# Lab2

Nico Ferrari (nife1600)
Erick 'Xiao' Guan (xigu1500)

October 2017

## 1 Setting up the system

For the first part of Lab2 we need to install the development environment called Arduino on our computer [1]. Arduino IDE is a free source software and contains sketch examples. After the IDE's setup, we started to build the circuit as described in Figure 1. We used four LEDs to represent numbers from 0 to 15 in binary notation. The LEDs are connected respectively to pins 13,12,11,10 of the Arduino board and, eachone to a 330 Ohm resistor. The board is connected to a laptop with an A B USB cable (also called printer cable) for serial communication and this connection is used to power it up and to program it.

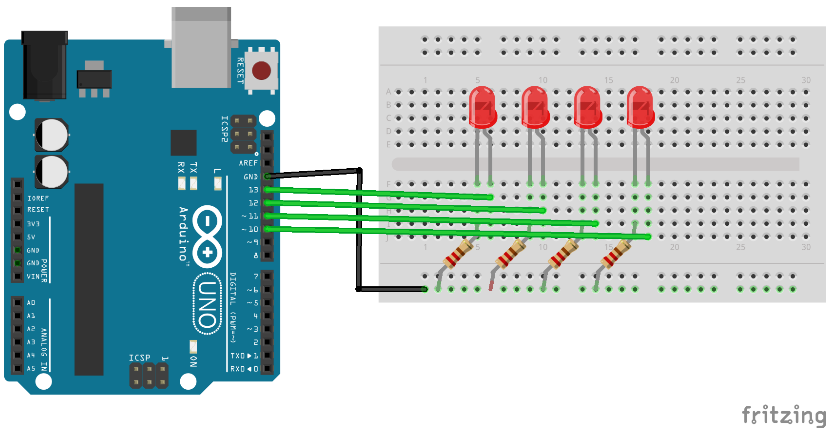

**Figure 1:** Blinking scheme

## 2 Counting on Arduino

To create a binary counter using Arduino Uno we use three special functions:

**setup routine:** we use a for loop to set each of the pins connected to the LEDs to be output pins. This routine runs once in the beginning;

**loop routine:** we use another loop to count from 0 to 15 and call for every number a function that activates the LEDs. This routine runs forever and we also introduced a fixed delay of 1 second between each repetition of the routine to make the final result visible;

**showNum function:** numbers are actualy stored in binary so we take advantage of this reading the values of the first four bits of the counter with the bitRead function and when it finds a 1, we turn on that LED on, otherwise we turn it off.

The result of this project is shown if Figure 2.

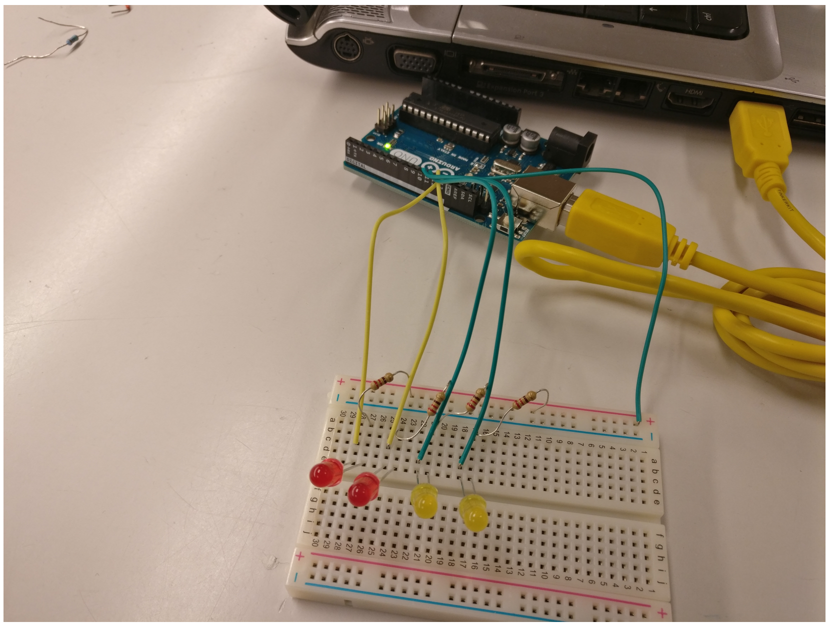

**Figure 2:** Blinking

## 3 Counting on FreeRTOS

FreeRTOS on Arduino is a port which means it has been changed to fullfill the requirements on hardware platform [2][3]. Arduino is a tiny microcontroller and the software is based on Wiring API [4]. It exposes two functions which is setup and loop. In the bottom of this, the software is still a large piece of C code. This FreeRTOS port on our hands makes no change on the interface but it uses the loop function as the idle task if there isn't one task occupying microcontroller. We set up environment as [5] described.

But FreeRTOS offers priorities, mutex, semaphores and events to the program developer. In this sense, we can make use of task scheduler to work on more problems.

Taking the problem to count from 0 to 15 by using 4 LEDs as a binary presentation, we devised our approach by using task suspense and resume. We proposed 5 tasks running at the same priority. So we have

$\mathcal{J} = \{J_c, J_1, J_2, J_3, J_4\}$. The coordinator $J_c$ is in charge of counting and supervise $J_x$ to run. Normal task $J_x$ then control its own LED and hibernate. We uses two C functions to model these two tasks. The strategy is further represented in Table 1.

**Table 1:** Strategy of task

| Coordinator $J_c$ | Normal $J_x$ $(i = 1, 2, 3, 4)$ |
|---|---|
| 1. Initiate counter $C$<br>2. Check $J_x$<br>2.a Hibernate $J_c$ if $J_x$ is not done<br>2.b Continue with step 3<br>4. Go to step 1 3. Increment $C$<br>4. Resume $J_x$ and hibernate $J_x$<br>5. Go to step 2 | 1. Check LED status<br>2. Change LED if needed<br>3. Resume $J_c$ and hibernate itself |

There are fixed delays in the coordinator to let us see the result. Our problem works flawlessly for the task. But we did observe random execution. Some LED lights off after others has lighted up.

The code is listed in `taskhandle/taskhandle.ino`.

## 4 Questions

### 4.1 Describe the task scheduler used by FreeRTOS.

The task scheduler used by FreeRTOS is a preemptive priority based scheduler that offers the option to disable preemption and use a cooperative scheduler. Tasks with equal priority are scheduled with Round Robin algorithm [6][7].

### 4.2 How many priority levels can be used in FreeRTOS and which are the higher priority?

If `configUSE_PORT_OPTIMISED_TASK_SELECTION` is set to 1,
then `configMAX_PRIORITIES` cannot be higher than 32. For our port, we have 6 priority levels [7].

### 4.3 What is your opinion about using Arduino UNO as the hardware platform when working with FreeRTOS?

It's tiny in ROM and RAM space for Ardunio so sometimes we can't compile and upload the program. For Erick's macOS, Arduino IDE failed to use linker to minimize compiled binaries. That sketch costs 33KB space. The assembly file is listed in `taskhandle/fat_binary.dump`. From the internet, someone mentioned we would only have 21% free storage space for our program [8]. Nico's machine yields smaller binary. So AVR toolchain definitely has some random behaviour over platforms. **The toolchain is not stable**!

Arduino FreeRTOS is supported as a port of FreeRTOS, it doesn't seems to be actively updated.

Despite the hardware, Arduino is quite funny to play around. Examples are abundant as well as hardware gadgets. The box is in particular beautiful.

## 5 Reference

[1] *Getting started with arduino uno,* `https://www.arduino.cc/en/Guide/ArduinoUno`, visited 2017-10-13.

[2] *Greiman/freertos,* `https://github.com/greiman/FreeRTOS-Arduino`, visited 2017-10-13.

[3] *Creating a new freertos port,* `http://web.archive.org/web/20171013120258/http://www.freertos.org/FreeRTOS-porting-guide.html`, visited 2017-10-13.

[4] *Arduino/wiring.c,* `https://github.com/arduino/Arduino/blob/b1231c39e9fa6d8602ac24305b07f074c1145633/hardware/arduino/avr/cores/arduino/wiring.c`, visited 2017-10-13.

[5] *Setting up freertos on arduino,* `http://web.archive.org/web/20171013120449/https://exploreembedded.com/wiki/Setting_Up_FreeRTOS_on_Arduino`, visited 2017-10-13.

[6] *Rtos task priority,* `http://web.archive.org/web/20171013123205/http://www.freertos.org/FAQSched.html`, visited 2017-10-13.

[7] *Rtos task priority,* `http://web.archive.org/web/20171013122708/http://www.freertos.org/RTOS-task-priority.html`, visited 2017-10-13.

[8] *Using freertos multi-tasking in arduino,* `http://web.archive.org/web/20171013124012/https://create.arduino.cc/projecthub/feilipu/using-freertos-multi-tasking-in-arduino-ebc3cc`, visited 2017-10-13.