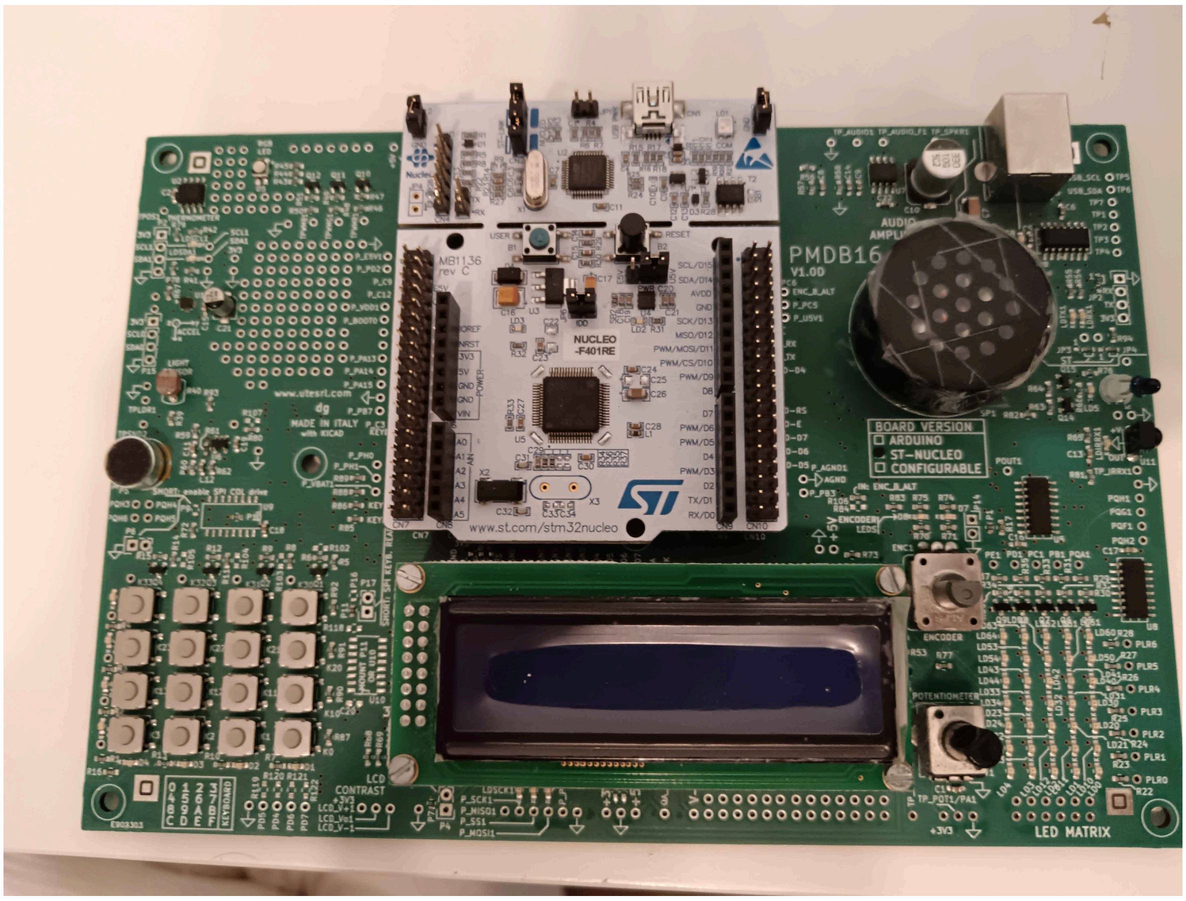
Computer Science Team B4

Sensor system

Homework Report



HOMEWORK NUMBER 4

2024 - 2025

POLITECNICO
MILANO 1863

Summary

I. Members work

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

II. Exercise 1

General functionality:
In this project, we send a few characters with USART2 using the DMA peripheral. Arduino IDE was used as an emulator to validate the implemented functionality.

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**
USART_TX	PA2	Transmit
USART_RX	PA3	Receive

2. **Board configuration**

USART2 is activated by default so we don't need to do anything to the pinout. In order to have a working UART with DMA we have to decide and modify some parameters. Also the receiving end of UART has to have these same parameters.

Under Connectivity → USART2, following settings and parameters are used with our serial communication.

Mode panel

- Mode	:	Asynchronous
- Hardware Flow control	:	Disable

Configuration panel

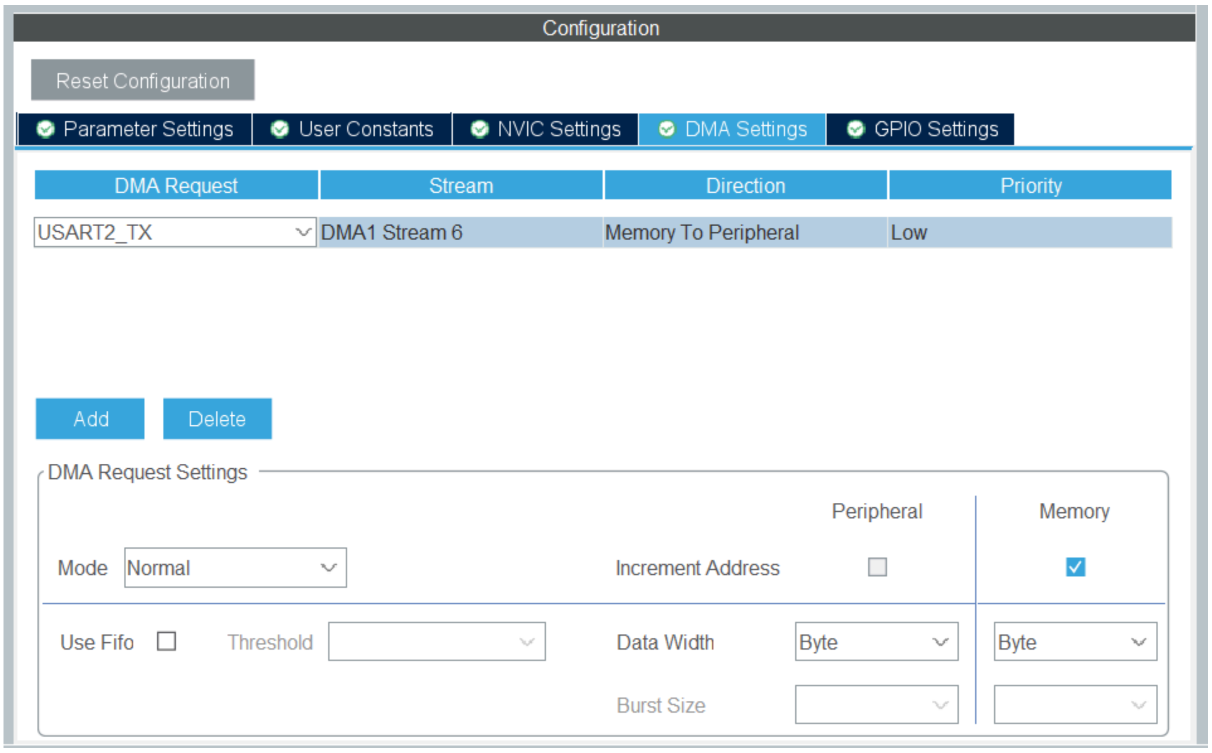
Parameter Settings:

- Baud Rate	:	115200 Bits/s
- Word Length	:	8 Bits (including Parity)
- Parity	:	None
- Stop Bits	:	1

Advanced Parameters can be left to default values.

DMA Settings:

We need to enable DMA with USART2_TX request in normal mode as shown in the picture below.



DMA Request	Stream	Direction	Priority
USART2_TX	DMA1 Stream 6	Memory To Peripheral	Low

DMA Request Settings

		Peripheral	Memory
Mode: Normal	Increment Address	☐	☑
Use Fifo ☐ Threshold:	Data Width	Byte	Byte
	Burst Size		

NVIC Settings:

We have to enable USART2 global interrupts and give it priority of more than zero. This is so we can use HAL_DELAY in our interrupt to decide the time how often we transmit data.

3. Code implementation

First, we create a private variable that is our “data” in this case it’s a string. Also we want to include string.h library so we can use the strlen() function to get the size of our data for the HAL_UART_Transmit_DMA function.

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <string.h>
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;
DMA_HandleTypeDef hdma_usart2_tx;
/* USER CODE BEGIN PV */
char StringToSend[] = "Heikki 2000\n";
/* USER CODE END PV */
```

The main is pretty much just the initialize function calls and the first transmit with UART using DMA. This is because the interrupt handler keeps sending the same string until the device is turned off.

```
int main(void)
{
 /* USER CODE BEGIN 1 */
 /* USER CODE END 1 */
 /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the Systick.
*/
 HAL_Init();
 /* USER CODE BEGIN Init */
 /* USER CODE END Init */
 /* Configure the system clock */
 SystemClock_Config();
 /* USER CODE BEGIN SysInit */
 /* USER CODE END SysInit */
 /* Initialize all configured peripherals */
 MX_GPIO_Init();
 MX_DMA_Init();
 MX_USART2_UART_Init();
 /* USER CODE BEGIN 2 */
 /* USER CODE END 2 */
 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
 //send the string for the first time
 HAL_UART_Transmit_DMA(&huart2,(uint8_t*)StringToSend, strlen(StringToSend));
 while (1)
 {
   /* USER CODE END WHILE */
   /* USER CODE BEGIN 3 */
 }
 /* USER CODE END 3 */
}
/**
 * @brief System Clock Configuration
 * @retval None
 */
```

The interrupt is called when the previous transmit is completed so we need to delay the sending so we don’t flood the receive and can make sure that our UART with DMA is working as planned. Callback delays the transmit by one second and then sends the same string again.

```
void HAL_UART_TxCpltCallback (UART_HandleTypeDef *huart)
{
	HAL_Delay(1000);    //wait for 1 second
	//USART2 interrupt priority has to more than zero so we don't get into
	//infinite loop with hal_delay
	//send again
	HAL_UART_Transmit_DMA(&huart2,(uint8_t*)StringToSend,
strlen(StringToSend));
}
```

4. Execution

The video was taken before I added \n to the end of the string. So in the video it prints to the same row instead of printing the next string to a new row.
Link to the video: Homework4

III. Exercise 2

General functionality:

Write name of all group members to lcd with help of premade library. Then names in alphabetical order will scroll on screen every one second.

Implementation:

1. **PIN lookup**

Pins are initialized as followed to support lcd-screen

Peripheral	**PIN**	**I/O**
LCD_BL_ON	PA4	GPIO output
LCD_E	PB1	GPIO output
LCD_RS	PB2	GPIO output
LCD_D4	PB12	GPIO output
LCD_D5	PB13	GPIO output
LCD_D6	PB14	GPIO output
LCD_D7	PB15	GPIO output

2. **Board configuration**

No further configurations were needed.

3. **Code implementation**

Control of lcd is a very straightforward process because there is premade library for that. To include library it's required to add corresponding header and source files. Then just simply type **#include** "PMDB16_LCD.h" you have all the necessary functions for lcd control.

At start it is mandatory to initialize lcd. We also define an array that includes names that we want to show on lcd. After that we can start by showing first name. It's easier to show it before the loop because the format is different than in other cases.

```
char ArrayToPrint [5][16] = {"Adrien", "Constantijn", "Heikki", "Nemo", "Osmo"};
// Init screen and show first name
lcd_initialize();
lcd_backlight_ON();
lcd_println(ArrayToPrint[0], 1);
HAL_Delay(1000);
lcd_clear();
```

After that we can move to the main loop. It does not contain anything special. Just show two names and wait for a second. Then move the lower name up and take next name to lower one.

```
while (1)
{
        for(int i = 0; i < 5; i++){
                // Second name is one index higher
                int SecondIndex = i;
                SecondIndex ++;
                // Second index is higher than num of elements in the list.
                if(SecondIndex >= 5){
                        SecondIndex = 0;
                }
                lcd_println(ArrayToPrint[i], 0);
                lcd_println(ArrayToPrint[SecondIndex], 1);
                HAL_Delay(1000);
                lcd_clear();
        }
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
}
```

4. Execution

Link to the video: Homework 4b

IV. Professors comments

EXERCISE 1 :
EXERCISE 2 :