Telecom RIO201 | Final Report

Cheng-Yen Wu

2022

1 Introduction

1.1 Motivation

I suffered from irregular life because I couldn't help watching drama or playing video games until 4 AM so I'm wondering if it exists an system of IoT devices that can make my life more regular. At the same time, it is hard for someone to wake up. They wake up later and then they sleep later. For me, it a terrible cycle. My application is motivated by this scenario and I want to provide a IoT system that can help people to rest. The functions that I wish to enable are the followings

1. It sends a pop up message on the screen when it is late. If users continue watching drama, it can cut the power off.
2. It collects the sleeping information of users so that it can determine the quality of sleep and decide when to wake users up so that when users wake up, they feel less tired.
3. It wakes users up in a peaceful way (For example, open the curtain and let the sunlight go into the room) and it gives a reward to users if they success the respact the rythme in order to show that it is not painful to wake up and to make it easier as a habit to sleep in time and wake up with power.

1.2 Model and Architecture

I model the IoT application system with three working nodes. Their functions are described below, and the architecture is also presented in Figure 1.

- SmartWatch: It is used to collects the sleeping data but here it measures the presure to simplify the model.
- SmartController: It connects to SmartWatch to collect the quality of sleep of users. Based on the data, it decides whether or not to extent sleep time for user. When it's the wake-up time, it turns on the light and open the curtain. On the other hand, at a end of day it sends a message to users' personal computer or TV to turn them down.
- SmartAppliance (light, curtain, PC or TV): It responses requests from SmartController and turns itself on/off.

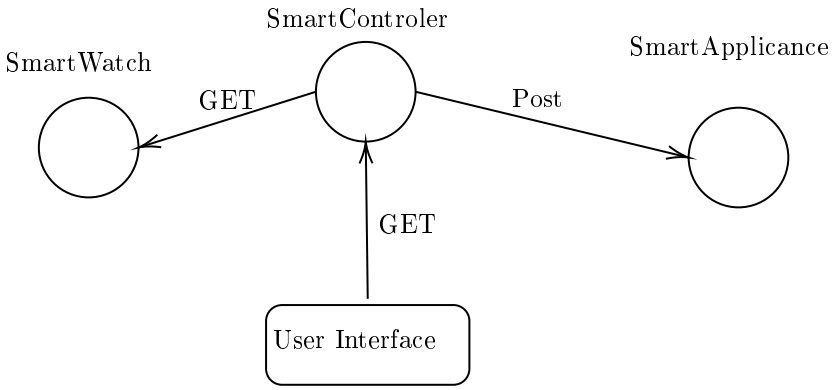

Figure 1: Architecture of Model

2 Implementation and demonstration

2.1 Implementation

I only implemented two devices out of three in my system, SmartWatch and SmartController.

- SmartWatch: I modelize the SmartWath with a `er-example-server` which sends back a sample of quality of sleeping at a given time. To simplify the model, it randomly gives a number from 0 to 9. The resource is named `new_pressure`, available with url `IPv6_ADDR:5683/my_res/new_pressure` and is implemented in file `resources/res-new-pressure.c`

- SmartControler: I defined three phases to implement SmartControler, `SLEEP` phrase, `WAKE` phrase and `DAY` phrase. In `SLEEP` phrase, it requests the quality of sleep once in a second during `SLEEP_MAX = 8` seconds. After `SLEEP` phrase, the SmartController enters `WAKE` phrase. The first thing the SmartControler does is to determine the extension sleep time (XST). In my simplified model the XST is determined by sum of 8 sample above devided by 10 seconds. After it waits XST seconds, it sends a message to SmartLight to turn it on and then it enters `DAY` phrase. It waits `24 - SLEEP_MAX - XST` seconds. It sends message to PC to turn it off then it enters again `SLEEP` phrase.
  Users may wonder the quality of sleep. Therefore, they can connect to SmartControler via `CoAP GET` request. To simplify the implementation, the SmartControler provide the last XST. The resource is named `res_new_delay`, available with url `IPv6_ADDR:5683/my_res/new_delay` and is implemented in file `resources/res-new-delay.c`. More precisely, `res_new_delay` use extern variable: `extern int delay_info` to share the information between the client side and the server side.

The part of code related with SmartAppliance is not implemented because of lack of time.

2.2 Demonstration

The demonstration is presented in the joined video `demo.webm`. The console above belongs to SmartWatch and the console below belongs to SmartControler. In `demo_delay.webm`, I showed that a user can request the last XST via coap get API and the value is indeed equal to the latest XST.

3 Comparison CoAP and HTTP

I didn't have an enough time to realize an experience that can illustrate the difference between CoAP and HTTP. Based on the slide of TP#2, It's interesting to count the number of packets exchanged and to measure the time of between sending and receiving via the command `time`.

Annex

I put my code in the folder `code`. They are

- `er-example-client.c` implements SmartControler.
- `er-example-server.c` implements SmartWatch.
- `res-new-pressure.c` implements the resource `new_pressure`.
- `res-new-pressure.c` implements the resource `new_delay`.
- `extern_var.h` declares the extern variable `delay_info`.