

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 Albrigì			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.			

LAB1.ioc - Pinout & Configuration

Pinout & Configuration

GPIO Mode and Configuration

Pin Na...	Signal on	GPIO out	GPIO mode	GPIO Pull	Maximum	User Label	Modified
PA5	n/a	Low	Output Pu...	No pull-up ...	Low		<input type="checkbox"/>
PA8	n/a	n/a	External In...	No pull-up ...	n/a		<input checked="" type="checkbox"/>
PC13_AN	n/a	n/a	External In...	No pull-up ...	n/a		<input checked="" type="checkbox"/>

GPIO mode: External Interrupt Mode with Rising edge trigger detection

Pinout view

STM32F401 LQFP64

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

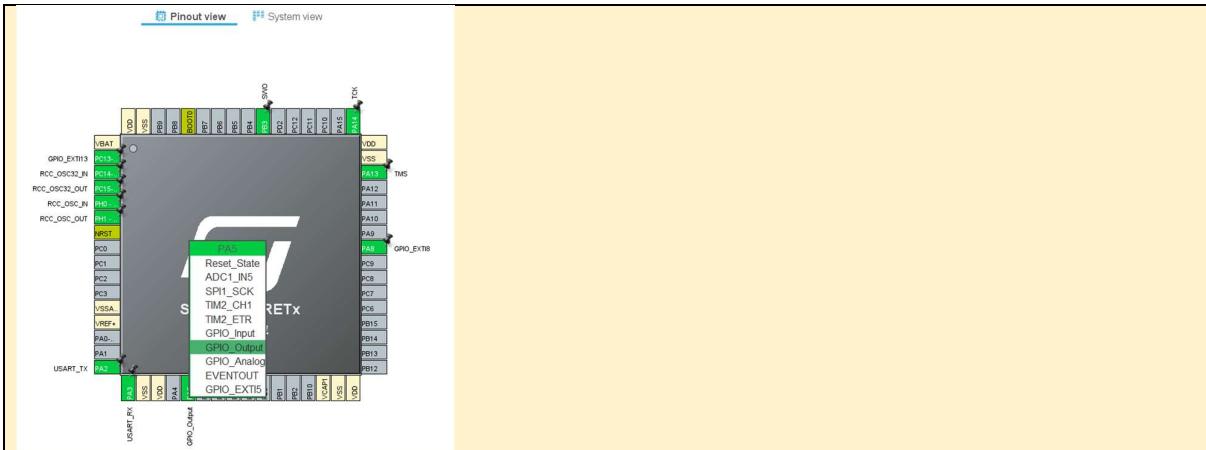
LAB1.ioc - Pinout & Configuration

NVIC Mode and Configuration

NVIC Interrupt Table

Interrupt	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base, System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
EXTI line[9:5] interrupts	<input checked="" type="checkbox"/>	0	0
USART2 global interrupt	<input type="checkbox"/>	0	0
EXTI line[15:10] interrupts	<input type="checkbox"/>	0	0
FPU global interrupt	<input type="checkbox"/>	0	0

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”:

```

58 /* USER CODE BEGIN 0 */
59 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
60 {
61     if(GPIO_Pin==GPIO_PIN_8)
62     {
63         HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
64     }
65 }
66 /* 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.

Project1_b.ioc - Pinout & Configuration

Pinout & Configuration Clock Configuration Project Manager Tools

Categories A-Z

System Core Analog Timers

RTC TIM1 TIM2 TIM3 TIM4 TIM5 TIM6 TIM9 TIM10 TIM11

Connectivity Multimedia Computing Middleware and Software Packs

Clock Configuration

Mode

Slave Mode: Disable Trigger Source: Disable Clock Source: Disable

Channel1: PWM Generation CH1 Channel2: Disable Channel3: Disable Channel4: Disable

Combined Channels: Disable Use ETR as Clearing Source XOR activation One Pulse Mode

Configuration

Reset Configuration

NVIC Settings DMA Settings GPIO Settings Parameter Settings User Constants

Configure the below parameters:

Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bits...): 8399 Counter Mode: Up Counter Period (AutoR...): 9999
Internal Clock Division (...): No Division auto-reload preload: Disable

Trigger Output (TRGO) Parameters

Master/Slave Mode (M...): Disable (Trigger input effect not de...)
Trigger Event Selection: Reset (UG bit from TIMx_EGR)

PWM Generation Channel 1

Mode: PWM mode 1 Pulse (32 bits value): 5000

Pinout view **System view**

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

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.

Configuration

Reset Configuration

NVIC Settings DMA Settings GPIO Settings Parameter Settings User Constants

Configure the below parameters:

Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bits...): 8399 Counter Mode: Up Counter Period (AutoR...): 9999
Internal Clock Division (...): No Division auto-reload preload: Disable

Trigger Output (TRGO) Parameters

Master/Slave Mode (M...): Disable (Trigger input effect not de...)
Trigger Event Selection: Reset (UG bit from TIMx_EGR)

PWM Generation Channel 1

Mode: PWM mode 1 Pulse (32 bits value): 5000
Output compare preload: Enable
Fast Mode: Disable
CH Polarity: High

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: