

Guía docente

Proyectos de inicio con Arduino



Echidna Educación



Índice

Presentación	3
Soluciones de Mejóralo	4
2.1 Hola Mundo	4
2.2 Semáforo.....	6
2.3 Interruptor de luz	9
2.4 Timbre	11
2.5 Brillo LED	13
2.6 Interruptor crepuscular	17
2.7 Piano de frutas	22
2.8 El echidna dice la temperatura.....	28
2.9 Mezclamos colores	32
2.10 Vúmetro	36
2.11 Regulador de luz con joystick	38
2.12 Nivel de burbuja	41
2.13 Control desde el ordenador.....	45



Presentación

Esta guía acompaña a [Proyectos de inicio con Arduino](#), los proyectos para programar la placa **EchidnaBlack2** con **Arduino IDE**. Está pensada para el profesorado y recoge lo que no aparece en la guía del alumnado.

Cada proyecto de la guía termina con el apartado **Mejóralo**: tres propuestas, de menos a más difícil, para que el alumnado amplíe el programa por su cuenta. Allí solo se da una **Pista**; aquí tienes una posible solución de cada propuesta. Cuando la solución solo cambia una parte del programa (por ejemplo, la función `loop()`), se muestra solo esa parte.

Todos los programas, tanto los de **Programamos** como las soluciones completas de **Mejóralo**, están también en archivos `.ino`, listos para abrir en Arduino IDE, en la carpeta [codigo](#) del repositorio.



Soluciones de Mejóralo

2.1 Hola Mundo

Mejora 1. Ritmo rápido

Solo cambia la función `loop()`:

```
void loop() {  
  digitalWrite(ledRojo, HIGH); // enciende el LED  
  delay(200);                  // espera 0,2 segundos  
  digitalWrite(ledRojo, LOW);  // apaga el LED  
  delay(200);                  // espera 0,2 segundos  
}
```

Mejora 2. Sombra de señal

Solo cambia la función `loop()`:

```
void loop() {  
  digitalWrite(ledRojo, HIGH); // enciende el LED  
  delay(2000);                  // encendido 2 segundos  
  digitalWrite(ledRojo, LOW);  // apaga el LED  
  delay(100);                   // apagado 0,1 segundos  
}
```



Mejora 3. Parpadeo alterno

Posible solución:

```
1 // Parpadeo alterno: el LED rojo y el verde se encienden por turnos
2
3 const int ledRojo = 13; // LED rojo conectado al pin 13
4 const int ledVerde = 11; // LED verde conectado al pin 11
5
6 void setup() {
7     pinMode(ledRojo, OUTPUT); // el pin del LED rojo es una salida
8     pinMode(ledVerde, OUTPUT); // el pin del LED verde es una salida
9 }
10
11 void loop() {
12     digitalWrite(ledRojo, HIGH); // enciende el rojo
13     digitalWrite(ledVerde, LOW); // apaga el verde
14     delay(1000); // espera 1 segundo
15     digitalWrite(ledRojo, LOW); // apaga el rojo
16     digitalWrite(ledVerde, HIGH); // enciende el verde
17     delay(1000); // espera 1 segundo
18 }
```



2.2 Semáforo

Mejora 1. Naranja intermitente

Solo cambia la función `loop()`:

```
void loop() {  
    // fase verde: 5 segundos  
    digitalWrite(ledVerde, HIGH);  
    delay(5000);  
    digitalWrite(ledVerde, LOW);  
  
    // fase naranja: 3 parpadeos rápidos  
    for (int i = 0; i < 3; i++) {  
        digitalWrite(ledNaranja, HIGH);  
        delay(300);  
        digitalWrite(ledNaranja, LOW);  
        delay(300);  
    }  
  
    // fase roja: 5 segundos  
    digitalWrite(ledRojo, HIGH);  
    delay(5000);  
    digitalWrite(ledRojo, LOW);  
}
```



Mejora 2. Semáforo de peatones

Posible solución:

```
1 // Semáforo con semáforo de peatones en el LED RGB
2
3 const int ledVerde = 11;    // LED verde conectado al pin 11
4 const int ledNaranja = 12;  // LED naranja conectado al pin 12
5 const int ledRojo = 13;     // LED rojo conectado al pin 13
6 const int peatonRojo = 9;   // rojo del LED RGB conectado al pin 9
7 const int peatonVerde = 5;  // verde del LED RGB conectado al pin 5
8
9 void setup() {
10     pinMode(ledVerde, OUTPUT);
11     pinMode(ledNaranja, OUTPUT);
12     pinMode(ledRojo, OUTPUT);
13     pinMode(peatonRojo, OUTPUT);
14     pinMode(peatonVerde, OUTPUT);
15 }
16
17 void loop() {
18     // coches en verde: peatones en rojo
19     digitalWrite(peatonRojo, HIGH);
20     digitalWrite(ledVerde, HIGH);
21     delay(5000);
22     digitalWrite(ledVerde, LOW);
23
24     // coches en naranja: peatones siguen en rojo
25     digitalWrite(ledNaranja, HIGH);
26     delay(2000);
27     digitalWrite(ledNaranja, LOW);
28
29     // coches en rojo: peatones en verde
30     digitalWrite(peatonRojo, LOW);
31     digitalWrite(peatonVerde, HIGH);
32     digitalWrite(ledRojo, HIGH);
33     delay(5000);
34     digitalWrite(ledRojo, LOW);
35     digitalWrite(peatonVerde, LOW);
36 }
```



Mejora 3. Semáforo sonoro adaptado

Posible solución:

```
1 // Semáforo sonoro: pitidos lentos en verde y silencio en rojo
2
3 const int ledVerde = 11;    // LED verde conectado al pin 11
4 const int ledNaranja = 12;  // LED naranja conectado al pin 12
5 const int ledRojo = 13;     // LED rojo conectado al pin 13
6 const int zumbador = 10;    // zumbador conectado al pin 10
7
8 void setup() {
9     pinMode(ledVerde, OUTPUT);
10    pinMode(ledNaranja, OUTPUT);
11    pinMode(ledRojo, OUTPUT);
12    pinMode(zumbador, OUTPUT);
13 }
14
15 void loop() {
16     // fase verde: 5 pitidos lentos (5 segundos)
17     digitalWrite(ledVerde, HIGH);
18     for (int i = 0; i < 5; i++) {
19         analogWrite(zumbador, 125); // suena
20         delay(500);
21         analogWrite(zumbador, 0);    // calla
22         delay(500);
23     }
24     digitalWrite(ledVerde, LOW);
25
26     // fase naranja: 2 segundos
27     digitalWrite(ledNaranja, HIGH);
28     delay(2000);
29     digitalWrite(ledNaranja, LOW);
30
31     // fase roja: 5 segundos en silencio
32     digitalWrite(ledRojo, HIGH);
33     delay(5000);
34     digitalWrite(ledRojo, LOW);
35 }
```



2.3 Interruptor de luz

Mejora 1. Luz cruzada (biestable)

Además de la constante y el pinMode del LED verde, cambia la función `loop()`:

```
void loop() {
    estadoSR = digitalRead(pulsadorSR);
    estadoSL = digitalRead(pulsadorSL);

    if (estadoSR == HIGH) {
        digitalWrite(ledRojo, HIGH); // SR: rojo encendido
        digitalWrite(ledVerde, LOW);  // verde apagado
    } else {
        if (estadoSL == HIGH) {
            digitalWrite(ledRojo, LOW); // SL: rojo apagado
            digitalWrite(ledVerde, HIGH); // verde encendido
        }
    }
}
```

Mejora 2. Los dos a la vez

Declara la constante del LED naranja (pin 12), configúralo como salida y añade al final de la función `loop()`:

```
// los dos pulsadores a la vez: LED naranja
if (estadoSR == HIGH && estadoSL == HIGH) {
    digitalWrite(ledNaranja, HIGH);
} else {
    digitalWrite(ledNaranja, LOW);
}
```



Mejora 3. Pulsador con memoria (conmutador)

Posible solución:

```
1 // Pulsador con memoria: SL enciende y apaga el LED rojo
2
3 const int pulsadorSL = 3; // pulsador izquierdo conectado al pin 3
4 const int ledRojo = 13;   // LED rojo conectado al pin 13
5
6 int estadoLED = LOW; // recuerda si el LED está apagado o encendido
7
8 void setup() {
9     pinMode(pulsadorSL, INPUT);
10    pinMode(ledRojo, OUTPUT);
11 }
12
13 void loop() {
14     if (digitalRead(pulsadorSL) == HIGH) {
15         // cambia el LED al estado contrario
16         if (estadoLED == LOW) {
17             estadoLED = HIGH;
18         } else {
19             estadoLED = LOW;
20         }
21         digitalWrite(ledRojo, estadoLED);
22
23         // espera a que sueltes el pulsador
24         while (digitalRead(pulsadorSL) == HIGH) {
25             }
26         delay(50); // evita los rebotes del pulsador al soltarlo
27     }
28 }
```

La clave es la variable **estadoLED**, que recuerda si el LED está apagado (**LOW**) o encendido (**HIGH**). Cada vez que pulsamos **SL**, el programa cambia el LED al estado contrario y actualiza la variable. El bucle **while** hace que el programa espere a que soltemos el pulsador; sin él, mientras lo mantenemos pulsado el LED se encendería y apagaría muchas veces seguidas. El `delay(50)` final evita los **rebotes**: al soltar el pulsador, sus contactos pueden abrirse y cerrarse varias veces en pocos milisegundos.



2.4 Timbre

Mejora 1. LED testigo

Además de las constantes y los pinMode de los LED, cambia la función `loop()`:

```
void loop() {
    estadoSL = digitalRead(pulsadorSL);

    if (estadoSL == HIGH) {
        analogWrite(zumbador, 125);    // suena
        digitalWrite(ledVerde, HIGH);  // verde encendido
        digitalWrite(ledRojo, LOW);     // rojo apagado
    } else {
        analogWrite(zumbador, 0);       // silencio
        digitalWrite(ledVerde, LOW);    // verde apagado
        digitalWrite(ledRojo, HIGH);    // rojo encendido
    }
}
```

Mejora 2. Alarma intermitente

Cambia el `if` de la función `loop()`:

```
if (estadoSL == HIGH) {
    analogWrite(zumbador, 125); // pitido
    delay(100);
    analogWrite(zumbador, 0);    // silencio
    delay(100);
} else {
    analogWrite(zumbador, 0);    // SL sin pulsar: silencio
}
```



Mejora 3. Timbre de dos tonos

Posible solución:

```
1 // Timbre de dos tonos: SL suena grave y SR, agudo
2
3 const int pulsadorSR = 2; // pulsador derecho conectado al pin 2
4 const int pulsadorSL = 3; // pulsador izquierdo conectado al pin 3
5 const int zumbador = 10; // zumbador conectado al pin 10
6
7 const int tonoGrave = 440; // frecuencia del tono de SL, en Hz
8 const int tonoAgudo = 880; // frecuencia del tono de SR, en Hz
9
10 int estadoSR = LOW; // guarda si SR está pulsado (HIGH) o no (LOW)
11 int estadoSL = LOW; // guarda si SL está pulsado (HIGH) o no (LOW)
12
13 void setup() {
14     pinMode(pulsadorSR, INPUT);
15     pinMode(pulsadorSL, INPUT);
16     pinMode(zumbador, OUTPUT);
17 }
18
19 void loop() {
20     estadoSR = digitalRead(pulsadorSR);
21     estadoSL = digitalRead(pulsadorSL);
22
23     if (estadoSL == HIGH) {
24         tone(zumbador, tonoGrave); // SL pulsado: tono grave
25     } else {
26         if (estadoSR == HIGH) {
27             tone(zumbador, tonoAgudo); // SR pulsado: tono agudo
28         } else {
29             noTone(zumbador); // ninguno pulsado: silencio
30         }
31     }
32 }
```

La función **tone** genera en el pin una señal que se enciende y se apaga tantas veces por segundo como indique la frecuencia, y esa frecuencia es la que marca el tono del zumbador. Como en el proyecto anterior, usamos un **if** dentro del **else**: si SL está pulsado suena el tono grave; si no, comprobamos SR para hacer sonar el agudo y, si no hay ninguno pulsado, **noTone** apaga el zumbador.



2.5 Brillo LED

Mejora 1. Respiración lenta

Cambia solo la constante:

```
const int espera = 30; // tiempo entre un brillo y el siguiente, en milisegundos
```

Mejora 2. Latido

Cambia la constante espera por estas dos:

```
const int esperaSubida = 2; // subida rápida: 2 milisegundos por paso
const int esperaBajada = 8; // bajada lenta: 8 milisegundos por paso
```

y cambia la función `loop()`:

```
void loop() {
    // el latido: sube deprisa...
    for (int brillo = 0; brillo <= 255; brillo = brillo + 1) {
        analogWrite(ledVerde, brillo);
        delay(esperaSubida);
    }

    // ...y baja despacio
    for (int brillo = 255; brillo >= 0; brillo = brillo - 1) {
        analogWrite(ledVerde, brillo);
        delay(esperaBajada);
    }

    delay(500); // pausa entre latidos
}
```



Mejora 3. Regulador de brillo

Posible solución:



```
1 // Regulador de brillo: SR sube el brillo del LED verde y SL lo baja
2
3 const int pulsadorSR = 2; // pulsador derecho conectado al pin 2
4 const int pulsadorSL = 3; // pulsador izquierdo conectado al pin 3
5 const int ledVerde = 11; // LED verde conectado al pin 11
6
7 const int espera = 10; // velocidad a la que cambia el brillo, en milisegundos
8
9 int estadoSR = LOW; // guarda si SR está pulsado (HIGH) o no (LOW)
10 int estadoSL = LOW; // guarda si SL está pulsado (HIGH) o no (LOW)
11 int brillo = 0; // guarda el brillo del LED (de 0 a 255)
12
13 void setup() {
14     pinMode(pulsadorSR, INPUT);
15     pinMode(pulsadorSL, INPUT);
16     pinMode(ledVerde, OUTPUT);
17 }
18
19 void loop() {
20     estadoSR = digitalRead(pulsadorSR);
21     estadoSL = digitalRead(pulsadorSL);
22
23     // SR pulsado: sube el brillo sin pasar de 255
24     if (estadoSR == HIGH) {
25         if (brillo < 255) {
26             brillo = brillo + 1;
27         }
28     }
29
30     // SL pulsado: baja el brillo sin bajar de 0
31     if (estadoSL == HIGH) {
32         if (brillo > 0) {
33             brillo = brillo - 1;
34         }
35     }
36
37     analogWrite(ledVerde, brillo); // enciende el LED con el brillo actual
38
39     delay(espera);
40 }
```



La variable `brillo` es **global** (se declara fuera de las funciones) para que **recuerde** su valor de una vuelta de `loop ( )` a la siguiente. Si sueltas los dos pulsadores, no cambia y el LED se queda con el último brillo.



2.6 Interruptor crepuscular

Mejora 1. Lee el sensor y calibra tu aula

Este programa solo lee el sensor y muestra su valor en el Monitor serie una vez por segundo:

```
1 // Lee el sensor de luz: muestra su valor en el Monitor serie
2
3 const int sensorLuz = A3; // sensor de luz (LDR) conectado al pin A3
4
5 int valorLuz = 0; // guarda la luz que mide el sensor (de 0 a 1023)
6
7 void setup() {
8   Serial.begin(9600); // abre la comunicación a 9600 baudios
9 }
10
11 void loop() {
12   valorLuz = analogRead(sensorLuz); // lee el sensor
13
14   Serial.print("Valor del sensor de luz: "); // texto
15   Serial.println(valorLuz);                // valor y salto de línea
16
17   delay(1000); // espera 1 segundo para poder leer los valores
18 }
```

Cuando sepas los valores, vuelve a cargar el programa del interruptor y cambia el número de la constante `umbral`. Este programa sirve para leer cualquier sensor analógico de la placa: solo hay que cambiar el pin.



Mejora 2. Luz blanca

Posible solución:

```
1 // Linterna automática: el LED RGB se enciende en blanco cuando hay poca luz
2
3 const int sensorLuz = A3; // sensor de luz (LDR) conectado al pin A3
4 const int rgbRojo = 9;    // LED RGB: color rojo en el pin 9
5 const int rgbVerde = 5;   // LED RGB: color verde en el pin 5
6 const int rgbAzul = 6;    // LED RGB: color azul en el pin 6
7
8 const int umbral = 200; // por debajo de este valor consideramos que es de noche
9
10 int valorLuz = 0; // guarda la luz que mide el sensor (de 0 a 1023)
11
12 void setup() {
13     pinMode(rgbRojo, OUTPUT); // los tres colores del LED RGB son salidas
14     pinMode(rgbVerde, OUTPUT);
15     pinMode(rgbAzul, OUTPUT);
16 }
17
18 void loop() {
19     valorLuz = analogRead(sensorLuz);
20
21     if (valorLuz < umbral) {
22         // poca luz: los tres colores encendidos dan luz blanca
23         digitalWrite(rgbRojo, HIGH);
24         digitalWrite(rgbVerde, HIGH);
25         digitalWrite(rgbAzul, HIGH);
26     } else {
27         // mucha luz: LED RGB apagado
28         digitalWrite(rgbRojo, LOW);
29         digitalWrite(rgbVerde, LOW);
30         digitalWrite(rgbAzul, LOW);
31     }
32 }
```



Mejora 3. Medidor de luz

Posible solución:



```
1 // Medidor de luz: cuanta más luz, más LED encendidos
2
3 const int sensorLuz = A3; // sensor de luz (LDR) conectado al pin A3
4 const int ledVerde = 11; // LED verde conectado al pin 11
5 const int ledNaranja = 12; // LED naranja conectado al pin 12
6 const int ledRojo = 13; // LED rojo conectado al pin 13
7
8 int valorLuz = 0; // guarda la luz que mide el sensor (de 0 a 1023)
9
10 void setup() {
11     pinMode(ledVerde, OUTPUT);
12     pinMode(ledNaranja, OUTPUT);
13     pinMode(ledRojo, OUTPUT);
14 }
15
16 void loop() {
17     valorLuz = analogRead(sensorLuz);
18
19     if (valorLuz < 200) {
20         // muy poca luz: ningún LED
21         digitalWrite(ledVerde, LOW);
22         digitalWrite(ledNaranja, LOW);
23         digitalWrite(ledRojo, LOW);
24     } else if (valorLuz < 500) {
25         // algo de luz: LED verde
26         digitalWrite(ledVerde, HIGH);
27         digitalWrite(ledNaranja, LOW);
28         digitalWrite(ledRojo, LOW);
29     } else if (valorLuz < 800) {
30         // bastante luz: verde y naranja
31         digitalWrite(ledVerde, HIGH);
32         digitalWrite(ledNaranja, HIGH);
33         digitalWrite(ledRojo, LOW);
34     } else {
35         // mucha luz: los tres LED
36         digitalWrite(ledVerde, HIGH);
37         digitalWrite(ledNaranja, HIGH);
38         digitalWrite(ledRojo, HIGH);
39     }
40 }
```



Con **else if** el programa va comprobando los umbrales de menor a mayor. Por ejemplo, si el sensor lee 650, no se cumple `valorLuz < 200` ni `valorLuz < 500`, pero sí `valorLuz < 800`, así que se encienden el verde y el naranja y el programa ya no comprueba nada más. Si no se cumple ninguna condición, se ejecuta el último `else`: hay mucha luz y se encienden los tres LED.



2.7 Piano de frutas

Mejora 1. Luz y sonido

Además de la constante y el pinMode del LED verde, cambia el if de la función loop():

```
if (valorMkMk > umbral) {  
    digitalWrite(ledVerde, HIGH);    // enciende el LED  
    tone(zumbador, notaDo, duracion); // suena la nota Do  
    delay(duracion);                // espera a que termine la nota  
    digitalWrite(ledVerde, LOW);    // apaga el LED  
}
```



Mejora 2. Escala musical

Posible solución:

```
1 // Escala musical: cuatro teclas MkMk tocan Do, Re, Mi y Fa
2
3 const int teclaDo = A0; // tecla Do en la entrada MkMk A0
4 const int teclaRe = A1; // tecla Re en la entrada MkMk A1
5 const int teclaMi = A2; // tecla Mi en la entrada MkMk A2
6 const int teclaFa = A3; // tecla Fa en la entrada MkMk A3
7 const int zumbador = 10; // zumbador conectado al pin 10
8
9 const int umbral = 350; // por encima de este valor hay contacto
10 const int notaDo = 262; // frecuencias de las notas, en Hz
11 const int notaRe = 294;
12 const int notaMi = 330;
13 const int notaFa = 349;
14 const int duracion = 250; // duración de cada nota, en milisegundos
15
16 void setup() {
17     pinMode(zumbador, OUTPUT); // el zumbador es una salida
18 }
19
20 void loop() {
21     if (analogRead(teclaDo) > umbral) {
22         tone(zumbador, notaDo, duracion); // tecla Do
23         delay(duracion);
24     }
25     if (analogRead(teclaRe) > umbral) {
26         tone(zumbador, notaRe, duracion); // tecla Re
27         delay(duracion);
28     }
29     if (analogRead(teclaMi) > umbral) {
30         tone(zumbador, notaMi, duracion); // tecla Mi
31         delay(duracion);
32     }
33     if (analogRead(teclaFa) > umbral) {
34         tone(zumbador, notaFa, duracion); // tecla Fa
35         delay(duracion);
36     }
37 }
```



Los cuatro `if` van uno detrás de otro, sin `else`: en cada vuelta de `loop()` el programa comprueba las cuatro teclas y toca la nota de la que estás tocando.



Mejora 3. Teclado de colores

Posible solución:



```
1 // Teclado de colores: cada tecla MkMk toca una nota y enciende un color
2
3 const int teclaDo = A0; // tecla Do en la entrada MkMk A0
4 const int teclaRe = A1; // tecla Re en la entrada MkMk A1
5 const int teclaMi = A2; // tecla Mi en la entrada MkMk A2
6 const int teclaFa = A3; // tecla Fa en la entrada MkMk A3
7 const int zumbador = 10; // zumbador conectado al pin 10
8 const int rgbRojo = 9; // LED RGB: color rojo en el pin 9
9 const int rgbVerde = 5; // LED RGB: color verde en el pin 5
10 const int rgbAzul = 6; // LED RGB: color azul en el pin 6
11
12 const int umbral = 350; // por encima de este valor hay contacto
13 const int notaDo = 262; // frecuencias de las notas, en Hz
14 const int notaRe = 294;
15 const int notaMi = 330;
16 const int notaFa = 349;
17 const int duracion = 250; // duración de cada nota, en milisegundos
18
19 void setup() {
20     pinMode(zumbador, OUTPUT);
21     pinMode(rgbRojo, OUTPUT);
22     pinMode(rgbVerde, OUTPUT);
23     pinMode(rgbAzul, OUTPUT);
24 }
25
26 void loop() {
27     if (analogRead(teclaDo) > umbral) {
28         digitalWrite(rgbRojo, HIGH); // Do: rojo
29         tone(zumbador, notaDo, duracion);
30         delay(duracion);
31     }
32     if (analogRead(teclaRe) > umbral) {
33         digitalWrite(rgbVerde, HIGH); // Re: verde
34         tone(zumbador, notaRe, duracion);
35         delay(duracion);
36     }
37     if (analogRead(teclaMi) > umbral) {
38         digitalWrite(rgbAzul, HIGH); // Mi: azul
39         tone(zumbador, notaMi, duracion);
40         delay(duracion);
41     }
42     if (analogRead(teclaFa) > umbral) {
43         digitalWrite(rgbRojo, HIGH); // Fa: blanco (los tres colores)
```



```
44     digitalWrite(rgbVerde, HIGH);
45     digitalWrite(rgbAzul, HIGH);
46     tone(zumbador, notaFa, duracion);
47     delay(duracion);
48 }
49
50 // apaga el LED RGB hasta la siguiente nota
51 digitalWrite(rgbRojo, LOW);
52 digitalWrite(rgbVerde, LOW);
53 digitalWrite(rgbAzul, LOW);
54 }
```

Cada `if` enciende su color, toca su nota y espera a que termine. Al final de `loop()` apagamos los tres colores, así el LED RGB solo está encendido mientras suena una nota. Si mantienes la mano en una fruta, el LED se apaga y se vuelve a encender tan rápido que lo verás siempre encendido.



2.8 El echidna dice la temperatura

Mejora 1. Como la tecla t

Además de la constante y el pinMode del pulsador, cambia la función `loop()`:

```
void loop() {  
  if (digitalRead(pulsadorSR) == HIGH) {  
    // SR pulsado: mide y dice la temperatura  
    lectura = analogRead(sensorTemperatura);  
    temperatura = (lectura * 0.4658) - 50.0;  
  
    Serial.print("Hola, ahora hace una temperatura de ");  
    Serial.print(temperatura);  
    Serial.println(" °C");  
  
    delay(1000); // espera 1 segundo para no repetir la frase  
  }  
}
```



Mejora 2. Grados Fahrenheit

Declara la constante del pulsador SL y la variable `fahrenheit`, configura SL como entrada y cambia la función `loop()`:

```
void loop() {  
  // mide la temperatura en las dos escalas  
  lectura = analogRead(sensorTemperatura);  
  temperatura = (lectura * 0.4658) - 50.0;  
  fahrenheit = temperatura * 1.8 + 32;  
  
  if (digitalRead(pulsadorSR) == HIGH) {  
    // SR pulsado: temperatura en grados Celsius  
    Serial.print("Hola, ahora hace una temperatura de ");  
    Serial.print(temperatura);  
    Serial.println(" °C");  
    delay(1000);  
  }  
  
  if (digitalRead(pulsadorSL) == HIGH) {  
    // SL pulsado: temperatura en grados Fahrenheit  
    Serial.print("Hola, ahora hace una temperatura de ");  
    Serial.print(fahrenheit);  
    Serial.println(" °F");  
    delay(1000);  
  }  
}
```



Mejora 3. Frío o calor

Posible solución:

```
1 // Frío o calor: el LED rojo avisa cuando la temperatura supera el umbral
2
3 const int sensorTemperatura = A6; // sensor de temperatura conectado al pin A6
4 const int ledRojo = 13;           // LED rojo conectado al pin 13
5 const int ledVerde = 11;          // LED verde conectado al pin 11
6
7 const float umbral = 28.0; // por encima de esta temperatura hace calor
8
9 int lectura = 0;             // guarda el valor que lee el sensor (de 0 a 1023)
10 float temperatura = 0.0; // guarda la temperatura en grados Celsius, con
    decimales
11
12 void setup() {
13     Serial.begin(9600);
14     pinMode(ledRojo, OUTPUT);
15     pinMode(ledVerde, OUTPUT);
16 }
17
18 void loop() {
19     lectura = analogRead(sensorTemperatura);
20     temperatura = (lectura * 0.4658) - 50.0;
21
22     Serial.print("Hola, ahora hace una temperatura de ");
23     Serial.print(temperatura);
24     Serial.println(" ºC");
25
26     if (temperatura > umbral) {
27         digitalWrite(ledRojo, HIGH); // calor: LED rojo
28         digitalWrite(ledVerde, LOW);
29     } else {
30         digitalWrite(ledRojo, LOW); // sin calor: LED verde
31         digitalWrite(ledVerde, HIGH);
32     }
33
34     delay(1000); // espera 1 segundo hasta la siguiente medida
35 }
```



El umbral es de tipo `float` porque lo comparamos con la temperatura, que tiene decimales. Si en tu aula hace más calor o más frío, cambia el valor de `umbral` para que el LED rojo se encienda solo al calentar el sensor con el dedo.



2.9 Mezclamos colores

Mejora 1. Más colores

Cambia solo las dos constantes:

```
const int paso = 15;    // 18 valores por color: 0, 15, 30... 255
const int espera = 10;  // cada color se ve 10 milisegundos
```

Ahora cada color toma 18 valores: $18 \times 18 \times 18 = \mathbf{5832 \text{ colores}}$, y la vuelta completa dura aproximadamente un minuto.

Mejora 2. De rojo a azul

Cambia las constantes paso y espera y la función loop():

```
const int paso = 5;      // los colores cambian de 5 en 5
const int espera = 30;   // tiempo entre un color y el siguiente, en milisegundos
```

```
void loop() {
    // del rojo al azul: el azul sube mientras el rojo baja
    for (int valor = 0; valor <= 255; valor = valor + paso) {
        analogWrite(rgbRojo, 255 - valor);
        analogWrite(rgbAzul, valor);
        delay(espera);
    }

    // del azul al rojo: el azul baja mientras el rojo sube
    for (int valor = 255; valor >= 0; valor = valor - paso) {
        analogWrite(rgbRojo, 255 - valor);
        analogWrite(rgbAzul, valor);
        delay(espera);
    }
}
```



Mejora 3. Termómetro de colores

Posible solución:



```
1 // Termómetro de colores: el LED RGB cambia de color según la temperatura
2
3 const int sensorTemperatura = A6; // sensor de temperatura conectado al pin A6
4 const int rgbRojo = 9;           // LED RGB: color rojo en el pin 9
5 const int rgbVerde = 5;          // LED RGB: color verde en el pin 5
6 const int rgbAzul = 6;           // LED RGB: color azul en el pin 6
7
8 const float umbralFrio = 20.0;   // por debajo de esta temperatura hace frío
9 const float umbralCalor = 28.0; // por encima de esta temperatura hace calor
10
11 int lectura = 0;                // guarda el valor que lee el sensor (de 0 a 1023)
12 float temperatura = 0.0;        // guarda la temperatura en grados Celsius, con
decimales
13
14 void setup() {
15     pinMode(rgbRojo, OUTPUT);
16     pinMode(rgbVerde, OUTPUT);
17     pinMode(rgbAzul, OUTPUT);
18     Serial.begin(9600);
19 }
20
21 void loop() {
22     lectura = analogRead(sensorTemperatura);
23     temperatura = (lectura * 0.4658) - 50.0;
24
25     Serial.print("Temperatura: ");
26     Serial.print(temperatura);
27     Serial.println(" °C");
28
29     if (temperatura < umbralFrio) {
30         // frío: azul
31         analogWrite(rgbRojo, 0);
32         analogWrite(rgbVerde, 0);
33         analogWrite(rgbAzul, 255);
34     } else if (temperatura > umbralCalor) {
35         // calor: rojo
36         analogWrite(rgbRojo, 255);
37         analogWrite(rgbVerde, 0);
38         analogWrite(rgbAzul, 0);
39     } else {
40         // temperatura agradable: verde
41         analogWrite(rgbRojo, 0);
42         analogWrite(rgbVerde, 255);
43     }
44 }
```



```
43     analogWrite(rgbAzul, 0);  
44 }  
45  
46 delay(1000); // espera 1 segundo hasta la siguiente medida  
47 }
```

El programa usa dos umbrales: si la temperatura está por debajo de `umbralFrio`, el LED se pone azul; si está por encima de `umbralCalor`, rojo; y si no se cumple ninguna de las dos condiciones, está entre los dos umbrales y se pone verde. Para probarlo, calienta el sensor tocándolo suavemente con el dedo y observa la temperatura en el Monitor serie; si hace falta, ajusta los umbrales a la temperatura de tu aula.



2.10 Vúmetro

Mejora 1. Calibra tu aula

Añade al final de `setup()`:

```
Serial.begin(9600); // abre la comunicación con el ordenador
```

y en `loop()`, justo después de leer el micrófono:

```
Serial.println(valorSonido); // envía el valor al Monitor serie
```

Cuando sepas los valores, cambia los números de las constantes `umbralSilencio` y `umbralRuido`.

Mejora 2. Una lectura más estable

Declara la constante `muestras` y la variable `suma`:

```
const int muestras = 32; // número de lecturas para calcular la media

int suma = 0; // va sumando las lecturas para calcular la media
```

y, en `loop()`, cambia la lectura del micrófono por:

```
// suma varias lecturas del micrófono y calcula la media
suma = 0;
for (int i = 0; i < muestras; i++) {
    suma = suma + analogRead(microfono);
}
valorSonido = suma / muestras;
```

Es importante poner `suma = 0` al principio de cada vuelta: si no, la suma iría acumulando las lecturas de las vueltas anteriores y la media saldría cada vez más grande. La variable `i` solo sirve para contar las lecturas: el bucle se repite mientras `i` sea menor que `muestras`, y `i++` es una forma corta de escribir `i = i + 1`.



Mejora 3. Encendido por palmada

Posible solución:

```
1 // Encendido por palmada: cada palmada enciende o apaga el LED rojo
2
3 const int microfono = A7; // micrófono conectado al pin A7
4 const int ledRojo = 13;    // LED rojo conectado al pin 13
5
6 const int umbralPalmada = 20; // por encima de este valor hay una palmada
7
8 int estadoLED = LOW; // recuerda si el LED está apagado o encendido
9
10 void setup() {
11     pinMode(ledRojo, OUTPUT);
12     analogReference(INTERNAL); // referencia de 1,1 V
13 }
14
15 void loop() {
16     if (analogRead(microfono) > umbralPalmada) {
17         // palmada: cambia el LED al estado contrario
18         if (estadoLED == LOW) {
19             estadoLED = HIGH;
20         } else {
21             estadoLED = LOW;
22         }
23         digitalWrite(ledRojo, estadoLED);
24
25         delay(300); // espera a que termine el sonido de la palmada
26     }
27 }
```

Cada vez que el micrófono supera `umbralPalmada`, el programa cambia el LED al estado contrario y actualiza la variable **estadoLED**. El `delay(300)` hace que el programa no vuelva a escuchar hasta que termine el sonido de la palmada; si no, una sola palmada podría encender y apagar el LED varias veces seguidas. Si el LED cambia al hablar, sube el umbral; si no reacciona a las palmadas, bájalo.



2.11 Regulador de luz con joystick

Mejora 1. Lee el joystick

Este programa solo lee el joystick y envía los dos ejes al ordenador:

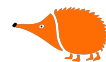
```
1 // Lee el joystick: dibuja los ejes X e Y en el Serial Plotter
2
3 const int joystickX = A0; // eje X del joystick en el pin A0
4 const int joystickY = A1; // eje Y del joystick en el pin A1
5
6 int valorX = 0; // guarda la posición del eje X
7 int valorY = 0; // guarda la posición del eje Y
8
9 void setup() {
10   Serial.begin(9600); // abre la comunicación a 9600 baudios
11 }
12
13 void loop() {
14   valorX = analogRead(joystickX);
15   valorY = analogRead(joystickY);
16
17   Serial.print("X:"); // nombre del eje X
18   Serial.print(valorX); // valor del eje X
19   Serial.print(" Y:"); // nombre del eje Y
20   Serial.println(valorY); // valor del eje Y y salto de línea
21
22   delay(100); // diez medidas por segundo
23 }
```



Mejora 2. Mando de colores

Posible solución:

```
1 // Mando de colores: el eje X controla el rojo y el eje Y, el azul
2
3 const int joystickX = A0; // eje X del joystick en el pin A0
4 const int joystickY = A1; // eje Y del joystick en el pin A1
5 const int rgbRojo = 9;    // LED RGB: color rojo en el pin 9
6 const int rgbAzul = 6;    // LED RGB: color azul en el pin 6
7
8 int valorX = 0; // guarda la posición del eje X
9 int valorY = 0; // guarda la posición del eje Y
10 int rojo = 0;   // intensidad del rojo (de 0 a 255)
11 int azul = 0;   // intensidad del azul (de 0 a 255)
12
13 void setup() {
14     pinMode(rgbRojo, OUTPUT);
15     pinMode(rgbAzul, OUTPUT);
16 }
17
18 void loop() {
19     valorX = analogRead(joystickX);
20     valorY = analogRead(joystickY);
21
22     // cada eje, de 0 a 1023, pasa a una intensidad de 0 a 255
23     rojo = map(valorX, 0, 1023, 0, 255);
24     azul = map(valorY, 0, 1023, 0, 255);
25
26     analogWrite(rgbRojo, rojo);
27     analogWrite(rgbAzul, azul);
28 }
```



Mejora 3. Theremín

Posible solución:

```
1 // Theremín: el eje X del joystick elige la nota del zumbador
2
3 const int joystickX = A0; // eje X del joystick en el pin A0
4 const int zumbador = 10; // zumbador conectado al pin 10
5
6 const int centroMin = 492; // zona del centro: de 492...
7 const int centroMax = 532; // ...a 532
8
9 const int notaGrave = 200; // frecuencia a la izquierda, en Hz
10 const int notaAguda = 1000; // frecuencia a la derecha, en Hz
11
12 int valorX = 0; // guarda la posición del joystick en el eje X
13 int frecuencia = 0; // guarda la nota que suena, en Hz
14
15 void setup() {
16     pinMode(zumbador, OUTPUT);
17 }
18
19 void loop() {
20     valorX = analogRead(joystickX);
21
22     if (valorX < centroMin || valorX > centroMax) {
23         // fuera del centro: suena una nota según la posición
24         frecuencia = map(valorX, 0, 1023, notaGrave, notaAguda);
25         tone(zumbador, frecuencia);
26     } else {
27         noTone(zumbador); // en el centro: silencio
28     }
29 }
```

Como tone sigue sonando hasta que llamamos a noTone, la nota cambia en cuanto movemos el joystick, sin cortes.



2.12 Nivel de burbuja

Mejora 1. Lee el acelerómetro

Este programa solo lee el acelerómetro y envía los ejes X e Y al ordenador:

```
1 // Lee el acelerómetro: dibuja los ejes X e Y en el Serial Plotter
2
3 #include <Adafruit_LIS3DH.h> // librería del acelerómetro
4
5 Adafruit_LIS3DH acelerometro; // el acelerómetro de la placa
6
7 void setup() {
8     Serial.begin(9600); // abre la comunicación a 9600 baudios
9     acelerometro.begin(0x18); // pone en marcha el acelerómetro
10 }
11
12 void loop() {
13     acelerometro.read(); // lee el acelerómetro
14
15     Serial.print("X:"); // nombre del eje X
16     Serial.print(acelerometro.x_g); // valor del eje X
17     Serial.print(" Y:"); // nombre del eje Y
18     Serial.println(acelerometro.y_g); // valor del eje Y y salto de línea
19
20     delay(100); // diez medidas por segundo
21 }
```

Mejora 2. Alarma de caída

Declara la constante del zumbador (pin 10), configúralo como salida en `setup()` y añade al final de la función `loop()`:

```
// muy inclinada hacia cualquier lado: suena la alarma
if (inclinacion > inclinacionMucha || inclinacion < -inclinacionMucha) {
    analogWrite(zumbador, 125);
} else {
    analogWrite(zumbador, 0);
}
```

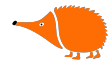


Mejora 3. La luz que rueda

Posible solución:



```
1 // La luz que rueda: la luz se desplaza por los LED hacia el lado inclinado
2
3 #include <Adafruit_LIS3DH.h> // librería del acelerómetro
4
5 const int rgbAzul = 6;      // LED RGB: color azul en el pin 6
6 const int ledVerde = 11;    // LED verde conectado al pin 11
7 const int ledNaranja = 12;  // LED naranja conectado al pin 12
8 const int ledRojo = 13;     // LED rojo conectado al pin 13
9
10 const float inclinacionPoca = 0.2; // inclinada (en g)
11 const int espera = 200;          // tiempo entre un paso y el siguiente, en ms
12
13 Adafruit_LIS3DH acelerometro; // el acelerómetro de la placa
14
15 float inclinacion = 0.0; // guarda la aceleración del eje Y, en g
16 int posicion = 1;        // LED encendido: 1 verde, 2 naranja, 3 rojo, 4 azul
17
18 void setup() {
19     pinMode(rgbAzul, OUTPUT);
20     pinMode(ledVerde, OUTPUT);
21     pinMode(ledNaranja, OUTPUT);
22     pinMode(ledRojo, OUTPUT);
23     acelerometro.begin(0x18);
24 }
25
26 void loop() {
27     acelerometro.read();
28     inclinacion = acelerometro.y_g;
29
30     // la luz avanza hacia el lado inclinado
31     if (inclinacion < -inclinacionPoca) {
32         posicion = posicion - 1;
33     } else if (inclinacion > inclinacionPoca) {
34         posicion = posicion + 1;
35     }
36
37     // al pasar de un extremo, vuelve por el otro
38     if (posicion > 4) {
39         posicion = 1;
40     }
41     if (posicion < 1) {
42         posicion = 4;
43     }
```



```
44
45 // enciende solo el LED de la posición actual
46 if (posicion == 1) {
47     digitalWrite(ledVerde, HIGH);
48 } else {
49     digitalWrite(ledVerde, LOW);
50 }
51 if (posicion == 2) {
52     digitalWrite(ledNaranja, HIGH);
53 } else {
54     digitalWrite(ledNaranja, LOW);
55 }
56 if (posicion == 3) {
57     digitalWrite(ledRojo, HIGH);
58 } else {
59     digitalWrite(ledRojo, LOW);
60 }
61 if (posicion == 4) {
62     digitalWrite(rgbAzul, HIGH);
63 } else {
64     digitalWrite(rgbAzul, LOW);
65 }
66
67 delay(espera);
68 }
```

El operador == («igual a») compara dos valores. No hay que confundirlo con =, que **guarda** un valor en una variable.



2.13 Control desde el ordenador

Mejora 1. Elige el color

Añade dentro del `if (Serial.available() > 0)`, después del `if` de la L:

```
if (letra == 'R') {  
    // R (de Red): solo el rojo  
    digitalWrite(rgbRojo, HIGH);  
    digitalWrite(rgbVerde, LOW);  
    digitalWrite(rgbAzul, LOW);  
    Serial.println("LED rojo");  
}  
  
if (letra == 'G') {  
    // G (de Green): solo el verde  
    digitalWrite(rgbRojo, LOW);  
    digitalWrite(rgbVerde, HIGH);  
    digitalWrite(rgbAzul, LOW);  
    Serial.println("LED verde");  
}  
  
if (letra == 'B') {  
    // B (de Blue): solo el azul  
    digitalWrite(rgbRojo, LOW);  
    digitalWrite(rgbVerde, LOW);  
    digitalWrite(rgbAzul, HIGH);  
    Serial.println("LED azul");  
}
```



Mejora 2. Brillo desde el teclado

Posible solución:



```
1 // Brillo desde el teclado: un número terminado en > da el brillo
2
3 const int rgbRojo = 9; // LED RGB: color rojo en el pin 9
4 const int rgbVerde = 5; // LED RGB: color verde en el pin 5
5 const int rgbAzul = 6; // LED RGB: color azul en el pin 6
6
7 char letra = ' '; // guarda cada carácter que llega del ordenador
8 int numero = 0; // va juntando las cifras del número
9 int brillo = 0; // guarda el brillo del LED (de 0 a 255)
10
11 void setup() {
12     pinMode(rgbRojo, OUTPUT);
13     pinMode(rgbVerde, OUTPUT);
14     pinMode(rgbAzul, OUTPUT);
15     Serial.begin(9600);
16 }
17
18 void loop() {
19     if (Serial.available() > 0) {
20         letra = Serial.read();
21
22         if (letra >= '0' && letra <= '9') {
23             // una cifra: la añade al final del número
24             numero = numero * 10 + (letra - '0');
25             if (numero > 255) {
26                 numero = 255; // como máximo, 255
27             }
28         } else if (letra == '>') {
29             // fin del número: lo usa como brillo
30             brillo = numero;
31             Serial.print("Brillo: ");
32             Serial.println(brillo);
33             numero = 0; // listo para el siguiente número
34         }
35     }
36
37     // enciende el LED en blanco con el brillo actual
38     analogWrite(rgbRojo, brillo);
39     analogWrite(rgbVerde, brillo);
40     analogWrite(rgbAzul, brillo);
41 }
```



El `if (numero > 255)` evita que, si escribes un número demasiado grande, `analogWrite` reciba un valor que no admite.



Mejora 3. Color a la carta

Posible solución:



```
1 // Color a la carta: 254r109g4b pone el LED RGB naranja Echidna
2
3 const int rgbRojo = 9; // LED RGB: color rojo en el pin 9
4 const int rgbVerde = 5; // LED RGB: color verde en el pin 5
5 const int rgbAzul = 6; // LED RGB: color azul en el pin 6
6
7 char letra = ' '; // guarda cada carácter que llega del ordenador
8 int numero = 0; // va juntando las cifras del número
9 int rojo = 0; // intensidad del rojo (de 0 a 255)
10 int verde = 0; // intensidad del verde (de 0 a 255)
11 int azul = 0; // intensidad del azul (de 0 a 255)
12
13 void setup() {
14     pinMode(rgbRojo, OUTPUT);
15     pinMode(rgbVerde, OUTPUT);
16     pinMode(rgbAzul, OUTPUT);
17     Serial.begin(9600);
18 }
19
20 void loop() {
21     if (Serial.available() > 0) {
22         letra = Serial.read();
23
24         if (letra >= '0' && letra <= '9') {
25             // una cifra: la añade al final del número
26             numero = numero * 10 + (letra - '0');
27             if (numero > 255) {
28                 numero = 255; // como máximo, 255
29             }
30         } else if (letra == 'r') {
31             rojo = numero; // el número es la intensidad del rojo
32             Serial.print("Rojo: ");
33             Serial.println(rojo);
34             numero = 0;
35         } else if (letra == 'g') {
36             verde = numero; // el número es la intensidad del verde
37             Serial.print("Verde: ");
38             Serial.println(verde);
39             numero = 0;
40         } else if (letra == 'b') {
41             azul = numero; // el número es la intensidad del azul
42             Serial.print("Azul: ");
43             Serial.println(azul);
```



```
44     numero = 0;
45   }
46 }
47
48 // mezcla los tres colores
49 analogWrite(rgbRojo, rojo);
50 analogWrite(rgbVerde, verde);
51 analogWrite(rgbAzul, azul);
52 }
```

Puedes cambiar un solo color sin tocar los demás: si después escribes 0g, el verde se apaga y el rojo y el azul siguen igual.