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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B01* | | |
| Homework number: | *HOMEWORK 10* | | |
| Due date: | 16/12/2023 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Francesco Scroccarello |  |  | *x* |
| Paolo Salvatore Galfano |  |  | *x* |
| William Stucchi |  |  | *x* |
| Giada Silvestrini |  |  | *x* |
| Francesco Maria Tranquillo |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name | IR communication | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  We successfully completed the homework.  **Transmit:** From the GUI, we set the following pins: PB10 as TIM2\_channel3, PC8, PC9, PC10, PC11 as GPIO\_Output and PC2, PC3, PC12, PC13 as GPIO\_Input. We enabled TIM2 with clock source as Internal Clock, Channel3 as PWM Generation CH3 and prescaler to 221-1, counter\_period to 10-1 (to get frequency of 38KHz), and pulse 5-1 to have a duty cycle of 50%. We then enabled TIM1 with clock source internal clock, precaler 3500-1, counter\_period 10-1 (to get frequency of 2400Hz, just like the baud rate); and TIM3 with clock source internal clock, precaler 8400-1, counter\_period 30-1 (to get frequency of 333Hz). For these last two timers we also enabled the relative interrupts. In the main.c code we started the timer 3 to scan the keyboard columns, and in its callback we set a flag to trigger the reading, which we perform in the while loop. To correctly manage the keyboard input we referred to the code that we previously developed for the Homework 9, but, instead of sending the character to the terminal, we transmit it via infrared. For this purpose, we define the function that takes an input byte and sends it through IR. Specifically, it fills a buffer with the bits that must be sent: a start bit (equal to 0), the 7 bits of data, a parity odd bit and a final stop bit (equal to 1), for a total of 10 bits. We then start TIM1, whose callback enables or disables the PWM generator if the bit to be sent is respectively 0 or 1.  **Receive:** From the GUI, we set the following pins: PA10 as USART1\_RX, PB6 as GPIO\_Output, PA7 as SPI1\_MOSI, PA5 as SPI1\_SCK.Then we activated TIM10 (prescaler:84-1 and counter\_period 4000-1, to have a frequency of 250 HZ), enabled the SPI1 in mode Transmit Only Master (in the parameter settings we set the prescaler to 4), added a SPI\_TX DMA configuration in Connectivity -> SPI1, Enabled USART1 in asynchronous mode (in parameter setting we set baud rate to 2400, parity odd and data direction as receive only), and enabled the TIM10 update interrupt. In the main.c code we defined the matrices representing each character that we want to display on the led matrix, then, before the while, we started the timer and enabled the UART to receive with interrupt. In the PeriodElapsedCallback we Reset the Pin to write on the led matrix, we transmitted the column to be displayed and then increased the index of such column. In the HAL\_SPI\_TxCpltCallback we set Writing pin (disabling the possibility to write on the led matrix); in the HAL\_UART\_RxCpltCallback, instead, we set a flag new\_command to true, in order to trigger the change of matrix to display: this happens asynchronously in the while loop. We did not manage directly the transmission via UART because it is already taken care of.  **Transmit and Receive:** to transmit and receive via IR from and to the same board, having done the two aforementioned projects like previously described, we simply copied the Pin and timer configurations, and we merged the two main.c into a single one. | | | |
| Professor comments: | | | |