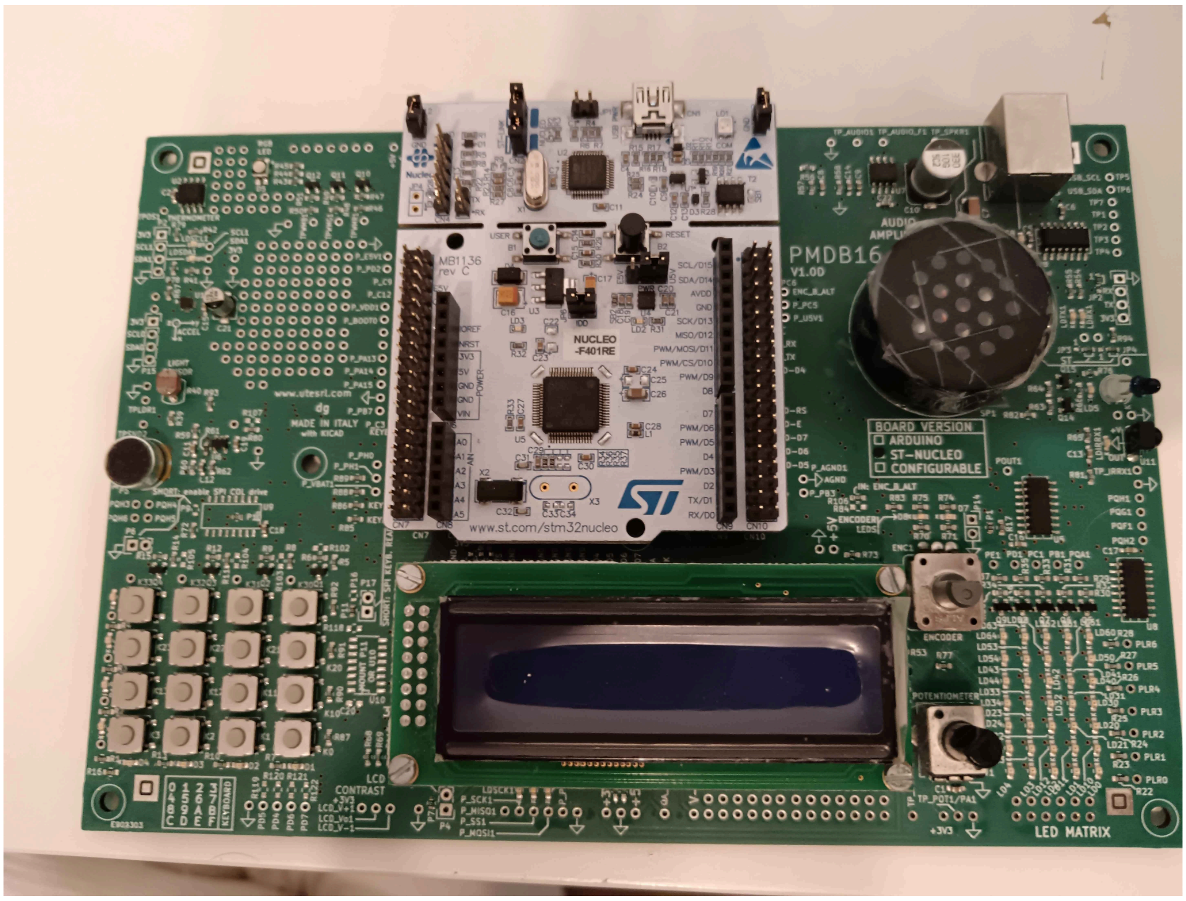
Computer Science Team B4

Sensor system

Homework Report



HOMEWORK NUMBER 6

2024 - 2025

POLITECNICO
MILANO 1863

Summary

I. Members work

Members	Exercise 1	Exercise 2	General
Cazin Némo	✓	✓	Reviewed report
Adrien Paliferro	✓	✓	Reviewed report
Heikki Leveelahti	✓	✓	Wrote the report for exercise 1
Osmo Kieksi	✓	✓	Wrote the report for exercise 2
Constantijn Coppers	✓	✓	Reviewed report

II. Exercise 1

General functionality:
In this project, we want to acquire 3 different voltages. One from the potentiometer, one from the temperature sensor and one internal reference voltage. The acquisition is started with Timer 2, which is set to trigger every 1 second. The ADC values then are converted to voltages or to a temperature. Then a string with calculated values is sent to a remote terminal with UART.

Implementation:
To obtain the desired functionality, the following steps are followed :

1. **PIN lookup**

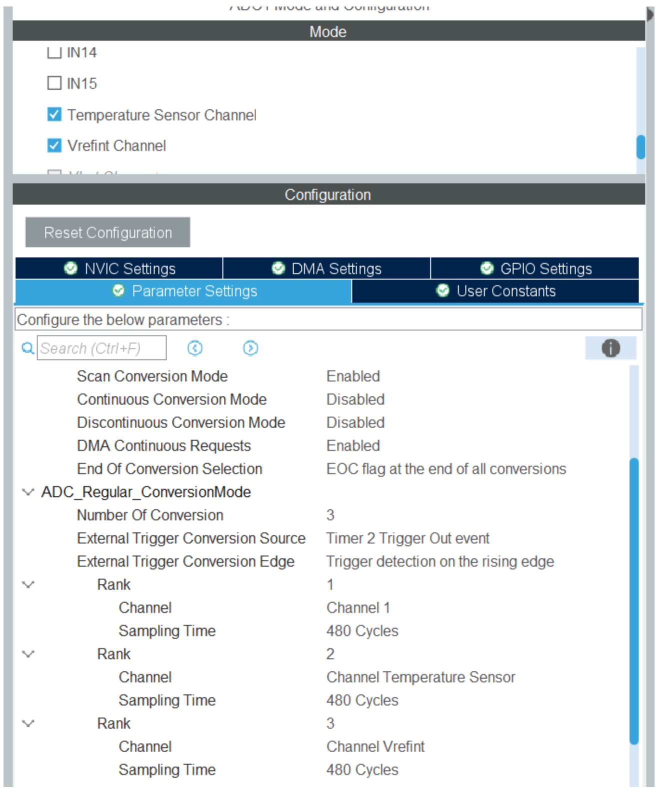
The used peripherals with their corresponding PIN are displayed in the table below.

Peripheral	**PIN**	**I/O**
ADC	PA1	ADC1_IN1
USART_TX	PA2	Transmit
USART_RX	PA3	Receive

The internal reference voltage and temperature sensor are connected straight to one of the ADC internal channels.

2. **Board configuration**

First, we need to configure the ADC. ADC conversion is set to start when TIM2 reaches ARR value so every 1 second the ADC conversion is started.
We put PA1, which is the potentiometer pin in the configuration "ADC1_IN1".
Because of this PIN Configuration, we need to configure the ADC channel 1.
We just need to change the parameter "Sampling Time" from 3 cycles to 480 cycles.
We also need to activate on the NVIC tab, the interruption of the ADC.
Also the Channels Temperature Sensor Channel and Vrefint Channels are chosen as active channels and configured as channel 1. We want to save the data using DMA so that it is also configured and DMA continuous Request is enabled as shown below in the screenshot.



IN14
IN15
☑ Temperature Sensor Channel
☑ Vrefint Channel

Configure the below parameters :

Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
End Of Conversion Selection	EOC flag at the end of all conversions
ADC_Regular_ConversionMode	
Number Of Conversion	3
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
Rank	2
Channel	Channel Temperature Sensor
Sampling Time	480 Cycles
Rank	3
Channel	Channel Vrefint
Sampling Time	480 Cycles

ADC global interrupt also needs to be activated.

USART is configured to send data with DMA as in the previous exercises.

We want to collect the data and send calculated values every 1 second, so TIM2 is configured as shown below.
So we need to configure the timer TIM2, we have these principal parameters :

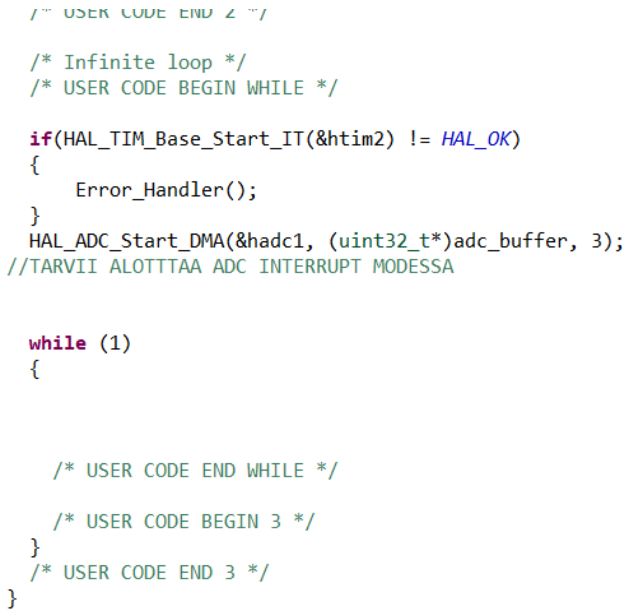
- Clock Source : Internal Clock
- Prescaler : 8399
- Period : 9999
- Trigger Event Selection : Update Event

Finally, we can activate the floats in the properties tab of the project for a proper conversion of the float tension value.

After the general configuration, we can work on the code in the main.c file.

3. Code implementation

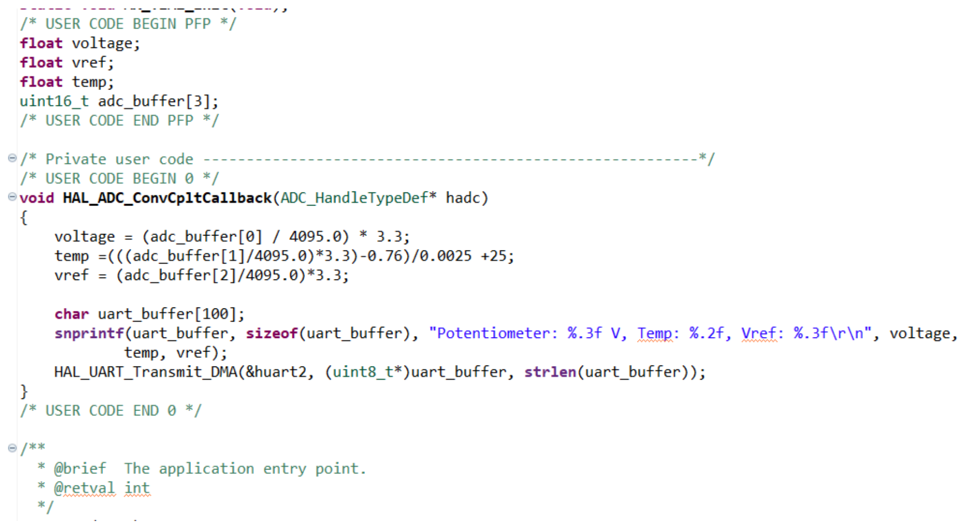
So first we need to activate the timer and start the ADC in DMA mode in the main function :



```
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */

  if(HAL_TIM_Base_Start_IT(&htim2) != HAL_OK)
  {
      Error_Handler();
  }
  HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buffer, 3);
//TARVII ALOTTTAA ADC INTERRUPT MODESSA


  while (1)
  {



    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

adc_buffer is an array of 3 with 16 bit values we can see this in the next picture.

The timer is now working and the ADC is triggered with hardware from now on with the timer 2. We can do the calculation of the different voltages and the temperature in the ADC callback. Also the UART communication is handled in the ADC callback:



```
/* USER CODE BEGIN PFP */
float voltage;
float vref;
float temp;
uint16_t adc_buffer[3];
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    voltage = (adc_buffer[0] / 4095.0) * 3.3;
    temp =(((adc_buffer[1]/4095.0)*3.3)-0.76)/0.0025 +25;
    vref = (adc_buffer[2]/4095.0)*3.3;

    char uart_buffer[100];
    snprintf(uart_buffer, sizeof(uart_buffer), "Potentiometer: %.3f V, Temp: %.2f, Vref: %.3f\r\n", voltage,
            temp, vref);
    HAL_UART_Transmit_DMA(&huart2, (uint8_t*)uart_buffer, strlen(uart_buffer));
}
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
```

4. Execution

Link to the video : homework6

III. Exercise 2

General functionality:

In this project, we read ADC value from the photoresistor once in millisecond. This value is saved to the transmission buffer with DMA. Interrupt is triggered when the buffer is half full and completely full. Buffer has room for 2000 values so interrupts are triggered every one second. Then the corresponding half of the buffer is passed to a function that calculates average resistance and corresponding lux value of the photoresistor. These values are then sent to the remote terminal via USART.

Implementation:

1. **PIN lookup**

Pins are initialized as followed :

Peripheral	**PIN**	**I/O**
ADC	PA0	ADC1_IN0
USART_TX	PA2	Transmit
USART_RX	PA3	Receive

2. **Board configuration**

First, we need to configure the ADC.
We put P01, which is a pin in the configuration "ADC1_IN0".
Because of this PIN Configuration, we need to configure the ADC channel 0.
We just need to change the parameter "Sampling Time" to 480 cycles.
We also need to activate on the NVIC tab, the interruption of the ADC. We want that timer to start ADC so we set timer 3 trigger out event as external trigger source.

We want to send the value acquired on a terminal, that's why we need to configure the USART.
The configuration of USART is by default, we have the principal parameters :

- Mode : Asynchronous
- RS232 : Disable
- Baud Rate : 115200 Bits/s

We also need to make the conversions at a rate of 1ms.
So we need to configure the timer TIM3, we have these principal parameters :

- Clock Source : Internal Clock
- Prescaler : 8399
- Period : 9

We also need to set trigger event selection to update events so ADC will be directly triggered when the timer reaches max value.

Because data will be saved and read directly from memory via DMA we need to activate DMA. First channel of DMA will be used with USART so it must be set to stream source. In the same way we set ADC to stream from the second DMA channel.

3. **Code implementation**

First we include some necessary libraries and initialize the array for ADC result and UART buffer.

```
#include <string.h>
#include <stdio.h>
#include <math.h>

uint16_t adcValues[2000]; // Array to store ADC values from three channels
char txBuffer[50]; // Buffer to store the string to send over USART
```

ADC array contains 2000 places so it can hold ADC conversions for the last two seconds.

In main function it's only needed to initialize peripherals and run following lines to start timer in interrupt mode and start ADC in dma mode:

```
if (HAL_TIM_Base_Start_IT(&htim3) != HAL_OK)
{
        Error_Handler();
}

if (HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adcValues, 2000) != HAL_OK){
         Error_Handler();
}
```

Timer interrupt function is triggered every one second and it's responsible for starting a new ADC in DMA mode.

ADC buffer is handled with two similar callback functions. Half Full callback is triggered when the buffer is half full and full callback when it's completely full.

```
void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc){
	// Call send data function, pointer to first value of array
	SendData(adcValues);
}
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc){
	// Call send data function, pointer to 1000th value of array
	SendData(adcValues + 1000);
}
```

In both functions SendData function is called. It gets a pointer to the ADC buffer as its parameter. It's either first or the 1000th value depending on how full the buffer is.

```
void SendData(uint16_t* HalfArray){
	// Calculate mean of desired half of array
	uint32_t adcMean = CalculateMean(HalfArray);
	// Calculate adc V, LDR and LUX values
	float adcV = adcMean / 4095.0 * 3.3;
	float LDR = adcV * 100000 / (3.3 - adcV);
	float LUX = pow((10 * (100000 / LDR)), 1.25);
	snprintf(txBuffer, sizeof(txBuffer), "LDR: %.2f Ohm, LUX: %.2f Lux\r\n", LDR,
LUX);
	HAL_UART_Transmit_DMA(&huart2, (uint8_t*)txBuffer, strlen(txBuffer));
}
```

First mean of array is calculated in CalculateMean function

```
uint32_t CalculateMean(uint16_t* HalfArray){
	uint32_t sum = 0;
	// Calulate and return mean value of array
	for(uint16_t index = 0; index < 1000; index++){
		sum += HalfArray[index];
	}
	uint32_t result = sum / 1000;
	return result;
}
```

Then adcV, LDR and LUX are calculated based on the mean of ADC conversions. Finally those are saved to the string and that string is sended to the remote terminal via USART and DMA.

4. Execution

Link to the video : Video

IV. Professors comments

EXERCISE 1 :
EXERCISE 2 :