

# Lab 11



R. Ferrero, E. Giusto  
Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)  
Torino - Italy

This work is licensed under the Creative Commons (CC BY-SA) License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/>



# Ex: Simon Game

- Implement Simon game.
- Using Key1, Key2, Int0 buttons, the player has to repeat a random sequence, displayed using LEDs 4, 5, 6.
- If the sequence inserted by the player is the same as the one to emulate, the player wins. At this point the game repeats, increasing by 1 the length of the sequence.
- Otherwise the player loses, and the game starts again from sequence length 1.

# Sequence shown by LEDs

- To show a sequence of length  $n$ , the timer has to trigger  $n*2$  interrupts, with an interval of 1,5 s.
- In interrupts 1, 3, 5, ...,  $n*2 - 1$  (odd), the interrupt handler of the timer generates a random number between 0 and 2 and powers on the corresponding LED: 0 -> led 4, 1 -> led 5, 2 -> led 6.
- In interrupts 2, 4, 6, ...,  $n*2$  (even) the interrupt handler powers off the only lit LED.

# Random Number Generation

- To obtain random numbers, you can check the value of the timer counter.
- Computing the modulo 3 of timer counter, it is possible to obtain a random number between 0 and 2.
- Suggestions:
  - use a timer different from the one used to handle the power on/off for the LEDs.
  - The timer counter can also be checked to handle the bouncing of a button.

# Outcome of the game

- If the player has pressed the buttons in the correct sequence, the 4-11 LEDs power on according to the binary number  $n$  to be displayed (LED 11 is least significant bit). The game starts again with  $n = n + 1$  as soon as the player pushes the button.
- If instead the  $i$ -th button is input wrong, on the 4-11 LEDs the  $i$  value is displayed. The game starts again with  $n = 1$  as soon as the player pushes the button.

# Suggestions

- To show the binary number  $x$  on LEDs 4-11, it is sufficient to assign to the low 8 bits of `LPC_GPIO2->FIOPIN` the value  $x$ .
- The handler of the timer has to manage their powering on with a random sequence and store that sequence in a statically allocated array (max 256 elements).
- The handler of the buttons has to verify that the pressed button corresponds to the  $i$ -th element of the sequence stored in memory.

## Suggestions (cont.)

- It is recommended to use a global variable, shared among handlers of timers and buttons, in order to keep track of the current situation of the game.
- A Finite state machine (FSM) is shown in the next slide.

# FSM

