MOBILE AND EMBEDDED COMPUTING

IoT ecosystem proof of concept

Mobile And Embedded Computing - 2020–2021 [INFO-Y113]

Author:

Jolan WATHELET : 000480582

UNIVERSITÉ LIBRE DE BRUXELLES ULB

August 18, 2021

Contents

1 Introduction

The purpose of this project is to develop a proof of concept for a new ecosystem of IoT. The devices should be able communicate to the Internet and interact with a user throughout a user interface.

The code is new compared to the group project from June. I have taken into account the grade we got and focused my work on making a better protocol, the control to iOt network part is still not working because of time constraint and the fact that i am doing it alone, but it is also improved compared to June since I explain how it could/should work.

This report explains how i designed this ecosystem and the difficulties encountered.

This github repository : `https://github.com/j0wa/embeddedUCLaugust`

2 Communication Protocol

My goal with this protocol was threefold, have a small packet size, have a constant packet size and finally have the possibility to expand the number of command supported in the future.

The packets are called messages and I will refer to them as such.

The messages are composed of three unsigned Bytes.

The first one is the message type, they are six type of messages that are defined :

- INIT - Used by a device when first connecting to the borderRouter.
- ACK_INIT - Used by the borderRouter to give a clientId to the newlly connected device.
- ERR_ID_UNDEFINED - Used by the borderRouter to tell a device it's clientId doesn't match the saved one.
- START_SUBSCRIBE - Used by the borderRouter to tell a device it would like to receive regularly the information it can provide (temperature, pression, etc).
- STOP_SUBSCRIBE - Used by the borderRouter to tell a device it would like to stop receiving the information.
- ACTION - Used by both the borderRouter and the devices to sent order, set value or warn of a trigger.

This set of message type is simple but there is room to expand with 250 more possible commands.

The second unsigned Byte is the clientId. I wanted the possibility to have multiple devices withe the same ip address, and this is why using unique clientId can be usefull. We could imagine a wall switch who can also do temperature sensor, each with their own client id.

The last unsigned Byte is the action field. When the message type is action this is were we define what action was performed or need to be done. Currently I have defined three action, on, off and toggle. The first 50 value are reserved for future action when the protocol is expended. When the value is more then 50 we consider that the field is used to transmit a value, this value depend on the device concerned. If we take a temperature sensor with a subscription from the border router, the temperature will be sent in the following format, the temperature messured + 50 to offset any negative temperature + 50 because the value need to be above 50 in the field.

Seeing that the value in the action field is above 50, the border router need to do minus 100 to get the temperature. If the message type is INIT the action filed is used to inform the border router about the type of device initiating a connection, the protocol can incorporate 256 type of device in its evolution.

The source code of the protocol is the file named **protocol.c** in the Github repository.

3 Implemented IoT Devices

The source codes of devices are in the folder **devices** in the Github repository.

3.1 Light

Using the 3 LEDs in the Z1 platform. Its function is to receive instructions. Respond to the following commands :

- toggle - on - off - general command to control the 3 LEDs together.

3.2 Thermostat

Its function is to receive instructions. Respond to the following commands :

- value - from 0 to 40 representing the desired temperature in Celsius.

3.3 Wall Switch

Using the button on the Z1. Its function is to send instructions. Send the following data :

- toggle.

3.4 Temperature Sensor

Using the TMP102 sensor on the Z1. Its function is to send instructions. Send the following data :

- value - from -40 to 125, the temperature in Celsius.

3.5 Movement Sensor

Fake sensor sending a signal every 10 to 30 seconds. Send the following data :

- toggle.

4 Server

In this project, a server is placed on the computer side and communicate with a border router on the Contiki side, so the communication works throughout a tunnel that can (should) be created by the tunslip6 program provided by Contiki.

This server allows the user to communicate with the devices via a command line interface.

As said in the introduction, due to time constrain and the fact that I am doing this project alone the server does not work, but I still worked on the logic part of it in cli.c. The server can ask the borderRouter for a list of active devices, using the activeDevices defined in the protocol.h file, this allows the server to present the information to the user to define subscription and give orders to the different available iOt devices. The server can just send a message to the borderRouter, who can look the clientId to transfet it.

The source code of the server is the file named **cli.c** in the Github repository.

5 Messages Exchange

Example of messages exchange between the borderRouter and the light :

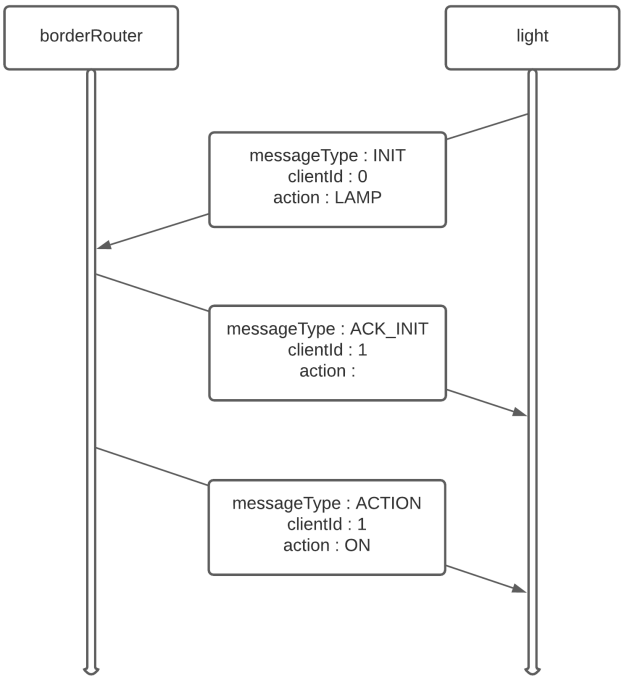

Figure 1: Diagram of the Init of the light

6 Automation

The automation is controlled on the borderRouter to make it more resilient. Two type of messageType are defined for automation, start and stop subscribe, when receiving these messages, the devices, depending on their type, will sent their value to the borderRouter in a predefined time interval. This allows the borderRouter to keep an automation table and react to the value received by giving the appropriate order to another device as defined by the user.

Since I don't have a working server to control it, the automation is not implemented in the borderRouter yet, but is present in the protocol.

7 Conclusion

Compared to June, the protocol is now properly defined and implemented in an usable format, it can be expanded through the addition of new commands in the future.