



MICROCONTROLADORES

MANEJO DE UNA LCD EN UN
MICROCONTROLADOR

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
CAMPUS SAN JUAN DEL RÍO
FACULTAD DE INGENIERÍA

INGENIERÍA ELECTROMECÁNICA

Desarrollado por:
Morán Morán Jesús Alberto — 292727



TABLA DE CONTENIDOS

MANEJO DE UNA LCD EN UN MICROCONTROLADOR	4
OBJETIVO —.....	4
INTRODUCCIÓN.....	4
METODOLOGÍA.....	7
RESULTADOS.....	8
I. Programación del controlador (Creación del archivo).	9
I.I Programación del controlador (programación)	10
II. Simulación de las instrucciones.....	12
RESULTADOS FÍSICOS	13
CONCLUSIONES.....	15
REFERENCIAS	15
MATERIAL ADICIONAL	15
Materiales.....	16
TABLA 1. <i>Materiales para simulación</i>	16
TABLA 2. <i>Materiales para la elaboración física</i>	16
Código de las instrucciones.....	16
Videos de complemento.....	19

TABLA DE FIGURAS

Figura 1 Placa de desarrollo con un microcontrolador STM32 F411CEU6®.....	4
Figura 2 Arquitectura de un microcontrolador (Velázquez, 2022).	5
Figura 3 Programador ST-LINK V2.....	5
Figura 4 Pantalla de cristal líquido (Mecafenix, 2017).....	6
Figuras 5 a) Configuración Pull-Up, b) Configuración Pull-Down.....	7
Figura 6 Proceso para la experimentación.	8
Figuras 7 a) Interfaz de bienvenida, b) Creación de un nuevo proyecto.....	9
Figuras 8 a) Menú de opciones de microcontroladores, b) Definición del nombre del proyecto.....	9
Figuras 9 a) Configuración de las librerías, b) Búsqueda del archivo de programación de las instrucciones.	10
Figura 10 Configuración en los registros para la función "wcom".....	10
Figura 11 Activación de los timers.	11



Figura 12 Configuración de reloj para los puertos I/O A.....	11
Figura 13 Tabla de instrucciones.....	11
Figura 14 Depuración de código.....	12
Figura 15 Valores de la memoria RAM y Flash usada.....	12
Figura 16 Diagrama para la simulación.....	13
Figuras 17 a) Funcionamiento con pull-down 1, b) funcionamiento con pull-up.....	13
Figura 18 Conexiones realizadas en una placa de prototipos.....	13
Figura 19 Conexiones físicas realizadas, con el mensaje inicial.....	14
Figura 20 Depuración y carga de las instrucciones.....	14
Figura 21 a) Configuración pull-up, b) Configuración pull-down.....	14

MANEJO DE UNA LCD EN UN MICROCONTROLADOR

Objetivo — Configurar en un microcontrolador parámetros relacionados a una LCD, mediante la comunicación entre las terminales de ambos dispositivos, con la finalidad de observar distintos mensajes dependiendo del estado de dos de los interruptores de diferente configuración¹.

INTRODUCCIÓN

Un microcontrolador es un circuito integrado que, de forma general, su principal función es la de desarrollar las tareas repetitivas descritas mediante las instrucciones grabadas por programación en su memoria, logrando una alta eficiencia en cuanto al consumo energético, siendo esta característica una de sus mayores ventajas.

La programación de las instrucciones que deba seguir el microcontrolador normalmente se realiza en un IDE (Integrated Development Environment por sus siglas en inglés), mismo que dependerá del tipo de microcontrolador, cabe agregar que la mayoría de estos elementos a pesar de que pueden diferir en el método o lenguaje de programación, la codificación de las instrucciones de este tipo de máquinas se está basado en el sistema numérico hexadecimal.

Un ejemplo de microcontrolador embebido está en la tarjeta de desarrollo observada en la figura 1, dicha tarjeta pertenece a una extensa familia de microcontroladores desarrollados en arquitectura ARM Cortex® -M RISC, basados en 32 bits.

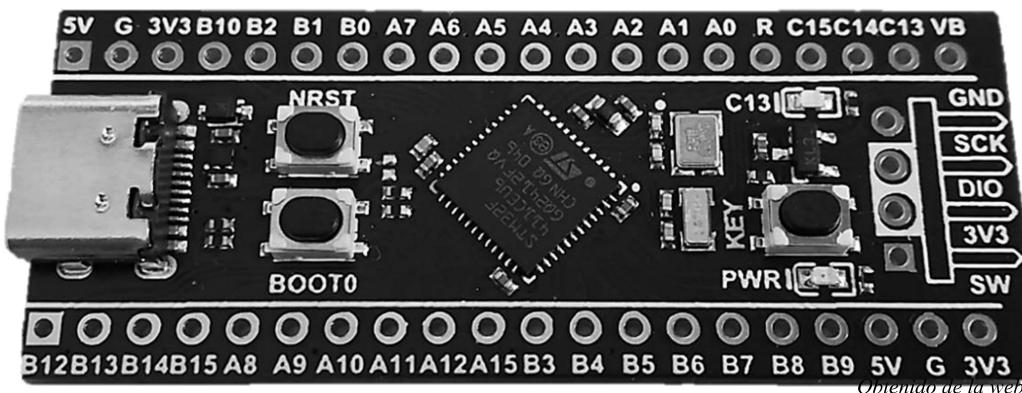


Figura 1 Placa de desarrollo con un microcontrolador STM32 F411CEU6®

Por lo general la arquitectura implementada en la mayoría de los microcontroladores es del tipo Harvard, de donde se sabe que es capaz de ejecutar instrucciones al doble de velocidad que la arquitectura Von Neumann, por otro lado, una segunda clasificación de la arquitectura de un microcontrolador es por el tamaño de ancho de bits de sus buses de datos y/o memoria (E-Marmolejo, 2017).

Un microcontrolador puede estar diseñado para diferentes propósitos, pero de forma general un dispositivo de esta naturaleza cuenta con al menos 3 de los componentes mencionados a continuación;

- **La Unidad Aritmética y Lógica** (ALU por sus siglas en inglés Arithmetic Logic Unit), es una unidad de control, y registros, compuesta por el arreglo de elementos de combinación electrónica, mismo que son los que permiten la realización de operaciones lógicas (OR, AND, XOR, NOT, etcétera), aritméticas (suma, resta, multiplicación) y misceláneas (operación de transferencia de bits)
- **La unidad de control** es aquella que permite el manejo de la lógica necesaria para la decodificación y ejecución de las instrucciones.
- **Las memorias** son parte importante en la estructura de un microcontrolador es, posible observar al menos dos tipos de memoria, que son la memoria ROM o memoria de almacenamiento no volátil y la memoria RAM o memoria de almacenamiento volátil, como se muestra en la figura 2, la memoria ROM

¹ La configuración de los interruptores es diferente; pull-up y pull down para cada uno.

por lo general es usada para el almacenamiento de las instrucciones, mientras que la memoria RAM se suele usar para el almacenamiento de los datos obtenidos por la ejecución de las instrucciones. De esta forma se puede intuir que un microcontrolador puede contener en su interior un procesador, pero un procesador no puede contener un microcontrolador (E-Marmolejo, 2017).

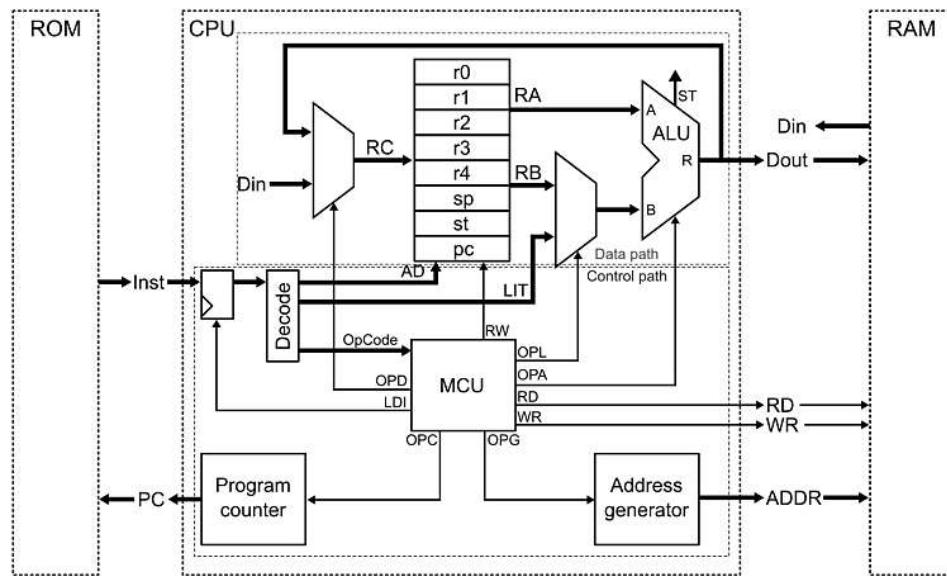


Figura 2 Arquitectura de un microcontrolador (Velázquez, 2022).

Dentro de las bondades que dispone un microcontrolador se tienen a los periféricos que lo componen, siendo así, algunos de sus periféricos los que a continuación se mencionan; por ejemplo, periféricos de comunicación (UART, I2C, SPI), los periféricos de adecuación de señales (ADC, DAC), y periféricos generales (Contador, Timer, Controlador de interrupciones, controlador del oscilador).

Los registros de un microcontrolador son espacios de memoria que sirven para el almacenamiento de datos, aunque existen unos registros especiales (SFR²) que internamente pueden considerarse como interruptores de control, permitiendo el manejo de algunos periféricos como los timer, los convertidores ADC, DAC, etcétera. Para acceder a ellos es necesario hacerlo según la información proporcionada por el fabricante.

Los métodos de programación dependerán del tipo de microcontrolador que se use, por ejemplo, existen placas de desarrollo que permite el uso de lenguajes de programación como Python, microphyton, C, C#, C++, por ello considerando un microcontrolador tipo STM32 F103 C8T6®, para su programación se requiere de elementos como un compilador, un IDE (STM32CubeIDE 1.10.1), y un programador como el de la figura 3.



Figura 3 Programador ST-LINK V2.

El STM32 F103C8T6® es un dispositivo que para programar requiere de la conexión de un programador, mismo que se conecta a un ordenador, el cual tiene la función de transmitir el código o instrucciones, para tal

² Por sus siglas en inglés Special Function Register.

efecto se requiere de la conexión de los puertos del programador denominados SWCLK, SWDIO, GND, 3.3 V, con los puertos CLK, DIO, GND y 3.3 V de la placa; aunque es importante señalar que el procedimiento de comunicación se debe ajustar al microcontrolador. Además, para la creación de un archivo de programación de instrucciones para la placa mencionada, inicialmente se requiere la creación del archivo directamente en el IDE, en este caso usando el lenguaje de programación C (sujeto a el tipo de microcontrolador).

Para el desarrollo de un archivo que contenga las instrucciones de operación, es necesario contemplar las operaciones lógicas, ya que estan sirven de apoyo para la elaboración de máscaras; las máscaras son recursos que permite la modificación bit a bit o extracción de forma selectiva la información contenida, las operaciones más usadas son las AND y OR.

TABLA 1. TABLA DE VERDAD DE LAS FUNCIONES LÓGICAS BÁSICAS

Entradas		AND	NAND	OR	NOR	EXOR
A	B	0	1	0	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	0	1	0	0
Funciones		A·B	$\overline{A} \cdot \overline{B}$	A+B	$\overline{A} + \overline{B}$	$A \oplus B$

Las capacidades de un microcontrolador son bastante extensas, de esta forma es posible realizar una configuración³ que permita el manejo electrónico de una pantalla de cristal líquido. Por ello es importante conocer las características de este elemento, dicha información puede estar contenida en la hoja de datos, misma que depende del tipo de configuración con el cuál se trabaja.

En el caso de una pantalla de cristal líquido de configuración 16x2, se tienen dos filas de 16 caracteres, donde cabe mencionar que el número de pixeles dependerá del tipo de construcción y modelo. Este dispositivo cuenta principalmente con 16 pines divididos en pines de alimentación, pines de control y pines de bus de datos bidireccional, además de los pines correspondientes a un led usado para la iluminación de retroalimentación, como se muestra en la figura 4.

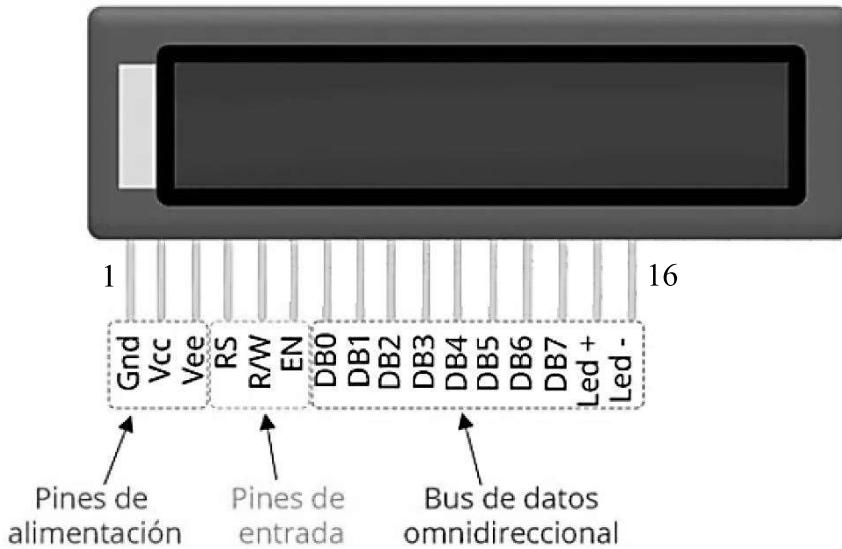


Figura 4 Pantalla de cristal líquido (Mecafenix, 2017).

A continuación, se detalla el propósito de cada una de las terminales comenzando de izquierda a derecha,

Pines de alimentación: la terminal 1 corresponde a la tierra, seguido de la terminal de alimentación positiva (5 voltios dependiendo del modelo), la terminal 3 está destinada a el control de contraste; en esta terminal es posible

³ Considere que la configuración para este documento tiene el propósito de desplegar información en forma de caracteres en la LCD.

realizar un divisor de voltaje mediante la colocación de un potenciómetro destinando la parte central a la conexión Vee de la terminal, mientras los dos terminales del potenciómetro a las terminales de alimentación.

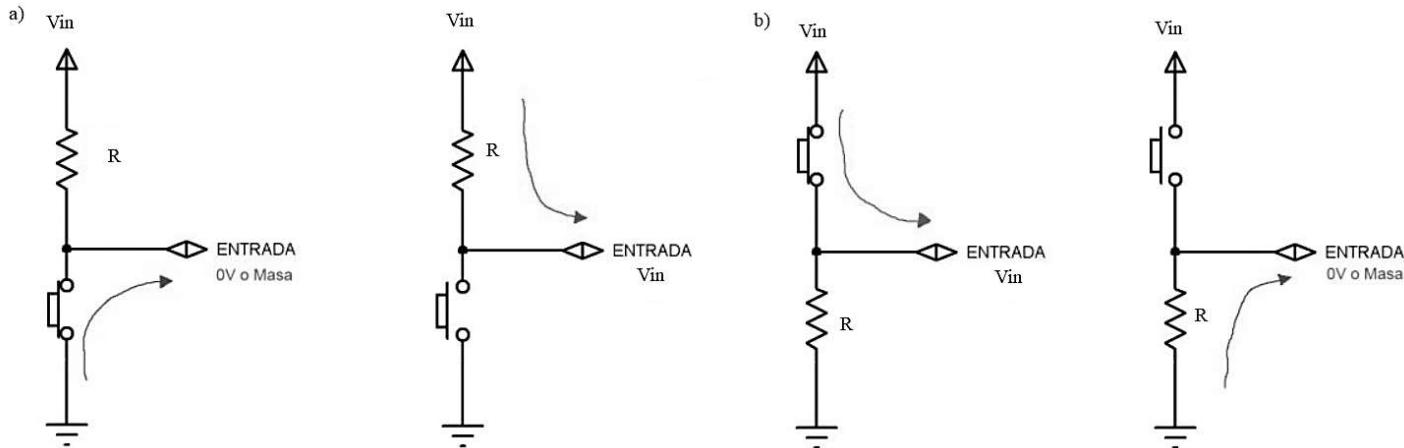
Pines de entrada: la terminal RS de la LCD permite la configuración de selección de registro de control de datos (0) o registro de datos (1), la segunda terminal de los pines de entrada denominada RW permite la escritura de un dato en la LCD (0) o la lectura de un dato (1), finalmente para los pines de entrada, la terminal del “Enable” permite la habilitación donde al existir la habilitación (1) es posible leer o escribir datos en la LCD

Pines de bus de datos omnidireccional: Comprenden D0 a D7 y son usados para la comunicación, pudiendo ser de 8 bits o usando los bits más significativos (4 bits de D4 a D7).

Además de lo anterior, con el uso de un microcontrolador es posible detectar cambios de estado lógico (0 o 1), en el caso de un pin digital la aplicación de una resistencia de pull-up o pull-down permite que dicho pin digital no se encuentre en el estado flotado, el cuál puede representar grandes problemas ya que cualquier alteración en el entorno produce cambios en el estado lógico.

Un arreglo electrónico para evitar el estado flotado es la aplicación de resistencias en Pull-up o resistencia en Pull-Down. Para la configuración de una entrada en Pull-Up se condiciona a tener un estado alto cuando no hay pulsación en el botón, mientras que al pulsar el botón la salida se aterraza directamente a la masa o tierra, colocando la salida en un estado bajo (0).

Para el arreglo en Pull-Down se condiciona a la salida a mantener un estado bajo hasta el momento en el que se presiona el botón, al presionar es circuito conecta la fuente cambiando a un estado alto (1), en las figuras 5 – a y 5 – b se muestran dichas configuraciones, donde es importante mencionar que para el microcontrolador no es necesario la colocación de la resistencia ya que esta se encuentra internamente.



Figuras 5 a) Configuración Pull-Up, b) Configuración Pull-Down.

Al ser otro dispositivo con memoria la LCD, es necesario el control de algunos comandos, mismos que están contenidos en su hoja de datos. Esencialmente se requiere de la manipulación de aspectos como la limpieza de la pantalla, la entrada al modo de configuración, así como el encendido y apagado de la pantalla.

METODOLOGÍA

En el desarrollo de un proceso de estudio de la teoría proporcionada, con relación a el manejo de las salidas de un microcontrolador, se agrega el siguiente diagrama de flujo, ver la figura 6, donde se detalla el procedimiento resumido de elaboración de las pruebas.

Inicialmente para la ejecución de una práctica se requiere de una planeación, misma indica los aspectos esperados como resultados, seguido de este proceso se recurre a la programación, etapa donde se describen las instrucciones basándose en los requerimientos, dentro de este proceso es importante realizar la compilación, con

el objetivo de detectar errores de programación, en caso de existir se regresa de nuevo a la programación, por otro lado en caso que el código generado no tenga errores se procede al siguiente punto que es la simulación.

En la simulación se realiza el diseño de las conexiones mismas que se basan en la hoja de datos de ambos dispositivos (tarjeta de desarrollo y LCD), en esta etapa se carga el código generado con las instrucciones. Dada la similitud con la realización física este punto permite observar errores, mismos que en caso de existir pueden ocasionar el retorno al punto de programación.

En la implementación física se repite un procedimiento similar al de simulación, aunque es importante mencionar que los elementos usados no son ideales, los cuales también pueden sufrir grandes daños en caso de una conexión errónea.

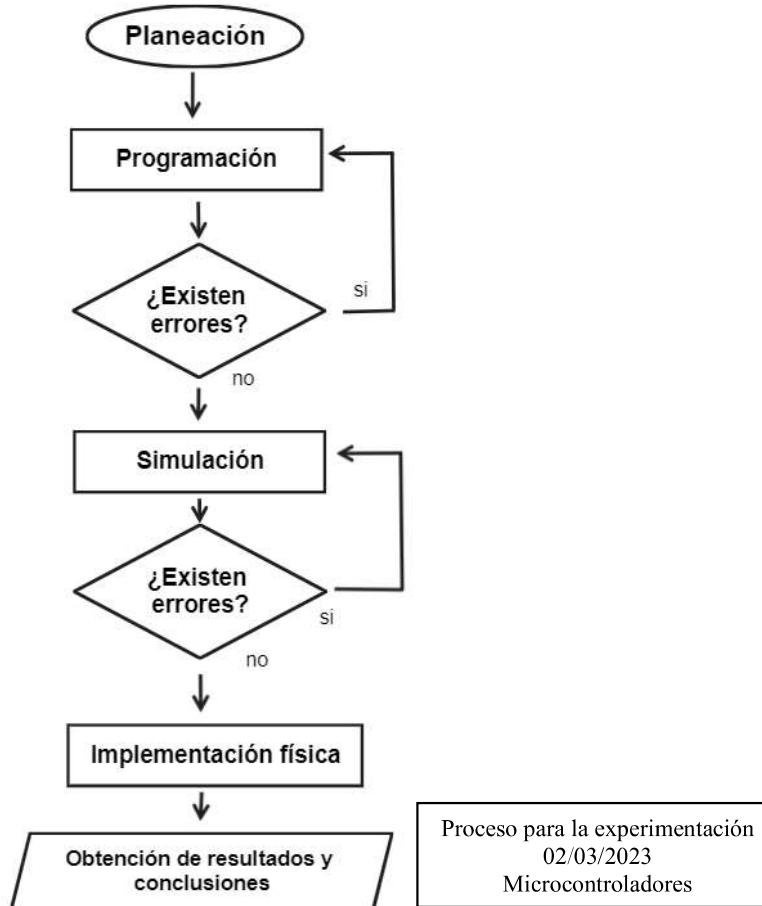


Figura 6 Proceso para la experimentación.

En resumen, en la programación se hace el manejo de las instrucciones, mediante el acceso a los registros, en la simulación se diseñan y realizan las conexiones necesarias, mismas que están regidas por los parámetros descritos en la programación, y finalmente en la implementación física es la puesta en práctica los procedimientos anteriores, donde se aplica pasos similares a los empleados en la simulación.

RESULTADOS

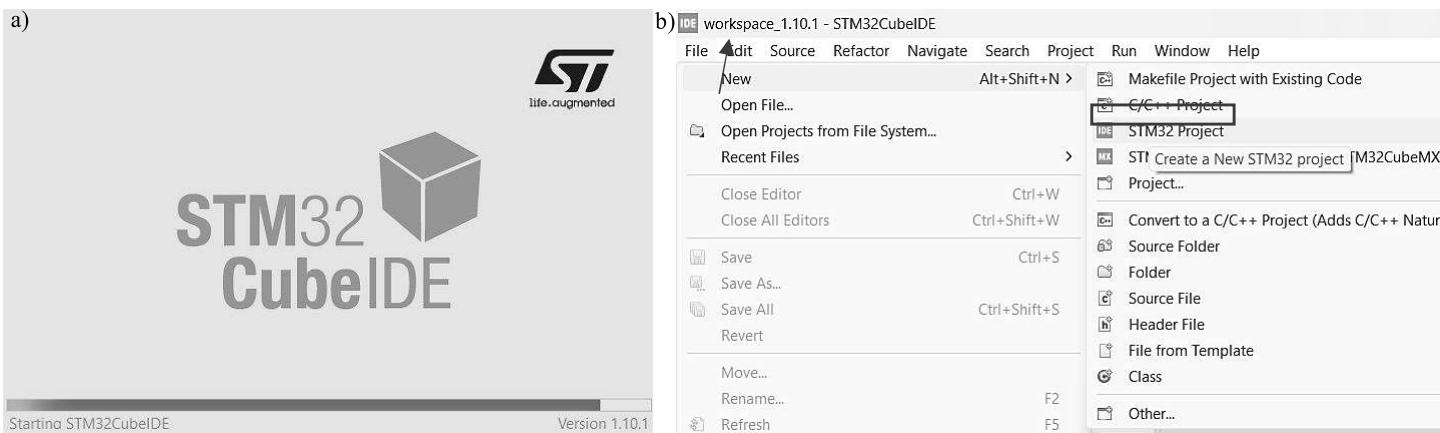
Para el desarrollo de esta sección, las fases de experimentación se dividen, a modo de poder seguir paso a paso los procedimientos para la simulación e implementación física del manejo de una LCD usando un microcontrolador STM32F103C8T6®, de esta forma se tiene la programación en el IDE, seguido de la simulación en un software de simulación de circuitos eléctricos, para finalmente tener la implementación física, así como la sección de conclusiones generadas.

*Es importante señalar que dentro de esta sección se encuentran los resultados obtenidos en cada una de las fases de desarrollo

I. Programación del controlador (*Creación del archivo*).

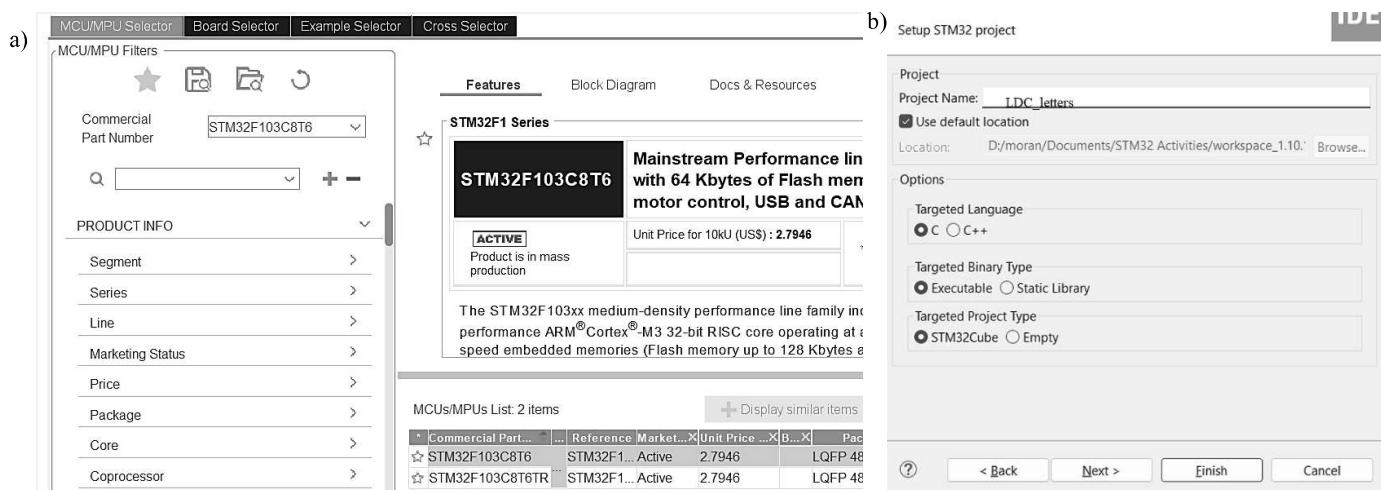
El proceso de programación de un controlador se puede realizar de diferentes métodos; para esta sección se usa la programación en el IDE STM32Cube 1.10.1® mediante el manejo de los registros del microcontrolador, para ello es necesario ajustarse a los parámetros indicados en la hoja de datos correspondiente al microcontrolador y además a la hoja de datos de la LCD.

Inicialmente al abrir la IDE es posible observar una interfaz de bienvenida como la mostrada en la figura 7-a; al cargar la interfaz se requiere la creación de un nuevo proyecto, para ello en la sección de archivos se selecciona la opción “STM32 Project” de la sección “nuevo”, ver la figura 7-b.



Figuras 7 a) Interfaz de bienvenida, b) Creación de un nuevo proyecto.

Al cargar los datos correspondientes, será posible observar un menú de opciones que corresponde a la selección del modelo del microcontrolador, en el caso para este documento, en la sección “Número de parte comercial” se introduce “STM32F103C8T6” que corresponde al chip empleado para el desarrollo de este documento, en el siguiente paso se define el nombre del proyecto, como se muestra en las figuras 8 – a y 8 – b.

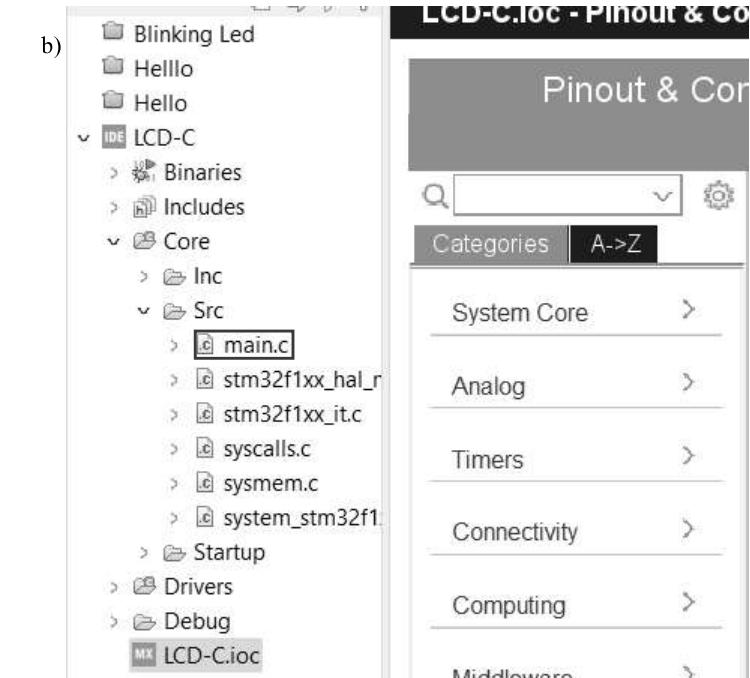
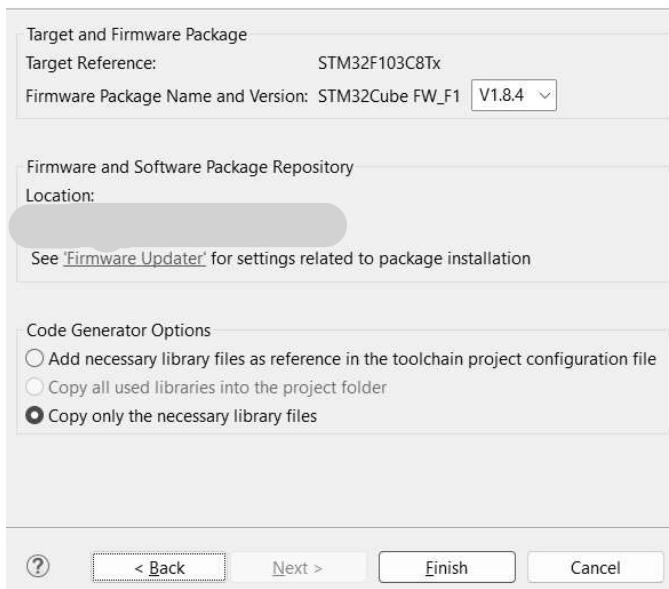


Figuras 8 a) Menú de opciones de microcontroladores, b) Definición del nombre del proyecto.

Para el siguiente paso se realizar la configuración relaciona a los paquetes de librerías, aunque por lo general se termina este paso con los parámetros definidos por defecto; Al terminar, el proyecto se crea, y por defecto es posible observar una ventana para la configuración de las terminales y/o periféricos. Para acceder al archivo de programación (“main.c”) es necesario buscar como dirección la ruta, donde “Nombre del proyecto se sustituye por el nombre correspondiente “*Nombre del proyecto/Core/Src/”, para abrir el archivo se puede haciendo doble clic izquierdo o con clic derecho en la opción “Abrir”, como se muestra en la figura 9-b.

a) Firmware Library Package Setup

Setup STM32 target's firmware



Figuras 9 a) Configuración de las librerías, b) Búsqueda del archivo de programación de las instrucciones.

I.I Programación del controlador (programación)

Para la programación, en base al microcontrolador seleccionado se usa como lenguaje de programación el lenguaje C, de esta forma se instruye inicialmente las librerías en la línea 1 del código, con el propósito de tener una función de retardo se crea dicha función donde el retardo consta de la repetición de 65 ciclos para el avance.

Para la comunicación y configuración se crea una función, en este caso llamada “wcom”, inicialmente mediante la configuración de los registros se activan los puertos A0, A1 de la tarjeta, que en base a la hoja de datos de la LCD y a las conexiones esperadas, corresponden a las terminales E y RS de la LCD. Donde el 0x3 permite la habilitación de registros, y la lectura y escritura. Por otro lado, para 0x2, se desactiva el registro de datos, pero se conserva la habilitación.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														E	RS
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
														0	0
														1	1
														3	

Figura 10 Configuración en los registros para la función "wcom".

Para la función que permite añadir caracteres (wdat), se realiza un ajuste similar al anterior, con la variante de tener desactivado el “Enable” deshabilitando la lectura y escritura, pero continua con RS activo lo que significa tener activo el registro de datos. Cabe señalar que para cada transición se agrega un retraso con la función diseñada, lo anterior debido a que la LCD trabaja a una velocidad inferior a la del corrimiento de las instrucciones, además de repetir la carga al menos dos veces con el propósito de asegurar la carga de datos.

Dentro de la función de inicialización de la pantalla se definen los parámetros de los periféricos, así como la configuración de estos mismos, y la secuencia. Considerando la activación con el oscilador (RCC) al reloj periférico APB2 (APB2ENR) a los puertos de I/O A y B, de donde la representación en el sistema numérico hexadecimal del número formado por los grupos de 4 bits es 0xC. Con la finalidad de mantener activo el reloj se crea una máscara con la acción lógica AND; Así, en base a los requerimientos proporcionados que indican que



las salidas se encuentren en los puertos A y B, en las terminales del 0 al 7, se usa “GPIOA” de control en bajo, mientras para las terminales 8 a 15, se usa “GPIOB” de control en alto.

Para la configuración de los puertos B, se habilitan los puertos 8 y 9, mandando un estado alto para el puerto 8. En la configuración de los puertos 8 y 9 se escoge 0x8 por qué permite configurar como entrada pull-up o pull-down.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved																
Res.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART 1EN	Res.	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	Reserved		IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN	
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	
Bit 2 IOPAEN : I/O port A clock enable Set and cleared by software. 0: I/O port A clock disabled 1:I/O port A clock enabled	Bit 3 IOPBEN : I/O port B clock enable Set and cleared by software. 0: I/O port B clock disabled 1:I/O port B clock enabled								0	0	0	1	1	0	0	0xC

Figura 11 Activación de los timers.

Mientras para las demás terminales en desuso se colocan en 0x4 a fin de colocarlas como entradas flotantes, ambas con el modo de entrada (00).

0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
CNF15[1:0]	MODE15[1:0]	CNF14[1:0]	MODE14[1:0]	CNF13[1:0]	MODE13[1:0]	CNF12[1:0]	MODE12[1:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CNF11[1:0]	MODE11[1:0]	CNF10[1:0]	MODE10[1:0]	CNF9[1:0]	MODE9[1:0]	CNF8[1:0]	MODE8[1:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
0	1	4	0	0	0	1	4	0	0	0	1	4	0	0	0	8

Figura 12 Configuración de reloj para los puertos I/O A.

Siguiendo con la programación se realiza la comunicación con la pantalla, donde mediante la instrucción 0xC se enciende la LCD, de donde se espera que el cursor se mueva a la derecha (I/D = 1).

Instruction	Instruction code										Description	Execution time (fosc= 270 KHZ)
	RS	R/M	DB ₅	DB ₄	DB ₃	DB ₂	DB ₁	DB ₀	-	-		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM and set DDRAM address to "00H" from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" From AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction And blinking of entire display	39us
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and Blinking of cursor (B) on/off Control bit.	
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display Shift control bit, and the Direction, without changing of DDRAM data.	39us
Function set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8-Bit/4-bit), numbers of display Line (N: =2-line/1-line) and, Display font type (F: 5x11/5x8)	39us
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address Counter.	39us
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.	39us
Read busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal Operation or not can be known By reading BF. The contents of Address counter can also be read.	0us
Write data to Address	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43us
Read data From RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43us

Figura 13 Tabla de instrucciones.

Con el fin de no tener información y conforme a la tabla de la figura 13, se limpia la pantalla con la instrucción de comunicación 0x1, para después entrar en el modo de entrada a la configuración, la cual permite el ingreso del

dato por medio de la función “wdat”. Con la función “wdat” es posible mandar un carácter a la pantalla con la sintaxis “wdat(‘carácter’)”.

Continuando con la estructura tradicional, en la función principal se llama por primer paso a la función de inicialización llamada “init_lcd” para después de iniciar entrar al ciclo repetitivo que se crea empleando un “while” de valor 1.

Considerando el objetivo se tiene que al no haber botones presionados se emite un mensaje que dice “Equipo 4”, mientras que al presionar el botón de configuración pull-down muestra el mensaje “Bryan Axel Mejía López”, por otro lado, para al presionar el botón de configuración pull-up muestra el mensaje “Jesus Alberto Moran Moran”.

Por lo que para el desarrollo se crean condiciones relacionadas al estado de los botones, donde al entrar en la condición, inicialmente se borra todos los caracteres en la pantalla, para después crear un retardo que sirva para que la LCD responda, seguido se entra al ciclo que indica que mientras se tenga presionado el botón el mensaje correspondiente seguirá apareciendo. Finalmente, cuando se deja de presionar se borra el mensaje y se crea el retardo, para después mostrar el mensaje inicial.

Por último, para el acomodo del código, se puede realizar con los comandos “ctrl + A” seguido del comando “ctrl + I”; para la depuración del código es necesario hacer clic en la opción señalada en la figura 14.



Figura 14 Depuración de código.

Al depurar el código es posible obtener los valores de memoria usados, en el caso para las instrucciones contenidas en el código desarrollado se tienen los valores mostrados en la figura 15.

LCD-C.elf - /LCD-C/Debug - Mar 2, 2023						
Memory Regions		Memory Details				
Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20004fff	20 KB	18.46 KB	1.54 KB	7.70%
FLASH	0x08000000	0x0800ffff	64 KB	62.22 KB	1.78 KB	2.78%

Figura 15 Valores de la memoria RAM y Flash usada.

II. Simulación de las instrucciones.

El proceso de armado para la simulación consta de la búsqueda de los componentes de forma virtual en el menú, para ello se toma como referencia la información de la tabla 2 del material adicional. Cabe agregar que en el caso del software utilizado fue necesario la definición de la fuente, mediante la ruta “Design/Configure power rails”, para ello se colocan en GND las etiquetas VSS, VSSA, mientras que en VCC/VDD se colocan las etiquetas VDD, VCC y VDDA.

Para cargar el archivo generado por el IDE, es necesario asegurarse que se haya producido el archivo con terminación .hex, en caso de no haberse producido se sigue la ruta en el IDE “C/C++ build/Settings/ MCU Post build outputs” y se selecciona la opción “Convert to Intel Hex file (-O ihex)”, para ello se accede con clic derecho en el proyecto en la sección de propiedades.

Con el archivo de las instrucciones hecho, se carga dicho archivo con doble clic sobre el microcontrolador, e inmediatamente se abre un menú, de donde en la carpeta se busca la dirección en donde se localiza el archivo y se carga.

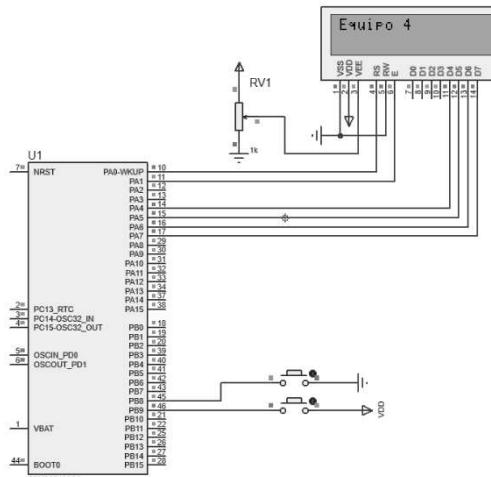
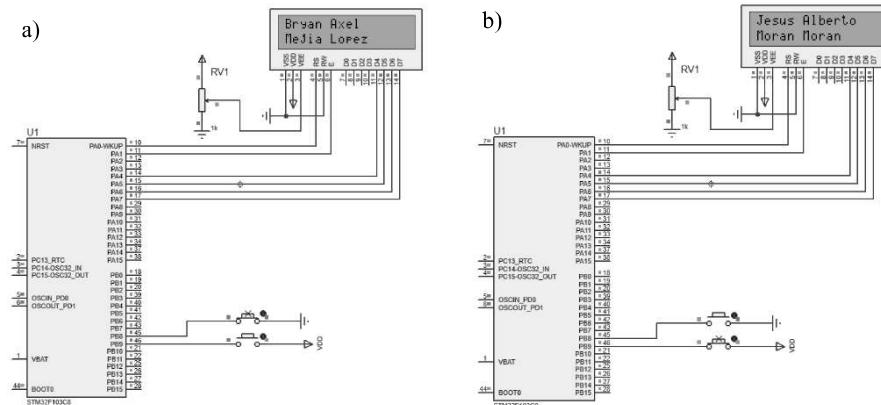


Figura 16 Diagrama para la simulación.

De esta forma, se inicializa la simulación y se observa el funcionamiento. En las figuras 17-a y 17-b se observa evidencia del funcionamiento correcto de la secuencia, donde también en la figura 16 se ve el mensaje inicial.



Figuras 17 a) Funcionamiento con pull-down 1, b) funcionamiento con pull-up.

De la simulación se puede observar las conexiones realizadas son similares a las que se realizarán de forma física. En el siguiente video se muestra el funcionamiento de la simulación

https://drive.google.com/file/d/1-2Zz6QDOhkqZ17ZrRgIxXhI8c4tX17sz/view?usp=share_link

RESULTADOS FÍSICOS

Para la implementación física, es necesario la instalación de los elementos en una placa de prototipos, para ello se usa como referencia la información de la figura 18, que es la representación más cercana a las conexiones físicas, de donde es importante señalar que para cargar la información se requerirá del puerto USB de la computadora, así como el programador ST-Link V2®.

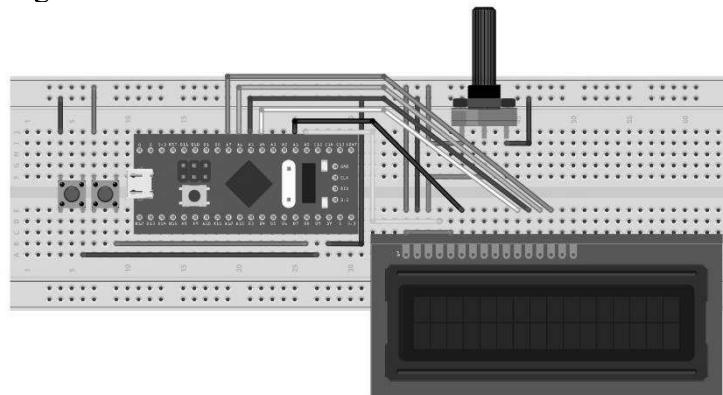


Figura 18 Conexiones realizadas en una placa de prototipos.

Para la conexión del programador, se sigue el diagrama contenido en la carcasa de este mismo, realizando conexiones con los puertos 3.3 V, SWDIO, SWCLK y GND con los puertos correspondientes a la tarjeta, ver figura 3.

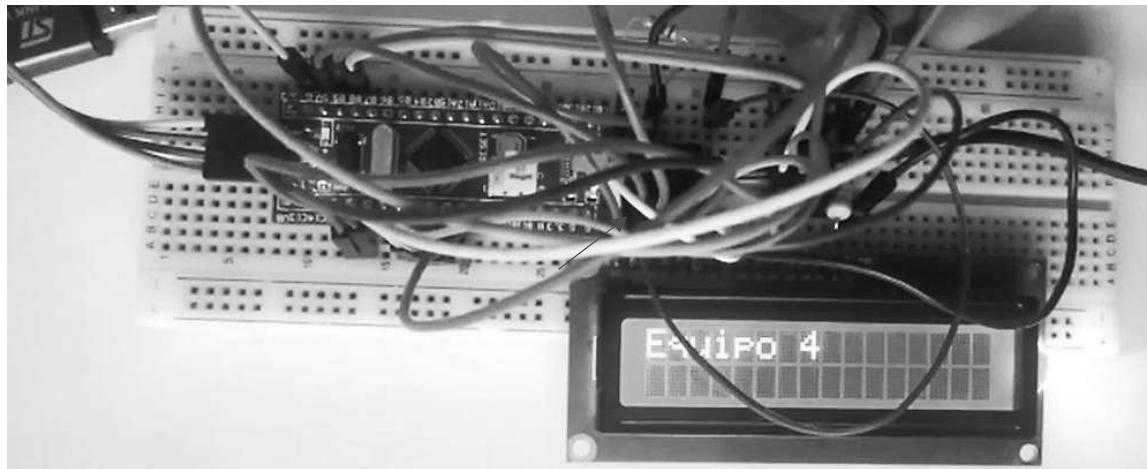


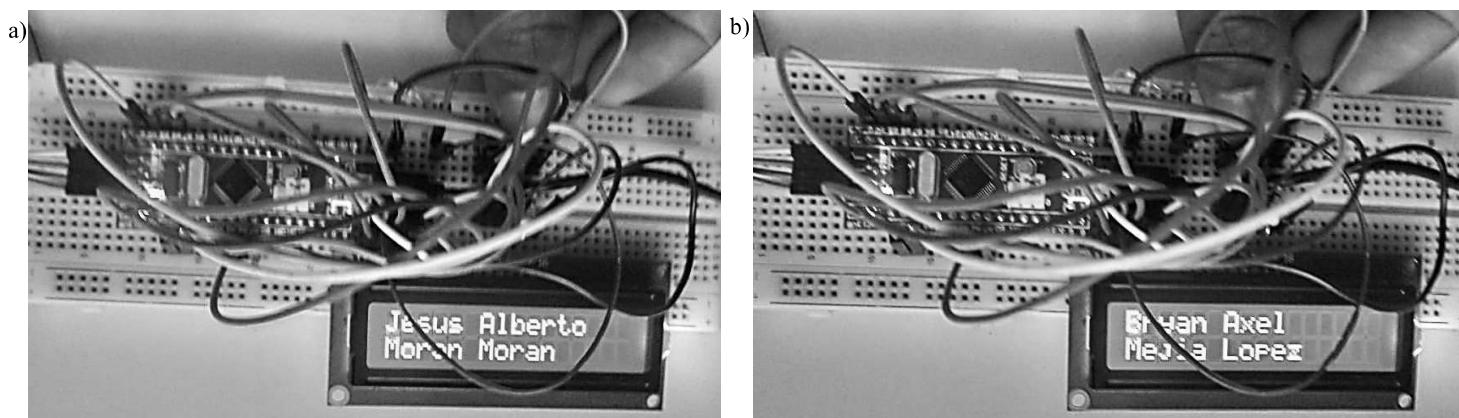
Figura 19 Conexiones físicas realizadas, con el mensaje inicial.

La operación para cargar las instrucciones se realiza desde el IDE en la opción de depurar, como se muestra en la figura 20, en esta parte es necesario tener el programador conectado junto con las conexiones anteriormente mencionadas entre el programador y la placa de desarrollo.



Figura 20 Depuración y carga de las instrucciones.

Al depurar, aparece un cuadro de dialogo el cual para continuar se debe aceptar, de esta forma al continuar, un led dentro del programador cambia de color indicando que las instrucciones se están cargando. Cabe agregar que se provee de energía (5 V de los mismos puertos disponibles de la tarjeta), para el correcto funcionamiento, es necesario la conexión mediante el puerto micro USB, ¡Es importante desconectar el programador a fin de evitar daños a este dispositivo!



Figuras 21 a) Configuración pull-up, b) Configuración pull-down.

Es importante considerar que el led de retroiluminación puede llegar a requerir una resistencia, para el caso de la pantalla de cristal líquido de las figuras 20, 21-a y 21-b no es necesario dicha resistencia ya que se encuentra internamente en la placa de la LCD



Materiales

Los materiales seleccionados y listados dentro de la tabla 1 fueron elegidos bajo el alcance de cumplir de forma capaz el objetivo planteado en un inicio, señalando que para el desarrollo completo de las actividades el conjunto de elementos son los mínimos y los sugeridos a emplear, pudiendo agregar más elementos conforme a el método de experimentación seguido.

TABLA 1. MATERIALES PARA SIMULACIÓN

Material	Descripción
Computadora	Este recurso servirá como soporte para el desarrollo de la simulación, así mismo poder visualizar información relacionada.
Entorno de desarrollo integrado (IDE)	Se usará para la programación y comunicación con la tarjeta de desarrollo.
Software de simulación	Este recurso dispondrá de la simulación de los resultados esperados realizando la comunicación entre el IDE y este mismo.
Software de diseño de ensamble	Proveerá del soporte necesario para las conexiones físicas, haciendo una simulación cercana a los elementos reales.

TABLA 2. MATERIALES PARA LA ELABORACIÓN FÍSICA

Material	Descripción
Tableta de prototipos	Servirá como soporte para el circuito eléctrico.
16 cables de conexión Macho-Macho	Serán usados para la conexión entre los componentes y sus terminales.
1 pantalla de cristal líquido	Permitirá la visualización de caracteres a modo de texto.
Cables micro USB	Permitirán el suministro de energía a la placa de desarrollo.
Convertidor USB a TTL (programador)	Permitirá la comunicación entre el puerto USB de la computadora y la tarjeta de desarrollo.
Placa de desarrollo (STM32 F103 C8T6)	Proveerá las salidas, así como la ejecución de la secuencia programada.
1 potenciómetro	Permitirá la modulación del contraste.

*Los elementos elegidos pueden ser modificados siempre y cuando las características de los elementos a remplazar sean similares o el funcionamiento sea el esperado según el objetivo.

Código de las instrucciones

El código implementado en el IDE, contienen las instrucciones que se ejecutarán por ello a continuación anexan dichas líneas de código:

```
#include "main.h" //Initialize the header file

//Create a function called "delay"
void delay(void) {
    int i; //Create a variable
    for(i=0;i<65;i++); //Use 65 cycles as a delay base on the main clock.
}

//Configure the communication between LCD and the board.
void wcom(char dato) {
    GPIOA->ODR &=~0x3;
    GPIOA->ODR =(dato&0xF0) | (GPIOA->ODR&0x0F);
    GPIOA->ODR |=0x2;
    delay();
    GPIOA->ODR &=~0x2;
    GPIOA->ODR =(dato<<4&0xF0) | (GPIOA->ODR&0x0F);
    GPIOA->ODR |=0x2;
    delay();
    GPIOA->ODR &=~0x2;
}

//Transfer the information at LCD.
```



```
void wdat(char dato){  
    GPIOA->ODR &= ~0x2;  
    GPIOA->ODR |= 0X1; //RS=1  
    GPIOA->ODR = (dato&0xF0) | (GPIOA->ODR&0x0F);  
    GPIOA->ODR |= 0x2;  
    delay();  
    GPIOA->ODR &= ~0x2;  
    GPIOA->ODR = (dato<<4&0xF0) | (GPIOA->ODR&0x0F);  
    GPIOA->ODR |= 0x2;  
    delay();  
    GPIOA->ODR &= ~0x2;  
}  
  
void init_lcd(void){  
    int i;  
    RCC->APB2ENR |= 0xC; //Configure the ports A & B.  
    while(!(RCC->APB2ENR&0xC)); //A & B  
    GPIOB->CRH = 0x44444488; //Configure as a input pull up and input reset port - 9.  
    GPIOA->CRL = 0x33334433; //Configure the ports 2 and 3.  
  
    GPIOA->ODR = 0;  
    GPIOB->ODR |= 0x100; //Put high the 8 port.  
    for(i=0;i<64000;i++); //Create a delay of 64000 clock cycles.  
    wcom(0x32);  
    for(i=0;i<16000;i++); //Create a delay of 16000 clock cycles.  
    wcom(0x28);  
    for(i=0;i<32000;i++); //Create a delay of 32000 clock cycles.  
    wcom(0x28);  
    for(i=0;i<16000;i++); //Create a delay of 16000 clock cycles.  
    wcom(0x0C); //ON  
    for(i=0;i<16000;i++); //Create a delay of 16000 clock cycles.  
    wcom(0x01); // CLEAR  
    for(i=0;i<16000;i++); //Create a delay of 16000 clock cycles.  
    wcom(0x06); //ENTRY MODE SET  
    wcom(0x80); //+0x4  
    wdat('E');  
    wdat('q');  
    wdat('u');  
    wdat('i');  
    wdat('p');  
    wdat('o');  
    wcom(0x87);  
    wdat('4');  
}  
  
long i;  
int main(void){  
    init_lcd(); //Call the function called init_lcd.  
    while(1){ //Infinite loop  
        if(((GPIOB->IDR)&0x100)!=0x100){  
            wcom(0x01); // CLEAR  
            for(i=0;i<1600;i++); //Create a delay of 16000 clock cycles.  
            while(((GPIOB->IDR)&0x100)!=0x100){  
                wcom(0x80); //+0x4  
                wdat(66); //B  
                wdat(114); //r  
                wdat(121); //y  
                wdat(97); //a  
                wdat(110); //n  
                wcom(0x86);  
                wdat(65); //A  
                wdat(120); //x
```