Mark	/11

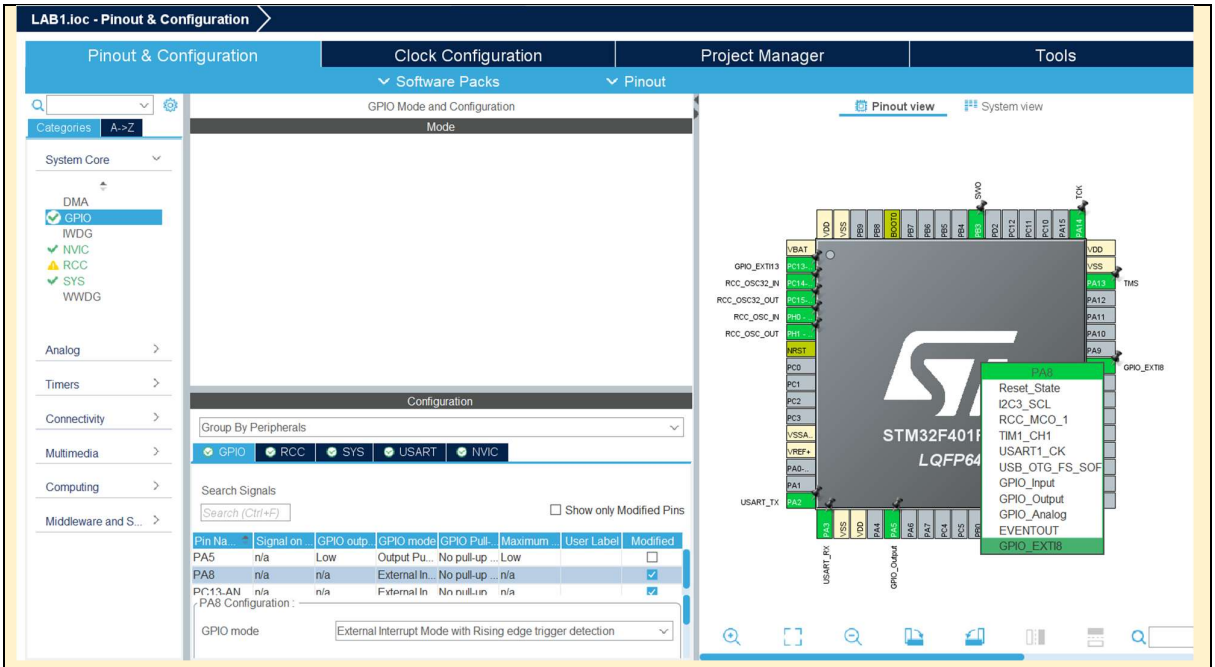
Team name:	*A14*		
Homework number:	*HOMEWORK 02*		
Due date:	28/09/25		
Contribution	NO	Partial	Full
Mattia Di Mauro			*x*
Francesca Biondi			*x*
Lorenzo Castelli			*x*
Pietro Albrigi			*x*
Notes: none			

Project name	HOMEWORK 2		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

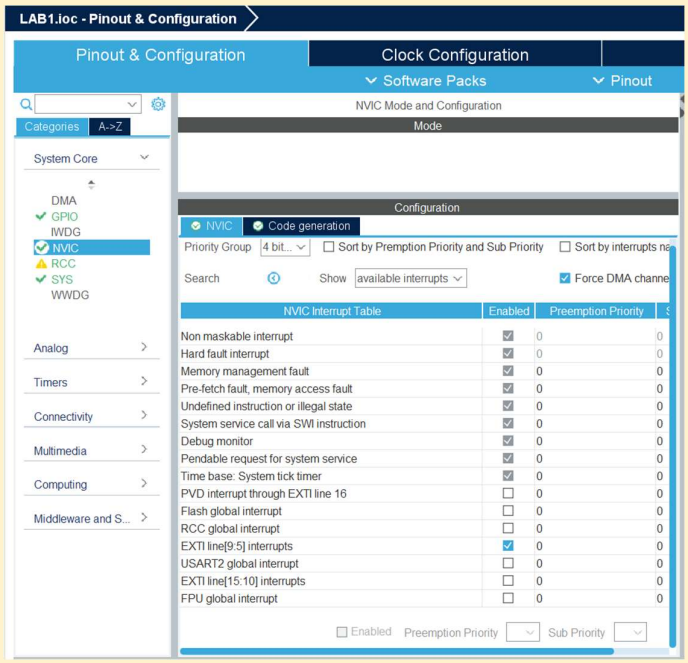
Part 1a:

In the "Hands-on Lab Schematics.pdf" document we found that the microphone (depicted on page 7, SND_IN) is connected to the STM32's pin PA8 (as shown on page 4).

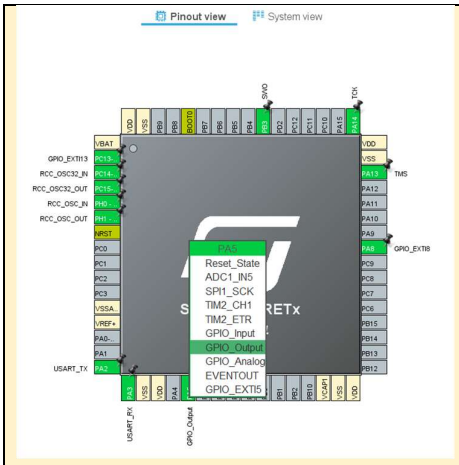
We therefore configured pin PA8 as GPIO_EXTI8, setting the GPIO mode to "External Interrupt Mode with Rising edge trigger detection" to generate an interrupt when the microphone's voltage signal transitions from LOW to HIGH. Our microphone is connected to a comparator (as described in the slides on "ARM microcontrollers"), so it sends a digital pulse whenever the sound exceeds a certain threshold, allowing the microcontroller to generate an interrupt when a snapping sound occurs.



We also enabled the EXTI interrupt (EXTI lines [9:5] interrupts, since the microphone is on pin PA8):



We then configured pin PA5 (corresponding to the green LED) as GPIO_output:



Lastly, we wrote the following code in the “main.c”, outside of the “int main”:

```
/* USER CODE BEGIN 0 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin==GPIO_PIN_8)
    {
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
    }
}
/* USER CODE END 0 */
```

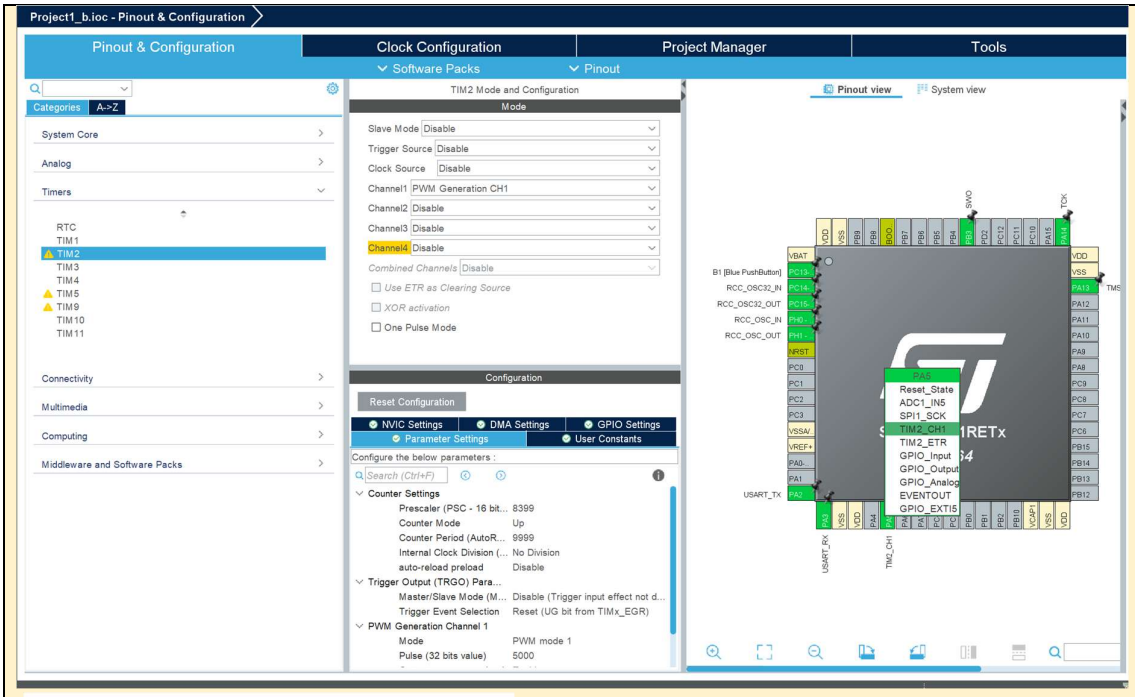
The “if” clause ensures that if the interrupt is generated by the pin connected to the microphone, the state of the LED pin is toggled. The code was then tested on the board and confirmed to work as intended.

We observed that the LED does not always respond correctly to finger snaps when they occur in very quick succession, likely because the microphone and comparator module may take some time to reset after a snap, so it does not always produce a clean pulse for the next one.

Part 1b:

We configure PA5, corresponding to the LD2 green LED, as TIM2_CH1, which is the alternate function connected to the timer.

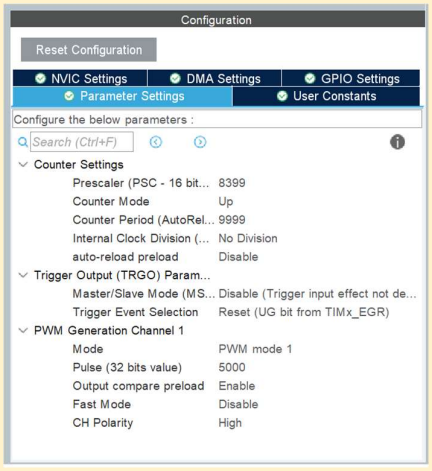
In the “Timers” section of CubeMX, we set TIM2 Channel 1 to PWM Generation CH1. Then, in the TIM2 configuration, we adjust the Prescaler and Counter Period (ARR) values to obtain a 1 Hz PWM frequency, so that the LED blinks once per second.



$$f_{PWM} = \frac{f_{TIM}}{(ARR+1)\cdot(PSC+1)}$$

Knowing that the System Clock (SYSCLK) is 84 MHz, we choose Prescaler = 8399 and Counter Period = 9999.

We select a duty cycle of about 50% by setting Pulse = 5000, which means the LED is on for half of the cycle and off for the other half. This combination produces a timer frequency low enough that the LED blink is visible to the human eye.



Finally, in main.c, we call the following function to start the PWM on TIM2 Channel 1:

```
92    /* Initialize all configured peripherals */
93    MX_GPIO_Init();
94    MX_USART2_UART_Init();
95    MX_TIM2_Init();
96    /* USER CODE BEGIN 2 */
97    HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
98    /* USER CODE END 2 */
```

The code was then tested on the board and confirmed to work as intended.

Professor comments: