Motorvehicle University of Emilia-Romagna (MUNER)

Master Degree in Electronic Engineering for Intelligent Vehicles (EEIV)

Vehicular Communications

Final Project

November 30, 2023

The final project is related to the management of an *on-board* vehicular network featuring the interaction between different OBUs equipping a vehicle V, in detail in charge of handling different functionalities of the vehicle itself (e.g., infotainment, keyless system, braking system, etc.). The project will rely on modelling part of this vehicular network at application layer (i.e., through software libraries) in a simulated environment in your laptop, but with a potential application on physical Single Board Computers (SBCs) deployed in a laboratory environment.

1 Infotainment System

With regard to the infotainment system, let's model the `RadioHandler` as a Python module supporting `GET` and `POST` HTTP requests as follows:

- `GET` (from an external HTTP client) on the `/radio` HTTP endpoint for obtaining the current radio station's name;
- `POST` (from an external HTTP client) on the `/radio-change` HTTP endpoint for changing the radio station's name.

Then, each time that the radio station's name is changed, the `RadioHandler` should publish on the `vc2324/radio` MQTT topic this update (meaning the new radio station name).

2 Weather Information

The `WeatherHandler` is in charge of receiving and internally storing the current weather informations and the forecasting for the next 2 hours, for the sake of management of the trip. More in detail, the `WeatherHandler` should be able to support the following HTTP requests:

- `POST` (from an external HTTP client) on the `/weather-current` HTTP endpoint for receiving the current weather conditions;
- `POST` (from an external HTTP client) on the `/weather-forecast` HTTP endpoint for receiving a forecasting on the weather conditions for the next 2 hours.

Then, each time the `WeatherHandler` receives an update, it should publish this update on a specific MQTT topic (namely: `vc2324/weather-current` and `vc2324/weather-forecast`, respectively).

3 Driver Monitoring System (DMS)

Since the modelled vehicle is a *smart* vehicle, it is interested also in considering the status of the driver, in order to understand if he/she is tired, angry, etc. Then, the `DMS` should internally (on a periodic basis) decide which is the current status of the driver, and it should support the following HTTP request:

- `GET` (from an external HTTP client) on the `/dms` HTTP endpoint for providing the current driver's status (e.g., as a string).

Then, each time the `DMS` changes the driver's status, it should publish this new information also on the `vc2324/dms` MQTT topic, as a useful information for the `CentralOBU`.

4 Central OBU

The `CentralOBU` is the core of the vehicle, since it is in charge of deciding how the vehicle should react on the basis of the inputs arriving from the other OBUs. To this end, the `CentralOBU` should subscribe to all the MQTT topics of interest and periodically estimate the risk level (the way in which the inputs take part in the decision it is up to you).

Then, in the case of a certain risk level, the `CentralOBU` should immediately publish an alert message on the `vc2324/alert/brake` MQTT topic, in order to request the braking system to safely stop the vehicle. At the same time, the `CentralOBU` would allow an external HTTP client to obtain the current status of each peripheral OBU through a `GET` HTTP request on the `/vehicle-status` HTTP endpoint.

5 Braking System

As mentioned before, the `BrakingHandler` is in charge of activating the braking system of the vehicle, so it should subscribe to its proper MQTT topic (denoted as `vc2324/alert/brake`) in order to receive possible alert messages. Please, pay attention that the vehicle should be secured in such a way, since an attacker might try to steal the vehicle or to activate the braking system when not needed. Therefore, the `BrakingHandler` should react ONLY in the case the remote keyless system has recognized the driver's key as the valid key, otherwise it should send an alert message on the MQTT topic denoted as `vc2324/alert/key-not-recognized`.

6 Remote Keyless System

Last (but not least), as anticipated, the vehicle is provided with a keyless system enabling the vehicle to start or to be stopped, given the presence of the right keycard in the neighborhood of the vehicle itself. Thus, a proper `KeylessOBU` should be in charge of handling this functionality, in detail supporting the following HTTP requests:

- `POST` (from an external HTTP client) on the `/keyless` HTTP endpoint, accepting a string containing the identity of the driver and a pseudo-random sequence to be recognized by the vehicle;
- `GET` (from an external HTTP client) on the `/driver` HTTP endpoint, returning the identity of the last driver opening the car, or the indication that someone (unrecognized) tried to open the vehicle.

Then, each time that a known driver is recognized by the proper OBU, the `KeylessOBU` should notify all the other OBUs publishing a proper message on the MQTT topic denoted as `vc2324/key-is-ok`.

7 Common Parameters & Suggestions

For the sake of completeness, the physical OBUs have been deployed in a laboratory environment with the following parameters:

- MQTT broker: `172.26.0.48:4883`
- Each HTTP server should be exposed on the HTTP port `8888`
- Dataplicity homepage: `https://www.dataplicity.com/`
- There exists four Raspberry Pi 4 with the following IP addresses:
  - `rpi-vc-01`:
    - Local IP address: `172.26.0.48`
    - Dataplicity username: `iotlab.unipr+rpivc01@gmail.com`
    - Password: `VC2324OBU`
    - Home folder: `/home/pi`
  - `rpi-vc-02`:
    - Local IP address: `172.26.0.51`
    - Dataplicity username: `iotlab.unipr+rpivc02@gmail.com`
    - Password: `VC2324OBU`
    - Home folder: `/home/pi`
  - `rpi-vc-03`:
    - Local IP address: `172.26.0.249`
    - Dataplicity username: `iotlab.unipr+rpivc03@gmail.com`
    - Password: `VC2324OBU`
    - Home folder: `/home/pi`
  - `rpi-vc-04`:

    - Local IP address: `172.26.0.227`
    - Dataplicity username: `iotlab.unipr+rpivc04@gmail.com`
    - Password: `VC2324OBU`
    - Home folder: `/home/pi`

- If you'll use Python, maybe you can use the `paho-mqtt` library (`https://github.com/eclipse/paho.mqtt.python`) for the MQTT protocol, and the `aiohttp` library (`https://docs.aiohttp.org/en/stable/web.html`) for the HTTP protocol.