2DT901 – Lab assignment 3

Goal for this lab:

- Getting started using the RPi Pico microcontroller and the development environment.
- Learn to program in the ARMv6-M Thumb instruction set.
- Learn to use the printf instruction to show output strings in a terminal.
- Learn to program the output pins and use them to control LEDs and a 7-segment display by using C functions from the API.
- Learn to use hardware addressed to program the hardware directly.

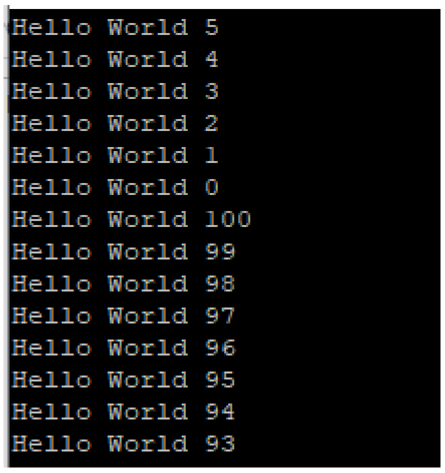
Presentation rules

1) You have to submit a report for assignment, with the LNU template.
2) You have to submit a file, either in *.docx* or *.pdf* format.
3) You are allowed to use this file to make you report, or you can use a new one. In the case you use a new file, make sure to refer to the exercises you are answering. In the case of a new file, you still have to use the LNU template.
4) The file you submit **must be renamed** as follows: <2DT901_goup_name_assignment3 >. Example: 2DT901_group2_assignment3.pdf.
5) Deadline is **2 June 2022**.
6) To pass the assignment, you must pass all the 5 tasks.
7) You have to make a short video in the presentation of task5. In the video, you are required to show that the LEDs can be turned on and off by pushing the button.

Tasks

Task 1:

- Download the source code for the Hello World program presented on page 24 in Stephen Smith's book.
- Modify the program so that the counter starts at 100, decreases by one each time the loop is repeated. Print the text Hello World followed by the counter number. When the counter reaches 0, it shall be reset to 100 and start decreasing again, in an infinite loop.
- Compile and upload the program to the Pico. Open minicom, putty or a similar terminal to show the result. Example:



```
Hello World 5
Hello World 4
Hello World 3
Hello World 2
Hello World 1
Hello World 0
Hello World 100
Hello World 99
Hello World 98
Hello World 97
Hello World 96
Hello World 95
Hello World 94
Hello World 93
```

Task 2:

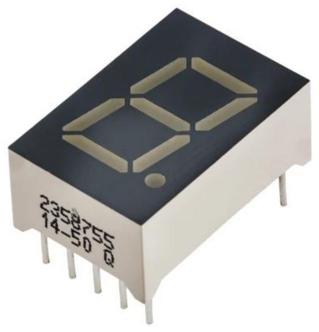
Connect one green, one yellow and one red light to three of the pins, in the same order as a traffic light. Then, write a program to make them flash like a traffic light.

Use pins GP0 for red, GP1 for yellow and GP2 for green LED! Also, make sure you understand how to connect an LED and remember to ALWAYS connect it in series to a resistor (approximately 220 Ohms)!

In this tasks, you should use C functions to control the pins. You can use the code in Listing 8-1 (page 150). Copy everything from folder Chapter 8/1/ and edit the .S file to write your own code!
Note: The functions link_gpio_put and link_gpio_set_dir are C functions, not assembly, but you don't need to care much about this, just **remember to copy the file sdklink.c**!

Task 3:

Connect a 7-segment display to the Pico. Implement a counter that counts from 0 up to 9, then reset to 0, and repeats infinitely. Make a delay of 1 second between the increments/decrements of the counter!



Use pins GP0, GP1, …, GP6 for segments A, B, …, G, respectively! You need to find a datasheet for the 7-segment display to find out how to connect it to the Pico!

Like in Task 2, you are allowed to use C functions from the Pico SDK to program the pins and for the delay function.

Task 4:

Connect a LED to GP0 and two pushbuttons: One connected to GP1 and one to GP2.

The pushbuttons look like this:

First, you have to find out how to connect the pushbutton and how to setup the port to read if it is pushed down or not!

Then, write a program with the following functionality:

If pushbutton on GP1 is down, turn on the LED. Then, let the LED stay on until it is turned off. If pushbutton on GP2 is pushed down, turn off the LED.

In this task, you should use C functions in the SDK to set up and read the pins!

Hint: You can use a C function in the RPi Pico SDK to read the input that works almost like the `gpio_put()` function. Search the SDK documentation to find out about this function. Remember to add it to the sdlink.c file because the function is an inline function!

Task 5:

Solve the same problem as in Task 5, but this time, you are not allowed to use C functions. You have to implement your own Assembly language functions to read the input from the buttons and to turn on and off the LED.

You **have to** write your own subroutines and call them to read the button and turn the LED on and off! A solution with all the code in the main loop is not okay!