|  |  |
| --- | --- |
| **Mark** | **B** |

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B5* | | |
| Homework number: | *2* | | |
| Due date: | Tuesday, October 10, 8:30 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Ghidini Alessandro |  |  | **✓** |
| Latino Francesco |  |  | **✓** |
| Luppi Eleonora |  |  | **✓** |
| Bravin Riccardo |  |  | **✓** |
| Feltrin Elia |  |  | **✓** |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | ***Play a song*** | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  This document describes two projects related to playing a song using a speaker with PWM generation. The first project involves playing a song with a delay function and snapping fingers as an interrupt. The second project aims to play a song without using the delay function. The document provides detailed instructions and code snippets for each project.  **Part 2a:**  In this project we want to play a song using the speaker with PWM generation. First of all we need to set the pins in the GUI. The pin of the speaker PWM\_SPKR is connected to PA9 so we set the latter as TIM\_1\_CHANNEL\_2. Therefore, in the timer tab we select the clock source as internal clock and the channel 2 as PWM generation. The correct frequency will be modified according to each note playing. In fact for each note we define a value of the Auto-Reload Register (ARR) as #define DO 3205 and we create a struct for the song that contains for each note both the ARR and the duration expressed as a ration of the TIME of one note which is defined as #define TEMPO 75 (75 ms).  Now we want to create a function that modifies each parameter of the timer since we need to set the timer at the right time for each note. In the code we double click and open declaration on the initialization of the timer. The function is *void playnote(struct note note\_playing)* which takes in input the struct of one note. In the function we copy and past the inizialization of the timer and we set *htim1.Init.Period = note\_playing.tone* in order to set the ARR and play the correct note and *sConfigOC.Pulse = note\_playing.tone/2* in order to have 50% of duty cycle (DC). After setting the correct parameters for the timer we start the PWM with *HAL\_TIM\_PWM\_Start(&htim1, TIM\_CHANNEL\_2).* We use the delay HAL\_Delay(note\_playing.duration\*TEMPO) which waits for the duration of the note before stopping *HAL\_TIM\_PWM\_Stop(&htim1, TIM\_CHANNEL\_2)*.  Finally the function void playsong() iterate on each note in the struct of the song and call playnote() for each one of them.  Now we want to do the same project but the song has to start once we snap a finger. So in this case we set as before the pin of the speaker. Moreover, the pin of the microphone is connected to PA8 so in the GUI we set PA8 as GPIOEXTI8. In the NVIC tab we enable RCC global interrupt and in the GPIO mode we set only on the rising edge since the interrupt must be called just when we snap the finger.  Now we generate code and in the file stm32f4xx\_it.c we open declaration of *HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_8)* and we copy in the main *void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin).* In this function with a switch case construct we use the detection of the finger snap as interrupt callback and we set a flag called *snapFinger* equal to 1 each time a sound is detected by the microphone. This flag is used in the while(1) with the if construct to check and playsong only when the sound is detected. The same flag is set to 0 after the song finishes.  **Part 2b:**  In this case we set PA9 as GPIO\_Output since we want to create a “manual” PWM without using the Delay function. We use the timer TIM 3 so we set in the timer tab TIM 3 with clock source internal clock and the AutoReaload register 65535.  We generate the code. In this case we use the function *HAL\_TIM\_Base\_Start\_IT(&htim3)* which starts the TIM Base generation in interrupt mode. In the file stm32f4xx\_it.c we write in the function *void TIM3\_IRQHandler(void).* In particular here we want to generate a square wave in the pin we previously associated as GPIO\_output, that is PA9. We use the function *HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_9)* and we increment a counter *t*. This counter is defined both in the file *stm32f4xx\_it.c* as an external variable **extern** **long** **int** t and in the main where it is also initialized as zero **long** **int** t=0. We use this variable to count the number of cycle of the generated timer. By doing this, we can modify the function *playsong()* in a way that the function playnote is called for each note and it waits until t does not reach the number of cycle of that note. After that the counter is set equal to zero again and it is ready to count for the following note. The cycle of each note is computed as the frequency of the interrupt multiplied by the duration of each note as *(TEMPO\*score[i].duration)\*(F\_OSC/((1+score[i].tone/2)\*(100))))*. In this case of implementation, we have that our interrupt has a frequency which is the double of the note’s one, so we need to divide *score[i].tone* by 2 to obtain the right value for the *htim3.Init.Period* register. Lastly, in the function playsong, we stop the timer generation with the function *HAL\_TIM\_Base\_Stop\_IT(&htim3).*  To obtain the snap finger effect, all we have to do is enabe pin PA8 as GPIOEXTI8 from the .ioc file and use a flag in the same way of the previous code. In this case we use an if construct to control that the speaker has already sensed a loud sound by setting the snapfinger flag to 1. Then, in the if structure we intialize the interrupt mode through *HAL\_TIM\_Base\_Start\_IT(&htim3)* and we call *playsong().* After the song has been played, the interrupt generation is stopped through *HAL\_TIM\_Base\_Stop\_IT(&htim3)* (already in the play function) and the flag is set to 0. When the microphone senses a new sound, *snapfinger* will be set again to 1 and the song will be played again. | | | |
| Professor comments:  Part A: “In the function we copy and paste the inizialization of the timer” 🡪 it is more convenient to use the functons \_HAL\_TIM\_SET\_AUTORELOAD and \_HAL\_TIM\_SET\_COMPARE.  Part 2B:  “In particular here we want to generate a square wave in the pin we previously associated as GPIO\_output, that is PA9. We use the function *HAL\_GPIO\_TogglePin(GPIOA, GPIO\_PIN\_9)* and we increment a counter *t*.” Why don’t you use PWM also in this project?  “This counter is defined both in the file *stm32f4xx\_it.c” I suggest never to modify this file, but just redefine the callbacks (which are weakly defined).* | | | |