Module: Internet of Things **2019/2020**

Lab Session 2: Hello World

Lecturers: Dr Payam Barnaghi, Dr Chuan H Foh

Demonstrators: Hamidreza Bagheri, Honglin Li, Roonak Rezvani, Narges Pourshahrokhi, Rob Pell

2.1 Introduction

During this lab session we are going to compile and run our first program on sensor hardware.

As tradition demands we are going to run a simple Hello World! program on a sensor node that outputs "Hello World". In this Session we have the following goals:

- See and understand Contiki C-programming code.
- Compile the code.
- Deploy code on a sensor node.
- Connect to the serial output of the sensor node and see the result.
- Understand the limitation of Contiki library.

2.2 Hello World! Code Example

```
1  #include "contiki.h"
2
3  #include <stdio.h> /* For printf() */
4  /*---------------------------------------------------------*/
5  PROCESS(hello_world_process, "Hello world process");
6  AUTOSTART_PROCESSES(&hello_world_process);
7  /*---------------------------------------------------------*/
8  PROCESS_THREAD(hello_world_process, ev, data)
9  {
10   PROCESS_BEGIN();
11
12   printf("Hello, world\n");
13
14   PROCESS_END();
15 }
16 /*---------------------------------------------------------*/
```

In the above listing we see our first program. To understand the code, we go through it line by line.

In L.1 we include the Contiki header files that include the Contiki OS into the compiled program and allow to access the scheduling and abstraction APIs of Contiki.

Contiki follows Protothreads (see also https://en.wikipedia.org/wiki/Protothreads), a concurrent programming model with a low-overhead.

L.3 includes the standard input/output library needed to write to the standard output.

L.5 defines the processes to be executed and included during runtime. In this case we include only one process: the Hello World process. It is possible and usually the case to define several concurrent processes, for example one for processing and collecting data, one for transmitting or receiving data.

L.6 tells the operating system which process to start on startup of the sensor node.

L.8-L.10 are the head of a function and L.14 defines the end of the process.

L.12 is where the magic happens and the String gets to the console via the printf command. Most of the Contiki commands rely on standard-C.

2.3 The Make Files

In order to run the code on the sensor node we need make files telling the compiler how to compile the programs for which specific platform. In this case we need two make files, one for telling the compiler where the Contiki headers can be found, and one defining the platform the code.

The following is the make file: "Makefile"

```
1 CONTIKI_PROJECT = hello-world
2 all: $(CONTIKI_PROJECT)
3
4 #UIP_CONF_IPV6=1
5
6 CONTIKI = ../..
7 include $(CONTIKI)/Makefile.include
```

The following is the make target: "Makefile.target" (optional)

```
1 TARGET=xm1000
```

Note that if you do not have "Makefile.target", you may provide the target in the command line during compilation.

2.4 Exercise 1

You can find "Hello World" program under "/home/user/contiki-2.6/examples/hello-world". You need the following files to compile:

- hello-world.c
- Makefile
- Makefile.target (optional)

Execute the following commands to compile, deploy and run the program.

```
$ make hello-world.upload
$ make login
```

The first command compiles and uploads the program to the connected sensor node. The second command logs in to the console of the node. After the successful login, restart the sensor node by pressing the red reset button on it and see the Hello World program running.

Note 1: To compile the program without uploading, you may use the following command:

```
$ make hello-world
```

Note 2: In case the USB device is not accessible, you may set the rights with:

```
$ chmod 777 /dev/ttyUSB0
```

Note 3: If your working directory does not have "Makefile.target", you may specify the TARGET in the command line by:

```
$ make TARGET=xm1000 hello-world.upload
$ make TARGET=xm1000 login
```

2.5 Exercise 2

Extend your code to print some random numbers between 0 and 1, and then display the sum of the random numbers.

Note: You may encounter problem printing floating point numbers.