**Academy Task 1**

**Basic Architecture**

*Low Voltage*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Debouncing Logic Description

The push-button connected to PC14 may generate mechanical noise (bounces)

when pressed or released. To avoid false triggers, a software debouncing

algorithm is used. The algorithm works as follows:

1) Read the current state of the button from PC14.

2) If the current reading differs from the last stable state, reset a timer

using HAL GetTick().

3) If the button state remains unchanged for more than DEBOUNCE DELAY

(e.g., 50 ms), accept the new state as valid.

4) When a valid press is detected (button stable and pressed), toggle the

LED on PC13 using HAL GPIO TogglePin().

This method avoids blocking delays by relying on the system tick timer,

ensuring that the microcontroller can continue executing other tasks while

waiting for the button state to stabilize.

[GitHub](https://github.com/Yahia-Sameh/Task_2)

[Video](https://youtu.be/gApH3puFDQ4)