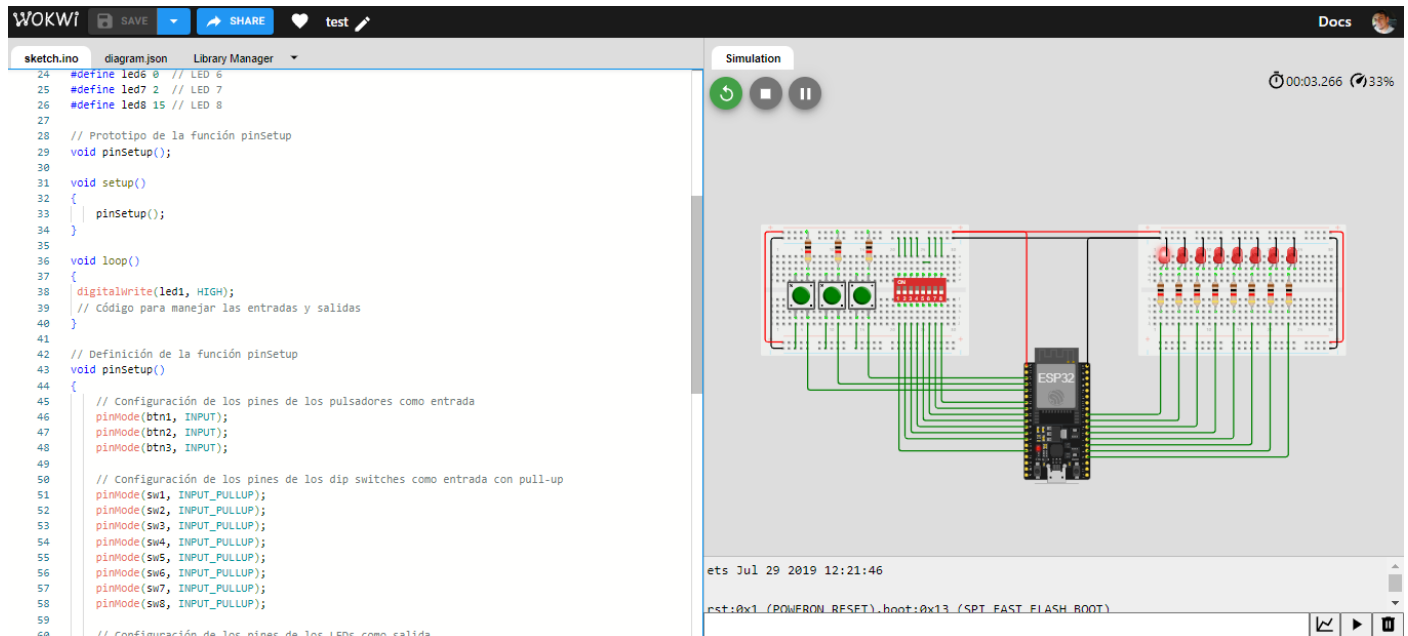


## Ejercicio 1: Encender un LED

- Enciende el led1 conectado al GPIO18 de forma continua.



## Código:

```

void loop()
{
  digitalWrite(led1, HIGH);
  // Código para manejar las entradas y salidas
}

```

## Ejercicio 2: Parpadeo de un LED

- Programa el led1 para que parpadee con un intervalo de 1 segundo.

**WOKWI** SAVE SHARE test Docs

sketch.ino • diagram.json Library Manager

```

19 #define led1 18 // LED 1
20 #define led2 5 // LED 2
21 #define led3 17 // LED 3
22 #define led4 16 // LED 4
23 #define led5 4 // LED 5
24 #define led6 0 // LED 6
25 #define led7 2 // LED 7
26 #define led8 15 // LED 8
27
28 // Prototipo de la función pinSetup
29 void pinSetup();
30
31 void setup()
32 {
33     pinSetup();
34 }
35
36 void loop()
37 {
38     digitalWrite(led1, HIGH);
39     delay(1000);
40     digitalWrite(led1, LOW);
41     delay(1000);
42     // Código para manejar las entradas y salidas
43 }
44
45 // Definición de la función pinSetup
46 void pinSetup()
47 {
48     // Configuración de los pines de los pulsadores como entrada
49     pinMode(btn1, INPUT);
50     pinMode(btn2, INPUT);
51     pinMode(btn3, INPUT);
52
53     // Configuración de los pines de los dip switches como entrada con pull-up
54     pinMode(swi1, INPUT_PULLUP);

```

Simulation

00:08.678 59%

clk\_drv:0x00,q\_drv:0x00,d\_drv:0x00,cs0\_drv:0x00,hd\_drv:0x00,wp\_drv:0x00  
mode:DIO, clock div:2  
load:0x3fff0030,len:1156  
load:0x40078000,len:11456  
ho 0 tail 12 room 4  
load:0x40080400,len:2972  
entry 0x400805dc

## Codigo:

```

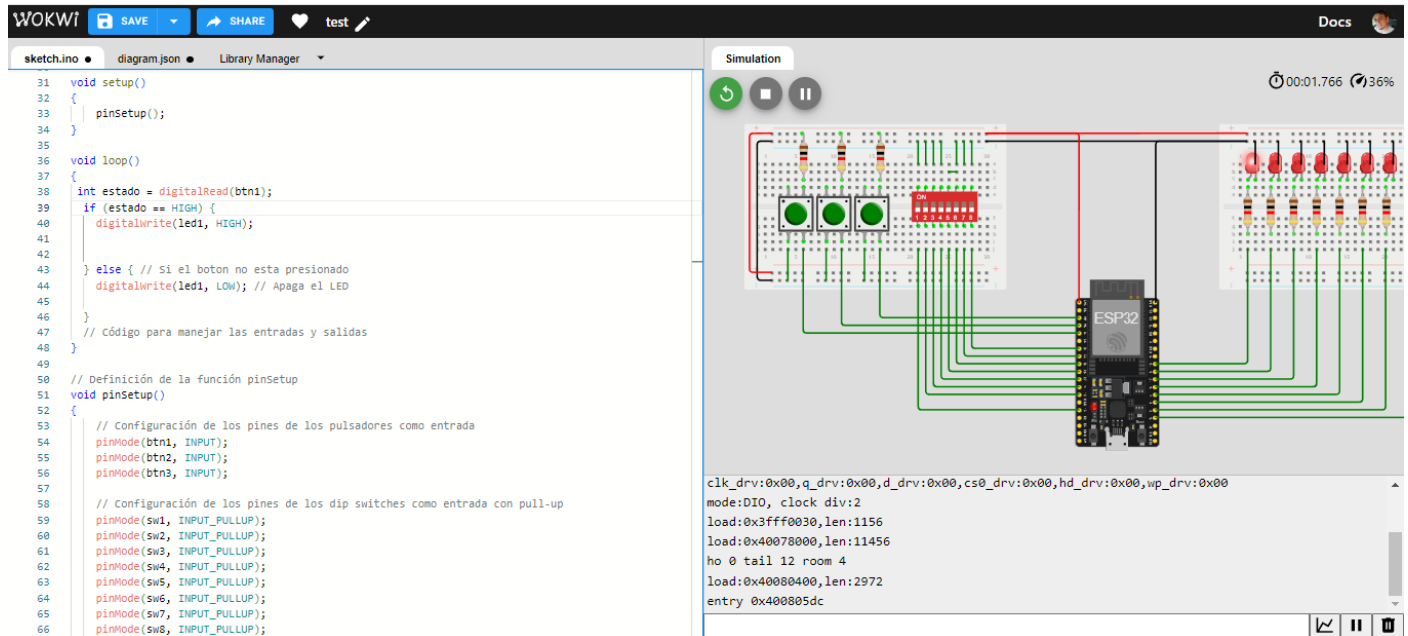
36 void loop()
37 {
38     digitalWrite(led1, HIGH);
39     delay(1000);
40     digitalWrite(led1, LOW);
41     delay(1000);
42     // Código para manejar las entradas y salidas
43 }
--

```



## Ejercicio 4: Control de LED con botón

- Usa el btn1 para encender el led1 mientras se mantenga presionado.



## Código:

```

37 void loop()
38 {
39   int estado = digitalRead(btn1);
40   if (estado == LOW) {
41     digitalWrite(led1, HIGH);
42   }
43   else { // Si el boton no esta presionado
44     digitalWrite(led1, LOW); // Apaga el LED
45   }
46   // Código para manejar las entradas y salidas
47 }
48
49 }
50

```

## Ejercicio 5: Uso de botón con estado

- Cambia el estado del led1 cada vez que se presione y suelte el btn1.

**WOKWI** SAVE SHARE test Docs

sketch.ino • diagram.json • Library Manager

```

37
38 void setup()
39 {
40   pinSetup();
41 }
42
43
44 void loop()
45 {
46   int estadoBtn1 = digitalRead(btn1); // controla el estado del boton
47   estadoBtn1 = digitalRead(btn1);
48
49   if (estadoBtn1 != anteriorestadoBtn1 && estadoBtn1 == HIGH){
50     // si el estado del boton es diferente a estado anterior
51     //y el estado del boton es igual a 1
52     estadoled1 = !estadoled1; // guarda el estado si es diferente al estado anterior
53     digitalWrite(led1, estadoled1);
54   }
55   anteriorestadoBtn1 = estadoBtn1; // Guardar el estado actual del boton para la próximo estado
56   // Código para manejar las entradas y salidas
57 }
58
59 // Definición de la función pinSetup
60 void pinSetup()
61 {
62   // Configuración de los pines de los pulsadores como entrada
63   pinMode(btn1, INPUT);
64   pinMode(btn2, INPUT);
65   pinMode(btn3, INPUT);
66
67   // Configuración de los pines de los dip switches como entrada con pull-up
68   pinMode(sw1, INPUT_PULLUP);
69   pinMode(sw2, INPUT_PULLUP);
70   pinMode(sw3, INPUT_PULLUP);
71   pinMode(sw4, INPUT_PULLUP);
72   pinMode(sw5, INPUT_PULLUP);
73   pinMode(sw6, INPUT_PULLUP);
74   pinMode(sw7, INPUT_PULLUP);
75   pinMode(sw8, INPUT_PULLUP);
76   pinMode(sw9, INPUT_PULLUP);
77   pinMode(sw10, INPUT_PULLUP);
78 }

```

**Simulation**

00:03.866 41%

clk\_drv:0x00,q\_drv:0x00,d\_drv:0x00,cs0\_drv:0x00,hd\_drv:0x00,wp\_drv:0x00  
mode:DIO, clock div:2  
load:0x3fff0030,len:1156  
load:0x40078000,len:11456  
ho 0 tail 12 room 4  
load:0x40080400,len:2972  
entry 0x400805dc

## Codigo:

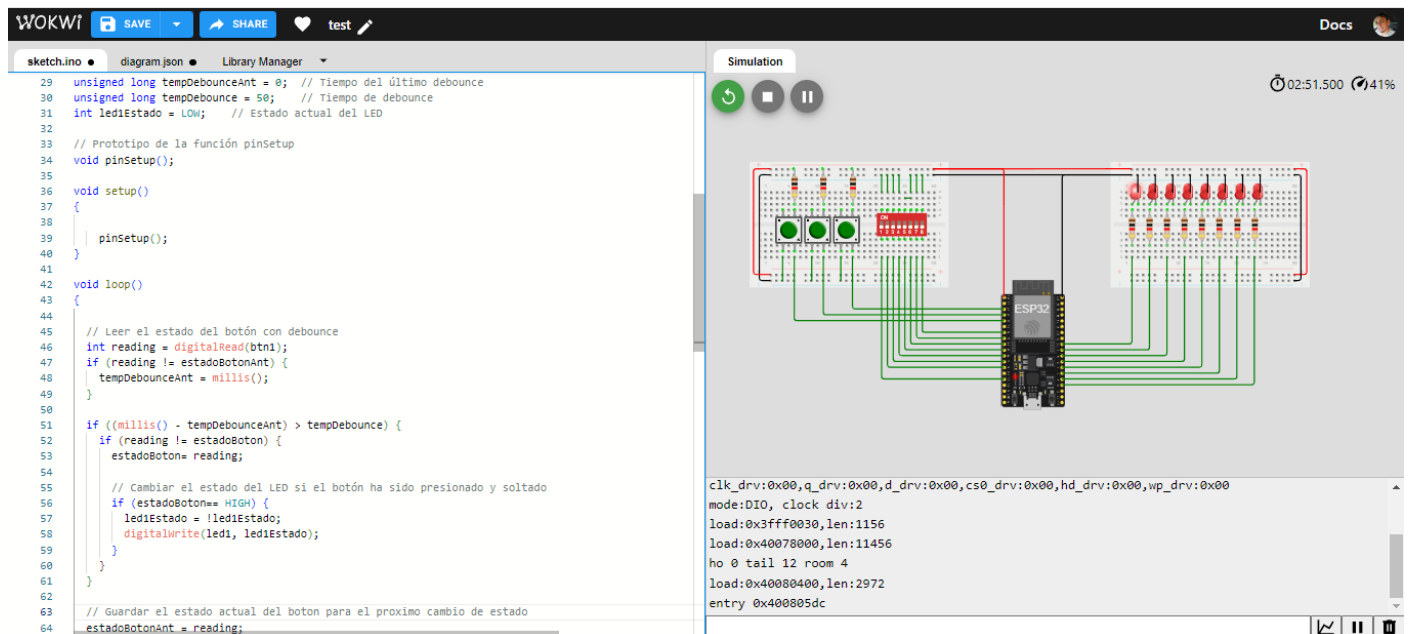
```

44 void loop()
45 {
46   int estadoBtn1 = digitalRead(btn1); // controla el estado del boton
47   estadoBtn1 = digitalRead(btn1);
48
49   if (estadoBtn1 != anteriorestadoBtn1 && estadoBtn1 == HIGH){
50     // si el estado del boton es diferente a estado anterior
51     //y el estado del boton es igual a 1
52     estadoled1 = !estadoled1; // guarda el estado si es diferente al estado anterior
53     digitalWrite(led1, estadoled1);
54   }
55   anteriorestadoBtn1 = estadoBtn1; // Guardar el estado actual del boton para la próximo estado
56   // Código para manejar las entradas y salidas
57 }

```

## Ejercicio 6: Debounce de botón

- Implementa una lógica de debounce en el btn1 para evitar lecturas erróneas.



### Codigo:

```

27 int estadoBoton = 0; // Estado actual del boton
28 int estadoBotonAnt = LOW; // Estado anterior del boton
29 unsigned long tempDebounceAnt = 0; // Tiempo del último debounce
30 unsigned long tempDebounce = 50; // Tiempo de debounce
31 int led1Estado = LOW; // Estado actual del LED

42 void loop()
43 {
44   // Leer el estado del botón con debounce
45   int lecturaBoton = digitalRead(btn1);
46   if (lecturaBoton != estadoBotonAnt) {
47     tempDebounceAnt = millis();
48   }
49
50   if ((millis() - tempDebounceAnt) > tempDebounce) {
51     if (lecturaBoton != estadoBoton) {
52       estadoBoton = lecturaBoton;
53
54       // Cambiar el estado del LED si el botón ha sido presionado y soltado
55       if (estadoBoton == HIGH) {
56         led1Estado = !led1Estado;
57         digitalWrite(led1, led1Estado);
58       }
59     }
60   }
61
62   // Guardar el estado actual del boton para el proximo cambio de estado
63   estadoBotonAnt = lecturaBoton;
64   // Código para manejar las entradas y salidas
65 }

```



solucion por software.

El debounce es una técnica utilizada en ingeniería de interfaces para eliminar las falsas pulsaciones que se producen en los dispositivos electrónicos, especialmente en los interruptores y botones. En Arduino, el proceso de eliminación de este rebote se llama “debounce”.

También se puede solucionar por hardware.

Para aplicar un debounce por hardware, se puede colocar un condensador en paralelo con el dispositivo. Un condensador de 1uF debería ser suficiente para filtrar la mayoría del ruido. Cuando se cierra el contacto, el condensador se descarga, y cuando se abre, absorbe el voltaje en el tiempo que se carga, eliminando los rebotes

## Ejercicio 7: Control de múltiples LEDs con botones

- Usa btn1 y btn2 para controlar el estado de led1 y led2 respectivamente.

WOKWI

SAVE

SHARE

test

sketch.ino

diagram.json

Library Manager

```

30 void pinSetup();
31
32 void setup()
33 {
34   pinSetup();
35 }
36
37
38 void loop()
39 {
40   if (digitalRead(btn1) == LOW){
41     digitalWrite(led1, HIGH);
42   }
43   else
44   {
45     digitalWrite(led1, LOW);
46     delay(50);
47   }
48   if (digitalRead(btn2) == LOW){
49     digitalWrite(led2, HIGH);
50   }
51   else
52   {
53     digitalWrite(led2, LOW);
54     delay(50);
55   }
56
57   // Código para manejar las entradas y salidas
58 }
59
60 // Definición de la función pinSetup
61 void pinSetup()
62 {
63   // Configuración de los pines de los pulsadores como entrada
64   pinMode(btn1, INPUT);
65   pinMode(btn2, INPUT);

```

Simulation

00:06.145

95%

```

clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc

```

## Codigo:

```

38 void loop()
39 {
40   if (digitalRead(btn1) == LOW){
41     digitalWrite(led1, HIGH);
42   }
43   else
44   {
45     digitalWrite(led1, LOW);
46     delay(50);
47   }
48   if (digitalRead(btn2) == LOW){
49     digitalWrite(led2, HIGH);
50   }
51   else
52   {
53     digitalWrite(led2, LOW);
54     delay(50);
55   }
56
57   // Código para manejar las entradas y salidas
58 }
59
60 // Definición de la función pinSetup

```

## Ejercicio 8: Uso de dip switches para control de LEDs

- Lee el estado de los dip switches sw1.1 a sw1.8 y refleja el estado en los led1 a led8.

WOKWI

SAVE

SHARE

test

docs

test

sketch.ino

diagram.json

Library Manager

```

33 {
34 }
35
36 void loop()
37 {
38   if (digitalRead(sw1) == LOW){
39     digitalWrite(led1, HIGH);
40   }
41   else
42   {
43     digitalWrite(led1, LOW);
44     delay(50);
45   }
46   if (digitalRead(sw2) == LOW){
47     digitalWrite(led2, HIGH);
48   }
49   else
50   {
51     digitalWrite(led2, LOW);
52     delay(50);
53   }
54   if (digitalRead(sw3) == LOW){
55     digitalWrite(led3, HIGH);
56   }
57   else
58   {
59     digitalWrite(led3, LOW);
60     delay(50);
61   }
62   if (digitalRead(sw4) == LOW){
63     digitalWrite(led4, HIGH);
64   }
65   else
66   {
67     digitalWrite(led4, LOW);
68     delay(50);
69   }
70 }
71

```

Simulation

00:11.794

28%

clk\_drv:0x00,q\_drv:0x00,d\_drv:0x00,cs0\_drv:0x00,hd\_drv:0x00,wp\_drv:0x00

mode:DIO, clock div:2

load:0x3fff0030,len:1156

load:0x40078000,len:11456

ho 0 tail 12 room 4

load:0x40080400,len:2972

entry 0x400805dc



## Codigo:

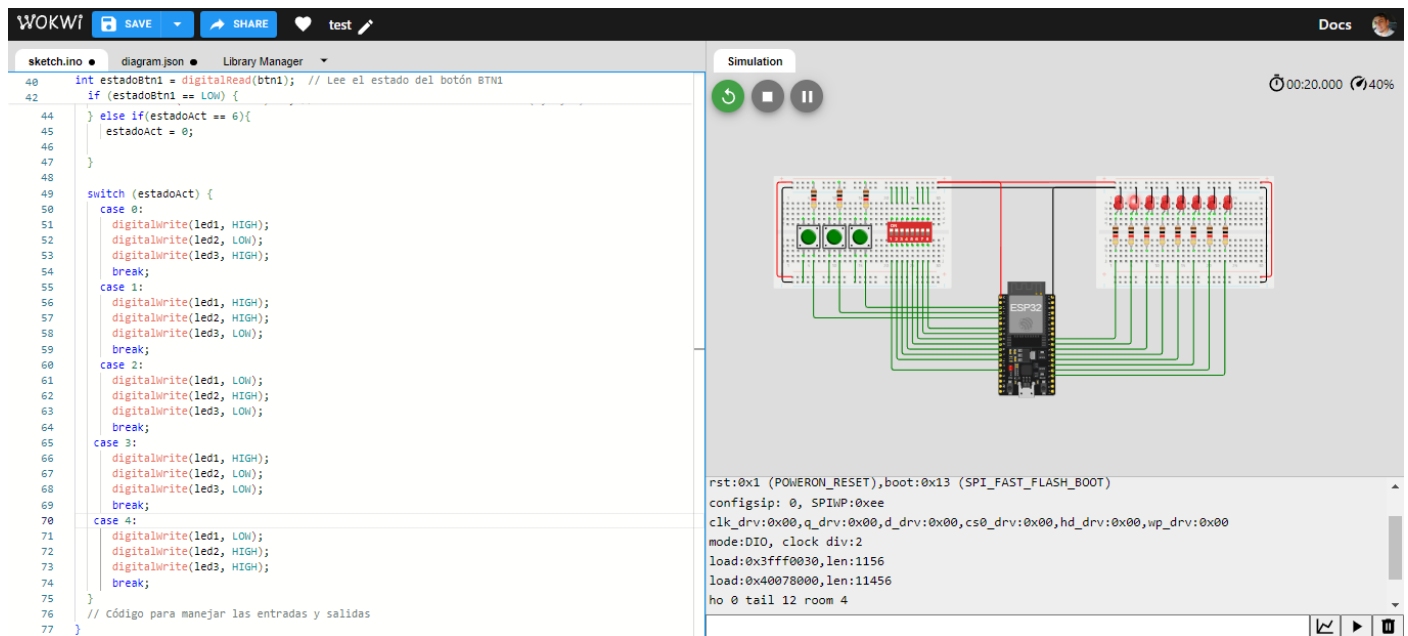
```

38 void loop()
39 {
40     if (digitalRead(sw1) == LOW){
41         digitalWrite(led1, HIGH);
42     }
43     else
44     {
45         digitalWrite(led1, LOW);
46         delay(50);
47     }
48     if (digitalRead(sw2) == LOW){
49         digitalWrite(led2, HIGH);
50     }
51     else
52     {
53         digitalWrite(led2, LOW);
54         delay(50);
55     }
56     if (digitalRead(sw3) == LOW){
57         digitalWrite(led3, HIGH);
58     }
59     else
60     {
61         digitalWrite(led3, LOW);
62         delay(50);
63     }
64     if (digitalRead(sw4) == LOW){
65         digitalWrite(led4, HIGH);
66     }
67     else
68     {
69         digitalWrite(led4, LOW);
70         delay(50);
71     }
72     if (digitalRead(sw5) == LOW){
73         digitalWrite(led5, HIGH);
74     }
75     else
76     {
77         digitalWrite(led5, LOW);
78         delay(50);
79     }
80     if (digitalRead(sw6) == LOW){
81         digitalWrite(led6, HIGH);
82     }
83     else
84     {
85         digitalWrite(led6, LOW);
86         delay(50);
87     }
88     if (digitalRead(sw7) == LOW){
89         digitalWrite(led7, HIGH);
90     }
91     else
92     {
93         digitalWrite(led7, LOW);
94         delay(50);
95     }
96     if (digitalRead(sw8) == LOW){
97         digitalWrite(led8, HIGH);
98     }
99     else
100    {
101        digitalWrite(led8, LOW);
102        delay(50);
103    }
104    // Código para manejar las entradas y salidas
105
106 }

```

## Ejercicio 9: Secuencia de LEDs con botón

- Crea una secuencia de luces que avance cada vez que se presione btn1.



### Código:

```

39 {
40     int estadoBtn1 = digitalRead(btn1); // Lee el estado del botón BTN1
41
42     if (estadoBtn1 == LOW) {
43         estadoAct = (estadoAct + 1) % 4; // Avanza el estado de la secuencia (0, 1, 2)
44     } else if (estadoAct == 6) {
45         estadoAct = 0;
46     }
47 }
48
49 switch (estadoAct) {
50     case 0:
51         digitalWrite(led1, HIGH);
52         digitalWrite(led2, LOW);
53         digitalWrite(led3, HIGH);
54         break;
55     case 1:
56         digitalWrite(led1, HIGH);
57         digitalWrite(led2, HIGH);
58         digitalWrite(led3, LOW);
59         break;
60     case 2:
61         digitalWrite(led1, LOW);
62         digitalWrite(led2, HIGH);
63         digitalWrite(led3, LOW);
64         break;
65     case 3:
66         digitalWrite(led1, HIGH);
67         digitalWrite(led2, LOW);
68         digitalWrite(led3, LOW);
69         break;
70     case 4:
71         digitalWrite(led1, LOW);
72         digitalWrite(led2, HIGH);
73         digitalWrite(led3, HIGH);
74         break;
75 }
    
```

## Investigación:

$\text{estadoAct} = (\text{estadoAct} + 1) \% 4$ ; Esto es una operación que se ejecuta si la condición en la línea anterior es verdadera.  $\text{estadoAct}$  parece ser una variable que representa el estado actual en algún tipo de secuencia (probablemente la secuencia de luces en este caso). Esta línea de código incrementa  $\text{estadoAct}$  en 1, y luego toma el resto de la división de  $\text{estadoAct}$  entre 4.

- Por ejemplo, si  $\text{estadoAct}$  es 0, después de ejecutar esta línea, se convierte en 1.
- Si  $\text{estadoAct}$  es 1, se convierte en 2.
- Si  $\text{estadoAct}$  es 2, se convierte en 3.
- Si  $\text{estadoAct}$  es 3, se convierte en 0 (debido al operador  $\%$  4).

## Ejercicio 10: Control de velocidad de parpadeo con dip switch

- Utiliza los dip switches sw1.1 a sw1.3 para controlar la velocidad de parpadeo de led1, asignando distintas velocidades.

The screenshot shows the WOKWI simulation environment. On the left, the Arduino IDE interface displays the following code:

```

34 {
35 }
36
37 }
38
39 void loop()
40 {
41   int estadoSw1 = digitalRead(sw1);
42   int estadoSw2 = digitalRead(sw2);
43   int estadoSw3 = digitalRead(sw3);
44
45
46   if (estadoSw1 == HIGH) {
47     velDelay = 100;
48   }
49   if (estadoSw2 == HIGH) {
50     velDelay = 500;
51   }
52   if (estadoSw3 == HIGH) {
53     velDelay = 1000;
54   }
55   digitalWrite(led1, HIGH);
56   delay(velDelay / 2);
57   digitalWrite(led1, LOW);
58   delay(velDelay / 2);
59 }
60
61 // Definición de la función pinSetup
62 void pinSetup()
63 {
64   // Configuración de los pines de los pulsadores como entrada
65   pinMode(btn1, INPUT);
66   pinMode(btn2, INPUT);
67   pinMode(btn3, INPUT);
68
69   // Configuración de los pines de los dip switches como entrada con pull-up
70   pinMode(sw1, INPUT_PULLUP);
71   pinMode(sw2, INPUT_PULLUP);
72   pinMode(sw3, INPUT_PULLUP);
73 }

```

On the right, the simulation window shows a breadboard circuit. It includes an ESP32 microcontroller, three push buttons (btn1, btn2, btn3), and three dip switches (sw1, sw2, sw3). The circuit is connected to a breadboard with various components like resistors and jumper wires. The simulation status bar at the top right shows a timer at 00:41.283 and a battery level at 84%.



## Código:

```

42 void loop()
43 {
44   int estadoSw1 = digitalRead(sw1); // Lee el estado del sw1
45   int estadoSw2 = digitalRead(sw2); // Lee el estado del sw2
46   int estadoSw3 = digitalRead(sw3); // Lee el estado del sw3
47   int estadoSw4 = digitalRead(sw4); // Lee el estado del sw4
48
49   if (estadoSw1 == HIGH && estadoSw2 == LOW && estadoSw3 == LOW && estadoSw4 == LOW) {
50     estadoAct = 1; }
51   else if (estadoSw1 == LOW && estadoSw2 == HIGH && estadoSw3 == LOW && estadoSw4 == LOW) {
52     estadoAct = 2; }
53   else if (estadoSw1 == LOW && estadoSw2 == LOW && estadoSw3 == HIGH && estadoSw4 == LOW) {
54     estadoAct = 3; }
55   else if (estadoSw1 == LOW && estadoSw2 == LOW && estadoSw3 == LOW && estadoSw4 == HIGH) {
56     estadoAct = 4; }
57   else if (estadoSw1 == HIGH && estadoSw2 == HIGH && estadoSw3 == LOW && estadoSw4 == LOW) {
58     estadoAct = 5; }
59   else if (estadoSw1 == HIGH && estadoSw2 == HIGH && estadoSw3 == HIGH && estadoSw4 == LOW) {
60     estadoAct = 6; }
61   else if (estadoSw1 == HIGH && estadoSw2 == HIGH && estadoSw3 == HIGH && estadoSw4 == HIGH) {
62     estadoAct = 7; }
63   else if (estadoSw1 == LOW && estadoSw2 == HIGH && estadoSw3 == LOW && estadoSw4 == HIGH) {
64     estadoAct = 8; }
65   else if (estadoSw1 == HIGH && estadoSw2 == HIGH && estadoSw3 == HIGH && estadoSw4 == HIGH) {
66     estadoAct = 9; }
67   else if (estadoSw1 == LOW && estadoSw2 == HIGH && estadoSw3 == LOW && estadoSw4 == LOW) {
68     estadoAct = 10; }
69   else if (estadoSw1 == LOW && estadoSw2 == HIGH && estadoSw3 == HIGH && estadoSw4 == LOW) {
70     estadoAct = 11; }
71   else if (estadoSw1 == HIGH && estadoSw2 == HIGH && estadoSw3 == LOW && estadoSw4 == HIGH) {
72     estadoAct = 12; }
73   else {
74     estadoAct = 0; // Ningún botón presionado o combinación no válida
75   }
76

```



```

switch (estadoAct) {
  case 0: // Enciende LED1, 107 ✓
    digitalWrite(led1, LOW); 108
    digitalWrite(led2, HIGH); 109
    digitalWrite(led3, LOW); 110
    digitalWrite(led4, LOW); 111
    digitalWrite(led5, HIGH); 112
    digitalWrite(led6, LOW); 113
    digitalWrite(led7, LOW); 114
    digitalWrite(led8, LOW); 115
    break; 116
  case 1: 117 ✓
    digitalWrite(led1, HIGH); 118
    digitalWrite(led2, LOW); 119
    digitalWrite(led3, HIGH); 120
    digitalWrite(led4, LOW); 121
    digitalWrite(led5, LOW); 122
    digitalWrite(led6, LOW); 123
    digitalWrite(led7, HIGH); 124
    digitalWrite(led8, LOW); 125
    break; 126
  case 2: 127 ✓
    digitalWrite(led1, HIGH); 128
    digitalWrite(led2, LOW); 129
    digitalWrite(led3, HIGH); 130
    digitalWrite(led4, LOW); 131
    digitalWrite(led5, HIGH); 132
    digitalWrite(led6, LOW); 133
    digitalWrite(led7, LOW); 134
    digitalWrite(led8, LOW); 135
    break; 136
  case 3: 137
    digitalWrite(led1, HIGH); 138
    digitalWrite(led2, LOW); 139
    digitalWrite(led3, LOW); 140
    digitalWrite(led4, LOW); 141
    digitalWrite(led5, LOW); 142
    digitalWrite(led6, LOW); 143
    digitalWrite(led7, HIGH); 144
    digitalWrite(led8, HIGH); 145
    break; 146
  case 4: 147
    digitalWrite(led1, HIGH); 148
    digitalWrite(led2, LOW); 149
    digitalWrite(led3, LOW); 150
    digitalWrite(led4, HIGH); 151
    digitalWrite(led5, LOW); 152
    digitalWrite(led6, HIGH); 153
    digitalWrite(led7, LOW); 154
    digitalWrite(led8, HIGH); 155
    break; 156
  case 5: 157
    digitalWrite(led1, HIGH); 158
    digitalWrite(led2, HIGH); 159
    digitalWrite(led3, LOW); 160
    digitalWrite(led4, HIGH); 161
    digitalWrite(led5, LOW); 162
    digitalWrite(led6, HIGH); 163
    digitalWrite(led7, LOW); 164
    digitalWrite(led8, HIGH); 165
    break; 166
  case 6: 167
    digitalWrite(led1, HIGH); 168
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    digitalWrite(led5, LOW);
    digitalWrite(led6, HIGH);
    digitalWrite(led7, LOW);
    digitalWrite(led8, HIGH);
    break;
  case 7:
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    digitalWrite(led5, HIGH);
    digitalWrite(led6, HIGH);
    digitalWrite(led7, LOW);
    digitalWrite(led8, HIGH);
    break;
  case 8:
    digitalWrite(led1, HIGH);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, HIGH);
    digitalWrite(led5, HIGH);
    digitalWrite(led6, HIGH);
    digitalWrite(led7, HIGH);
    digitalWrite(led8, HIGH);
    break;
}

```



## Ejercicio 12: Medidor de pulsaciones

- Programa un contador de pulsaciones utilizando btn1. El número de pulsaciones debe mostrarse en una secuencia de LEDs (por ejemplo, led5 a led8 donde cada LED representa una cantidad de pulsaciones).

**WOKWI** SAVE SHARE test Docs

sketch.ino • diagram.json • Library Manager

```

38
39 void loop()
40 {
41   int estadoBtn1 = digitalRead(btn1);
42
43   if (estadoBtn1 == HIGH) {
44     estadoAct = (estadoAct + 1) % 10;
45   }
46
47   switch (estadoAct) {
48     case 0:
49     digitalWrite(led5, LOW);
50     digitalWrite(led6, LOW);
51     digitalWrite(led7, LOW);
52     digitalWrite(led8, LOW);
53     break;
54     case 1:
55     digitalWrite(led5, HIGH);
56     digitalWrite(led6, LOW);
57     digitalWrite(led7, LOW);
58     digitalWrite(led8, LOW);
59     break;
60     case 2:
61     digitalWrite(led5, LOW);
62     digitalWrite(led6, HIGH);
63     digitalWrite(led7, LOW);
64     digitalWrite(led8, LOW);
65     break;
66     case 3:
67     digitalWrite(led5, LOW);
68     digitalWrite(led6, LOW);
69     digitalWrite(led7, HIGH);
70     digitalWrite(led8, LOW);
71     break;
72     case 4:
73     digitalWrite(led5, LOW);
74     digitalWrite(led6, LOW);
75     digitalWrite(led7, LOW);
76     digitalWrite(led8, HIGH);
77     break;
78   }
79 }

```

<https://wokwi.com>

**Simulation** 01:45.066 31%

clk\_drv:0x00,q\_drv:0x00,d\_drv:0x00,cs0\_drv:0x00,hd\_drv:0x00,wp\_drv:0x00  
mode:DIO, clock div:2  
load:0x3fff0030,len:1156  
load:0x40078000,len:11456  
ho 0 tail 12 room 4  
load:0x40080400,len:2972  
entry 0x400805dc

## Codigo:

```
int estadoBtn1 = digitalRead(btn1);

if (estadoBtn1 == HIGH) {
    estadoAct = (estadoAct + 1) % 10;
}

switch (estadoAct) {
    case 0:
        digitalWrite(led5, LOW);
        digitalWrite(led6, LOW);
        digitalWrite(led7, LOW);
        digitalWrite(led8, LOW);
        break;
    case 1:
        digitalWrite(led5, HIGH);
        digitalWrite(led6, LOW);
        digitalWrite(led7, LOW);
        digitalWrite(led8, LOW);
        break;
    case 2:
        digitalWrite(led5, LOW);
        digitalWrite(led6, HIGH);
        digitalWrite(led7, LOW);
        digitalWrite(led8, LOW);
        break;
    case 3:
        digitalWrite(led5, LOW);
        digitalWrite(led6, LOW);
        digitalWrite(led7, HIGH);
        digitalWrite(led8, LOW);
        break;
    case 4:
        digitalWrite(led5, LOW);
        digitalWrite(led6, LOW);
        digitalWrite(led7, HIGH);
        digitalWrite(led8, LOW);
        break;
    case 5:
        digitalWrite(led5, LOW);
        digitalWrite(led6, LOW);
        digitalWrite(led7, LOW);
        digitalWrite(led8, HIGH);
        break;
    case 6:
        digitalWrite(led5, HIGH);
        digitalWrite(led6, HIGH);
        digitalWrite(led7, LOW);
        digitalWrite(led8, LOW);
        break;
    case 7:
        digitalWrite(led5, HIGH);
        digitalWrite(led6, LOW);
        digitalWrite(led7, HIGH);
        digitalWrite(led8, LOW);
        break;
    case 8:
        digitalWrite(led5, HIGH);
        digitalWrite(led6, LOW);
        digitalWrite(led7, LOW);
        digitalWrite(led8, HIGH);
        break;
}
```

## Ejercicio 13: Contraseña con botones

- Implementa un sistema de contraseña usando btn1, btn2, y btn3 donde una secuencia específica de pulsaciones activa led1. Si la secuencia es incorrecta, led2 debería encenderse.

WOKWI

SAVE

SHARE

test

Docs

sketch.ino

diagram.json

Library Manager

```

42 void loop()
43 {
44
45     int estadoBtn1 = digitalRead(btn1);
46     int estadoBtn2 = digitalRead(btn2);
47     int estadoBtn3 = digitalRead(btn3);
48
49     if (estadoBtn1 == LOW) { // Si el botón está presionado
50         Serial.print("Valor del btn1: ");
51         Serial.println(Contbtn1);
52         Contbtn1++; // Incrementar el contador de pulsaciones
53     }
54     if (estadoBtn2 == LOW) { // Si el botón está presionado
55         Serial.print("Valor del btn2: ");
56         Serial.println(Contbtn2);
57         Contbtn2++; // Incrementar el contador de pulsaciones
58     }
59     if (estadoBtn3 == LOW) { // Si el botón está presionado
60         Serial.print("Valor del btn3: ");
61         Serial.println(Contbtn3);
62         Contbtn3++; // Incrementar el contador de pulsaciones
63     }
64
65     if (Contbtn1 == 7 && Contbtn2 == 7 && Contbtn3 == 7) {
66         digitalWrite(led1, HIGH);
67         digitalWrite(led2, LOW);
68     } else {
69         digitalWrite(led1, LOW);
70         digitalWrite(led2, HIGH);
71     }
72 }
73
74 // Definición de la función pinSetup
75 void pinSetup()
76 {
77

```

Simulation

00:05.866 19%

Valor del btn2: 6

Valor del btn3: 0

Valor del btn3: 1

Valor del btn3: 2

Valor del btn3: 3

Valor del btn3: 4

Valor del btn3: 5

## Código:

```
42 void loop()
43 {
44
45     int estadoBtn1 = digitalRead(btn1);
46     int estadoBtn2 = digitalRead(btn2);
47     int estadoBtn3 = digitalRead(btn3);
48
49     if (estadoBtn1 == LOW) { // Si el botón está presionado
50         Serial.print("valor del btn1: ");
51         Serial.println(Contbtn1);
52         Contbtn1++; // Incrementar el contador de pulsaciones
53     }
54     if (estadoBtn2 == LOW) { // Si el botón está presionado
55         Serial.print("Valor del btn2: ");
56         Serial.println(Contbtn2);
57         Contbtn2++; // Incrementar el contador de pulsaciones
58     }
59     if (estadoBtn3 == LOW) { // Si el botón está presionado
60         Serial.print("Valor del btn3: ");
61         Serial.println(Contbtn3);
62         Contbtn3++; // Incrementar el contador de pulsaciones
63     }
64
65     if (Contbtn1 == 7 && Contbtn2 == 7 && Contbtn3 == 7) {
66         digitalWrite(led1, HIGH);
67         digitalWrite(led2, LOW);
68     } else{
69         digitalWrite(led1, LOW);
70         digitalWrite(led2, HIGH);
71     }
72 }
73 }
```

## Ejercicio 14: Aplicación de timers para control de LEDs

- Utiliza el temporizador del ESP32 para controlar el parpadeo de led1 a led4 sin usar la función delay(), permitiendo que el programa ejecute otras tareas mientras los LEDs parpadean.

WOKWI

SAVE

SHARE

test

Docs

sketch.ino

diagram.json

Library Manager

```

27 int estadoBtn1 = 0;
28 int estadoBtn2 = 0;
29 int Contbtn1 = 0;
30 int Contbtn2 = 0;
31 int Contbtn3 = 0;
32 bool ha_expirado = false; // Variable para indicar si la alarma se ha disparado
33 hw_timer_t * temporizador = NULL; // Puntero al temporizador
34 // Prototipo de la función pinSetup
35 void pinSetup();
36
37 // Función de interrupción del temporizador
38 void IRAM_ATTR interrupcionTemporizador() {
39     ha_expirado = true; // Indica que la alarma se ha disparado
40     timerAlarmDisable(temporizador); // Deshabilita la alarma del temporizador
41 }
42
43 // Configuración del temporizador
44 void configurarTemporizador() {
45     temporizador = timerBegin(0, 80, true); // Inicia el temporizador 0 con un divisor de 80 (valor pr
46     timerAttachInterrupt(temporizador, &interrupcionTemporizador, true); // Adjunta la función de inte
47     timerAlarmWrite(temporizador, 1000000, false); // Establece la alarma del temporizador en 1 segund
48     timerAlarmEnable(temporizador); // Habilita la alarma del temporizador
49 }
50
51 void setup()
52 {
53     configurarTemporizador(); // Configura el temporizador
54 }
55
56 void loop()
57 {
58     // Si la alarma se ha disparado, se alterna el estado del LED
59     if (ha_expirado) {
60         digitalWrite(led1, !digitalRead(led1)); // Alterna el estado del LED
61         ha_expirado = false; // Reinicia la variable
62     }
63 }
```

Simulation

00:52.733

28%

```

clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
```

## Código:

```
bool ha_expirado = false; // Variable para indicar si la alarma se ha disparado
hw_timer_t * temporizador = NULL; // Puntero al temporizador
// Prototipo de la función pinSetup
void pinSetup();

// Función de interrupción del temporizador
void IRAM_ATTR interrupcionTemporizador() {
    ha_expirado = true; // Indica que la alarma se ha disparado
    timerAlarmDisable(temporizador); // Deshabilita la alarma del temporizador
}
// Configuración del temporizador
void configurarTemporizador() {
    temporizador = timerBegin(0, 80, true); // Inicia el temporizador 0 con un divisor de 80
    (valor predeterminado) y habilita la interrupción de nivel superior
    timerAttachInterrupt(temporizador, &interrupcionTemporizador, true); // Adjunta la
    función de interrupción al temporizador y habilita la interrupción de nivel superior
    timerAlarmWrite(temporizador, 1000000, false); // Establece la alarma del temporizador
    en 1 segundo (en microsegundos) y no se repite
    timerAlarmEnable(temporizador); // Habilita la alarma del temporizador
}
void setup()
{
    configurarTemporizador(); // Configura el temporizador
}

void loop()
{
    // Si la alarma se ha disparado, se alterna el estado del LED
    if (ha_expirado) {
        digitalWrite(led1, !digitalRead(led1)); // Alterna el estado del LED
        ha_expirado = false; // Reinicia la variable
    }
}
```

## investigación:

Función configurarTemporizador():

- **temporizador = timerBegin(0, 80, true);**: Esta línea inicia el temporizador 0. El primer parámetro (0) indica el número de identificador del temporizador. El segundo parámetro (80) es el divisor del temporizador, que determina la frecuencia de la señal de salida del temporizador. En este caso, la frecuencia será de 80 Hz. El tercer parámetro (true) habilita la interrupción de nivel superior para el temporizador.

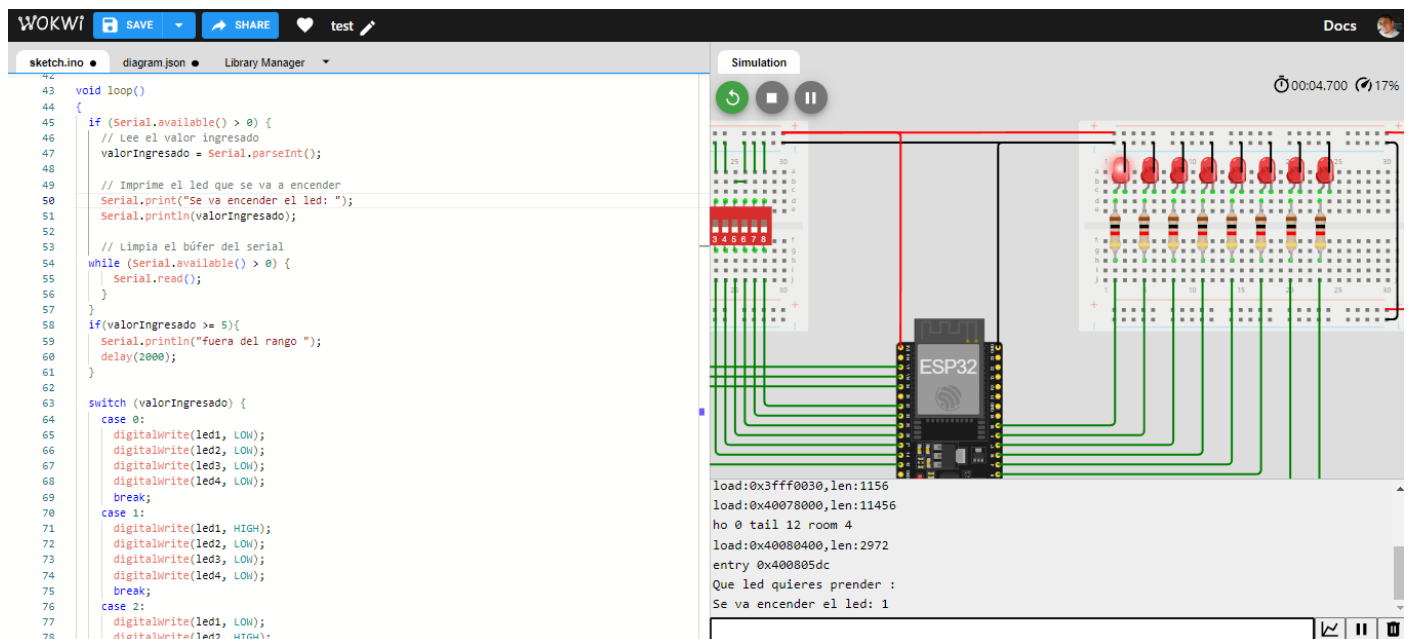
- **timerAttachInterrupt(temporizador, &interrupcionTemporizador, true);**: Esta línea asocia la función interrupcionTemporizador con el temporizador 0 y habilita la interrupción de nivel superior. Cuando la alarma del temporizador se dispara, se llamará a la función interrupcionTemporizador.
- **timerAlarmWrite(temporizador, 1000000, false);**: Esta línea configura la alarma del temporizador para que se dispare después de 1 segundo. El primer parámetro (temporizador) es el identificador del temporizador. El segundo parámetro (1000000) es el tiempo de espera de la alarma en microsegundos. El tercer parámetro (false) indica que la alarma no se repetirá una vez que se dispare.
- **timerAlarmEnable(temporizador);**: Esta línea habilita la alarma del temporizador. Una vez habilitada, la alarma se disparará después del tiempo de espera configurado.

## Función interrupcionTemporizador():

- **ha\_expirado = true;**: Esta línea establece la variable ha\_expirado en true. Esta variable se puede usar para indicar que la alarma del temporizador se ha disparado.
- **timerAlarmDisable(temporizador);**: Esta línea desactiva la alarma del temporizador. Esto evita que la alarma se dispare nuevamente.

## Ejercicio 15: Control de LEDs mediante comunicación serial

- Escribe un programa que reciba comandos a través del puerto serie para controlar los LEDs. Por ejemplo, enviar '1' podría encender led1, '2' apagar led2, etc.



The image shows the Wokwi IDE interface. On the left, the sketch.ino file contains the following code:

```

43 void loop()
44 {
45   if (Serial.available() > 0) {
46     // Lee el valor ingresado
47     valorIngresado = Serial.parseInt();
48
49     // Imprime el led que se va a encender
50     Serial.print("Se va encender el led: ");
51     Serial.println(valorIngresado);
52
53     // Limpia el búfer del serial
54     while (Serial.available() > 0) {
55       Serial.read();
56     }
57   }
58   if (valorIngresado >= 5) {
59     Serial.println("fuera del rango ");
60     delay(2000);
61   }
62
63   switch (valorIngresado) {
64     case 0:
65       digitalWrite(led1, LOW);
66       digitalWrite(led2, LOW);
67       digitalWrite(led3, LOW);
68       digitalWrite(led4, LOW);
69       break;
70     case 1:
71       digitalWrite(led1, HIGH);
72       digitalWrite(led2, LOW);
73       digitalWrite(led3, LOW);
74       digitalWrite(led4, LOW);
75       break;
76     case 2:
77       digitalWrite(led1, LOW);
78       digitalWrite(led2, HIGH);

```

On the right, the simulation window shows an ESP32 board connected to a breadboard with four LEDs. The serial monitor displays the following output:

```

load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
Que led quieres prender :
Se va encender el led: 1

```

## Código:

```

32  int valorIngresado = 0;
33  // Prototipo de la función pinSetup
34  void pinSetup();
35
36  void setup()
37  {
38      Serial.begin(9600); // Inicia la comunicación serial a 9600 baudios
39      Serial.println("Que led quieres prender :"); // se pide que ingrese que led
40      pinSetup();
41  }
42
43  void loop()
44  {
45      if (Serial.available() > 0) {
46          // Lee el valor ingresado
47          valorIngresado = Serial.parseInt();
48
49          // Imprime el led que se va a encender
50          Serial.print("Se va encender el led: ");
51          Serial.println(valorIngresado);
52
53          // Limpia el búfer del serial
54          while (Serial.available() > 0) {
55              Serial.read();
56          }
57      }
58      if(valorIngresado >= 5){
59          Serial.println("fuera del rango ");
60          delay(2000);
61      }
62
63      switch (valorIngresado) {
64          case 0:
65              digitalWrite(led1, LOW);
66              digitalWrite(led2, LOW);
67              digitalWrite(led3, LOW);
68              digitalWrite(led4, LOW);
69              break;
70          case 1:
71              digitalWrite(led1, HIGH);
72              digitalWrite(led2, LOW);
73              digitalWrite(led3, LOW);
74              digitalWrite(led4, LOW);
75              break;
76          case 2:
77              digitalWrite(led1, LOW);
78              digitalWrite(led2, HIGH);
79              digitalWrite(led3, LOW);
80              digitalWrite(led4, LOW);
81              break;
82          case 3:
83              digitalWrite(led1, LOW);
84              digitalWrite(led2, LOW);
85              digitalWrite(led3, HIGH);
86              digitalWrite(led4, LOW);
87              break;
88          case 4:
89              digitalWrite(led1, LOW);
90              digitalWrite(led2, LOW);
91              digitalWrite(led3, HIGH);
92              digitalWrite(led4, LOW);
93              break;
94      }

```



## Ejercicio 16: Secuencia de luces de emergencia

- Simula luces de emergencia con los LEDs, donde led1 y led2 parpadean alternativamente en un patrón rápido, mientras que led3 y led4 lo hacen en un patrón más lento.

WOKWI

SAVE

SHARE

test

Docs

sketch.ino

diagram.json

Library Manager

```

28 // Prototipo de la función pinSetup
29
30 void setup()
31 {
32   pinSetup();
33 }
34
35 void loop()
36 {
37   digitalWrite(led1, HIGH); // Enciende el LED1
38   digitalWrite(led3, HIGH); // Enciende el LED3
39   delay(1000);
40   digitalWrite(led3, LOW); // Apaga el LED3
41   digitalWrite(led4, HIGH); // Enciende el LED4
42   delay(500);
43   digitalWrite(led4, LOW); // Apaga el LED4
44   delay(500);
45   digitalWrite(led1, LOW); // Apaga el LED1
46   digitalWrite(led2, HIGH); // Enciende el LED2
47   delay(1000);
48   digitalWrite(led2, LOW); // Apaga el LED2
49 }
50 // Definición de la función pinSetup
51 void pinSetup()
52 {
53   // Configuración de los pines de los pulsadores como entrada
54   pinMode(btn1, INPUT);
55   pinMode(btn2, INPUT);
56   pinMode(btn3, INPUT);
57
58   // Configuración de los pines de los dip switches como entrada con pull-up
59   pinMode(sw1, INPUT_PULLUP);
60   pinMode(sw2, INPUT_PULLUP);
61   pinMode(sw3, INPUT_PULLUP);
62   pinMode(sw4, INPUT_PULLUP);
63   pinMode(sw5, INPUT_PULLUP);
64   pinMode(sw6, INPUT_PULLUP);

```

Simulation

00:07.709

77%

clk\_drv:0x00,q\_drv:0x00,d\_drv:0x00,cs0\_drv:0x00,hd\_drv:0x00,wp\_drv:0x00

mode:DIO, clock div:2

load:0x3fff0030,len:1156

load:0x40078000,len:11456

ho 0 tail 12 room 4

load:0x40080400,len:2972

entry 0x400805dc

## Codigo:

```

35 void loop()
36 {
37   digitalWrite(led1, HIGH); // Enciende el LED1
38   digitalWrite(led3, HIGH); // Enciende el LED3
39   delay(1000);
40   digitalWrite(led3, LOW); // Apaga el LED3
41   digitalWrite(led4, HIGH); // Enciende el LED4
42   delay(500);
43   digitalWrite(led4, LOW); // Apaga el LED4
44   delay(500);
45   digitalWrite(led1, LOW); // Apaga el LED1
46   digitalWrite(led2, HIGH); // Enciende el LED2
47   delay(1000);
48   digitalWrite(led2, LOW); // Apaga el LED2

```