CONNECTED & SMART HOME – SECOND REPORT

Problem definition

The aim of this project is to design a privacy-friendly self-hosted smart home system. With this system the user does not have to trust that the company makes good use of his data, since he has them under control.

A smart home system collects a large amount of data for its operation when it is not strictly necessary. In addition, many of these companies trade in such data, which can be a problem for many users.

State of the Art

General solutions:

- Amazon echo family and Alexa: Brand of smart speakers developed by Amazon. Thanks to its integration with Alexa assistant, manufacturers can use its API and add support for their smart devices. Supports Wi-Fi and Bluetooth, but the most majority of devices they integrate with only use Wi-Fi, making it a poor choice for low-power devices.
- Samsung SmartThings family: Samsung's smart home brand. Unlike Amazon's Alexa, is a set of devices manufactured exclusively by Samsung and integrated with its services. The hub connects directly to the Internet via Wi-Fi. It also supports communication protocols such as ZigBee, Z-Wave to communicate with the various sensors.
- Phillips Hue family: Set of products manufactured by Phillips. It has less variety of devices than Samsung's brand because they are mainly focused on lighting products such as light bulbs or lamps. The connection between the hub and the bulbs uses the Zigbee protocol, while the hub can be connected to the internet via Wi-Fi or ethernet.

Privacy focused solutions:

- Ethical Smart Home[i]: This is an incomplete or abandoned project to providing a privacy-aware smart home system. Although it is developed by a company, they do not seem to sell any product; they simply provide vague instructions on how to build the devices. The design of all devices is based around Arduino. For wireless communication it uses a NRF24L01 chip which uses the Enhanced ShockBurst protocol.
- Candle[ii]: This project similar to the previous one. However, it seems complete and much more extensive. They still do not sell any devices but provide concrete instructions on how to build each device in addition to the source code. Candle's designs are based on the RF-Nano Arduino, which also uses the Enhanced ShockBurst protocol. The hub also makes use of a raspberry pi for operation, thanks to which it can connect to Wi-Fi.
- PrivHome: Privacy-Preserving Authenticated Communication in Smart Home Environment[iii]: This is a research paper proposing an authentication system, secure data storage, and query for smart home systems.
- Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping Sleeping[iv]: This is a research article focused on evaluating possible threats to smart home security systems and proposing some solutions.

After carrying out this small market research, we can conclude that from the perspective of our problem there are two very specific types of products:

- Generalist products that are easy to use and well supported, but with dubious respect for the owner's privacy.
- Niche products that respect the user's privacy but require certain knowledge on the part of the user.

Ideally, our product should be able to have the best of both worlds: a series of devices with good integration that are easy to use, but that also respect the user's privacy by keeping the data in the main node or in a local server.

Solution definition

This system would consist of a small server and a set of sensors and actuators. The server would connect to the internet via Wi-Fi or Ethernet and to the sensors/actuators using 6LoWPaN and CoAPP. Later the user could access this server through an app configuring the IP of the house. If the user does not have a static IP or has problems with this configuration, then the company can offer an intermediary service to facilitate the connection between the server and the app. Since the product is privacy-focused, users are not expected to use this intermediary service by default.

In comparison with other similar services, this project is a complete product that does not need to be built by the user. In addition, by removing the Arduinos from the design, we can considerably reduce power consumption, allowing the sensors to use batteries and have a long life.

Potential users of this product are people who want the convenience of a smart home system but are concerned about the privacy implications that a normal system may have.

IoT Architecture

This project requires the usual sensors and actuators in a smart home system:

- Sensors: Smoke and fire detectors, light sensors, water sensors, motion sensors, temperature sensor, and weather sensor.
- Actuators: Actuators to open/close doors, windows, cabinets, drawers, etc., blind actuators, and electronic radio actuator for the control of the heating power of the heater.

Since this project is focused on privacy, the user must be able to have control over the data. This is why both the processing and storage of data is carried out by the central node that will act as the server. This node will also communicate with the user through an application installed on the mobile phone. thanks to this App the user will be able to see all the data through tables and/or graphs.

A system like this consists of a variety of nodes. In fact, for every functionality there will be several nodes (actuators and sensors separately):

<u>Nodes for the functionality of detecting smoke and fire</u>

- Sensors: Smoke and fire detectors.

Example 1: With a warning from the app, the user can act quickly. There is also the possibility of calling the fire department.

<u>Nodes for the functionality of lighting control</u>

- Sensors: Light sensors and motion sensor.
- Actuators: Actuators to turn on/off lights.

Example 1: If after leaving one room we leave the light on, as the presence sensor does not detect us, the light will be turned off automatically.
Example 2: If there is enough light in a corridor detected by the light sensor, the light will not be switched on automatically when presence is detected.

<u>Nodes for the functionality of flood detection</u>

- Sensors: Water sensors.
- Actuators: Actuators to cut off the water supply.

Example 1: If the tap is left open and a room is flooded, the actuators automatically cut off the water supply. In addition, the system sends an alert to the owner who receives it on his smartphone and allows him to act as quickly as possible.

Nodes for the functionality of opening/closing doors, windows, cabinets, etc.
- Sensors: Sensors to check the status of each element.
- Actuators: Actuators to open/close doors, windows, cabinets, drawers, etc.

Example 1: The user is notified through his smartphone or laptop when something at home is opened unexpectedly.

Nodes for the functionality of acting according to the weather
- Sensors: Weather sensor.
- Actuators: Actuators to open/close doors, windows, cabinets, drawers, etc.

Example 1: If the sensor detects that it is raining and we the windows are still open, it warns the user and allows him to close them.

Nodes for the functionality of acting according to the temperature
- Sensors: Temperature sensor.
- Actuators: Electronic radio actuator for the control of the heating power of the heater.

We could say that this project has a level 6 architecture since we have multiple sensors and actuators. However, this is not exactly so since both the storage and the data processing are done in the control node. There is no level that accurately represents this architecture.

As it is based on a star topology, our system will have a segment for each connection between the central node and the rest of the nodes (sensors/actuators), in addition to the segment referring to the connection with the user's mobile phone. Therefore, the number of segments will be equal to the number of nodes. Some segments:

- Central node - Each smoke and fire detector
- Central node - Each light sensor
- Central node - Each water sensor
- Central node - Each motion sensor
- Central node - Temperature sensor
- Central node - Weather sensor
- Central node - Each sensor that checks the status of doors, windows, drawers etc.
- Central node - Each actuator to open/close doors, windows, drawers etc.
- Central node - Each actuator to turn lights on/off
- Central node - Actuator that cut off the water supply
- Central node - Actuator for the control of the heating power of the heater.
- Central node – User's mobile phone.

Performance planning

The frequency with which each node sends data depends very much on its function; for example, measuring temperature, which varies more or less slowly and predictably, is not the same as measuring smoke in a room. If you take too long to measure the temperature, you may lose your house.
- Smoke and fire detector → Once every 30 seconds
- Light sensor → Once every five or ten minutes
- Flooding sensor → Once every minute; since a flood is potentially catastrophic, it is important that this data is constantly updated.
- Motion sensor → Once every two minutes

- Temperature sensor → Once every five minutes
- Weather sensor → Once every minute
- Sensors that checks the status of doors, windows, drawers etc. → These sensors would record every opening/closing by using interruptions, so there is no need for continuously checking its status.

In a smart home system, there are a variety of different nodes, so each type of node needs different types of data. In addition, in most cases each sensor node has its actuator counterpart, which will provide it with the data it needs.

Communication protocols

Given the nature of this system, there are two types of connections:

- Sensor/Actuator ⟷ Server
- User ⟷ Server

Each one uses different protocols and therefore must be analyzed differently.

1) <u>Sensor/Actuator ⟷ Server:</u>

This type of connection uses CoAP over 6LoWPAN. During the design we also considered using LoRaWAN or Bluetooth LE for wireless communication. We saw that the maximum range of LoRaWAN was too wide, which may not be the most suitable protocol for a smart home system. Bluetooth LE seems to be a good technology, but since at the time of writing this project has only been given in class 6LoWPAN, we decided to opt for the latter.

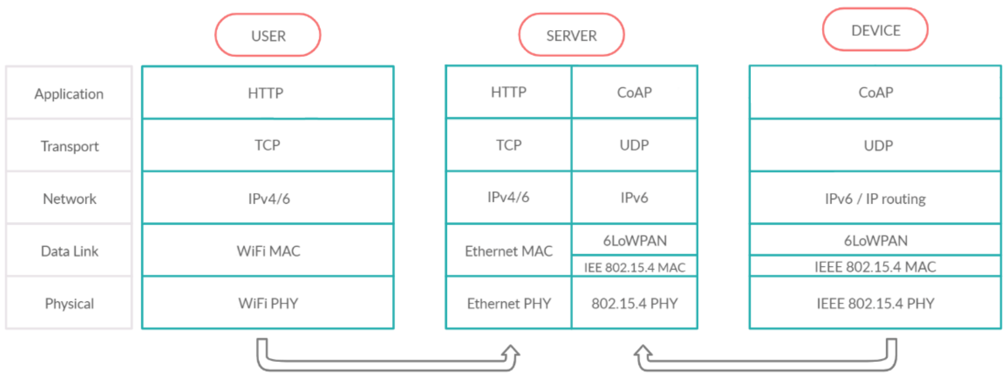
In the case of communication at application layer level we had two options: MQTT and CoAP. We chose the second one because of its option to observe resources and its similarity with a REST architecture. This last feature has the potential to simplify the development of the Server⟷User connection.

- Link layer: IEEE 802.15.4 MAC and 6LoWPAN
- Network layer: IPv6
- Transport layer: UDP
- Application layer: CoAP

2) <u>User ⟷ Server:</u>

For the User ⟷ Server connection we have opted for a REST architecture, as this allows us to develop both an App and a web frontend with practically the same architecture in the backend. Furthermore, thanks to the similarity between REST and CoAP, the server can implement a mapping between both so that the user has control over the actuator nodes.

- Link layer: Wi-Fi MAC
- Network layer: IPV4 or IPV6 depending on the network configuration
- Transport layer: TCP
- Application layer: HTTP



Each segment of our smart home will have one or more resources; some of them will simply return data items, others sensor readings, etc. Each of them will be hosted in the server and the user will be able to access them through the CRUD "verbs" that we already know (Create, Retrieve, Update, Delete). By GET requests, the user interested in the status of one of the resources will be able to initiate a request to the server. After this, the server will return a response with a representation of the requested resource at the time of the request.

On the other hand, some actuators need to be controlled by the user directly from the app. In this case the app will make a post HTTP request to the server, which will convert that request into a CoAP request that will modify the state of the actuator node.

Smoke and fire detectors, light sensors, water sensors, motion sensors, temperature sensor, and weather sensor will return the sensor readings at the time of the request. Remember that the user will be able to check the graphs and tables of the status of each sensor throughout the application. Likewise, the status of doors, windows, drawers, etc. can be checked by accessing the resources that refer to the readings of these sensors.

The sensors that provide useful information to the user, such as temperature or weather station, send this information each time the measurement is made; waking up the deep sleep node unnecessarily is avoided and at the same time the user has the most updated data possible. On the other hand, the emergency warning sensors (flood or smoke) only send the information when there is a positive threat, thus saving energy. The sensors that watch over the doors and windows must be configured by the user, so that if he wants them to act as a security measure, then they will work as the emergency warning sensors. On the contrary if the user only looks for statistics of how long the windows are open then these nodes will collect data and send it when they accumulate a certain amount of detections. In this way, doors and windows that experience less activity will save more energy.

[i] http://www.cloakingcompany.com/home/
[ii] https://www.candlesmarthome.com/
[iii] https://www.researchgate.net/publication/332861515_PrivHome_Privacy-Preserving_Authenticated_Communication_in_Smart_Home_Environment
[iv] https://content.sciendo.com/view/journals/popets/2019/3/article-p128.xml