Final Project Report – Group D606

Rafael Antonio Echevarria Silva

Project: The Variable Message Sign (VMS)

The project involves the implementation of a variable message sign (VMS) system based on the Sense HAT sensor and controlled from a web interface. This system includes:

- Web interface
- Data acquisition script
- REST API with Flask
- SQLite database for persistent storage

Web User Interface

Endpoint: `/`

The `main.html` template correctly implements the main view, which includes:

- Title: `Group D606 VMS Control Panel`
- Current date displayed in the footer
- Last message displayed in the VMS (Sense HAT)
- Form with text field and button to submit new messages
- Links to each variable (`temperature`, `humidity`, `pressure`) on its own page
- Links to the live endpoints for each sensor (`/live/<variable_id>`)

Endpoint: `/variable/<variable_id>`

Each variable has its own view (`variable.html`) where the following are displayed:

- Variable name
- Last value recorded with your unit
- Link to return to the main page

Data Acquisition

The `dataadquisition.py` script runs in isolation and performs the following functions within an infinite loop with a 1-second interval:

- Sensor readings:
- Temperature
- Humidity
- Pressure
- Storing readings in the database (`measures`)
- Activating automatic conditions:

Automatic conditions implemented:

- **Night mode** (Sense HAT in `low_light`):
- Activated between 8:00 PM and 7:00 AM
- **Conditional messages** displayed on the LED:
- `"ICE: mandatory use of tire chains"` if the temperature is below 5°C
- `"Temperature is more higher"` if the temperature is above 35°C
- `"Drive carefully, it can rain at any time"` if the humidity is above 80% and the pressure is less than 900Pa
- "Have a nice trip" under normal conditions

Flask RESTful API

The `api.py` script implements multiple endpoints that allow real-time interaction with the system and querying historical data.

Available endpoints:

`GET /probe`

- Returns the string `"VMS is working properly"` to check the server status.

`GET /`

- Displays the main interface with VMS status and interaction forms.

`GET /variable/<variable_id>`

- Returns an HTML page with the most recent value of the requested variable.

`GET /live/<variable_id>`

- Returns a JSON file with the live value of the corresponding variable.
- Example output: `{"temperature": 24.3}`

`POST /` (web form)

- Sends a new message to the VMS.
- Stores the message along with the current sensor measurements in the database (`messages` + `measures`).

`GET /search?date=YYYY-MM-DD`

- Returns all messages stored on a specific date with their associated measurements.
- Results displayed on the side of the main template.

DB Design + SQLite File

The `initdb.py` script creates the initial structure of the `Project.db` database, consisting of the following tables:

`sensors`

Field	Type	Description
id	INTEGER	Primary key
name	TEXT	Sensor name
description	TEXT	Description
virtual	INTEGER	0 if physical, 1 if virtual

`variables`

Field	Type	Description
id	INTEGER	Primary key
sensor_id	INTEGER	Foreign key to `sensors.id`
name	TEXT	Variable name
description	TEXT	Description
units	TEXT	Units of measurement

`measures`

Field	Type	Description
id	INTEGER	Primary key
variable_id	INTEGER	Foreign key to `variables.id`
measure	REAL	Measured value
date	TIMESTAMP	Date and time of measurement

`messages`

| Field | Type | Description | |-----------|----------| | message | TEXT | Message displayed on the LED matrix | | date | TIMESTAMP | Date and time of the message | | modify | BIT | 1 if it is a manual message, 0 if it was generated by a trigger |

Initial data

- 3 registered physical sensors: Pressure, Humidity, and Temperature
- Variables to associated with each sensor with its units (Pa, %, ºC)