# [2025-HS1-SWE30011-IoT Programming-H1](https://swinburne.instructure.com/courses/66782)

# Individual Assignment (Practical)

# VO DANG KHOA NGUYEN

# 105306088

Contents

[2025-HS1-SWE30011-IoT Programming-H1 1](#_Toc196573237)

[1. Project Summary 1](#_Toc196573238)

[2. Conceptual design: 2](#_Toc196573239)

[2.1. Block diagrams: 2](#_Toc196573240)

[2.2. Use case diagram: 6](#_Toc196573241)

[2.3. Flow chart: 8](#_Toc196573242)

[3. Implementation 11](#_Toc196573243)

[3.1. Sensors: 11](#_Toc196573244)

[3.2. Actuators: 12](#_Toc196573245)

[3.3 Software & Libraries 13](#_Toc196573246)

[4. Resources Used 14](#_Toc196573247)

[5. Appendix 14](#_Toc196573248)

## 1. Project Summary

The Temperature-Controlled Fan Syste**m** is a simple, yet effective IoT application designed to monitor environmental conditions and automatically respond to changes in temperature and light. It integrates basic hardware components (sensors and actuators) with an edge device (Raspberry Pi) to demonstrate real-time automation, data storage, and user interaction.

This system uses an Arduino Uno as the microcontroller platform to collect data from a DHT11 digital temperature/humidity sensor and an LDR analog light sensor. Based on the readings, the Arduino activates a DC fan if the room temperature exceeds a threshold and turns on an LED when ambient light drops below a certain level. All data is transmitted via serial communication to the Raspberry Pi, where it is stored in a MariaDB database and displayed via a Flask-based web interface.

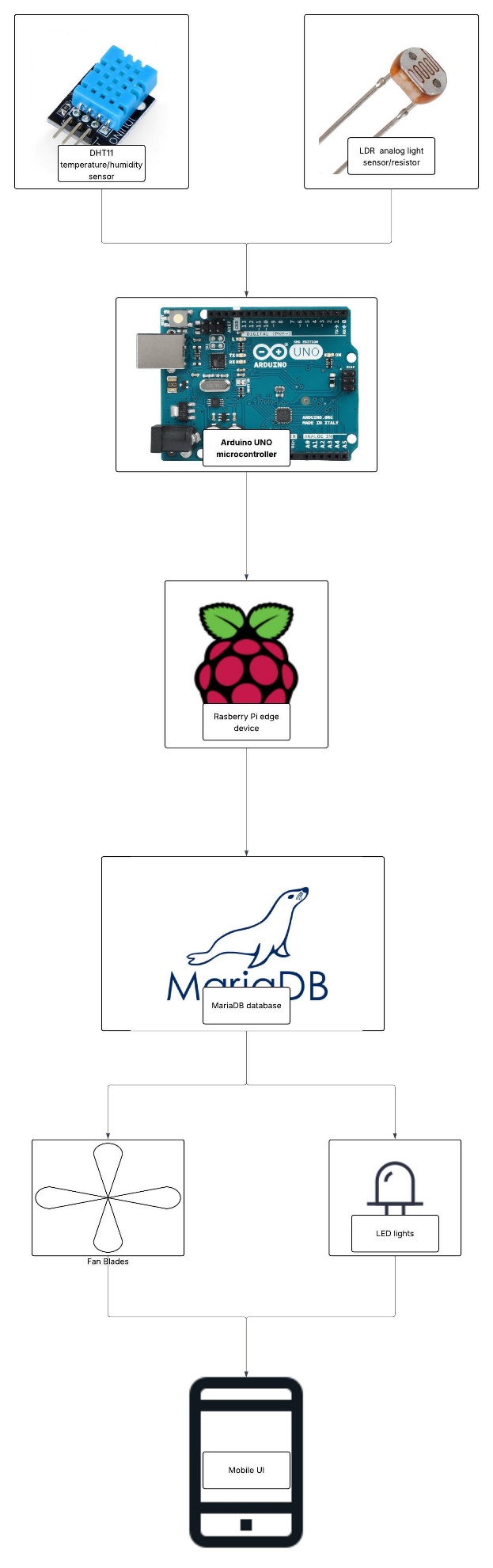
The system allows users to:

* View real-time sensor data (temperature, humidity, light)
* Adjust temperature/light thresholds via the UI
* Monitor actuator status
* Perform simple analytics (e.g., average temperature)

This application provides a foundational demonstration of how edge computing, physical IoT devices, and user interfaces can come together to automate tasks in home or office environments.

2. Conceptual design:

2.1. Block diagrams:



The block diagram illustrates the architecture of the IOT project, demonstrating the flow of data and control between the hardware and software components:

1. Sensors

* DHT11 Temperature and Humidity Sensor:  
  Measures the environmental temperature (°C) and humidity (%). This data is crucial for determining whether the room requires cooling or not.
* LDR (Light Dependent Resistor) Analog Sensor:  
  Measures the ambient light intensity. It detects when the surroundings are dark and triggers the LED lights accordingly.

Both sensors are connected directly to the Arduino Uno for data collection.

2. Arduino UNO Microcontroller

The Arduino acts as the physical layer device.

* It reads collected sensor data from the DHT11 and LDR sensors.
* It sends the collected sensor data through Serial Communication (USB) to the Raspberry Pi virtual edge device.
* It also receives control signals (e.g., turn fan on/off, LED on/off) from the Raspberry Pi and activates the appropriate actuators.

3. Raspberry Pi Edge Device

The Raspberry Pi acts as the edge computing server.

* It reads incoming serial data from the Arduino microcontroller.
* It stores all sensor data into a MariaDB database.
* It performs edge analytics by checking conditions (temperature above a threshold, light level below a threshold).
* It sends actuation commands back to the Arduino based on rule evaluations.

4. MariaDB Database

The MariaDB database on the Raspberry Pi:

* Stores past sensor data (temperature, humidity, light intensity).
* Allows real-time and historical analysis.
* Supports the web interface by providing data to be visualized or analyzed.

5. Actuators

* Fan Blades:  
  Activated when the measured temperature exceeds the predefined threshold, providing cooling to the environment.
* LED Lights:  
  Activated when the light intensity drops below a set threshold, providing visibility to the environment.

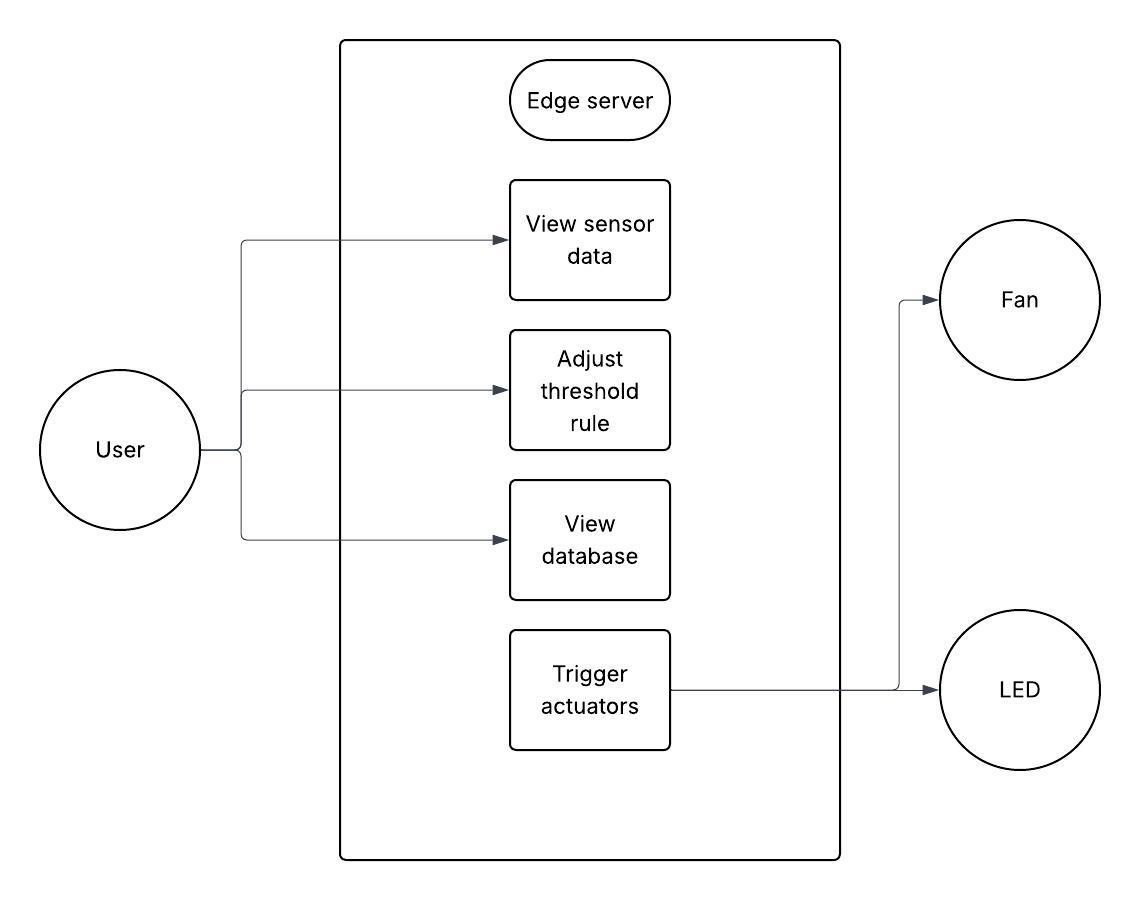
Arduino controls the actuators based on commands received from the Raspberry Pi.

6. Mobile User Interface (UI)

The Flask-based Mobile UI (accessible from smartphones or laptops):

* Displays real-time sensor readings.
* Shows actuator status (fan/LED on or off).
* Allow users to update system thresholds (e.g., change the fan activation temperature or light threshold).
* Provides basic analysis like average temperature or maximum humidity.

### 2.2. Use case diagram:



1. User

* The User is the main actor who interacts with the system via a web-based User Interface (UI) hosted on the Edge Server (Raspberry Pi).
* The User can:
  + View sensor data: Monitor real-time environmental conditions such as temperature, humidity, and light intensity.
  + Adjust threshold rule: Modify system rules, such as the temperature limit for fan activation or the light intensity limit for LED activation.
  + View database: Access historical sensor data stored in the MariaDB database.

2. Edge Server

* The Edge Server acts as the core of the IoT system, handling both data and automation rules.
* It provides services to the user and manages communication as well as calculation between the physical sensors and actuators.
* The system functions handled by the Edge Server include:
  + View sensor data: Receives sensor readings from Arduino and displays them on the user interface.
  + Adjust threshold rule: Updates internal logic thresholds based on user input.
  + View database: Queries and presents historical data to the user.
  + Trigger actuators: Based on sensor readings and thresholds, sends control signals to activate the Fan and LED.

3. Fan and LED (Actuators)

* Fan:
  + Activated by the Edge Server when the room temperature exceeds a user-defined threshold.
* LED:
  + Activated by the Edge Server when ambient light falls below a user-defined threshold.
* The actuators do not communicate directly with the user; they are controlled automatically based on logic inside the Edge Server.

### 2.3. Flow chart: A diagram of a company's flowchart AI-generated content may be incorrect.

1. Arduino collects sensor data

* The Arduino reads temperature and humidity values from the DHT11 sensor and light intensity from the LDR sensor.

2. Send sensor data to Raspberry Pi

* The collected sensor readings are transmitted via USB to the virtual device Raspberry Pi.

3. Store sensor data in MariaDB database

* The Raspberry Pi receives the data and immediately stores it in a MariaDB database for record-keeping and future analysis.

4. Run edge analytics

* The edge server (Raspberry Pi) runs analytics on the newly received data.
* It checks if the temperature or light intensity exceeds or falls below the set thresholds.

5. Conditional checks:

a) Temperature check

* If the temperature reading is higher than the threshold:
  + The system sends a "FAN\_ON" command to the Arduino.
* If the temperature reading is lower than the threshold:
  + The system sends a "FAN\_OFF" command to the Arduino.

b) Light intensity check

* If the light reading is lower than the threshold:
  + The system sends a "LED\_ON" command to the Arduino.
* If the light reading is higher than the threshold:
  + The system sends a "LED\_OFF" command to the Arduino.

6. Arduino activates actuators

* The Arduino receives the commands and controls the actuators:
  + Activates or deactivates the fan based on temperature.
  + Activates or deactivates the LED light based on light levels.

7. User interacts with the UI

* The user accesses the web interface to:
  + View real-time sensor readings.
  + Monitor the status of actuators.
  + Analyze stored data.
  + Adjust temperature or light threshold

8. Threshold update

* Through the UI, the user can update the temperature or light thresholds.
* The updated thresholds are applied in the edge analytics for future decision-making.

## 3. Implementation

### 3.1. Sensors:

1. DHT11 (Digital Sensor)

* What are you measuring?  
  Measures temperature (°C) and humidity (%) in the environment.
* Why measure it?  
  Monitoring temperature is crucial to determine whether to activate the fan for cooling. High humidity can also affect user comfort and system performance.
* How will you integrate it?
  + Connect the DHT11 sensor's output pin to a digital input pin on the Arduino (e.g., D2).
  + Use the DHT.h library to read sensor values periodically.
  + Arduino sends the temperature and humidity readings through USB to the virtual device Raspberry Pi.
  + The Raspberry Pi stores the data in MariaDB and applies automation rules (e.g. turning on the fan if the temperature exceeds a threshold, turning off the fan if the temperature falls behind a threshold ).

2. Light Dependent Resistor (LDR) (Analog Sensor)

* What are you measuring?  
  Measures ambient light intensity.
* Why measure it?  
  Detects if the room becomes dark — if so, a LED light will be automatically activated for illumination.
* How will you integrate it?
  + Set up the LDR in a voltage divider circuit with a resistor.
  + Connect the output voltage to an analog input pin on Arduino (e.g., A0).
  + Arduino reads the analog light value and sends it via USB to the virtual device Raspberry Pi.
  + Raspberry Pi applies a light threshold condition to decide if the LED should be turned on or off.

### 3.2. Actuators:

1. DC Fan

* What does it do?  
  Cools down the environment when the measured temperature exceeds a user-defined threshold.
* How will you integrate it?
  + Connect the fan to Arduino output pin through a NPN transistor (e.g., 2N2222) or relay module (because the Arduino can't drive a fan directly).
  + When Arduino receives a "FAN\_ON" command from the Raspberry Pi over Serial, it drives the pin HIGH, activating the fan.
  + When the "FAN\_OFF" command is received, Arduino drives the pin LOW, turning the fan off.

2. LED Light

* What does it do?  
  Lights up when ambient light intensity falls below a specified threshold (detected by the LDR), providing visibility or system status indication.
* How will you integrate it?
  + Connect the LED to a digital output pin on Arduino through a current-limiting resistor (~220Ω).
  + Arduino receives "LED\_ON" or "LED\_OFF" command from Raspberry Pi based on light conditions.
  + Arduino toggles the LED accordingly.

### 3.3 Software & Libraries

* Arduino Code:
  + DHT.h: For reading DHT11 sensor values.
  + Serial: For sending sensor data and receiving control commands.
* Python on Raspberry Pi:
  + pyserial: For reading data from Arduino over USB serial.
  + mysql-connector-python: To insert sensor data into MariaDB.
  + Flask: To serve the web interface.
  + threading: To manage simultaneous reading and web serving.
* Database (MariaDB):
  + Tables store:
    - timestamp
    - temperature
    - humidity
    - light\_intensity
* Web Interface (Flask App):
  + Displays live sensor values.
  + Allows adjusting thresholds (temperature, light).
  + Displays analytics (mean temperature, max humidity).

## 4. Resources Used

| Resource | Description |
| --- | --- |
| Arduino Documentation | Arduino tutorials and sample sketches |
| DHT11 Datasheet | Wiring and reading values |
| Flask Documentation | Building simple web apps |
| MariaDB Official Guide | Installing and connecting MariaDB |
| PySerial Guide | Python serial communication |
| Circuit Tutorials | Wiring fan and LED control via transistor |

## 5. Appendix

* Arduino Sketch: Full .ino file for sensor reading, serial sending, command receiving.
* Python Script:
  + Serial listener
  + Database insertion
  + Condition checking and actuation control
* Flask Application Code:
  + UI display
  + Threshold updating feature
* SQL Script: MariaDB database and table creation