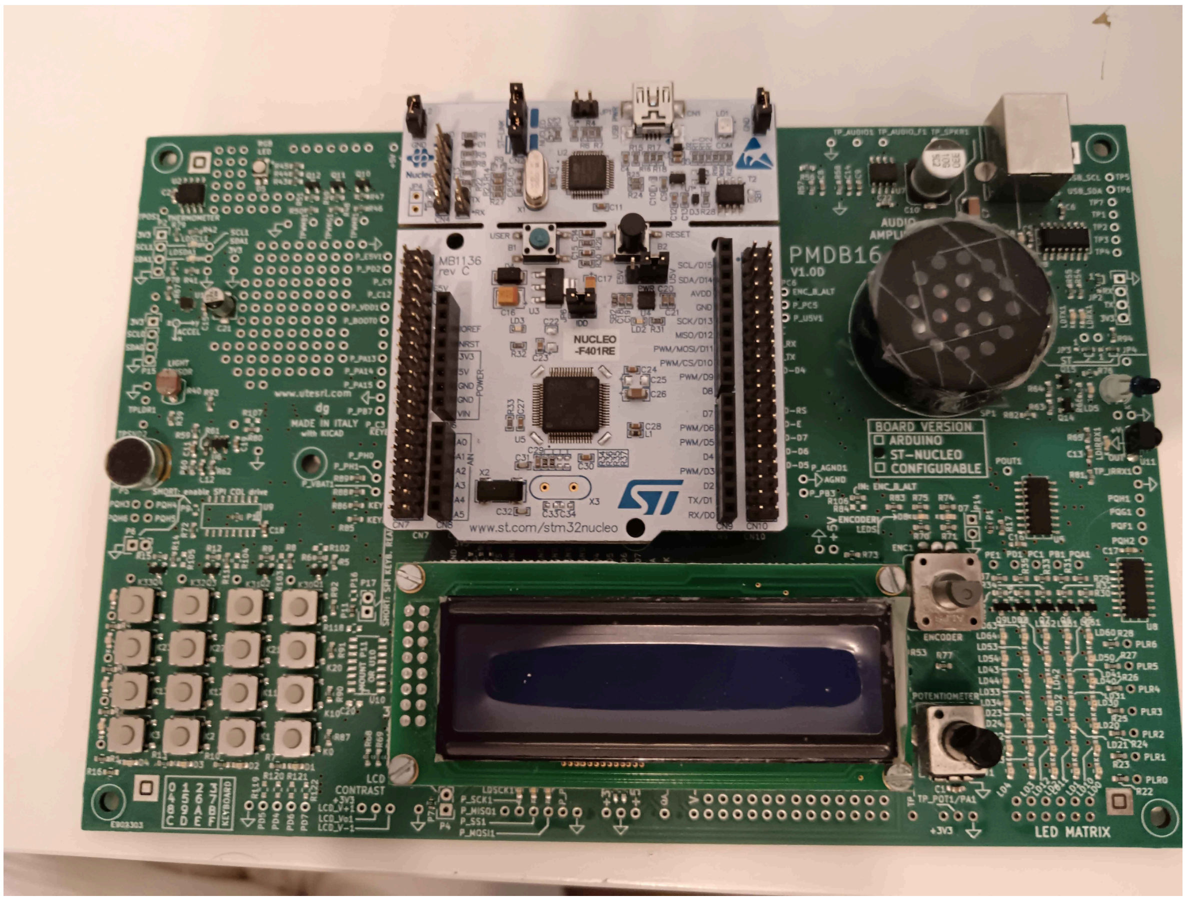
Computer Science Team B4

Sensor system

Homework Report



HOMEWORK NUMBER 9

2024 - 2025

POLITECNICO
MILANO 1863

Summary

I. Members work

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

II. Exercise 1

General functionality:
The aim of this project is to display multiple letters on a LED matrix. To achieve this, a timer of 4ms is configured to display each row of LEDs of the letter, and a second timer of 1sec is configured to change the letter displayed.

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

1. **PIN lookup**

The SPI protocol requires three lines (SCK, MOSi and SS in Transmit Only Master mode), hence two PINs must be configured:

PIN	**I/O**
SCK	PA5
MOSI	PA7
SS	PB6 (to configure manually in GPIO_OUTPUT)

2. **Board configuration**

SPI configuration
In the ***connectivity > SPI1*** tab, we enable the SPI communication to work in the mode ***"Transmit Only Master"***. Furthermore, we set up the DMA TX in the ***DMA Settings*** tab and we also enable the SPI1 event and error interrupt, under the ***NVIC Settings*** tab.

TIM configuration
Finally, we configure the timer of 1s and 4ms, as demonstrated in previous projects.

3. Code implementation

First, we include a header file for some functions (*memcpy principally*) , we define some structs to represent the letters and the matrix, and we define some private variables (variables to represent letters and alphabet, which is necessary to change the letter displayed)

```
/* USER CODE BEGIN Includes */
#include <string.h>
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/** Struct of a letter **/
typedef uint8_t letter_t[5];

/** Struct to display all the columns of the letter **/
typedef struct {
       uint8_t index;
       letter_t letter;
} display;

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define GET_LETTER_BYTES(col1,col2,col3,col4,col5)
{{col1<<1,16},{col2<<1,8},{col3<<1,4},{col4<<1,2},{col5<<1,1}}
#define SS_PIN                GPIOB, GPIO_PIN_6
#define TWO_BYTES             2
#define MAX_COLUMN_MATRIX     4
#define MAX_ALPHABET          2
/* USER CODE END PD */

. . .

/* USER CODE BEGIN PV */
display matrix; // Matrix variable
letter_t letterA =  GET_LETTER_BYTES(31, 36, 68,  36, 31);
letter_t letterI =  GET_LETTER_BYTES(65, 65, 127, 65, 65);
letter_t *alphabet [2] = {&letterA, &letterI}; // Just 2 letter but can be done with all
the letters
int index_alphabet = 0; // Index for when we change the letter to display
/* USER CODE END PV
```

The define "GET_LETTER_BYTES" represent the letters with bytes for each columns, so it is easier to represent letters when we know the value of each row of the matrix

As explained above, we first make a timer callback and ***in the first timer of 4ms, we display each column of the current letter in the matrix variable***. The speed of display is so fast that our eyes see all the columns light up at the same time even if one column is light up at a time. In the timer, we activate the SS pin before lightning up the LEDs to have the matrix to work as a slave. After that we reset the state of the SS pin.

We also have ***a second timer of 1 sec that changes the current letter in the display variable.***

```c
/**
* @brief Timers Callback
*        Timer of 1 sec : Change the letter to be displayed
*        Timer of 4 ms  : Display each columns of the letter on the LED matrix
*/
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
 /** 1sec Timer (to change the letter) **/
 if (htim->Instance == htim3.Instance)
 {
        ChangeLetter();
 }
 /** 4ms Timer (to display the letter) **/
 if(htim->Instance == htim1.Instance)
 {
        HAL_GPIO_WritePin(SS_PIN, GPIO_PIN_SET);
        // For each column
        if(++matrix.index > MAX_COLUMN_MATRIX)
        {
               matrix.index = 0;
        }
        HAL_GPIO_WritePin(SS_PIN, GPIO_PIN_RESET);
        HAL_SPI_Transmit_DMA(&hspi1, matrix.letter[matrix.index], TWO_BYTES);
 }
}
```

For the *"ChangeLetter"* function, we just ***change the current letter of the matrix by the next one when the timer is triggered***. We just have a condition to not have the index above the maximum of letters.

```c
/**
* @brief Change the letter of the matrix
*/
void ChangeLetter (void)
{
        // If the index is higher than the max of the alphabet
        if(++index_alphabet == MAX_ALPHABET)
        {
               index_alphabet = 0;
        }
        // Change the letter by the next one
        memcpy(matrix.letter, *alphabet[index_alphabet], sizeof(*alphabet[index_alphabet]));
}
```

Finally, we just have to initialize the first letter to be displayed and to start the timers in the main function outside of the infinite while loop.

```
. . .
/* USER CODE BEGIN 2 */
      /** Copy the first letter in the struct to be displayed **/
      memcpy(matrix.letter, alphabet[index_alphabet], sizeof letterA);
      /** Start the Timer (4ms) **/
      HAL_TIM_Base_Start_IT(&htim1);
      /** Start the Timer (1sec) **/
      HAL_TIM_Base_Start_IT(&htim3);
 /* USER CODE END 2 */
. . .
```

Now we are ready to run the code!

Note that '. . .' is a replacement for irrelevant code.

4. Proof of project

Link to the video : https://youtu.be/BOJ2sM8D-Xw