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

Lab Session 4: Sensor Reading

Lecturers: Dr Mohammad Shojafar, Dr Chuan H Foh, Dr Ahmed Elzanaty

4.1 Introduction

In the last lab session, we learned how to compile and run code on a sensor node, and some embedded C programming. The goal of this session is to read the values from the light and temperature sensors of a mote and output the values to the standard output (or console). We use the program from the last session and extend it with the sensor reading routines.

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

4.2 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;
- How to use a timer to create a periodic task.

In this lab session, you will learn:

- How to implement a given transfer function for a sensor.

4.3 Sensor Reading

To access the sensor functionalities, we have to include the header files for the light and temperature sensors.

```
#include "dev/light-sensor.h"
#include "dev/sht11-sensor.h"
```

After the process begins, we have to activate the sensors on the sensor node with:

```
SENSORS_ACTIVATE(light_sensor);
SENSORS_ACTIVATE(sht11_sensor);
```

And then eventually get the sensor readings from them with:

```
int light_sensor.value(LIGHT_SENSOR_PHOTOSYNTHETIC);
int sht11_sensor.value(SHT11_SENSOR_TEMP);
```

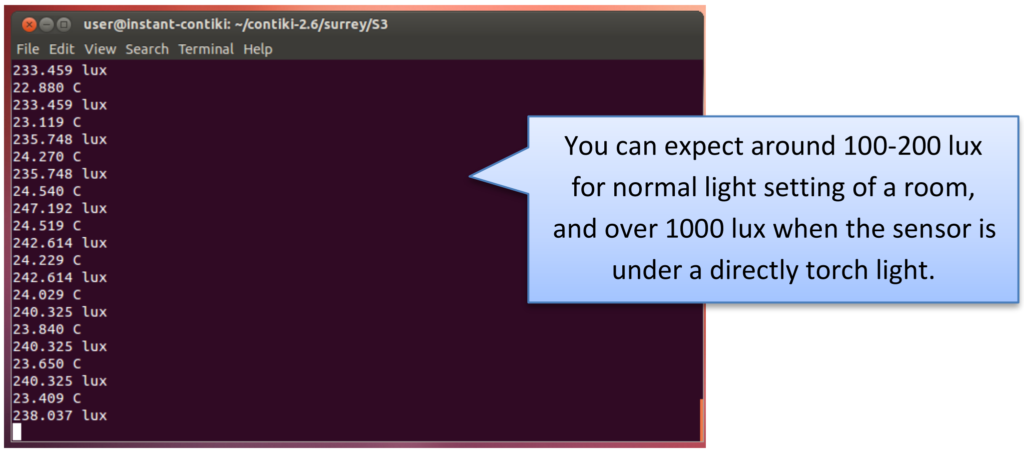
4.4 Exercise

You task is to implement a periodical process that outputs the light level and the temperature reading to the console every second. You may use the provided skeleton code for this task.

Sensors produce raw readings. You need to find an appropriate function to convert them to meaning values. You may refer to the lecture slides which have covered the basic concept of a transfer function.

You also need to read the sensor data sheets to derive the transfer function for the sensors. The data sheets are given on SurreyLearn. Note that in the lecture, we have done a step-by-step walkthrough on how to derive the transfer function for the light sensor based on the data sheet.

The following shows the expected outcome that you should produce.


user@instant-contiki: ~/contiki-2.6/surrey/S3
File Edit View Search Terminal Help

```
233.459 lux
22.880 C
233.459 lux
23.119 C
235.748 lux
24.270 C
235.748 lux
24.540 C
247.192 lux
24.519 C
242.614 lux
24.229 C
242.614 lux
24.029 C
240.325 lux
23.840 C
240.325 lux
23.650 C
240.325 lux
23.409 C
238.037 lux
```

You can expect around 100-200 lux for normal light setting of a room, and over 1000 lux when the sensor is under a directly torch light.

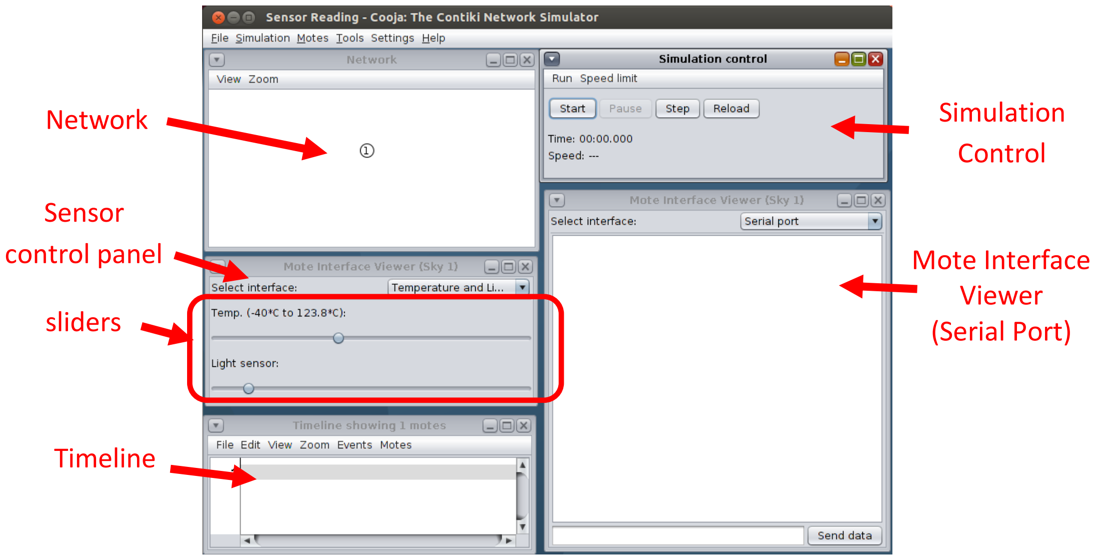
4.5 Working on XM1000 (skip this section if you cannot collect a mote)

You can find skeleton code for this lab under "/home/user/contiki-2.6/surrey/S4". In the skeleton code, a periodic process has been implemented where the process prints the number of visits to the console during each visit. You first need to find the appropriate timer setting so that the process is executed every second. You then need to implement sensor reading and print meaningful measures to the console.

4.6 Working with Cooja Simulator

You can find skeleton code for this lab under "/home/user/contiki-2.6/surrey/S4". In the skeleton code, a periodic process has been implemented where the process prints the number of visits to the console during each visit. You first need to find the appropriate timer setting so that the process is executed every second. You then need to implement sensor reading and print meaningful measures to the console.

You can find the simulation environment "cooja_sensor.csc" in the folder too. After launching Cooja and loading "cooja_sensor.csc", you should see the following window:


Sensor Reading - Cooja: The Contiki Network Simulator
File Simulation Motes Tools Settings Help
Network
View Zoom
Simulation control
Run Speed limit
Start Pause Step Reload
Time: 00:00.000
Speed: ---
Mote Interface Viewer (Sky 1)
Select interface: Serial port
Send data
Select interface: Temperature and Li...
Temp. (-40*C to 123.8*C):
Light sensor:
Timeline showing 1 motes
File Edit View Zoom Events Motes
Network
Sensor control panel
sliders
Timeline
Simulation Control
Mote Interface Viewer (Serial Port)

We have prepared the simulation environment for this experiment. The simulation environment contains the following panels:

- Network: it shows the network setup.
- **Sensor Control Panel** (new!): it provides two sliders for users to control the sensor inputs. Users can 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.
- Timeline: it shows the events happening during a simulation run.
- 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 the 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.

IMPORTANT NOTE: Please pay attention that for simulation, a customized "sky mote" is used instead of XM1000. As a result, the temperature sensor uses 12-bit ADC which gives 0.04 resolution (instead of 0.01 as in XM1000). Besides, the following function is used to read the raw data from the temperature sensor. Please check that you are using the following function in your simulation.

```
int sht11_sensor.value(SHT11_SENSOR_TEMP_SKYSIM);
```