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

|  |  |  |  |
| --- | --- | --- | --- |
| Team name: | *B5* | | |
| Homework number: | *04* | | |
| Due date: | Tuesday, October 24, 8:30 | | |
|  |  |  |  |
| Contribution | NO | Partial | Full |
| Ghidini Alessandro |  |  | *x* |
| Latino Francesco |  |  | *x* |
| Luppi Eleonora |  |  | *x* |
| Bravin Riccardo |  |  | *x* |
| Feltrin Elia |  |  | *x* |
| Notes: | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Project name |  | | |
| Not done | Partially done  (major problems) | Partially done  (minor problems) | Completed |
|  |  |  | *x* |
| Explanation:  The homework goal is to correctly read data from potentiometer and transmit them through UART first and then display them on the LCD. In Part 4a of the homework, the potentiometer value is read by setting up the pin, the ADC, and Timer1 and transmitted by habilitating the USART peripheral to show data on terminal. In Part 4b, the setup was similar, but without USART and with additional configuration for the LCD to display ADC values on the top row and a bottom bar whose length is proportional to potentiometer values.  **Part 4a:**  To be able to read the potentiometer value from the board it was necessary at first to find the pin to which it is connected in the “Green PCB board schematics.pdf” file. Once we knew it is connected to PA1 that pin needed to be setup as ADC\_IN1. From there in the ANALOG/ADC1 tab we needed to set the ADC\_Regular\_ConversionMode/Rank/Sampling Time to the maximum value of 480 cycles to ensure the complete charge/discharge of the capacitor connected to the potentiometer and ADC\_Regular\_ConversionMode/Rank/External Trigger Conversion Source to: Timer 1 Capture Compare 1 Event. Also the ADC1 Global Interrupt flag needed to be set to ensure that the function HAL\_ADC\_ConvCpltCallback gets correctly called after each completed read. Then Timer1 needed to be setup correctly to generate the interrupts for the ADC, it whas thus enabled on channel 1 as PWM generation and the values for the prescaler (8400-1), Counter Period (10000-1) and Pulse (1) were set.  We then needed to setup the USART connection to the PC by enabling it in the Connectivity/USART2 tab. In there we also had to enable the flag USART2 global interrupt to ensure a proper reset of the connection after each transfer.  In the code the Timer1 and ADC needed then to be started just before the while(1) start with the functions HAL\_TIM\_PWM\_Start and HAL\_ADC\_Start\_IT. The HAL\_ADC\_ConvCpltCallback needed then to be redefined to read the value from the ADC with the HAL\_ADC\_GetValue function, generate the string by converting the ADC retrieved value in the range 0-3.3 V which would then be sent through the UART with the function HAL\_UART\_Transmit.  By opening Putty or the Arduino monitor we could then read each second the string “Voltage: x.xxx” with the updated value being received.  **Part 4b:**  The second part of this project the setup is quite the same, because we also needed to set the potentiometer pin PA1 as ADC\_IN1. Then the ADC has to be configured with the same parameters as before, so with Sampling Time of 480 cycles and External Trigger Conversion Source to Timer 1 Capture Compare 1. Then, for Timer1, we need only to setup the correct values of Prescaler (8400-1) and Counter Period (2000-1) to have a 5 Hz PWM generation.  The difference from the first part is that we must set the pins to make the LCD work properly, so we set PA4, PB1, PB2, PB12, PB13, PB14, PB15 as GPIO\_Output.  From the NVIC panel we enable interrupt for ADC, then we save and generate the code.  In the code we need to include the “PMDB16\_LCD.h” library, which has been previously copied in the “Inc” folder, together with “string.h” and “stdio.h”. In the “Src” folder we copy the .c file corresponding to the LCD library.  In the main all we have to do is initializing the LCD, with the two functions lcd\_initialize() and lcd\_backlight\_ON(), the ADC in interrupt mode, by calling HAL\_ADC\_Start\_IT(&hadc1), and the PWM on Timer 1 by using HAL\_TIM\_PWM\_Start\_IT(&htim1, TIM\_CHANNEL\_1).  Then, in the HAL\_ADC\_ConvCpltCallback() we store the conversion value into two variables. The first is used to convert the value into a voltage to be shown on the top row of the the LCD, while the other is used to convert the acquired value have a conversion which is suitable for the function lcd\_drawBar(), so in the range 0-80: in this case, the converted value is stored in a float variable and then it is casted to an “int” when it is given in input to the bar function.  Lastly, to make something appear on the LCD we use the functions lcd\_println(string, 0) to put the string “Voltage: x.xxx V” (snprintf(string, sizeof(string), "Voltage: %.3f V\r\n", voltage)) on the top row and the function lcd\_drawBar((int) value) to draw a bar with a proportional length w.r.t. the acquired value on the bottom row.  By using the potentiometer, the bar changes its length accordingly to variations of the acquired values. | | | |
| Professor comments:  Part 4a:   * “Then Timer1 needed to be setup correctly to generate the interrupts for the ADC” 🡪 you should use the timer as a TRIGGER and not as an INTERRUPT. Triggers make peripherals to communicate without disturbing the core. Indeed in your project you used it correctly, but you described it not properly in the report. * “it whas thus enabled on channel 1 as PWM generation...” 🡪 You don’t need a PWM, you can use the timer in time base, and use the update event as trigger. | | | |