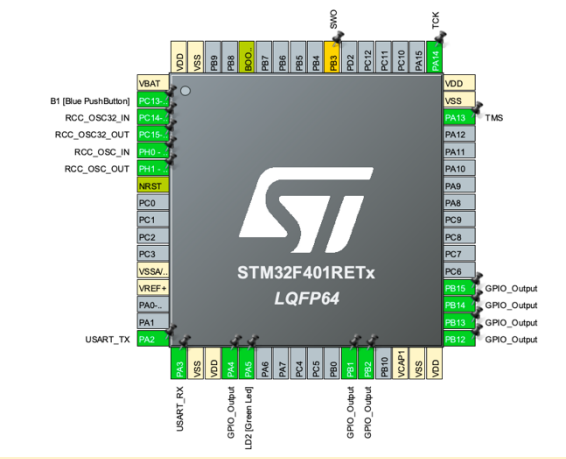
Mark	/11

Team name:	*A1*		
Homework number:	*HOMEWORK 05*		
Due date:	22/10/24		
Contribution	NO	Partial	Full
Piombo			*x*
Fumagalli			*x*
Pierfederici			*x*
Zenoni			*x*
Ferraro			*x*
Notes:			

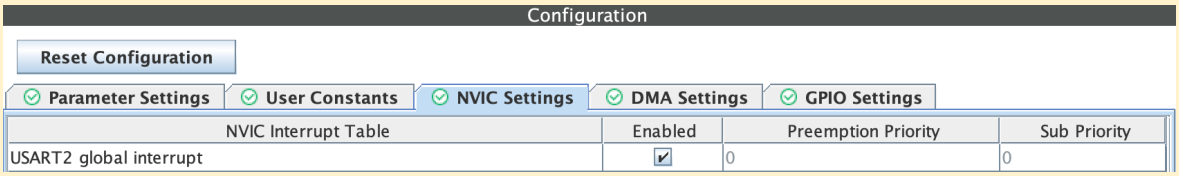
Project name	UART + ADC		
Not done	Partially done (major problems)	Partially done (minor problems)	Completed
			x

Part 1:

We set the UART and LCD pins using our UI as below

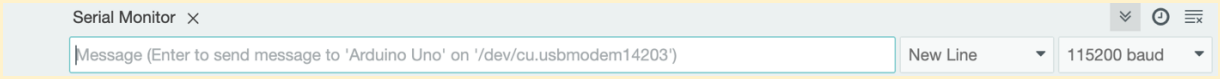


We decided to exploit the UART interrupt to perform the communication



We chose to receive each character of the received string singularly, looking for the new line character (\n) to end the string and print it on the LCD.

We used the Arduino IDE serial monitor in order to send strings to the board.



In the main function we only initialized the LCD and started the first UART receive.

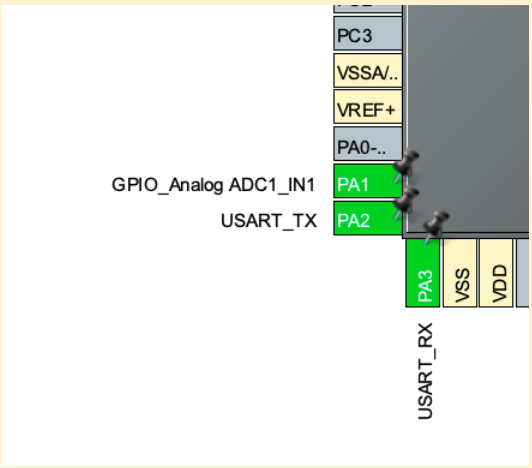
```
/* USER CODE BEGIN 2 */

lcd_initialize();
lcd_backlight_ON();

HAL_UART_Receive_IT(&huart2, &c, 1);

/* USER CODE END 2 */
```

In the UART ISR we save each character in a string, and print eventually the string when is finished.

```
void HAL_UART_RxCpltCallback (UART_HandleTypeDef *huart) {
    if (huart == &huart2) {
        string_rx[i] = c;                       // save received character in our string
        if (c == '\n') {                        // string is finished
            string_rx[i] = '\0';                // terminate properly the string
            lcd_println(string_rx, 0);          // print the string on the LCD
            i = 0;
        } else {
            i++;
        }
        HAL_UART_Receive_IT(&huart2, &c, 1);    // receive the new character
    }
}
```

Part 2:

The potentiometer is connected to the PA1 pin of the microcontroller, so we set it as ADC1_IN1. We also checked proper configuration of UART pins.



Then we configured properly our ADC, setting it to start the conversion by the timer 2 trigger out event, and we decided to operate it in interrupt mode.

ADC1 Mode and Configuration

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

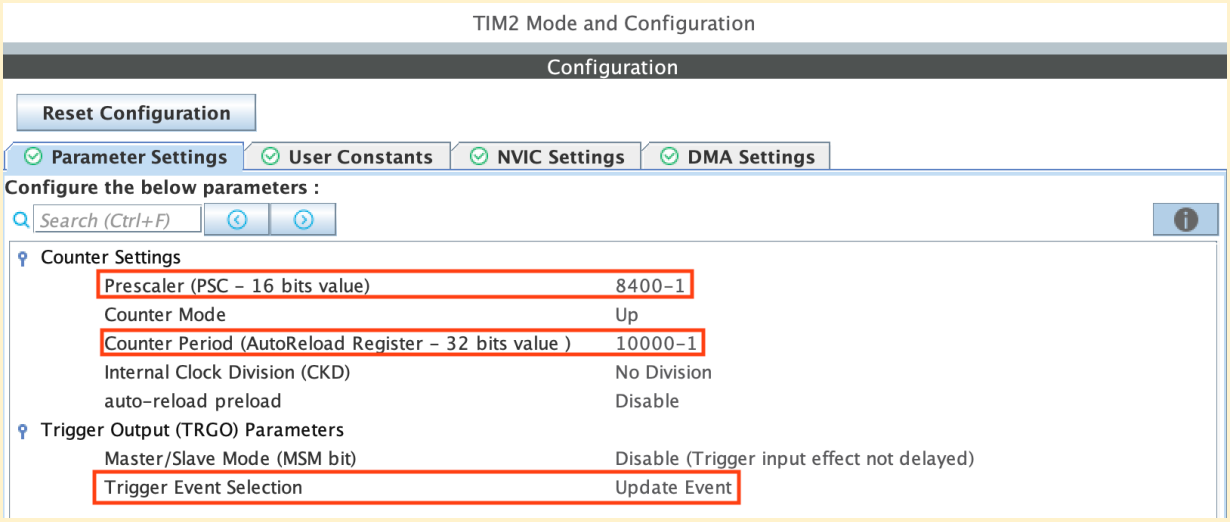
Search (Ctrl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	PCLK2 divided by 4
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Disabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	EOC flag at the end of single channel conversion
ADC_Regular_ConversionMode	
Number Of Conversion	1
External Trigger Conversion Source	Timer 2 Trigger Out event
External Trigger Conversion Edge	Trigger detection on the rising edge
Rank	1
Channel	Channel 1
Sampling Time	480 Cycles
ADC_Injected_ConversionMode	
Number Of Conversions	0
WatchDog	
Enable Analog WatchDog Mode	☐

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
ADC1 global interrupt	☑	0	0

In order to do this, we needed also to configure properly the timer trigger out parameters. We also set timer 2 parameters in order to have a frequency of 1Hz.



In the main function we configured properly the ADC and we started our timer.

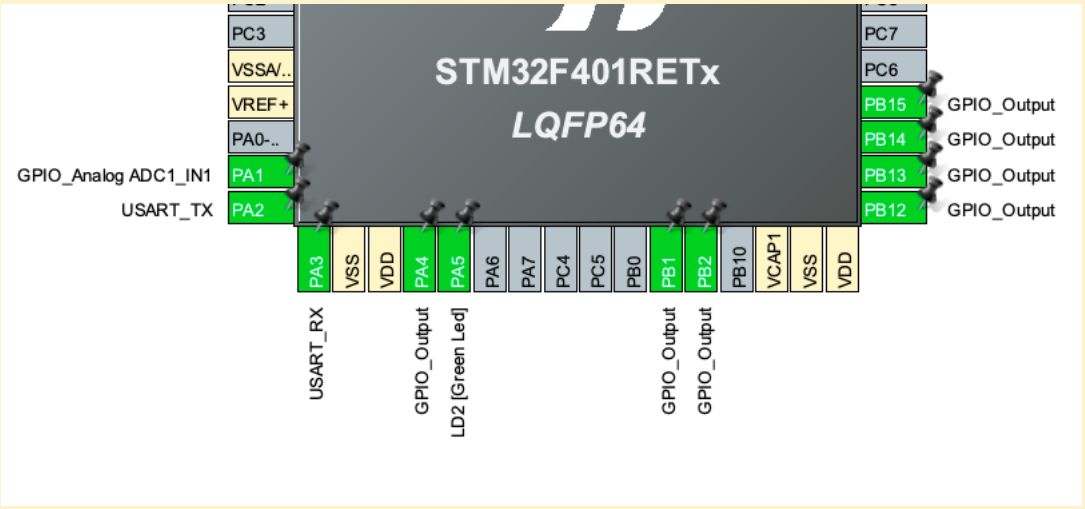
```
HAL_ADC_Start_IT(&hadc1);          // to properly configure the peripheral
__HAL_TIM_SET_COUNTER(&htim2, 0);
HAL_TIM_Base_Start(&htim2);        // start the timer
```

Then the ADC ISR computes the analog voltage value (LSB is defined as 3.3/4096.0), and send it using UART.

```
void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef *hadc) {
    if (hadc == &hadc1) {
        digital_voltage = HAL_ADC_GetValue(&hadc1);
        analog_voltage = LSB*digital_voltage;
        length = snprintf(string, sizeof(string), "Voltage: %.3fV\n", analog_voltage);
        HAL_UART_Transmit(&huart2, string, length, 100);
    }
}
```

Part 3:

We set the potentiometer pin as part 2 and set also LCD pins as below



ADC and timer 2 configuration is the same of part 2.

The main function is the same as part 2, the only difference is the LCD initialization.

We changed the ADC ISR to print the analog voltage on the LCD and properly draw the bar proportional to the voltage value (BAR_STEP is a conversion factor defined as 80.0/4096.0).

```
void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef *hadc) {
    if (hadc == &hadc1) {
        digital_voltage = HAL_ADC_GetValue(&hadc1);
        analog_voltage = LSB*digital_voltage;
        length = snprintf(string, sizeof(string), "Voltage: %.3fV", analog_voltage);
        lcd_println(string, 0);
        lcd_drawBar(BAR_STEP*digital_voltage);
    }
}
```