# Periféricos y Dispositivos de Interfaz Humana


UNIVERSIDAD DE GRANADA

**CURSO 2024 - 2025**

# Práctica 3. Experimentación con Arduino

**GABRIEL VICO ARBOLEDAS**
**RAÚL RODRÍGUEZ RODRÍGUEZ**

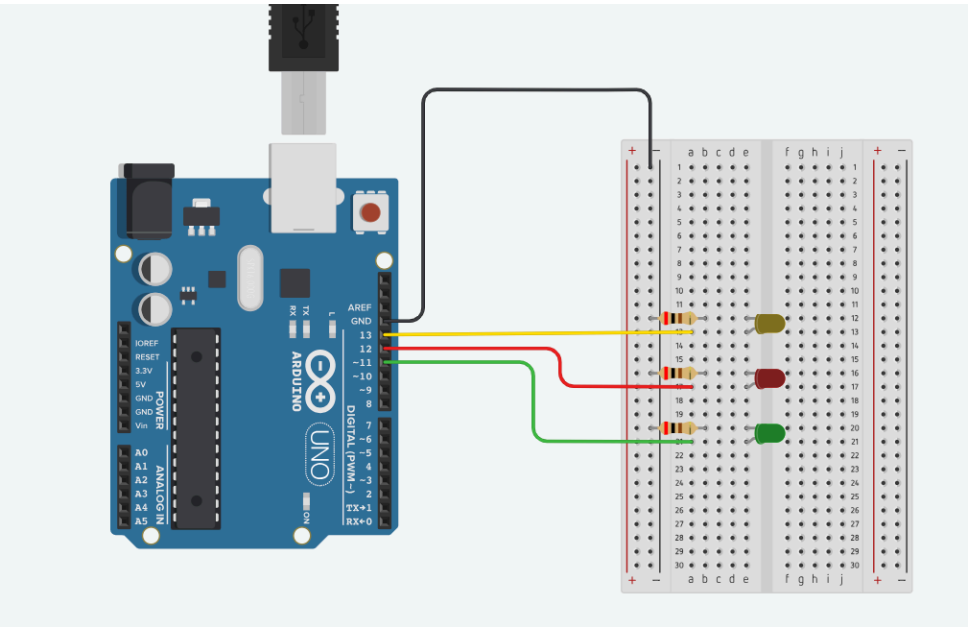
# Índice

# 1. Ejercicios obligatorios

## 1.1 Implementar el programa de parpadeo de LED, para que encienda y apague alternativamente tres LED

- Componentes eléctricos:
  - Arduino Uno
  - Placa de pruebas
  - 4 cables
  - 3 resistencias (220 ohmios)
  - 3 leds (amarillo, rojo, verde)

- Esquema de conexiones eléctricas:



- Código:

```
void setup()
{
    // Configura el pin 13 como salida
    pinMode(13, OUTPUT);

    // Configura el pin 12 como salida
    pinMode(12, OUTPUT);

    // Configura el pin 11 como salida
    pinMode(11, OUTPUT);
}

void loop()
{
    // Enciende el LED conectado al pin 13
```

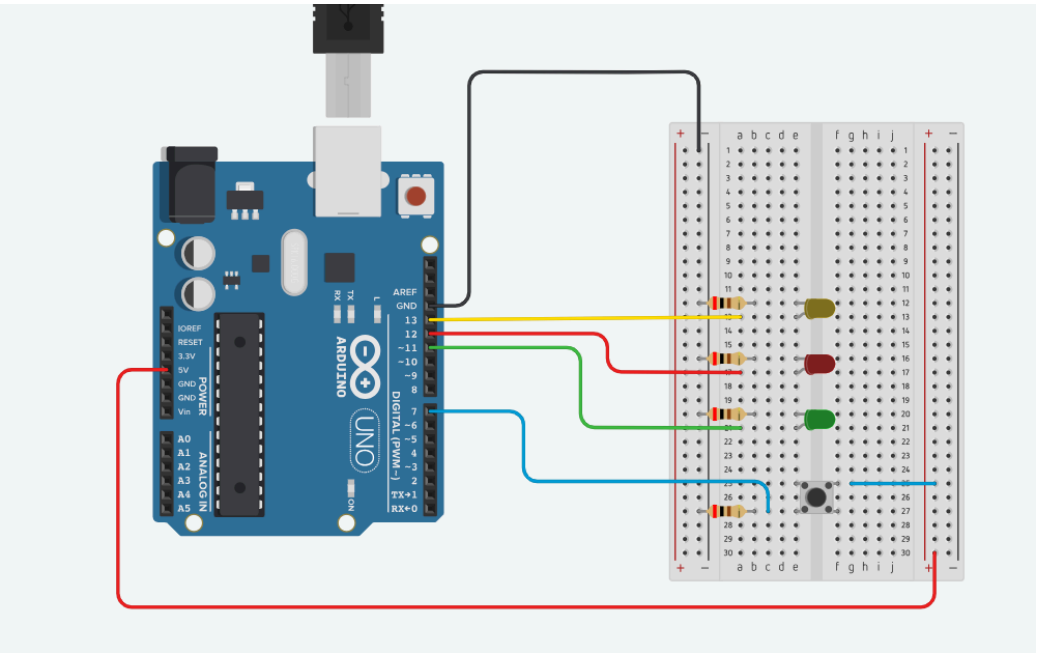
```
    digitalWrite(13, HIGH);

    // Espera 1.5 segundos
    delay(1500);

    // Apaga el LED conectado al pin 13
    digitalWrite(13, LOW);

    // Enciende el LED conectado al pin 12
    digitalWrite(12, HIGH);

    // Espera 1.5 segundos
    delay(1500);

    // Apaga el LED conectado al pin 12
    digitalWrite(12, LOW);

    // Enciende el LED conectado al pin 11
    digitalWrite(11, HIGH);

    // Espera 1.5 segundos
    delay(1500);

    // Apaga el LED conectado al pin 11
    digitalWrite(11, LOW);
}
```

En el repositorio de github se adjunta video del funcionamiento del arduino.

## 1.2 Implementar el programa de parpadeo de LEDs para que se encienda el LED rojo solo cuando se pulse un interruptor conectado a la entrada digital 7, y en ese momento se apaguen los LEDs amarillo y verde

- Componentes eléctricos:
    - Arduino Uno
    - Placa de pruebas
    - 7 cables
    - 4 resistencias (220 ohmios)
    - 3 leds (amarillo, rojo, verde)
    - 1 pulsador

- Esquema de conexiones eléctricas:



- Código:

```
// Variable para almacenar el estado
int val = 0;

void setup()
{
  // Configura el pin 13 como salida
  pinMode(13, OUTPUT);

  // Configura el pin 12 como salida
  pinMode(12, OUTPUT);

  // Configura el pin 11 como salida
  pinMode(11, OUTPUT);

  // Configura el pin 7 como entrada
  pinMode(7, INPUT);
}

void loop()
{
  // Lee el estado del pin digital 7 y lo guarda en la variable 'val'
  val = digitalRead(7);

  // Si el interruptor está presionado (valor LOW)
  if (val == LOW) {
    // Enciende el pin 12
    digitalWrite(12, HIGH);
```

```
    // Apaga el pin 13
    digitalWrite(13, LOW);

    // Apaga el pin 11
    digitalWrite(11, LOW);

    // Espera 1.5 segundos
    delay(1500);

    // Apaga el LED rojo
    digitalWrite(12, LOW);

    // Espera otros 1.5 segundos (el rojo queda apagado por este tiempo)
    delay(1500);
  }
  else {
    // Apaga el LED rojo si el interruptor no está presionado
    digitalWrite(12, LOW);

    // Enciende el pin 13
    digitalWrite(13, HIGH);

    // Espera 1.5 segundos con el LED verde encendido
    delay(1500);

    // Apaga el pin 13
    digitalWrite(13, LOW);

    // Enciende el pin 11
    digitalWrite(11, HIGH);

    // Espera 1.5 segundos con el LED amarillo encendido
    delay(1500);

    // Apaga el pin 11
    digitalWrite(11, LOW);
  }
}
```

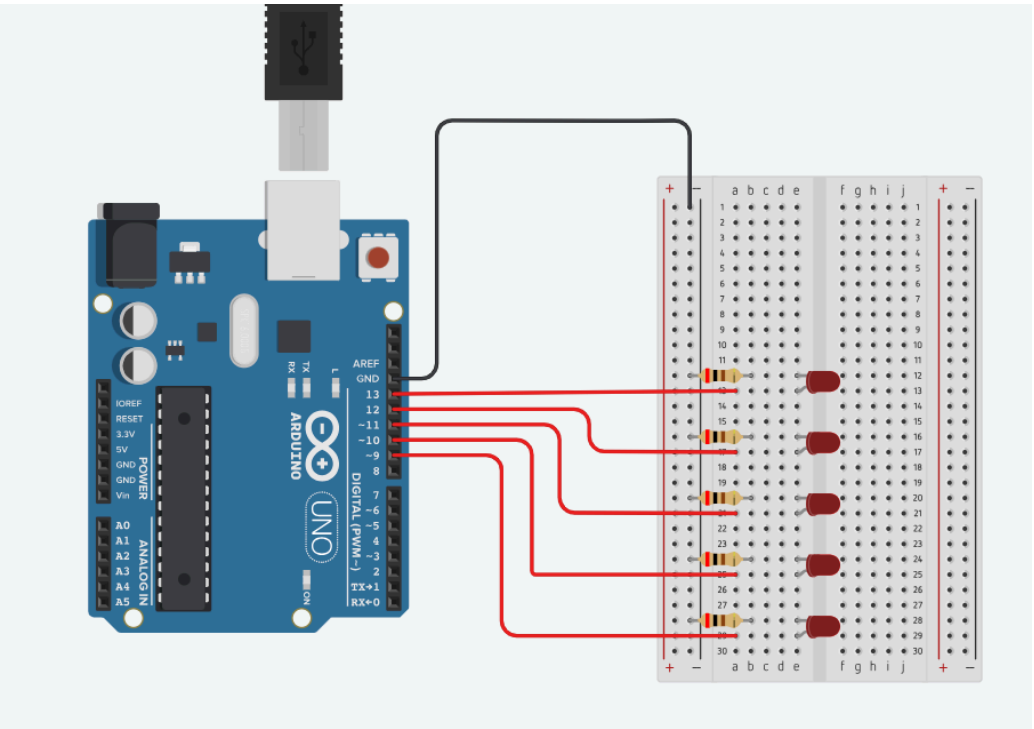
En el repositorio de github se adjunta video del funcionamiento del arduino.

# 2. Ejercicios opcionales

## 2.1 Secuencia de LEDs, encendiendo y apagando 4 LEDs secuencialmente

- Componentes eléctricos:
  - Arduino Uno
  - Placa de pruebas
  - 6 cables
  - 5 resistencias (220 ohmios)
  - 5 leds rojos

- Esquema de conexiones eléctricas:



- Código:

```
void setup()
{
  // Configura los pines 13 a 9 como salidas digitales (para los LEDs)
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
}


void loop()
{
  // Etiqueta para salto con goto: "derecha"
  derecha:

  // Bucle for que va del pin 13 al 9 (de mayor a menor)
  for (int i = 13; i >= 9; --i) {
    // Si llegamos al pin 9, saltamos a la etiqueta "izquierda"
    if(i == 9){
      goto izquierda;
    }

    // Enciende el LED en el pin actual
    digitalWrite(i, HIGH);
    delay(150); // Espera 150 milisegundos

    // Apaga el LED actual
    digitalWrite(i, LOW);
    // Asegura que el siguiente LED (en orden descendente) esté apagado también
    digitalWrite(i-1, LOW);
    delay(150); // Espera 150 milisegundos
  }

  // Etiqueta para salto con goto: "izquierda"
  izquierda:

  // Bucle for que va del pin 9 al 13 (de menor a mayor)
  for (int i = 9; i <= 13; ++i) {
    // Si llegamos al pin 13, saltamos a la etiqueta "derecha"
    if(i == 13){
      goto derecha;
    }

    // Enciende el LED en el pin actual
    digitalWrite(i, HIGH);
    delay(150); // Espera 150 milisegundos
```

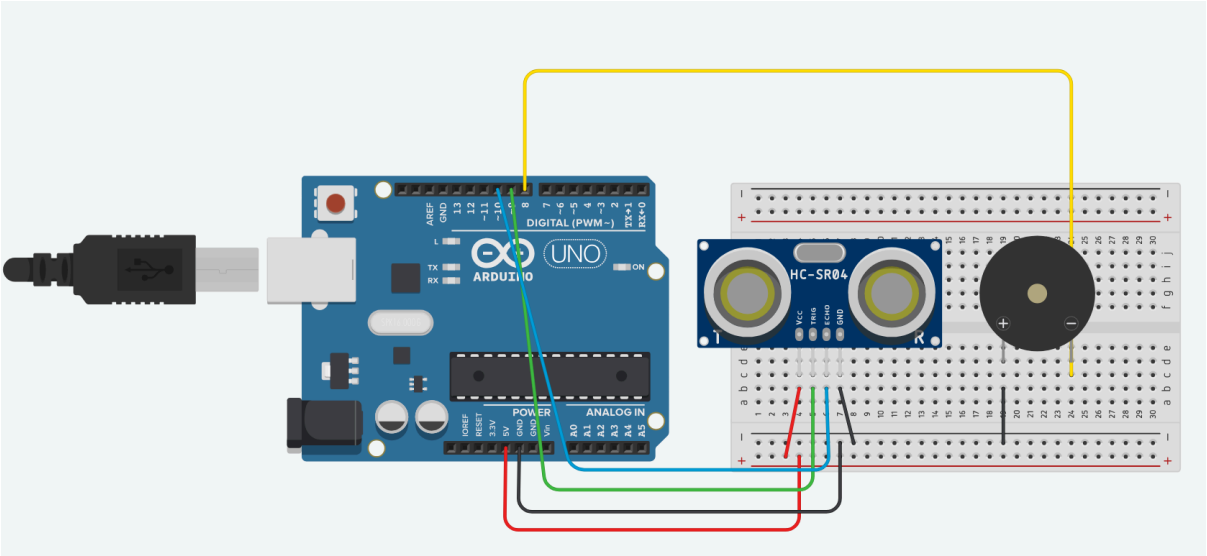
```
    // Apaga el LED actual
    digitalWrite(i, LOW);
    // Asegura que el siguiente LED (en orden ascendente) esté apagado también
    digitalWrite(i+1, LOW);
    delay(150); // Espera 150 milisegundos
  }
}
```

En el repositorio de github se adjunta video del funcionamiento del arduino.

## 2.2 Detector de la distancia a un objeto

- Componentes eléctricos:
    - Arduino Uno
    - Placa de pruebas
    - 8 cables
    - 1 sensor de ultrasonidos
    - 1 buzzer

- Esquema de conexiones eléctricas:



- Código:

```
// Definición de pines
const int trigPin = 9;     // Pin Trigger del sensor
const int echoPin = 10;    // Pin Echo del sensor
const int buzzerPin = 8;   // Pin del buzzer

// Variables
long duration;             // Duración del pulso de eco
int distance;              // Distancia en cm
int frequency;             // Frecuencia del buzzer

void setup() {
  pinMode(trigPin, OUTPUT); // Configura el pin Trigger como salida
  pinMode(echoPin, INPUT);  // Configura el pin Echo como entrada
  pinMode(buzzerPin, OUTPUT); // Configura el pin del buzzer como salida

  Serial.begin(9600);      // Inicia la comunicación serial para
depuración
}

void loop() {
  // Limpia el pin Trigger
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Envía un pulso de 10 microsegundos al pin Trigger
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Lee el pin Echo y devuelve el tiempo de viaje del sonido en
microsegundos
  duration = pulseIn(echoPin, HIGH);

  // Calcula la distancia (la velocidad del sonido es 343 m/s o 0.0343
cm/µs)
  distance = duration * 0.0343 / 2;

  // Mapea la distancia a una frecuencia audible (20-2000 Hz es el rango
típico)
  // Distancias más cercanas = frecuencia más alta
  frequency = map(distance, 2, 200, 2000, 200);

  // Limita la frecuencia a valores razonables
  frequency = constrain(frequency, 200, 2000);
```

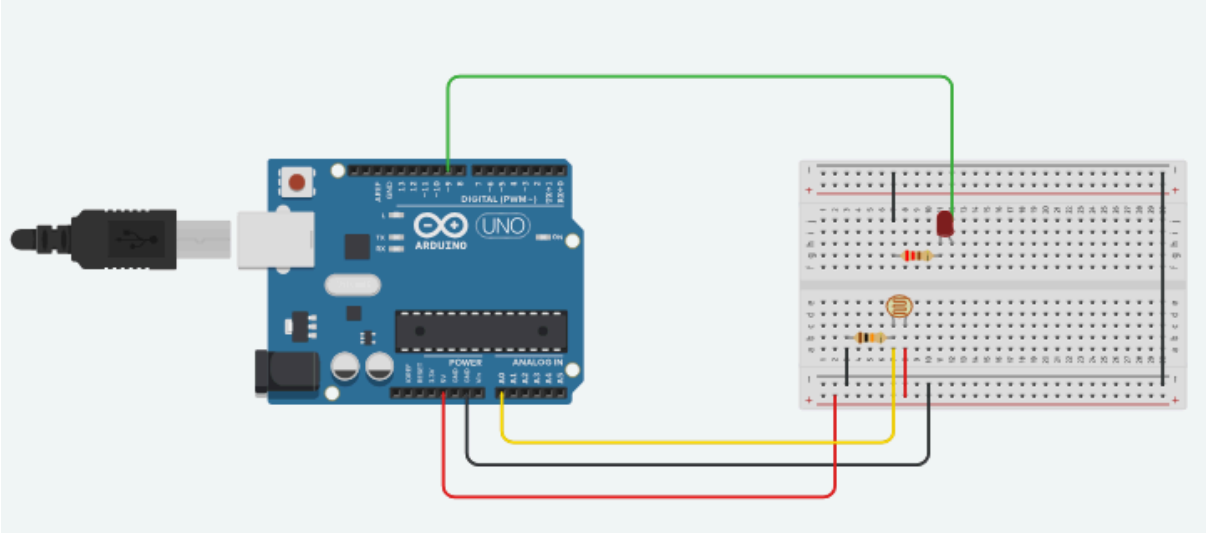
```
  // Si el objeto está dentro del rango (2cm - 200cm), activa el buzzer
  if (distance >= 2 && distance <= 200) {
    tone(buzzerPin, frequency, 100); // Emite un tono de 100ms de
duración
  } else {
    noTone(buzzerPin); // Apaga el buzzer si está fuera de rango
  }

  // Muestra la distancia en el monitor serial (para depuración)
  Serial.print("Distancia: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Pequeña pausa entre mediciones
  delay(100);
}
```

En el repositorio de github se adjunta video del funcionamiento del arduino.

## 2.3 Detector de la cantidad de luz

- Componentes eléctricos:
  - Arduino Uno
  - Placa de pruebas
  - 8 cables
  - 2 resistencias (220 ohmios)
  - 1 leds rojo
  - 1 fotosensor

- Esquema de conexiones eléctricas:



- Código:

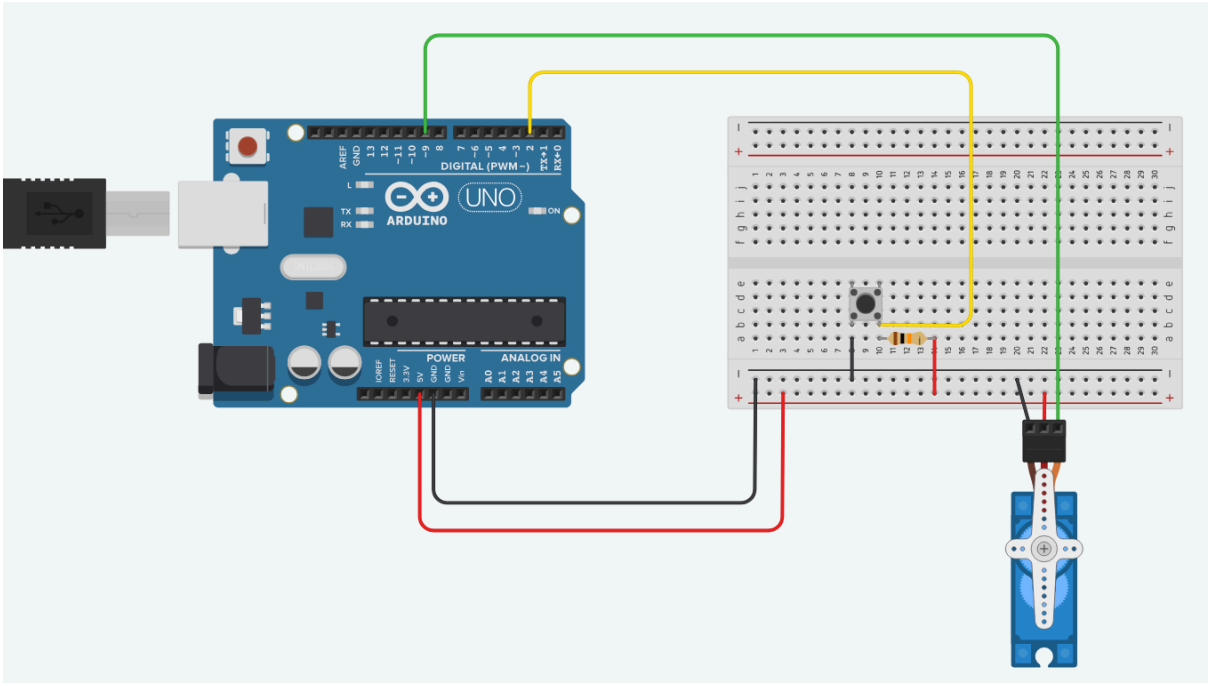
```
const int ldrPin = A2;      // Pin del LDR
const int ledPin = 9;       // Pin del LED (debe ser PWM ~)
int ldrValue = 0;           // Valor leído del LDR
int ledBrightness = 0;      // Brillo del LED (0-255)

void setup() {
  pinMode(ledPin, OUTPUT);  // Configura el pin del LED como salida
  Serial.begin(9600);       // Inicia comunicación serial para
depuración
}

void loop() {
  ldrValue = analogRead(ldrPin);               // Lee el valor del LDR
(0-1023)

  if (ldrValue < 30){ // Umbral de oscuridad
    analogWrite(ledPin, 0); // Se apaga por completo
  } else {
    ledBrightness = map(ldrValue, 30, 800, 255, 0); // Invierte el valor
(más luz = menos resistencia)
    analogWrite(ledPin, ledBrightness);          // Ajusta el brillo del
LED
  }

  delay(100); // Pequeña pausa para estabilidad
}
```

En el repositorio de github se adjunta video del funcionamiento del arduino.

## 2.4 Implementar un proyecto en el que se active un motor

- Componentes eléctricos:
  - Arduino Uno
  - Placa de pruebas
  - 8 cables
  - 1 resistencias (220 ohmios)
  - 1 pulsador
  - 1 Servo motor

- Esquema de conexiones eléctricas:




- Código:

```
#include <Servo.h>

const int buttonPin = 2;  // Pin del pulsador
const int servoPin = 9;   // Pin del servo
Servo myServo;

void setup() {
  pinMode(buttonPin, INPUT); // Usa resistencia pull-up interna
  myServo.attach(servoPin);
  myServo.write(0);  // Posición inicial (0°)
}

void loop() {
  if (digitalRead(buttonPin) == LOW) { // LOW porque usamos PULLUP
    myServo.write(90);  // Gira a 90° al presionar el botón
  } else {
    myServo.write(0);   // Vuelve a 0° al soltar
  }
  delay(500);          // Pequeña pausa
}
```

En el repositorio de github se adjunta video del funcionamiento del arduino.