MÓDULO 4: PROJECTO 11.01 "El bus I2C"

CURSO PROGRAMACIÓN DE PLACAS ROBÓTICAS

Objetivos

- Comprender el bus I2C.
- Conectar un display I2C.
- Enviando mensajes a la pantalla

El bus I2C

- A medida que la capacidad integración en un único chip aumentaba, el número de componentes comerciales disponibles, aumentaba exponencialmente. Cada vez era, y es, más sencillo fabricar bloques de construcción electrónicos integrados en un único chip, y pronto el grosor de los catálogos de los fabricantes, engordó peligrosamente.

- Era relativamente fácil encontrar esos bloques de construcción pero cuando tu diseño requería usar una docena de esos bloques, ponerlos de acuerdo y conseguir que se comunicaran eficazmente, se convirtió en un problema.

- Por eso, en los primeros 80, uno de los grandes fabricantes de electrónica (Phillips), propuso una norma de comunicación digital, entre los diferentes componentes de una sistema electrónico.

Una norma que especificaba la velocidad, niveles de tensión, y el protocolo a seguir para conseguir esa comunicación y la hizo abierta.

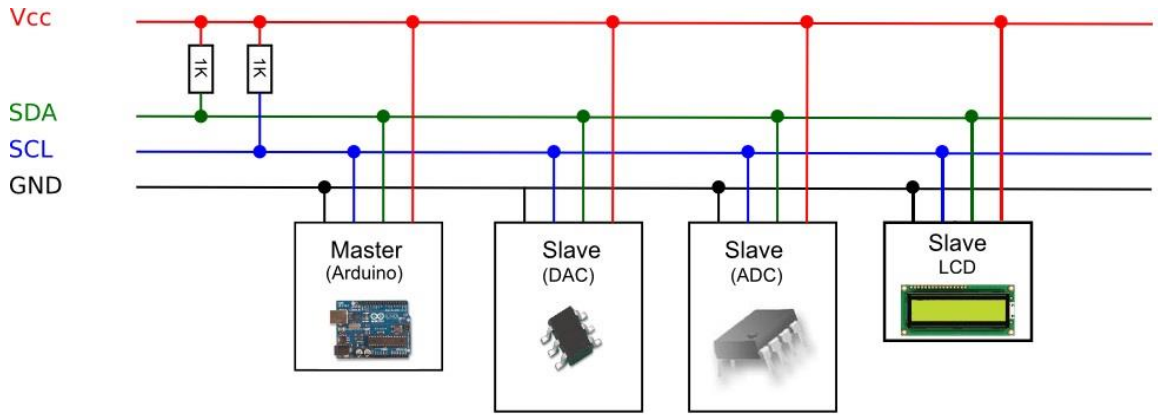
El bus I2C

- Una norma que especificaba la velocidad, niveles de tensión, y el protocolo a seguir para conseguir esa comunicación y la hizo abierta.

- Esa norma se llamó Inter Integrated Circuits bus, o IIC, y pronto se convirtió en un estándar de facto en la industria. Las especificaciones han ido mejorando con los años, pero la idea básica sigue siendo la misma:
  - Protocolo de dos hilos de control, uno para transmitir los datos, SDA y otro, el reloj asíncrono que indica cuando leer los datos SCL. Mas GND y 5V (cuando se requiera).
  - Cada dispositivo conectado al **bus I2C** y cada uno tiene su dirección exclusiva, de 7 bits, (Asi que, en teoría, podemos conectar $2^7 = 128$, dispositivos).

El bus I2C

- Uno de estos componentes, debe actuar como master, es decir controla el reloj.
- No se requiere una velocidad de reloj estricta, ya que es el master quien controla el Clock.
- Es multi master, el master puede cambiar, pero solo uno puede estar activo a la vez, y proporciona un protocolo de arbitraje y detección de colisiones. (Si no has entendido esto, no te preocupes, todavía es pronto).



El bus I2C

- La idea es que todos los componentes se conecten en paralelo a las dos líneas del Bus, SDA y SCL. En cada momento solo puede haber un master, en este caso, nuestro Duino y los demás se configuran como esclavos.
  - Puede haber más de un master. La norma propone un sistema de arbitraje, para transferir el control de uno a otro, pero en un instante dado, solo uno puede ser el master.
  - Fijaros también que hay unas resistencias de Pullup conectadas a SDA y SCL. Son imperativas, ya que el bus es activo bajo (Esto es, la señal activa es un 0, no un 1. Pero tranquilo, que esto no te afecta)

El bus I2C

- Cuando vayas a conectar algo al busI2C, es imprescindible que leas el manual para saber si los pullups los tienes que poner tú, o vienen puestos en el componente.
- En el caso del display I2C que vamos a usar, normalmente incluyen los pullups.

Arduino lo soporta de fábrica con una librería estándar, que utiliza dos de los pines analógicos para las funciones SDA (Datos) y SCL (Clock).

- En el Arduino UNO, los pines I2C están en los pines analógicos A4 (SDA) y A5 (SCL).
- En el Arduino Mega y DUE, son el 20 (SDA) y en el 21(SCL).

El bus I2C

- La librería I2C, en Arduino se llama Wire, y gestiona el protocolo de comunicaciones completo, lo que es un detalle, pues nos ahorra la parte aburrida de estudiar el protocolo y escribir programas para ello.
- Esto no es vagancia, sino construir sobre el trabajo de terceros. Es una de las muy grandes virtudes de la comunidad Arduino. Muchas librerías para incorporar a nuestros proyectos, sin necesidad de mancharte las manos de grasa.
- En esta sesión vamos a conectar un display LCD de 16×2 con interface I2C, y así podréis comprobar porque en la última sesión os recomendé usarlo, en lugar del que se conecta directamente con 16 pines.

Scanner de I2C

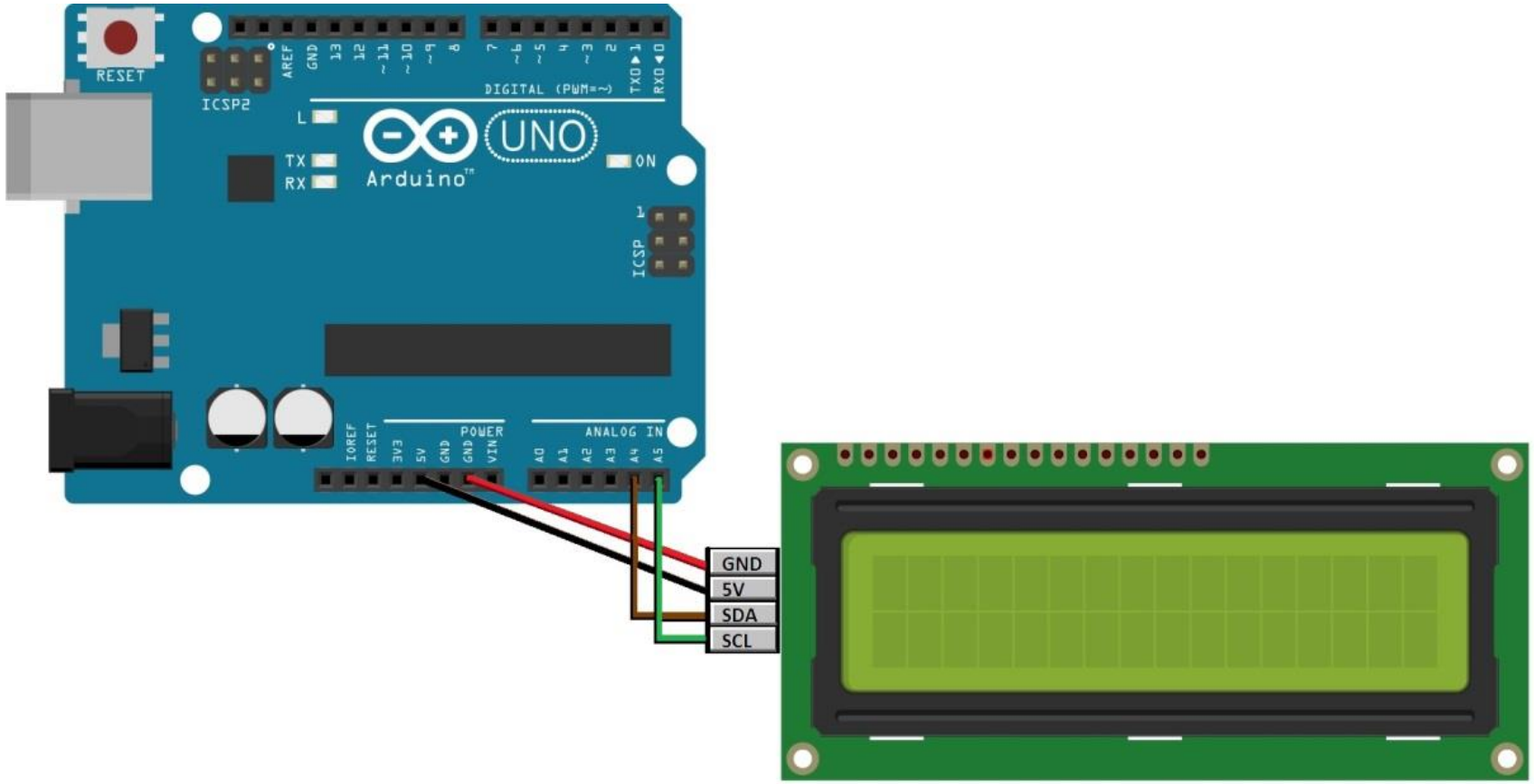
- Cada componente que conectamos al **bus I2C** tiene una dirección única, y cada mensaje y orden que transmitimos al bus, lleva anexa esta dirección, indicando cuál de los muchos posibles, es el receptor del mensaje.
- Pero, claro, esto implica que sabemos la dirección del componente. Lo normal es comprobar la información técnica del fabricante del componente, y ahí suele decirnos cuál es la dirección por defecto.
- Pero como ya sabemos, que en la vida las cosas rara vez son como en los cuentos, algún alma caritativa (y con mucho mundo a sus espaldas), hizo un programita para Arduino, que nos informa, de lo que hay en nuestro bus y con qué dirección. I2C scanner

Scanner de I2C

- Naturalmente, este programa, no tiene ni idea de quien responde y lo que hace, pero bastante es que nos informe de que hay alguien en la dirección xx.

- Si no sabemos en qué dirección está un componente dado, basta con colocarlo solo en el bus, y ver qué dirección nos reporta el I2C scanner. EL resultado para el LCD que tengo es 0x27 Hexadecimal.

Montando el circuito



Comentario del código

- Lo primero, es que descarguéis una nueva librería para maneja el display con I2C, Se llama LiquidCrystal_I2C y es un añadido para la librería estándar que viene con tu Duino
- Descarga la librería LiquidCrystal_I2C , y vamos a instalarla lo primero
  - \\Programa\ImportarLibreria\Añadir \Librería
- Busca en tu directorio de descargas, la librería LiquidCrystal_I2C .zip y haz doble click, para que se instale en nuestro IDE.
- Ahora vamos a incluir la librería I2c que en Arduino se llama Wire, y como es estándar la incluyes con:
  - \\Programa\ImportarLibreria\Wire

Comentario del código

- Ahora hay que incluir la librería LiquidCrystal normal y después la de LiquidCrystal_I2C.Deberiamos tener 3lineas como estas:
  - #include <Wire.h>
  - #include <LCD.h>
  - #include <LiquidCrystal_I2C.h>
- Vamos ahora a definir una variable con la dirección de nuestro display. En mi caso la 0x27
  - byte dir = 0x27 // Ese 0x significa que es hexadecimal, no decimal
- Y por último creamos una instancia del objeto LiquidCrystal_I2C:

LiquidCrystal_I2C lcd(dir, 2, 1, 0, 4, 5, 6, 7);

Comentario del código

- Al que le pasamos la dirección dir del display y varios números que indican que pines debe usar la librería para el display, no de Arduino. Ignóralo y lo copias así para los displays. El resto queda así:

```
#include <Wire.h>

#include <LCD.h>

#include <LiquidCrystal_I2C.h>

#define I2C_ADDR    0x27

LiquidCrystal_I2C          lcd(I2C_ADDR,2, 1, 0, 4, 5, 6, 7);

void setup() {

lcd.begin (16,2);    // Inicializar el display con 16 caraceres 2 lineas
lcd.setBacklightPin(3,POSITIVE); lcd.setBacklight(HIGH); lcd.home ();                    // go home
lcd.print("Prometec.net"); lcd.setCursor (0,1);        // go to the 2nd line lcd.print("Malpartida
```

Comentario del código

```
lcd.setBacklightPin(3,POSITIVE);

lcd.setBacklight(HIGH);

lcd.home ();                    // go home

lcd.print(“Hola Fonde Norte");

lcd.setCursor ( 0, 1 );         // go to the 2nd line

lcd.print(“Hola Fondo Sur");

}

void loop() {}
```