Informe de Práctica 3: Manejo de un *liquid crystal display* (LCD)

Máximo García Aroca

May 3, 2024

1 Introducción

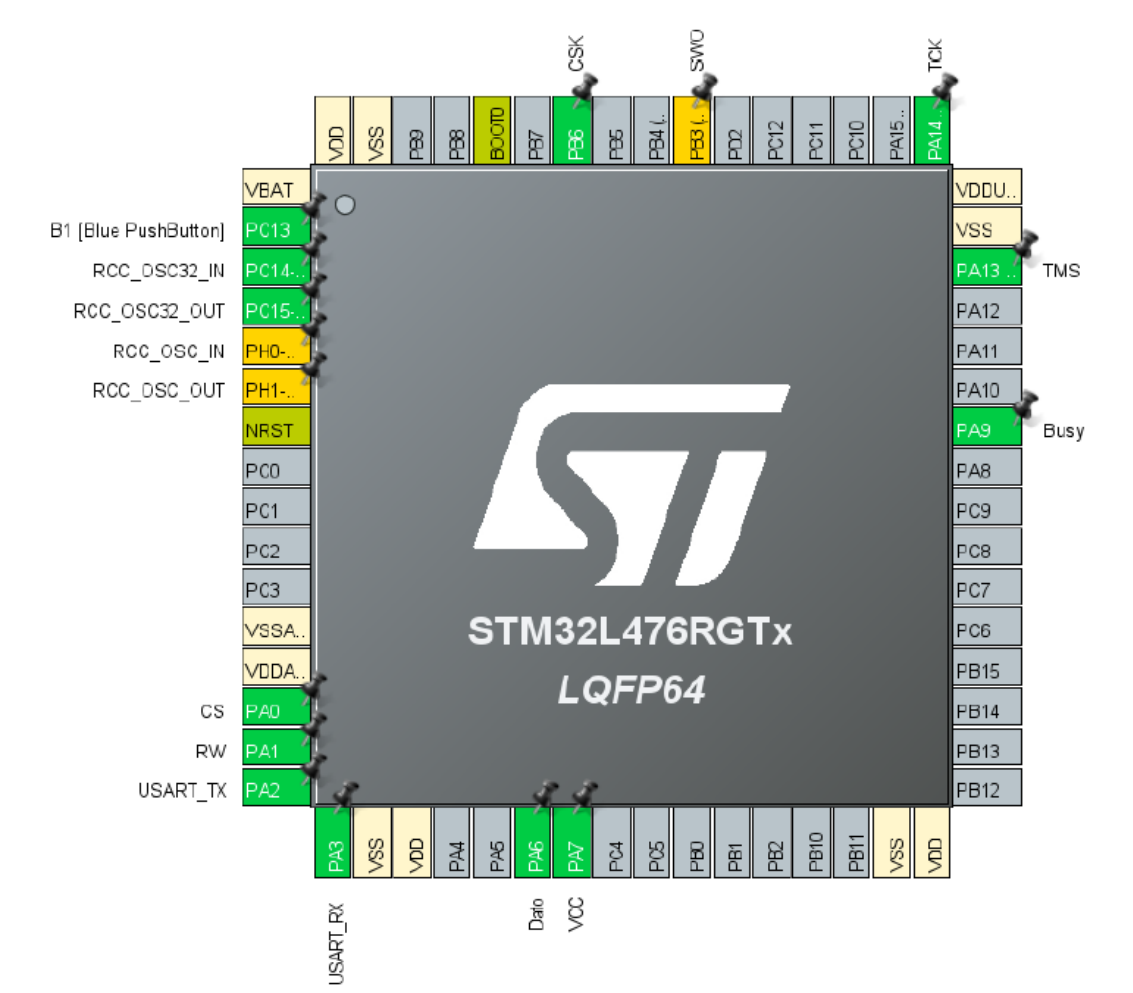
En esta práctica hemos conectado un *display* LCD según los manuales aportados y el guión de la práctica al microcontrolador L47RG para mostrar el mensaje "HOLA!!!".

2 Diseño físico

El conexionado es el siguiente:

- Pin 1: alimentación 5V.
- Pin 2: GND.
- Pin 3: Chip select.
- Pin 4: Selección comando o dato (C/D).
- Pin 5: señal de ocupado (BUSY).
- Pin 6: señal de reloj SCK.
- Pin 7: señal de entrada de datos serie.

Esta ha sido la configuración que hemos programado mi compañero y yo en STM32CubeMX:

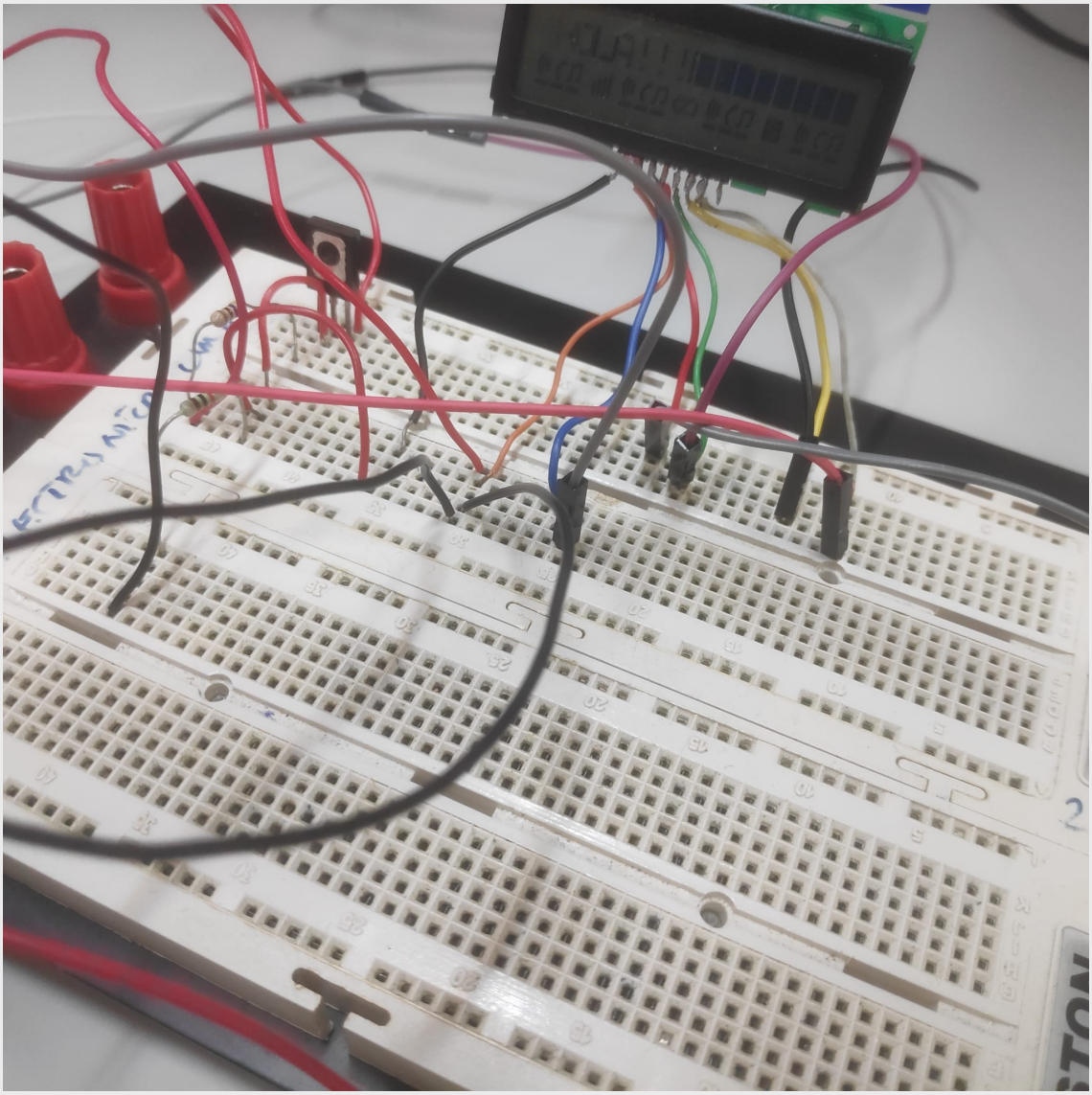


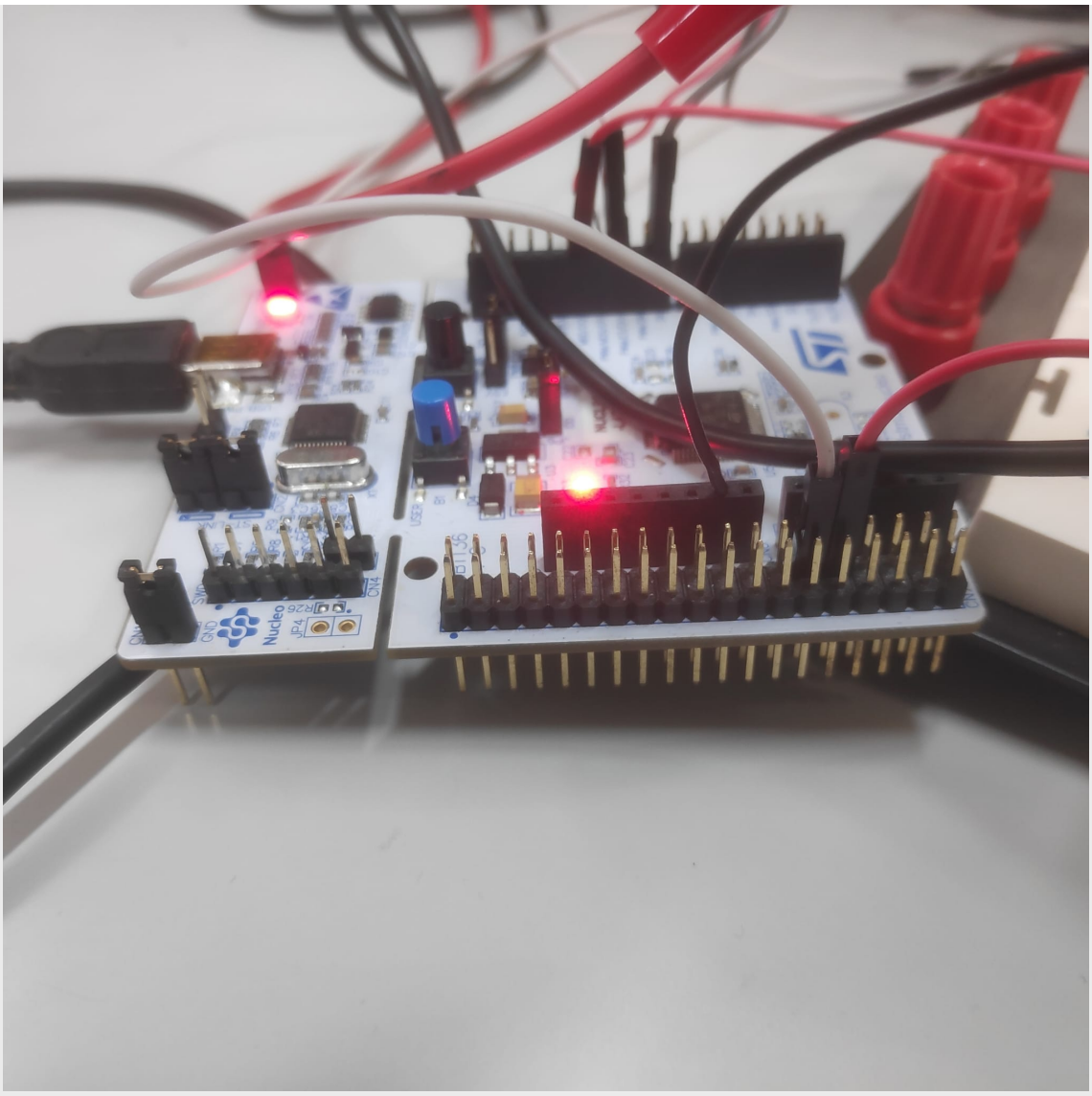
Donde *RW* hace referencia a *C/D* y *CSK* a *SCK*.

Para el montaje hemos usado el manual del *display LCD* para conectar 5 de los pines al microcontrolador y los otros dos de alimentación y *GND* al generador de voltaje.

Como en la práctica se nos indica había que hacer uso de un transistor BD679 y una resistencia de 400 ohmios.

Este ha sido el montaje:




Nucleo
GND
JP4
CN4

3 Código

Se nos proporciona una serie de funciones a implementar. Y también una serie de comandos que usa el *display*:

Mnemónico	Operación	Código Hexadecimal
SFF	Set Frame Frequency	10-14
SMM	Set Multiplexing Mode	18-1F
DISP OFF	Display Off	08
DISP ON	Display On	09
LDPI	Load Data Pointer with Inmediate	80-B1,C0-F1
SRM	Set Read Mode	60-63
SWM	Set Write Mode	64-67
SORM	Set Or Mode	68-6B
SANDM	Set AND Mode	6C-6F
SCML	Set Character Mode with Left entry	71
SCMR	Set Character Mode with Right entry	72
BRESET	Bit Reset	20-3F
BSET	Bit Set	40-5F
CLCURS	Clear Cursor	7C
WRCURS	Write Cursor	7D
STOP	Set Stop Mode	01

Tabla 1. Lista de Comandos del uPD7229A.

Lo primero de todo ha sido inicializar el *display*, para eso hemos indicado que la entrada en principio era de comandos poniendo a 0 *RW*. Con **LCD_enviarByte** enviámos el comando.

```
94  void LCD_inicialización(int banco, int entrada_caracteres){
95      HAL_GPIO_WritePin(GPIOA, CS_Pin, GPIO_PIN_SET);
96      HAL_Delay(10);
97      HAL_GPIO_WritePin(GPIOA, CS_Pin, GPIO_PIN_RESET);
98
99      //SFF
100     uint8_t sff = 0x11;
101     LCD_enviarByte(1, sff);
102
103     //Banco0
104     uint8_t banco0 = 0x1F;
105     if(banco == 0){
106         LCD_enviarByte(1,banco0);
107     }
108
109     uint8_t disp_on = 0x09;
110     LCD_enviarByte(1, disp_on);
111
112     uint8_t scmbr = 0x72;
113     uint8_t sandm = 0x6f;
114
115
116     LCD_enviarByte(1, sandm);
117     LCD_enviarByte(1, scmbr);
118
119 }
```

En **LCD_inicialización** lo que hacemos es:

1. Activamos ChipSelect.

2. Encendemos el *display* con el comando *sff*.
3. Definimos el banco que vamos a utilizar.
4. Definimos el *display* como *"on"*.
5. Por último con scmbr hará que el byte entre por la derecha y sandm, por la izquierda.

```
66  void LCD_ocupado(void){
67      while(!HAL_GPIO_ReadPin(GPIOA, Busy_Pin));
68  }
```

LCD_ocupado solo se queda en bucle consultando si el *LCD* está libre leyendo el pin *"Busy"*.

```
70  void LCD_enviarByte(uint8_t comando, uint8_t byte){
71      LCD_ocupado();
72
73      if(comando == 1){
74          HAL_GPIO_WritePin(GPIOA, RW_Pin, GPIO_PIN_SET);
75      }else{
76          HAL_GPIO_WritePin(GPIOA, RW_Pin, GPIO_PIN_RESET);
77      }
78
79      for(int j = 0; j <= 7; j++){
80          HAL_GPIO_WritePin(GPIOB, CSK_Pin, GPIO_PIN_RESET);
81          HAL_Delay(1);
82          if(byte & 0x80){
83              HAL_GPIO_WritePin(GPIOA, Dato_Pin, GPIO_PIN_SET);
84          }else{
85               HAL_GPIO_WritePin(GPIOA, Dato_Pin, GPIO_PIN_RESET);
86          }
87          HAL_GPIO_WritePin(GPIOB, CSK_Pin, GPIO_PIN_RESET);
88          HAL_GPIO_WritePin(GPIOB, CSK_Pin, GPIO_PIN_SET);
89
90          byte = (byte << 1);
91      }
92  }
```

LCD_enviarByte comprueba si el *display* está libre. Cuando lo esté comprueba si el Byte enviado es un comando o un dato. Después envía el byte escribiendo en el $Dato_Pin$.

Estos son unos vídeos que dejo para que se vea el funcionamiento del *display*.

```
155  while (1)
156  {
157        LCD_inicialización(0,1);
158
159        LCD_enviarByte(0,'!');
160        HAL_Delay(10);
161        LCD_enviarByte(0,'!');
162        HAL_Delay(10);
163        LCD_enviarByte(0,'!');
164        HAL_Delay(10);
165
166        LCD_enviarByte(0,'A');
167        HAL_Delay(10);
168        LCD_enviarByte(0,'L');
169        HAL_Delay(10);
170        LCD_enviarByte(0,'O');
171        HAL_Delay(10);
172        LCD_enviarByte(0,'H');
173        HAL_Delay(10);
174        if (!HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13)){
175            HAL_Delay(800);
176            LCD_enviarByte(1, 0x08);
177            while(1)
178            {
179                 if (!HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13)){
180                     break;
181                 }
182            }
183        }
184  }
185  /* USER CODE END 3 */
```

En el *"while"* primero se inicializa el *display* para después enviar byte a byte los *"char"* por el pin *"Dato"*. El último *"if"* sirve para implementar una funcionalidad que hace que si se pulsa el botón de usuario de nuestro microcontrolador se apaga la pantalla, y si se vuelve a pulsar se enciende y prosigue con mostrar el mensaje.

Estos son unos vídeos que dejo para que se vea el funcionamiento del *display*.

`https://drive.google.com/file/d/1emGAVylVMtxRbOj8ffS_6nuYGW2CAK6b/view?usp=drive_link`

`https://drive.google.com/file/d/1Hwe1uNEnIik6LqIZ8Hda7-ViKZWryS5J/view?usp=drive_link`