

PROYECTO INTERGRADOR I

DOCENTE: GONZALO VERA

ALUMNO/A: KARINA JAZMIN BARBERO

CARRERA: TÉCNICO SUPERIOR EN TELECOMUNICACIONES

Trabajo Practico #3 Transductores binarios

Objetivos

- Practicas con el framework de Arduino en VsCode
- Primera aproximación a un entrenador básico
- Practica con sensores y actuadores digitales
- Primera aproximación a un controlador

Ejercicios a resolver:

Nivel Principiante

Ejercicio 1:

Encender un LED

- Enciende el led1 conectado al GPIO18 de forma continua.

WOKWI SALVAR COMPARTIR TP3 Ejercicio 1: Encender un LED Docs

sketch.ino diagram.json Administrador de la biblioteca

```
1
2 const int ledPin = 18; Pin del LED
3
4 Configuración nula() {
5   pinMode(ledPin, OUTPUT); Configurar el pin como salida
6 }
7
8 Bucle vacío() {
9   digitalWrite(ledPin, HIGH); Encender el LED
10  No hay tiempo de espera (delay) aquí
11 }
12
13
```

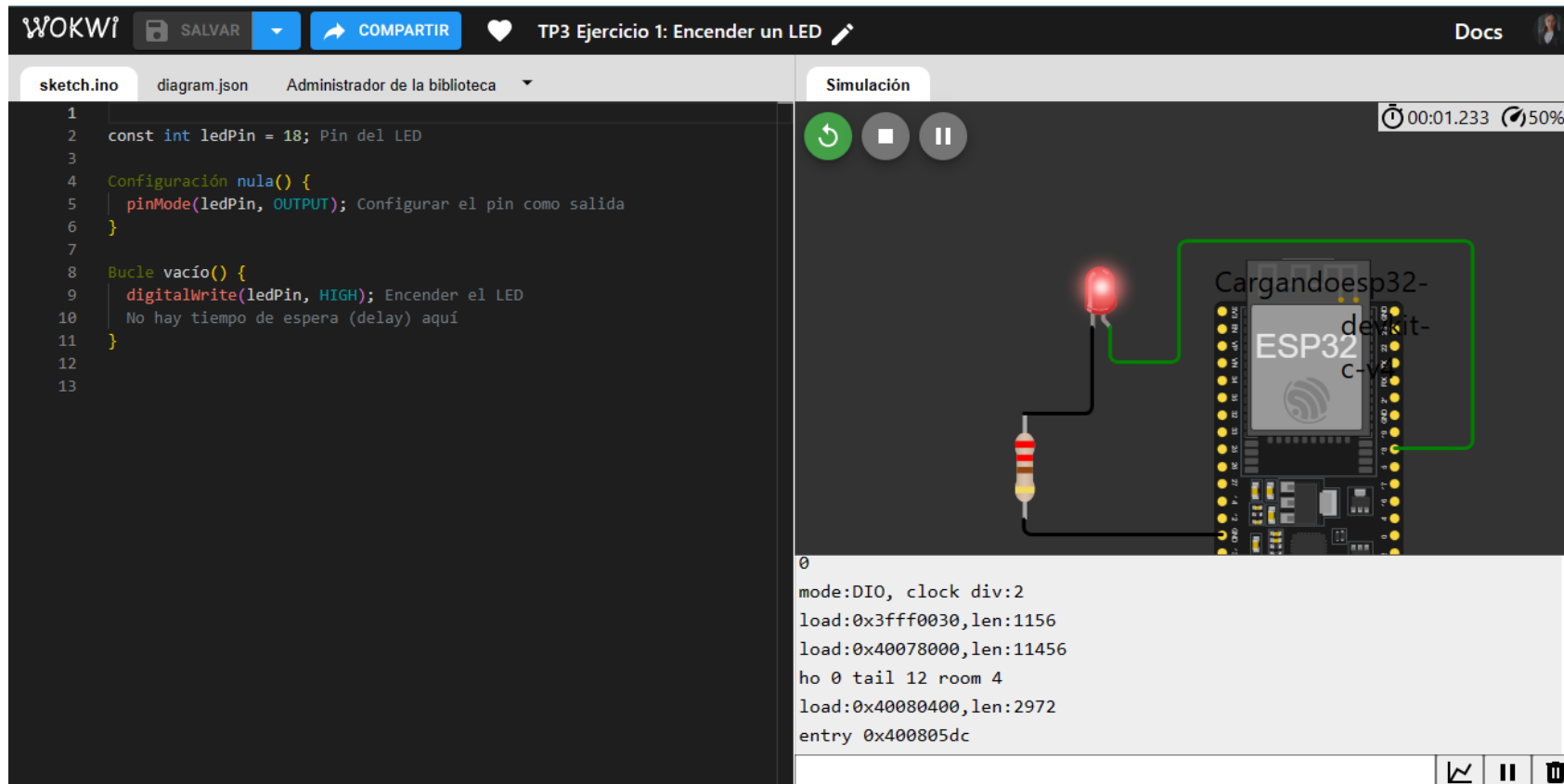
Simulación

00:01.233 50%

Cargando esp32-
de kit-
cy

ESP32

0
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



<https://wokwi.com/projects/397871568616607745>

Ejercicio 2:

Parpadeo de un LED

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

The image shows a Wokwi web IDE interface for a simulation titled "TP3 Ejercicio 2 : Parpadeo de un LED".

Code (sketch.ino):

```
1 Este código simula el parpadeo de un LED conectado al pin 18 (GPIO18) con un intervalo de 1 segundo
2
3 const int ledPin = 18; Pin del LED
4
5 Configuración nula() {
6   pinMode(ledPin, OUTPUT); Configurar el pin como salida
7 }
8
9 Bucle vacío() {
10  digitalWrite(ledPin, HIGH); Encender el LED
11  retraso(1000); Esperar 1 segundo
12
13  digitalWrite(ledPin, LOW); Apagar el LED
14  retraso(1000); Esperar 1 segundo
15 }
16
```

Simulation: The simulation shows an ESP32 microcontroller connected to an LED. The LED is currently lit. The simulation controls include a play button, a stop button, and a pause button. The timer shows 00:00.833 and the zoom level is 75%.

Log:

```
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
```

<https://wokwi.com/projects/396873154990287873>

Ejercicio 3:

Secuencia de LEDs

- Crea una secuencia que encienda los LEDs del led1 al led3 de forma sucesiva, cada uno durante 500ms.

The image shows the Wokwi online IDE interface. On the left is the code editor with the following sketch:

```
1 // Definición de pines para los LEDs
2 const int led1Pin = 18;
3 const int led2Pin = 19;
4 const int led3Pin = 21;
5
6 Configuración nula() {
7   // Configura los pines como salidas
8   pinMode(led1Pin, SALIDA);
9   pinMode(led2Pin, SALIDA);
10  pinMode(led3Pin, SALIDA);
11 }
12
13 Bucle vacío() {
14   Enciende el LED 1 durante 500 ms
15   digitalWrite(led1Pin, HIGH);
16   retraso(500); Espera 500 ms
17   digitalWrite(led1Pin, LOW);
18
19   Enciende el LED 2 durante 500 ms
20   digitalWrite(led2Pin, HIGH);
21   retraso(500); Espera 500 ms
22   digitalWrite(led2Pin, LOW);
23
24   Enciende el LED 3 durante 500 ms
25   digitalWrite(led3Pin, HIGH);
26   retraso(500); Espera 500 ms
27   digitalWrite(led3Pin, LOW);
28 }
```

On the right is the simulation window, titled "Simulación". It shows a virtual ESP32 board connected to three red LEDs. The second LED is currently lit. The simulation controls include a play button, a pause button, and a stop button. The timer shows 00:40.438 and 99% completion. The console at the bottom displays the load and entry addresses.

<https://wokwi.com/projects/397878960832195585>

Ejercicio 4:

Control de LED con botón

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

WOKWI SALVAR COMPARTIR TP Ejercicio 4: Control de LED con botón Docs

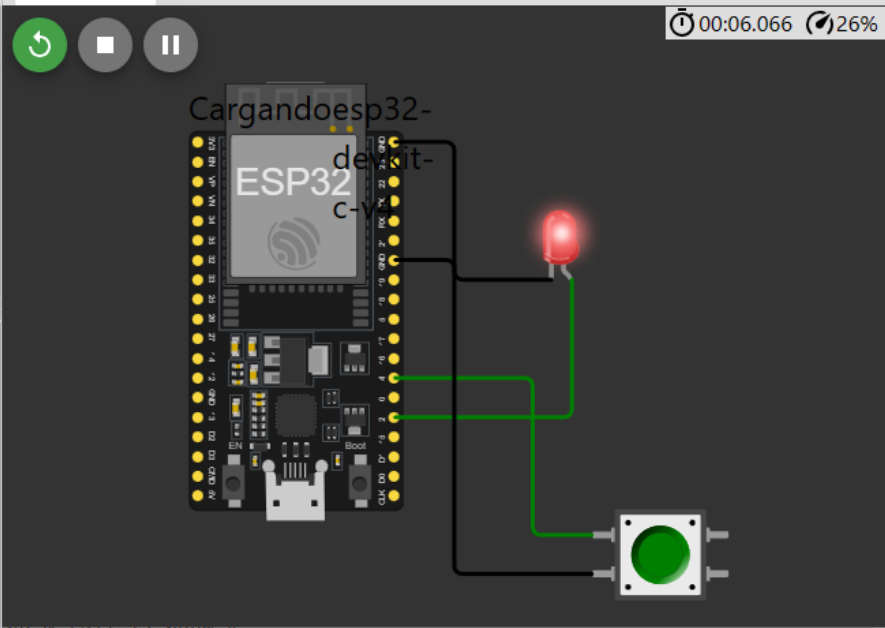
sketch.ino diagram.json ● Administrador de la biblioteca

```
1 const int ledPin = 2; Pin del LED
2 const int buttonPin = 4; Pin del botón
3
4
5 Configuración nula() {
6   Ponga su código de instalación aquí, para ejecutarlo una vez:
7   Serie.begin(115200);
8   Serie.println("!Hola, ESP32!");
9
10  pinMode(ledPin, OUTPUT);
11
12  pinMode(buttonPin, INPUT_PULLUP); Configura el botón con resistencia pul
13 }
14
15 Bucle vacío() {
16   if (digitalRead(buttonPin) == LOW) { Si el botón está presionado
17     digitalWrite(ledPin, HIGH); Enciende el LED
18   } más {
19     digitalWrite(ledPin, LOW); Apaga el LED
20   }
21 }
22
23
```

Simulación

00:06.066 26%

Cargando esp32-
devkit-c



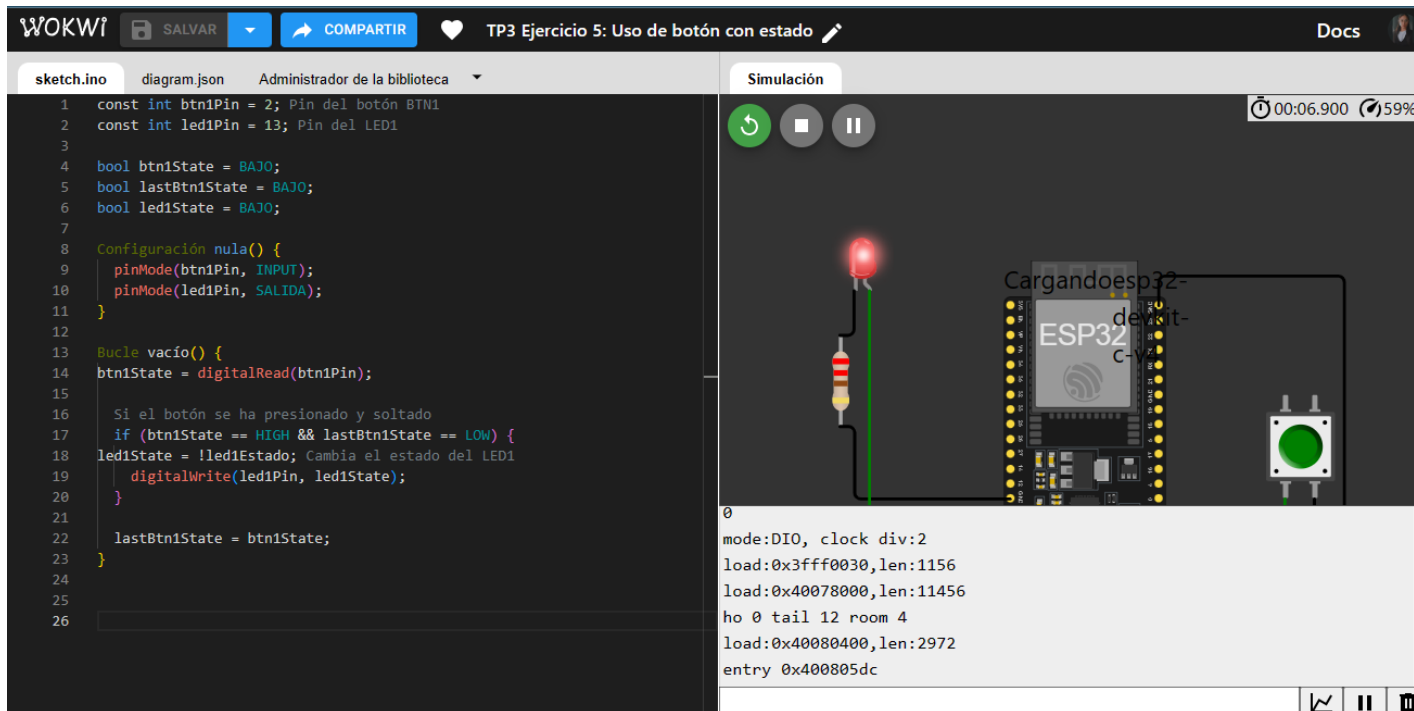
no 0 call 12 Room 4
load:0x40080400,len:2972
entry 0x400805dc
Hello, ESP32!

<https://wokwi.com/projects/396878487947781121>

Nivel Intermedio

Ejercicio 5: Uso de botón con estado

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



The image shows the Wokwi web IDE interface. The left pane displays the sketch code, and the right pane shows the simulation of the hardware.

Sketch Code:

```
1  const int btn1Pin = 2; Pin del botón BTN1
2  const int led1Pin = 13; Pin del LED1
3
4  bool btn1State = BAJ0;
5  bool lastBtn1State = BAJ0;
6  bool led1State = BAJ0;
7
8  Configuración nula() {
9    pinMode(btn1Pin, INPUT);
10   pinMode(led1Pin, SALIDA);
11 }
12
13 Bucle vacío() {
14   btn1State = digitalRead(btn1Pin);
15
16   Si el botón se ha presionado y soltado
17   if (btn1State == HIGH && lastBtn1State == LOW) {
18     led1State = !led1Estado; Cambia el estado del LED1
19     digitalWrite(led1Pin, led1State);
20   }
21
22   lastBtn1State = btn1State;
23 }
24
25
26
```

Simulation: The simulation window shows an ESP32 module connected to a red LED (pin 13) and a push button (pin 2). The LED is currently lit. The simulation controls (play, pause, stop) and a timer (00:06.900) are visible. The console output shows the following messages:

```
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
```

<https://wokwi.com/projects/396882436325402625>

Ejercicio 6:

Debounce de botón

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

WOKWI SALVAR COMPARTIR TP3 Ejercicio 6: Debounce de botón Docs

sketch.ino diagram.json Administrador de la biblioteca

```
1 const int buttonPin = 2; DONDE ESTÁ CONECTADO EL PIN DEL PULSADOR
2 const int ledPin = 13;
3
4 botón bool volátilPresionado = false;
5
6 manija void buttonPress() {
7   buttonPressed = true;
8 }
9
10 Configuración nula() {
11   pinMode(buttonPin, INPUT_PULLUP);
12   pinMode(ledPin, OUTPUT);
13   attachInterrupt(digitalPinToInterrupt(buttonPin), handleButtonPress, FALLING);
14 }
15
16 Bucle vacío() {
17   if (buttonPressed) {
18     digitalWrite(ledPin, !digitalRead(ledPin)); Cambia el estado del LED
19   }
20   buttonPressed = false;
21 }
22
23
```

Simulación

00:14.100 56%

Cargando esp32-
de kit-
c-

0
mode:DI0, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc

<https://wokwi.com/projects/397313341730031617>

Ejercicio 7:

Control de múltiples LEDs con botones

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

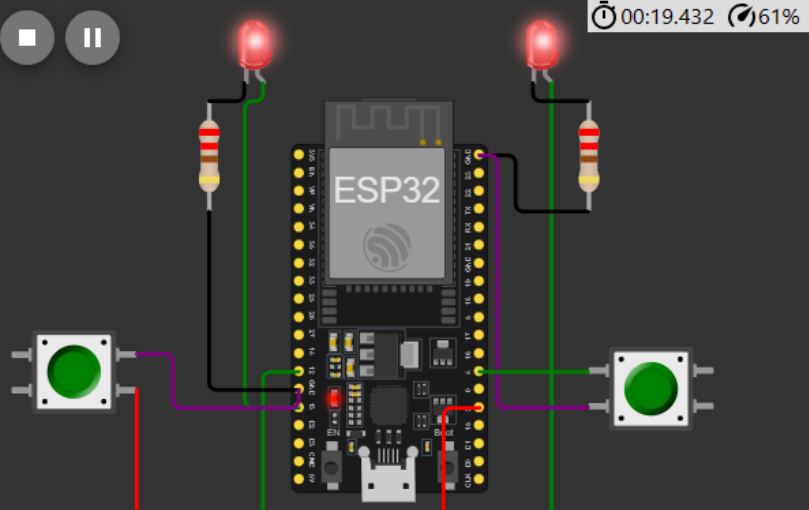
WOKWI SAVE SHARE TP3 Ejercicio 7: Control de múltiples LEDs con botones Docs

sketch.ino diagram.json Library Manager

```
1 const int led1Pin = 13; // Pin para el LED 1
2 const int led2Pin = 12; // Pin para el LED 2
3 const int btn1Pin = 2; // Pin para el botón 1
4 const int btn2Pin = 4; // Pin para el botón 2
5
6 bool led1State = false; // Estado inicial del LED 1 (apagado)
7 bool led2State = false; // Estado inicial del LED 2 (apagado)
8
9 void setup() {
10   pinMode(led1Pin, OUTPUT);
11   pinMode(led2Pin, OUTPUT);
12   pinMode(btn1Pin, INPUT_PULLUP);
13   pinMode(btn2Pin, INPUT_PULLUP);
14 }
15
16 void loop() {
17   // Lee el estado de los botones
18   bool btn1Pressed = !digitalRead(btn1Pin); // Invertimos el estado porque
19   bool btn2Pressed = !digitalRead(btn2Pin); // Invertimos el estado porque
20
21   // Actualiza el estado de los LEDs según los botones
22   if (btn1Pressed) {
23     led1State = !led1State; // Cambia el estado del LED 1
24     digitalWrite(led1Pin, led1State);
25   }
26
27   if (btn2Pressed) {
28     led2State = !led2State; // Cambia el estado del LED 2
29     digitalWrite(led2Pin, led2State);
30   }
31 }
```

Simulation

00:19.432 61%



mode:DI0, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc

<https://wokwi.com/projects/397315571978113025>

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.

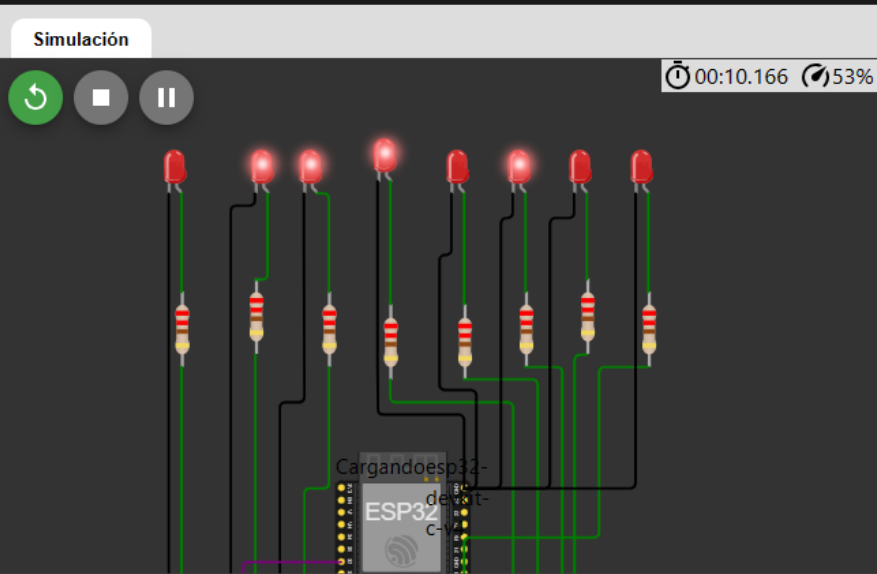
WOKWI SALVAR COMPARTIR TP3 Ejercicio 8: Uso de dip switches para control de LEDs Docs

sketch.ino diagram.json Administrador de la biblioteca

```
1 Definición de pines para los dip switches y los LEDs
2 const int dipSwitchPins[] = {2, 3, 4, 5, 25, 26, 27, 32, };
3 const int ledPins[] = { 12, 13, 14, 15, 16, 17, 18, 19,};
4
5 Configuración nula() {
6   Configura los pines de los dip switches como entradas
7   for (int i = 0; i < 8; i++) {
8     pinMode(dipSwitchPins[i], INPUT);
9   }
10
11   Configura los pines de los LEDs como salidas
12   for (int i = 0; i < 8; i++) {
13     pinMode(ledPins[i], SALIDA);
14   }
15 }
16
17 Bucle vacío() {
18   Lee el estado de los dip switches
19   for (int i = 0; i < 8; i++) {
20     int switchState = digitalRead(dipSwitchPins[i]);
21
22     Si el dip switch está encendido, activa el LED correspondiente
23     if (switchState == HIGH) {
24       digitalWrite(ledPins[i], HIGH);
25     } más {
26       digitalWrite(ledPins[i], LOW);
27     }
28   }
29 }
30
```

Simulación

00:10.166 53%



Cargando esp32-
de
C-

ets Jul 29 2019 12:21:46

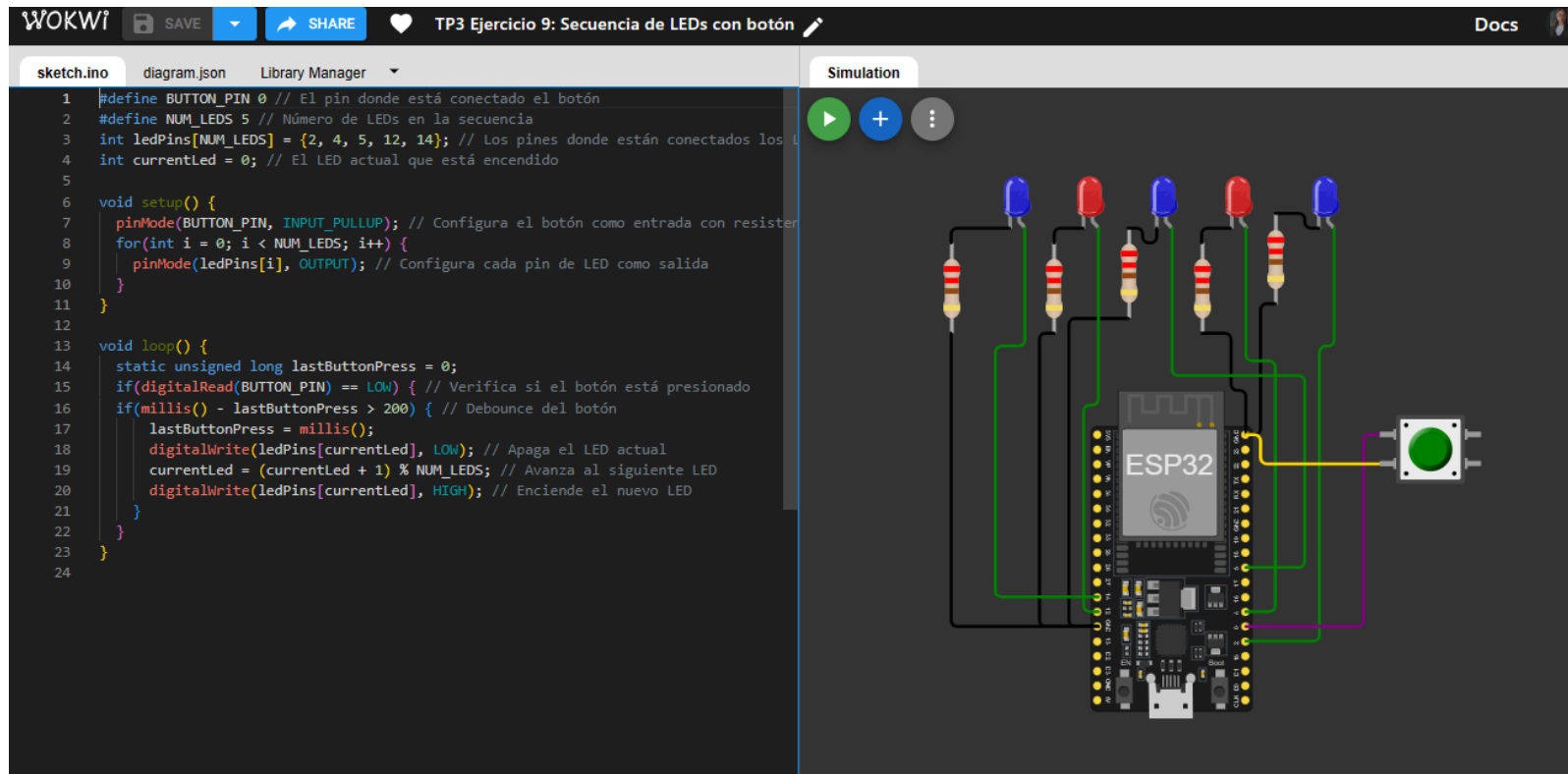
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
config: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2

<https://wokwi.com/projects/397321393877431297>

Nivel Avanzado

Ejercicio 9: Secuencia de LEDs con botón

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



<https://wokwi.com/projects/397981072347534337>

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.

WOKWI SAVE SHARE TP3: Ejercicio 10: Control de velocidad de parpadeo con dip switch Docs

sketch.ino diagram.json Library Manager

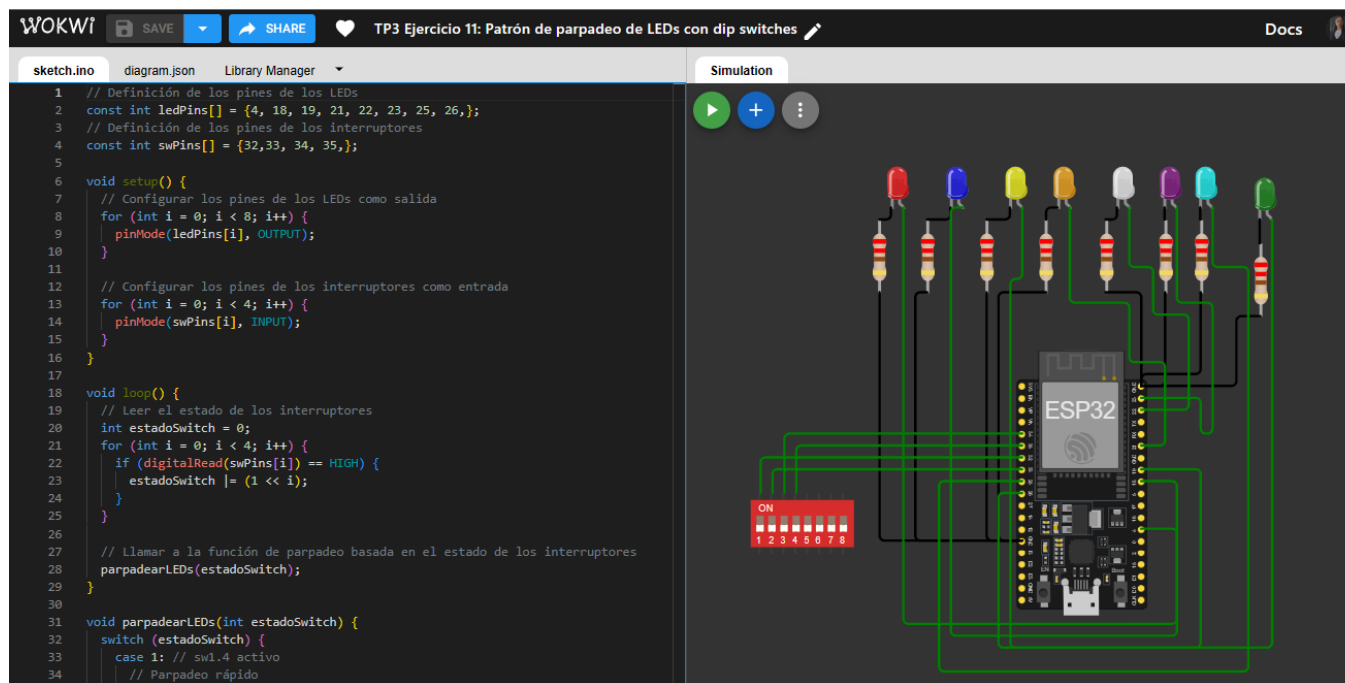
```
1 // Define los pines para los dip switches y el LED
2 const int dipSwitch1 = 2; // SW1.1
3 const int dipSwitch2 = 3; // SW1.2
4 const int dipSwitch3 = 4; // SW1.3
5 const int ledPin = 13; // LED1
6
7 void setup() {
8   // Configura los pines de los dip switches como entrada
9   pinMode(dipSwitch1, INPUT);
10  pinMode(dipSwitch2, INPUT);
11  pinMode(dipSwitch3, INPUT);
12
13  // Configura el pin del LED como salida
14  pinMode(ledPin, OUTPUT);
15 }
16
17 void loop() {
18   // Lee el estado de los dip switches
19   int switchState = digitalRead(dipSwitch1) + (digitalRead(dipSwitch2) << 1) + (digitalRead(dipSwitch3) << 2);
20
21   // Asigna una velocidad de parpadeo basada en el estado de los dip switches
22   int blinkSpeed;
23   switch (switchState) {
24     case 0: blinkSpeed = 1000; break; // Todos los switches apagados
25     case 1: blinkSpeed = 500; break; // SW1.1 activado
26     case 2: blinkSpeed = 250; break; // SW1.2 activado
27     case 3: blinkSpeed = 125; break; // SW1.1 y SW1.2 activados
28     // Agrega más casos según sea necesario
29     default: blinkSpeed = 1000; // Configuración por defecto
30   }
31
32   // Hace parpadear el LED a la velocidad asignada
33   digitalWrite(ledPin, HIGH);
34   delay(blinkSpeed);
35   digitalWrite(ledPin, LOW);
36   delay(blinkSpeed);
37 }
```

Simulation

<https://wokwi.com/projects/397987750312257537>

Ejercicio 11: Patrón de parpadeo de LEDs con dip switches

- Establece un patrón de parpadeo para los led1 a led8 basado en la combinación de estados de sw1.1 a sw1.4. Por ejemplo, cada posición activa del switch puede representar un patrón diferente (como parpadeo rápido, lento, secuencial, etc.)



<https://wokwi.com/projects/397999449558761473>

Ejercicio 12: Medidor de pulsaciones

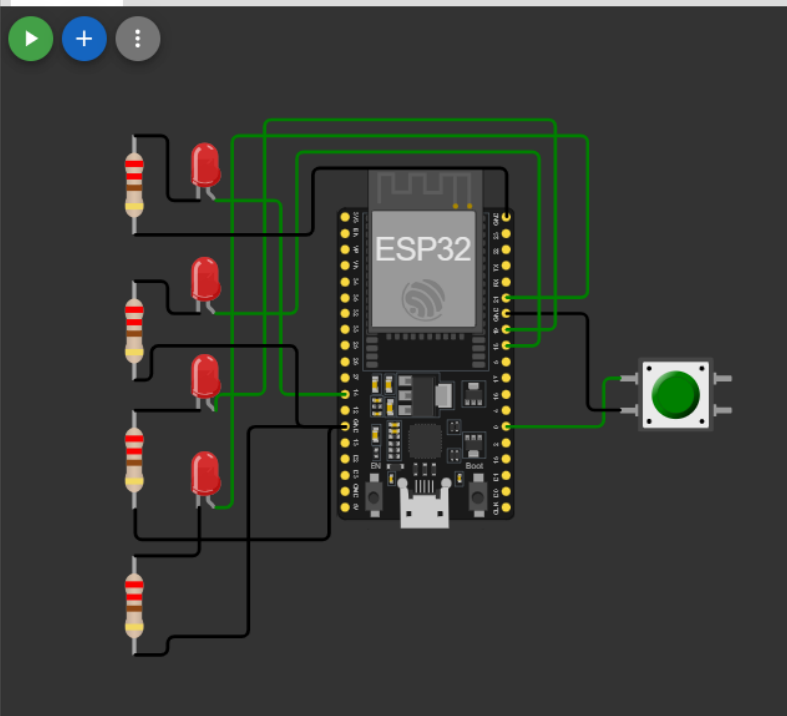
- Programa un contador de pulsaciones utilizando btn1. El número de pulsaciones debe mostrarse en una secuencia de LEDs (por ejemplo, led5 a led8 donde cada LED representa una cantidad de pulsaciones).

WOKWI SAVE SHARE TP3 Ejercicio 12: Medidor de pulsaciones Docs

sketch.ino diagram.json Library Manager

```
1 // Definición de los pines GPIO para los LEDs y el botón
2 const int btn1 = 0; // Botón conectado al GPIO 0
3 const int led5 = 14; // LED conectado al GPIO 5
4 const int led6 = 18; // LED conectado al GPIO 18
5 const int led7 = 19; // LED conectado al GPIO 19
6 const int led8 = 21; // LED conectado al GPIO 21
7
8 // Variables para almacenar el estado del botón y el contador de pulsaciones
9 int estadoBtn1 = 0;
10 int contadorPulsaciones = 0;
11
12 // Variables para el control de parpadeo de los LEDs
13 unsigned long tiempoAnterior = 0;
14 const long intervaloLed5 = 500; // Intervalo para led5
15 const long intervaloLed6 = 300; // Intervalo para led6
16 const long intervaloLed7 = 250; // Intervalo para led7
17 const long intervaloLed8 = 150; // Intervalo para led8
18
19 void setup() {
20   // Configurar el botón como entrada con pull-up interno
21   pinMode(btn1, INPUT_PULLUP);
22   // Configurar los LEDs como salidas
23   pinMode(led5, OUTPUT);
24   pinMode(led6, OUTPUT);
25   pinMode(led7, OUTPUT);
26   pinMode(led8, OUTPUT);
27 }
28
29 void loop() {
30   // Leer el estado del botón
31   estadoBtn1 = digitalRead(btn1);
32
33   // Si el botón está presionado (estado bajo debido al pull-up)
34   if (estadoBtn1 == LOW) {
```

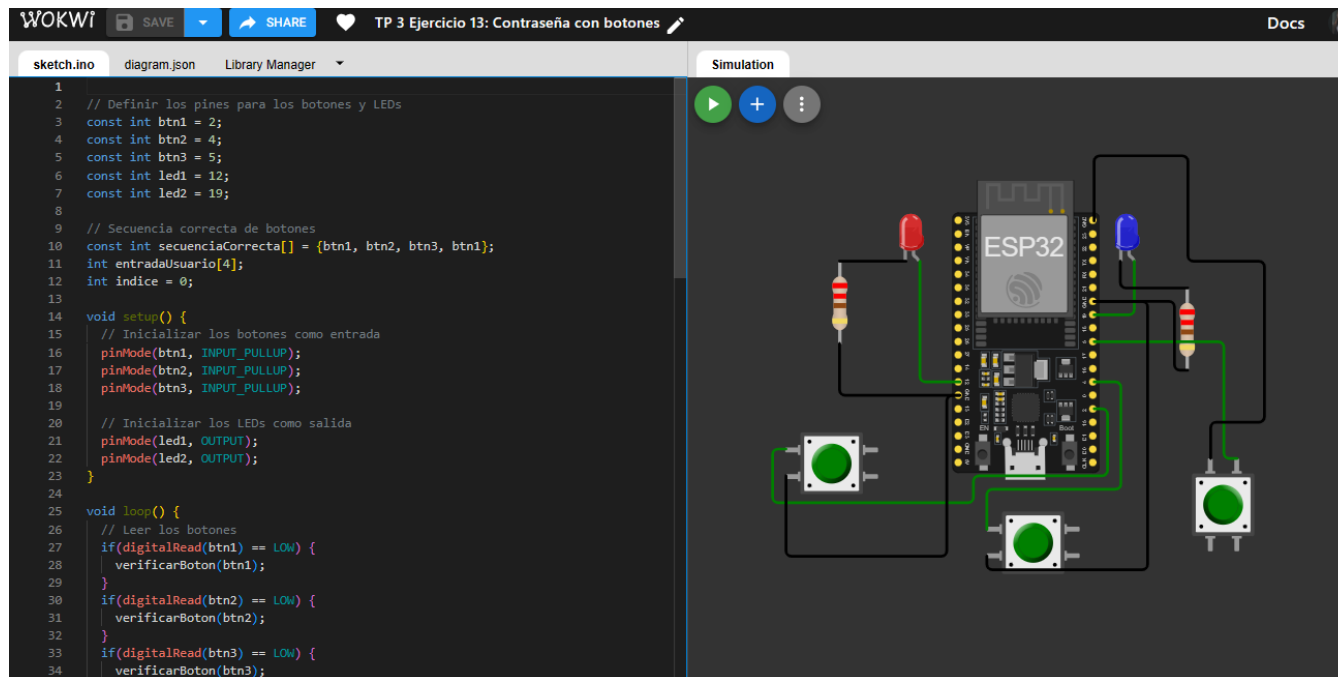
Simulation



<https://wokwi.com/projects/398033745515581441>

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.



<https://wokwi.com/projects/398062826293800961>

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 SALVAR COMPARTIR TP3 Ejercicio 14: Aplicación de timers para control de LEDs Docs

sketch.ino diagram.json Administrador de la biblioteca

```
1 const int ledPins[] = {5, 18, 19, 21}; DONDE ESTÁN LOS PINES DE LOS LEDS
2 intervalo largo const = 1000; Intervalo de parpadeo en milisegundos
3
4 sin signo largo anteriorMillis = 0; Almacenará la última vez que se actualizó el LED
5
6 Configuración nula() {
7   Inicializa los pines de los LEDs como salidas
8   for (int i = 0; i < 4; Yo++) {
9     pinMode(ledPins[i], SALIDA);
10  }
11 }
12
13 Bucle vacío() {
14   corriente larga sin signoMillis = millis();
15
16   if (currentMillis - previousMillis >= intervalo) {
17     // Guarda la última vez que parpadeo los LEDs
18     anteriorMillis = currentMillis;
19
20     for (int i = 0; i < 4; Yo++) {
21       int ledState = digitalRead(ledPins[i]); Lee el estado actual del LED
22       digitalWrite(ledPins[i], !ledState); Cambia el estado del LED
23     }
24   }
25 }
26
27 }
28
```

Simulación

<https://wokwi.com/projects/398069575690036225>

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 led1, etc.

WOKWI SAVE SHARE TP3 Ejercicio 15: Control de LEDs mediante comunicación serial Docs

sketch.ino diagram.json Library Manager

```
1 #include "Arduino.h"
2
3 // Definir los pines de los LEDs
4 const int led1 = 23; // PIN DEL LED 1
5 const int led2 = 22; // PIN DEL LED 2
6
7
8 void setup() {
9   Serial.begin(115200);
10  pinMode(led1, OUTPUT);
11  pinMode(led2, OUTPUT);
12
13 }
14
15 void loop() {
16   if (Serial.available() > 0) {
17     char comando = Serial.read();
18     switch (comando) {
19       case '1':
20         digitalWrite(led1, HIGH); // Encender led1
21         break;
22       case '2':
23         digitalWrite(led1, LOW); // Apagar led1
24         break;
25       case '3':
26         digitalWrite(led2, HIGH); // Encender led2
27         break;
28       case '4':
29         digitalWrite(led2, LOW); // Apagar led2
30         break;
31       default:
32         // Si el comando no es reconocido, no hacer nada
33         break;
34     }
35   }
36 }
```

Simulation

<https://wokwi.com/projects/398071415393705985>

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 TP3 Ejercicio 16: Secuencia de luces de emergencia Docs

sketch.ino diagram.json Library Manager Simulation

```
1
2 // Definir los pines de los LEDs
3 const int led1 = 23;
4 const int led2 = 22;
5 const int led3 = 21;
6 const int led4 = 19;
7
8 void setup() {
9   pinMode(led1, OUTPUT);
10  pinMode(led2, OUTPUT);
11  pinMode(led3, OUTPUT);
12  pinMode(led4, OUTPUT);
13 }
14
15 void loop() {
16   // Patrón rápido: led1 y led2 parpadean alternativamente
17   digitalWrite(led1, HIGH);
18   delay(250); // Ajustar para cambiar la velocidad del parpadeo rápido
19   digitalWrite(led1, LOW);
20   digitalWrite(led2, HIGH);
21   delay(250); // Ajustar para cambiar la velocidad del parpadeo rápido
22   digitalWrite(led2, LOW);
23
24   // Patrón lento: led3 y led4 parpadean alternativamente
25   digitalWrite(led3, HIGH);
26   delay(1000); // Ajustar para cambiar la velocidad del parpadeo lento
27   digitalWrite(led3, LOW);
28   digitalWrite(led4, HIGH);
29   delay(1000); // Ajustar para cambiar la velocidad del parpadeo lento
30   digitalWrite(led4, LOW);
31 }
32
33
```

<https://wokwi.com/projects/398071772052169729>