Universität
Paderborn
Computer networks
group

Winter term 2020/21

Networked Embedded Systems

# Homework 02 : Energy Consumption

Due date: 2020-12-13 23:59 CET

## 1 Introduction

Building on the previous exercises, we will now investigate the impact of firmware on a sensor node's energy consumption. To do this, we will be using the *Energest* energy monitoring framework of Contiki, which can count how much time individual components, e.g., CPU, LEDs, transmitter, receiver, etc. of a mote have been active. As an example use case, we will develop firmware that uses Pulse-Width Modulation PWM (see fig. 1) to make a LED appear lit at different levels of its brightness. Since we do not have real hardware to test the developed firmware, we have to take advantage of the simulator Cooja to confirm the correct implementation of the PWM scheme.
**To pass this homework assignment, you have to successfully submit:**

- the implemented source code & Makefile within an archive, e.g., zip.
  **and**
- a PDF document explaining details of the submitted solution showing screenshots of the correct output when the programs are executed. Please do not put the PDF document within an archive.

**Note:** Submit the homework assignment via PANDA; all team members have to confirm the submission in PANDA in order that it is counted as valid submission. The submission has to be done until the above outlined due date.


Figure 1: Pulse-Width Modulation (PWM)

## 2 PWM and Energy Consumption

1. Extract the given skeleton code found on PANDA.

2. First, we will investigate how to use *event timer* to generate timed events and build a firmware that toggles the red LED on the sensor mote every $1\,\mathrm{s}$. The documentation of *event timer* is available at our shared directory[1].

   (a) Create a process that performs the following operations in an *endless loop*:
   (1) toggle the red LED on the sensor mote;
   (2) set an *event timer* to report a timed event after $1\,\mathrm{s}$;
   (3) let the process wait for the expired event of the timer we have just set, then yield control to the operating system.
   Helpful functions: `PROCESS_WAIT_EVENT_UNTIL`, `etimer_set`, `etimer_expired`.
   Helpful constant: `CLOCK_SECOND`.

   (b) Run the firmware in the *Cooja* simulator by invoking `make energy.sim` and assert that the red LED is blinking.

3. Extend the process in item 2a that allows non-equal time intervals for the LED on and LED off durations, e.g., the LED is on for $1\,\mathrm{s}$ and off for $0.5\,\mathrm{s}$. Global variables can be used to store the on/off intervals.

4. Next, we will learn how to use PWM to control the brightness of a LED. The idea is that if we turn the LED on and off fast enough, the viewer will see the LED with a constant light output which appears dimmer instead of flashing. The brightness of the LED is then decided by the duty-cycle (the total amount of time a pulse is 'on' over the duration of the cycle). Of course that is not possible to visualize when the firmware runs in the simulator – however the correct behavior can be shown by investigating the individual LED's turn on/off times.

   We now extend the current firmware that can support making the red LED appear lit at $3$ different levels of its brightness, namely $10\,\%$, $50\,\%$, and $90\,\%$ (see fig. 1).

   (a) Calculate the intervals for the LED on and LED off durations for each of these brightness levels.
   Assume that we use the oscillating speed of $50\,\mathrm{Hz}$, or 50 times per second.

   (b) Create a second process that performs the following operations in an *endless loop*:

[1] see https://github.com/contiki-os/contiki/wiki/Timers#The_Etimer_Library

(1) let the process wait for the event that the *user button* is pressed, then yield control to the
operating system (*note*: the *user button* needs to be activated before using);
(2) when the process resumes because the button was pressed, change the time intervals for
LED on/off durations (e.g., $10\,\% \rightarrow 50\,\% \rightarrow 90\,\% \rightarrow 10\,\%$ ...).

5. Finally, we now look at the impact of firmware on a mote's energy consumption, using *Energest*[2] module.

   (a) Create a third process that uses *Energest* to collect (every second) how long the CPU and the red
   LED were active in the last second, expressed in time ticks (the default unit of Energest) and
   then outputs these two values.
   Helpful functions:

   - `energest_init(void)`: initialize the *Energest* module
   - `energest_flush(void)`: flush the time tick values for all components which are currently
     turned on
   - `unsigned long energest_type_time(int type)`: read the toal time tick value of a specific
     component type

   Helpful constants: `ENERGEST_TYPE_CPU`, `ENERGEST_TYPE_LED_RED`.

This concludes our exercises on "Energy Consumption".
For questions please visit the Tutorial Session on Thursday 09-11.

## Contact


Florian Klingler <florian.klingler@uni-paderborn.de>
Course Website https://panda.uni-paderborn.de/course/view.php?id=16255


[2] see https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-Energest