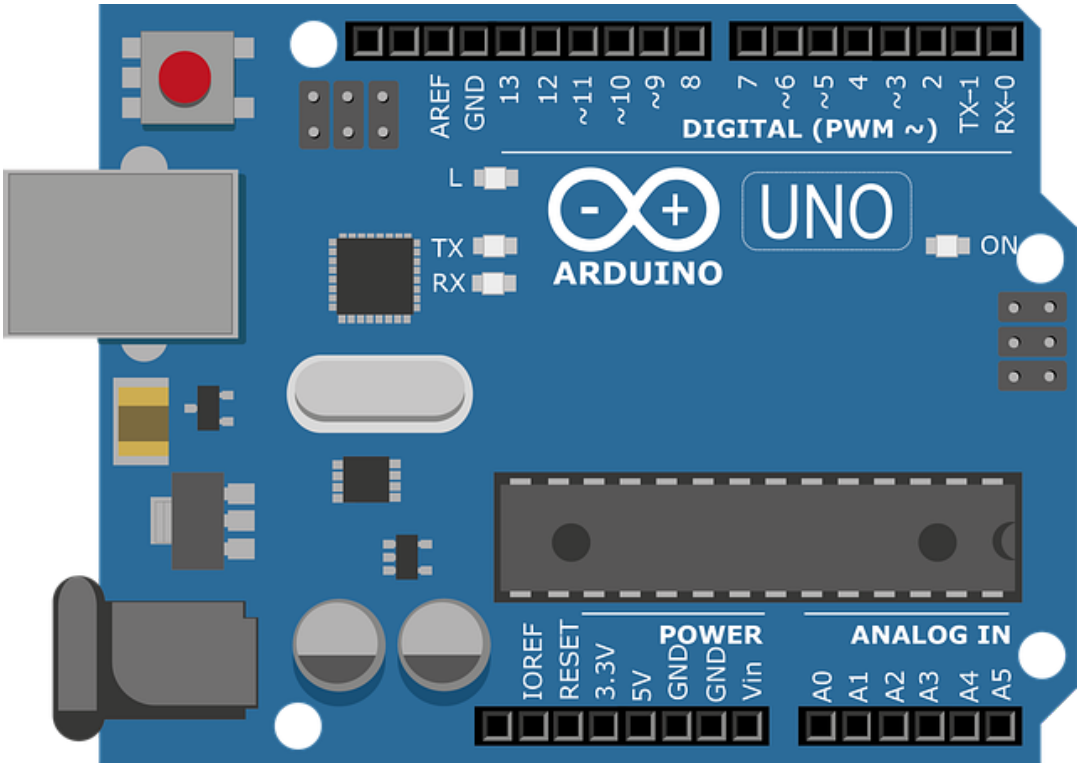
PRÁCTICAS CON ARDUINO



Práctica 12: Pantalla 16x2 LCD con I2C

Grupo de Trabajo

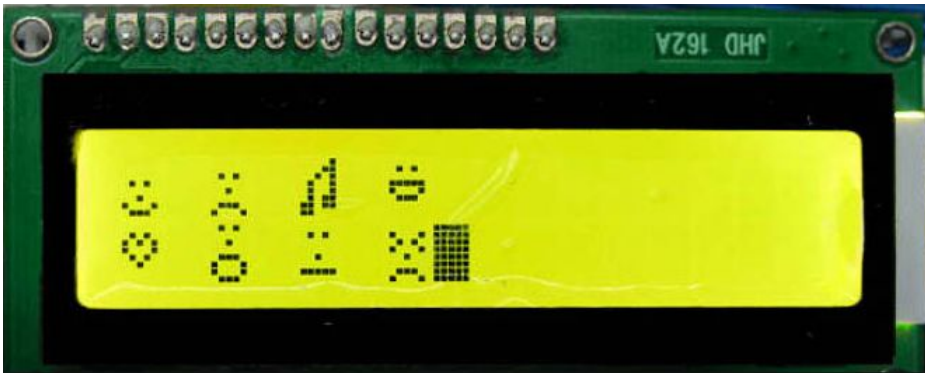
Mónica Cañón Pascual
IES Pino Rueda
Departamento de Tecnología

Abril 2018

Finalidad de la práctica

1. Conocer, montar, y programar la pantalla LCD 16x2 y el módulo I2C.

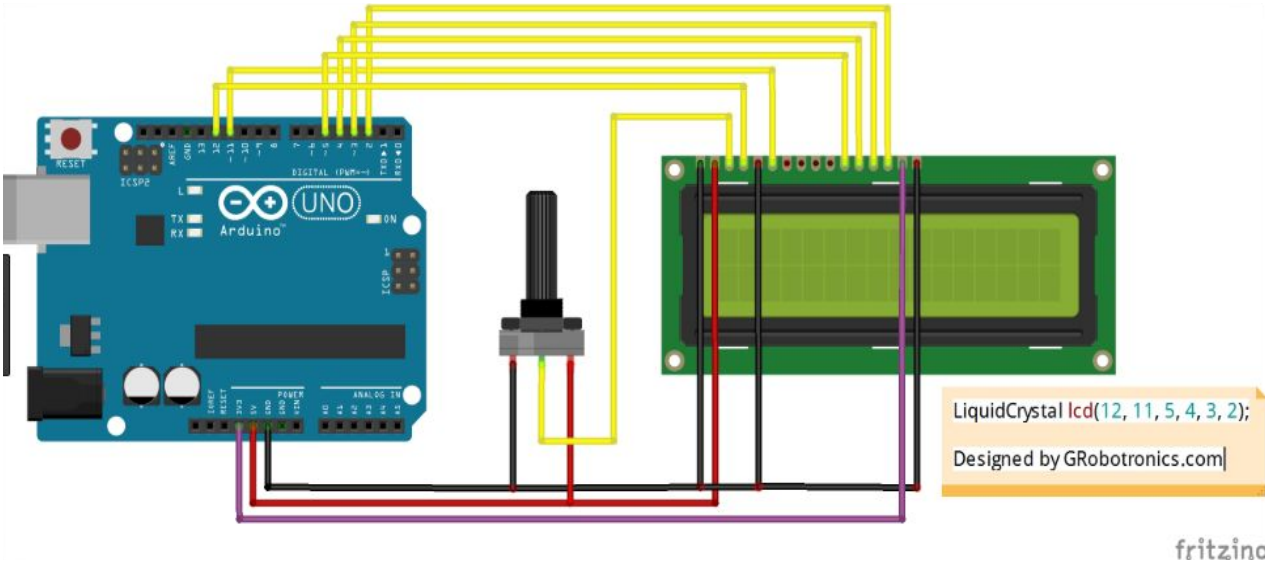
2. Probar el envío de mensajes a la pantalla.
3. Diseño de algunos caracteres sencillos.



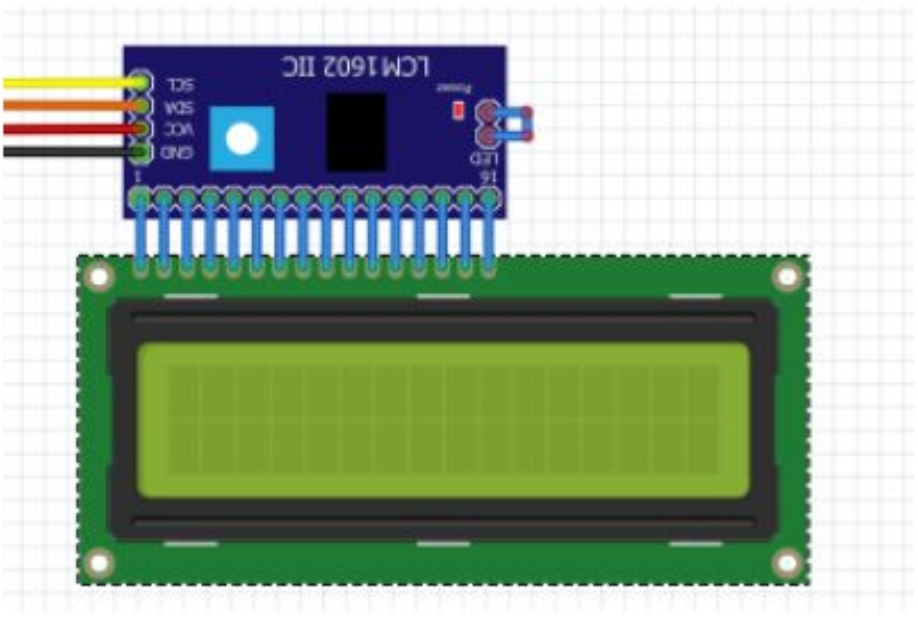
Información

1. La pantalla LCD (Liquid Crystal Display), sirve para visualizar datos, comunicaciones con la placa de Arduino. Normalmente tiene 6 pines para

comunicarse con Arduino, más un potenciómetro para calibrar el contraste. Este dispositivo tiene 2 líneas con 16 caracteres cada una.



2. En esta práctica vamos a programar una pantalla de módulo LCD que se comunican con Arduino mediante una placa anexa, comunicando con dos cables a través del bus I2C 1602. El Módulo adaptador LCD a I2C que usaremos está basado en el controlador I2C PCF8574, el cual es un

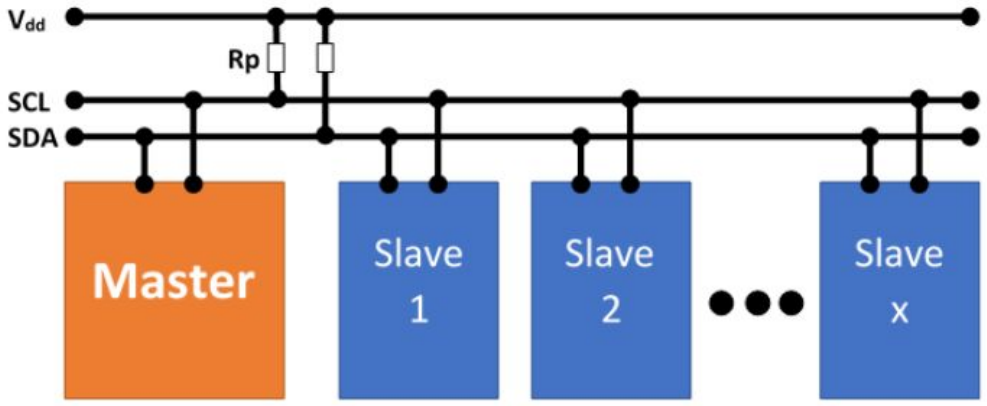


Expansor de Entradas y Salidas digitales controlado por I2C. Este último requiere sólo necesita dos conexiones para datos más dos para

alimentación a 5vdc y a tierra. El módulo tiene un potenciómetro para ajustar el contraste.

3. I2C es un bus de comunicación a través de dos cables. La comunicación puede ser con multitud de dispositivo. La comunicación es serie, es decir, los datos se hacen uno a uno a una velocidad de hasta 3,4 Mbts/s y hace uso de una estructura Maestro-esclavo (hasta 128 dispositivos conectados).

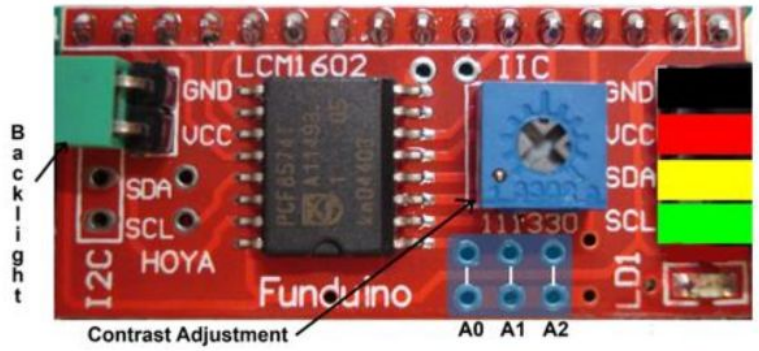
   Cuando la pantalla LCD no está integrado con la placa I2C. Para que un sistema basado en I2C funcione correctamente ha de llevar asociadas un par de resistencias, con un valor típico de 4K7 tal como se muestra en la imagen:



   En el caso que nos ocupa, la interfaz I2C basada en el PCF8574 las lleva ya asociadas.

4. El diagrama de pins del I2C es: (En las placas Arduino el SDA es el A4 y el SCL a A5)

Pinout Diagram:



Pin/Control Descriptions:

Pin #	Name	Type	Description
1	GND	Power	Supply & Logic ground
2	VCC	Power	Digital I/O 0 or RX (serial receive)
3	SDA	I/O	Serial Data line
4	SCL	CLK	Serial Clock line
A0	A0	Jumper	Optional address selection A0 - see below
A1	A1	Jumper	Optional address selection A1 - see below
A2	A2	Jumper	Optional address selection A2 - see below
Backlight		Jumper	Jumpered - enable backlight, Open - disable backlight
Contrast		Pot	Adjust for best viewing

Addressing:

A0	A1	A2	Address
Open	Open	Open	0x27
Jumper	Open	Open	0x26
Open	Jumper	Open	0x25
Jumper	Jumper	Open	0x24
Open	Open	Jumper	0x23
Jumper	Open	Jumper	0x22
Open	Jumper	Jumper	0x21
Jumper	Jumper	Jumper	0x20

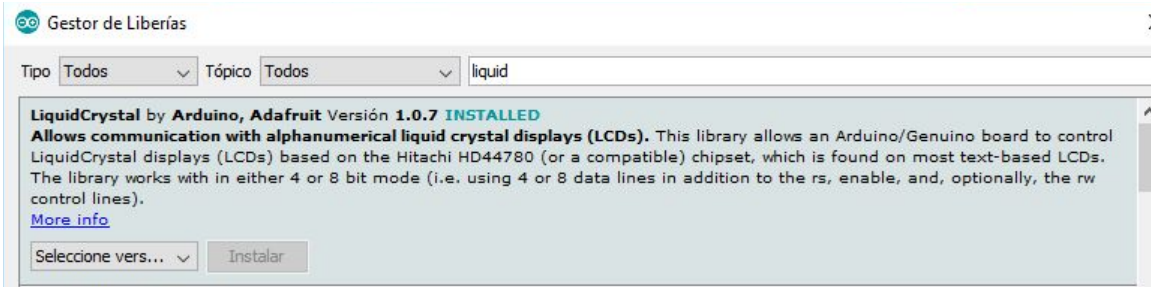
4. Arduino gestiona este tipo de conexiones a través de la librería Wire. Hay librerías de más alto nivel que la incluyen.
5. Para configurar la pantalla LCD con Arduino, hay que realizar dos pasos:
    a. Si conocemos la dirección I2C de la pantalla, la utilizaremos. Sino utilizaremos un software que se llama i2cscaner que nos permitirá obtener este valor.

       La dirección I2C por defecto del módulo puede ser **0x3F** o en otros casos **0x27**. Es muy importante identificar correctamente la dirección I2C de nuestro módulo, pues de otra forma nuestro programa no funcionará correctamente. Si en caso existiera la necesidad de trabajar con más de un LCD podemos modificar la dirección I2C del módulo adaptador. Para esto es necesario soldar los puentes A0, A1 y A2 presentes en el módulo, estos tres puentes son los bits menos significativos de la dirección I2C del módulo. La dirección 0x3F en binario sería: 0|0|1|1|1|A2|A1|A0 y la dirección 0x27: 0|0|1|0|0|A2|A1|A0. Por defecto A0, A2, A1 vale 1 pero si soldamos los puentes, estos se conectan a tierra teniendo un valor 0. Por ejemplo si soldamos los tres puentes la nueva dirección sería 0|0|1|0|0|0|0|0 (0x20), para un chip que anteriormente era 0x27.

    b. Ajustar el nivel de contraste. Para ello utilizaremos el potenciómetro. La luz de fondo se controla principalmente por

software desde el Arduino, pero el módulo también permite desconectar el Led de la luz de fondo removiendo un jumper LED.

6. En nuestro programa de Arduino siempre debemos incluir la libreria “LiquidCrystal_i2C”. Esto se hace en “Gestionar librerías” o incluir un archivo zip que la incluya. Estos tienen incluidos ejemplos, como el “Hello world!”.



https://www.arduinolibraries.info/libraries/liquid-crystal-i2-c

https://codeload.github.com/marcoschwartz/LiquidCrystal_I2C/zip/master

Sobre la programación

La pantallas está compuesta de 16 caracteres en dos filas. Estos puntos se representan mediante coordenadas: (Y,X), es decir, columnas y filas. Hay 2 filas.

16 Columnas

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
1																

Así la instrucción siguiente quedaría:

```
lcd.setCursor(3,0);
lcd.print("Hello, world!");
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				H	E	L	L	O		W	O	R	L	D	!	
1																

La librería **LiquidCrystal_I2C** dispone de métodos similares (algunos idénticos) a los de la librería oficial.

- **LiquidCrystal_I2C() –** Constructor de la clase, configura el hardware.
- **init() –** Prepara el LCD para su uso.
- **clear() –** Borra todos los caracteres de la pantalla LCD.
- **setCursor(col, row) –** Permite mover el cursor a la posición indicada en sus parámetros.
- **print() –** Imprime una variable o literal en la pantalla
- **scrollDisplayLeft() y scrollDisplayRight() –** Recorre el contenido de la pantalla a la izquierda o a la derecha
- **backlight() y noBacklight() –** Métodos para encender / apagar la iluminación de fondo
- **createChar(num, data) –** Crear un carácter definido por el usuario en la memoria del controlador de pantalla

Las pantallas LCD 16×2 permiten definir hasta 8 caracteres personalizados, la librería LiquidCrystal_I2C provee las facilidades para realizar esta tarea de manera muy sencilla. Cada carácter se define como un grupo de 8 bytes que se envían a la memoria CGRAM. Por ejemplo, podemos crear un icono con una carita feliz de la siguiente forma:

Representación en binario de un carácter personalizado:

	1	2	3	4	5
1					
2		■		■	
3		■		■	
4		■		■	
5					
6	■				■
7		■	■	■	
8					

1° parte					2° parte					Código C
0	b	0	0	0	0	0	0	0	0	byte **smile[8]** = {
0	b	0	0	0	0	**1**	0	**1**	0	0b00000000,
0	b	0	0	0	0	**1**	0	**1**	0	0b00001010,
0	b	0	0	0	0	**1**	0	**1**	0	0b00001010,
0	b	0	0	0	0	**1**	0	**1**	0	0b00001010,
0	b	0	0	0	0	0	0	0	0	0b00000000,
0	b	0	0	0	**1**	0	0	0	**1**	0b00010001,
0	b	0	0	0	0	**1**	**1**	**1**	0	0b00001110,
0	b	0	0	0	0	0	0	0	0	0b00000000,
										};

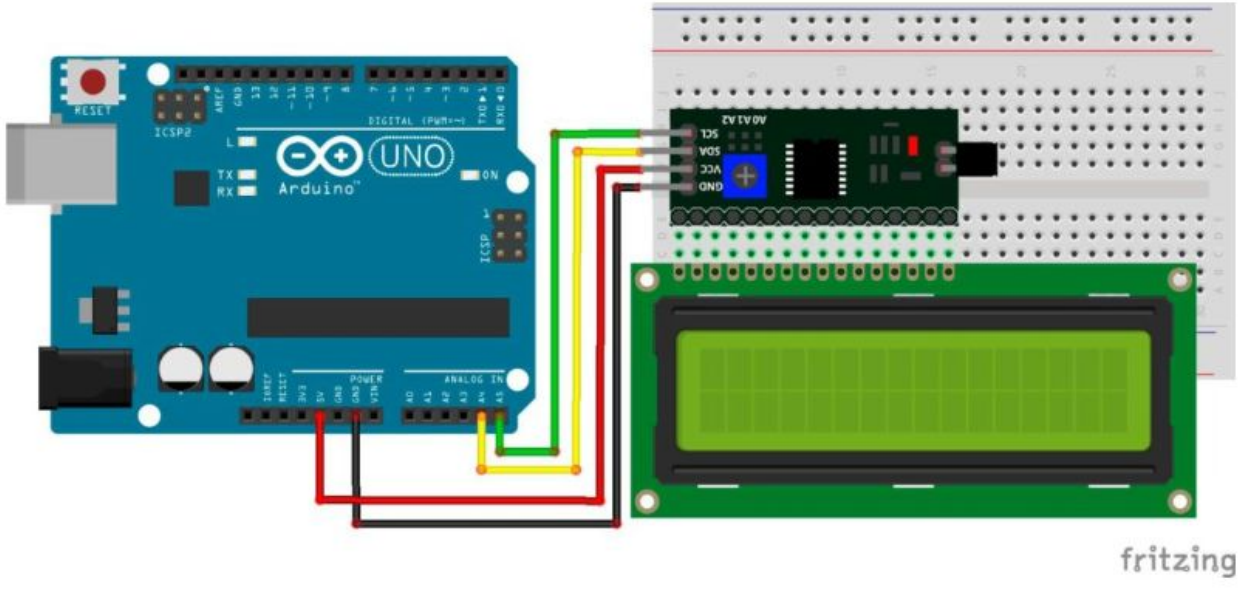
Así se pueden crear otras formas con el siguiente diseño básico:

	1	2	3	4	5
1					
2					
3					
4					
5					
6					
7					
8					

	1	2	3	4	5
1					
2					
3					
4					
5					
6					
7					
8					

Hardware necesario

I. Esquema de conexiones



Programación

Snap4Arduino	IDE arduino

???	//YWROBOT
	//Compatible with the Arduino IDE 1.0
	//Library version:1.1
	
	#include <Wire.h>
	#include <LiquidCrystal_I2C.h>
	
	LiquidCrystal_I2C lcd(0x27,16,2);
	// set the LCD address to 0x3F para display de 16 caracteres y 2 líneas
	
	void setup() {
	lcd.init(); // Iniciar el lcd
	lcd.init(); // Print a message to the LCD.
	lcd.backlight();
	lcd.setCursor(3,0);
	lcd.print("Hello, world!");
	lcd.setCursor(2,1);
	lcd.print("Ywrobot Arduino!");
	}
	void loop() {
	}

Actividades y propuestas de mejora

- Diseñar distintos caracteres, como caritas (feliz, triste, serio...)
- Utilizar las distintas funciones de la libreria: scrollright, scrollleft,...
-

Otros anexos

Código para scanner al I2C.

```
// i2c_scanner Version 1
//  This program (or code that looks like it)
//  last Version 6, November 27, 2015.
//  Added waiting for the Leonardo serial communication
// This sketch tests the standard max 7-bit addresse
#include <Wire.h>
```

```
void setup() {
  Wire.begin();
  Serial.begin(9600);
  while (!Serial);            //  wait for serial monitor
  Serial.println("\nI2C Scanner"); }

void loop() {
  byte error, address;
  int nDevices;
  Serial.println("Scanning...");

  nDevices = 0;
     for(address = 1; address < 127; address++ )   {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)     {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;     }
    else if (error==4)     {
      Serial.print("Unknown error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);     }    }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);          // wait 5 seconds for next scan
}
```