SISTEMAS DE PROCESAMIENTO DE DATOS

SISTEMA DE RIEGO

Integrantes:

Bruno Rubini,

Mateo Regis,

Luca Paoloni,

Dante Toledo,

Jenson Medina

Comision 8, SPD, 2023

Informe del Proyecto de Sistema de Riego en Tinkercad

Introducción:

El presente informe detalla el desarrollo de un sistema de riego automatizado creado en Tinkercad. El proyecto se realizó en colaboración con un grupo de estudiantes con el objetivo de aprender sobre electrónica, programación y automatización de procesos.

Descripción del Proyecto

El sistema de riego automatizado utiliza sensores de humedad y temperatura para controlar el riego de una planta en función de las condiciones ambientales. También se incorpora un sensor de luz para ajustar la iluminación y un mecanismo de ventilación en caso de que la temperatura supere un umbral específico.

Objetivos

Los objetivos principales de este proyecto son:

Diseñar un sistema de riego que mantenga un nivel óptimo de humedad en un jardín.

Controlar la temperatura del entorno y activar la ventilación cuando sea necesario para evitar temperaturas extremas.

Ajustar la iluminación en función de la luz ambiente.

Aprender sobre programación en Arduino y la utilización de sensores para la automatización de tareas.

Metodología

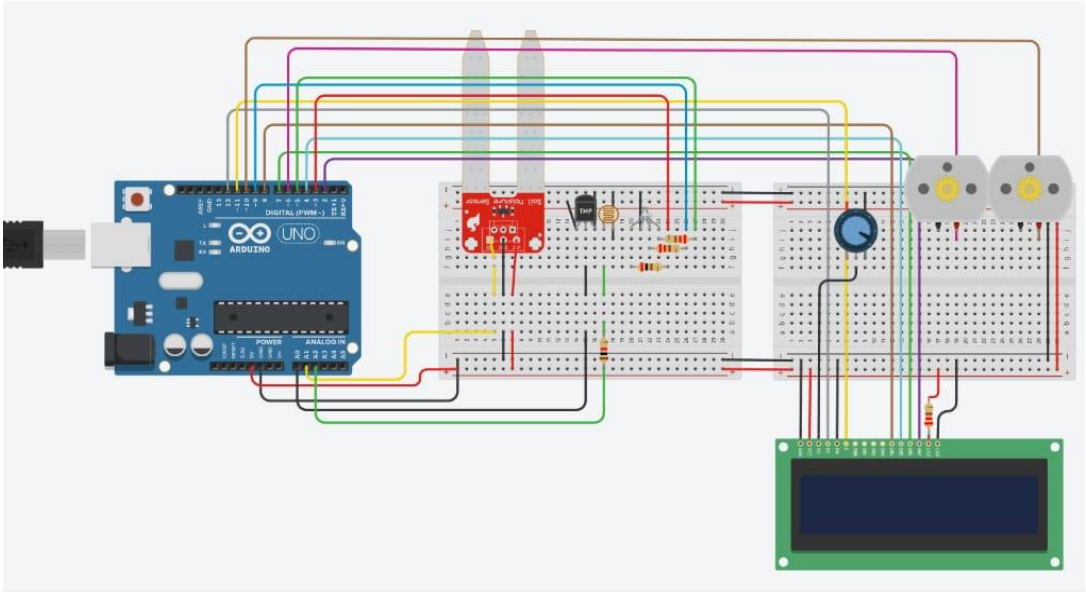
El proyecto se desarrolló siguiendo los siguientes pasos:

Diseño del circuito: Se utilizó Tinkercad para simular el circuito que incluye sensores de humedad, temperatura y luz, así como actuadores para el riego y la ventilación.

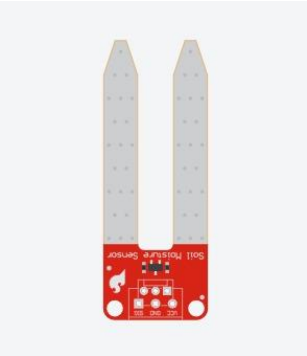
Programación: Se escribió el código en Arduino para adquirir datos de los sensores y controlar los actuadores en función de las lecturas.

Depuración y prueba: Se realizaron pruebas en el entorno de simulación para garantizar el correcto funcionamiento del sistema.

Optimización: Se realizaron mejoras en el código para una ejecución más eficiente y se agregaron comentarios explicativos para facilitar la comprensión.



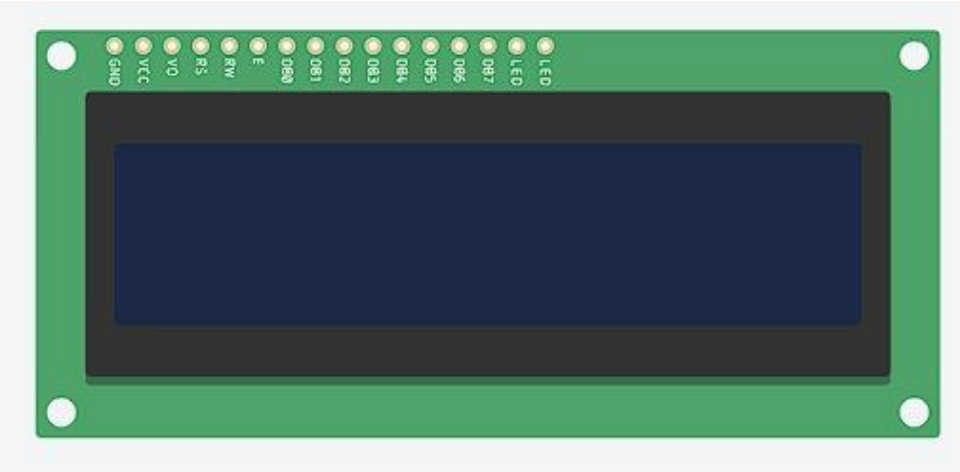
Sensor de humedad: Mide la temperatura ambiente y generalmente devuelve valores en grados Celsius o Fahrenheit.



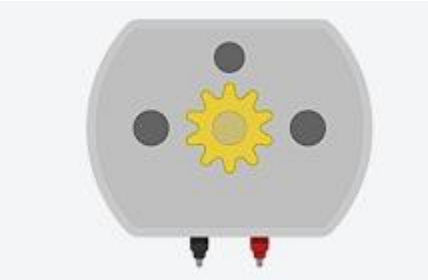
Fotorresistencia: Su resistencia varía en función de la intensidad de la luz a la que está expuesta. Puede usarse para detectar la luminosidad en el entorno.

Sensor de Temperatura: Mide la temperatura ambiente y generalmente devuelve valores en grados Celsius o Fahrenheit.

Pantalla LCD (Liquid Crystal Display): Una pantalla para mostrar información. Puede mostrar texto o números, y se controla a través de Arduino.



Motores CC: Son motores de corriente continua que se utilizan para realizar acciones físicas, como el riego en tu proyecto.



Potenciómetro: Es un dispositivo de ajuste que permite variar la resistencia eléctrica. Se usa comúnmente para ajustar valores como el brillo de una pantalla o la velocidad de un motor

Luz RGB: Es un tipo de LED (diodo emisor de luz) especial que puede emitir luz de varios colores. Un LED RGB está compuesto por tres LED individuales dentro de un solo componente: uno rojo, uno verde y uno azul. Cada uno de estos LEDs puede ser controlado de forma independiente para variar la intensidad de luz que emiten, lo que permite la mezcla de colores y la creación de una amplia gama de tonos.

Resistencias: Las resistencias son dispositivos electrónicos pasivos que se utilizan para limitar el flujo de corriente eléctrica en un circuito. En términos sencillos, las resistencias obstaculizan el paso de la corriente eléctrica y ayudan a controlar la cantidad de corriente que fluye a través de un componente o una parte específica de un circuito.

Comentarios en el Código

A continuación, se presentan comentarios explicativos para el código del proyecto:

```
#include <LiquidCrystal.h>
```

```
LiquidCrystal lcd(12, 11, 8, 4, 7, 2); // rs, enabled, d4, d5, d6, d7

int pinSensorHumedad = A1;
int pinSensorTemperatura = A0;
int pinSensorLuz = A2;
int pinRiego = 10;
int pinVentilacion = 6;
int pinRojo = 3;
int pinVerde = 5;
int pinAzul = 9;

void setup() {
  lcd.begin(16, 2);
  pinMode(pinVentilacion, OUTPUT);
  pinMode(pinRiego, OUTPUT);
  pinMode(pinRojo, OUTPUT);
  pinMode(pinVerde, OUTPUT);
  pinMode(pinAzul, OUTPUT);
}

void loop() {
  lcd.scrollDisplayLeft();
  float humedad = analogRead(pinSensorHumedad);
```

```
  int porcentajeHumedad = map(humedad, 0, 876, 0, 100);


  lcd.setCursor(0,0);
  lcd.print("H: ");
  lcd.setCursor(4,0);
  lcd.print(porcentajeHumedad);
  lcd.print("% ");
  if(porcentajeHumedad<35){
    analogWrite(pinRiego, 255);// Como utilizamos un pin PWM , podemos setear la velocidad mandandole
    //un valor entre 0 y 255; (0 es el minimo, 255 el maximo);
    lcd.setCursor(11,0);
   lcd.print(" Muy seco ");
  } else if (porcentajeHumedad >= 35 && porcentajeHumedad < 60) {
    analogWrite(pinRiego, 150); //Bajamos la velocidad, ya que hay un mayor porcentaje de humedad
    lcd.setCursor(11, 0);
    lcd.print(" Suelo seco ");
  } else {
    digitalWrite(pinRiego, LOW);// Como el suelo esta muy humedo, apagamos el motor de riego
    lcd.setCursor(11, 0);
    lcd.print(" Humedo      ");
  }
```

```
  float valorLeidoTMP = analogRead(pinSensorTemperatura);
  float temperatura = (((5 * valorLeidoTMP * 100) / 1024) - 50);
  lcd.setCursor(0, 1);
  lcd.print("T: ");
  lcd.print(temperatura);
  lcd.print("C ");
  if (temperatura > 30) {
    analogWrite(pinVentilacion, 255);// Usamos la maxima velocidad del motor de ventilacion
    lcd.print(" Mucho Calor ");
  }
  else if(temperatura >= 15 && temperatura <= 30)
  {
    lcd.print("  Buen Clima ");
    analogWrite(pinVentilacion, 90);// Bajamos la velocidad del motor de ventilacion, ya que tenemos una temperatura media
  }
  else {
    analogWrite(pinVentilacion, 0);
    lcd.setCursor(11, 1);
    lcd.print(" Mucho Frio ");
  }
```

```
  delay(100);


  int foto = analogRead(pinSensorLuz);
  delay(100);
  if (foto > 500) {
    analogWrite(pinRojo, 200);
    analogWrite(pinAzul, 100);
    analogWrite(pinVerde, 0);
  } else {
        analogWrite(pinRojo, 0);
    analogWrite(pinAzul, 0);
    analogWrite(pinVerde, 0);
  }
}
```

Problemas a la hora del desarrollo:

Durante todo el proceso de desarrollo nos enfrentamos a ciertos problemas, como por ejemplo el funcionamiento de la pantalla LCD.

Hubo un momento en donde la pantalla imprimía unos caracteres “especiales” lo cual hacía que el mensaje que nosotros queríamos mostrar se viera distorsionado.

Este problema surgía porque nosotros estábamos haciendo lcd.println(); y esto así no existe en la librería liquid crystal, lo correcto es usar lcd.print();

Otro problema que tuvimos con la pantalla fue que al aplicar un lcd.clear(); la pantalla quedaba titilando, por lo que el mensaje que intentamos mostrar no se podía leer.

Este error se debía a que el lcd.clear() lo usábamos en conjunto con lcd.scrollDisplayLeft();

Para solucionar este problema optamos por no usar el lcd.clear(); alineamos las palabras y dejamos espacios en blanco al imprimir algunas palabras para que tengan la misma cantidad de caracteres y los mensajes se pudieran ver bien.

Conclusiones

El proyecto de sistema de riego automatizado ha sido una experiencia educativa valiosa que nos ha permitido aprender sobre Arduino, programación y automatización. Se han logrado los objetivos establecidos y se ha desarrollado un sistema funcional.

Este proyecto demuestra la utilidad de la automatización en varios ámbitos, como en este caso la agricultura y destaca la importancia de la programación y el uso de sensores en la toma de decisiones basadas en datos ambientales.