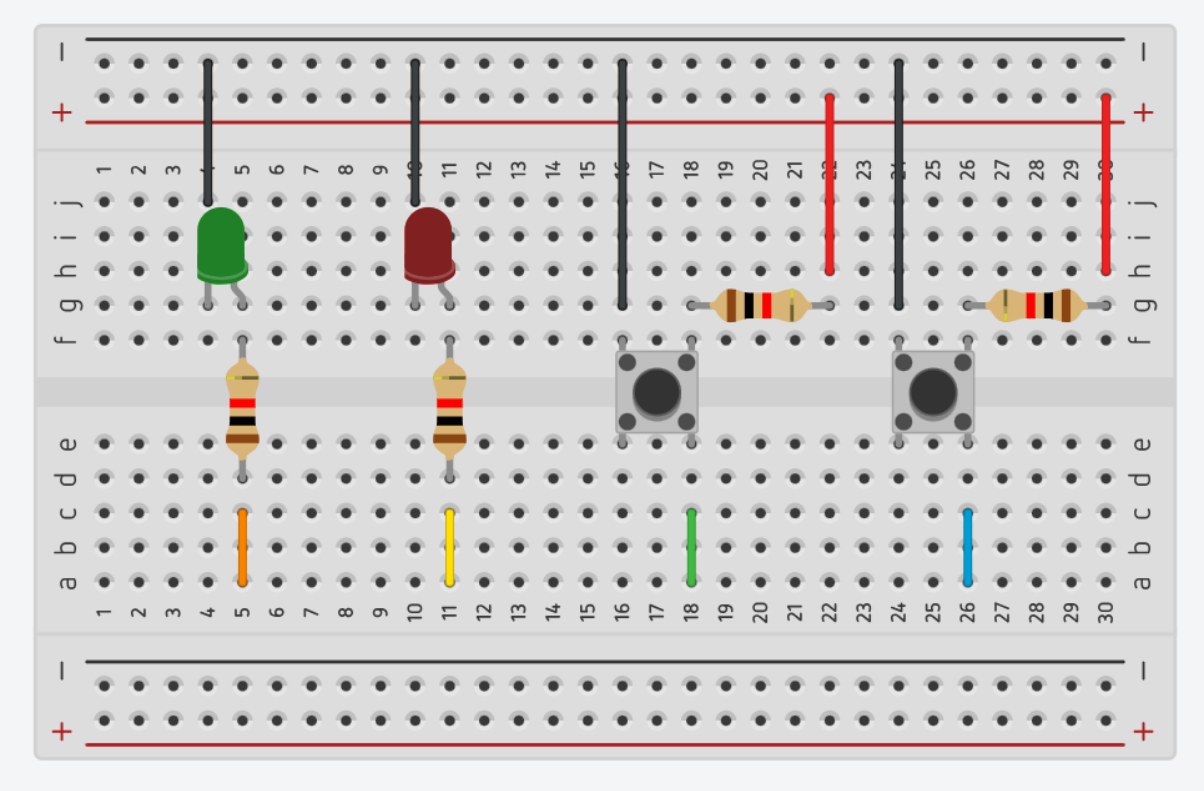
ES LAB 3: Interrupts.



Circuit used: red wires are to simulate power to the component that is connected to the 5V pin. The black wires are to simulate being connected to the GND pin. The coloured wires are to simulate the signals pins. These are pins I have chosen and will conduct my research upon.

Student: Johnson Domacasse
Date: 26 sept 2023
Student#: 4471709
Teacher: Suzana Andova

Introduction:

Interrupts are situations where the current execution of a regular program is suspended and the processor is forced to take care of “other” important events. You can think of it like a delay function that has priority over everything else. For this Lab I will be using the above shows circuit diagram to perform my research. The goal is to give myself a better understanding of interrupts and how I can use the external interrupts (EXTI).

Procedure:

Setup:

COMPONENT	PIN
BUTTON_0 (B0)	PA0
BUTTON_1 (B1)	PA1
LED_0	PA8
LED_1	PA10

Table 1: Hardware Pins

For this research I have my circuit wired as seen on the front page. Each of the differently coloured wires (i.e. not red or black) are corresponding to a certain GPIO pin. In table 1 you can find the pins that are specific to their components. These are chosen using the STM32 user manual. Like the previous LAB I will try to keep the power consumption low by only using 1 clock for this research.

You may have noticed I haven’t used the same pin number for both buttons, this is because the way external interrupt handling is set in place requires me to use different pin numbers. This is due to the multiplexers only being able to accept one of the GPIO pins to put on its external interrupt **line.**

Design choices:

Based off of the requirements that were given to me, I came up with the following key conclusions:

- The program must have an interrupt handler to deal with the buttons at the highest priority, since button press checking can’t be done with polling.
- Since the interrupts need to be short, I will have two separate interrupts. Each to handle their specific buttons. I will implement the LEDs in the main loop.
- I will try to debounce my buttons to avoid noise activations in the program.
- I will try to add my setup code into functions like the previous LAB.
- I will use HAL_GetTick in my handlers to make use of the debouncing and the calculation of the short and long presses.
- When I detect a button press and it’s in between a short and long press I will count it as a short press.
- When I detect a button press and it’s longer than a long press I will count it as a long press.

In the end the final design will have 2 interrupt handlers, 1 for each button and the code for the LED I will do in the main loop. I will also debounce the buttons within the handlers. This keeps my handlers short.

Implementation:

I set up a simple system configuration function by setting up the clock and details of my used components. This was tested by implementing some old code used in the previous LABs. After confirming that it worked fine, it was on to the next part of my implementation.

For the next part, I will be preparing the system to be able to handle interrupts. Like before it is mostly set up using information from the reference manual, slides and videos that were provided. I already have my clock enabled, so we first enable the interrupt, set which pins we want as our source for the external interrupt, set the trigger selection and finally enabling that line as the external interrupt.

Before I do it for the second button as well, I want to test a simple blinking program to check it works correctly using the handler. It worked correctly. I noticed that my button isn't debounced so I need to do that as well. After adding the debounce to the handler, the behaviour is more or less what I have in mind. I did the same for both of my handlers.

Lastly it was a matter of implementing my solution to trigger a short or a long press for each button respectively. Again this solution was based on the above mentioned design choices.

Results:

In my first implementation, my results were not completely correct. For the first button, you would need to double short press for the LED to turn on. The long press worked flawlessly for both buttons. I think this is an issue with debouncing and couldn't find a fix for it.

These are the results for the first handler:

- A double short press will turn it on
- A long press will turn it off
- On some occasions the short press can turn the LED off.

These are the results for the second handler:

- A long press will turn the LED ON
- A double short press will turn it off
- On some occasions the short press will turn the LED ON.

These results are the complete inverse of eachother, which is understandable considering the code used for both of them are of similar syntax.

Issues:

As I was making this assignment, I was initially using pin number 0 and 2 on port A for my buttons. I ran into a big issue when I realized that my second interrupt (button on PA2) would not trigger. I tried debugging my code and found no issues, I checked the wiring and found no issues. I tried making new projects with the same code and found no issues. I even tried changing the pin to pin 3 on the same port but ran into the same issue. So I was at a loss for what the problem was. At school Mohammed and I tried doing the same step by step guide, we both kept at it for 2 hours and found no issues. Then we borrowed a multi-meter and noticed that the pins (both 2 and 3) were not giving the right amount of voltage. We looked into the reference manual and found the those 2 pins specifically have priority when we are working with a Uart connection. This caused that my interrupts were never triggered in the first place. Because of this, we ended up changing the pin to 1 on the same port and the second interrupt finally started triggering.