**Academy Task 2**

**ARM**

*Low Voltage*

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

# DEBOUNCING

## What is bouncing

Switch bounce is the rapid, unintended connecting and breaking of contact that occurs when a mechanical switch is actuated. Instead of a single clean transition from open to closed (or vice versa), because when the two contact plates of a push button connect the vibrate unintentionally as they hit each other so bouncing occurs

## How to prevent it

There are a lot of methods to prevent it the easiest and the most straightforward way to debounce a button it to wait between each signal read and another by adding delay using HAL\_Delay() function found in page 50 or we can solve it via hardware methods like using an RC low-pass + Schmitt trigger or via a software like

State-machine approach:

This means that we continuously read the button but only accept a new state when it reads a set of consecutive reads and they all return the sane state

Timer-based debouncing:

Which is basically the same idea as the above but it triggers when there is a change in state and count for how many milliseconds will it stays if it stabilize then it change the state to the new stable state

Software low-pass filter:

Lastly software low-pass filter and this the one used in my code ; it relays on a weighted mean approach to measure if the change is effective and stable or not using and comapare the result to a threshold using this formula (xk​=α⋅xk−1​+(1−α)⋅uk​) : state=state\*0.7f+currentState\*0.3f;

Links

Drive : <https://drive.google.com/drive/folders/1L80MlBoGXIevwYfPC8e-d5KvjPAvsK6q?usp=drive_link>

Git hub: