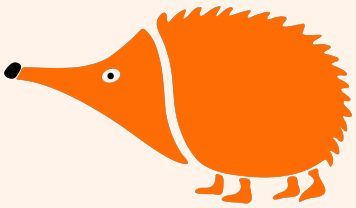
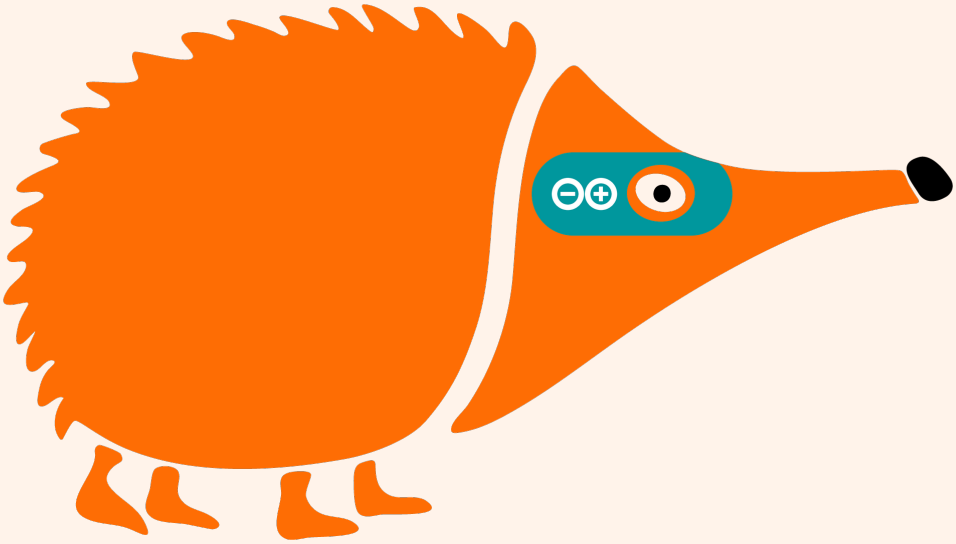


Guía docente

Proyectos de inicio con Arduino IDE



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:

```
// Parpadeo alterno: el LED rojo y el verde se encienden por turnos

const int ledRojo = 13;   // LED rojo conectado al pin 13
const int ledVerde = 11;  // LED verde conectado al pin 11

void setup() {
  pinMode(ledRojo, OUTPUT);   // el pin del LED rojo es una salida
  pinMode(ledVerde, OUTPUT);  // el pin del LED verde es una salida
}

void loop() {
  digitalWrite(ledRojo, HIGH);  // enciende el rojo
  digitalWrite(ledVerde, LOW);  // apaga el verde
  delay(1000);                  // espera 1 segundo
  digitalWrite(ledRojo, LOW);   // apaga el rojo
  digitalWrite(ledVerde, HIGH); // enciende el verde
  delay(1000);                  // espera 1 segundo
}
```

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:

```
// Semáforo con semáforo de peatones en el LED RGB

const int ledVerde = 11;    // LED verde conectado al pin 11
const int ledNaranja = 12;  // LED naranja conectado al pin 12
const int ledRojo = 13;     // LED rojo conectado al pin 13
const int peatonRojo = 9;   // rojo del LED RGB conectado al pin 9
const int peatonVerde = 5;  // verde del LED RGB conectado al pin 5

void setup() {
  pinMode(ledVerde, OUTPUT);
  pinMode(ledNaranja, OUTPUT);
  pinMode(ledRojo, OUTPUT);
  pinMode(peatonRojo, OUTPUT);
  pinMode(peatonVerde, OUTPUT);
}

void loop() {
  // coches en verde: peatones en rojo
  digitalWrite(peatonRojo, HIGH);
  digitalWrite(ledVerde, HIGH);
  delay(5000);
  digitalWrite(ledVerde, LOW);

  // coches en naranja: peatones siguen en rojo
  digitalWrite(ledNaranja, HIGH);
  delay(2000);
  digitalWrite(ledNaranja, LOW);

  // coches en rojo: peatones en verde
  digitalWrite(peatonRojo, LOW);
  digitalWrite(peatonVerde, HIGH);
  digitalWrite(ledRojo, HIGH);
  delay(5000);
  digitalWrite(ledRojo, LOW);
  digitalWrite(peatonVerde, LOW);
}
```

Mejora 3. Semáforo sonoro adaptado

Posible solución:

```
// Semáforo sonoro: pitidos lentos en verde y silencio en rojo

const int ledVerde = 11;    // LED verde conectado al pin 11
const int ledNaranja = 12;  // LED naranja conectado al pin 12
const int ledRojo = 13;     // LED rojo conectado al pin 13
const int zumbador = 10;    // zumbador conectado al pin 10

void setup() {
  pinMode(ledVerde, OUTPUT);
  pinMode(ledNaranja, OUTPUT);
  pinMode(ledRojo, OUTPUT);
  pinMode(zumbador, OUTPUT);
}

void loop() {
  // fase verde: 5 pitidos lentos (5 segundos)
  digitalWrite(ledVerde, HIGH);
  for (int i = 0; i < 5; i++) {
    analogWrite(zumbador, 125);  // suena
    delay(500);
    analogWrite(zumbador, 0);    // calla
    delay(500);
  }
  digitalWrite(ledVerde, LOW);

  // fase naranja: 2 segundos
  digitalWrite(ledNaranja, HIGH);
  delay(2000);
  digitalWrite(ledNaranja, LOW);

  // fase roja: 5 segundos en silencio
  digitalWrite(ledRojo, HIGH);
  delay(5000);
  digitalWrite(ledRojo, LOW);
}
```

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:

```
// Pulsador con memoria: SL enciende y apaga el LED rojo

const int pulsadorSL = 3;  // pulsador izquierdo conectado al pin 3
const int ledRojo = 13;    // LED rojo conectado al pin 13

int estadoLED = LOW;  // recuerda si el LED está apagado o encendido

void setup() {
  pinMode(pulsadorSL, INPUT);
  pinMode(ledRojo, OUTPUT);
}

void loop() {
  if (digitalRead(pulsadorSL) == HIGH) {
    // cambia el LED al estado contrario
    if (estadoLED == LOW) {
      estadoLED = HIGH;
    } else {
      estadoLED = LOW;
    }
    digitalWrite(ledRojo, estadoLED);

    // espera a que sueltes el pulsador
    while (digitalRead(pulsadorSL) == HIGH) {
    }
    delay(50);  // evita los rebotes del pulsador al soltarlo
  }
}
```

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:

```ino
// Timbre de dos tonos: SL suena grave y SR, agudo

const int pulsadorSR = 2;  // pulsador derecho conectado al pin 2
const int pulsadorSL = 3;  // pulsador izquierdo conectado al pin 3
const int zumbador = 10;   // zumbador conectado al pin 10

const int tonoGrave = 440;  // frecuencia del tono de SL, en Hz
const int tonoAgudo = 880;  // frecuencia del tono de SR, en Hz

int estadoSR = LOW;  // guarda si SR está pulsado (HIGH) o no (LOW)
int estadoSL = LOW;  // guarda si SL está pulsado (HIGH) o no (LOW)

void setup() {
  pinMode(pulsadorSR, INPUT);
  pinMode(pulsadorSL, INPUT);
  pinMode(zumbador, OUTPUT);
}

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

  if (estadoSL == HIGH) {
    tone(zumbador, tonoGrave);  // SL pulsado: tono grave
  } else {
    if (estadoSR == HIGH) {
      tone(zumbador, tonoAgudo);  // SR pulsado: tono agudo
    } else {
      noTone(zumbador);  // ninguno pulsado: silencio
    }
  }
}
```

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:

```ino
// Regulador de brillo: SR sube el brillo del LED verde y SL lo baja

const int pulsadorSR = 2;  // pulsador derecho conectado al pin 2
const int pulsadorSL = 3;  // pulsador izquierdo conectado al pin 3
const int ledVerde = 11;   // LED verde conectado al pin 11

const int espera = 10;  // velocidad a la que cambia el brillo, en milisegundos

int estadoSR = LOW;  // guarda si SR está pulsado (HIGH) o no (LOW)
int estadoSL = LOW;  // guarda si SL está pulsado (HIGH) o no (LOW)
int brillo = 0;      // guarda el brillo del LED (de 0 a 255)

void setup() {
  pinMode(pulsadorSR, INPUT);
  pinMode(pulsadorSL, INPUT);
  pinMode(ledVerde, OUTPUT);
}

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

  // SR pulsado: sube el brillo sin pasar de 255
  if (estadoSR == HIGH) {
    if (brillo < 255) {
      brillo = brillo + 1;
    }
  }

  // SL pulsado: baja el brillo sin bajar de 0
  if (estadoSL == HIGH) {
    if (brillo > 0) {
      brillo = brillo - 1;
    }
  }

  analogWrite(ledVerde, brillo);  // enciende el LED con el brillo actual

  delay(espera);
}
```

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:

```
// Lee el sensor de luz: muestra su valor en el Monitor serie

const int sensorLuz = A3;  // sensor de luz (LDR) conectado al pin A3

int valorLuz = 0;  // guarda la luz que mide el sensor (de 0 a 1023)

void setup() {
  Serial.begin(9600);  // abre la comunicación a 9600 baudios
}

void loop() {
  valorLuz = analogRead(sensorLuz);  // lee el sensor

  Serial.print("Valor del sensor de luz: ");  // texto
  Serial.println(valorLuz);                   // valor y salto de línea

  delay(1000);  // espera 1 segundo para poder leer los valores
}
```

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:

```
// Linterna automática: el LED RGB se enciende en blanco cuando hay poca luz

const int sensorLuz = A3;  // sensor de luz (LDR) conectado al pin A3
const int rgbRojo = 9;     // LED RGB: color rojo en el pin 9
const int rgbVerde = 5;    // LED RGB: color verde en el pin 5
const int rgbAzul = 6;     // LED RGB: color azul en el pin 6

const int umbral = 200;  // por debajo de este valor consideramos que es de noche

int valorLuz = 0;  // guarda la luz que mide el sensor (de 0 a 1023)

void setup() {
  pinMode(rgbRojo, OUTPUT);  // los tres colores del LED RGB son salidas
  pinMode(rgbVerde, OUTPUT);
  pinMode(rgbAzul, OUTPUT);
}

void loop() {
  valorLuz = analogRead(sensorLuz);

  if (valorLuz < umbral) {
    // poca luz: los tres colores encendidos dan luz blanca
    digitalWrite(rgbRojo, HIGH);
    digitalWrite(rgbVerde, HIGH);
    digitalWrite(rgbAzul, HIGH);
  } else {
    // mucha luz: LED RGB apagado
    digitalWrite(rgbRojo, LOW);
    digitalWrite(rgbVerde, LOW);
    digitalWrite(rgbAzul, LOW);
  }
}
```

Mejora 3. Medidor de luz

Posible solución:

```ino
// Medidor de luz: cuanta más luz, más LED encendidos

const int sensorLuz = A3;   // sensor de luz (LDR) conectado al pin A3
const int ledVerde = 11;    // LED verde conectado al pin 11
const int ledNaranja = 12;  // LED naranja conectado al pin 12
const int ledRojo = 13;     // LED rojo conectado al pin 13

int valorLuz = 0;  // guarda la luz que mide el sensor (de 0 a 1023)

void setup() {
  pinMode(ledVerde, OUTPUT);
  pinMode(ledNaranja, OUTPUT);
  pinMode(ledRojo, OUTPUT);
}

void loop() {
  valorLuz = analogRead(sensorLuz);

  if (valorLuz < 200) {
    // muy poca luz: ningún LED
    digitalWrite(ledVerde, LOW);
    digitalWrite(ledNaranja, LOW);
    digitalWrite(ledRojo, LOW);
  } else if (valorLuz < 500) {
    // algo de luz: LED verde
    digitalWrite(ledVerde, HIGH);
    digitalWrite(ledNaranja, LOW);
    digitalWrite(ledRojo, LOW);
  } else if (valorLuz < 800) {
    // bastante luz: verde y naranja
    digitalWrite(ledVerde, HIGH);
    digitalWrite(ledNaranja, HIGH);
    digitalWrite(ledRojo, LOW);
  } else {
    // mucha luz: los tres LED
    digitalWrite(ledVerde, HIGH);
    digitalWrite(ledNaranja, HIGH);
    digitalWrite(ledRojo, HIGH);
  }
}
```

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

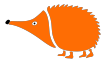
Además de la constante y el `pinMode` del LED verde, cambia el `if` de la función `loop()`:

```
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:

```
// Escala musical: cuatro teclas MkMk tocan Do, Re, Mi y Fa

const int teclaDo = A0;   // tecla Do en la entrada MkMk A0
const int teclaRe = A1;   // tecla Re en la entrada MkMk A1
const int teclaMi = A2;   // tecla Mi en la entrada MkMk A2
const int teclaFa = A3;   // tecla Fa en la entrada MkMk A3
const int zumbador = 10;  // zumbador conectado al pin 10

const int umbral = 350;    // por encima de este valor hay contacto
const int notaDo = 262;    // frecuencias de las notas, en Hz
const int notaRe = 294;
const int notaMi = 330;
const int notaFa = 349;
const int duracion = 250;  // duración de cada nota, en milisegundos

void setup() {
  pinMode(zumbador, OUTPUT);  // el zumbador es una salida
}

void loop() {
  if (analogRead(teclaDo) > umbral) {
    tone(zumbador, notaDo, duracion);  // tecla Do
    delay(duracion);
  }
  if (analogRead(teclaRe) > umbral) {
    tone(zumbador, notaRe, duracion);  // tecla Re
    delay(duracion);
  }
  if (analogRead(teclaMi) > umbral) {
    tone(zumbador, notaMi, duracion);  // tecla Mi
    delay(duracion);
  }
  if (analogRead(teclaFa) > umbral) {
    tone(zumbador, notaFa, duracion);  // tecla Fa
    delay(duracion);
  }
}
```



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 estés tocando.

Mejora 3. Teclado de colores

Posible solución:

```ino
// Teclado de colores: cada tecla MkMk toca una nota y enciende un color

const int teclaDo = A0;   // tecla Do en la entrada MkMk A0
const int teclaRe = A1;   // tecla Re en la entrada MkMk A1
const int teclaMi = A2;   // tecla Mi en la entrada MkMk A2
const int teclaFa = A3;   // tecla Fa en la entrada MkMk A3
const int zumbador = 10;  // zumbador conectado al pin 10
const int rgbRojo = 9;    // LED RGB: color rojo en el pin 9
const int rgbVerde = 5;   // LED RGB: color verde en el pin 5
const int rgbAzul = 6;    // LED RGB: color azul en el pin 6

const int umbral = 350;    // por encima de este valor hay contacto
const int notaDo = 262;    // frecuencias de las notas, en Hz
const int notaRe = 294;
const int notaMi = 330;
const int notaFa = 349;
const int duracion = 250;  // duración de cada nota, en milisegundos

void setup() {
  pinMode(zumbador, OUTPUT);
  pinMode(rgbRojo, OUTPUT);
  pinMode(rgbVerde, OUTPUT);
  pinMode(rgbAzul, OUTPUT);
}

void loop() {
  if (analogRead(teclaDo) > umbral) {
    digitalWrite(rgbRojo, HIGH);  // Do: rojo
    tone(zumbador, notaDo, duracion);
    delay(duracion);
  }
  if (analogRead(teclaRe) > umbral) {
    digitalWrite(rgbVerde, HIGH);  // Re: verde
    tone(zumbador, notaRe, duracion);
    delay(duracion);
  }
  if (analogRead(teclaMi) > umbral) {
    digitalWrite(rgbAzul, HIGH);  // Mi: azul
    tone(zumbador, notaMi, duracion);
    delay(duracion);
  }
  if (analogRead(teclaFa) > umbral) {
    digitalWrite(rgbRojo, HIGH);  // Fa: blanco (los tres colores)
```

```
    digitalWrite(rgbVerde, HIGH);
    digitalWrite(rgbAzul, HIGH);
    tone(zumbador, notaFa, duracion);
    delay(duracion);
  }

  // apaga el LED RGB hasta la siguiente nota
  digitalWrite(rgbRojo, LOW);
  digitalWrite(rgbVerde, LOW);
  digitalWrite(rgbAzul, LOW);
}
```

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(" ºC");

    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(" ºC");
    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(" ºF");
    delay(1000);
  }
}
```

Mejora 3. Frío o calor

Posible solución:

```
// Frío o calor: el LED rojo avisa cuando la temperatura supera el umbral

const int sensorTemperatura = A6;  // sensor de temperatura conectado al pin A6
const int ledRojo = 13;            // LED rojo conectado al pin 13
const int ledVerde = 11;           // LED verde conectado al pin 11

const float umbral = 28.0;  // por encima de esta temperatura hace calor

int lectura = 0;          // guarda el valor que lee el sensor (de 0 a 1023)
float temperatura = 0.0;  // guarda la temperatura en grados Celsius, con
decimales

void setup() {
  Serial.begin(9600);
  pinMode(ledRojo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
}

void loop() {
  lectura = analogRead(sensorTemperatura);
  temperatura = (lectura * 0.4658) - 50.0;

  Serial.print("Hola, ahora hace una temperatura de ");
  Serial.print(temperatura);
  Serial.println(" ºC");

  if (temperatura > umbral) {
    digitalWrite(ledRojo, HIGH);  // calor: LED rojo
    digitalWrite(ledVerde, LOW);
  } else {
    digitalWrite(ledRojo, LOW);   // sin calor: LED verde
    digitalWrite(ledVerde, HIGH);
  }

  delay(1000);  // espera 1 segundo hasta la siguiente medida
}
```

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 × 18 × 18 = **5832 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:

```
// Termómetro de colores: el LED RGB cambia de color según la temperatura

const int sensorTemperatura = A6;  // sensor de temperatura conectado al pin A6
const int rgbRojo = 9;             // LED RGB: color rojo en el pin 9
const int rgbVerde = 5;            // LED RGB: color verde en el pin 5
const int rgbAzul = 6;             // LED RGB: color azul en el pin 6

const float umbralFrio = 20.0;   // por debajo de esta temperatura hace frío
const float umbralCalor = 28.0;  // por encima de esta temperatura hace calor

int lectura = 0;          // guarda el valor que lee el sensor (de 0 a 1023)
float temperatura = 0.0;  // guarda la temperatura en grados Celsius, con
decimales

void setup() {
  pinMode(rgbRojo, OUTPUT);
  pinMode(rgbVerde, OUTPUT);
  pinMode(rgbAzul, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  lectura = analogRead(sensorTemperatura);
  temperatura = (lectura * 0.4658) - 50.0;

  Serial.print("Temperatura: ");
  Serial.print(temperatura);
  Serial.println(" ºC");

  if (temperatura < umbralFrio) {
    // frío: azul
    analogWrite(rgbRojo, 0);
    analogWrite(rgbVerde, 0);
    analogWrite(rgbAzul, 255);
  } else if (temperatura > umbralCalor) {
    // calor: rojo
    analogWrite(rgbRojo, 255);
    analogWrite(rgbVerde, 0);
    analogWrite(rgbAzul, 0);
  } else {
    // temperatura agradable: verde
    analogWrite(rgbRojo, 0);
    analogWrite(rgbVerde, 255);
```

```
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:

```
// Encendido por palmada: cada palmada enciende o apaga el LED rojo

const int microfono = A7;  // micrófono conectado al pin A7
const int ledRojo = 13;    // LED rojo conectado al pin 13

const int umbralPalmada = 20;  // por encima de este valor hay una palmada

int estadoLED = LOW;  // recuerda si el LED está apagado o encendido

void setup() {
  pinMode(ledRojo, OUTPUT);
  analogReference(INTERNAL);  // referencia de 1,1 V
}

void loop() {
  if (analogRead(microfono) > umbralPalmada) {
    // palmada: cambia el LED al estado contrario
    if (estadoLED == LOW) {
      estadoLED = HIGH;
    } else {
      estadoLED = LOW;
    }
    digitalWrite(ledRojo, estadoLED);

    delay(300);  // espera a que termine el sonido de la palmada
  }
}
```

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:

```
// Lee el joystick: dibuja los ejes X e Y en el Serial Plotter

const int joystickX = A0;  // eje X del joystick en el pin A0
const int joystickY = A1;  // eje Y del joystick en el pin A1

int valorX = 0;  // guarda la posición del eje X
int valorY = 0;  // guarda la posición del eje Y

void setup() {
  Serial.begin(9600);  // abre la comunicación a 9600 baudios
}

void loop() {
  valorX = analogRead(joystickX);
  valorY = analogRead(joystickY);

  Serial.print("X:");      // nombre del eje X
  Serial.print(valorX);    // valor del eje X
  Serial.print(" Y:");     // nombre del eje Y
  Serial.println(valorY);  // valor del eje Y y salto de línea

  delay(100);  // diez medidas por segundo
}
```

Mejora 2. Mando de colores

Posible solución:

```
// Mando de colores: el eje X controla el rojo y el eje Y, el azul

const int joystickX = A0;  // eje X del joystick en el pin A0
const int joystickY = A1;  // eje Y del joystick en el pin A1
const int rgbRojo = 9;     // LED RGB: color rojo en el pin 9
const int rgbAzul = 6;     // LED RGB: color azul en el pin 6

int valorX = 0;  // guarda la posición del eje X
int valorY = 0;  // guarda la posición del eje Y
int rojo = 0;    // intensidad del rojo (de 0 a 255)
int azul = 0;    // intensidad del azul (de 0 a 255)

void setup() {
  pinMode(rgbRojo, OUTPUT);
  pinMode(rgbAzul, OUTPUT);
}

void loop() {
  valorX = analogRead(joystickX);
  valorY = analogRead(joystickY);

  // cada eje, de 0 a 1023, pasa a una intensidad de 0 a 255
  rojo = map(valorX, 0, 1023, 0, 255);
  azul = map(valorY, 0, 1023, 0, 255);

  analogWrite(rgbRojo, rojo);
  analogWrite(rgbAzul, azul);
}
```

Mejora 3. Theremín

Posible solución:

```
// Theremín: el eje X del joystick elige la nota del zumbador

const int joystickX = A0;  // eje X del joystick en el pin A0
const int zumbador = 10;   // zumbador conectado al pin 10

const int centroMin = 492;  // zona del centro: de 492...
const int centroMax = 532;  // ...a 532

const int notaGrave = 200;   // frecuencia a la izquierda, en Hz
const int notaAguda = 1000;  // frecuencia a la derecha, en Hz

int valorX = 0;      // guarda la posición del joystick en el eje X
int frecuencia = 0;  // guarda la nota que suena, en Hz

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

void loop() {
  valorX = analogRead(joystickX);

  if (valorX < centroMin || valorX > centroMax) {
    // fuera del centro: suena una nota según la posición
    frecuencia = map(valorX, 0, 1023, notaGrave, notaAguda);
    tone(zumbador, frecuencia);
  } else {
    noTone(zumbador);  // en el centro: silencio
  }
}
```

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:

```
// Lee el acelerómetro: dibuja los ejes X e Y en el Serial Plotter

#include <Adafruit_LIS3DH.h>  // librería del acelerómetro

Adafruit_LIS3DH acelerometro;  // el acelerómetro de la placa

void setup() {
  Serial.begin(9600);        // abre la comunicación a 9600 baudios
  acelerometro.begin(0x18);  // pone en marcha el acelerómetro
}

void loop() {
  acelerometro.read();  // lee el acelerómetro

  Serial.print("X:");               // nombre del eje X
  Serial.print(acelerometro.x_g);   // valor del eje X
  Serial.print(" Y:");              // nombre del eje Y
  Serial.println(acelerometro.y_g); // valor del eje Y y salto de línea

  delay(100);  // diez medidas por segundo
}
```

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:

```ino
// La luz que rueda: la luz se desplaza por los LED hacia el lado inclinado

#include <Adafruit_LIS3DH.h>  // librería del acelerómetro

const int rgbAzul = 6;      // LED RGB: color azul en el pin 6
const int ledVerde = 11;    // LED verde conectado al pin 11
const int ledNaranja = 12;  // LED naranja conectado al pin 12
const int ledRojo = 13;     // LED rojo conectado al pin 13

const float inclinacionPoca = 0.2;  // inclinada (en g)
const int espera = 200;             // tiempo entre un paso y el siguiente, en ms

Adafruit_LIS3DH acelerometro;  // el acelerómetro de la placa

float inclinacion = 0.0;  // guarda la aceleración del eje Y, en g
int posicion = 1;         // LED encendido: 1 verde, 2 naranja, 3 rojo, 4 azul

void setup() {
  pinMode(rgbAzul, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  pinMode(ledNaranja, OUTPUT);
  pinMode(ledRojo, OUTPUT);
  acelerometro.begin(0x18);
}

void loop() {
  acelerometro.read();
  inclinacion = acelerometro.y_g;

  // la luz avanza hacia el lado inclinado
  if (inclinacion < -inclinacionPoca) {
    posicion = posicion - 1;
  } else if (inclinacion > inclinacionPoca) {
    posicion = posicion + 1;
  }

  // al pasar de un extremo, vuelve por el otro
  if (posicion > 4) {
    posicion = 1;
  }
  if (posicion < 1) {
    posicion = 4;
  }
```

```
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

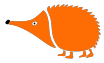
Posible solución:

```ino
// Brillo desde el teclado: un número terminado en > da el brillo

const int rgbRojo = 9;   // LED RGB: color rojo en el pin 9
const int rgbVerde = 5;  // LED RGB: color verde en el pin 5
const int rgbAzul = 6;   // LED RGB: color azul en el pin 6

char letra = ' ';  // guarda cada carácter que llega del ordenador
int numero = 0;    // va juntando las cifras del número
int brillo = 0;    // guarda el brillo del LED (de 0 a 255)

void setup() {
  pinMode(rgbRojo, OUTPUT);
  pinMode(rgbVerde, OUTPUT);
  pinMode(rgbAzul, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    letra = Serial.read();

    if (letra >= '0' && letra <= '9') {
      // una cifra: la añade al final del número
      numero = numero * 10 + (letra - '0');
      if (numero > 255) {
        numero = 255;  // como máximo, 255
      }
    } else if (letra == '>') {
      // fin del número: lo usa como brillo
      brillo = numero;
      Serial.print("Brillo: ");
      Serial.println(brillo);
      numero = 0;  // listo para el siguiente número
    }
  }

  // enciende el LED en blanco con el brillo actual
  analogWrite(rgbRojo, brillo);
  analogWrite(rgbVerde, brillo);
  analogWrite(rgbAzul, brillo);
}
```

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:



```
// Color a la carta: 254r109g4b pone el LED RGB naranja Echidna

const int rgbRojo = 9;   // LED RGB: color rojo en el pin 9
const int rgbVerde = 5;  // LED RGB: color verde en el pin 5
const int rgbAzul = 6;   // LED RGB: color azul en el pin 6

char letra = ' ';  // guarda cada carácter que llega del ordenador
int numero = 0;    // va juntando las cifras del número
int rojo = 0;      // intensidad del rojo (de 0 a 255)
int verde = 0;     // intensidad del verde (de 0 a 255)
int azul = 0;      // intensidad del azul (de 0 a 255)

void setup() {
  pinMode(rgbRojo, OUTPUT);
  pinMode(rgbVerde, OUTPUT);
  pinMode(rgbAzul, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    letra = Serial.read();

    if (letra >= '0' && letra <= '9') {
      // una cifra: la añade al final del número
      numero = numero * 10 + (letra - '0');
      if (numero > 255) {
        numero = 255;  // como máximo, 255
      }
    } else if (letra == 'r') {
      rojo = numero;  // el número es la intensidad del rojo
      Serial.print("Rojo: ");
      Serial.println(rojo);
      numero = 0;
    } else if (letra == 'g') {
      verde = numero;  // el número es la intensidad del verde
      Serial.print("Verde: ");
      Serial.println(verde);
      numero = 0;
    } else if (letra == 'b') {
      azul = numero;  // el número es la intensidad del azul
      Serial.print("Azul: ");
      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.