Module: Internet of Things **2024/25**

Lab Session 6: A Simple Alarm

*Lecturers: Dr Moh***ammad Shojafar, Dr Chuan H Foh, Dr Ahmed Elzanaty**

6.1 Introduction

The goal of this session is to develop a simple temperature detector and alarm. This detector is able to take the measurement of the temperature regularly.

You need to implement an alarm mechanism which will be activated if the temperature exceeds a threshold, e.g. 28°C.

If a sensor measurement exceeds the temperature threshold, the LED light will be activated and a text "Alarm!" will be printed on the screen.

Equipment and materials:

- ❑ Lab PC or your own computer
- ❑ An XM1000 mote (optional, collect it during the lab)
- ❑ InstantContiki-2.6 image file, see “Lab Session 1” for the download link

6.2 Learning Objectives

We expect you to know the following from the previous lab sessions:

- How to launch Contiki-2.6 in the lab (if you are attending the lab) or from your own PC (if you are working remotely)
- How to compile and upload your code onto XM1000 mote, and debug your code by logging debugging messages onto the console (if you have a mote);
- How to load a Cooja simulation environment, start and stop the simulation, and debug your code by logging debugging messages onto the console;
- Be familiar with Cooja environment and Contiki programming structure;
- Be able to read the display meaningful sensor readings;

In this lab session, you will learn:

- How to control LEDs on the mote;
- How to detect a push button event (the white button on the mote);

6.3 Exercise

Your task is to build a fire alarm application. The following is the specification of the application:

- The application should carry a temperature threshold. This threshold can be set arbitrarily during the initialization.
- After the initialization, the application periodically senses the surrounding temperature and logs the temperature reading on the console.
- When the sensed temperature exceeds the threshold, an alarm is activated. During this time, the application logs an “Alarm” text on the console and turns on all LEDs on the mote.

- The alarm can only be deactivated by a button event. When the user pushes the user button, the threshold is first adjusted to a value 0.5°C higher than the last sensed temperature value, and the alarm is deactivated. The "Alarm" text is no longer shown on the console, and all LEDs are turned off immediately.
- Should the temperature exceed the new threshold, alarm is activated again. Likewise, the alarm can only be deactivated by the push button event.

You may use the provided skeleton code for this task. The skeleton code implements a simple periodic process that increments a counter. The counter is initially set to 0. When the counter reaches 10, all LEDs on the mote are turned on. When the user button is pressed, the counter is reset, all LEDs are turned off.

The following is how we control LEDs. We first include the following header file:

```
#include "dev/leds.h"
```

There are several useful functions defined in the header file for controlling the LEDs:

```
unsigned char leds_get(void);
void leds_on(unsigned char leds);
void leds_off(unsigned char leds);
void leds_toggle(unsigned char leds);
void leds_invert(unsigned char leds);
```

In the skeleton code, we use the following function to turn on and off LEDs:

```
leds_on(LEDS_ALL);
leds_off(LEDS_ALL);
```

The following is how we use the push button. We first include the following header file:

```
#include "dev/button-sensor.h"
```

We then active the button by the following instruction:

```
SENSORS_ACTIVATE(button_sensor);
```

In the skeleton code, we detect a button event in a loop. We need to wait for an event, and check that if the event is a button event:

```
PROCESS_WAIT_EVENT(); // wait for an event
if (ev==sensors_event && data==&button_sensor)
{
  // a button event is detected, do something...
}
```

For your implementation, you first need to navigate to "/home/user/contiki-2.6/surrey/S6" and find the skeleton provided in the folder for this exercise.

If you are working in Cooja simulator, you can find the simulation environment "cooja_alarm.csc" for this experiment. Modify the skeleton code to implement the fire alarm application. Once you have completed the implementation, you may launch Cooja and load "cooja_alarm.csc" to test your code.

The simulation environment contains 6 panels:

- Network: it shows the network setup.
- Sensor Control Panel: it provides two sliders for users to control the sensor inputs. Users are able to adjust the temperature and light intensity of the environment which will be picked up by the sensors. You will need to use the sliders to change the temperature and light intensity of the environment to test whether your program can produce the correct readings.
- **LEDs** (new!): it shows the status of LEDs.
- **Push button** (new!): it provides users to simulate pushing the button on the mote.

- Simulation Control: it provides several buttons for you to start, pause, run step-by-step, or reload to restart the simulation.
- Mote Interface Viewer: it provides user interface to control the mote. In the current setting, we control the USB of the mote via virtual serial port. This way, we can capture the logging messages and show the messages on this panel.