TEMA 3 ENTRADAS Y SALIDAS ANALÓGICAS PWM

Entradas y Salidas Analógicas en Arduino

Arduino ofrece soporte para manejar señales analógicas mediante entradas analógicas (lecturas de sensores) y salidas PWM (modulación por ancho de pulso) que simulan señales analógicas.

1. Entradas Analógicas

Las entradas analógicas en Arduino permiten leer señales variables como las provenientes de sensores (temperatura, luz, etc.). El valor leído se convierte en un número digital utilizando un ADC (Convertidor Analógico-Digital).

1.1 Funcionamiento

- Arduino convierte la señal analógica (0-5V) en un valor digital de 10 bits.
- El valor obtenido varía entre 0 y 1023:
    - 0 corresponde a 0V.
    - 1023 corresponde a 5V.

1.2 Función analogRead(pin)

- Lee el valor analógico de un pin específico.
- pin: Número del pin analógico (A0, A1, ..., A5 en Arduino UNO).

EJEMPLO

```
int sensorValue;
void setup() {
  Serial.begin(9600); // Iniciar comunicación serial
}
void loop() {
  sensorValue = analogRead(A0); // Leer el pin A0
  Serial.println(sensorValue);  // Mostrar el valor en el monitor serie
  delay(500);
}
```

2. Salidas Analógicas (PWM)

Arduino no genera señales analógicas reales, pero puede simularlas mediante PWM (Pulse Width Modulation). En este método, se enciende y apaga rápidamente un pin de salida para variar el promedio de voltaje percibido.

2.1 Funcionamiento

- El valor de PWM oscila entre 0 (0%) y 255 (100%).
- PWM utiliza pines digitales marcados con el símbolo ~ (tilde).
- La frecuencia de la señal PWM en Arduino UNO es de aproximadamente 490 Hz.

2.2 Función analogWrite(pin, value)

- Escribe un valor de PWM en un pin.
- pin: Número del pin PWM.
- value: Nivel de PWM (0-255).

Ejemplo: Controlar el brillo de un LED.

EJEMPLO

```
void setup() {
  pinMode(9, OUTPUT); // Configurar el pin 9 como salida
}


void loop() {
  for (int i = 0; i <= 255; i++) {
    analogWrite(9, i); // Incrementar el brillo del LED
    delay(10);
  }
  for (int i = 255; i >= 0; i--) {
    analogWrite(9, i); // Reducir el brillo del LED
    delay(10);
  }
}
```

Programa Completo: Control de un Motor con Potenciómetro

Este programa utiliza un potenciómetro para controlar la velocidad de un motor a través de una señal PWM.

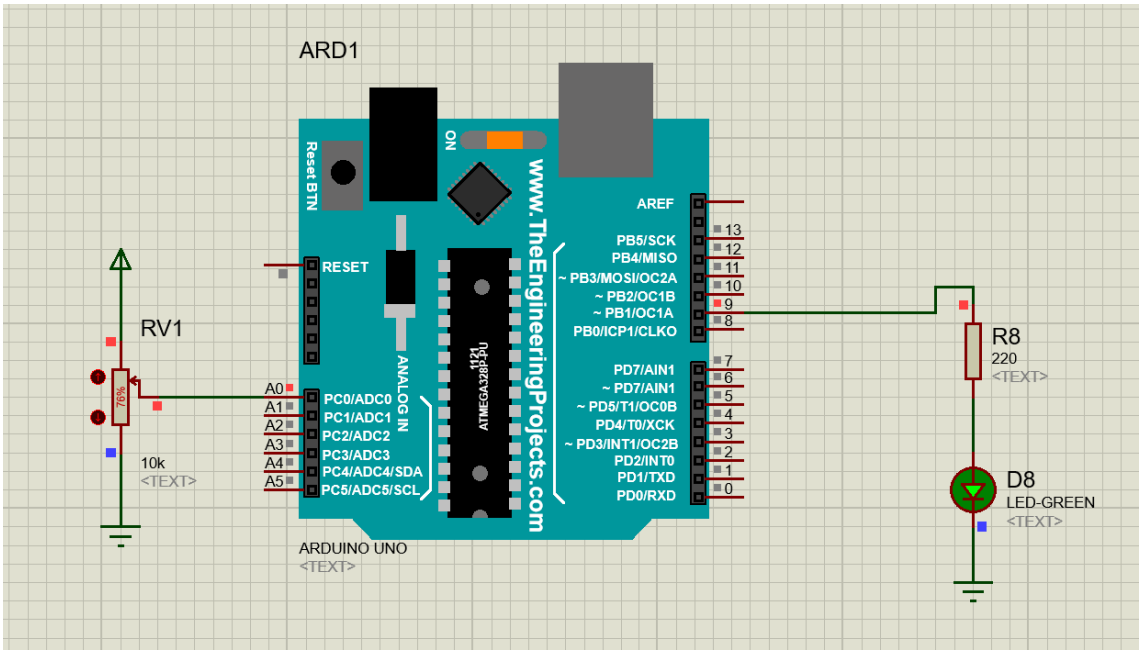
Circuito

1. Conecta el potenciómetro al pin A0.
2. Conecta el motor (con transistor y diodo de protección) al pin PWM (por ejemplo, el pin 9).

Código

```
const int potPin = A0; // Pin del potenciómetro

const int motorPin = 9; // Pin del motor (PWM)

int potValue = 0;      // Valor leído del potenciómetro

int motorSpeed = 0;    // Valor de velocidad del motor

void setup() {

  pinMode(motorPin, OUTPUT); // Configurar el pin del motor como salida

  Serial.begin(9600);        // Iniciar comunicación serial

}

void loop() {

  potValue = analogRead(potPin);             // Leer el valor del potenciómetro

  motorSpeed = map(potValue, 0, 1023, 0, 255); // Convertir rango 0-1023 a 0-
255

  analogWrite(motorPin, motorSpeed);         // Aplicar PWM al motor

  Serial.print("Potenciometro: ");

  Serial.print(potValue);

  Serial.print(" -> Velocidad motor: ");

  Serial.println(motorSpeed);

  delay(100);

}
```

3.1. Leer un potenciómetro y variar el brillo de un LED.



Materiales necesarios

1. Arduino Uno (o cualquier otra placa Arduino).
2. LED.
3. Resistencia de 220 ohmios.
4. Potenciómetro (de 10 kΩ o similar).
5. Protoboard (opcional).
6. Cables de conexión.

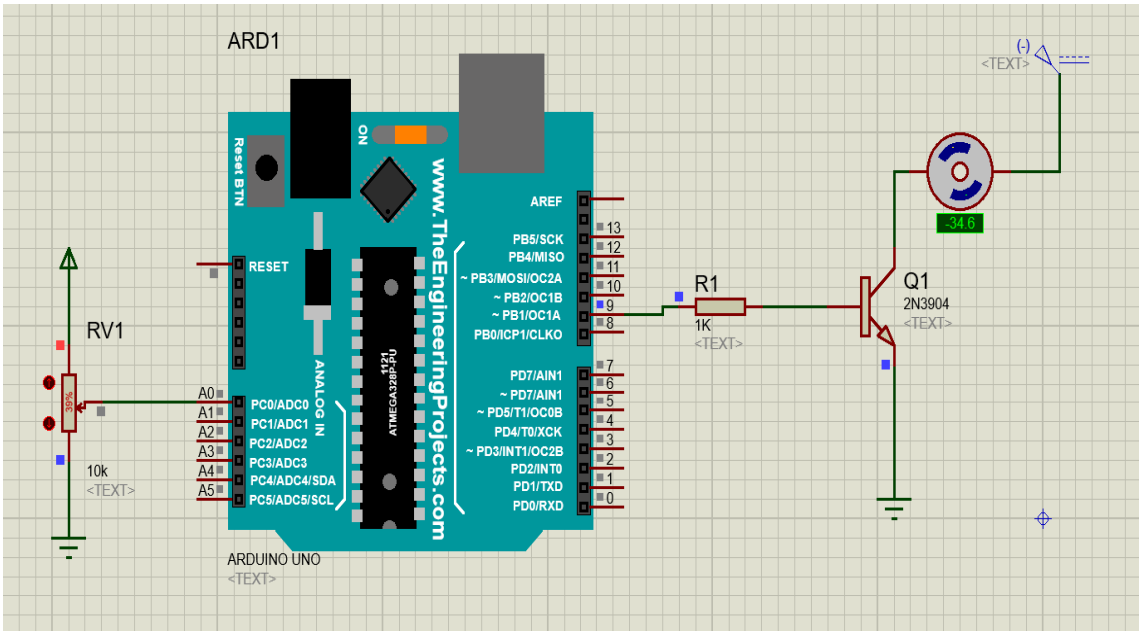
Esquema de conexión

1. Conecta el ánodo del LED (pata larga) al pin 9 de Arduino.
2. Conecta el cátodo del LED (pata corta) a una resistencia de 220 ohmios, y luego a GND.
3. Conecta:
    - Una pata lateral del potenciómetro a 5V.
    - La otra pata lateral a GND.
    - La pata central (cursor) al pin A0 de Arduino.

PROGRAMA:

```
const int potPin = A0; // Pin del potenciómetro (entrada analógica)
const int ledPin = 9;  // Pin del LED (salida PWM)
void setup() {
  pinMode(ledPin, OUTPUT); // Configurar el pin del LED como salida
}
void loop() {
  // Leer el valor del potenciómetro (0 a 1023)
  int potValue = analogRead(potPin);
  // Convertir el valor del potenciómetro (0-1023) a rango PWM (0-255)
  int ledBrightness = map(potValue, 0, 1023, 0, 255);

  // Ajustar el brillo del LED
  analogWrite(ledPin, ledBrightness);

  // Opcional: Añadir un pequeño retraso para estabilizar
  delay(10);
}
```

3.2. CONTROL MOTOR DC MEDIANTE PWM



Componentes necesarios:

1. Arduino Uno o similar.
2. Motor DC.
3. Transistor (como el NPN 2N2222 o MOSFET IRF520).
4. Diodo (1N4007 o similar).
5. Resistencia de 220 Ω (para la base del transistor).
6. Fuente de alimentación externa (si el motor requiere más potencia).
7. Protoboard y cables.

Conexiones:

1. **Motor DC:**
   - Uno de los terminales del motor se conecta al colector del transistor o al drenador del MOSFET.
   - El otro terminal del motor se conecta al positivo de la fuente de alimentación externa.

2. **Transistor/MOSFET:**
    - Colector (o drenador): Conectado a un terminal del motor.
    - Emisor (o fuente): Conectado a GND.
    - Base (o compuerta): Conectada a un pin PWM de Arduino (por ejemplo, pin 9) a través de una resistencia de 220 Ω.
3. **Diodo de rueda libre:**
    - Conecta el cátodo al positivo del motor y el ánodo al colector (o drenador) del transistor.
4. **Arduino:**
    - Pin PWM → Resistencia de 220 Ω → Base/compuerta del transistor/MOSFET.
    - GND → GND de la fuente externa.

PROGRAMA

```
 // Pines
const int motorPin = 9;     // Pin PWM para el motor
const int potPin = A0;      // Pin analógico para el potenciómetro

void setup() {
  pinMode(motorPin, OUTPUT);
}

void loop() {
  // Leer el valor del potenciómetro (0 a 1023)
  int potValue = analogRead(potPin);

  // Convertir el valor del potenciómetro a rango PWM (0 a 255)
  int pwmValue = map(potValue, 0, 1023, 0, 255);
```

```
  // Aplicar PWM al motor
  analogWrite(motorPin, pwmValue);

  delay(10); // Pequeño retardo para suavizar la señal
}
```