LEARNING MATERIAL OF HOME ASSISTANT

SCHOOL OF APPLIED DIGITAL TECHNOLOGY
IN COMPUTER ENGINEERING

EXAMINING COMMITTEE

ADVISOR BY : Aj. THOGCHAI YOOYATIVONG

COMMITTEE 1 BY : Aj. CHAYAPOL KAMYOD

COMMITTEE 2 BY : Mr. KATAWUT ROJANASAROCH

Project Members

Krittichat Chompuming

Kanjanaroj Khamkhom

Naphaphut Suwannasai

Background and Rational

As smart home technology becomes more accessible, Home Assistant stands out as a powerful open-source platform that integrates various devices into one system. However, its complexity can challenge new users. This manual aims to simplify the process by providing clear, step-by-step guidance on installation, setup, and integration. It helps users harness Home Assistant’s full potential to create a smarter, safer, and more energy-efficient home.

Objective

Provide basic knowledge

Introduce beginners to Home Assistant’s core concepts, features, and benefits for a clear understanding of the system.

Installation introduction

Explain installation steps and initial setup to help users start using the system smoothly and efficiently.

Develop user skills

Enhance users’ ability to customize and manage devices such as sensors, lighting, and security systems to suit their specific needs.

Expected Result

1. **Target Audience:** Beginners, students, and IoT enthusiasts.
2. **Purpose:** Provide a comprehensive demonstration system for educational use.
3. **Focus:** Offer a user-friendly Home Assistant guide that encourages students and enthusiasts to explore IoT development.
4. **Goal**: Promote system learning and practical skill development among students.
5. **Application:** Utilize the system for hands-on learning experiences and technical growth.

Scope

1 **Installation and Initial Configuration**

- **Installing Home Assistant:** Step-by-step guide for popular platforms such as Raspberry Pi and other supported environments.
- **Initial Setup:** Configuring the system for first-time use, creating an admin account, and setting up network and remote access.
- **Configuration Files:** Understanding key Home Assistant configuration components.

Scope

2 **Utilizing the Home Assistant Dashboard**

- **Dashboard Interface:** Overview of the interface, main components, and navigation.
- **Device & UI Management:** Adding and configuring entities, customizing the Lovelace UI, and organizing cards and views for efficient control.

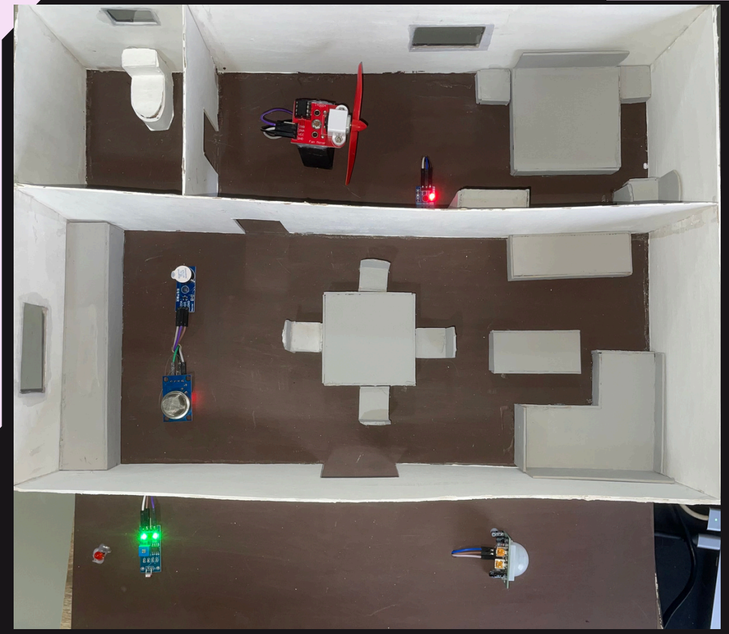
3 **Connecting and Controlling Smart Devices**

- **Device Integration:** Connecting smart lights, motion sensors, and temperature sensors.
- **Automations & Scripts:** Creating automation rules, writing scripts, and using triggers, conditions, and actions for smart control.

Scope

4 **Learning Kit and Example Project**

- **Smart Home Demonstration:** Step-by-step guide to building a sample smart home system.
- **Implementation:** Smart lighting and temperature control integrated with real-world devices.
- **Sample Layout:** One-bedroom house with a bathroom, kitchen, garage, and a vegetable & fruit garden.



Laboratory

Laboratory 1: Installing Home Assistant, ESPHome, Database On Docker

Objectives (Home Assistant on Docker)

- Learn to install Docker and Docker Compose
- Run the official Home Assistant container
- Set up persistent storage and networking
- Access Home Assistant via web browser
- Install and use ESPHome add-on
- Create and connect custom IoT devices
- Build a scalable smart home system with Docker

Experiment: Installing Home Assistant with Docker on All OS

Step 1: Install Docker

1. macOS (M1/M2/Intel)

1. Go to https://www.docker.com
2. Download .dmg ⟶ Drag to Applications ⟶ Launch Docker Desktop
3. Wait until the whale icon appears in the menu bar
4. Open Terminal ⟶ check Docker:

Bash:

```
docker --version
docker compose version
```

or

```
docker-compose version
```

2. Windows 10/11

1. Go to Docker Desktop for Windows
2. Download installer ⟶ Run ⟶ Enable WSL2 integration when prompted
3. Open PowerShell ⟶ check Docker:

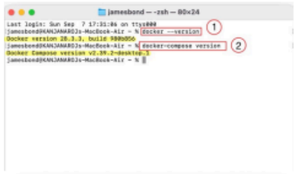
Bash:

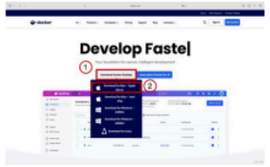
```
docker --version
docker compose version
```

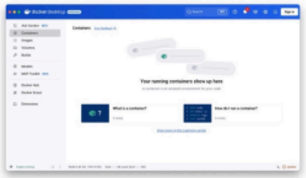
3. Linux (Ubuntu/Debian)

Bash:

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg lsb-release
curl -fsSL https://get.docker.com | sh
docker --version
docker compose version
```







Step 2: Prepare Folder Structure

- macOS / Linux:

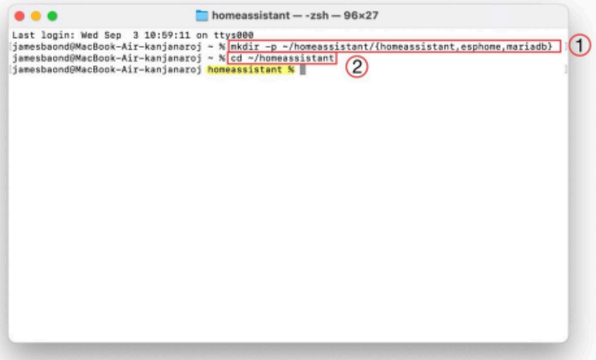
Bash:

```
mkdir -p ~/homeassistant/{homeassistant,esphome,mariadb}
cd ~/homeassistant
```

- Windows (PowerShell):

PowerShell:

```
mkdir C:\homeassistant\homeassistant
mkdir C:\homeassistant\esphome
mkdir C:\homeassistant\mariadb
cd C:\homeassistant
```



Step 3: Create docker-compose.yml

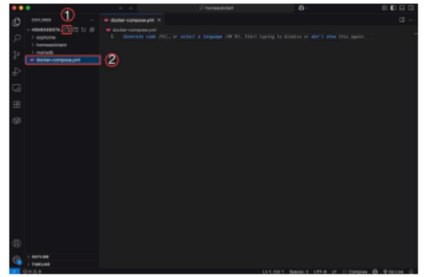
Create a file docker-compose.yml in VS code and paste the following:

Yaml:

```
version: '3.9'
services:
  homeassistant:
    container_name: homeassistant
    image: ghcr.io/home-assistant/home-assistant:stable
    volumes:
      - ./homeassistant/config:/config
      - /etc/localtime:/etc/localtime:ro
    restart: unless-stopped
    ports:
      - "8123:8123"
    depends_on:
      - mariadb

  esphome:
    container_name: esphome
    image: ghcr.io/esphome/esphome
    volumes:
      - ./esphome/config:/config
      - /etc/localtime:/etc/localtime:ro
    restart: unless-stopped
    ports:
      - "6052:6052"

  mariadb:
    container_name: mariadb
    image: mariadb:11
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword1122
      MYSQL_DATABASE: homeassistant
      MYSQL_USER: admin
      MYSQL_PASSWORD: Home1122
    volumes:
      - ./mariadb/data:/var/lib/mysql
    ports:
      - "3306:3306"

  phpmyadmin:
    container_name: phpmyadmin
    image: phpmyadmin:latest
    restart: unless-stopped
    ports:
      - "8080:80"
    environment:
      PMA_HOST: mariadb
      PMA_USER: admin
      PMA_PASSWORD: Home1122
    depends_on:
      - mariadb
```

Explanation:

- Home Assistant: Web UI ⟶ port 8123
- ESPHome: Web UI ⟶ port 6052
- MariaDB: DB ⟶ port 3306 (persistent)



Step 4: Run Docker Containers

- Open Terminal / PowerShell in folder with docker-compose.yml:

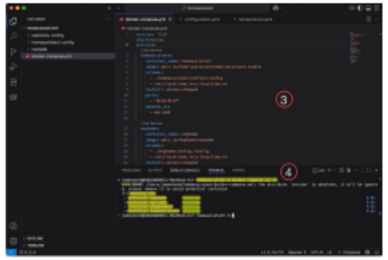
  Bash:

  ```
  docker compose up -d
  ```

- Check running containers:

  Bash:

  ```
  docker ps
  ```



Step 5: Accessing the configuration.yaml File with Docker

yaml:

```
recorder:
  db_url: mysql://admin:Home1122@mariadb/homeassistant?charset=utf8mb4
  purge_keep_days: 7   # เก็บ log 7 วัน
  commit_interval: 1   # บันทึกทุก 1 นาที
```

- Access the Files:

  Navigate to that folder on your host machine. This is where all your Home Assistant configuration files are stored, including configuration.yaml.

- Edit the Code:

  Open the **configuration.yaml** file using a code editor

  Add the code snippet you need to the file.

  Note: Pay close attention to indentation. YAML is very sensitive to proper spacing, and incorrect indentation can cause Home Assistant to fail.

  Save the file after making your changes.

- Restart Home Assistant:

  For the changes to take effect, you must restart the Home Assistant container.

  If you are using docker-compose, open your terminal in the directory containing the docker-compose.yamlfile and run the command:

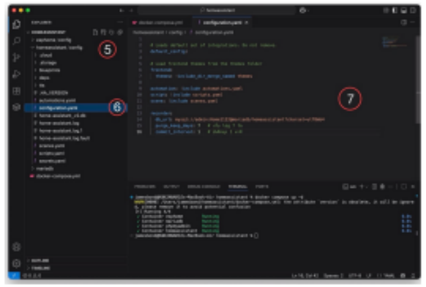
  Bash:

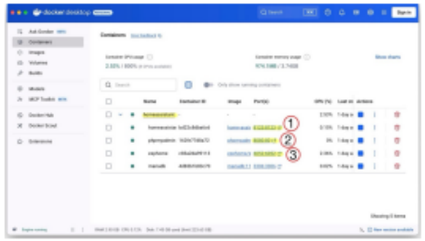
  ```
  docker-compose restart
  ```

  If you are using a **docker run** command, you'll need to stop the old container and start it again with the same parameters.

  Bash:

  ```
  docker stop homeassistant
  docker start homeassistant
  ```





Step 5: Access Home Assistant

- Open browser:

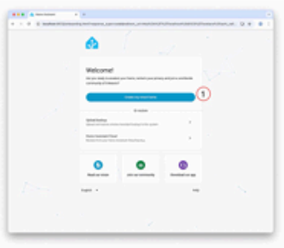
  http://localhost:8123

- Home Assistant Onboarding

  This set of images shows the initial setup process for Home Assistant after it's been installed.

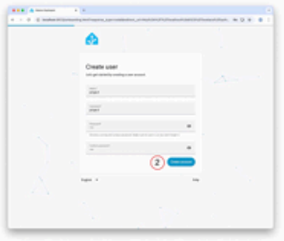
Step 1: Welcome Screen

- The first screen you'll see is a welcome page. To get started with your smart home setup, click the **"Create my smart home"** button.



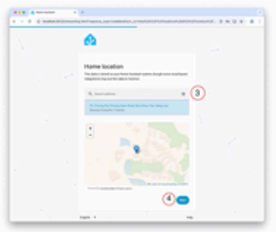
Step 2: Create User

- You need to create a user account to manage your Home Assistant instance. Fill in your desired **Name**, **Username**, and a strong **Password**.
- Click **"Create account"** to proceed.



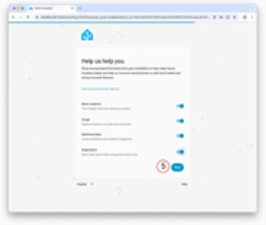
Step 3: Home Location

- Set your home's physical location. This is important for features like weather forecasting and automations that depend on sunrise and sunset times. You can either type in an address or drag the pin on the map.
- Click **"Next"**.



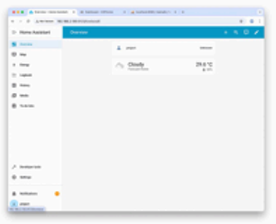
Step 4: Help Us Help You

- This screen asks for your consent to share anonymous data to help improve the Home Assistant platform. You can toggle these options on or off based on your preference.
- Click **"Next"**.



Step 5: Finish Setup

- The initial setup is complete. Click **"Finish"** to go to your Home Assistant dashboard.



Step 6: Access ESPHome

- Open browser:

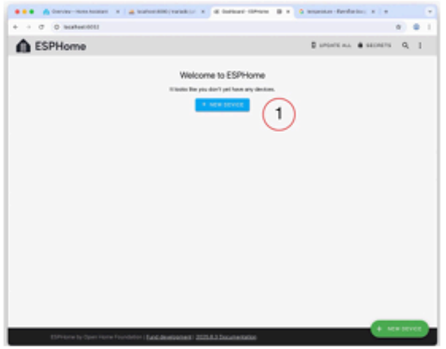
  http://localhost:6052

- **ESPHome User Guide for Beginners**

ESPHome is a tool that lets you easily set up and control your Wi-Fi devices without needing to write complex code.

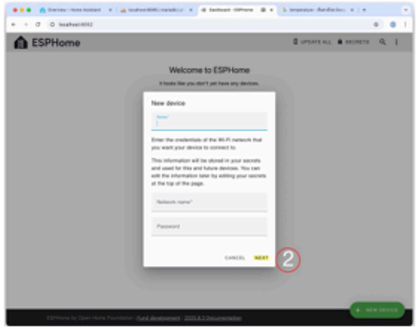
Step 1: Start the Device Setup

- **Open the Dashboard:** Open the ESPHome Dashboard on your computer.
- **Create a New Device:** At the bottom right of the screen, click the **"+ NEW DEVICE"** button to begin the process of adding a new device.



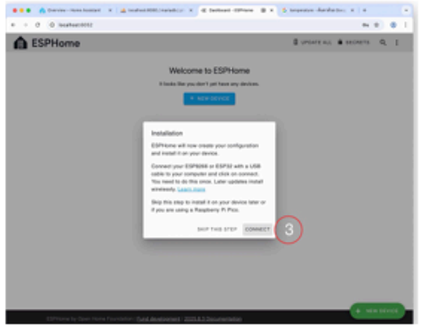
Step 2: Name Your Device and Connect to Wi-Fi

- **Name the Device:** A new window will pop up. Enter a name for your device in the **"Name"** field (this name will identify your device on the network).
- **Enter Wi-Fi Details:** Input your Wi-Fi network name in the **"Network name"** field and the password in the **"Password"** field.
- **Proceed:** Click **"NEXT"**.



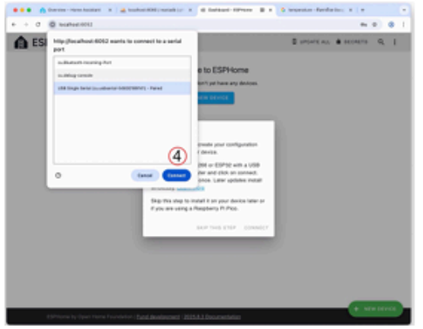
Step 3: Connect the Device to Your Computer

- **Plug in the USB:** Take your ESP8266 or ESP32 board and plug it into a USB port on your computer.
- **Initiate Connection:** Click **"CONNECT"**.



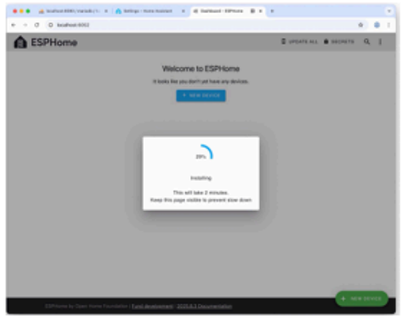
Step 4: Select the Connection Port

- **Choose the Port:** A small window will appear with a list of connection ports. Select the one that corresponds to your device (it's often labeled "USB Single Serial" or something similar).
- **Confirm Connection:** Click **"Connect"** to allow ESPHome to communicate with your device.



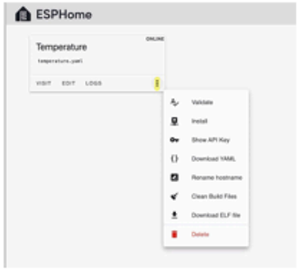
Step 5: Install the Firmware

- **Wait for Installation:** ESPHome will now begin uploading the generated firmware to your device. A progress bar will be displayed.
- **Important:** Do not unplug the USB cable or close the browser window during this process, as it could cause the installation to fail.



Step 6: Check and Manage Your Device

- **View Your Device:** Once the installation is complete, your new device will appear on the ESPHome Dashboard with an **"ONLINE"** status.
- **Access Settings:** On the device card, you'll see various options. You can click **"EDIT"** to modify the YAML configuration file, or click the three dots to access additional options like updating the firmware or deleting the device.

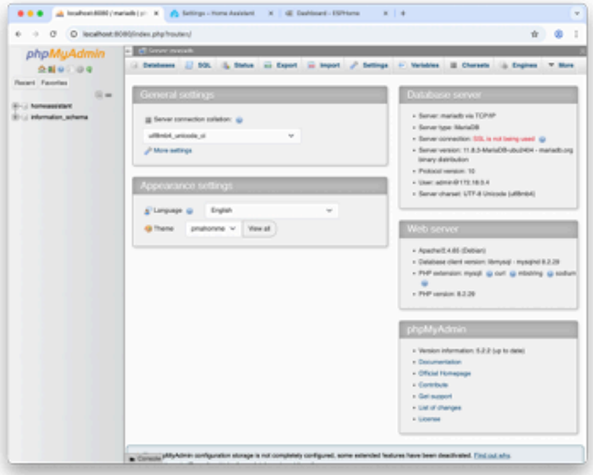


Step 7: Optional – Access phpMyAdmin

- Open browser:

  http://localhost:8080

- phpMyAdmin available on port 8080:
  - User: admin
  - Password: Home1122
  - Database: homeassistant



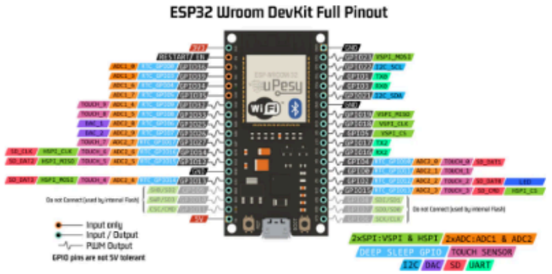
Laboratory

Laboratory 2: Integration of Sensors with ESP32

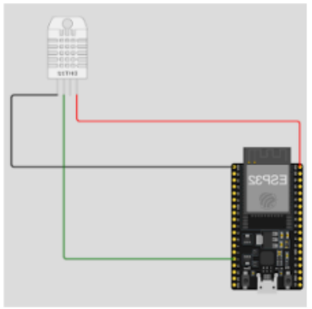
Objectives:

- Learn how to wire the DHT11 and MQ-2 sensors to the ESP32 board.
- Understand the basic setup and configuration using ESPHome.
- Test and view sensor data through Home Assistant.

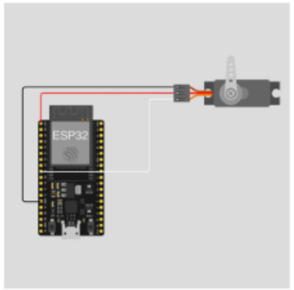
1. Wiring the Sensors to the ESP Board



- DHT11 (Temperature and Humidity Sensor)
  - VCC ⟶ 3.3V
  - GND ⟶ GND
  - DATA ⟶ GPIO22



- Fan Motor
  - VCC ⟶ 3.3V
  - GND ⟶ GND
  - DATA ⟶ GPIO25



2. Open ESPHome and Select the Device
   - Go to the ESPHome Dashboard
   - Select your device (e.g., sensor)
   - Click EDIT to modify the YAML configuration

3. Add Code for the Sensors in the YAML File

   Add the following to your configuration file:

   - DHT11 (Temperature and Humidity Sensor)

```
sensor:
  - platform: dht
    pin: 22
    model: DHT11
    temperature:
      name: "Temperature"
      id: temperature
      unit_of_measurement: "°C"
    humidity:
      name: "Humidity"
      id: humidity
      unit_of_measurement: "%"
    update_interval: 10s
```

   - Fan Motor

```
output:
  - platform: ledc
    pin: 25
    id: fan_pwm
    frequency: 25000 Hz
    inverted: true

fan:
  - platform: speed
    name: "DC Fan"
    output: fan_pwm
    speed_count: 100
    restore_mode: ALWAYS_ON
```

4. Save and Install the Firmware
   - Click SAVE
   - Click INSTALL
   - Choose Wirelessly or USB depending on your setup
   - Wait for the upload to complete and let the device reboot

5. View Sensor Data in Home Assistant
   - Go to your Home Assistant Overview Dashboard
   - You should now see sensor readings such as:

- Temperature / Humidity from DHT11
- Gas Detection Level from MQ-2

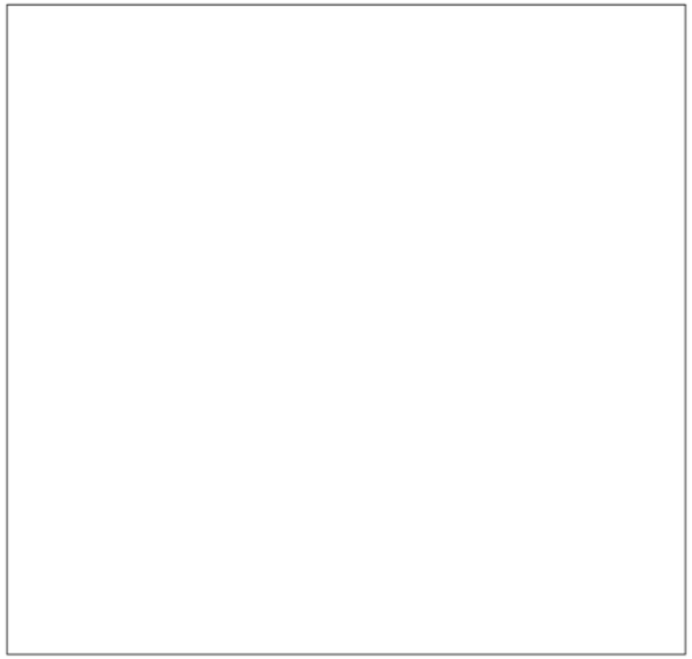
Take 1: Design and Build an Automated Fan Control System Using Temperature and Humidity Sensors.

Requirements:

1. Use an ESP32/ESP8266 board and a DHT11 sensor.
2. Connect a DC fan controlled via PWM (LEDC) to allow variable fan speed.
3. Read temperature and humidity every 10 seconds.
4. Combine the sensor reading and fan control code into a single ESPHome YAML file.
5. mplement automation so the fan operates according to the following conditions:
    - Temperature ≥ 30°C ⟶ Fan runs at 100% (full speed)
    - Temperature < 27°C ⟶ Fan turns OFF
6. Display temperature and humidity values in Home Assistant.

Home Assistant Dashboard:

Code:



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

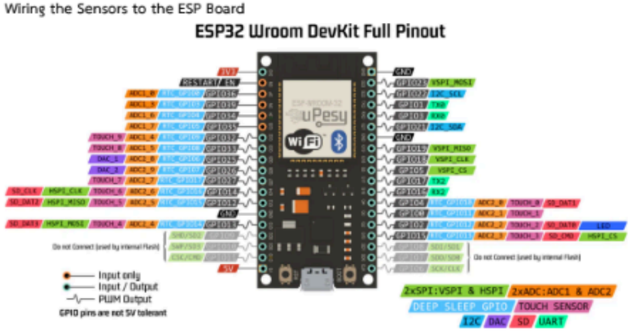
Laboratory

Laboratory 3: Integration of Sensors with ESP32

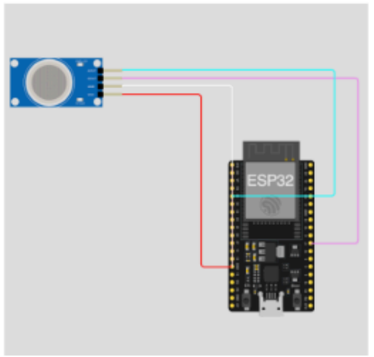
Objectives

- Connect multiple sensors (LDR, LED, PIR) to the ESP32 board.
- Write configuration in ESPHome to read sensor data and control the LED.
- Display sensor data on Home Assistant.

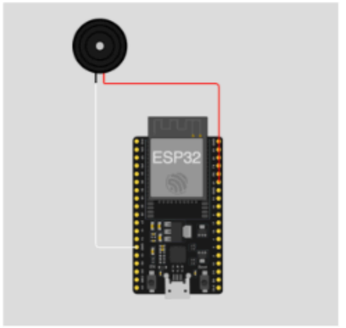
1. Wiring the Sensors to the ESP Board



- MQ-2
  - VCC ⟶ 5V
  - GND ⟶ GND
  - A0 ⟶ GPIO34 (ADC input)
  - D0 ⟶ GPIO17 (digital input)



- Active Buzzer
  - VCC ⟶ VCC
  - AOUT ⟶ GPIO21
  - GND ⟶ GND



2. Open ESPHome and Select the Device

- Go to the ESPHome Dashboard
- Select your device (e.g., sensor)
- Click EDIT to modify the YAML configuration

3. Add Code for the Sensors in the YAML File

Add the following to your configuration file:

- MQ-2

```
sensor:
  - platform: adc
    pin: GPIO34
    name: "MQ-2 Gas Sensor Analog"
```

```
    id: mq2_sensor
    update_interval: 2s
    attenuation: 12db
    filters:
      - multiply: 3.3

  binary_sensor:
    - platform: gpio
      pin: GPIO17
      name: "MQ-2 Gas Sensor Digital"
      id: mq2_digital
```

- Buzzer Module Active

```
  output:
    - platform: gpio
      pin: GPIO21
      id: buzzer_output
      inverted: true

  switch:
    - platform: output
      name: "Buzzer"
      output: buzzer_output
```

4. Save and Install the Firmware

- Click SAVE
- Click INSTALL
- Choose Wirelessly or USB depending on your setup
- Wait for the upload to complete and let the device reboot

5. View Sensor Data in Home Assistant

- Go to your Home Assistant Overview Dashboard
- You should now see sensor readings such as:
  - Gas Detection Level from MQ-2
  - Buzzer Module Active

Take 1: Design and build a gas alarm system using MQ-2 sensors and buzzers.

Requirements:

1. Use an ESP32/ESP8266 board and an MQ-2 Gas Sensor.
2. Connect an Active Buzzer to indicate gas levels.
3. Read MQ-2 Analog values every 2 seconds.
4. Combine the sensor reading and buzzer control code into a single ESPHome YAML file.
5. Implement automation so the buzzer operates according to the following conditions:
   - MQ-2 Analog ≥ 1.5V ⟶ Buzzer turns ON (alert)
   - MQ-2 Analog < 1.5V ⟶ Buzzer turns OFF
6. Display MQ-2 Analog values (and optionally Digital output) in Home Assistant.

Continued on the next page.

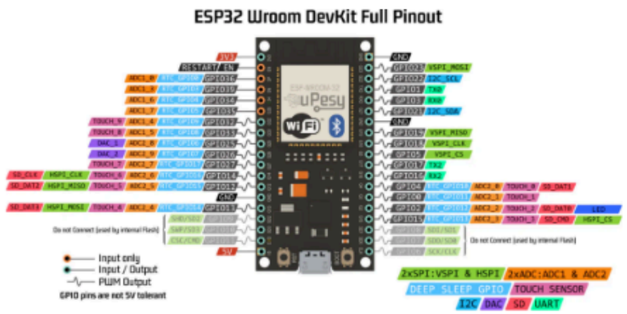
Laboratory

Laboratory 4: Integration of Sensors with ESP32

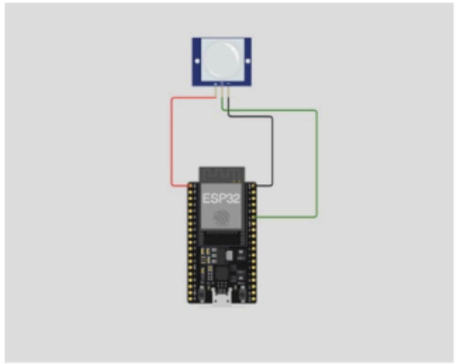
Objectives

- Connect a Servo Motor and Buzzer to the ESP32 board.
- Configure ESPHome to control these components.
- Monitor and control the devices through Home Assistant.

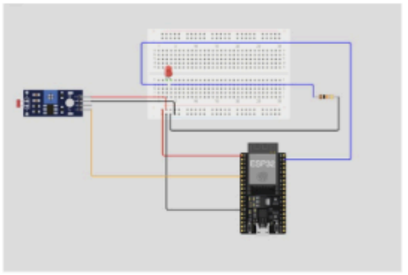
1. Wiring the Sensors to the ESP Board



- Motion Sensor Detector Module HC-SR501
  - VCC ⟶ 3V
  - GND ⟶ GND
  - OUT ⟶ GPIO14



- LDR Photosensitive Sensor Module Light Sensor
  - VCC ⟶ 3V
  - GND ⟶ GND
  - OUT ⟶ GPIO14



2. Open ESPHome and Select the Device

- Go to the ESPHome Dashboard
- Select your device (e.g., sensor)
- Click EDIT to modify the YAML configuration

3. Add Code for the Sensors in the YAML File

Add the following to your configuration file:

- Motion Sensor Detector Module HC-SR501

```yaml
# PIR Motion Sensor
binary_sensor:
  - platform: gpio
    pin:
      number: GPIO14
      mode: INPUT
    name: "Living Room Motion"
    device_class: motion
```

- LDR Photosensitive Sensor Module Light Sensor

```
# ------------------------------
# Analog LDR (AO)
# ------------------------------
sensor:
  - platform: adc
    pin: GPIO35
    name: "Living Room Light (Analog)"
    update_interval: 10s
    attenuation: 11db
    filters:
      - multiply: 3.3

# ------------------------------
# Digital LDR (DO)
# ------------------------------
binary_sensor:
  - platform: gpio
    pin: GPIO26
    name: "Bright Light Detected (Digital)"
    device_class: light
    filters:
      - delayed_on: 100ms
      - delayed_off: 100ms
```

4. Save and Install the Firmware

- Click SAVE
- Click INSTALL
- Choose Wirelessly or USB depending on your setup
- Wait for the upload to complete and let the device reboot

5. View Sensor Data in Home Assistant

- Go to your Home Assistant Overview Dashboard
- You should now see sensor readings such as:
  - Servo Motor (PWM)
  - Buzzer

Take 1: Design an LED control system using a light sensor (LDR) to automatically turn ON the LED when it is dark and OFF when it is bright.

Requirements:

1. Use an ESP32 board and an LDR sensor (Analog + Digital).
2. Connect an LED to indicate light/dark condition.
3. Read LDR Analog values every 1 second.
4. Combine the Analog/Digital sensor reading and LED control code into a single ESPHome YAML file.
5. Implement automation so the LED operates according to the following conditions:
   - LDR Analog < 2.0V ⟶ LED turns ON (dark)
   - LDR Analog ≥ 2.0V ⟶ LED turns OFF (bright)

Laboratory

Laboratory 5: Creating a Smart Home Dashboard in Home Assistant (Lovelace UI)

Objectives

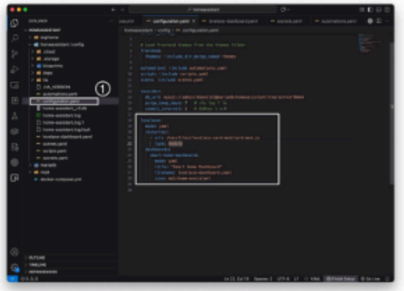
- Learn how to design a Lovelace Dashboard to visualize real-time sensor data.
- Understand how to use various Lovelace cards such as Entities, Graph, Gauge, and Button.
- Practice integrating data from sensors and actuators to build an interactive Smart Home UI.

1. **Enable YAML Mode for Lovelace Dashboard**

- Edit the file configuration.yaml in your /config/ directory and add:

Yaml :

```
lovelace:
  mode: yaml
  resources:
    - url: /hacsfiles/lovelace-card-mod/card-mod.js
      type: module
  dashboards:
    smart-home-dashboard:
      mode: yaml
      title: "Smart Home Dashboard"
      filename: lovelace-dashboard.yaml
      icon: mdi:home-assistant
```



💡 Note:

- Restart Home Assistant after editing.
- The dashboard file lovelace-dashboard.yaml will be created and stored in /config/.

- The dashboard name must contain a hyphen (-) instead of an underscore (_).

  Example: ✅ smart-home-dashboard

  ❌ smart_home_dashboard

- After saving, go to

  Settings ⟶ Developer Tools ⟶ YAML ⟶ Check Configuration and then click Restart if no errors appear.

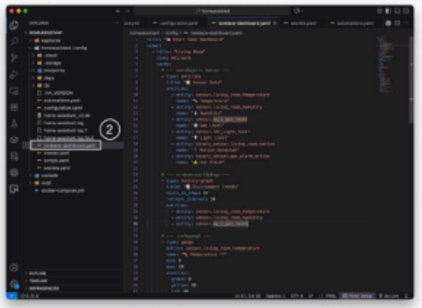
2. **Create the Dashboard File**

- Create a new file named lovelace-dashboard.yaml in the /config/ directory.
- Paste the code below:

Yaml:

```
title: "🏠 Smart Home Dashboard"
views:
  - title: "Living Room"
    icon: mdi:sofa
    cards:
      # — Sensor Data —
      - type: entities
        title: "📋 Sensor Data"
        entities:
          - entity: sensor.living_room_temperature
            name: "🌡️ Temperature"
          - entity: sensor.living_room_humidity
            name: "💧 Humidity"
          - entity: sensor.mq_2_gas_level
            name: "💨 Gas Level"
          - entity: sensor.ldr_light_level
            name: "💡 Light Level"
          - entity: binary_sensor.living_room_motion
            name: "🚶 Motion Detected"
          - entity: binary_sensor.gas_alarm_active
            name: "⚠️ Gas Alarm"

      # — Historical Graphs —
      - type: history-graph
        title: "📈 Environment Trends"
        hours_to_show: 12
        refresh_interval: 30
        entities:
          - entity: sensor.living_room_temperature
          - entity: sensor.living_room_humidity
          - entity: sensor.mq_2_gas_level

      # — Temperature Gauge —
      - type: gauge
        entity: sensor.living_room_temperature
        name: "🌡️ Temperature °C"
        min: 0
        max: 50
        severity:
          green: 0
          yellow: 30
          red: 40
```

```
# — Light Level Gauge —
- type: gauge
  entity: sensor.ldr_light_level
  name: "💡 Light Level (V)"
  min: 0
  max: 3.3
  severity:
    red: 0.5
    yellow: 1.0
    green: 2.0

# — Device Controls —
- type: grid
  title: "🔧 Device Controls"
  columns: 3
  cards:
    - type: button
      entity: fan.living_room_fan
      name: "Fan"
      icon: mdi:fan
      tap_action: { action: toggle }
    - type: button
      entity: switch.active_buzzer
      name: "Buzzer"
      icon: mdi:volume-high
      tap_action: { action: toggle }
    - type: button
      entity: switch.indicator_led
      name: "LED"
      icon: mdi:led-on
      tap_action: { action: toggle }

# — Map (for Lab 9) —
- type: map
  title: "📍 Sensor Location"
  default_zoom: 15
  entities:
    - entity: sensor.living_room_temperature
      name: "Living Room Sensor"
```



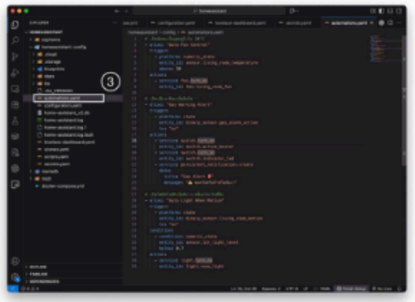
3. Add Automations for Device Control

- Edit /config/automations.yaml and add:

Yaml:

```
# Turn on the fan when temperature exceeds 30°C
- alias: "Auto Fan Control"
  trigger:
    - platform: numeric_state
      entity_id: sensor.living_room_temperature
      above: 30
  action:
    - service: fan.turn_on
      entity_id: fan.living_room_fan

# Turn on buzzer and LED when gas is detected
- alias: "Gas Warning Alert"
  trigger:
    - platform: state
      entity_id: binary_sensor.gas_alarm_active
      to: "on"
  action:
    - service: switch.turn_on
      entity_id: switch.active_buzzer
    - service: switch.turn_on
      entity_id: switch.indicator_led
    - service: persistent_notification.create
      data:
        title: "Gas Alert 🚨"
        message: "⚠️ Gas or smoke detected in the room!"

# Automatically turn on the light when motion detected in darkness
- alias: "Auto Light When Motion"
```

```
trigger:
  - platform: state
    entity_id: binary_sensor.living_room_motion
    to: "on"
condition:
  - condition: numeric_state
    entity_id: sensor.ldr_light_level
    below: 0.7
action:
  - service: light.turn_on
    entity_id: light.room_light
```



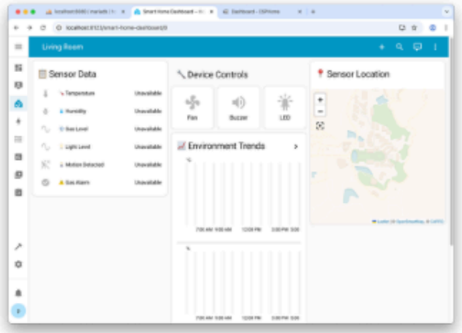
4. View the Dashboard

Open:

👉 http://localhost:8123/lovelace/smart-home-dashboard

You should see a clean, interactive dashboard containing:

- Real-time sensor readings
- Graphs and gauges
- Control buttons for fan, buzzer, and LED
- A map card (for use in Lab 9)



Take 1: Organize and Add a "Bedroom" View (Bonus Task)

Requirements

1. Add a new view named Bedroom to your existing YAML dashboard.

The view must include:

- One Entities Card showing at least: sensor.bedroom_temperature, sensor.bedroom_humidity, and binary_sensor.bedroom_motion
- At least one Gauge Card (for example, temperature)
- At least one Grid Card with two control buttons (e.g., Light and Buzzer)

2. Organize your dashboard layout for clarity and readability:

- Group cards logically into Sensor Data, Graph/Visualization, and Device Controls sections.
- Use clear titles, icons, and consistent styling for each view (Living Room & Bedroom).

3. (Optional — Bonus) Use card-mod styling or other Lovelace resources to improve spacing or hide unwanted headers.

- Make sure the resource is declared properly under lovelace: resources: in your YAML configuration.
- Restart Home Assistant after updating the configuration.

4. Submission Requirements:

- Paste your updated lovelace-dashboard.yaml containing both Living Room and Bedroom views into a Notepad file.
- Capture and submit screenshots of the Bedroom dashboard showing your entities and control buttons working.

Home Assistant Dashboard:

Continued on the next page.

Laboratory

LAB 6: Smart Notification System using Telegram and Home Assistant

Objectives

- To configure Telegram Bot with Home Assistant for real-time alert notifications.
- To create automations that trigger alerts when sensors detect abnormal conditions.
- To demonstrate a simple IoT smart home notification system.

Step 1 – Download and Install Telegram

Before creating your Telegram Bot, make sure the Telegram application is installed on your device.

- **For Mobile (Recommended)**

  1. Visit https://www.telegram.org
  2. Choose Android or iPhone / iPad.
  3. Tap Download Telegram to open the Play Store or App Store.
  4. Install the app and open it after installation.
  5. Sign in using your phone number and verification code.

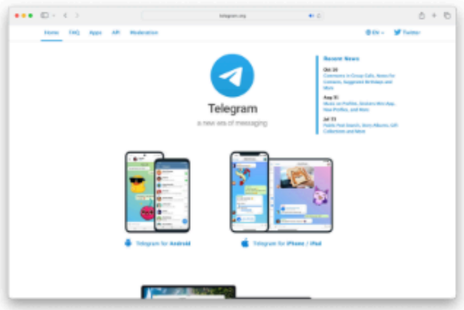
- **For Desktop (Windows / macOS / Linux)**

  1. Go to https://www.telegram.org or the main site's Get Telegram for PC / Mac section.
  2. Select your operating system (Windows, macOS, or Linux).
  3. Download the installer file and follow the on-screen instructions.
  4. After installation, log in by scanning the QR code with your mobile Telegram app or entering your phone number.

💡 **Tip**

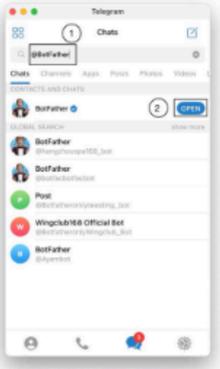
• Using the same Telegram account on both mobile and desktop lets you receive Home Assistant alerts anywhere.

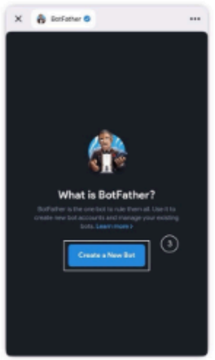
• Once Telegram is installed and logged in, you can proceed to Step 1 – Create a Telegram Bot from the Lab 6 document.

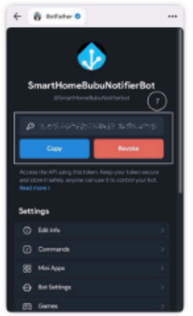


Step 2 – Create a Telegram Bot

- Open Telegram ⟶ search for @BotFather.







- 👉 Copy the **API Token** — it will be used in Home Assistant.

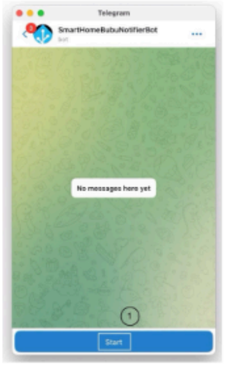
Step 3 – Start the Bot Conversation

⚠️ This step is required — otherwise Telegram will block all messages sent from Home Assistant.

- Open the link of your bot, e.g.

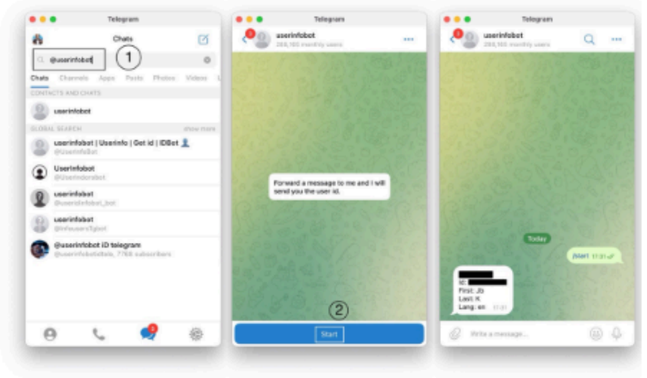
  https://t.me/SmartHomeBubuNotifierBot

- Click Start
- Once it replies "You started this bot", it's ready for use ✅



Step 4 – Get Your Chat ID

- In Telegram, search for @userinfobot.
- Click Start, and you will receive a message like:



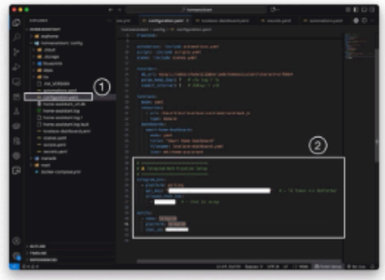
Step 5 – Configure Telegram in Home Assistant

- Edit your configuration.yaml file and add the following block at the end:

Yaml:

```
# ==============================
# 🔔 Telegram Notification Setup
# ==============================
telegram_bot:
  - platform: polling
    api_key: "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx "   # ⟵ Replace with your own Token
    allowed_chat_ids:
      - 987654321   # ⟵ Replace with your own Chat ID
```

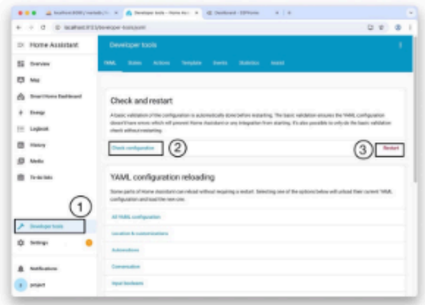
```
notify:
  - name: telegram
    platform: telegram
    chat_id: 987654321
```

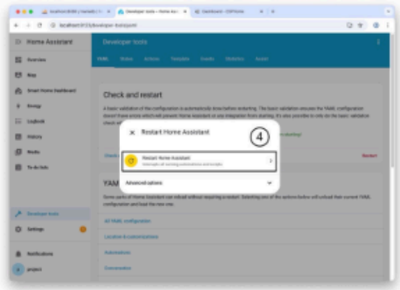


💡 **Tip:** For better security, store the token and chat ID in secrets.yaml like:

Step 6 – Check Configuration and Restart

- Go to Developer Tools ⟶ YAML ⟶ Check Configuration
- If no errors appear ⟶ Go to Settings ⟶ System ⟶ Restart



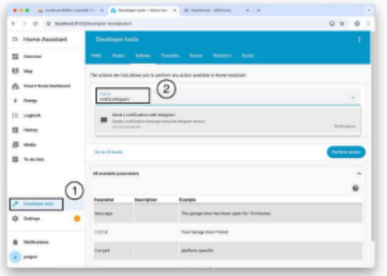


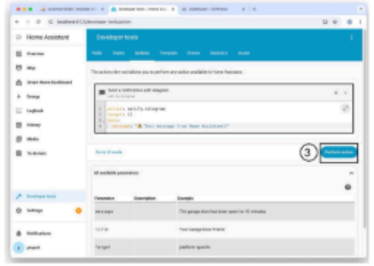
Step 7 – Test Telegram Notification

- Go to Developer Tools ⟶ Actions
- Paste this YAML snippet:

Yaml:

```
action: notify.telegram
target: {}
data:
  message: "✅ Test message from Home Assistant!"
```





- Click Perform action

  📱 If you see a message in Telegram saying

  "✅ Test message from Home Assistant!"

  then your integration works perfectly 🎉

Step 8 – Create Automation Rules

- Open the file /config/automations.yaml and add these automations:

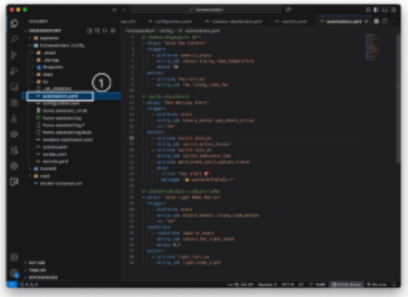
  Yaml:

```
# ============================================
# 🚨 Automation 1: Gas Detection Alert
# ============================================
- alias: "Alert: Gas Detected"
  trigger:
    - platform: state
      entity_id: binary_sensor.gas_alarm_active
      to: "on"
    - platform: numeric_state
      entity_id: sensor.living_room_temperature
      above: 32
  action:
    - service: notify.telegram
      data:
        message: >
          🔥 High temperature alert!
          Current temperature: {{ states('sensor.living_room_temperature') }} °C
```

- After saving the file, reload automations:

  Yaml:

```
action: automation.reload
target: {}
data: {}
```



Take 1: Combined Smart Alert (Bonus)

Requirements

1. Create one automation that sends a detailed summary report to Telegram. whenever any of the following occurs:
   - Gas alarm ON
   - Motion detected
   - Temperature > 32 °C
2. The message must include:
   - Current values of all sensors
   - Timestamp ({{ now().strftime('%H:%M:%S') }})
   - Room location
3. Use multiline message format (>)
4. Test the automation by simulating one condition

Laboratory

Laboratory 7: Location-Based Smart Notification using Telegram + Home Assistant

Objectives

- Enable Home Assistant to send Telegram alerts with embedded Google Maps location links.
- Practice using templated automation messages in Home Assistant.
- Allow users to know where an event occurred - e.g., Kitchen, Living Room, Bedroom.
- Demonstrate a location-aware Smart Home alert system.

System Concept

When an event occurs (e.g., gas leak, motion, or high temperature), Home Assistant sends a Telegram message with a Google Maps link showing the sensor's location.

Example Telegram message:

⚠️ Gas leak detected in the kitchen.

📍 See location on map.

1. Define Sensor Coordinates

- Assign fixed GPS coordinates for each sensor location:

Location	Latitude	Longitude
Living Room	13.9123	100.5211
Kitchen	13.9125	100.5213
Bedroom	13.9126	100.5215

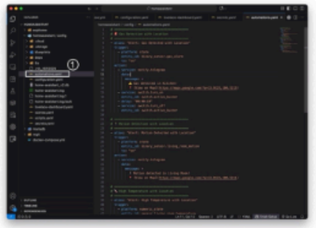
💡 **Tip:** To find coordinates, open Google Maps, right-click on a location ⟶ "What's here?" ⟶ copy latitude and longitude.

2. Add Automation Rules in Home Assistant

- Edit /config/automations.yaml and add the following:

```
    to: "on"
  action:
    - service: notify.telegram
      data:
        message: >
          🚶 Motion detected in Living Room!
          📍 [View on Map](https://maps.google.com/?q=13.9123,100.5211)

# ========================================
# 🌡️ High Temperature with Location
# ========================================
- alias: "Alert: High Temperature with Location"
  trigger:
    - platform: numeric_state
      entity_id: sensor.living_room_temperature
      above: 32
  action:
    - service: notify.telegram
      data:
        message: >
          🔥 High temperature detected!
          Current: {{ states('sensor.living_room_temperature') }} °C
          📍 [View on Map](https://maps.google.com/?q=13.9126,100.5215)
```


```
    to: "on"
  action:
    - service: notify.telegram
      data:
        message: >
          🚶 Motion detected in Living Room!
          📍 [View on Map](https://maps.google.com/?q=13.9123,100.5211)

# ========================================
# 🌡️ High Temperature with Location
# ========================================
- alias: "Alert: High Temperature with Location"
  trigger:
    - platform: numeric_state
      entity_id: sensor.living_room_temperature
      above: 32
  action:
    - service: notify.telegram
      data:
        message: >
          🔥 High temperature detected!
          Current: {{ states('sensor.living_room_temperature') }} °C
          📍 [View on Map](https://maps.google.com/?q=13.9126,100.5215)
```



3. Validate and Reload Automations

- Go to Developer Tools ⟶ YAML ⟶ Check Configuration to ensure no syntax errors.
- Then go to Developer Tools ⟶ Actions and execute:

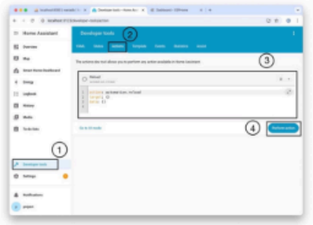
Yaml:

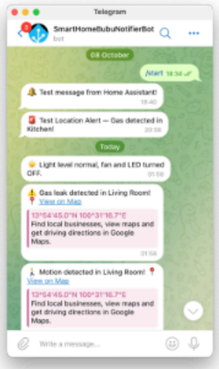
action: automation.reload

target: {}

data: {}

- Once reloaded, your new automations are active ✅





Take 1: Design and Implement a Smart Unified Notification System for Multiple Events

Objective:

To design and implement a location-aware smart notification system in Home Assistant that can detect multiple simultaneous events (Gas Leak, Motion, and High Temperature) and send a single summarized alert to Telegram — instead of multiple separate messages.

Requirements:

1. Hardware and Platform
    - Use an ESP32 or ESP8266 board integrated with ESPHome.
    - Use the following sensors:
    - MQ-2 gas sensor ⟶ detects gas or smoke.
    - HC-SR501 motion sensor ⟶ detects movement.
    - DHT11 sensor ⟶ measures temperature and humidity.
    - All sensors must be connected and registered in Home Assistant.
2. Automation and Notification
    - Create a single automation in Home Assistant that combines all three triggers:
    - Gas sensor (binary_sensor.gas_alarm)
    - Motion sensor (binary_sensor.living_room_motion)
    - Temperature sensor (sensor.living_room_temperature)
    - The automation should:
    - Wait for 5 seconds to ensure all events occurring close in time are grouped.
    - Collect current values of all sensors.
    - Send one unified message to Telegram that summarizes all detected events.
    - Include the current time and Google Maps location link of the affected area.

Code:

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

Laboratory

Laboratory 8: Smart Alert Logging with MariaDB and History Dashboard

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:

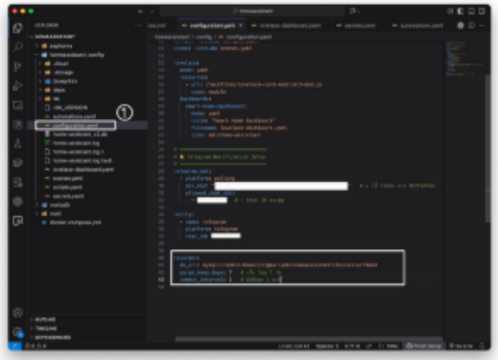
Yaml:

```
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 🎉



Yaml:

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

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

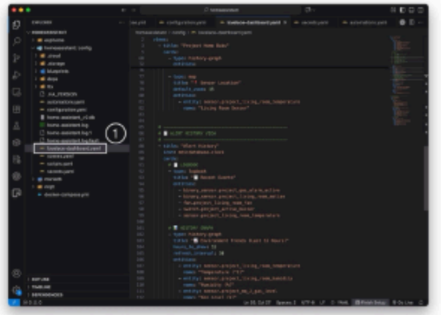
This ensures all device state changes and events are stored and displayed in the "History" and "Logbook" sections.

- 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"
```



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:

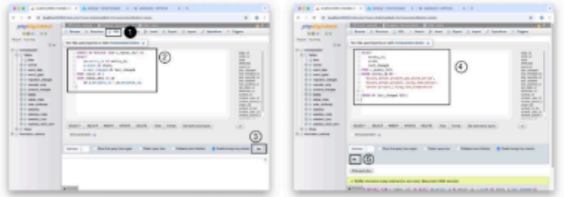
SQL:

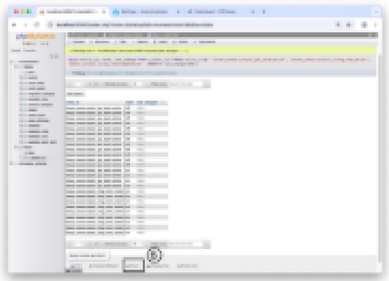
```
SELECT
  entity_id,
  state,
  last_changed
FROM v_states_full
WHERE entity_id IN (
  'binary_sensor.project_gas_alarm_active',
  'binary_sensor.project_living_room_motion',
  'sensor.project_living_room_temperature'
)
ORDER BY last_changed DESC;
```

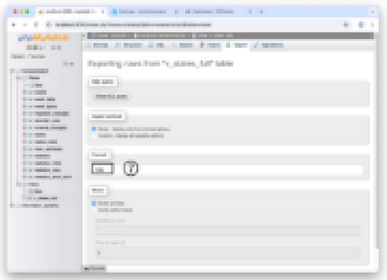
Step 4: Export to CSV

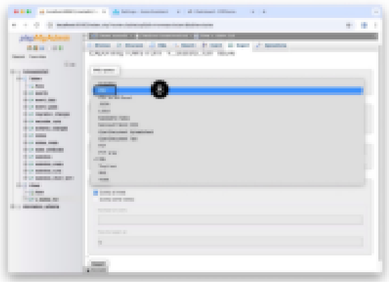
- Click Export at the top of phpMyAdmin
- Choose CSV as the format
- Set a file name (e.g. sensor_log_report.csv)
- Click Go ✅

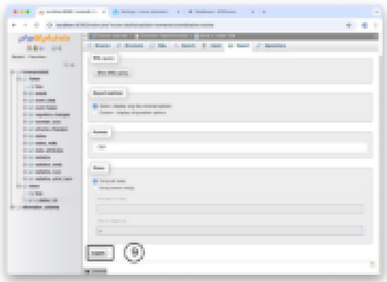
You'll get a downloadable CSV file that can be opened in Excel or Google Sheets.











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 ±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).

Laboratory

Laboratory 9: GPS Integration (ESPHome To Home Assistant Map)

Objectives

- To integrate GPS (latitude and longitude) into smart sensors for location tracking.
- To display sensor positions on Home Assistant’s interactive map.
- To understand how location data can be used for monitoring and automation in IoT systems.

Hardware and Software Requirements

Component	Function
ESP32 DevKit	Main controller board
DHT11 Sensor	Measures temperature and humidity
MQ-2 Sensor	Detects gas concentration
LDR Sensor	Detects light intensity
PIR Sensor	Detects human motion
Fan (Relay-controlled)	Air circulation control
LED	Indicates brightness or darkness

Component	Function
Active Buzzer	Emits sound alarm on gas detection
Home Assistant + ESPHome	Visualization and automation control

Step 1 — Setup and Upload the ESPHome Code

- Use the following ESPHome YAML configuration (no modification needed):

Yaml:

```
sensor:
  # DHT11
  - platform: dht
    pin: GPIO4
    model: DHT11
    temperature:
      name: "Living Room Temperature"
      id: temp_living
      accuracy_decimals: 1
      latitude: 19.9057
      longitude: 99.8355
    humidity:
      name: "Living Room Humidity"
```

id: hum_living

accuracy_decimals: 1

latitude: 19.9057

longitude: 99.8355

update_interval: 10s

This configuration enables:

- Temperature and humidity monitoring via DHT11
- Gas detection via MQ-2 sensor
- Light intensity monitoring with LDR
- Motion detection using PIR
- Automatic fan operation when temperature exceeds 30°C
- LED control depending on light intensity
- Gas alarm triggering the buzzer above 0.40V
- Location tracking using predefined GPS coordinates

Step 2 — Adding GPS Coordinates for Each Sensor

- Each sensor and binary sensor includes latitude and longitude values.

  Example:

  Yaml:

  temperature:

  name: "Living Room Temperature"

  id: temp_living

  latitude: 19.9057

  longitude: 99.8355

- Repeat this for all sensors (DHT11, MQ-2, LDR, PIR) to allow Home Assistant to plot them on a map.

Step 3 — Upload to ESP32

- Open ESPHome inside Home Assistant.
- Click INSTALL ⟶ WIRELESS (or via USB).
- Wait for build and upload to complete.
- When successful, ESPHome will show:

  ✅ Successfully compiled program.

✅ INFO Uploading /config/.esphome/build/project/firmware.bin

- The device should appear Online on the ESPHome dashboard.

Step 4 — Add Map View in Home Assistant

- Add the following card in your **Lovelace Dashboard** (edit ⟶ Add card ⟶ YAML):

  Yaml:

  - type: map

  title: " 📍 Smart Room Sensor Locations"

  default_zoom: 15

  entities:

  - entity: sensor.project_living_room_temperature

  name: "Temperature"

  - entity: sensor.project_mq_2_gas_level

  name: "Gas Sensor"

  - entity: sensor.project_ldr_light_level

  name: "Light Sensor"

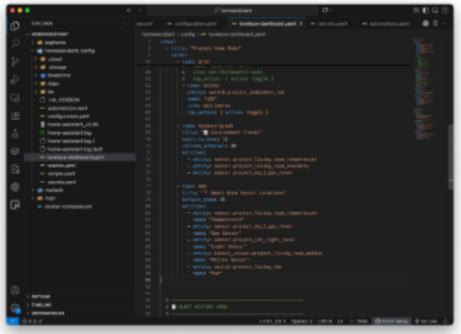
  - entity: binary_sensor.project_living_room_motion

  name: "Motion Sensor"

- entity: switch.project_living_fan

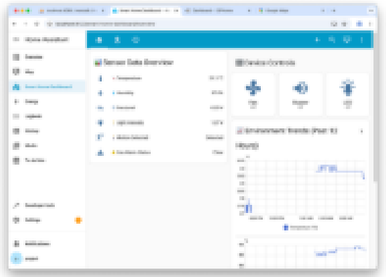
name: "Fan"

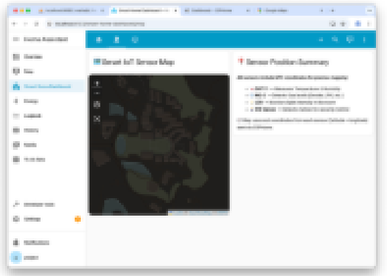
This displays pins on the Home Assistant map corresponding to each sensor's coordinates.

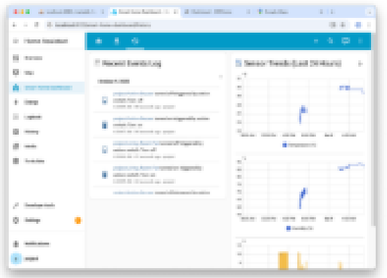


Step 5 — Observation

- The map shows all devices' GPS positions.
- Clicking a pin displays live sensor values.
- Automatic control works as follows:
  - Temperature > 30°C ⟶ Fan turns ON
  - Gas level > 0.40V ⟶ Buzzer turns ON
  - Light < 1.4V ⟶ LED turns ON
  - PIR detects motion ⟶ "Motion Detected" status changes to ON







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

THANK YOU!