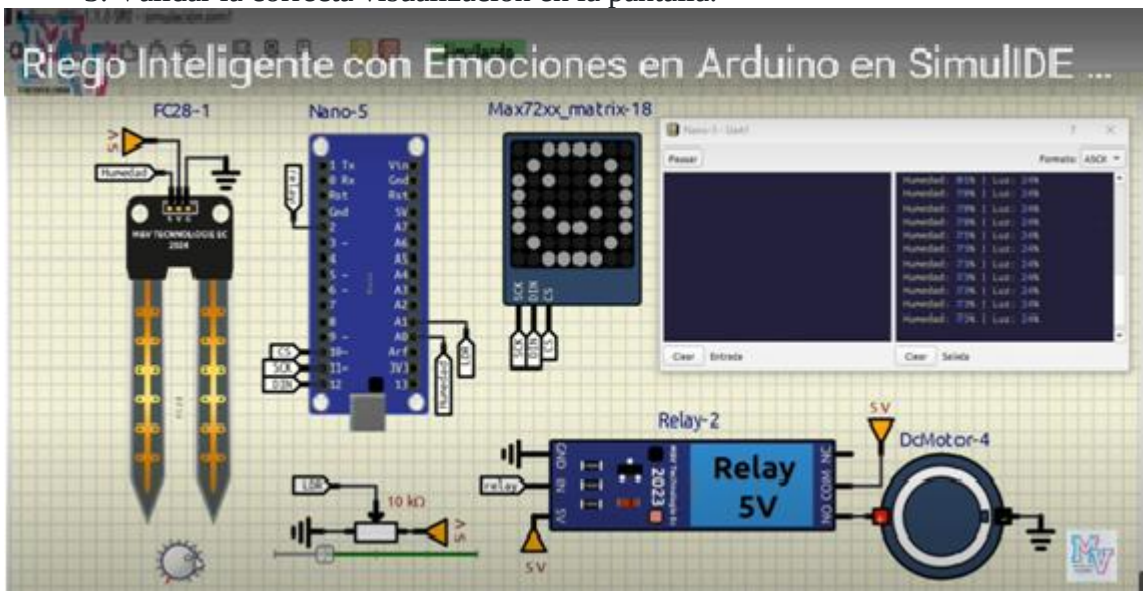
SENSORES Y ACTUADORES – TST 2024

Tp8-e4 #58. Salida a una pantalla LCD de 16x2 ó 20x4. de este Proyecto

- [x] 1. Configurar la pantalla LCD en el entorno de simulación.
- [x] 2. Implementar el código necesario para mostrar los valores de los sensores.
- [x] 3. Validar la correcta visualización en la pantalla.



Para implementar la simulación de un **Sistema de Riego Inteligente** en un entorno de simulación como **Wokwi** o **Proteus** usando un **Arduino** (o un ESP32), puedes seguir los pasos que se detallan a continuación. El sistema utilizará tres sensores: un sensor de humedad resistivo (YL-69 o similar), un sensor de humedad capacitivo (V1.2), y un sensor de lluvia resistivo (YL-83). Los datos de los sensores se mostrarán en una pantalla **LCD 16x2** o **20x4**.

Descripción del Proyecto:

El proyecto consiste en configurar y simular una pantalla LCD (de 16x2 o 20x4) para visualizar el estado del sistema, específicamente la humedad del suelo y el estado del riego. El objetivo es que el usuario pueda ver en tiempo real la información proporcionada por los sensores del sistema de monitoreo y control de riego.

Tareas Detalladas:

1. Configurar la pantalla LCD en el entorno de simulación

Descripción:
Esta tarea consiste en establecer y conectar correctamente una pantalla LCD (16x2 o 20x4) en un entorno de simulación, como **Wokwi** o **Proteus**, junto con los demás componentes del sistema (sensores de humedad y sistema de riego).

Pasos a seguir:

- Elegir el entorno de simulación adecuado (**Wokwi** o **Proteus**).
- Conectar la pantalla LCD al microcontrolador (Arduino, por ejemplo).
  - Conectar los pines de alimentación (VCC, GND).
  - Conectar los pines de control (RS, E).
  - Conectar los pines de datos (D4 a D7 si es en modo 4 bits o D0 a D7 si es en modo 8 bits).
  - Añadir un potenciómetro para controlar el contraste de la pantalla.
- Realizar una prueba básica de encendido y ajuste de contraste para verificar que la pantalla funciona.

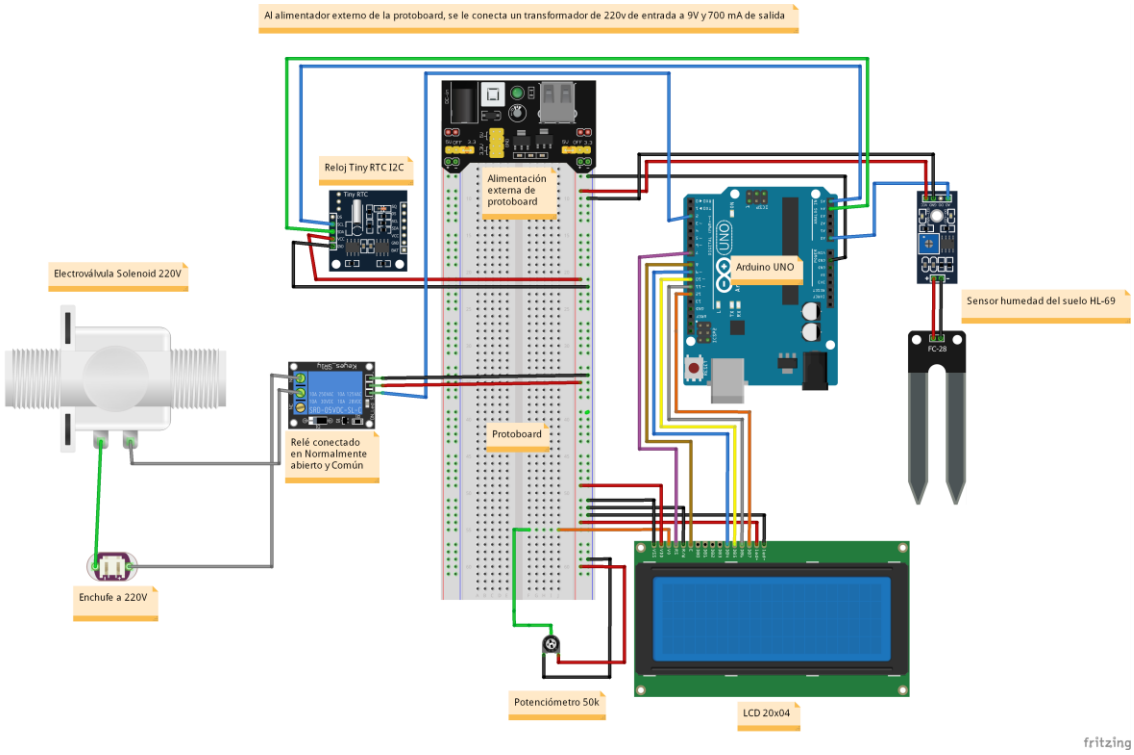
Entregable:

- Pantalla LCD conectada y lista para ser programada en el entorno de simulación.

2. Implementar el código necesario para mostrar los valores de los sensores

Descripción:
Implementar el código necesario para que la pantalla LCD muestre información relevante en tiempo real, como la humedad del suelo y el estado del sistema de riego (ON/OFF).
Pasos a seguir:

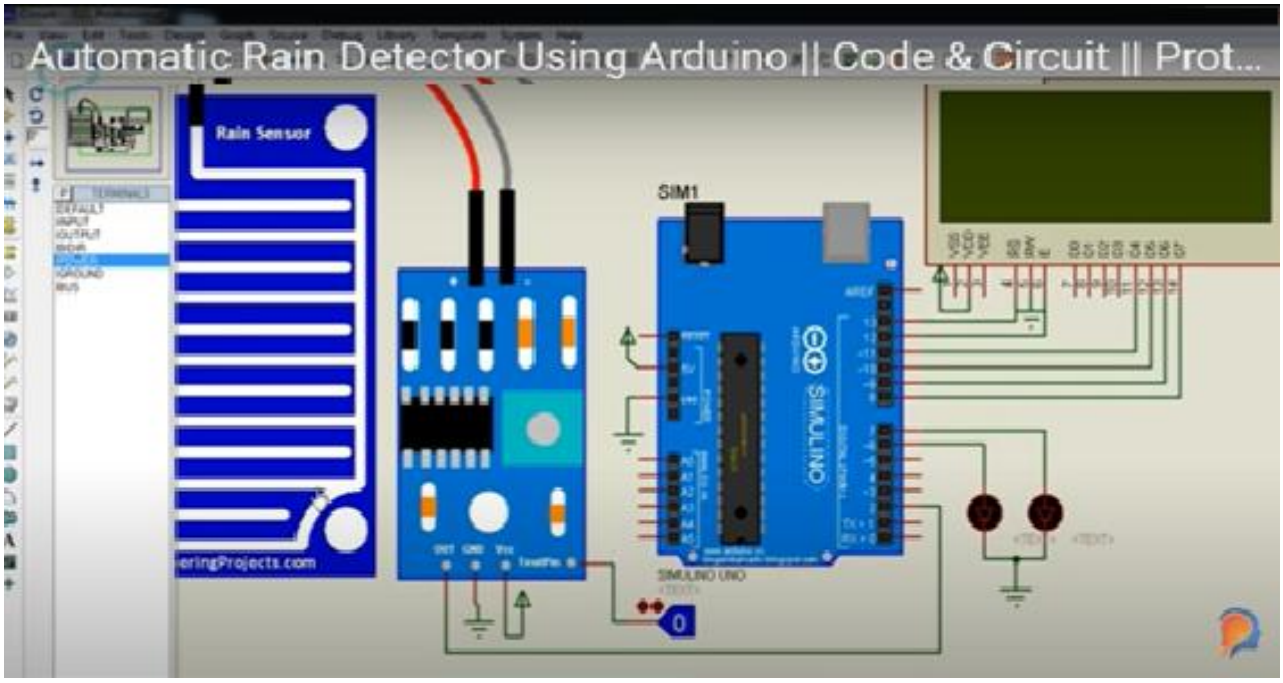
- Incluir las librerías necesarias para manejar la pantalla LCD. En caso de Arduino, por ejemplo, usar la librería `LiquidCrystal`.
- Configurar los pines en el código según la conexión física (RS, E, D4-D7).
- Crear las funciones que lean el valor de los sensores de humedad.
- Asignar condiciones lógicas para determinar el estado del riego (ej. Si la humedad está por debajo de un umbral, el sistema de riego está ON).
- Mostrar en la primera línea de la pantalla el valor de la humedad, y en la segunda línea el estado del riego.
- Actualizar los datos mostrados en la pantalla en tiempo real.



Paso 1: Configurar la pantalla LCD en el entorno de simulación

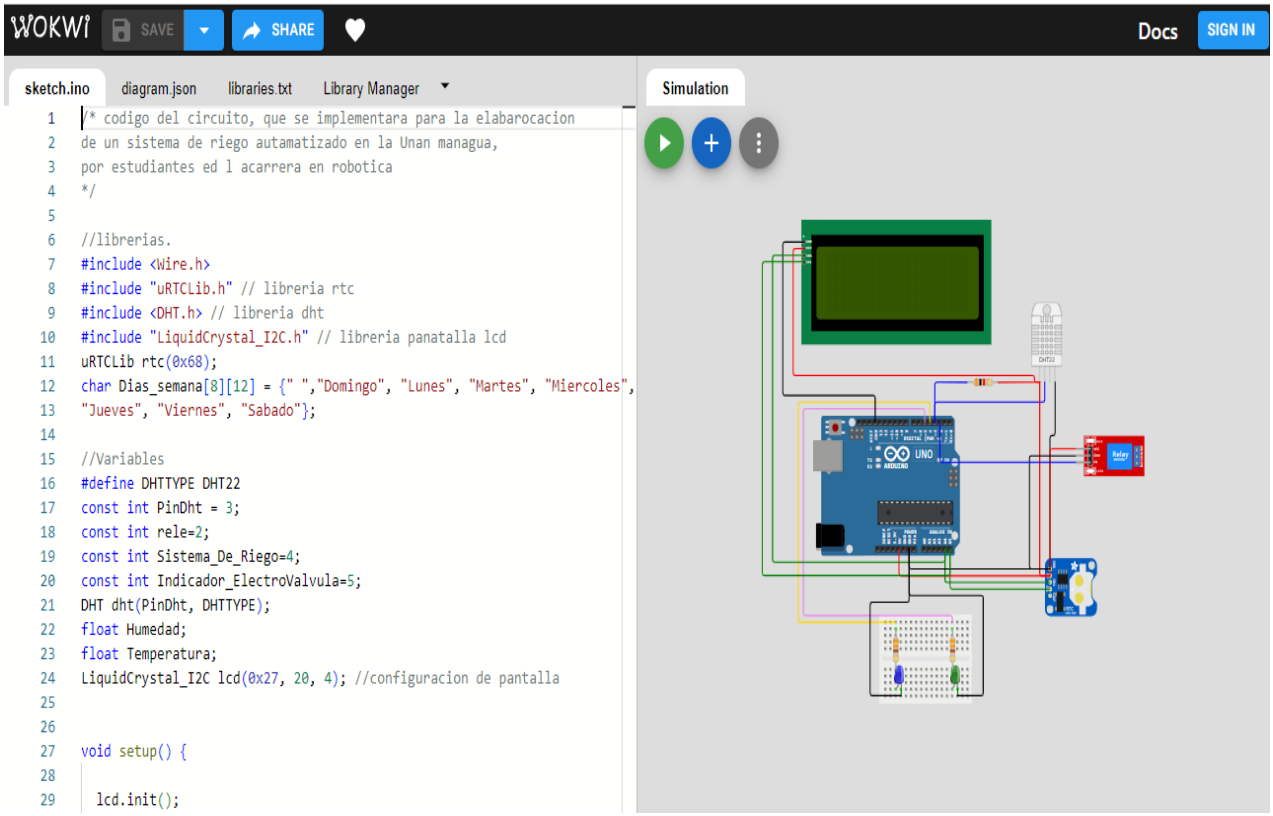
En **Wokwi** o **Proteus**, se debe configurar la conexión entre los pines del Arduino y la pantalla **LCD**. La pantalla LCD utilizará el protocolo **I2C** para simplificar las conexiones y reducir el número de pines.

- **Conexión en Wokwi** (puedes buscar un proyecto preconfigurado o crear uno nuevo):
  - Instancia un Arduino UNO o ESP32.
  - Agrega una pantalla **LCD 16x2** (o 20x4) con interfaz I2C.
  - Agrega los sensores:
    - **Sensor de humedad resistivo (YL-69 o HL-69)**.
    - **Sensor de humedad capacitivo (V1.2)**.
    - **Sensor de lluvia resistivo (YL-83)**.
  - Conecta los dispositivos a los pines adecuados según la tabla de pines (detallada en el paso de conexiones).



Paso 2: Implementar el código

A continuación, se presenta el código para leer los datos de los sensores y mostrar los valores en una pantalla LCD de 16x2.



Código de Arduino:

```
  // Determinar el estado del riego
  lcd.setCursor(0, 1);
  if (humedad < 500) {  // Ejemplo de umbral
    lcd.print("Riego: ON");
  } else {
    lcd.print("Riego: OFF");
  }

  delay(1000); // Actualizar cada segundo
}
```

```
#include <LiquidCrystal.h>

// Inicializar los pines del LCD: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Pin donde se conectará el sensor de humedad
int sensorPin = A0;
int humedad;

void setup() {
  lcd.begin(16, 2); // Inicializar la pantalla LCD 16x2
  pinMode(sensorPin, INPUT);
}

void loop() {
  // Leer el valor de humedad del sensor
  humedad = analogRead(sensorPin);

  // Limpiar la pantalla antes de mostrar nueva información
  lcd.clear();

  // Mostrar el valor de humedad en la primera línea
  lcd.setCursor(0, 0);
  lcd.print("Humedad: ");
  lcd.print(humedad);
```

Paso 3: Conexiones para la simulación

1. Conexiones de la pantalla LCD (con I2C):

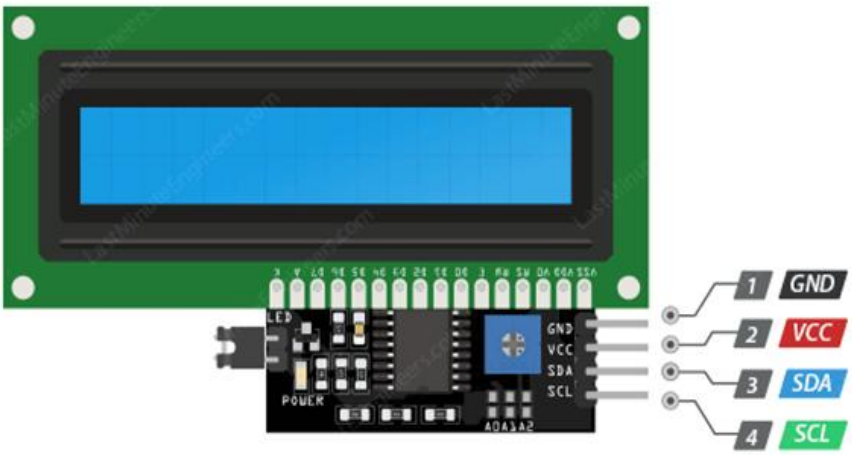
- **VCC** -> 5V del Arduino.
- **GND** -> GND del Arduino.
- **SDA** -> Pin A4 del Arduino.
- **SCL** -> Pin A5 del Arduino.

2. Conexiones de los sensores:

- **Sensor de Humedad Resistivo (YL-69, HL-69)**:
  - **VCC** -> 5V.
  - **GND** -> GND.
  - **A0** -> Pin A0 del Arduino (analogRead).
- **Sensor de Humedad Capacitivo (V1.2)**:
  - **VCC** -> 5V.
  - **GND** -> GND.
  - **A1** -> Pin A1 del Arduino (analogRead).
- **Sensor de Lluvia Resistivo (YL-83)**:
  - **VCC** -> 5V.
  - **GND** -> GND.
  - **A2** -> Pin A2 del Arduino (analogRead).

Configuración de pines de la pantalla LCD I2C

Una pantalla LCD I2C tiene solo 4 pines que la conectan con el mundo exterior. Las conexiones son las siguientes:



I2C LCD Pinout

Last Minute ENGINEERS.com

GND es un pin de tierra. Conéctalo a tierra del Arduino.

VCC suministra energía al módulo y al LCD. Conéctalo a la salida de 5V de Arduino o a una fuente de alimentación externa de 5V.

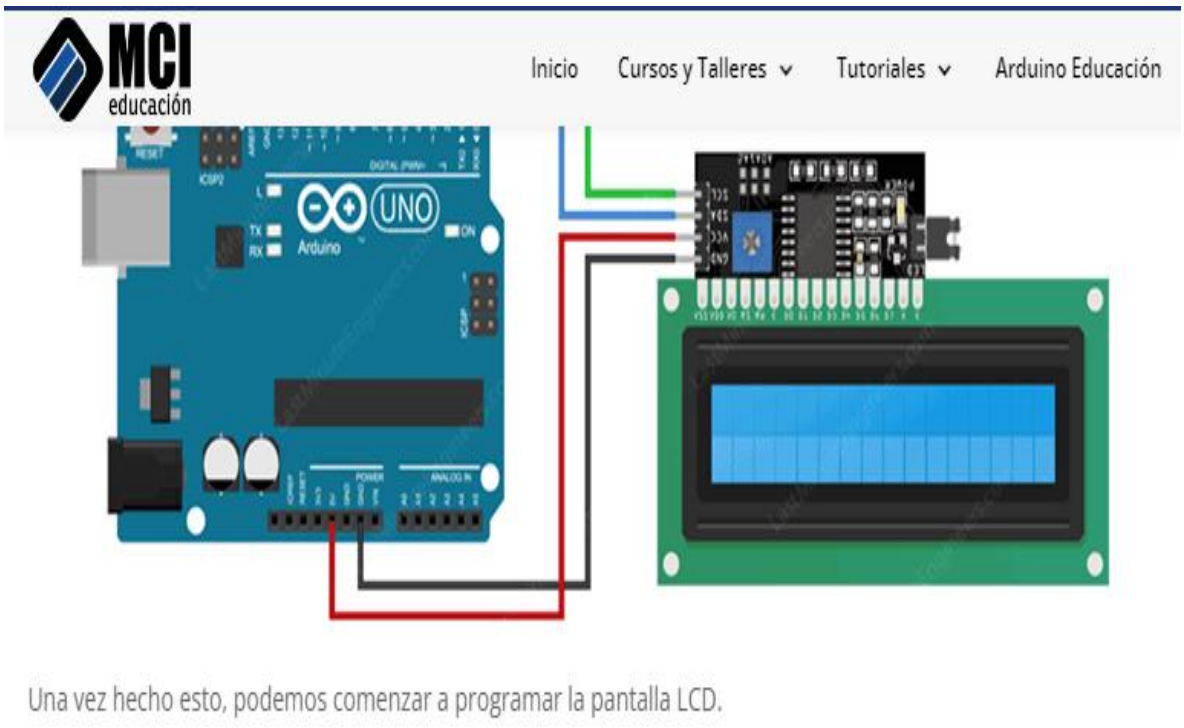
Paso 4: Validar la simulación

- Una vez que hayas implementado el circuito y cargado el código en el entorno de simulación, asegúrate de que los valores de los sensores se actualicen correctamente y se muestren en la pantalla LCD.
- Para validar, verifica que:
    - Los valores de humedad resistiva y capacitiva estén entre 0 y 100%.
    - El sensor de lluvia cambie entre "Lluvia" y "Seco" según el valor leído.

Estos ejemplos detallados de codificación, basados en el sistema de riego inteligente que quieres simular con los tres sensores: **Sensor de humedad resistivo (YL-69 o similar)**, **sensor de humedad capacitivo (V1.2)** y **sensor de lluvia resistivo (YL-83)**, mostrando los resultados en una pantalla **LCD I2C 16x2**.

Al dividir el ejemplo de la codificación en dos secciones:

1. **Código para Arduino en Wokwi (o Proteus)**.
2. **Explicación paso a paso del código y las conexiones**.



Una vez hecho esto, podemos comenzar a programar la pantalla LCD.

1. Código Completo para Arduino

El código de Arduino para leer los sensores y mostrar los valores en una simulación.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Definimos la dirección I2C de la pantalla (generalmente 0x27 o 0x3F)
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Definimos los pines para los sensores
const int sensorHumResistivoPin = A0;  // Sensor de humedad resistivo (YL-69, HL-69)
const int sensorHumCapacitivoPin = A1; // Sensor de humedad capacitivo (V1.2)
const int sensorLluviaPin = A2;        // Sensor de lluvia resistivo (YL-83)

void setup() {
  // Inicializamos la pantalla LCD
  lcd.begin(16, 2);
  lcd.backlight(); // Encendemos la retroiluminación

  // Inicializamos el monitor serie para depuración
  Serial.begin(9600);

  // Mostramos un mensaje de inicio en la pantalla
  lcd.setCursor(0, 0);
  lcd.print("Sistema Riego");
  lcd.setCursor(0, 1);
  lcd.print("Iniciando...");
  delay(2000);  // Pausa de 2 segundos para leer el mensaje
  lcd.clear();
}
```

```
void loop() {
  // Leemos los valores de los sensores
  int humedadResistivo = analogRead(sensorHumResistivoPin);  // Lectura del sensor resisti
  int humedadCapacitivo = analogRead(sensorHumCapacitivoPin); // Lectura del sensor capaci
  int lluvia = analogRead(sensorLluviaPin); // Lectura del sensor de lluvia

  // Convertimos los valores de los sensores a porcentajes o rangos legibles
  int humedadResistivoPorc = map(humedadResistivo, 1023, 0, 0, 100);  // Escalamos a porce
  int humedadCapacitivoPorc = map(humedadCapacitivo, 1023, 0, 0, 100);
  String estadoLluvia = lluvia < 500 ? "Lluvia" : "Seco";  // Definimos si hay lluvia o no

  // Mostramos los datos en el monitor serie para depuración
  Serial.print("Humedad Resistivo: ");
  Serial.print(humedadResistivoPorc);
  Serial.println("%");

  Serial.print("Humedad Capacitivo: ");
  Serial.print(humedadCapacitivoPorc);
  Serial.println("%");

  Serial.print("Estado de Lluvia: ");
  Serial.println(estadoLluvia);
```

///…

...///

```
  // Mostramos los datos en la pantalla LCD
  lcd.setCursor(0, 0);  // Primera fila
  lcd.print("Hum. Res: ");
  lcd.print(humedadResistivoPorc);
  lcd.print("%");

  lcd.setCursor(0, 1);  // Segunda fila
  lcd.print("Hum. Cap: ");
  lcd.print(humedadCapacitivoPorc);
  lcd.print("%");

  lcd.setCursor(11, 1);  // Segunda fila, después de los datos de humedad
  lcd.print(estadoLluvia);

  delay(2000);  // Actualización cada 2 segundos
}
```

2. Explicación del código

Librerías utilizadas

- **Wire.h**: Esta librería se utiliza para la comunicación **I2C** entre el Arduino y la pantalla LCD.
- **LiquidCrystal_I2C.h**: Se encarga de controlar la pantalla **LCD 16x2** o **20x4** utilizando el protocolo **I2C**.

Inicialización de la pantalla LCD

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

En este caso, la pantalla LCD tiene una dirección I2C de **0x27**, que es la dirección común para muchos módulos LCD I2C. Si no funciona en tu simulación, se puede probar con la dirección **0x3F**.

Pines de los sensores

```
const int sensorHumResistivoPin = A0;  // Pin del sensor resistivo
const int sensorHumCapacitivoPin = A1; // Pin del sensor capacitivo
const int sensorLluviaPin = A2;        // Pin del sensor de lluvia
```

Definir los pines **A0**, **A1** y **A2** para los sensores.

Configuración en el setup()

```
void setup() {
  lcd.begin(16, 2);  // Configura la pantalla LCD de 16x2
  lcd.backlight();    // Habilita la retroiluminación
  Serial.begin(9600); // Inicia el monitor serie para ver los datos en tiempo real
}
```

Iniciar la pantalla **LCD** con el tamaño de **16x2**, activamos la retro iluminación y también inicializamos el **monitor serie** a 9600 baudios para depuración.

Lectura de los sensores en `loop()`
Leer los valores de los tres sensores utilizando la función **`analogRead().`**

```
int humedadResistivo = analogRead(sensorHumResistivoPin);
int humedadCapacitivo = analogRead(sensorHumCapacitivoPin);
int lluvia = analogRead(sensorLluviaPin);
```

Conversión de valores a porcentaje
Para tener los valores en un rango más intuitivo (0 a 100%), usamos la función `map()` que escala los valores de los sensores: El estado de lluvia se determina comparando el valor leído con un umbral. En este caso, si el valor es menor a 500, se considera que está lloviendo:

```
int humedadResistivoPorc = map(humedadResistivo, 1023, 0, 0, 100);
int humedadCapacitivoPorc = map(humedadCapacitivo, 1023, 0, 0, 100);
```

```
String estadoLluvia = lluvia < 500 ? "Lluvia" : "Seco";
```

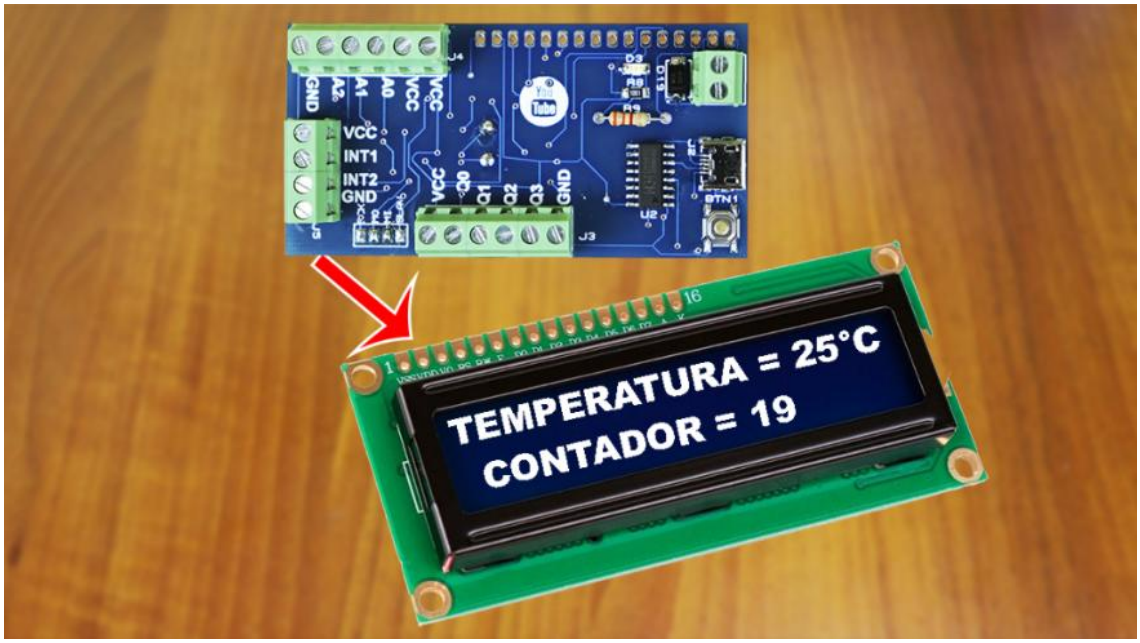
Mostrar valores en el LCD

```
lcd.setCursor(0, 0);  // Primera fila
lcd.print("Hum. Res: ");
lcd.print(humedadResistivoPorc);
lcd.print("%");

lcd.setCursor(0, 1);  // Segunda fila
lcd.print("Hum. Cap: ");
lcd.print(humedadCapacitivoPorc);
lcd.print("%");

lcd.setCursor(11, 1);  // Estado de lluvia
lcd.print(estadoLluvia);
```

La pantalla mostrará los porcentajes de humedad resistiva y capacitiva, además del estado de lluvia ("Lluvia" o "Seco").



Espera y actualización de datos

```
delay(2000);  // Espera 2 segundos antes de actualizar los datos
```

Este delay asegura que los valores se actualicen cada 2 segundos.

3. Conexiones para la simulación

Conexión del LCD I2C:

- **VCC** -> 5V del Arduino.
- **GND** -> GND del Arduino.
- **SDA** -> Pin A4 del Arduino.
- **SCL** -> Pin A5 del Arduino.

Conexión de los sensores:

- **Sensor de humedad resistivo (YL-69)**:
  - **VCC** -> 5V.
  - **GND** -> GND.
  - **Pin de señal** -> A0 (analogRead).
- **Sensor de humedad capacitivo (V1.2)**:
  - **VCC** -> 5V.
  - **GND** -> GND.
  - **Pin de señal** -> A1 (analogRead).
- **Sensor de lluvia resistivo (YL-83)**:
  - **VCC** -> 5V.
  - **GND** -> GND.
  - **Pin de señal** -> A2 (analogRead).

4. Simulación en Wokwi

En **Wokwi**, puedes montar este mismo esquema conectando los pines correspondientes en la simulación, asegurándote de que la pantalla **LCD I2C** esté conectada correctamente (SDA, SCL) y los sensores a los pines **A0**, **A1** y **A2**.

Ejemplo de código:

```
#include <LiquidCrystal.h>

// Inicializar los pines del LCD: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Definir el pin del sensor de humedad
const int sensorPin = A0;

// Definir umbral para activar/desactivar el riego
const int umbralRiego = 500;

// Variable para almacenar el valor de la humedad
int humedad;

void setup() {
  // Inicializar la pantalla LCD con 16 columnas y 2 filas
  lcd.begin(16, 2);

  // Mostrar mensaje inicial en la pantalla
  lcd.setCursor(0, 0);
  lcd.print("Sistema Riego");
  lcd.setCursor(0, 1);
  lcd.print("Iniciando...");
  delay(2000);  // Mostrar por 2 segundos

  // Limpiar pantalla después del mensaje inicial
  lcd.clear();
```

/ / /...

.../ / /

```
  // Mostrar el valor de la humedad en la primera línea
  lcd.setCursor(0, 0);  // Colocar el cursor en la primera línea
  lcd.print("Humedad: ");
  lcd.print(humedad);  // Mostrar el valor de la humedad

  // Verificar el estado del riego
  lcd.setCursor(0, 1);  // Colocar el cursor en la segunda línea
  if (humedad < umbralRiego) {
    lcd.print("Riego: ON");  // Mostrar que el riego está activo
  } else {
    lcd.print("Riego: OFF");  // Mostrar que el riego está inactivo
  }

  // Esperar 1 segundo antes de actualizar de nuevo
  delay(1000);
}
```

Entregable:

- Código funcional para mostrar la humedad del suelo y el estado del riego en la pantalla LCD.

3. Validar la correcta visualización en la pantalla

Descripción:

Se debe verificar que la información mostrada en la pantalla LCD sea legible y corresponda a los valores obtenidos de los sensores en tiempo real.

Pasos a seguir:

- Ejecutar la simulación y observar el comportamiento de la pantalla.
- Validar que el valor de la humedad se actualiza correctamente según el sensor.
- Verificar que el estado del riego cambia de ON a OFF o viceversa cuando la humedad supera o baja de un umbral determinado.
- Ajustar el contraste de la pantalla para asegurar la visibilidad adecuada.
- Documentar posibles errores o problemas en la visualización y corregirlos.

Entregable:

- Capturas de pantalla de la simulación mostrando los datos correctos en la pantalla LCD.
- Informe de validación que incluya los resultados obtenidos y posibles ajustes realizados.

Roles y Responsabilidades:

Responsable:

Un integrante del equipo será el responsable de llevar a cabo estas tareas, con el apoyo del resto del equipo en la medida en que se necesite verificar el funcionamiento conjunto de todo el sistema (sensores, riego, LCD).

Recursos Recomendados:

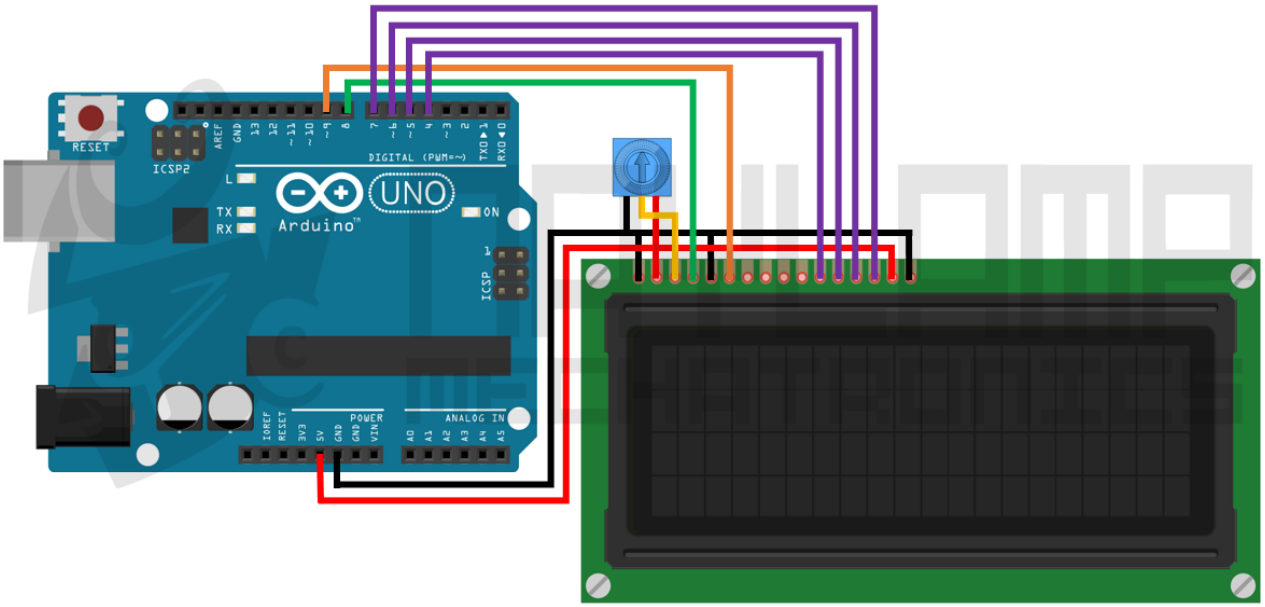
- Entorno de simulación: Tinkercad, Proteus.
- Librería de Arduino para pantalla LCD: **`LiquidCrystal.`**
- Documentación sobre el uso de sensores de humedad del suelo, como el sensor de humedad capacitivo o resistivo.

Tiempo Estimado:

1. Configuración del entorno y simulación de la pantalla LCD: **2 horas**
2. Implementación del código para la lectura y visualización de datos: **3-4 horas**
3. Validación de la visualización y ajustes: **1-2 horas**

Con estos pasos, se podrá simular con éxito una pantalla LCD que muestre en tiempo real el estado del sistema de riego basado en la humedad del suelo.

Un ejemplo de código para la simulación de un sistema que utiliza una pantalla LCD de 16x2 o 20x4 conectada a un **Arduino** para mostrar el estado de la **humedad del suelo** y el **estado del riego**. Se asume que el sensor de humedad del suelo está conectado al pin analógico A0 y que la pantalla LCD está conectada en modo de 4 bits.



Ejemplo de Código:

```
#include <LiquidCrystal.h>

// Inicializar los pines del LCD: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Definir el pin del sensor de humedad
const int sensorPin = A0;

// Definir umbral para activar/desactivar el riego
const int umbralRiego = 500;

// Variable para almacenar el valor de la humedad
int humedad;

void setup() {
  // Inicializar la pantalla LCD con 16 columnas y 2 filas
  lcd.begin(16, 2);

  // Mostrar mensaje inicial en la pantalla
  lcd.setCursor(0, 0);
  lcd.print("Sistema Riego");
  lcd.setCursor(0, 1);
  lcd.print("Iniciando...");
  delay(2000);  // Mostrar por 2 segundos

  // Limpiar pantalla después del mensaje inicial
  lcd.clear();

  // Configurar el pin del sensor de humedad como entrada
  pinMode(sensorPin, INPUT);
}

void loop() {
  // Leer el valor de humedad del sensor
  humedad = analogRead(sensorPin);

  // Limpiar la pantalla antes de mostrar nueva información
  lcd.clear();

  // Mostrar el valor de la humedad en la primera línea
  lcd.setCursor(0, 0);  // Colocar el cursor en la primera línea
  lcd.print("Humedad: ");
  lcd.print(humedad);  // Mostrar el valor de la humedad
```

///…

…///

```
  // Verificar el estado del riego
  lcd.setCursor(0, 1);  // Colocar el cursor en la segunda línea
  if (humedad < umbralRiego) {
    lcd.print("Riego: ON");  // Mostrar que el riego está activo
  } else {
    lcd.print("Riego: OFF");  // Mostrar que el riego está inactivo
  }

  // Esperar 1 segundo antes de actualizar de nuevo
  delay(1000);
}
```

Explicación del Código:

1. **Librería `LiquidCrystal`:**
   Se incluye la librería `LiquidCrystal` que se utiliza para manejar la pantalla LCD. La pantalla está conectada a los pines digitales del Arduino (12, 11, 5, 4, 3, 2).
2. **Definir pines y variables:**
   Se define el pin del sensor de humedad en **A0**. También se define un umbral **(`umbralRiego`)** de 500, que será el valor límite para decidir si el sistema de riego debe estar encendido o apagado.
3. **setup()**
   En esta función se inicializa la pantalla LCD con el método **lcd.begin** `(16, 2)` (para una pantalla de 16x2). Se muestra un mensaje inicial "Sistema Riego Iniciando...". Tras una breve pausa de 2 segundos, la pantalla se limpia para comenzar a mostrar los datos en tiempo real.
4. **loop()**
   El código entra en un bucle continuo donde:
   - Se lee el valor de humedad desde el pin analógico **A0** con **analogRead().**
   - Se limpia la pantalla con **lcd.clear()** antes de actualizar la información.
   - Se muestra el valor de la humedad en la primera línea de la pantalla.
   - Se verifica si el valor de humedad es inferior al umbral definido:
     - Si es menor al umbral, el sistema de riego se activa y se muestra **"Riego: ON".**
     - Si es mayor o igual al umbral, el riego se apaga y se muestra **"Riego: OFF".**
   - La pantalla se actualiza cada segundo con `delay(1000)`.

Diagrama de Conexiones:

- **LCD (16x2 o 20x4):**
  - VSS -> GND
  - VDD -> 5V
  - V0 -> Potenciómetro central (para ajuste de contraste)
  - RS -> Pin digital 12
  - E -> Pin digital 11
  - D4 -> Pin digital 5
  - D5 -> Pin digital 4
  - D6 -> Pin digital 3
  - D7 -> Pin digital 2
  - RW -> GND (para modo de escritura)
  - A -> 5V (retroiluminación)
  - K -> GND (retroiluminación)
- **Sensor de humedad del suelo:**
  - VCC -> 5V
  - GND -> GND
  - Signal (señal) -> Pin analógico A0

Simulación en Wokwi (o Proteus):

Para realizar la simulación en **Wokwi (o Proteus)**

1. **Crear el circuito:**
   - Arrastra el componente de la pantalla LCD de 16x2 o 20x4.
   - Conecta el potenciómetro al pin V0 de la pantalla para ajustar el contraste.
   - Conecta el sensor de humedad a los pines de alimentación y señal.
2. **Cargar el código en el entorno de simulación:**
   - Copia y pega el código en el IDE del simulador.
   - Ejecuta la simulación para observar cómo los datos de humedad se muestran en tiempo real en la pantalla LCD.

Con este código y las instrucciones, se está mostrando continuamente el valor de la humedad del suelo y el estado del riego en una pantalla LCD de 16x2 (o 20x4). La simulación te permitirá validar el funcionamiento del sistema antes de implementarlo físicamente.

Una característica para **activar un sistema de riego** según las lecturas del sensor de humedad es una excelente mejora. Esto permitirá automatizar el riego en función de las condiciones del suelo.

Nueva Funcionalidad: Automatización del Riego

La idea es que cuando el valor de humedad del suelo esté por debajo de un umbral predefinido, se active automáticamente un sistema de riego (como una bomba de agua). Cuando la humedad supere ese umbral, el riego se apagará.

Componentes adicionales:

- **Bomba de agua o LED (simulando la bomba)** conectada a un pin digital para activarla o desactivarla según el nivel de humedad del suelo.

Programador de Riego Automático

Pantalla LCD de grandes dimensiones con interfaz de usuario de navegación sencilla. Se monta con sólo dos tornillos.

CódigoModificado:

```
#include <LiquidCrystal.h>

// Inicializar los pines del LCD: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Definir el pin del sensor de humedad y el pin de la bomba
const int sensorPin = A0;
const int bombaPin = 9;  // Pin digital que controla la bomba de riego

// Definir umbral para activar/desactivar el riego
const int umbralRiego = 500;

// Variable para almacenar el valor de la humedad
int humedad;

void setup() {
  // Inicializar la pantalla LCD con 16 columnas y 2 filas
  lcd.begin(16, 2);

  // Mostrar mensaje inicial en la pantalla
  lcd.setCursor(0, 0);
  lcd.print("Sistema Riego");
  lcd.setCursor(0, 1);
  lcd.print("Iniciando...");
  delay(2000);  // Mostrar por 2 segundos

  // Limpiar pantalla después del mensaje inicial
  lcd.clear();

  // Configurar el pin del sensor de humedad como entrada y el pin de la bomba como salida
  pinMode(sensorPin, INPUT);
  pinMode(bombaPin, OUTPUT);

  // Apagar la bomba inicialmente
  digitalWrite(bombaPin, LOW);
}

void loop() {
  // Leer el valor de humedad del sensor
  humedad = analogRead(sensorPin);

  // Limpiar la pantalla antes de mostrar nueva información
  lcd.clear();

  // Mostrar el valor de la humedad en la primera línea
  lcd.setCursor(0, 0);  // Colocar el cursor en la primera línea
  lcd.print("Humedad: ");
  lcd.print(humedad);  // Mostrar el valor de la humedad
```

///...

```
  // Verificar el estado del riego
  lcd.setCursor(0, 1);  // Colocar el cursor en la segunda línea
  if (humedad < umbralRiego) {
    lcd.print("Riego: ON");  // Mostrar que el riego está activo
    digitalWrite(bombaPin, HIGH);  // Encender la bomba
  } else {
    lcd.print("Riego: OFF");  // Mostrar que el riego está inactivo
    digitalWrite(bombaPin, LOW);  // Apagar la bomba
  }

  // Esperar 1 segundo antes de actualizar de nuevo
  delay(1000);
}
```

…///

Explicación de la Funcionalidad de Riego Automático:

1. **Pin de Control de la Bomba:**
   Se utiliza el pin **9** (puedes cambiarlo según tu configuración) para controlar la bomba de riego, que se activa o desactiva según la lectura del sensor de humedad.
2. **Lógica del Riego:**
   - **Riego ON:** Si la humedad está por debajo del umbral definido (500 en este caso), el sistema activará la bomba (poniendo en alto el pin correspondiente).
   - **Riego OFF:** Si la humedad es igual o superior al umbral, la bomba se apagará (poniendo en bajo el pin correspondiente).
3. **Actualización Visual:**
   - En la pantalla LCD, se muestra el valor actual de la humedad y si el sistema de riego está activo ("Riego: ON") o inactivo ("Riego: OFF") en tiempo real.

Diagrama de Conexiones Adicionales:

- **Bomba de agua o LED:**
  - **BombaPin** (Pin 9) -> Control de la bomba o LED (que simula el riego)
  - **GND** -> GND
  - **VCC** -> Fuente de alimentación adecuada para la bomba (si usas una bomba real).

Pruebas y Simulación:

- En un simulador como **Tinkercad**, puedes usar un LED o motor para representar el sistema de riego (bomba).
- Configura diferentes valores de humedad en el simulador para observar cómo el sistema enciende y apaga la bomba automáticamente.

Posibles Mejoras Futuras:

- **Control de tiempo de riego:** Puedes agregar un temporizador para que la bomba funcione solo durante un tiempo determinado cuando se activa.
- **Alerta de nivel de agua bajo:** Agregar un sensor de nivel de agua que apague la bomba si el tanque está vacío.

Con esta nueva característica, el sistema no solo monitorea la humedad, sino que también **automatiza el riego**, proporcionando una solución más completa para el control de riego.

Conclusión:

El proyecto logró configurar y simular una pantalla LCD de 16x2 para mostrar en tiempo real la humedad del suelo y el estado del riego, usando un entorno de simulación como **Wokwi (o Proteus)**. La visualización en tiempo real permite un monitoreo eficiente, y el sistema es flexible y ajustable para diferentes condiciones. Además, la simulación ayuda a validar el diseño antes de implementarlo físicamente. En el futuro, el sistema puede expandirse con más sensores y automatización del riego, mejorando su funcionalidad.

Aspectos Clave Alcanzados:

1. **Simulación Correcta del Sistema:** La pantalla LCD fue configurada en modo 4 bits, optimizando el uso de pines, y la simulación mostró con precisión cómo el sistema reacciona a los datos del sensor de humedad del suelo.
2. **Visualización en Tiempo Real:** Los usuarios pueden monitorear fácilmente la humedad del suelo y el estado del riego, con actualizaciones en tiempo real para garantizar información precisa.
3. **Flexibilidad y Ajuste del Sistema:** El umbral de humedad es ajustable para adaptarse a diferentes cultivos, y un potenciómetro permite ajustar el contraste de la pantalla para asegurar una buena visibilidad.

Recomendaciones Futuras:

- **Expansión del Sistema:** Integrar más sensores (como temperatura o luz) y considerar pantallas más avanzadas, como OLED o TFT, para mayor información.
- **Automatización del Riego:** Automatizar el riego con una bomba controlada por el microcontrolador cuando el nivel de humedad sea bajo.

Beneficios del Sistema:

El sistema permite un monitoreo eficiente del riego, con información clara y accesible. Su simulación en plataformas como Tinkercad ayuda a validar el diseño antes de implementarlo físicamente, ahorrando tiempo y evitando errores.

Este proyecto ofrece una solución clara, fácil de implementar y eficiente para la visualización del estado de riego y la humedad del suelo, siendo una base sólida para un sistema de control de riego automatizado.

ANEXO

Tutorial Arduino: Pantalla LCD

Cómo usar un LCD de 16×2 caracteres con Arduino

LCD y Arduino

Este tutorial incluye todo lo que necesitas saber sobre el control de una pantalla LCD de caracteres con Arduino. Hemos incluido un diagrama de cableado y muchos códigos de ejemplo. Estas pantallas, LCD, son ideales para mostrar datos o texto de los sensores y también son bastante baratas.
La primera parte de este artículo cubre los aspectos básicos de la visualización de texto y números. En la segunda mitad, entraremos en más detalles sobre cómo mostrar caracteres personalizados y cómo puedes usar las otras funciones de la biblioteca LiquidCrystal Arduino.
Como verás, se necesitan bastantes conexiones para controlar estas pantallas. Por lo tanto, se puede utilizar con un módulo de interfaz I2C montado en la parte posterior. Con este módulo I2C, sólo se necesitan dos conexiones para controlar la pantalla LCD.

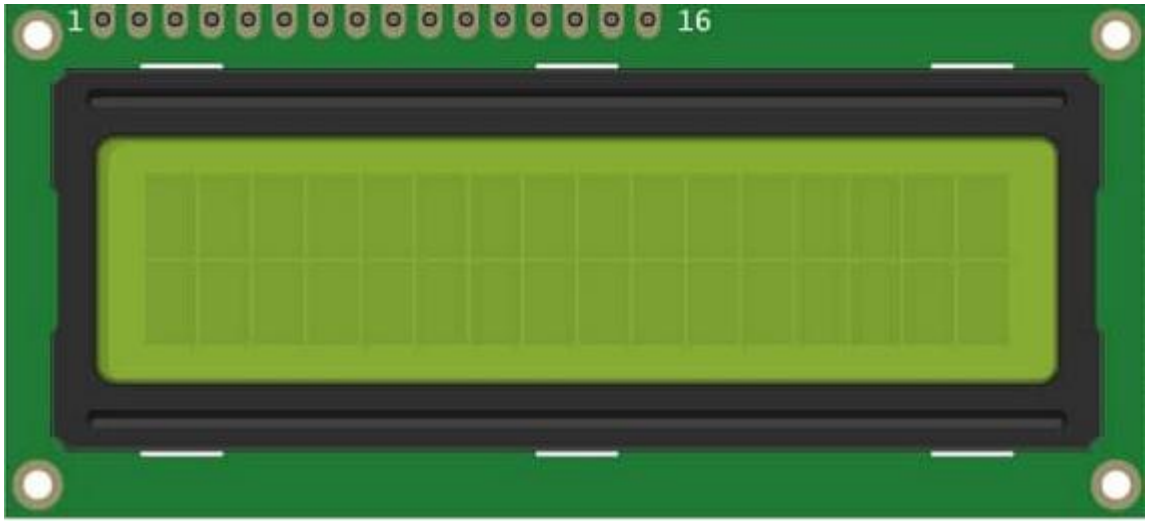
Descripción general del hardware

Estos LCD están disponibles en muchos tamaños diferentes (16×2 1602, 20×4 2004, 16×1, etc.), pero todos utilizan el mismo chip controlador LCD de interfaz paralela HD44780 de Hitachi. Esto significa que puedes cambiarlos fácilmente. Sólo tendrás que cambiar las especificaciones de tamaño en tu código de Arduino.

- Especificaciones de la pantalla: LCD 16×2
- Tensión de funcionamiento: 5 V
- Controlador Hitachi HD44780 Controlador LCD
- Resolución de pantalla 2 líneas x 16 caracteres
- Resolución de caracteres 5 x 8 píxeles
- Dimensiones del módulo 80 x 36 x 12 mm
- Dimensiones del área de visualización 64,5 x 16,4 mm

Pinout LCD 16×2

La pantalla LCD tiene 16 pines de conexión, numerados del 1 al 16 de izquierda a derecha.

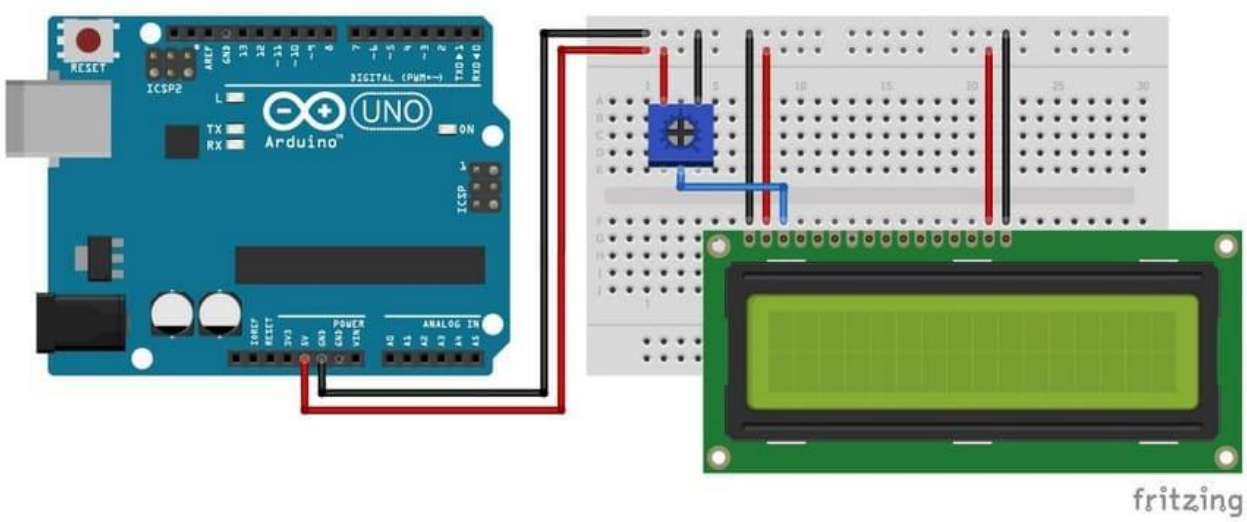


El pinout de una pantalla LCD HD44780 estándar se muestra en la siguiente tabla:

Pin no	**Símbolo**	**Conexión**	**Función**
1	VSS	GND Arduino	Masa de señal
2	VDD	5 V Arduino	Alimentación lógica para LCD
3	V0	10 kΩ potenciómetro	Ajuste de contraste
4	RS	Pin 2 Arduino	Señal de selección de registro
5	R/W	GND Arduino	Señal de selección de lectura/escritura
6	E	Pin 3 Arduino	Señal de habilitación de funcionamiento
7 – 14	D0 – D7	–	Líneas de bus de datos utilizadas para el modo de 8 bits
11 – 14	D4 – D7	Pin 4 – 7 Arduino	Líneas de bus de datos utilizadas para el modo de 4 bits
15	A (LED+)	5 V Arduino	Ánodo para la retroiluminación del LCD
16	K (LED-)	GND Arduino	Cátodo para la retroiluminación del LCD

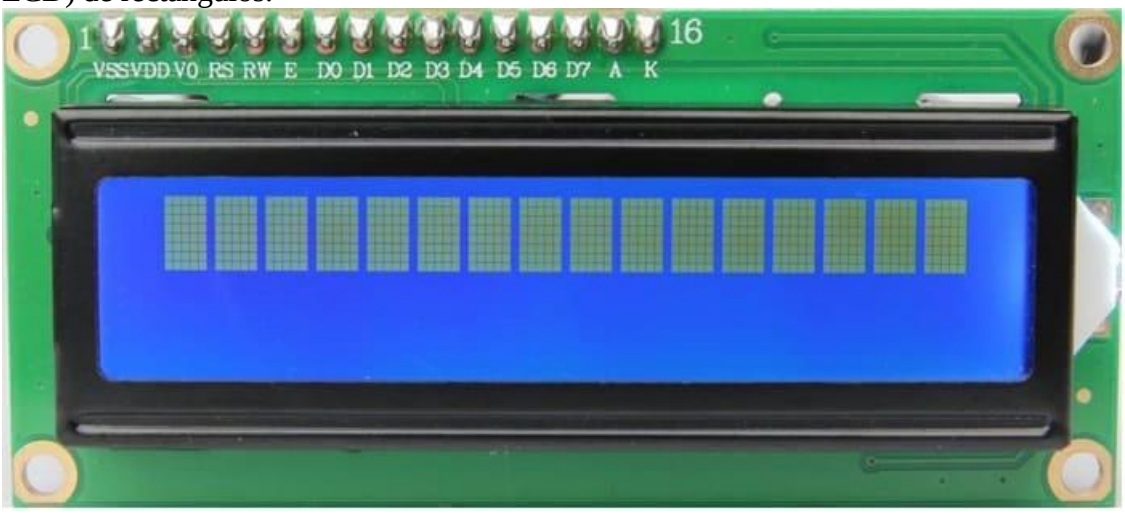
Para seguir este tutorial necesita los siguientes componentes:
Componentes de hardware:
Software:
Prueba de la pantalla LCD y ajuste del contraste

Para probar la pantalla, deberás realizar las conexiones como se muestra en la siguiente figura.



La mayoría de los LCD tienen una resistencia de serie incorporada para la retro iluminación LED. La encontrarás en la parte posterior de la pantalla LCD conectada a la clavija 15 (ánodo). Si tu pantalla no incluye una resistencia, tendrás que añadir una entre 5 V y el pin 15. Debería ser seguro usar una resistencia de 220Ω, pero este valor puede hacer que tu pantalla se oscurezca un poco. Puedes comprobar en la hoja de datos la máxima intensidad nominal de la retro iluminación y utilizarla para seleccionar un valor de resistencia adecuado.

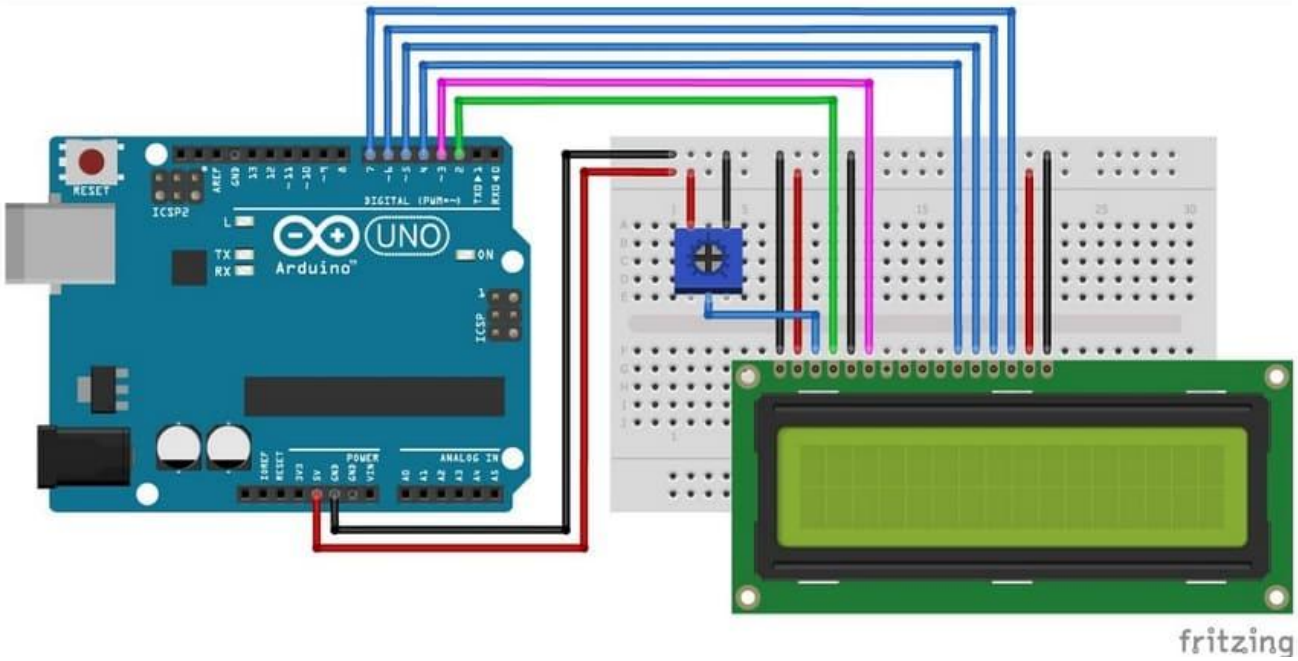
Una vez que hayas cableado la pantalla LCD, tendrás que ajustar el contraste de la pantalla. Esto se hace girando el potenciómetro 10 kΩ en el sentido de las agujas del reloj o en sentido contrario. Enchufa el conector USB del Arduino para alimentar la pantalla LCD. Deberías ver que la luz de fondo se ilumina. Ahora gira el potenciómetro hasta que aparezca una (16×2 LCD) o dos filas (20×4 LCD) de rectángulos.



Puedes ajustar el contraste más tarde si es necesario.

Cómo conectar el LCD a Arduino UNO

Para poder controlar la pantalla LCD y los caracteres de la pantalla, tendrás que añadir algunas conexiones adicionales. Comprueba el diagrama de cableado que aparece a continuación y la tabla de pines de la introducción de este artículo.



Usaremos la pantalla LCD en modo de 4 bits, lo que significa que no es necesario conectar nada a D0-D3. La clavija R/W está conectada a tierra, esto tirará de la clavija LOW y ajustará la pantalla LCD al modo WRITE.

Una vez que hayas cableado todo, podemos empezar a programar la pantalla LCD.

Código Arduino de ejemplo para LCD

Para controlar la pantalla LCD utilizaremos la biblioteca LiquidCrystal. Esta librería debería venir preinstalada con el IDE de Arduino. Puedes encontrarla en Sketch > Incluir biblioteca > LiquidCrystal. La biblioteca LiquidCrystal viene con muchas funciones incorporadas y hace que el control de las pantallas LCD de caracteres sea súper fácil.

El siguiente código de ejemplo muestra cómo mostrar un mensaje en la pantalla LCD. A continuación, te mostraremos cómo funciona el código y cómo puedes usar las otras funciones de la biblioteca LiquidCrystal.

```
/* Basic Arduino example code for displaying text on 16x2, 20x4 etc. character LCDs. */
```

```
// Include the library:
#include

// Create an LCD object. Parameters: (RS, E, D4, D5, D6, D7):
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  // Specify the LCD's number of columns and rows. Change to (20, 4) for a 20x4 LCD:
  lcd.begin(16, 2);
}

void loop() {
  // Set the cursor on the third column and the first row, counting starts at 0:
  lcd.setCursor(2, 0);
  // Print the string 'Hello World!':
  lcd.print("Hello World!");
  // Set the cursor on the third column and the second row:
  lcd.setCursor(2, 1);
  // Print the string 'LCD tutorial':
  lcd.print("LCD tutorial");
}
```

Deberías ver el siguiente texto de salida en la pantalla LCD:
Cómo funciona el código

Después de incluir la librería, el siguiente paso es crear una nueva instancia de la clase LiquidCrystal. Esto se hace con la función LiquidCrystal(rs, enable, d4, d5, d6, d7). Como parámetros utilizamos los pines de Arduino a los que conectamos la pantalla. Ten en cuenta que hemos llamado a la pantalla'lcd'. Puedes darle un nombre diferente si lo desea como 'menu_display'. Tendrás que cambiar 'lcd' por el nuevo nombre en el resto del sketch.

```
// Include the library:
#include

// Create an LCD object. Parameters: (RS, E, D4, D5, D6, D7):
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);
```

En *setup()* la pantalla LCD se inicia con la función *begin(cols,rows)*. Cuando utilices una pantalla LCD de 20×4 cambie esta línea a *lcd.begin(20,4);*

```
void setup() {
  // Specify the LCD's number of columns and rows. Change to (20, 4) for a 20x4 LCD:
  lcd.begin(16, 2);
}
```

En el ***loop()*** el cursor se coloca en la tercera columna y primera fila de la pantalla LCD con ***lcd.setCursor(2,0)*.** Ten en cuenta que el conteo comienza en 0, y el primer argumento especifica la columna. Si no se especifica la posición del cursor, el texto se imprimirá en la posición de inicio predeterminada (0,0) si la pantalla está vacía, o detrás del último carácter impreso.

A continuación, la cadena “Hello World” se imprime con ***lcd.print (“Hello World!”)*.** Ten en cuenta que debe colocar comillas (“”) alrededor del texto. Cuando desees imprimir números o variables, no se necesitan comillas.

```
void loop() {
  // Set the cursor on the third column and the first row, counting starts at 0:
  lcd.setCursor(2, 0);
  // Print the string 'Hello World!':
  lcd.print("Hello World!");
  // Set the cursor on the third column and the second row:
  lcd.setCursor(2, 1);
  // Print the string 'LCD tutorial':
  lcd.print("LCD tutorial");
}
```

Puedes ver cómo usar un sensor ultrasónico HC-SR04 con una pantalla LCD en este tutorial.

Otras funciones de la biblioteca LiquidCrystal

La librería LiquidCrystal Arduino tiene muchas otras funciones incorporadas que te pueden resultar útiles. Puedes encontrar una descripción general de ellos abajo con explicaciones y algunos fragmentos de código.

clear()

Borra la pantalla LCD y posiciona el cursor en la esquina superior izquierda (primera fila y primera columna) de la pantalla. Puedes utilizar esta función para visualizar diferentes palabras en un ciclo.

```
#include
```

```
// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  lcd.clear();
  lcd.print("Monday");
  delay(2000);
  lcd.clear();
  lcd.print("13:45");
  delay(2000);
}
```

inicio()

Posiciona el cursor en la esquina superior izquierda de la pantalla LCD. Utilice clear() si también desea borrar la pantalla.

cursor()

Muestra el cursor de la pantalla LCD: un guión bajo (línea) en la posición del siguiente carácter que se va a imprimir.

noCursor()

Oculta el cursor del LCD. El siguiente ejemplo crea un cursor parpadeante al final de **"cursor()".**

#include

```
// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("cursor()");
}

void loop() {
  lcd.cursor();
  delay(500);
  lcd.noCursor();
  delay(500);
}
```

blink()

Crea un cursor LCD estilo bloque parpadeante: un rectángulo parpadeante en la posición del siguiente carácter a imprimir.

noBlink()

Desactiva el cursor LCD de estilo de bloque. El siguiente ejemplo muestra el cursor parpadeante durante 5 segundos y luego lo desactiva durante 2 segundos.
#include

```
// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("blink() example");
}

void loop() {
  lcd.blink();
  delay(5000);
  lcd.noBlink();
  delay(2000);
}
```

display()

Esta función enciende la pantalla LCD y muestra cualquier texto o cursor que se haya impreso en la pantalla.

noDisplay()

Esta función desactiva cualquier texto o cursor impreso en la pantalla LCD. El texto/los datos no se borran de la memoria LCD. Esto significa que se volverá a mostrar cuando se llame a la función display().

El siguiente ejemplo crea un efecto de texto intermitente.
write()

Esta función se puede utilizar para escribir un carácter en la pantalla LCD. Consulte la sección sobre la creación y visualización de caracteres personalizados más abajo para obtener más información.
scrollDisplayLeft()

Desplaza el contenido de la pantalla (texto y cursor) un espacio a la izquierda. Puedes utilizar esta función en la sección de bucle del código en combinación con delay(500), para crear una animación de texto desplazable.
#include

```
// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
  lcd.print("scrollDisplayLeft() example");
}

void loop() {
  lcd.scrollDisplayLeft();
  delay(500);
}
```

scrollDisplayRight()

Desplaza el contenido de la pantalla (texto y cursor) un espacio a la derecha.

autoscroll()

Esta función activa el desplazamiento automático de la pantalla LCD. Esto hace que la salida de cada carácter a la pantalla empuje los caracteres anteriores más de un espacio. Si la dirección actual del texto es de izquierda a derecha (la predeterminada), la pantalla se desplaza hacia la izquierda; si la dirección actual es de derecha a izquierda, la pantalla se desplaza hacia la derecha. Esto tiene el efecto de enviar cada nuevo carácter a la misma ubicación en la pantalla LCD.

El siguiente esquema de ejemplo permite el desplazamiento automático e imprime el carácter 0 a 9 en la posición (16,0) de la pantalla LCD. Cámbielo a (20,0) para una pantalla LCD de 20×4.

#include

```
// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  lcd.autoscroll();
  lcd.setCursor(16, 0);
  for (int x = 0; x < 10; x++) {
    lcd.print(x);
    delay(500);
  }
  lcd.clear();
}
```

noAutoscroll()

Desactiva el desplazamiento automático de la pantalla LCD.

leftToRight()

Esta función hace que el texto fluya hacia la derecha desde el cursor, como si la pantalla estuviera justificada a la izquierda (por defecto).

rightToLeft()

Esta función hace que el texto fluya hacia la izquierda desde el cursor, como si la pantalla estuviera justificada a la derecha.

¿Cómo crear y mostrar caracteres personalizados?

Con la función ***createChar()*** es posible crear y mostrar caracteres personalizados en la **pantalla LCD.** Esto es especialmente útil si desea visualizar un carácter que no forma parte del juego de caracteres **ASCII** estándar.

Información técnica. Las pantallas **LCD** que se basan en el controlador **LCD Hitachi HD44780** tienen dos tipos de memorias: **CGROM y CGRAM (ROM y RAM del Generador de Caracteres).**

CGROM genera todos los patrones de caracteres de 5 x 8 puntos a partir de los códigos de caracteres estándar de 8 bits. CGRAM puede generar patrones de caracteres definidos por el usuario.

Para pantallas de 5 x 8 puntos, CGRAM puede escribir hasta 8 caracteres personalizados y para pantallas de 5 x 10 puntos 4.

Caracteres personalizados Código de ejemplo de Arduino

El siguiente esquema de ejemplo crea y muestra ocho caracteres personalizados (numerados del 0 al 7).

```
/* Example sketch to create and display custom characters on character LCD with Arduino and LiquidCrystal library. */

#include

// Creates an LCD object. Parameters: (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd = LiquidCrystal(2, 3, 4, 5, 6, 7);

// Make custom characters:
byte Heart[] = {
  B00000,
  B01010,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000,
  B00000
};
byte Bell[] = {
  B00100,
```

```
  B01110,
  B01110,
  B01110,
  B11111,
  B00000,
  B00100,
  B00000
};
byte Alien[] = {
  B11111,
  B10101,
  B11111,
  B11111,
  B01110,
  B01010,
  B11011,
  B00000
};
byte Check[] = {
  B00000,
  B00001,
  B00011,
  B10110,
  B11100,
  B01000,
  B00000,
  B00000
};
byte Speaker[] = {
  B00001,
  B00011,
  B01111,
  B01111,
  B01111,
  B00011,
  B00001,
  B00000
};
byte Sound[] = {
  B00001,
  B00011,
  B00101,
  B01001,
  B01001,
  B01011,
  B11011,
  B11000
};
```

```
byte Skull[] = {
  B00000,
  B01110,
  B10101,
  B11011,
  B01110,
  B01110,
  B00000,
  B00000
};
byte Lock[] = {
  B01110,
  B10001,
  B10001,
  B11111,
  B11011,
  B11011,
  B11111,
  B00000
};

void setup() {
  // Specify the LCD's number of columns and rows:
  lcd.begin(16, 2);

  // Create a new characters:
  lcd.createChar(0, Heart);
  lcd.createChar(1, Bell);
  lcd.createChar(2, Alien);
  lcd.createChar(3, Check);
  lcd.createChar(4, Speaker);
  lcd.createChar(5, Sound);
  lcd.createChar(6, Skull);
  lcd.createChar(7, Lock);

  // Clears the LCD screen:
  lcd.clear();

  // Print a message to the lcd:
  lcd.print("Custom Character");
}

void loop() {
  // Print all the custom characters:
  lcd.setCursor(0, 1);
  lcd.write(byte(0));
  lcd.setCursor(2, 1);
  lcd.write(byte(1));
```

```
  lcd.setCursor(4, 1);
  lcd.write(byte(2));
  lcd.setCursor(6, 1);
  lcd.write(byte(3));
  lcd.setCursor(8, 1);
  lcd.write(byte(4));
  lcd.setCursor(10, 1);
  lcd.write(byte(5));
  lcd.setCursor(12, 1);
  lcd.write(byte(6));
  lcd.setCursor(14, 1);
  lcd.write(byte(7));
}
```

Deberías ver la siguiente salida en la pantalla LCD:

Cómo funciona el código

Después de incluir la biblioteca y crear el objeto LCD, se definen las matrices de caracteres personalizadas. Cada matriz consta de 8 bytes, 1 byte por cada fila. En este ejemplo se crean 8 caracteres personalizados.

```
// Make custom characters:
byte Heart[] = {
  B00000,
  B01010,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000,
  B00000
};
```

Al observar de cerca la matriz, verás lo siguiente. Cada fila consta de 5 números que corresponden a los 5 píxeles en un carácter de 5 x 8 puntos. Un 0 significa píxel apagado y un 1 significa píxel encendido.

Es posible editar cada fila a mano, pero recomiendo usar esta herramienta visual en GitHub. Esta aplicación crea automáticamente la matriz de caracteres y puede hacer clic en los píxeles para activarlos o desactivarlos.

En *setup()*, los caracteres personalizados se crean con *lcd.createChar(num, data)*.

El primer argumento de esta función es el número del carácter personalizado (0-7) y el segundo es la matriz de caracteres que hemos creado.

```
// Create a new characters:
lcd.createChar(0, Heart);
lcd.createChar(1, Bell);
lcd.createChar(2, Alien);
lcd.createChar(3, Check);
lcd.createChar(4, Speaker);
lcd.createChar(5, Sound);
lcd.createChar(6, Skull);
lcd.createChar(7, Lock);
```

En el loop() todos los caracteres se muestran con lcd.write(). Como parámetro utilizamos el número del carácter que reservamos.

```
lcd.setCursor(0, 1);
lcd.write(byte(0));
```

Conclusión

En este artículo te hemos enseñado cómo usar una pantalla LCD alfanumérica con Arduino.

Esperamos que te haya resultado útil e informativo. Si lo hiciste, por favor compártelo con un amigo que también le guste la electrónica y hacer cosas con su placa Arduino.

Última actualización el 2019-10-27 / Enlaces de afiliados / Imágenes de la API para Afiliados
https://www.electrogeekshop.com/como-usar-un-lcd-de-16x2-caracteres-con-arduino/?srsltid=AfmBOop8pN7M3L_gz9wOJWbkwQzjIc_-HYzvFZ9lvLgs0YgEc77kgIDx

BIBLIOGRAGIA

- https://www.youtube.com/watch?v=cqJ5fXaShP0
- https://www.youtube.com/watch?v=6ODn1XrysGw
- https://www.youtube.com/watch?v=mH_h1LS9wMo
- https://www.youtube.com/watch?v=Wx1Vi0EPhQU
- https://naylampmechatronics.com/blog/34_tutorial-lcd-conectando-tu-arduino-a-un-lcd1602-y-lcd2004.html
- Tutorial Arduino: Pantalla LCD
https://www.electrogeekshop.com/como-usar-un-lcd-de-16x2-caracteres-con-arduino/?srsltid=AfmBOop8pN7M3L_gz9wOJWbkwQzjIc_-HYzvFZ9lvLgs0YgEc77kgIDx
- Circuito Prototipo Sistema de Riego.
https://wokwi.com/projects/398613169728760833

.