Embedded systems: lab 3

November 8, 2026

1 Installation

1.1 Using a fully configured virtual machine (recommended)

Download `INFO0064.ova`, containing a virtual machine with everything installed. In VirtualBox, select import appliance, chose the file recently downloaded and you are done! To launch the VM, power it on, then select "troubleshooting", hit enter and hit enter again twice. Then, the password is INFO0064 and you are ready for the lab.

1.2 Manual installation, tested on Linux Fedora, no guarantee on other systems

Install the following packages:

- `arm-none-eabi-binutils-cs`
- `arm-none-eabi-gcc-cs`
- `arm-none-eabi-newlib`
- `openocd`

Create a new directory called STM32F4. In there, clone the following repository: `https://github.com/libopencm3/libopencm3`, with `git clone https://github.com/libopencm3/libopencm3`. Then `cd libopencm3` and type `make`.

Then, go to a directory like `usr/lib/gcc/arm-none-eabi`. This directory should contain a single directory whose name is the version number of the package (as of 6 December 2025: 14.1.0). Add a symbolic link to it by typing `sudo ln -s 14.1.0 latest`. The result of this is that when referring to "`latest`", your system will find `14.1.0`.

Now, download `FreeRTOSv202406.04-LTS.zip` and `preemptionv2.zip`, extract these files in the directory STM32F4. Go to the directory `preemption`. Type `make`. If no error is reported, you are ready for the lab!

2 Manipulations

Foreword

The board that we will use in this lab is fragile and expensive! Please, try not to damage it. If you feel unsure of what you are doing, ask for help before it is too late.

2.1 Programming the STM32F469 discovery board

Connect the USB cable to a USB port on your machine and to the board, in the virtual machine manager, seclect USB and click on something like "connect to physical port". The name of the board, as seen from the USB port is "ST-Link". In the virtual machine, open a terminal in the directory named "preemption-stm32F469". Type `make flash`. This will compile the code in this directory and transfer it to the board. You should now see a spinning MyUliège logo on the screen. If you press the blue button, you will see two LEDs being toggled. This is because the button triggers an interrupt that turns the green LED on and releases a binary semaphore, which wakes up a task whose effect is to toggle the red LED. Let's look at the code. Open `main.c` and `utils.c`. This is a real program using an RTOS! Explain with a short pseudo-code the architecture of `main.c`.

2.2 Measuring delays

Now, we will measure the response delay to an interrupt and to a task wakening. For this, download `preemption-v2.zip`, extract it, compile and flash it onto the board as before. This program is almost the same as the previous one. The only difference is that it dupplicates signals on user accessible pins. Thanks to this, we can now measure the delays. LEDs are accessible on pins named D5 and D6 on the board. The button is not. Ask for help from the teaching assistant to measure the signal from the button.

2.3 A custom voltmeter

Now, we will use this evaluation board as a voltmeter. Caution: Only use a voltage between 0 and 3 volts! Other values will dammage the board permanently.

Download `MyUliegeVoltmeter.zip`. As before, unzip it then, compile and flash the code. On top of the screen is the average voltage measured by the ADC (analog to digital converter) of the STM32. This average is made over 100,000 successive readings. Let's look at the code. Open `main.c` and `utils.c`. Explain with a short pseudo-code the architecture of `main.c`.

The pin to which the ADC input is connected is D3. Do your readings make sense?

Now, we would like to measure the root-mean-square value of the voltage read on the ADC. How could we do that ? What are the changes to be done with respect to the current implementation ?

2.4 A custom true-RMS voltmeter

Implement the changes that you have listed above, in order to measure and display the RMS value of the voltage.

A The root-mean-square voltage

The root-mean-square of a voltage signal gives information about the power that it can transfer. It is therefore very useful to measure this value instead of the mean. The following formula gives the definition of V_{RMS}, of a signal V of period T:

$$V_{\text{RMS}} = \sqrt{\frac{1}{T}\int_0^T dt\,(V(t))^2}$$

If the period of the signal is not known, the following approximation can be used:

$$V_{\text{RMS}} \approx \lim_{T\to\infty} \sqrt{\frac{1}{T}\int_0^T dt\,(V(t))^2}$$