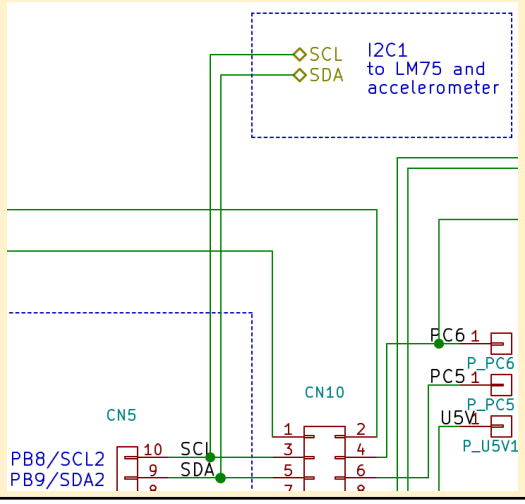
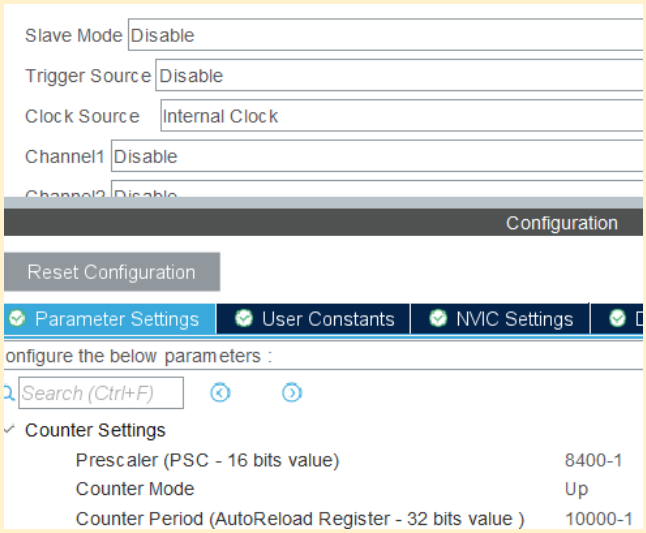
Mark	1/11

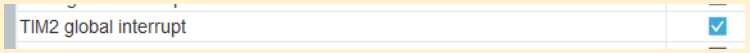
Team name:	*B5*		
Homework number:	*HOMEWORK 07*		
Due date:	5/11/24		
Contribution	NO	Partial	Full
Marenghi Manuela			*x*
Fellegara Tommaso			*x*
Giammusso Samuele			*x*
Cattani Luca			*x*
Csata Dániel			*x*
Notes: none			

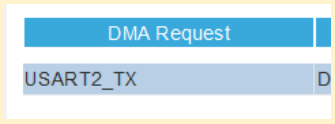
Project name	Test		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

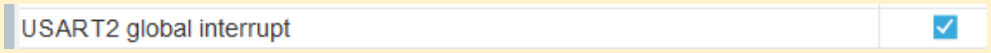
Part 1b (version LM75):

- Setup the pin of the I2C to the temperature sensor LM75
  - SCL is connected to PB8, and SDA is connected to PB9
  - 

- Setup the I2C by enabling it in the connectivity tab, and keep it in Standard Mode
- Setup the timer T2, so that it operates at 1hz, and generates an interrupt:
  - 

  - 

- Enable the USART2 DMA and the global interrupt:
  - 

  - 

- How we implemented the code:
  - in the global scope:
    - for writing the LSB needs to be 0
    - for reading we need to add 1, so that the LSB is 1
  - ```
    #define PER_ADDRESS_WRITE 144 // equal to 0b10010000
    #define PER_ADDRESS_READ 145 // equal to the previous+1
    ```
    - we declared some variables useful later:
  - ```
    uint8_t zero_value = 0;
    int count=0;

    uint8_t temp_value[2] = {0};
    char str[64];
    int len;
    float converted_value;
    ```
  - in the main, we start the timer and we start the I2C communication
  - ```
    HAL_TIM_Base_Start_IT(&htim2);
    HAL_I2C_Master_Transmit(&hi2c1, PER_ADDRESS_WRITE, &zero_value, 1, 100);
    ```
  - in the timer callback we implemented the logic, described in the comment of the code
    - The transmission of data to the Matlab terminal is done using the UART
    ```

```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
    if (htim->Instance == TIM2) {
        if(HAL_I2C_Master_Receive(&hi2c1, PER_ADDRESS_READ, &temp_value, 2, 100) == HAL_OK) {
          temp_value[1] = temp_value[1] >> 7;  //right shift the value of 7 positions
          if(temp_value[0] >> 7) {    //if the MSB of the integer part is one, then it's a negative value
              temp_value[0] = (~temp_value[0]); //reverse all the bits
              temp_value[1] = (temp_value[1] ^ 1) + 1;  //complement of 1, and then add 1

              if (temp_value[1] > 1){//if there is overflow
                  temp_value[1] = 0; //reset the fractional part to 0
                  temp_value[0]++; //and increment the integer part
               }
              // fractional part can be either 1/2=0.5 or 0/2=0
              converted_value = -(temp_value[0] + (float)temp_value[1] / 2.0);
          } else {
              converted_value = temp_value[0] + (float)temp_value[1] / 2.0;
          }

          len = snprintf(str, sizeof(str), "%d, The temperature is %.3f C\r\n",count, converted_value);
          count++;
          if( HAL_UART_GetState(&huart2)==HAL_UART_STATE_READY ){
              HAL_UART_Transmit_DMA(&huart2, str, len);
          }
      }
    }
}
```

Part 1b (version LM75B):

- The setup is the same as before, except for the TIMER 2 prescaler, which now is 8400/5 = 1680, in order to achieve a 5hz interrupt.
- The implementation is described below:
    - We defined some variables: an array to keep the samples, a counter and a value for the negative temperatures
    - ```c
      float temperatures[SAMPLES];
      int cnt = 0;
      int neg = 0;
      ```
    - ```c
      #define SAMPLES 5
      ```
    - And the 2 called functions are: a sorting algorithm, and a check for the negative sign bit
    - ```c
      void insertionSort() {
          for(int i = 1; i < SAMPLES; i++) {
              float key = temperatures[i];
              int j = i - 1;

              while(j >= 0 && temperatures[j] > key) {
                  temperatures[j + 1] = temperatures[j];
                  j--;
              }
              temperatures[j + 1] = key;
          }
      }

      int isNegative(uint8_t msb){
          return msb >> 7;
      }
      ```
      ```

```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim){
    if(htim->Instance == TIM2){
        cnt++; //increment the counter
        uint8_t p[2] = {0,0}; //array that memorize the byte coming from the sensor
              int len = 0;
              char string[64];
              if(HAL_I2C_Master_Receive(&hi2c1, TEMP_ADDRESS, &p, 2, 100)== HAL_OK){

                  p[1]= p[1]>>5; //right shift the fractional part bits of 5 positions
                  if(isNegative(p[0])){
                      neg=1;
                      p[0] =~ p[0]; //complement of 1, to obtain the positive part
                      p[1] ^= 0b00000111; // XOR operator
                      if (++p[1] > 0b00000111){ //if there is overflow
                          p[1] = 0; //the fractional part become zero
                          p[0]++; //increment the integer part
                      }
                  }
                  float x = p[0] + (float)p[1]/8.0; //resolution of 3 bits, so 2^3=8
                  temperatures[cnt - 1] = x; //save the sample in the array

                  //if all the samples have been collected
                  if(cnt == SAMPLES && HAL_UART_GetState(&huart2)==HAL_UART_STATE_READY ){
                      cnt = 0;
                      insertionSort(); //sort the samples
                      if(neg==1){ //if the temperature is negative
                          //print the median value
                          len = snprintf(string, 64, "Temperature -%.3f C\n", temperatures[SAMPLES / 2]);
                          neg=0;
                      }else{
                          len = snprintf(string, 64, "Temperature %.3f C\n", temperatures[SAMPLES / 2]);
                      }
                      HAL_UART_Transmit_DMA(&huart2, string, len);
                  }
              }
    }
}
```

- There are some differences with the previous implementation due to a different version of the sensor:
    - the right shift of the fractional part, in this case of only 5 positions

```c
p[1]= p[1]>>5; //right shift the fractional part bits of 5 positions
```

	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
LM75B	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X	X	X

    - the computation of the value of the fractional part, which now has 3 digit resolution

```c
float x = p[0]+ (float)p[1]/8.0; //resolution of 3 bits, so 2^3=8
```

How we solved the problem:

- Explanation of the bug:
    - when the temperature change, sometimes the sensor flips only the integer bits (the MSByte) or only the fractional bits (the LSByte)
    - This cause to have some inconsistent data
- Our Solution:
    - We use a timer of 5 hz so that every second we have 5 values of temperature.
    - For example:
    - 21.000, 21.875, **22.875**, 22.000, 22.125 (the red number is the manifestation of the bug)
    - So if we sort it:
    - 21.000, 21.875, **22.000**, 22.125, 22.875
    - And then if we keep only the median, we obtain a value that is accurate enough, which is 22.000

Professor comments:

Better to use I2C in DMA mode.
The proposed method for solving the sensor bug is fine