

		
Laboratory 8: Smart Alert Logging with MariaDB and History Dashboard	School of Applied Digital Technology	
Name:	ID:	Section:
Name:	ID:	Section:
Date:	<b>Due date:</b>	

## Objectives

- Log all smart home events (Gas, Motion, Temperature, Light) into MariaDB.
- Enable Home Assistant Recorder and History visualization.
- Create a new Dashboard tab to visualize sensor trends and alert history.
- Verify data storage directly from phpMyAdmin.

### 1. Database Configuration (MariaDB)

- Your setup already includes the recorder integration in configuration.yaml 
- Make sure this section looks like this:

Yaml:

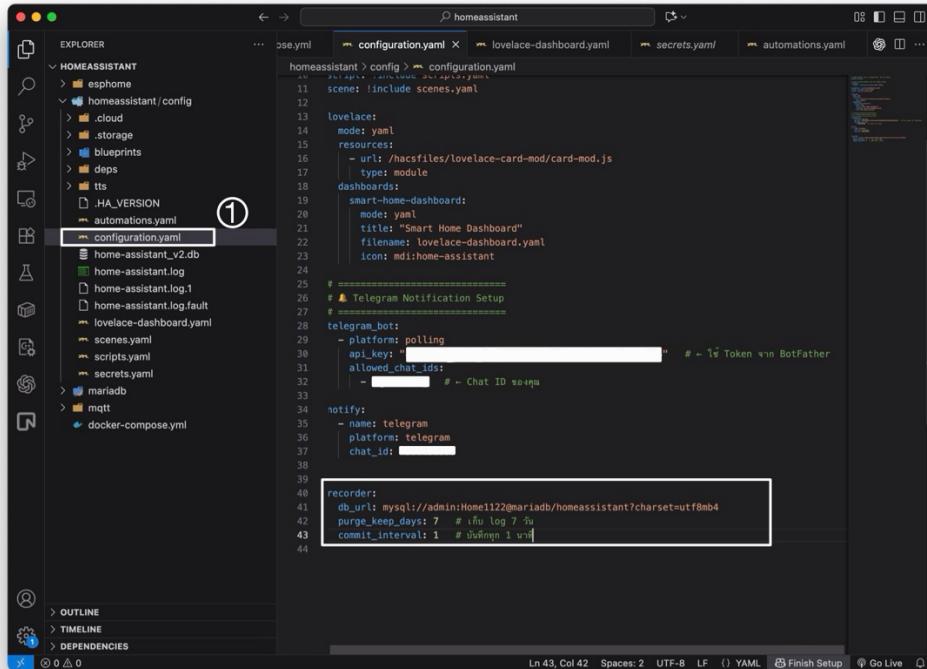
recorder:

```
db_url: mysql://admin:Home1122@mariadb/homeassistant?charset=utf8mb4
purge_keep_days: 7
commit_interval: 1
```

## Verify MariaDB Connection

1. Run docker ps to confirm the container mariadb is running.
  2. Open phpMyAdmin in your browser → <http://localhost:8080>
    - Server: mariadb
    - Username: admin
    - Password: Home1122
    - Database: homeassistant
  3. Check if you can see tables like states, events, recorder\_runs.

If yes — Home Assistant is successfully logging data to MariaDB 



## 2. Enable History and Logbook

- Add or update the following lines in your configuration.yaml:

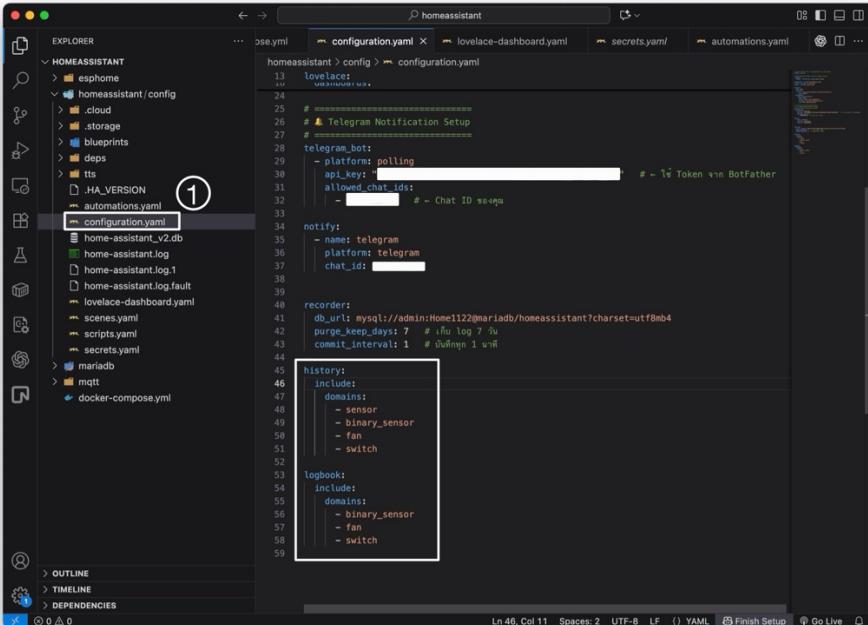
Yaml:

```
history:  
  include:  
    domains:  
      - sensor  
      - binary_sensor  
      - fan  
      - switch
```

logbook:

```
include:  
  domains:  
    - binary_sensor  
    - fan  
    - switch
```

This ensures all device state changes and events are stored and displayed in the “History” and “Logbook” sections.



```
homeassistant > config > configuration.yaml  
13   history:  
14     include:  
15       domains:  
16         - sensor  
17         - binary_sensor  
18         - fan  
19         - switch  
20  
21   logbook:  
22     include:  
23       domains:  
24         - binary_sensor  
25         - fan  
26         - switch
```

- Open: /config/lovelace-dashboard.yaml

Then add this **new view** at the bottom (after your Lab 7 dashboard views):

Yaml:

```

- title: "Alert History"
  icon: mdi:database-clock
  cards:
    #  Logbook View
    - type: logbook
      title: "  Recent Events"
      entities:
        - binary_sensor.project_gas_alarm_active
        - binary_sensor.project_living_room_motion
        - fan.project_living_room_fan
        - switch.project_active_buzzer
        - sensor.project_living_room_temperature

    #  History Graphs
    - type: history-graph
      title: "  Environment Trends (Last 12 Hours)"
      hours_to_show: 12
      refresh_interval: 30
      entities:
        - entity: sensor.project_living_room_temperature
          name: "Temperature (°C)"
        - entity: sensor.project_living_room_humidity
          name: "Humidity (%)"
        - entity: sensor.project_mq_2_gas_level
  
```

```

        name: "Gas Level (V)"

- entity: sensor.project_ldr_light_level

        name: "Light Level (V)"

```

## # ⚡ Sensor Status Summary

```

- type: entities

title: "⚡ Sensor States Summary"

entities:

- entity: binary_sensor.project_gas_alarm_active

        name: "Gas Alarm"

- entity: binary_sensor.project_living_room_motion

        name: "Motion Sensor"

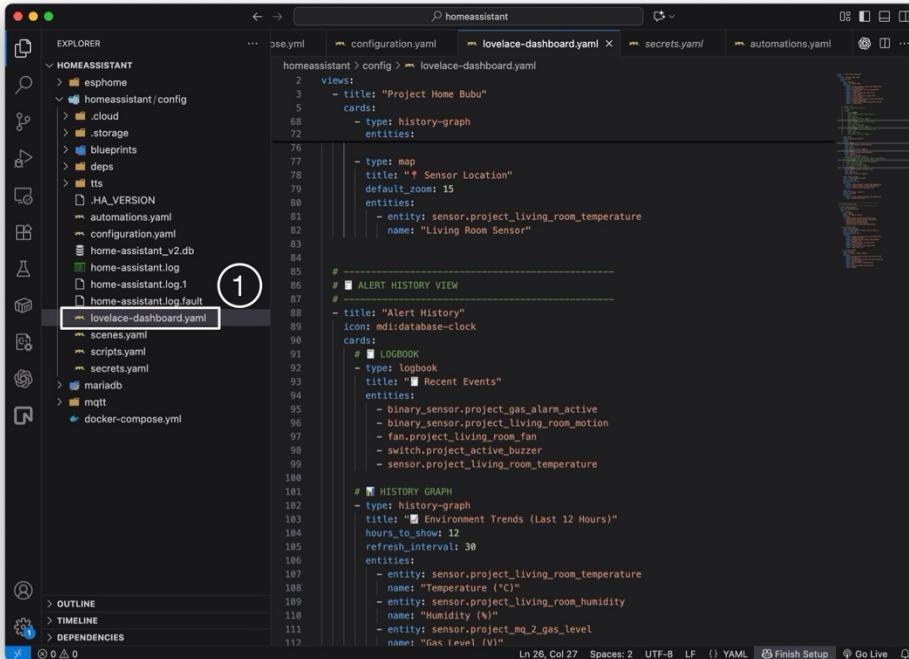
- entity: fan.project_living_room_fan

        name: "Fan"

- entity: switch.project_active_buzzer

        name: "Buzzer"

```



The screenshot shows the Home Assistant configuration files in VS Code. The 'EXPLORER' sidebar on the left lists the project structure, including 'HOMEASSISTANT', '.HA\_VERSION', 'automations.yaml', 'configuration.yaml', 'home-assistant\_v2.db', 'home-assistant.log', 'home-assistant.log.1', 'home-assistant.log.fault', 'lovelace-dashboard.yaml', 'scenes.yaml', 'scripts.yaml', 'mariaDB', 'mqtt', and 'docker-compose.yaml'. A circled '1' is placed over the 'lovelace-dashboard.yaml' file in the sidebar.

The main editor area displays the 'lovelace-dashboard.yaml' file content:

```

views:
  - title: "Project Home Bubu"
    cards:
      - type: history-graph
        entities:
          - entity: sensor.project_living_room_temperature
            name: "Living Room Sensor"

# ----- ALERT HISTORY VIEW -----
# ----- HISTORY GRAPH -----

```

The code continues with definitions for 'ALERT HISTORY' (with 'LOGBOOK' card), 'HISTORY GRAPH' (with 'Enviroment Trends (Last 12 Hours)' card), and various entities like 'binary\_sensor.project\_gas\_alarm\_active', 'binary\_sensor.project\_living\_room\_motion', 'fan.project\_living\_room\_fan', 'switch.project\_active\_buzzer', and 'sensor.project\_living\_room\_temperature'.

#### 4. Test and Verify

- Simulate Sensor Triggers
- Go to Developer Tools → States and manually change values:

Test Case	Entity	New State	Expected Result
Gas alert	binary_sensor.project_gas_alarm_active	on	Telegram alert sent
High temperature sensor	project_living_room_temperature	35	Telegram alert sent
Motion alert	binary_sensor.project_living_room_motion	on	Telegram alert sent
Light alert	sensor.project_ldr_light_level	1.0	LED ON (if configured)

Then check:

1. Dashboard → Alert History shows your recent events.
2. History Graph updates every 30 seconds with real-time data.
3. phpMyAdmin → states table updates with new entity states.

#### 5 (Optional). Export Data from MariaDB

- If you'd like to export logs into a .csv report:

Step 1: Open phpMyAdmin

- Go to <http://localhost:8080>
- Select the database → **homeassistant**

Step 2: Create a “View” that joins both tables

- In the **SQL** tab, run this command:

SQL:

```
CREATE OR REPLACE VIEW v_states_full AS  
SELECT  
    sm.entity_id AS entity_id,  
    s.state AS state,  
    s.last_changed AS last_changed  
FROM states AS s  
JOIN states_meta AS sm  
ON s.metadata_id = sm.metadata_id;
```

✓ Once executed, you'll see a new view called `v_states_full` in the sidebar under the “Views” section.

Step 3: Query the sensors you want to export

- Run this SQL command:

SQL:

```
SELECT
```

```
    entity_id,
```

```
    state,
```

```
last_changed
```

```
FROM v_states_full
```

```
WHERE entity_id IN (
```

```
'binary_sensor.project_gas_alarm_active',
```

```
'binary_sensor.project_living_room_motion',
```

```
'sensor.project_living_room_temperature'
```

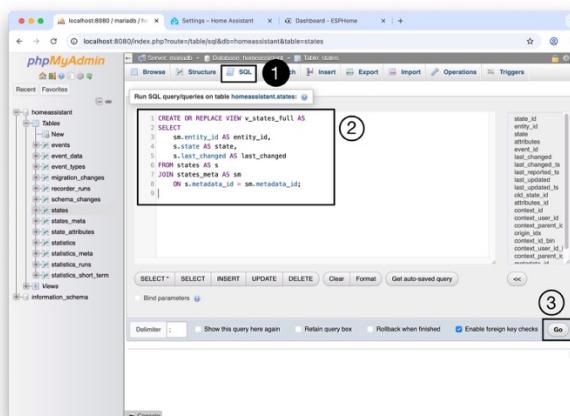
```
)
```

```
ORDER BY last_changed DESC;
```

#### Step 4: Export to CSV

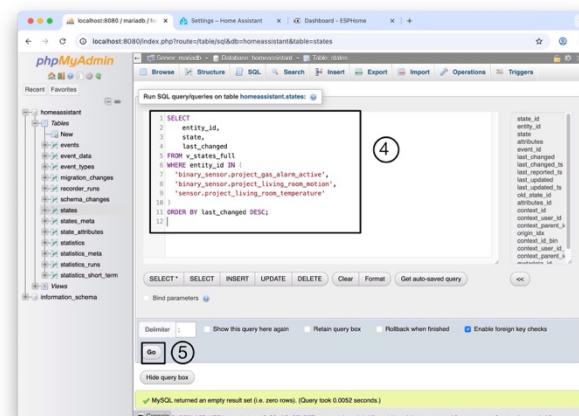
- Click Export at the top of phpMyAdmin
- Choose CSV as the format
- Set a file name (e.g. sensor\_log\_report.csv)
- Click Go 

You'll get a downloadable CSV file that can be opened in Excel or Google Sheets.



The screenshot shows the phpMyAdmin interface with the following details:

- Database:** homeassistant
- Table:** v\_states\_full
- Query:** A SQL query to create a view named v\_states\_full. The query includes aliases for sm.entity\_id as entity\_id, s.state as state, and s.last\_change as last\_changed. It selects from states AS s and joins metadata AS sm on sm.metadata\_id = s.metadata\_id.
- Buttons:** SELECT\*, INSERT, UPDATE, DELETE, Clear, Format, Get auto-saved query, Bind parameters, Delimiter, Show this query here again, Retain query box, Rollback when finished, Enable foreign key checks.
- Numbered Callouts:** ① points to the 'Tables' section in the sidebar; ② points to the query editor; ③ points to the 'Go' button.



The screenshot shows the phpMyAdmin interface with the following details:

- Database:** homeassistant
- Table:** v\_states\_full
- Query:** The same SQL query as the previous screenshot, now with additional WHERE clauses and ORDER BY clause.
- Buttons:** SELECT\*, INSERT, UPDATE, DELETE, Clear, Format, Get auto-saved query, Bind parameters, Delimiter, Show this query here again, Retain query box, Rollback when finished, Enable foreign key checks.
- Numbered Callouts:** ④ points to the query editor; ⑤ points to the 'do' button.
- Status Bar:** MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

The image shows two screenshots of the phpMyAdmin interface for the 'homeassistant' database.

**Screenshot 6:** The left screenshot shows the 'Tables' section with the 'v\_states\_full' table selected. The table structure is displayed with columns: entity\_id, state, last\_changed. A SQL query is shown at the top: `SELECT entity_id, state, last_changed FROM v_states_full WHERE entity_id IN ('binary_sensor.project_gas_alarm_active', 'binary_sensor.project_living_room_temperature')`. Below the table, there are 'Query results operations' buttons: 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'. The 'Export' button is highlighted with a circled number 6.

**Screenshot 7:** The right screenshot shows the 'Export' dialog for the 'v\_states\_full' table. The 'Format' dropdown is set to 'SQL' (circled with 7). The 'Rows' section has 'Dump all rows' selected. The 'Export method' dropdown is set to 'Quick - display only the minimal options'.

**Screenshot 8:** The left screenshot shows the 'Export' dialog again, this time with the 'Format' dropdown set to 'CSV' (circled with 8). The 'Rows' section has 'Dump all rows' selected. The 'Export method' dropdown is set to 'Quick - display only the minimal options'.

**Screenshot 9:** The right screenshot shows the 'Export' dialog with the 'Format' dropdown set to 'CSV' (circled with 9). The 'Rows' section has 'Dump all rows' selected. The 'Export method' dropdown is set to 'Quick - display only the minimal options'.

Continued on the next page.

## Take 1: Correlate Events and States

### Objective

To understand how events and states in Home Assistant are related and to analyze how system activities (events) trigger changes in sensor states.

You will use SQL to find correlations between event records and state changes within a defined time window.

### Requirements

1. Use the MariaDB database connected to Home Assistant (homeassistant schema).

2. Focus on these key tables:

- events — stores all system-level actions (automation triggered, service called, etc.)
- states — stores entity states and sensor readings.
- states\_meta — maps metadata\_id to the entity\_id name.

3. Write a query to:

- Detect event-state pairs that happened within  $\pm 60$  seconds of each other.
- Filter only for binary\_sensor.project\_gas\_alarm\_active entity.
- Show the event\_type, event timestamp, state change timestamp, and entity\_id.

4. Analyze the output — explain what kind of events usually happen when the gas alarm is triggered.

### Database Relationships Overview

- states.metadata\_id  states\_meta.metadata\_id → provides the entity\_id name.
- events.time\_fired\_ts → timestamp (double precision, Unix time format).

- `states.last_changed_ts` → timestamp (double precision).
- You can use `ABS(events.time_fired_ts - states.last_changed_ts)` to measure time difference in seconds.

Code:

----- Have a good day -----

Answer:

```
SELECT  
    sm.entity_id,  
    s.state,  
    FROM_UNIXTIME(s.last_changed_ts) AS state_time,  
    e.event_type,  
    FROM_UNIXTIME(e.time_fired_ts) AS event_time,  
    ROUND(ABS(e.time_fired_ts - s.last_changed_ts), 1) AS time_diff_sec  
FROM homeassistant.events AS e  
JOIN homeassistant.states AS s  
ON ABS(e.time_fired_ts - s.last_changed_ts) <= 60  
JOIN homeassistant.states_meta AS sm  
ON s.metadata_id = sm.metadata_id  
WHERE sm.entity_id = 'binary_sensor.project_gas_alarm_active'  
ORDER BY s.last_changed_ts DESC  
LIMIT 15;
```