Mark	A

Commented [FV1]: Cancellate il voto dal template prima di sottomettereli

Commented [FV2R1]: Il vero voto è B

Team name:	*A1*		
Homework number:	*HOMEWORK 4*		
Due date:	22/10/2022		
Contribution	NO	Partial	Full
Monti Pietro			*x*
Moretto Alessia			*x*
Pallotto Francesco			*x*
Perna Alessandro			*x*
Ventura Ludovico			*x*
Notes:			

Project name	ADC triggered by TIM and ADC triggered by TIM to LCD		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Explanation:
We successfully completed the homework.

Part 1a:

Firstly we started configuring the ADC of STM32 to respond to a timer-trigger event. We set the Clock Prescaler to its maximum value (PCLK2/4) with a resolution of 12 bits and with a Sampling Time of 480 cycles.
We than set the USART setting it in the Asynchronous mode with a Baud Rate of 115200 Bits/s.
Finally we used the timer TIM2 to generate a periodical output event. To achieve a 1Hz timer frequency we set the Prescaler (PSC) to 8400-1 and the Auto-Reload Register (ARR) to 10000-1.
After preparing the GUI we set ADC1, USART2 and TIM2 as global interrupts from the NVIC table.

In ADC_value we store the sampled value associated to hadc1, in voltage we acquire the voltage value computed through the conversion. The array “text” is where we write the message we need to transmit and in length we find the length of the message.

```
uint32_t ADC_value;
float voltage;
char text[64];
int length;
```

We defined the interrupt ADC callback function which sends the stored value to the remote terminal through the UART interface. It transmits the message in “text” and makes the ADC starting in interrupt mode.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim == &htim2) {
        HAL_UART_Transmit(&huart2, (uint8_t *)text, length, 20);
        HAL_ADC_Start_IT(&hadc1);
    }
}
```

When the sampling phase is completed we get the ADC value and we compute its corresponding voltage value as voltage=ADC_value*3.3/4096.0
After doing that we format the message that we need to transmit to the UART writing it in “text”.

```c
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc) {
    if (hadc == &hadc1){
        ADC_value = HAL_ADC_GetValue(&hadc1);
        voltage = ADC_value*3.3/4096.0;
        length = snprintf(text,sizeof(text),"voltage=%3f V\r\n",voltage);
    }
}
```

Part 1c:

The code produces an ADC reading once every second, making use of the ADC in interrupt mode as well as timer TIM2 for time management. The two are set up in such a way as to prompt a reading once every second and print it to a remote terminal.

Implementation Steps:

Initialization of Array and Variables:

```c
uint32_t ADC_value;
float voltage;
int length;
char string_LCD[16];
```

We started by declaring the necessary variables: “ADC_value” is used to save the voltage level as it is converted by the ADC, whereas “voltage” stores the conversion of the ADC reading into Volts, (the ADC is configured with a 3.3V reference voltage and is set to a 12-bit resolution, meaning it has 4096 possible values). The “length” variable is instead used to keep track of the number of characters written to “string_LCD”.

Initialization of LCD, ADC & TIM2 Timer:

To achieve our desired timing, we initialized the timer to trigger once every second. This is done by configuring its Prescaler to 8400-1 and its period to 10000-1. The timer's flag is cleared to prevent it from triggering immediately upon activation. As for the ADC, the Clock Prescaler is set to PCLK2/4, with a Resolution of 12 bits and a Sampling Time of 480 cycles. At this point, TIM2 and ADC can be started.

We then used the functions **`lcd_initialize()`** and **`lcd_backlight_ON()`** to initialize the LCD and turn on its backlight. Both functions are declared in the "PMDB16_LCD" file, which was imported into the project files and then included in the ‘main.c’ file.

```c
__HAL_TIM_CLEAR_FLAG(&htim2, TIM_FLAG_UPDATE);
HAL_TIM_Base_Start_IT(&htim2);
HAL_ADC_Start_IT(&hadc1);

lcd_initialize();
lcd_backlight_ON();
```

Handling the Callbacks:

The conversion from ADC output value to the voltage being read is performed in the ADC callback. After saving the ADC reading into “ADC_value”, we converted it to the appropriate voltage value by multiplying by the reference voltage (3.3V) and dividing by the maximum value a 12-bit ADC can represent ($2^{12} - 1 = 4095$). Finally, we formatted the string to be printed out while making sure to avoid buffer overflow.

```c
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc) {
  if (hadc == &hadc1){
    ADC_value = HAL_ADC_GetValue(&hadc1);
    voltage = ADC_value * 3.3 / 4095.0;
    length = snprintf(string_LCD, sizeof(string_LCD), "Voltage = %3f V\r\n", voltage);
  }
}
```

The other callback, relative to TIM2, was used to print the string to the first row of the LCD, and the bar graph representing it on the second. The arguments of **LCD_drawBar** make sure the graph spans the entirety of the LCD row when "voltage" is at its highest (the value range goes from 0 to 80). Finally, **HAL_ADC_Start_IT(&hadc1)** restarts the ADC, allowing for the entire process to repeat itself every second.

```
1  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
2    if (htim == &htim2) {
3      lcd_println(string_LCD, 0);
4      lcd_drawBar(voltage / 3.3 * 80);
5      HAL_ADC_Start_IT(&hadc1);
6    }
7  }
```

Professor comments:

Part A:

- "we set ADC1, USART2 and TIM2 as global interrupts from the NVIC table." → NOO!
  You don't need TIM2 interrupt, we use it as a TRIGGER! (triggers make peripherals to communicate, without disturbing the core).
  You enabled USART interrupt, but then you used it in blocking mode.
- You should not start the ADC by software, but using the trigger, so this part is not correct:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim == &htim2) {
        HAL_UART_Transmit(&huart2, (uint8_t *)text, length, 20);
        HAL_ADC_Start_IT(&hadc1);
    }
}
```

Same problems also in the other part.