448.063 Sensor Networks, Laboratory
Exercise 1 – Wireless Communication using Rime

Task description

The Sensor Networks Laboratory course offers practical exercises on wireless sensor networks' programming based on the Contiki operating system and MTM-5000-MSP wireless sensor nodes. The latter are TelosB clones manufactured by the Spanish company Advanticsys and embed a MSP430 microcontroller, a CC2420 radio transceiver operating in the 2.4 GHz band, as well as several sensors and LEDs. The first exercise of this laboratory course aims to get you acquainted with the Contiki operating system and with the programming of wireless sensor nodes.

Preparation. Use the provided virtual machine running Xubuntu or install the MSP430 toolchain and Contiki natively on your machine. If you use the VM, all necessary software is pre-installed and you can log in to the system using "`sensornetworks`" as both username and root password.

1. Take a careful look at the slides presented during the first tutorial and compile Contiki's hello-world example program using `sky` as target. Connect the MTM-5000-MSP sensor node via USB and upload the binary. Make sure you can correctly read the "hello world" message from serial.

   Completed ☑

2. Download the `example_floating` folder from the TeachCenter. Run this example and verify that it correctly prints increasing floating numbers on the console. Make sure your `Makefile` and `project-conf.h` files are configured as shown in the tutorial.

   Completed ☑

3. Take a careful look at the slides presented during the first tutorial and get acquainted with the Rime networking stack by understanding the examples in the `contiki\examples\rime` folder.

   Completed ☑

Understand timers Create a new folder called `example_timers` and develop a modified version of the hello-world example. Your application should print "Hello World!" every two seconds. Use `ctimers` and/or `etimers` to accomplish this goal and verify the output on the console. Make sure your `Makefile` and `project-conf.h` files are configured as shown in the tutorial.

Completed ☑

Understand buttons Extend the previous program to print "Hello World!" also whenever the user button is pressed. All three LEDs should be toggled whenever the user button is pressed as well.

Completed ☑

Collecting sensor data. Create a folder called `exercise_1` and develop a Contiki application that, after `W = 5` seconds from boot-up time, measures and displays the RSSI noise floor using the CC2420 radio, switches on the blue LED for `P = 0.5` seconds (then it turns it off again), and waits for user input. Whenever the user button is pressed, the node should measure the photosynthetic active radiation (PAR), the total solar radiation (TSR) using the S1087/S1133 photodiodes, as well as humidity and temperature using the SHT11 sensor. Temperature readings should be converted to Celsius degrees and humidity readings should be temperature-compensated and expressed in RH%. Both temperature and humidity readings should have float precision. The measured PAR and TSR readings do not need to be converted into lux, and can be kept as raw values. All sensed values should be printed on the console.

Sending data to a sink node. After the sensor data has been collected and printed on the console, this information should be sent to a sink node wirelessly. The sink accepts unicast connections on Rime channel `SINK_RIME_UNICAST_CHANNEL = 181` and uses radio channel `SINK_PHY_CHANNEL = 25`. The sink expects messages following the format shown in Figure 1, where `sequence_number` is an integer number that is incremented by 1 whenever a new measurement is sent to the sink. Location information (`x_coordinate` and `y_coordinate`) are hard-coded for this task, according to the map of the lab that can be downloaded from the TeachCenter. In the next assignments, these values will be computed at runtime. The fields `led_r`, `led_g`, `led_b`, and `reserved` should be set to `0`. All messages should be sent to the sink with a transmission power of `-10 dBm`. If your messages are correctly received by the sink, your node ID and its temperature will be displayed on the projector in the laboratory.

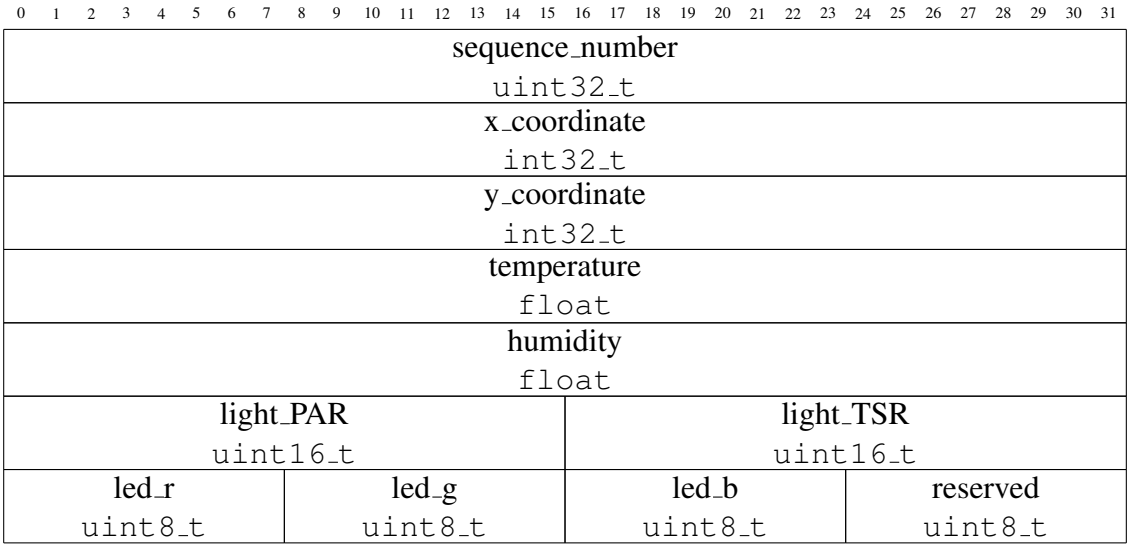

Figure 1: Structure of a unicast packet.

Replies from the sink node. The sink node replies to the received messages with an acknowledgement (ACK) on the same Rime and radio channel (i.e., on the `SINK_RIME_UNICAST_CHANNEL` and the `SINK_PHY_CHANNEL`). The ACK messages have the same format of the original message (`struct sink_message`) and the same sequence number; all other fields are set to 0. If an ACK packet does not meet these specifications it should be discarded, otherwise a message should be displayed on the console indicating the correct reception of the acknowledgement.

If no ACK is received from the sink within `T = 0.5` seconds, the same packet should be resent to the sink, i.e., no new sensor measurement needs to be taken. To give you the possibility to test if your retransmission scheme works correctly, the sink node will only reply to a message with a probability of 60%. Therefore, if you do not receive a specific ACK, it may not necessarily be due to errors in your code.

Data broadcasted from a sink node. The sink node also periodically broadcasts packets using the same `struct sink_message` with its own sensor measurements. In particular, the sink uses an identified broadcast connection on Rime channel `SINK_RIME_BROADCAST_CHANNEL = 182`. Your program should be able to receive these broadcast packets and display them correctly on the console. In addition to its own sensor measurements, the sink also broadcasts its current LED status. If the red, green, or blue LEDs of the sink are switched on, the fields `led_r`, `led_g`, and `led_b` will respectively be set to 1. If the LEDs are switched off, those fields will be set to 0. Upon reception of a broadcast packet, your program should also turn on or off the LEDs of your node with the exact same LED configuration received from the sink. Any broadcast packet received from a node that is not the sink should be discarded and a message displayed on the console accordingly.

Measuring the quality of the wireless signal. Your application should finally measure and display the received signal strength (RSSI) and the link quality indicator (LQI) of the received messages from the sink. The RSSI values of each packet should be displayed on the console in dBm.

Protocol settings and sink address. All communications should be carried out without radio duty cycling protocol and medium access control, i.e., your application should use `nullmac` and `nullrdc`. Please set your `project-conf.h` file accordingly. This is fundamental, as otherwise the sink will not be able to receive any of your packets.

There will be two different sink nodes, one in each laboratory. The sink in the Lab North has node ID `143` and its network address is set to 0 (i.e., the sink node has Rime addresses `143.0`). Please ignore sink `134.0` for this assignment.

Implementation remarks. Write any remark about your implementation in the space below:

For this assignment we implemented basic one to one communication with the sink node. Receiving broadcast message was a simple broadcast callback where we parsed the message to our packet struct and used it to read out the LED states of sink to update our own LEDs. I implemented the unicast communication and retransmission in the main thread. There I wait for a button press event and after it happens I read out all my sensors and prepare them in a struct according to the given packet scheme. Then I do the transmission repeatedly in a loop untill the sequence number is changed. This is done due to my sequence number being incremented only after reception of a correct ACK packet from sink (in the unicast receive callback). After finishing the transmission successfully the main thread again waits for the button press event.

Please submit the filled PDF and all your source files as a `zip` archive to the TeachCenter as illustrated in the tutorial within **Tuesday, 23.05.2017, 23:55 CET**.

Student ID : 1031384

Given name : Darjan

Family name : Salaj

Finalize Form