "_id" : ObjectId("6792f0a9daacf13f5fb612a3"), "timestamp" : "2025-01-24T09:4
5:13Z", "data" : "[2025-01-24 09:45:13] Sleeping session started." }
{ "_id" : ObjectId("6792f0a9daacf13f5fb612a4"), "timestamp" : "2025-01-24T09:4
5:13Z", "data" : "[2025-01-24 09:45:13] Fan turned ON" }
{ "_id" : ObjectId("6792f0b6daacf13f5fb612a5"), "timestamp" : "2025-01-24T09:4
5:26Z", "data" : "[2025-01-24 09:45:26] Sleeping session ended." }
{ "_id" : ObjectId("6792f0b8daacf13f5fb612a7"), "timestamp" : "2025-01-24T09:4
5:28Z", "data" : "[2025-01-24 09:45:28] Fan turned OFF" }
{ "_id" : ObjectId("6792f10cdaacf13f5fb612a9"), "timestamp" : "2025-01-24T09:4
6:52Z", "data" : "Alarm time set to 09:47H" }
{ "_id" : ObjectId("6792f114daacf13f5fb612aa"), "timestamp" : "2025-01-24T09:4
7:00Z", "data" : "[2025-01-24 09:47:00] Alarm triggered: Servo and LED ON" }
{ "_id" : ObjectId("6792f119daacf13f5fb612ac"), "timestamp" : "2025-01-24T09:4
7:05Z", "data" : "[2025-01-24 09:47:05] Alarm stopped: Fan ON." }
{ "_id" : ObjectId("6792f11edaacf13f5fb612ad"), "timestamp" : "2025-01-24T09:4
7:10Z", "data" : "[2025-01-24 09:47:10] Fan turned ON" }
{ "_id" : ObjectId("6792f120daacf13f5fb612ae"), "timestamp" : "2025-01-24T09:4
7:12Z", "data" : "[2025-01-24 09:47:12] LED turned OFF" }
{ "_id" : ObjectId("6792f120daacf13f5fb612b0"), "timestamp" : "2025-01-24T09:4
7:12Z", "data" : "[2025-01-24 09:47:12] Fan turned OFF" }
>
```

Troubleshooting: If Data Is Not Stored in MongoDB

❖ **Check the Python Script**:

- Ensure that wakeEase_script.py is running without errors.
- Confirm that MQTT topics in wakeEase_script.py (notification, duration, and response_time) match the ones defined in WakeEase.ino and MQTT broker.
- Confirm the external IP address in wakeEase_script.py (line 11) matches the current VM instance's external IP.

❖ **Verify MongoDB Service**:

- Confirm MongoDB is running:
  ```
  sudo systemctl status mongodb
  ```
- Restart MongoDB if it's not running:
  ```
  sudo systemctl restart mongodb
  ```