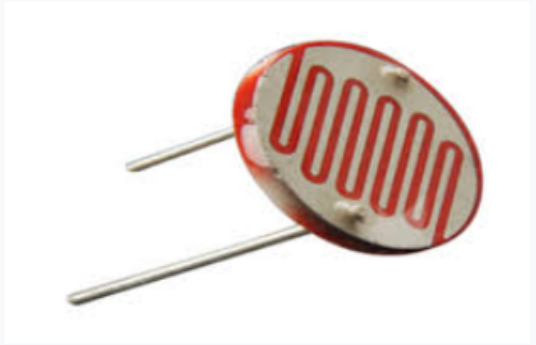
LDR y REGULACIÓN DEL NIVEL DE LUMINOSIDAD DE UN LED

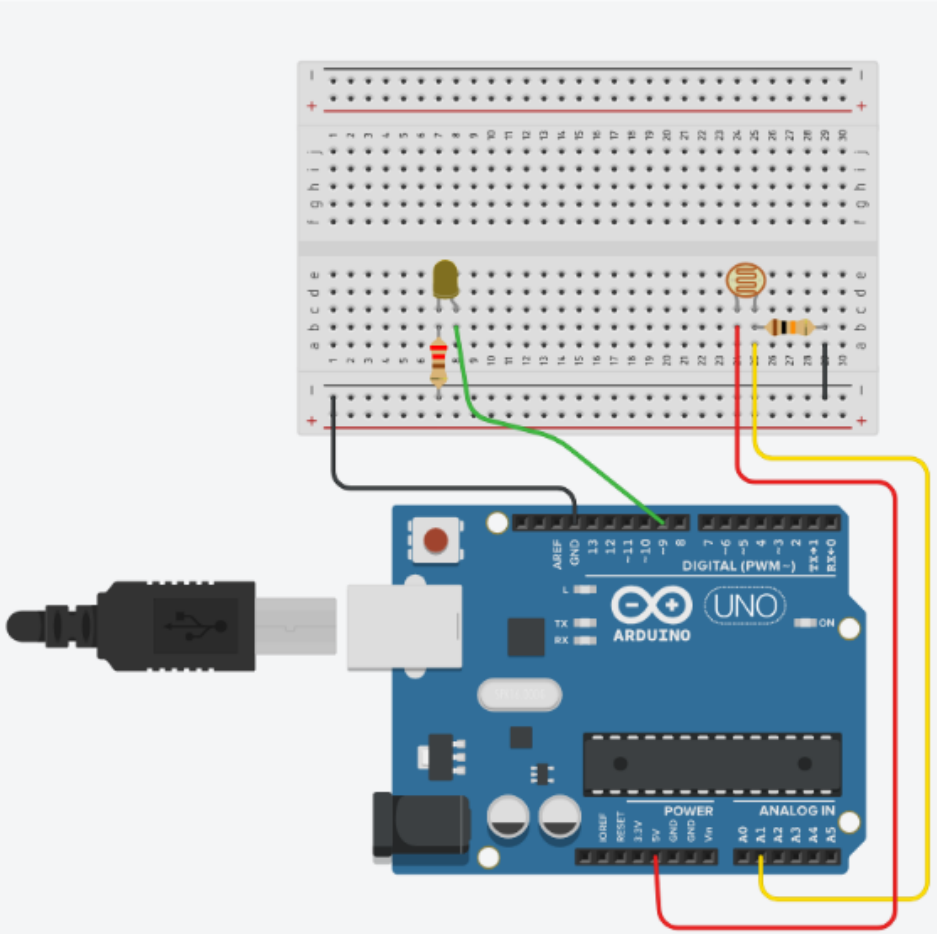
Para **regular la luminosidad de un LED** podemos optar por una solución que requiere intervención manual, un **potenciómetro**. O bien podemos realizar la regulación detectando el nivel de luz ambiente (con una **LDR)** y controlando la corriente eléctrica que atraviesa el LED en función de ese nivel de luz. Para ello vamos a usar un pin **PWM** al que conectaremos el LED.

¿Cómo se comporta la LDR?: la resistencia disminuirá en presencia de luz y aumentará en ausencia de ella.



Las LDR se conectan a un **pin analógico** y proporcionan una **salida entre 0-1024**.

CONEXIONADO



LDR y REGULACIÓN DEL NIVEL DE LUMINOSIDAD DE UN LED

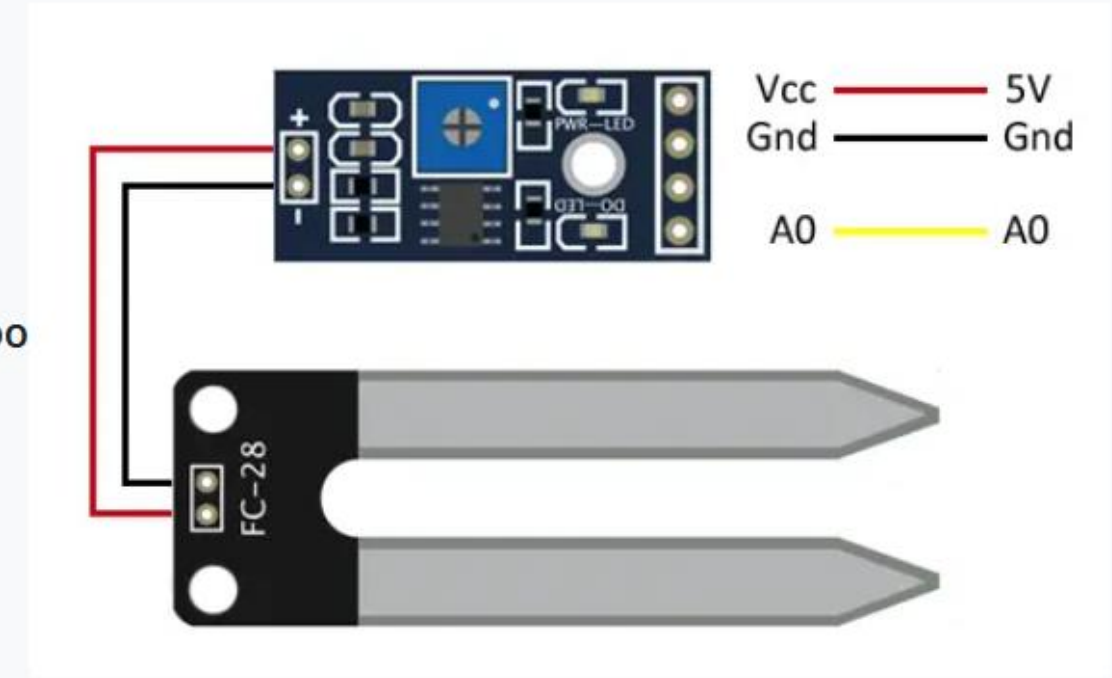
```
int ldr = A0;  // Pin de la LDR
int led = 9;   // Pin del LED (PWM)
int valorLuz;  // Valor leído del sensor
int brillo;    //vamos a mapear el nivel de luz detectado 0-1023 a nivel de potencia 0-255

void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);  // Para ver valores de luz en el monitor serie
}

void loop() {
  valorLuz = analogRead(ldr);  // Lee la luz (0-1023)
  Serial.println(valorLuz);    // Muestra valor en puerto serie

  // Convertimos el valor de nivel de luz detecgtado (0-1024) a un rango de brillo (0-255)
  brillo = map(valorLuz, 0, 1023, 255, 0);
  // Cuanta más luz haya, menos brillo tendrá la farola

  analogWrite(led, brillo);  // Ajusta intensidad del LED
  delay(100);                // Pequeña pausa
}
```

SENSOR DE HUMEDAD DE SUELO

Estos sensores son útiles en aplicaciones de agricultura, jardinería, sistemas de riego automatizados para garantizar que las plantas reciban la cantidad adecuada de agua.

* Los sensores resistivos de humedad de suelo devuelven valores INVERTIDOS:
* - Valor ALTO (cerca de 1023) → suelo SECO
* - Valor BAJO (cerca de 0) → suelo MUY HÚMEDO

CONEXIONADO



SENSOR DE HUMEDAD DE SUELO

```ino
const int PIN_SENSOR = A0;
const int PIN_LED_VERDE = 7;
const int PIN_LED_ROJO = 6;

// --- Umbrales (ajusta estos valores según tu sensor) ---
// Suelo seco:     lecturas altas  (~700 - 1023)
// Humedad ideal:  lecturas medias (~300 - 700)
// Suelo encharcado: lecturas bajas (~0  - 300)
const int UMBRAL_SECO = 700;  // Por encima → demasiado seco   → LED rojo
                              //por debajo → LED verde

const int INTERVALO_LECTURA = 2000;  // Milisegundos entre lecturas

unsigned long ultimaLectura = 0;

void setup() {
  pinMode(PIN_LED_VERDE, OUTPUT);
  pinMode(PIN_LED_ROJO, OUTPUT);

  // Apagar ambos LEDs al inicio
  digitalWrite(PIN_LED_VERDE, LOW);
  digitalWrite(PIN_LED_ROJO, LOW);
}
```

```ino
void loop() {
  unsigned long ahora = millis();

  if (ahora - ultimaLectura >= INTERVALO_LECTURA) {
    ultimaLectura = ahora;

    int lectura = analogRead(PIN_SENSOR);


    // Determinar estado
    if (lectura > UMBRAL_SECO) {
      // Suelo demasiado seco
      digitalWrite(PIN_LED_VERDE, LOW);
      digitalWrite(PIN_LED_ROJO, HIGH);

    } else {
      // Humedad correcta
      digitalWrite(PIN_LED_ROJO, LOW);
      digitalWrite(PIN_LED_VERDE, HIGH);
    }
  }
}
```

El programa **humedad_v1.ino** funciona así: un sensor de humedad de suelo conectado al pin A0 y dos ledes, uno rojo (pin 6) y uno verde (pin 7). Cuando el grado de humedad es correcto, se enciende el led verde, en caso contrario, el rojo.

SENSOR DE HUMEDAD DE SUELO

TAREAS PROPUESTAS

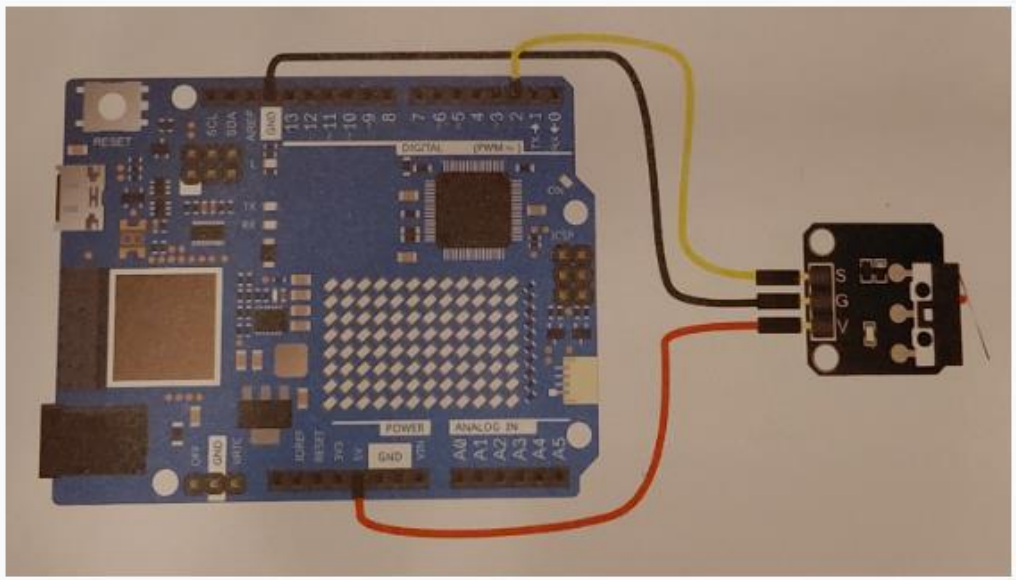
1. Modifica el programa anterior para que los **valores de humedad leídos se muestren en el puerto serie.**
2. Modifica el programa anterior para que en lugar de encender el rojo fijo, que parpadee cuando el suelo está muy seco. usa **millis()** en lugar de **delay() Ver nota al final (*) ESTO ES PARA PERSONAS CON CONOCIMIENTOS ALTOS DE ARDUINO**
3. Añadir un **tercer LED amarillo** para el estado intermedio "casi seco, riega pronto". Trabaja con tres umbrales en lugar de dos.
4. Añadir un **buzzer** en un pin digital que suene cuando el suelo lleva **más de X lecturas consecutivas en estado seco**.

FINAL DE CARRERA

Un final de carrera se utiliza para detectar la posición o fin de mocimiento de un objeto en sistemas automáticos. Funciona como un interruptor que se activa cuando se presiona su palanca.

No necesita resistencia externa. La placa activa internamente una resistencia que mantiene el pin en HIGH cuando el final de carrera no está pulsado. Al pulsarlo, el pin se conecta a GND y lee LOW. Por eso la lógica parece invertida: **pulsado = LOW, libre = HIGH**

CONEXIONADO



El programa adjunto funciona así:

```
* Conexiones:
*   Final de carrera:
*     - Un terminal  → Pin 2
*     - Otros terminales → GND y 5V
*
*   LED verde → Pin 7 (con resistencia 330Ω) → HIGH si puerta cerrada. Y se mostrará una cara alegre en matriz de led.
*   LED rojo  → Pin 6 (con resistencia 330Ω) → HIGH si puerta abierta. Y se mostrará una cara trite en matriz de led.
*
* Cómo funciona INPUT_PULLUP:
*   - Sin pulsar: el pin lee HIGH (el final de carrera no toca nada)
*   - Pulsado:    el pin lee LOW  (el final de carrera está activado)
```

FINAL DE CARRERA

```
#include "Arduino_LED_Matrix.h"
//Crea una instancia de la matriz
ArduinoLEDMatrix Matriz;

const int PIN_FINAL = 2;
const int PIN_LED_VERDE = 7;
const int PIN_LED_ROJO = 6;

bool estadoAnterior = HIGH;  // Guardamos el estado previo para detectar cambios

void setup() {

  Matriz.begin();  //inicializa la matriz de ledes

  pinMode(PIN_FINAL, INPUT_PULLUP);  // Resistencia pull-up interna activada
  pinMode(PIN_LED_VERDE, OUTPUT);
  pinMode(PIN_LED_ROJO, OUTPUT);

  digitalWrite(PIN_LED_VERDE, LOW);
  digitalWrite(PIN_LED_ROJO, LOW);
}
```

FINAL DE CARRERA

```
void loop() {
  bool estadoActual = digitalRead(PIN_FINAL);

  // Solo actuamos si el estado ha CAMBIADO (evita repetir mensajes)
  if (estadoActual != estadoAnterior) {
    delay(20);                                  // Pequeña pausa para evitar rebotes
    estadoActual = digitalRead(PIN_FINAL);  // Leemos de nuevo tras el rebote
    estadoAnterior = estadoActual;

    if (estadoActual == LOW) {
      // Final de carrera PULSADO → puerta cerrada
      digitalWrite(PIN_LED_ROJO, LOW);
      digitalWrite(PIN_LED_VERDE, HIGH);
      Matriz.loadFrame(LEDMATRIX_EMOJI_HAPPY);  //CARA SONRIENTE SI PUERTA CERRADA
    } else {
      // Final de carrera LIBRE → puerta abierta
      digitalWrite(PIN_LED_VERDE, LOW);
      digitalWrite(PIN_LED_ROJO, HIGH);
      Matriz.loadFrame(LEDMATRIX_EMOJI_SAD); //CARA TRISTE AL ABRIR LA PUERTA
    }
  }
```

TAREA PROPUESTA

Modifica el programa anterior para añadir una LCD que muestre "puerta abierta" o "puerta cerrada" según corresponda.