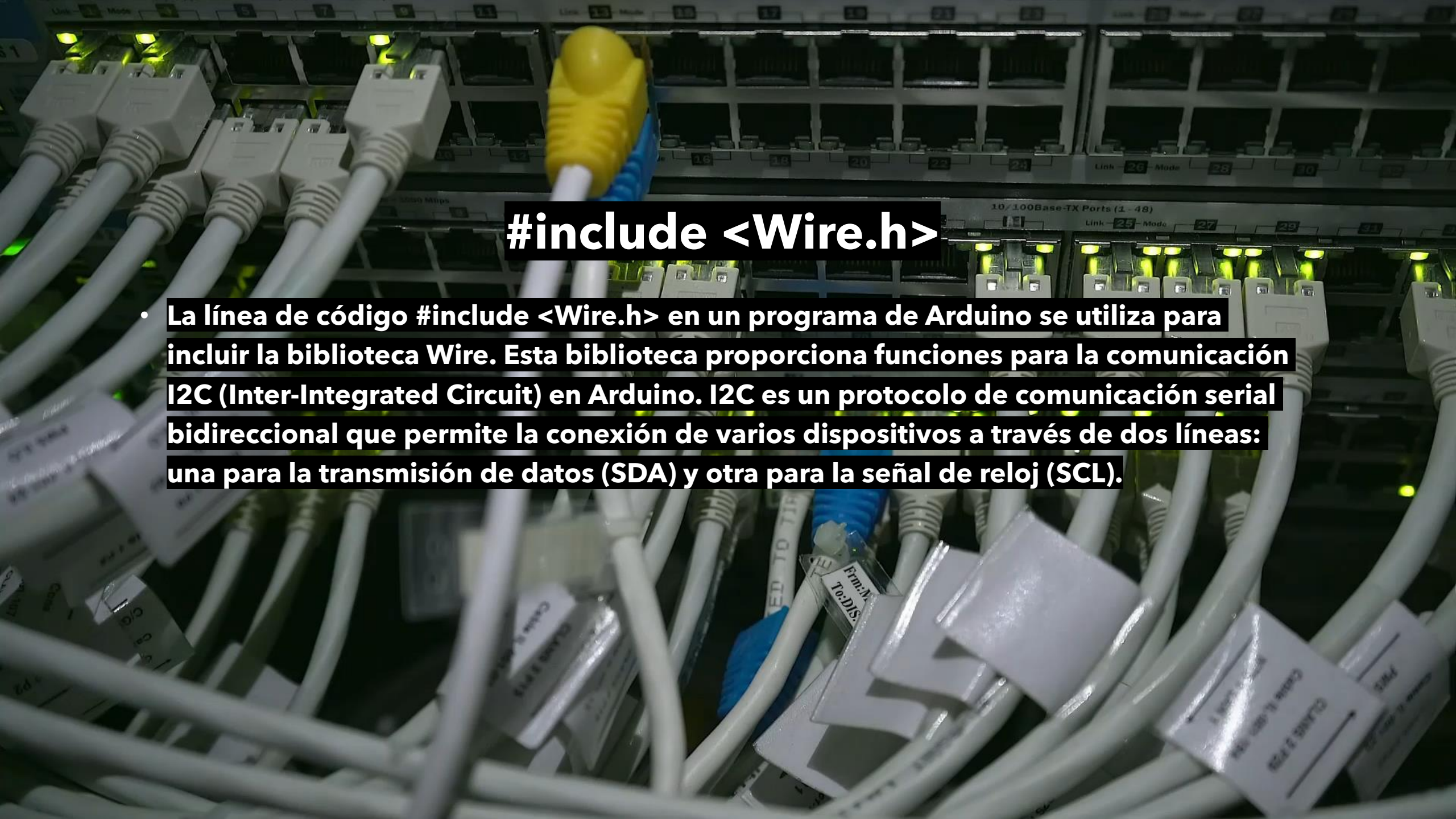
Explicación invernadero automatizado

Carlos González

carlos.gonzalezn@usm.cl

Explicación del código

```
/* importamos las librerias necesarias para el proyecto*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT11.h>
```

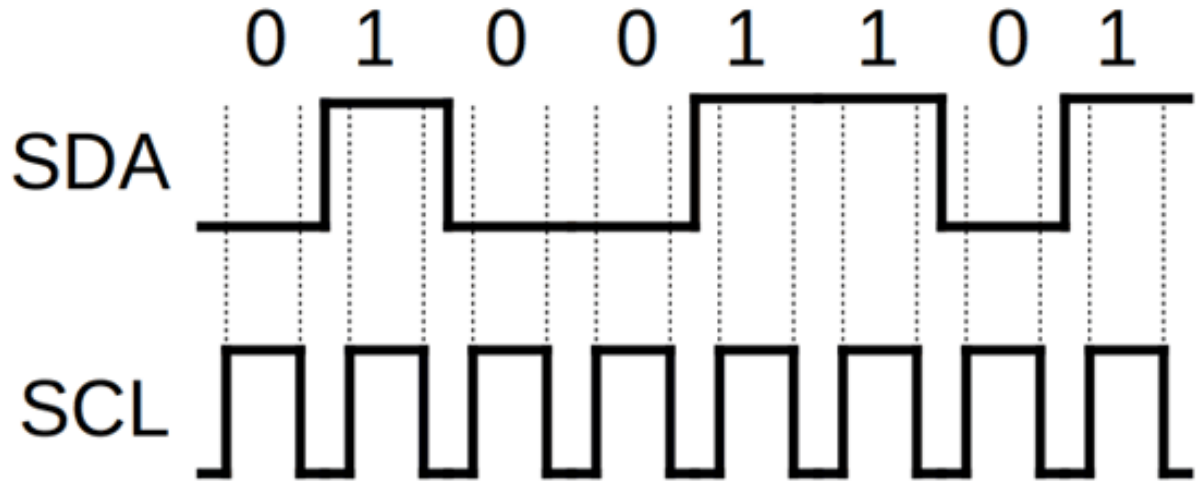


#include <Wire.h>

- La línea de código #include <Wire.h> en un programa de Arduino se utiliza para incluir la biblioteca Wire. Esta biblioteca proporciona funciones para la comunicación I2C (Inter-Integrated Circuit) en Arduino. I2C es un protocolo de comunicación serial bidireccional que permite la conexión de varios dispositivos a través de dos líneas: una para la transmisión de datos (SDA) y otra para la señal de reloj (SCL).

Protocolo de comunicación I2C

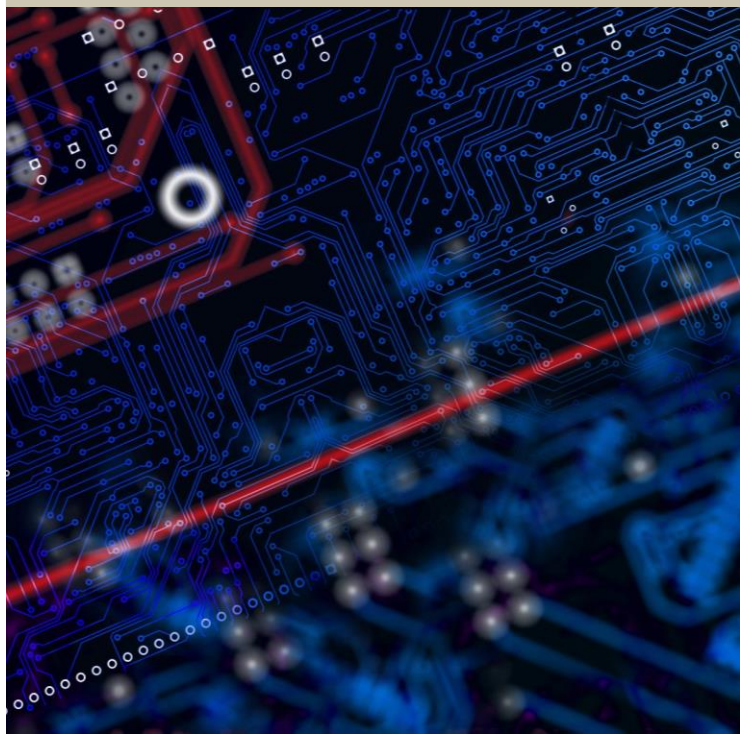
Decimal	Hexadecimal	Binario
77	0x4D	0b01001101



#include <LiquidCrystal_I2C.h>

- La línea de código #include <LiquidCrystal_I2C.h> en un programa de Arduino se utiliza para incluir la biblioteca LiquidCrystal_I2C. Esta biblioteca es comúnmente utilizada para controlar pantallas LCD (Liquid Crystal Displays) mediante la comunicación I2C.

- La ventaja de utilizar una pantalla LCD con comunicación I2C es que requiere menos pines en comparación con una conexión directa, lo que es especialmente útil cuando hay restricciones en la cantidad de pines disponibles en la placa Arduino.



#include <DHT11.h>

- La línea de código #include <DHT11.h> en un programa de Arduino se utiliza para incluir la biblioteca DHT11. Esta biblioteca es específica para trabajar con el sensor de temperatura y humedad DHT11 en proyectos de Arduino.

- Al incluir DHT11.h en el programa, habilitamos el uso de funciones y objetos específicos para interactuar con el sensor DHT11. Estas funciones incluyen la lectura de la temperatura y la humedad, así como la gestión de posibles errores durante la comunicación con el sensor.

```
#define printByte(args) write(args);
```

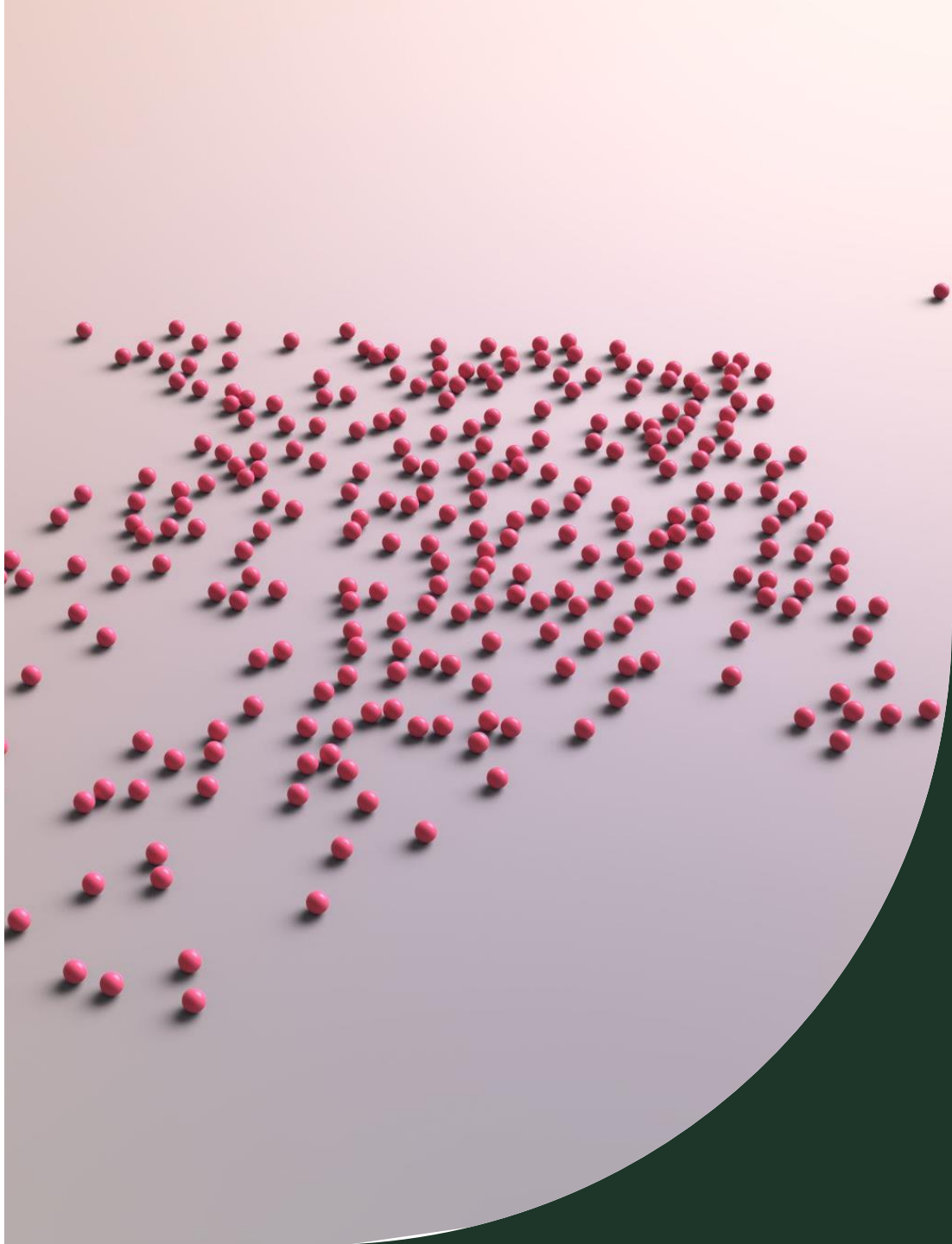
#define printByte(args) write(args);

- La línea de código #define printByte(args) write(args); define un macro en el código. En este caso, el macro se llama printByte y tiene un parámetro args. Cuando este macro se utiliza en el código, se sustituye por la instrucción write(args);.

```
/* Declaracion de variables */
unsigned int temperatura = 0;
unsigned int humedad = 0;
byte selector = 0;
byte contador = 0;
char *estadosDelSuelo[] = { "Seco", "Humedo", "Mojado" };
const int valorDelAire = 630;
const int valorDelAgua = 260;
int intervalos = (valorDelAire - valorDelAgua) / 3;
unsigned int humedadDelSuelo = 0;
byte agua = 7;
byte iluminacion = 6;
byte extractor = 5;
```

char *estadosDelSuelo[] = { "Seco", "Humedo", "Mojado" };

- Declara una variable llamada estadosDelSuelo y la inicializa como un array de punteros a caracteres (char *). Cada elemento del array es una cadena de caracteres que representa un estado del suelo.



- char *estadosDelSuelo[]: Declara una variable llamada estadosDelSuelo que es un array de punteros a caracteres (char *).

- char *estadosDelSuelo[]: Declara una variable llamada estadosDelSuelo que es un array de punteros a caracteres (char *).

- { "Seco", "Humedo", "Mojado" }: Inicializa el array con tres elementos, donde cada elemento es una cadena de caracteres representando un estado del suelo. En este caso, los estados son "Seco", "Humedo" y "Mojado".

- Por lo tanto, estadosDelSuelo es un array que contiene tres punteros a cadenas de caracteres, se puedes acceder a cada estado del suelo utilizando la notación de array. Por ejemplo, estadosDelSuelo[0] representará la cadena "Seco", estadosDelSuelo[1] representará "Humedo", y estadosDelSuelo[2] representará "Mojado".

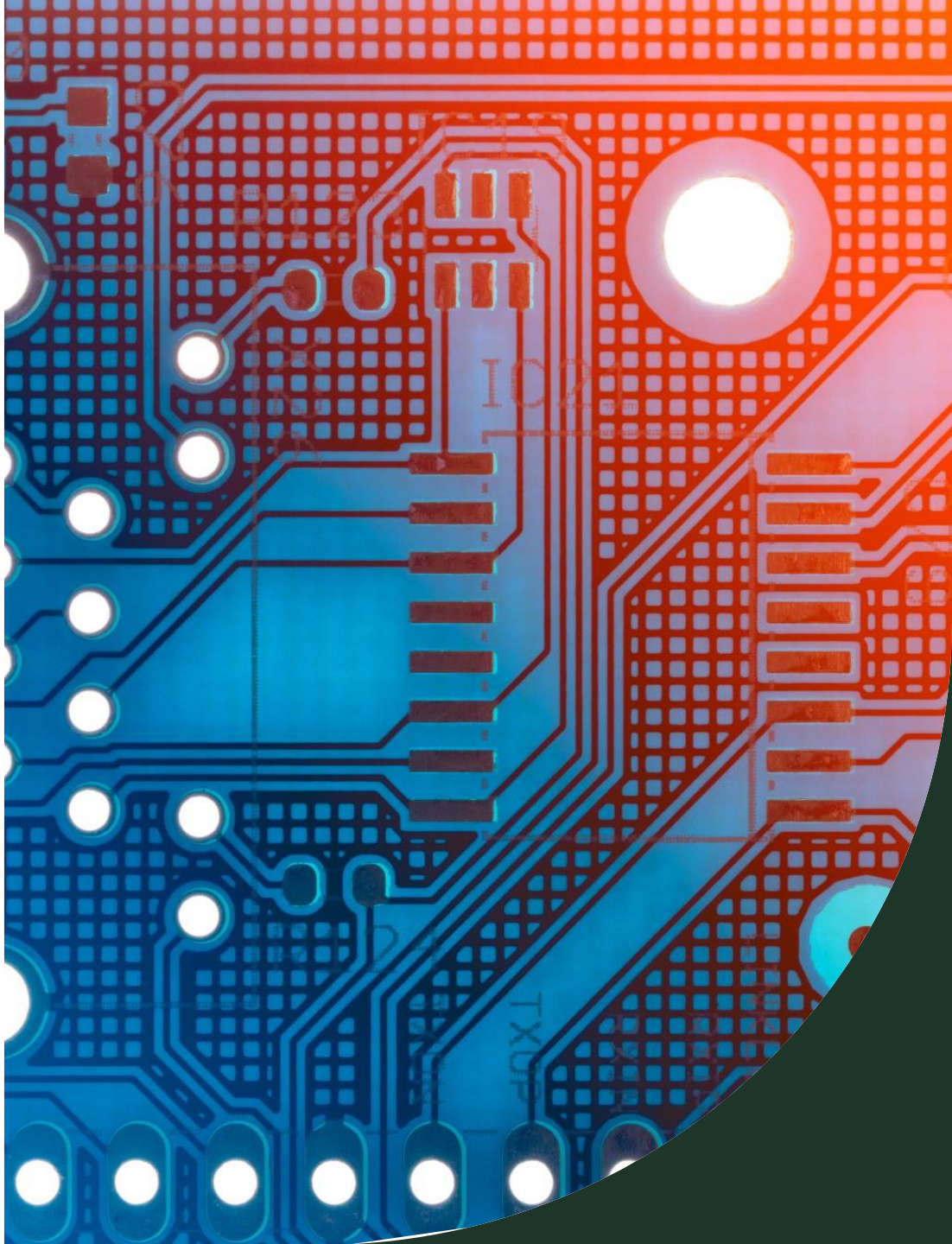
- En la línea de código char *estadosDelSuelo[] = { "Seco", "Humedo", "Mojado" };, el * indica que estadosDelSuelo es un array de punteros a caracteres (strings o cadenas de caracteres).

```
/* creacion de objetos necesarios para el proyecto */
DHT11 dht11(2);
LiquidCrystal_I2C lcd(0x27, 16, 2);
uint8_t grado[8] = { 0xe, 0x11, 0x11, 0xe, 0x0, 0x0, 0x0 };
```

DHT11 dht11(2);

- DHT11 dht11(2); se utiliza para declarar un objeto de la clase DHT11 llamado dht11 y asignarle el pin al que está conectado el sensor DHT11.

- DHT11: indica que estamos creando un objeto de la clase DHT11. La clase DHT11 suele ser una implementación de software para interactuar con el sensor de temperatura y humedad DHT11.



- dht11: Este es el nombre del objeto que estamos creando. Se usa este nombre para acceder a las funciones y variables asociadas con el objeto DHT11.

- (2): Esto es un argumento. En este caso, el argumento 2 indica el número del pin al que está conectado el sensor DHT11 en la placa Arduino.

LiquidCrystal_I2C lcd(0x27, 16, 2);

- La línea de código LiquidCrystal_I2C lcd(0x27, 16, 2); se utiliza para declarar un objeto de la clase LiquidCrystal_I2C llamado lcd y configurarlo con los parámetros específicos para controlar una pantalla LCD con interfaz I2C.

- LiquidCrystal_I2C: Esto indica que estamos creando un objeto de la clase
- lcd: Este es el nombre del objeto que estamos creando. Se usa este nombre para acceder a las funciones y variables asociadas con el objeto LiquidCrystal_I2C.
- 0x27: Esta es la dirección I2C de la pantalla LCD. La dirección I2C es un valor hexadecimal que identifica un dispositivo en el bus I2C. En este caso, 0x27 es una dirección I2C comúnmente utilizada para módulos de pantalla LCD con un módulo I2C incorporado.
- 16: Este es el número de columnas en la pantalla LCD. En este caso, la pantalla tiene 16 columnas.
- 2: Este es el número de filas en la pantalla LCD. En este caso, la pantalla tiene 2 filas.

uint8_t grado[8] = { 0xe, 0x11, 0x11, 0xe, 0x0, 0x0, 0x0 };

- La línea de código uint8_t grado[8] = { 0xe, 0x11, 0x11, 0xe, 0x0, 0x0, 0x0 }; se utiliza para declarar e inicializar un array de 8 elementos de tipo uint8_t (entero sin signo de 8 bits). Este array se llama grado, y cada elemento es un valor hexadecimal que representa un patrón gráfico.
- uint8_t: Esto indica que cada elemento del array es de tipo uint8_t, es decir, un entero sin signo de 8 bits.
- grado[8]: Esto declara el array llamado grado con un total de 8 elementos. Los elementos pueden ser accedidos utilizando índices que van desde 0 hasta 7.
- { 0xe, 0x11, 0x11, 0xe, 0x0, 0x0, 0x0 }: Estos son los valores iniciales proporcionados para cada elemento del array. Cada valor es un número hexadecimal de 8 bits (un byte). Estos representan un patrón gráfico utilizado para crear un carácter personalizado en una pantalla LCD.
- Este patrón grafico se utiliza para visualmente representar un símbolo de grado Celsius (°) en la pantalla LCD