Complete the “yellow” tabs and delate the phrases in italics.  
You can duplicate the table “Project”, if more than one project are due for the homework.

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *A1* | | |
| Homework number: | *2* | | |
| Due date: | *03/10/2022* | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| 1 *Giorgio Donato Carlo* |  |  | *x* |
| 2 Lodari Gianmarco |  |  | *x* |
| 3 Lenzi Francesco |  |  | *x* |
| 4 Chiapparo Lenn |  |  | *x* |
| 5 *Lanzini Alessio* |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Play a Song (interrupt) | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| Explanation: as shown by the professors during the lab session we identified the correct output pin for the loudspeaker and, after trying to play just a single note, we managed to make the struct and define the notes. Afterwards we connected the microphone pin as an interrupt and started the playsong() function inside the interrupt service routine. The problem in this case was probably that the audio coming from the loudspeaker triggered the interrupt, resulting in an infinite interrupt sequence that made the loudspeaker play just the first note. To solve this issue we instantiated a flag (that enables the usage of the playsong() function in a while(1) loop in the main) as a global variable, toggled it when the interrupt was called and untoggled it in the main loop when the playsong() function finished its routine. In this way we managed to make the program work as intended. | | | |
| Professor comments:  OK very good solution.  In class we will suggest another possible solution. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | Play a Song (interrupt, no HAL\_DELAY) | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Successfully completed |
|  |  |  | *x* |
| Explanation: to avoid the usage of the HAL\_DELAY function, on every play\_note() call we used TIM2 as a time base and exploited the interrupt that toggles on overflow to control when to stop the note. We set the prescaler to have a 0.5ms delay every count and set the autoreload register every note played. As a result, when TIM2 resets, the next note is played. The hard part was to find out how to manage the time base interrupt, clearing the flags and managing the callback. | | | |
| Professor comments:  Ok very good! | | | |