Inventory Management

CYBERPHYSICAL SYSTEMS AND INTERNET OF THINGS - M.EIC - 2022-2023

João Carlos Machado Rocha Pires (UP201806079)

U. PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Agenda

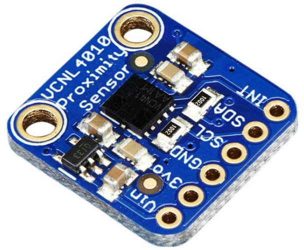
Application

- Fully-functional prototype of an inventory management system, using software and hardware, the connection between both, and a physical mockup of a warehouse for demonstration purposes.
- Each interconnected device is associated to a section of the warehouse:
  - Responsible to track the physical products associated to;
  - Highlight itself whenever needed;
  - Gather environmental information using the sensors associated to each device.
- Usage of AWS IoT Core services for bi-directional communication between the interconnected devices and the Outsystems application.
- All the communication is made by WiFi using the MQTT protocol (publisher/subscriber).

Planned Proof-of-Concept

Materials:

- 3 x ESP32 (M5 Atom Lite)
- 1 / 2 x Load Cell HX711 Amplifier (Sensor de Peso)
- 1 / 2 x VCNL4010 (Sensor de Proximidade)



Software:

- Azure IoT services
- Microsoft Power Apps

Realised Proof-of-Concept

Materials:

- 3 x ESP32 (M5 Atom Lite)
- 1 x TVOC/eCO2 Gas Sensor Unit (SGP30)
- 2 x ENV III Unit with Temperature, Humidity, and Air Pressure Sensor (SHT30+QMP6988)

Software:

- AWS IoT Core services
- Outsystems

Realised Proof-of-Concept - Reasons for change

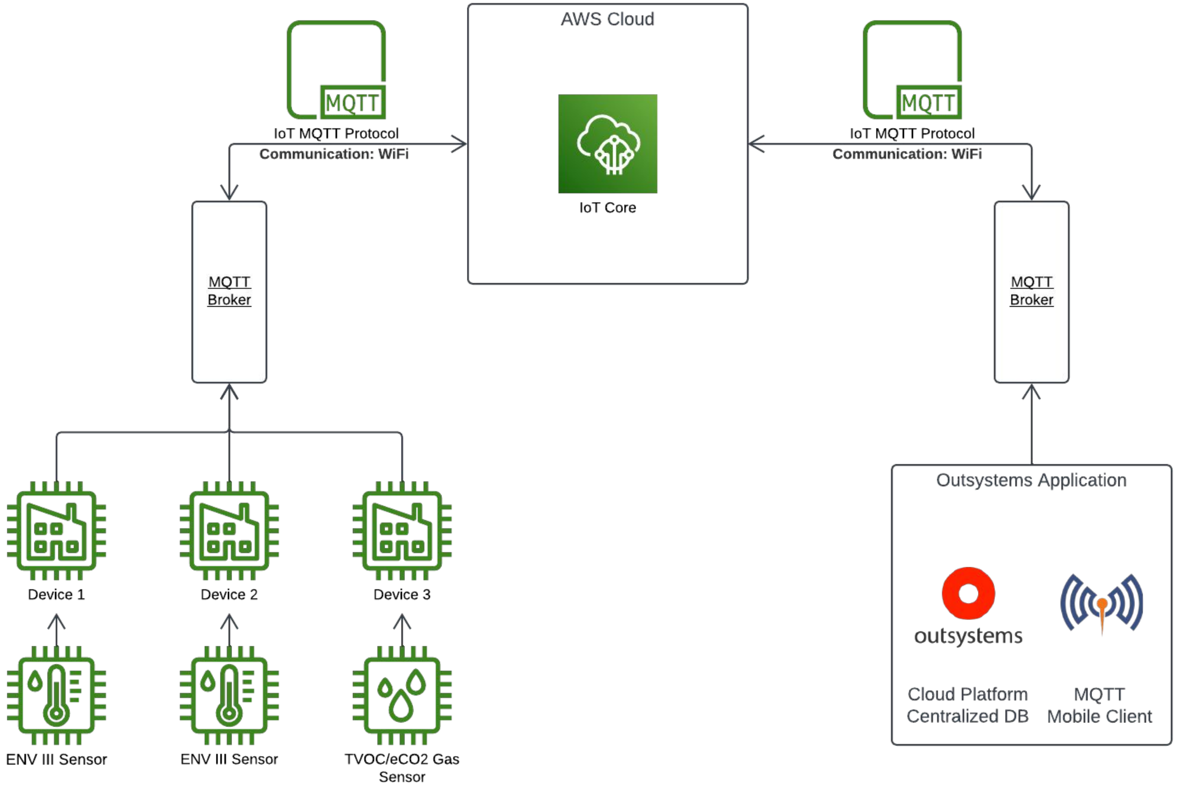
Materials

- Not available during the project development timeframe

Software

- Problems with the setup led to the usage of an alternative technology
- With the change on the IoT cloud platform, the technology to build the frontend was also changed

Architecture



Architecture

On one hand, each one of the devices is connected to a sensor and to the IoT Core services of AWS Cloud through the usage of the MQTT Broker and the IoT MQTT Protocol. All the communication is made via WiFi.

On the other hand, the Outsystems application, which uses the MQTT Mobile Client component, has a centralized DB on the cloud, accessible in all the instances in which the application is running. The connection with the IoT Core services of AWS Cloud is made in the same way as the one with the devices.

The only difference in the connection with AWS is that devices connect via endpoint and certificates, while the application uses the endpoint and keys.

Decision Justification

The usage of WiFi for communication is justified by the larger communication range and better penetration through walls (especially important in warehouses). It has the disadvantage of the higher energy consumption, which is an acceptable drawback for the use case.

The usage of AWS IoT Core services is justified by the diversity of IoT features, by the strong support and complete documentation that AWS offers. It has the disadvantage of the limit of messages per month, which is not relevant for the project according to the needs.

The usage of Outsystems as the development tool for the application is justified by the acceleration of the development cycle and by the offering of all the needed functionalities for the application that was developed.

Project Flow - Devices

Each device starts by connecting to a predefined WiFi network on power up. Then, connects to AWS IoT and subscribes the following topics:

- *info*
- *new_request*
- *finished*

Before looping into the device's logic, the device publishes a message on the topic *initialization*.

Project Flow - Devices

Inside the logic loop of the device, every 30 seconds, the device publishes the readings of the sensors on the topic *sensors_readings*.

If there's a request on going [1], the button is clickable and there's a timeframe of 5 seconds [2] to record the number of clicks. Once the timeframe ends, a message is published on the topic *product_pickup* with the quantity of the product taken.

[1] - A request on going means that a message was published by the application on the topic *new_request* including at least one product associated with the device. Also, it's only on going as long as the button was not pressed as many times as the ones referring to the desired quantity.

[2] - The 5 seconds timeframe refers to the period that is reset on every click, i.e. if there's a click, the following one is within 5 seconds and the one after that one is within 5 seconds after the second press, it will count as three clicks. Once that more than 5 seconds pass since the last click, the timeframe ends.

Project Flow - Devices

If a message was published in the topic *finished*, then the device will turn its LED to red.

If a message was published in the topic *new_request*, the device will check if it has at least one requested product associated and the desired quantity. If the desired quantity is equal or less than the one the device has, the LED will turn to green. If the desired quantity is higher than the one the device has, then the LED will turn to blue. Otherwise, it will remain red.

If a message was published in the topic *info*, the device will update the product name and quantity it has associated with the information contained in that message.

Project Flow - Application

The application is required to have a WiFi connection to work properly. As the devices, it starts by connecting with AWS.

Then, when a message is published on the topic *sensor_readings*, the DB is updated with the most recent information and the dashboard refreshed with the newest data.

On a new submitted request, a message with the list of products and corresponding quantities is published on the topic *new_request*.

By pressing the button ‘Finish’, a message is published on the topic *finished*.

Project Flow - Application

Upon the publication of a message on the topic *initialization* by a device, the application publishes the “answer” on the topic *info* with the device’s product and quantity.

Upon the publication of a message on the topic *product_pickup* by a device, the application logic checks if the request is finished. If yes, publishes a message on the topic *finished*. Otherwise, it just updates on the DB the quantity of each device’s products.

Messages

Listed in this and in the following slides are the prototypes of the messages used in the project.

- *initialization*

```
{
  "id": T
}
```

Note: T is a placeholder for text and N for a number.

Messages

- *info*

```
{
  "id": T,
  "product": T,
  "quantity": N
}
```

- *sensor_readings*

```
{
  "id": T,
  "tmp": N,
  "hum": N,
  "pressure": N,
  "tvoc": N,
  "eco2": N,
  "rawh2": N,
  "rawethanol": N
}
```

Note: Depending on the device, some sensor measures might be null. e.g. The ENV III sensor only measures *tmp*, *hum* and *pressure*, so the remaining ones will be null.

Messages

- *new_request*

```
{
  "products": [T, T],
  "quantities": [N, N]
}
```

- *product_pickup*

```
{
  "product": T,
  "quantity": N
}
```

- *finished*

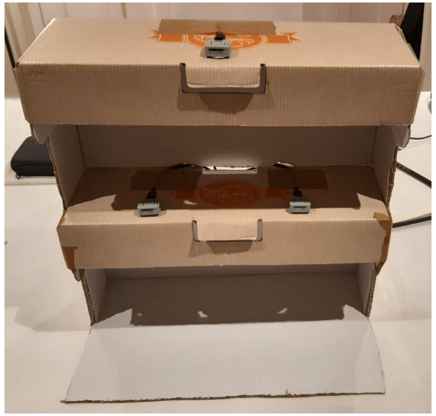
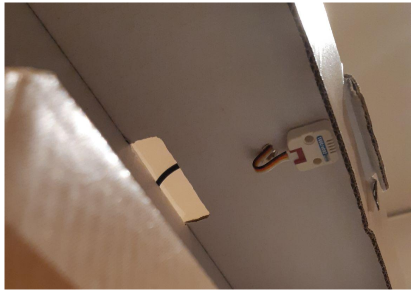
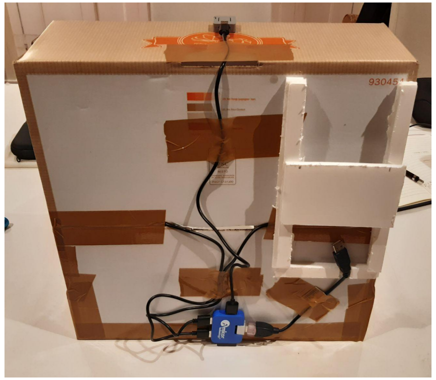
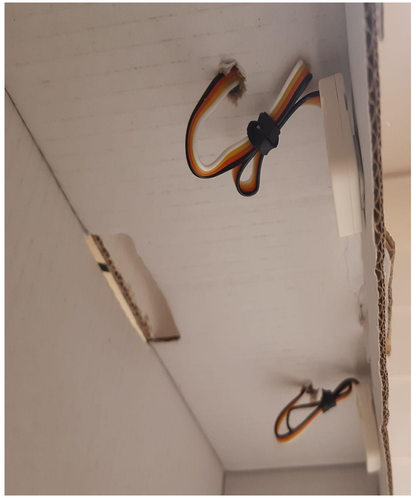
```
{
}
```

Physical Mockup

For demonstration purposes, a physical mockup was built.

The sensors associated to each device were hidden under the shelves.

A USB hub facilitated the connection of all the devices to a single USB output cable that connects to a power source (e.g. powerbank).

Demo Video

Biggest Challenges

- Setup of the devices
    - WiFi communication
- Setup of the cloud platform
    - MQTT protocol implementation
    - Communication with the devices
- Testing
- Time constraints

What I’ve learned

- Program an ESP32 device with WiFi capabilities
- Setup of an IoT cloud project (work with AWS IoT Core services)
- Implementation and setup of the MQTT protocol in different technologies
- Not everything goes as planned, so alternatives have to be considered prior to the start of a project (have a plan B / backup)

Self-evaluation

Taking into account that

- I’ve made the project all by myself,
- I’m a working student,
- I have 6 curricular units this semester,
- I only received the material for the project a couple weeks after the time I expected to receive it, and some material never came, actually,

I managed to do an acceptable MVP for the proposed subject.

I recognize that the project has some limitations, but it was the best I could have done considering the above mentioned constraints.

That said, I would propose 15 as the project grade.

References

ATOM Lite Official Documentation

ATOM Lite Official Github

ATOM Lite - Get Started with Arduino

AWS IoT Core services setup

Arduino programming with MQTT protocol

AWS Credentials Generator guide

AWS IoT Core services connection with ESP32 devices

Outsystems - build IoT mobile application

Outsystems - connection with AWS IoT Core services

Outsystems Forge - MQTT Mobile Client