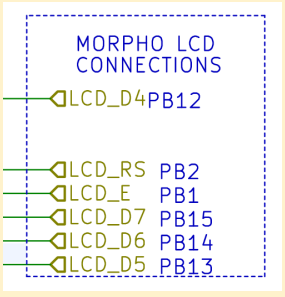
Mark	1/11

Team name:	*B5*		
Homework number:	*HOMEWORK 05*		
Due date:	22/10/24		
Contribution	NO	Partial	Full
Marenghi Manuela			*x*
Fellegara Tommaso			*x*
Giammusso Samuele			*x*
Cattani Luca			*x*
Csata Dániel			*x*
Notes: none			

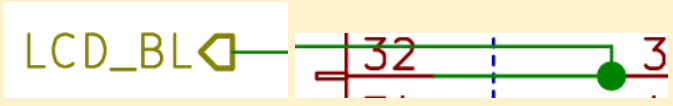
Project name	Test		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Part 1: from UART (DMA) to LCD

- To properly use the LCD, we set the pin PA4, PB1, PB2, PB12, PB13, PB14 and PB15 to be GPIO_output



-



-

CN7 even pins	
Name	**Pin**
PA4	32

- We copied and pasted the “PMDB16_LCD.c” in the Src folder and the “PMDB16_LCD.h” in the Inc folder, then in the main.c code we included the library (#include “PMDB16_LCD.h”).
- We created the timer TIM2 to generate an interrupt every second:
  - Channel 2 is set as ‘Output compare CH2’
  - Prescaler: 8400-1
  - counter period: 10000-1
  - Pulse: 5000-1
  - And the TIM2 global interrupt is enabled
- First we initialize LCD using lcd_initialize() and we start the background light using lcd_backlight_ON(), then we start the timer in interrupt mode:

```
123     lcd_initialize();
124     lcd_backlight_ON();
125     if(HAL_TIM_Base_Start_IT(&htim2) != HAL_OK){      //Start the timer in interrupt
126           //err;
127         }
```

- We implemented the code as shown below:
  - Every second, thanks to the timer interrupt, the function ReceiveString() gets called
  - The function ReceiveString() is used to receive one character at a time until the character is different from the ‘$’.
  - Every character gets inserted into the buffer
  - At the end of the reception, the terminator ‘\0’ is inserted in the buffer
  - Then the content of the buffer is printed

```
void ReceiveString() {
    int ind = 0;
    char buffer[100];
    char c = '$';

    do {
        HAL_UART_Receive(&huart2, &c, 1, HAL_MAX_DELAY);

        if(c != '$') buffer[ind++] = c;

    } while(c != '$');

    buffer[ind] = '\0';

    lcd_clear();
    lcd_println(buffer, 0);
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim) {
    if(htim->Instance == TIM2) {
        ReceiveString();
    }
}
```

- We modified the UART_send_string.m file as shown below:
  - we start by calculating the length of the string
  - then for every character of the string to send, we send the character on the UART interface and we wait for 0.5 second

- in the end, the '$' is sent, to communicate that the string is finished

```
%% Step 2: Asynchronous Write
stringToSend = "Fausto 2020"; % This is the only part you need to modify
len = length(stringToSend);
%write(s, len, "uint8");  %send the lenght of the string

for i=1:len
    write(s, stringToSend(i), "char"); %send one character at the time
    pause(0.5);                        %optional pause to allow the receiver to receive
end

write(s, '$', "char");
```

Part 2b - ADC triggered by TIM

- Setup the potentiometer to the ADC: from the ioc file, click on pin PA1 and select GPIO_Analog and ADC1_IN1
- Setup the ADC in interrupt mode: in the Analog category, set the sapling time to 480, in NVIC select ADC1 global interrupt
- Set up ADC conversion for Timer 2 TRGO event

ADC_Regular_ConversionMode	
Number Of Conversion	1
External Trigger Conversion S...	Timer 2 Trigger Out event
External Trigger Conversion E...	Trigger detection on the rising edge

- Set up the Timer 2 to generate a 1Hz TRGO event

Counter Settings	
Prescaler (PSC - 16 bits value)	8400-1
Counter Mode	Up
Counter Period (AutoReload R...	10000-1
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Trigger Event Selection	Update Event

- Setup the USART:

Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples

- In the main function start ADC & Timer

```
if(HAL_TIM_Base_Start_IT(&htim2) != HAL_OK){
    //err;
}
if(HAL_ADC_Start_IT(&hadc1) != HAL_OK){
    //err;
}
```

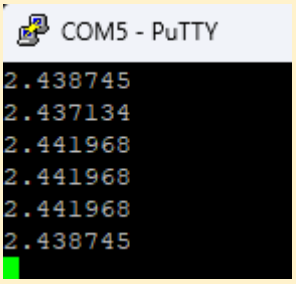
- Create the interrupt callback functions to convert the raw ADC data to voltage and send it via UART

```c
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc){
    uint32_t adcval = HAL_ADC_GetValue(&hadc1);
    float supply = 3.3;
    float voltage = adcval * supply / 4096;
    char uartstring[64];
    int uartlength = snprintf(uartstring, sizeof(uartstring), "%f\r\n", voltage);
    HAL_UART_Transmit(&huart2, uartstring, uartlength, 100000);

}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if(htim->Instance == TIM2) {
        // The ADC conversion will be triggered automatically by the timer
    }
}
```

- the output for 2b:



Part 2c - ADC triggered by TIM to LCD

- Configure LCD in the IOC file the same way as in Part1.
- In the main(), initialize LCD and turn the backlight on

```c
/* USER CODE BEGIN 2 */
lcd_initialize();
lcd_backlight_ON();
```

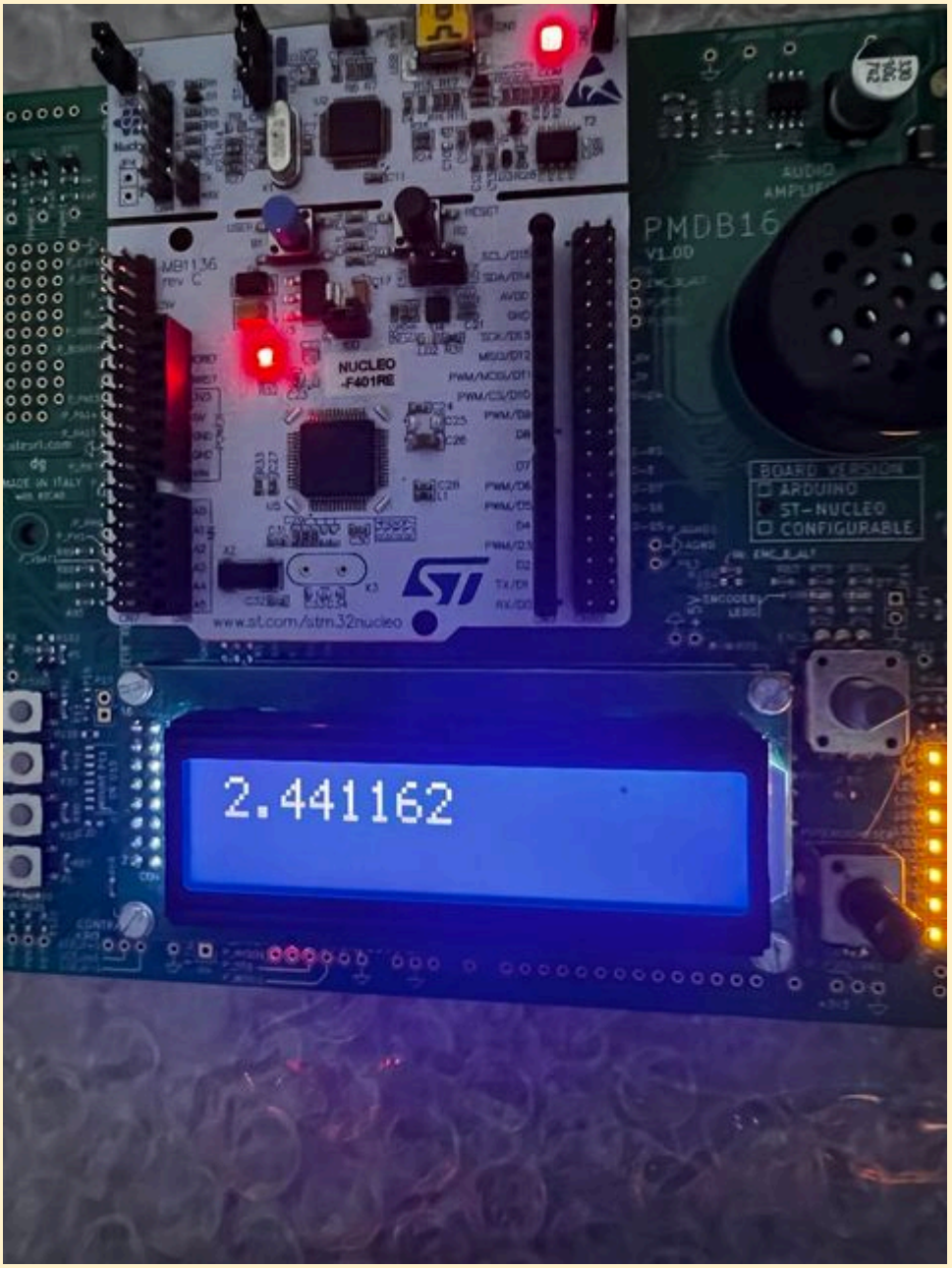
- Handle the ADC conversion the same way as in Part 2b
- Print the ADC voltages on the LCD

```c
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc){
    uint32_t adcval = HAL_ADC_GetValue(&hadc1);
    float supply = 3.3;
    float voltage = adcval * supply / 4096;
    //char uartstring[64];
    char lcdstring[64];
    //int uartlength = snprintf(uartstring, sizeof(uartstring), "%f\r\n", voltage);
    int lcdlength = snprintf(lcdstring, sizeof(lcdstring), "%f", voltage);
    //HAL_UART_Transmit(&huart2, uartstring, uartlength, 100000);
    lcd_println(lcdstring, 0);
    lcd_drawBar(voltage*80.0/3.3);
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if(htim->Instance == TIM2) {
        // The ADC conversion will be triggered automatically by the timer
    }
}
```

- We used 2 different strings so there is a linebreak character on the virtual COM port but that is not needed for the LCD, since we only display 1 line at the time
- The outputs for 2c:

-



Professor comments:

Part 1: The proposed solution is not reliable, since it is base on a sort of polling (every 1 s), instead you need to use IT to be sure not to miss any char. You will see during the lab how to receive properly single chars and strings.

Part2: you don't need to activate TIM2 in IT mode.
You are interrupting the CPU for no reasons.