Third_Week_Implementation

João Passos 1210646
Pedro Vicente 1180558

Major changes in hardware:

- In the project planning stage, we missed the fact that the Raspberry Pi GPIO (General Purpose Input Output) doesn't read analog signals nor has ADCs (Analog to Digital Converters). So, we introduced the ESP32 board to fulfill the task of reading the infrared sensors.
- Since the 3.2 LCD Display makes it difficult to connect more cables in the GPIO due to the lack of space, the motors were moved to the ESP32 board.
- An external power supply was inserted in order to power up the servo motors.

Main program:

- The main program was updated, so it would behave as expected in all different situations:
    - User ID arrival: a query to the DB is made in order to check if the user ID exists in Database. If so, publish a 'go' flag to MQTT topic.
    - User PIN arrival: when received in an MQTT topic, the PIN is compared with assembly function 'comparePIN'.
- Instead of controlling the infrared sensors and the motors directly as planned, the main program needs to publish an MQTT message to the ESP32 board to activate/read the motors and infrared sensors, respectively.
- Main program now inserts/updates the amount of capsules consumed for each user in the database.
- A thread was created in order to keep the mosquitto broker alive. With that, the main program is still able to run other code.
- Some global variables were used, but only because they are needed inside callbacks, which can't get more arguments passed since they are from external libraries.

MQTT

- The MQTT protocol was implemented with success in the Python Interface:
- The MQTT protocol was implemented with success in the ESP board.
- For security purposes, the Broker requires a user and password to access the broker. With this, we prevent unwanted users from accessing the topics.
- Through this protocol we exchange all type of information between the 3 systems:
    - Login info
    - Capsules choices
    - Stock
    - Other topics used for transmitting information relative to the end of execution of some subsystem. For instance, when all the capsules are dispensed, the

ESP32 publishes in order for the interface to know and proceed with its operation.

Python Interface:

- The 3.2 LCD Display was introduced to the system:
  The python interface display was moved from the laptop (as before for testing purposes) to the 3.2 LCD display
- The developed interface needed reorganization of all the elements since the RaspberryPi 4B doesn't allow a 320x240 headless resolution
- All the pages transitions were corrected depending on the buttons press sequences and published messages
- Refill is not implemented as it was specified but, to prove the concept, when the topic 'capsules/refill' is published, the interface resets the amount of capsules available.

ESP32:

- The ESP32 board was introduced in order to control the motors and read information from the infrared sensors. Due to the lack of pins and ADC, this implementation was a must-have.
- ESP32 is responsible for dispensing the capsules. To do this, the system receives the capsules choices via MQTT in a single topic. The information is published in a single string and then is divided in substrings to get the amount of capsules for each flavor.
- Infrared sensor reading is made when something is published in 'stockReq' topic. The system reads the sensors and publishes the result.
- After dispensing the capsules, the ESP32 publishes into different topics (one for each flavor) the amount of capsules chosen in order for the main program to update the database with the total number of capsules consumed by the current user.