Practicum 1: Accessing GPIOs

General Instructions

- Use the template (provided in Moodle) to create new projects for each task.
- Ensure that all **calculations** along with **formulas** and their **relevant parameter** necessary for the tasks are documented within the report. Make sure to list all the **values** you used for the parameters. Provide an explanation if **customized parameters** and values are chosen.
- Code has to be **clearly structured** and follow **common coding guidelines:**
  - ✓ Use **meaningful variable names** that reflect their purpose or content. Avoid single-letter variable names except for loop counters (e.g., i, j, k).
  - ✓ **Add comments** to explain complex logic, algorithms, or any part of the code that might not be immediately clear to others (or yourself in the future).
  - ✓ Use **constants** (#define or const) to define constants instead of hardcoding values. This improves code readability and makes it easier to update values later.
  - ✓ Break your code into **smaller, logical functions** to improve readability, maintainability, and reusability. Each function should ideally perform a single, well-defined task.
  - ✓ **Avoid using** "**magic numbers**" (hardcoded numeric constants) in your code. Instead, define them as named constants with descriptive names.
- Please ensure to **submit a report** as part of this exercise. Kindly adhere to the **guidelines** outlined in the provided **template** when creating the report.
- If you encounter any issues regarding the tasks, please consult the **FAQ section** in **Moodle**. If you cannot find the answer you're looking for, feel free to post your queries in the **forum** or contact us directly via email.

Useful Tips:

```
HAL_Delay(Time);
HAL_GPIO_TogglePin(Port, Pin);
```

For a detailed description of the HAL library, please refer to the function definitions in the library documentation (https://www.st.com/resource/en/user_manual/um1725-description-of-stm32f4-hal-and-lowlayer-drivers-stmicroelectronics.pdf).

Source code for the HAL library is located inside the Drivers folder (Project/Drivers/STM32H7xx_HAL_Driver/Src/*.c)

Task A: Blink an LED

In our first task, we would like to program an LED on the board and blink it according to a varying duty cycle. Create a program that manipulates a digital GPIO to generate pulses of 66.6% duty cycle for a cycle time of 3 seconds.

Requirements:

- ✓ Use **PJ7 (SEGDP)** as digital output
- ✓ **Cycle time** should be **3 seconds** and **duty cycle** should be **66.6%**

Task B: Vary the Cycle Time

Create a program that changes the frequency of a Blinking LED automatically. The blinking frequency should change automatically after the LED blinked 15 times exactly. The cycle time should switch among three levels (LOW, MID, and HIGH).

Requirements:

- ✓ Use **PJ7 (SEGDP)** as digital output
- ✓ Use **66.6%** duty cycle
- ✓ Use the following cycle time
    a. **LOW = 1000 ms**
    b. **MID = 500 ms**
    c. **HIGH = 200 ms**
- ✓ LED blink counts = **15**

Task C: Vary the Duty Cycle

Create a program that manipulates the duty cycle of the LED, while the cycle time remains fixed at 2 seconds. The change should happen in three steps with duty cycles of 70%, 50%, and 30%. The LED should blink exactly 15 times before switching the duty cycle automatically.

Requirements:

- ✓ Use **PJ7 (SEGDP)** for digital output
- ✓ Cycle time should be **2 seconds**
- ✓ Use the following duty cycles
    a. **Step 1: 70%**
    b. **Step 2: 50%**
    c. **Step 3: 30%**
- ✓ LED blink counts = **15**

Task D: Vary the Blinking Frequency with Push Buttons

In the task, we want to increase and decrease the cycle time of LED blinking based on user input. The user can use two on-board buttons (located next to the 7-Segment-Display). These buttons are connected to the pins PJ12 and PJ13, which are labelled as BTN1 and BTN2 respectively. You are required to create a program that reads the user input from BTN1 and BTN2 and changes the cycle time by 100 ms. BTN1 should be used to decrease the cycle time and BTN2 should be used to increase the cycle time. Every time a push button is released, the cycle time is increased/decreased by one step. One step is equal to 100 ms.

Requirements:

- ✓ Minimum cycle time = **100 ms**, Maximum cycle time = **1000 ms**
- ✓ Step size = **100ms**
- ✓ Duty cycle: **50%**
- ✓ Use **PJ7 (SEGDP)** for digital output
- ✓ Use **PJ12 (BTN1)** and **PJ13 (BTN2)** for digital inputs
- ✓ Check the button values to identify release operations

Task E: Display the Current Step

In this task, you are required to extend the previous task. The user would now like to have immediate feedback about the current step. The maximum cycle task has been updated to 1900 ms while the minimum cycle time stays the same as the previous task. A cycle time of 100 ms corresponds to step 0 and a cycle time of 1900 ms corresponds to step 9. Store and display the current step number on the 7-segment display attached to the board. This digital display is controlled by 8 digital pins. The pins of the 7-segment display are connected to the Pins of Port J (from 0 to 7) of our STM32 microcontroller as shown in the figure below.

Requirements:

- ✓ Use **bit manipulation** as explained in the lecture.
- ✓ Do not set every bit individually for each number!
- ✓ Create a **method** for displaying a number with a reasonable parameter list (**Reusability**)
- ✓ Minimum cycle time = **100 ms**, Maximum cycle time = **1900 ms,** Step size = **200 ms**
- ✓ Use **PJ7 (SEGDP)** as digital output (connected to DP)
- ✓ Use **PJ12 (BTN1)** and **PJ13 (BTN2)** for digital inputs

GPIO settings for 7-segment display:

- ✓ PJ0 (SEGA) connected to A
- ✓ PJ1 (SEGB) connected to B
- ✓ PJ2 (SEGC) connected to C
- ✓ PJ3 (SEGD) connected to D
- ✓ PJ4 (SEGE) connected to E
- ✓ PJ5 (SEGF) connected to F
- ✓ PJ6 (SEGG) connected to G