CS 458 IoT Final Project
Spring 2024

Instructions: Read over all parts before starting. Attempt all. Work as a team of two.
Due: Next Monday 8th April, 2024 (or sooner)
This evaluates your ability to create smart objects, deal with communication and application protocols, as well as analytics of data collected.
[80 points]

Create a two smart devices that monitor at least temperature, humidity, and light intensity in two different locations. The two devices should run the same code. One should be in doors and the other outdoors. They will be used to collect data concurrently.
Temperature information should be collected every 6 seconds. Humidity information every 3 seconds. The light intensity should be read every 6 seconds. If you have any extra sensors, choose the intervals at which you will read the values. Readings should be displayed on a local LCD screen as they are obtained. Records should be saved on the flash memory once every minute. This will mean that whenever the smart object is powered or at any point in time, it should be possible to view some history for the previous minute at least.

The sensor data collected will be posted by two application protocols. Method one is by a direct HTTP post to a database. The second method is by publishing to an MQTT broker. Develop a separate script to fetch the published readings into the same database. The two methods will NOT be used concurrently, but a user must select which method is used without uploading new code.

The smart object should have a heartbeat signal/LED that blink-blinks every 2 seconds.

Design a suitable database (It may be relational or NoSQL, but should not be a single table if relational). It should at the minimum hold information on the temperature, humidity and light intensity (and any other sensor data) as well as a timestamp (e.g. at time of entry into database is acceptable). It should be a possible to determine which data was submitted by a particular smart node and it should be possible to add any number of additional smart nodes without changing the database design or modifying the smart object code. The name and location of the smart node should also be kept.

On a web server:

- Provide web pages for browsing/viewing historical data of the readings
- Provide an interface for adding additional smart nodes. Be sure to map the "identifier" or "ID" of the smart object to a more descriptive name in the db.
- Provide a means to view data submitted by any one of the smart objects.

The smart object also controls a mains (AC) powered electric fan. The fan should ran whenever the temperature exceeds a set value. Provide a means to update the code on the smart object over the air. It will be tested by changing the heartbeat rate to another value.

A user can use a phone or PC to connect to the smart object directly. Its IP address should be visible on the LCD screen. The user should be presented with a web portal (this is not the same as the one on the web server.) Allow the user to interact with the smart object and configure it using its “settings” page.

The smart object presents three or more web pages.

- The first page enables the following:
    - Links to the other pages
    - The page also shows the most current sensor values on the page (They are updated every 5 seconds, preferably by AJAX)
    - Button to Start AC fan manually
    - Button to Stop AC fan manually
    - Button to show the last 25 recorded LDR sensor data (displayed on this same page using AJAX, retrieved from DB).
- The second page shows the saved temperature & humidity data (retrieved from the local store/flash memory)
- The third page is a configuration page.
    - You can change the identification of the smart object by assigning a unique ID/Name This name or ID will help identify it in the db, but additionally, you will have a more descriptive name in the db.
    - A means to switch from using http POSTing to using MQTT publishing.
    - Manual override switch to allow manual control of AC fan
    - Set trigger temperature for auto starting fan. (e.g. you can use a textbox or other input to specify a trigger temperature.
- Hint: you can save configuration information in the flash.

This project will be graded by demonstration.

Using your smart objects, collect data for at least **one hour** at some time between 7 and 10am, also again for at least one hour between 4 and 6:30 pm. You should record data when one smart object is indoors and the other is outdoors. You may however choose other times if you prefer. The suggested times give a reasonable variation.
Using data assembled in your database, the objective is to predict temperature values given location, LDR value, humidity, time of day and any additional data point of your choice.

Using pandas or scikit-learn, create a suitable model with a subset of your data. Validate the model using another subset of your data.

- Determine the Mean Absolute Error (MAE) of the model.
- Predict what the temperature is expected to be, given the 100th entry in your database. Your work will however be graded by choosing some random values as a test.

Hint: it is impossible to do the analytics with text fields. You will need to replace text with numbers.
Submit

- Your data – (eg dump of your data,)
- your jupyiter notebook script(s)
- your **MAE**, and **predicted temperature** should be printed clearly in your jupyter notebook

Extra credit: You may earn up to 10% extra (8 points) by doing additional work e.g.

- make use of RTOS, have a sleep mode, [2 points each]
- have a smart object start, not knowing what network to connect and allow a user to configure it to connect to some access point, and use that access point subsequently [4 points] etc.

Hints:
You will need to think through how your MQTT client script can read data from two or more different devices posting under different but related topics. Remember wild cards, and hierarchical topics?

Grading Criteria [30]
Grading:
Concurrent posting by 2 or more devices, same code [2]
Use of time/timed events: [4]

- Intervals of data collection: 3sec (humidity), 6 sec(temp & LDR)
- Heart beat 2sec intervals

Smart object: [18]

- LCD display
- Flash data at 1min. can retrieve on page 2
- Manual ON/OFF
- Auto turn ON/OFF
- Can connect to smart object over wifi
- AJAX page refresh (5sec)
- 25 Historical LDR by AJAX from remote db
- Page 2- Temp & Humidity from flash
- OTA support

Settings: [10]

- Device ID
- Post-MQTT setting
- Manual -Auto setting
- Trigger temp setting
- Flash Saved configuration

Communication [8]

- Http post into db

- MQTT
  - Publish for more than one sensor value.
  - Script subscribes and posts to DB
  - Script subscribes to and handles any number of smart objects correctly.

Database & Web server [8]

- Db design (**relational**) + tables, fields needed, timestamp, name+loc data etc
- Browse data
- Add smart object
- Query/report

Code review (ESP32 sketch, MQTT client, backend etc] [6]
Overall evaluation (integration, slickness usability, and extra credit features) [10]
[66 points]

Analytics: [14]
raw data will be inspected
[2] retrieved data in suitable form
[5] data exploration & cleaning
[5] suitable model creation & validation for temperature.
[2] MAE, predicted values given.

<u>**Submission:**</u>

- Add all the code + picture of your set up into one pdf. At the start of the document, indicate who the team members area. Add a couple of sentences describing your work.
  - You may additionally/optionally zip of all Code (ESP32 sketch, MQTT subscriber script, Web backend, dump of database and Jupyter notebook)
  - Picture of set up (smart objects) needed. Schematic is optional