|  |  |
| --- | --- |
| **Mark** | **A** |

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B5* | | |
| Homework number: | *01* | | |
| Due date: | Tuesday, October 03, at 8:30 am | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Ghidini Alessandro |  |  | **✓** |
| Latino Francesco |  |  | **✓** |
| Luppi Eleonora |  |  | **✓** |
| Bravin Riccardo |  |  | **✓** |
| Feltrin Elia |  |  | **✓** |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Prepare the basics for the project “Play a song” | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | **✓** |
| Explanation:  For the first part we had to setup the microphone output to trigger an interrupt that would toggle the led (LD2) on the board while for the second one the timer needed to be set up such that a PWM signal at 1Hz and 50% duty cycle was generated on the board led (LD2) lane. In the next two sections a more detailed explanation is presented.  **Part 1a:**  From the PCB schematics we have found that the microphone is conveyed through the line SND\_IN to the pin PA8 of the microprocessor. The microphone produces voltage peaks when it detects sounds with an energy greater than a specific threshold. To detect these voltage peaks, we have used an interrupt service routine: the instructions that make the LED switched on and off are executed only when the event (sound with a proper energy) is detected by the microphone. From the graphical IDE we have set the pin PA8 as GPIO\_EXTI8 so we have enabled the interrupt on the pin.  Since we want to maintain the status of the led after the event, we have set the pin dedicated to generating the ISR as sensible only to raising edge of the potential. We have selected in the section “System Core” the voice “GPIO” and modified the GPIO mode by selecting “External Interrupt Mode with Raising Edge”.  The ISR must be enabled at the level of the microprocessor so, from the section “System Core”>“NVIC”, we have allowed the generation of ISR on the EXTI line [9:5].  Now we must write the code of the ISR: this function is by default defined as a weak function, so we need to overwrite the associated code in the main to make ISR no more weak and executed with the assigned priority. From the file stm32f4xx\_it.c we select the function “void EXTI9\_5\_IRQHandler(void)” and open the declaration of the function “HAL\_GPIO\_EXTI\_IRQHandler” by clicking twice on it (or right click). We must copy the structure of this function in the main in the section “user code” to write the associated code and overwrite the ISR. To detect the event, we use a switch-case construct using as condition a generic GPIO\_Pin: when the event happens, the pin associated to the microphone is activated and we run the toggle the status associated to the LD2 (PA5).  Additional:  In the interrupt function (HAL\_GPIO\_EXTI\_Callback) we created a static variable which kept count of the last callback on GPIO\_PIN\_8, then we toggled the led only if 50ms had passed from the latest call. This way the flickering caused by multiple concatenated interrupts was reduced if not completely removed.  **Part 1b:**  In the timer settings from the graphical interface the timer TIM2 was activated with the internal clock as source, and with PWM generation on CHANNEL\_1. The prescaler field, which sets a divider for the clock source enabled us to generate a signal with a lower frequency to work with. We set it to 84-1, such that we could have an input of 1MHz. The ARR is the auto-reload register and by setting it we define the value that overflows the counter. After overflow the counter restarts from zero. We set it to 1’000’000-1 to achieve a frequency of 1Hz. In CCRx we set a value that is used to compare the value in the counter and correct the potential on the specific line. When the counter value is equal to the CCRx value the potential on the corresponding line x is set RESET (low value) otherwise it’s SET (high value). By setting it to 500’000-1 we were able to generate a square wave with duty cycle of 50%.  Pin PA5 (LD2) was then configured to have TIM2\_CH1 as output from the graphical IDE. In the code just HAL\_TIM\_PWM\_Start(&htim2, TIM\_CHANNEL\_1) needed to be called in the main before the while to start the PWM signal generation. | | | |
| Professor comments:  Very good! You proposed a good solution to debounch the mic. | | | |