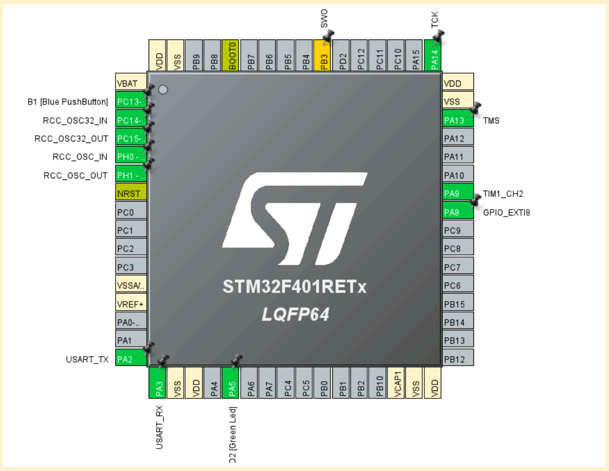
Mark	1/11

Team name:	*B5*		
Homework number:	*HOMEWORK 03*		
Due date:	08/10/24		
Contribution	NO	Partial	Full
Marenghi Manuela			*x*
Fellegara Tommaso			*x*
Giammusso Samuele			*x*
Cattani Luca			*x*
Csata Dániel			*x*
Notes: none			

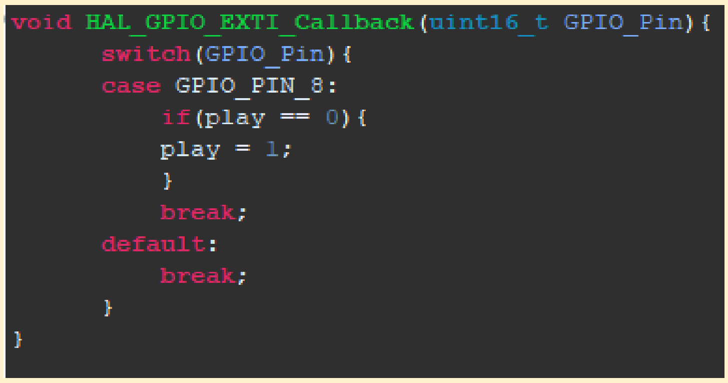
Project name	Test		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Part 1 - play a song, with Hal Delay

Using the information from the previous homework we set up the microphone interrupt (PA8 pin) and we also set up the speaker pin (PA9) .

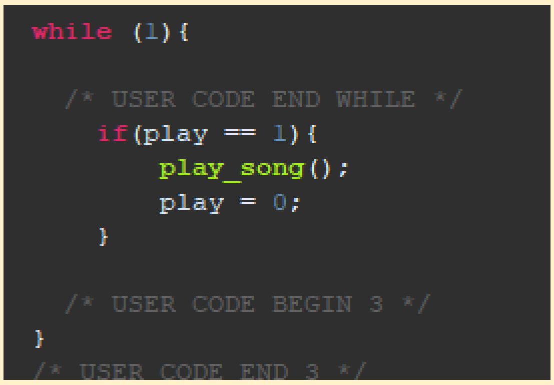


We modify the interrupt service routine to set a variable true upon the snap of the fingers.



```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
	switch(GPIO_Pin){
	case GPIO_PIN_8:
		if(play == 0){
		play = 1;
		}
		break;
	default:
		break;
	}
}
```

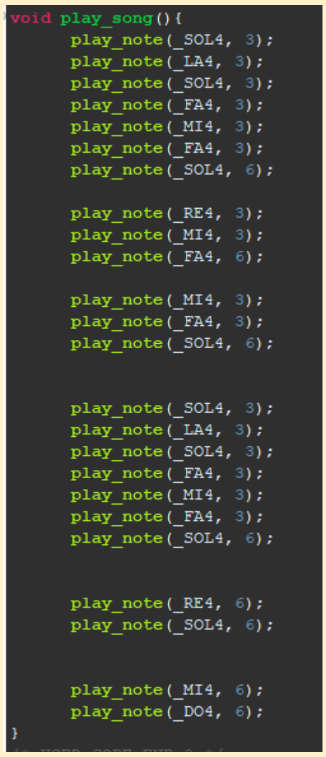
If it has already been set that means that the song is already playing so we should not replay it. Otherwise we set it to be true and we check the value in the main function.



```
 while (1){

   /* USER CODE END WHILE */
	if(play == 1){
		play_song();
		play = 0;
	}

   /* USER CODE BEGIN 3 */
 }
 /* USER CODE END 3 */
```

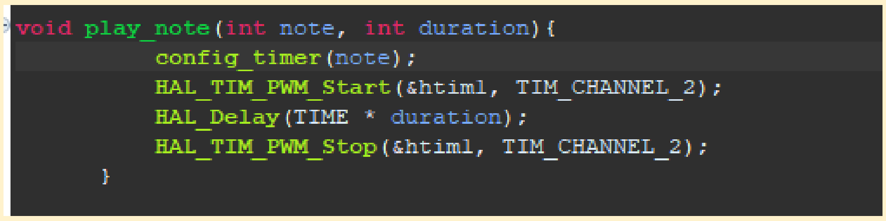
After the song has been played we reset the variable.

To play a song we play each note for the given time.



```
void play_song(){
	play_note(_SOL4, 3);
	play_note(_LA4, 3);
	play_note(_SOL4, 3);
	play_note(_FA4, 3);
	play_note(_MI4, 3);
	play_note(_FA4, 3);
	play_note(_SOL4, 6);

	play_note(_RE4, 3);
	play_note(_MI4, 3);
	play_note(_FA4, 6);

	play_note(_MI4, 3);
	play_note(_FA4, 3);
	play_note(_SOL4, 6);


	play_note(_SOL4, 3);
	play_note(_LA4, 3);
	play_note(_SOL4, 3);
	play_note(_FA4, 3);
	play_note(_MI4, 3);
	play_note(_FA4, 3);
	play_note(_SOL4, 6);


	play_note(_RE4, 6);
	play_note(_SOL4, 6);


	play_note(_MI4, 6);
	play_note(_DO4, 6);
}
```

To play a note we configure the PWM signal to the frequency of the given note, then start the PWM timer, wait the given time and stop the timer.


```
void play_note(int note, int duration){
          config_timer(note);
          HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
          HAL_Delay(TIME * duration);
          HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_2);
      }
```

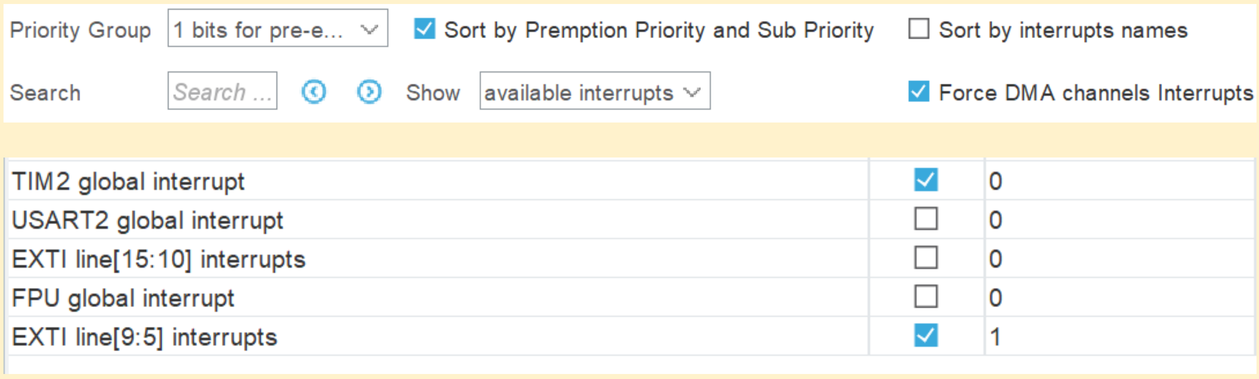
To configure the PWM signal for different frequencies we use the same function as the one which is generated to initialize the timer peripheral.

Part 2 - play a song, without Hal Delay

To have the same behavior but withouting using the HAL_DELAY function, we used a timer.

First of all we set up the timer 2 with channel 1 (as Output compare CH1) in the configuration view.

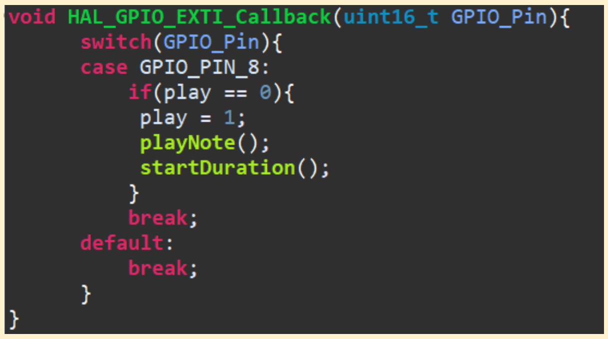
We modify the group priority to 1 bit and then we set the preemption priority of the microphone to 1 (EXTI line 9:5) and also enable the TIM 2 global interrupt.


Priority Group: 1 bits for pre-e... ☑ Sort by Premption Priority and Sub Priority ☐ Sort by interrupts names

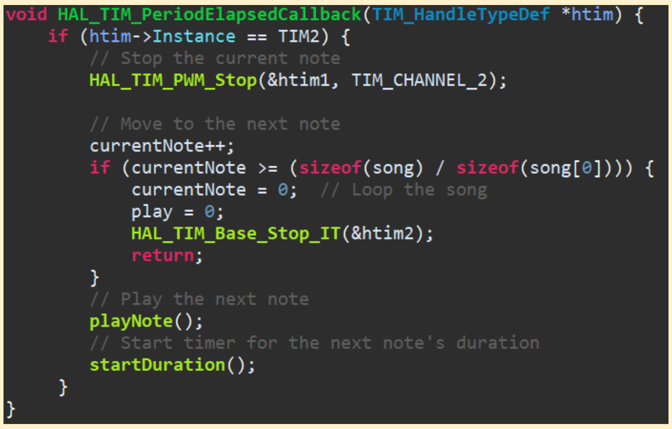
Search: Search ... Show: available interrupts ☑ Force DMA channels Interrupts

TIM2 global interrupt	✓	0
USART2 global interrupt	☐	0
EXTI line[15:10] interrupts	☐	0
FPU global interrupt	☐	0
EXTI line[9:5] interrupts	✓	1

In the code we added the handle of the interrupt of the microphone. When a loud noise is heard, the interrupt handler executes the playNote function to play a note and then the timer 2 is started in the startDuration function.


```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
      switch(GPIO_Pin){
      case GPIO_PIN_8:
          if(play == 0){
           play = 1;
           playNote();
           startDuration();
          }
          break;
      default:
          break;
      }
}
```

When the timer 2 generates the interrupt, since it has a higher priority the HAL_TIM_PeriodElapsedCallback function is executed:
- firstly the current note is stopped,
- then the index move to the next note to play,
- then there is a check to verify that the song is not ended yet,
- then play the next note and restart the timer 2.



```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim->Instance == TIM2) {
        // Stop the current note
        HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_2);

        // Move to the next note
        currentNote++;
        if (currentNote >= (sizeof(song) / sizeof(song[0]))) {
            currentNote = 0;  // Loop the song
            play = 0;
            HAL_TIM_Base_Stop_IT(&htim2);
            return;
        }
        // Play the next note
        playNote();
        // Start timer for the next note's duration
        startDuration();
    }
}
```

The song goes on until the last note is played, moreover the variables currentNote and play are set to 0 and the function HAL_TIM_BASE_Stop_IT is called to stop the timer 2 to send interrupts.

The playNote() function was created by copying the timer 1 initialization function, and by modifying the period and the pulse values according to the current note playing.
Also in the end the function `HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);` is called to start the PWM generator

The startDuration() function was created by copying the timer 2 initialization function, and by modifying the period value according to the current note playing.
Also in the end the function `HAL_TIM_Base_Start_IT(&htim2);` is called to start the timer and enable the interrupt generation

Professor comments:

Your function play_song is not easily scalable.
Better to use indexes and for cycles