

# **PRÁCTICA 3**

## **Índice**

1. Objetivos.
2. Introducción.
3. Desarrollo de la práctica.
4. Conclusión.

## **1. Objetivos**

- Ilustrar el conversor Analógico – Digital en STM32.
- Digitalizar la botonera analógica del LCD-Shield.
- Mostrar el resultado en el LCD del shield.
- Ilustrar el puerto serie en STM32.
- Enviar la información de la botonera al PC a través de un puerto serie virtual.
- Monitorizar el estado de la botonera en un terminal en el PC.

## **2. Introducción**

En esta práctica abordaremos dos conceptos clave:

-La conversión de analógico a digital, que implica transformar una señal analógica dentro de un rango específico de voltaje en una secuencia de números digitales discretos con un número fijo de bits. Para trabajar con esto, utilizaremos la botonera integrada en el shield LCD, que usamos en la práctica 2. Este circuito está diseñado de manera que al presionar diferentes botones, se alteran los valores de las resistencias en un divisor de tensión, obteniendo distintos valores analógicos que utilizaremos en la conversión, empleando los conversores analógico-digital presentes en la placa.

-La comunicación serie, una forma sencilla de permitir la comunicación entre microcontroladores. En los ordenadores, estos se conocen como puertos "COM". Hay dos líneas de datos, Tx y Rx, que deben conectarse de forma cruzada entre los dispositivos, es decir, la Tx de un dispositivo debe conectarse a la Rx del otro y viceversa. No hay una línea de reloj, ya que el transmisor y el receptor deben estar configurados con el mismo tiempo de bit, conocido como "baudrate".

### 3. Desarrollo de la práctica

- En la fase 1 queremos mostrar en el display el valor digital, la tensión, en voltios, y el botón pulsado.

Primero, configuramos el reloj “HCLK” a 80MHz y los pines del microcontrolador basándonos en la siguiente tabla proporcionada en la práctica:

| PIN  | ETIQUETA |
|------|----------|
| PA2  | Led_LCD  |
| PA15 | E_LCD    |
| PB1  | D6_LCD   |
| PB4  | D5_LCD   |
| PB2  | RS_LCD   |
| PA4  | D7_LCD   |
| PA3  | D4_LCD   |

Generamos código tras configurarlo y añadimos los ficheros adjuntos a la práctica, “hd44780.c” en la carpeta src y “hd44780.h” en la carpeta inc de nuestro proyecto. Para finalizar hay que añadir en “main.c”, en la sección de código de “/\* USER CODE BEGIN Includes \*/” lo siguiente:

```
#include "hd44780.h"
```

Realizamos una prueba para comprobar si todo se ha ido bien. Añadimos en “main.c”, en la sección de “/\* USER CODE BEGIN 2 \*/” el siguiente código:

```
lcd_reset();  
lcd_display_settings(1, 0, 0);  
lcd_clear();
```

```
lcd_print("Funciona!");
```

En la segunda parte de la Fase 1 mostraremos en el display la lectura del ADC. Para ello, añadimos en “main.c”, dentro del while(1) el siguiente código:

```
uint16_t sample;
```

```
HAL_ADC_Start(&hadc1);  
HAL_ADC_PollForConversion(&hadc1, 100);  
sample = HAL_ADC_GetValue (&hadc1);
```

```
moveToXY(0, 0);  
lcd_print("Sp. ");  
writeIntegerToLCD(sample);
```

```
HAL_Delay(100);
```

La tercera parte de la Fase 1 implica mostrar el voltaje en milivoltios utilizando el valor leído del ADC. Para ello, añadimos en “main.c”, dentro del while(1), el siguiente código, reemplazando el anterior:

```
uint16_t sample;

HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, 100);
sample = HAL_ADC_GetValue (&hadc1);

moveToXY(0, 0);
lcd_print("Sp. ");
writeIntegerToLCD(sample);

moveToXY(1, 0);
writeIntegerToLCD(sample*3300/4095);
lcd_print("mV");

HAL_Delay(100);
```

La cuarta y última parte de la Fase 1 consiste en indicar el botón que se está presionando. Para ello, crearemos una función en “main.c”, en la sección de “/\* USER CODE BEGIN 4 \*/” con el siguiente código:

```
uint8_t getButton(uint16_t sample){

    if (sample > 4000){

        return 0;

    } else if (sample > 2500){

        return 4;

    } else if (sample > 1500){

        return 3;

    } else if (sample > 500){

        return 2;

    } else {

        return 1;

    }

}
```

```
}
```

Añadimos en “main.c”, en el while(1), el siguiente código, suprimiendo el de la anterior parte:

```
uint16_t sample;  
uint8_t button;
```

```
HAL_ADC_Start(&hadc1);  
HAL_ADC_PollForConversion(&hadc1, 100);  
sample = HAL_ADC_GetValue (&hadc1);
```

```
button = getButton(sample);
```

```
moveToXY(0, 0);  
lcd_print("Sp. ");  
writeIntegerToLCD(sample);
```

```
switch (button) {
```

```
    case 1:
```

```
        lcd_print(" - Right");  
        break;
```

```
    case 2:
```

```
        lcd_print(" - Up  ");  
        break;
```

```
    case 3:
```

```
        lcd_print(" - Down ");  
        break;
```

```
    case 4:
```

```
        lcd_print(" - Left ");  
        break;
```

```
    default:
```

```
        lcd_print(" - None ");  
        break;
```

```
}
```

```
moveToXY(1, 0);
writeIntegerToLCD(sample*3300/4095);
lcd_print("mV");
```

```
HAL_Delay(100);
```

- En la fase 2 queremos mostrar por consola lo que mostramos en el display. Hay que configurar los pines PB7 y PB6 como USART1\_RX y USART2\_TX respectivamente:

```
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, HAL_GPIO_ReadPin(B1_GPIO_Port,
B1_Pin));
```

Luego , añadiremos en “main.c”, en “/\* USER CODE BEGIN PV \*/” el siguiente código:

```
#ifndef __GNUC__
/* With GCC/RAISONANCE, small printf (option LD Linker->Libraries->Small printf
set to 'Yes') calls __io_putchar() */
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
PUTCHAR_PROTOTYPE{

    HAL_UART_Transmit(&huart1,(uint8_t *) &ch,1,1000);
    return ch;

}
```

Por último, queda añadir en “main.c”, en el while(1), el siguiente código, suprimiendo el de la anterior Fase 1:

```
uint16_t sample;
uint8_t button;

HAL_ADC_Start(&hadc1);
HAL_ADC_PollForConversion(&hadc1, 100);
sample = HAL_ADC_GetValue (&hadc1);

button = getButton(sample);

moveToXY(0, 0);
lcd_print("Sp. ");
writeIntegerToLCD(sample);

switch (button) {

    case 1:
```

```

        lcd_print(" - Right");
        break;

    case 2:

        lcd_print(" - Up ");
        break;
    case 3:

        lcd_print(" - Down ");
        break;
    case 4:

        lcd_print(" - Left ");
        break;

    default:

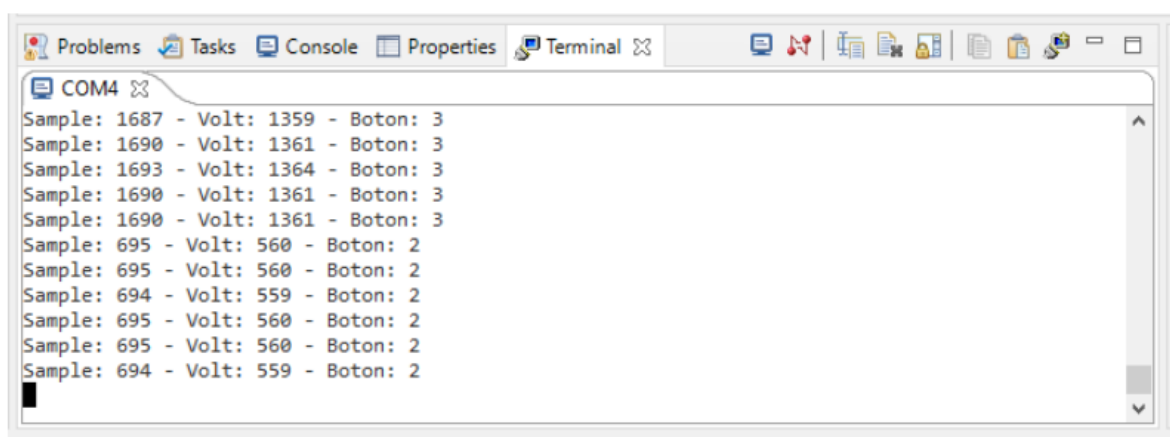
        lcd_print(" - None ");
        break;
    }
    moveToXY(1, 0);
    writeIntegerToLCD(sample*3300/4095);
    lcd_print("mV");

    printf("Sample: %d - Volt: %d - Boton: %d \r\n", sample, sample*3300/4095, button);

    HAL_Delay(100);

```

Una vez realizados los cambios, tiene que salirnos esto:



```

COM4
Sample: 1687 - Volt: 1359 - Boton: 3
Sample: 1690 - Volt: 1361 - Boton: 3
Sample: 1693 - Volt: 1364 - Boton: 3
Sample: 1690 - Volt: 1361 - Boton: 3
Sample: 1690 - Volt: 1361 - Boton: 3
Sample: 695 - Volt: 560 - Boton: 2
Sample: 695 - Volt: 560 - Boton: 2
Sample: 694 - Volt: 559 - Boton: 2
Sample: 695 - Volt: 560 - Boton: 2
Sample: 695 - Volt: 560 - Boton: 2
Sample: 694 - Volt: 559 - Boton: 2

```

## **4. Conclusión**

Todas las fases se completaron con éxito siguiendo las instrucciones detalladas en la hoja de la práctica.

En cuanto a mi experiencia personal, la práctica fue breve pero valiosa, pues he aprendido a leer los datos enviados mediante la botonera del LCD a través del puerto serie y a convertir esos datos analógicos en formato digital.