SIMULACIÓN DEL ESP32 EN WOKWI PARA EL APRENDIZAJE PRÁCTICO DE SISTEMAS EMBEBIDOS

Gonzalo Sánchez Aréchiga
Estudiante de Ingeniería Aeronáutica
Instituto Politécnico Nacional – ESIME Ticomán
gsancheza2100@alumno.ipn.mx

Dr. Juan Carlos Herrera Lozada
Instituto Politécnico Nacional – CIDETEC
jlozada@ipn.mx

RESUMEN

El presente artículo tiene como objetivo explicar el uso de Wokwi, un simulador en línea de circuitos electrónicos y microcontroladores, aplicado al desarrollo de proyectos con la tarjeta ESP32. Esta plataforma permite probar circuitos desde el navegador, sin necesidad de tener todos los componentes físicos desde el inicio. Para mostrar su funcionamiento se presentan dos ejemplos prácticos: el primero consiste en encender un LED mediante un pushbutton, mientras que el segundo desarrolla un contador hexadecimal en un display de siete segmentos. Con estos ejercicios se puede observar cómo Wokwi ayuda a comprender mejor la relación entre el código, las conexiones y el comportamiento del circuito. Además, permite detectar errores antes de armar el proyecto físicamente, lo cual resulta útil para estudiantes que están aprendiendo sistemas embebidos.

ABSTRACT

This article aims to explain the use of Wokwi, an online simulator for electronic circuits and microcontrollers, applied to the creation of projects with the ESP32 board. This platform allows users to test circuits directly from a web browser, without needing all the physical components from the beginning. To show how it works, two practical examples are presented: the first one consists of turning on an LED using a pushbutton, while the second one develops a hexadecimal counter on a seven-segment display. Through these exercises, it is possible to see how Wokwi helps users better understand the relationship between the code, the connections, and the behavior of the circuit. In addition, it allows errors to be detected before building the project physically, which is useful for students who are learning about embedded systems.

1. Introducción

Cuando se trabaja con el ESP32, al principio los proyectos pueden parecer sencillos porque normalmente solo se necesita escribir un código básico y conectar algunos componentes. Sin embargo, conforme el proyecto crece, también aumentan las conexiones, los cables, las resistencias y los posibles errores.

En una práctica física, un cable mal conectado o una resistencia incorrecta puede provocar que el circuito no funcione, que el componente se caliente o incluso que se dañe. Por esta razón, antes de armar un circuito real, es muy útil probarlo en un simulador.

Wokwi permite hacer esto de una manera sencilla, ya que muestra el circuito y el código en una misma pantalla. Así, el usuario puede verificar si el problema está en la programación, en las conexiones o en la forma en que se está usando algún componente. Esto ayuda a aprender de manera más ordenada y reduce el riesgo de cometer errores en el montaje físico.

2. ¿Qué es Wokwi?

Wokwi es un simulador en línea que permite crear y probar circuitos electrónicos. Se puede usar con componentes básicos como LEDs, resistencias, botones, sensores, pantallas y microcontroladores como el ESP32.

Una de sus ventajas es que no se necesita instalar un programa pesado en la computadora, ya que funciona directamente desde el navegador. Además, permite escribir el código y ver cómo responde el circuito en tiempo real.

También se puede trabajar con archivos como *diagram.json*, donde se definen los componentes y las conexiones del circuito. Esto puede parecer complicado al inicio, pero ayuda mucho cuando se quiere tener un diseño más ordenado o cuando el circuito tiene muchos cables.

3. Funcionamiento general de Wokwi

Al entrar por primera vez a Wokwi, lo primero que se debe hacer es elegir el tipo de microcontrolador con el que se va a trabajar o seleccionar la opción “Blank” para crear un proyecto nuevo desde cero. Una vez generado el proyecto, el entorno de trabajo se divide en dos partes principales: del lado izquierdo se escribe el código y del lado derecho se visualiza el circuito electrónico. Esto permite observar de manera directa cómo se comporta el sistema conforme se realizan cambios en la programación o en las conexiones.

Para la parte del hardware, Wokwi trabaja con un archivo llamado *diagram.json*, donde se definen los componentes, sus posiciones y las conexiones del circuito. Aunque se pueden escribir los elementos directamente en este archivo, también existe una forma más sencilla de agregarlos. En la pantalla de visualización aparece un signo de “+”, el cual despliega una biblioteca de componentes con un buscador. Esto facilita encontrar rápidamente LEDs, resistencias, botones, pantallas, sensores y otros elementos necesarios para el proyecto.

En los ejemplos desarrollados se utiliza lenguaje C++ dentro del entorno de Arduino, ya que es una de las formas más comunes de programar la tarjeta ESP32. Para esto, se puede crear un archivo con extensión .ino, donde se escribe la lógica que controlará el funcionamiento del microcontrolador.

Este proceso es parecido al que se realiza en programas como Arduino IDE o VS Code cuando se carga código a una tarjeta física.

Después de configurar el circuito en el archivo diagram.json y escribir las instrucciones en el archivo .ino, se presiona el botón de Play para iniciar la simulación. En ese momento, el programa se compila y el circuito comienza a funcionar. A partir de ahí, se pueden presionar botones, observar el encendido de LEDs, revisar pantallas o probar sensores de forma interactiva, como si se tratara de un montaje físico.

Finalmente, el proyecto puede guardarse en la nube mediante una cuenta gratuita de Wokwi. Esto permite conservar los avances y abrirlos nuevamente desde cualquier dispositivo con solo iniciar sesión. De esta forma, la plataforma resulta práctica para realizar pruebas, corregir errores y continuar trabajando en un proyecto sin depender siempre del material físico.

4. Diseños desarrollados

4.1 Control de LED con pushbutton

Para este primer diseño, se selecciona el ESP32 como base para armar el circuito. Después se buscan los componentes necesarios para su correcto cableado y funcionamiento; estos se pueden encontrar en el buscador del botón "+" de Wokwi o, si se tiene el nombre exacto del componente, se puede escribir directamente en la sección "parts" del código .json junto con sus coordenadas y atributos. Un ejemplo para un LED verde sería:

```
{
  "type": "wokwi-led",
  "id": "led1",
  "top": 95.2,
  "left": 198.6,
  "attrs": { "color": "limegreen", "flip": "1" }
}
```

Para las conexiones, se puede hacer clic directamente en la visualización (un clic para el origen del cable y otro para el destino), o bien, definirlas en el bloque de "connections" del archivo .json. En este caso, se indica qué pines se conectan, el color del cable y la ruta en coordenadas que tomará para que el diseño se vea ordenado. Por ejemplo, para el cable rojo que conecta el pin de 3.3V del ESP32 con el botón:

```
[ "btn1:1.r", "esp:3V3", "red", [ "v38.6", "h-220.8", "v-45.28" ] ]
```

El objetivo de este diseño es controlar una salida digital mediante una entrada física. Como se observa en la Figura 1, para que el circuito funcione de forma segura y precisa, se integraron los siguientes elementos:

1. LED y resistencia (330 Ω): Se conectó el LED al pin D4. Se incluyó la resistencia en serie para limitar la corriente y evitar que el componente se queme, simulando lo que haríamos en una práctica real.

2. Pushbutton y resistencia (10 kΩ): El botón se conectó al pin D5. Aquí se utilizó una configuración de resistencia pull-down (conectada a tierra). Esto es fundamental para que, cuando el botón no esté presionado, el pin de entrada reciba un "0" constante (GND) y no se quede "flotando", lo que causaría encendidos erráticos del LED por ruido eléctrico.

El programa escrito en el entorno de Arduino es sencillo pero funcional. Primero, se definen las constantes para los pines y una variable llamada buttonState para almacenar el estado del botón.

En el setup(), se inicia la comunicación serial a 115200 baudios (estándar para el ESP32) para poder monitorear los datos en la terminal de Wokwi. Se configuran los pines: el del botón como INPUT y el del LED como OUTPUT.

En el loop(), el microcontrolador realiza lo siguiente de forma cíclica:

1. Lectura: Con la función digitalRead(buttonPin) se revisa si el botón está suministrando voltaje.

2. Impresión: Se envía el estado al monitor serie para verificar el funcionamiento en tiempo real.

3. Condicional: Se aplica un if. Si el estado es HIGH (botón presionado), se envía una señal de encendido al LED con digitalWrite(ledPin, HIGH). Si no se detecta presión, el LED se mantiene en LOW.

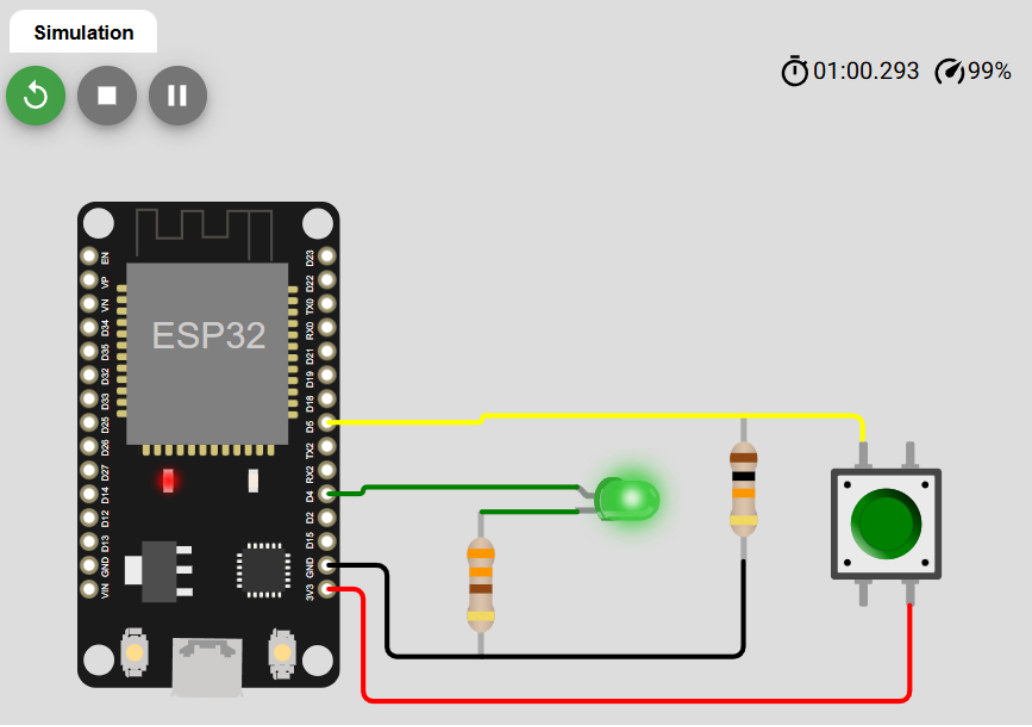

Figura 1. Entorno de simulación en Wokwi para el control de un LED mediante un pushbutton.

4.2 Contador hexadecimal en display de siete segmentos

Para este segundo diseño, el nivel de complejidad aumenta al tener que controlar siete salidas digitales de forma sincronizada para formar caracteres. Se utiliza un display de siete segmentos de cátodo común y un pushbutton que funciona como reset. Al igual que en el ejemplo anterior, los componentes se agregan mediante el menú "+" o editando el archivo .json. Un detalle importante en este diseño es la configuración del atributo common en el display para que sea compatible con la lógica del código:

```
{
  "type": "wokwi-7segment",
  "id": "sevseg",
  "top": 110.58,
  "left": 225.88,
  "attrs": { "common": "cathode", "color": "red" }
}
```

En este montaje, el ESP32 debe gestionar múltiples conexiones simultáneas. En la Figura 2 se puede apreciar la distribución del cableado, para lo cual se consideraron los siguientes puntos:

1. Display de 7 segmentos: Se conectaron los pines 15, 2, 4, 5, 18, 19 y 21 a los segmentos A, B, C, D, E, F y G respectivamente. Se agregaron resistencias de 330 Ω en las líneas comunes (COM) para simular la protección de corriente que se requiere en físico.

2. Botón de Reset: Se conectó al pin D23. Al igual que en el circuito anterior, se utilizó una resistencia de 10 kΩ en configuración pull-down. Su función aquí es reiniciar el contador a cero en cualquier momento, sin importar en qué número vaya la cuenta.

3. Cableado: Debido a la cantidad de conexiones, se utilizó el archivo JSON para organizar los cables por colores (azul, verde, amarillo, etc.), lo que facilita identificar qué pin del ESP32 controla cada segmento del display.

El código usa un decodificador y un contador automático que va del 0 al 15 (F en hexadecimal). Lo más importante es el uso de la "multitarea" mediante el uso de la función millis().

1. Tabla de verdad: Se creó una matriz de 16 x 7 llamada segmentos_display. Cada fila representa un número o letra (del 0 al F) y cada columna indica si un segmento debe encenderse (1) o apagarse (0).

2. Control de tiempo (Multitarea): En lugar de usar un delay(1000), que congelaría el programa, se usa una comparación de tiempo con millis(). Esto permite que el ESP32 "cuente" un segundo de forma interna mientras sigue revisando constantemente si alguien presionó el botón de reset.

3. Botón de Reset: El código lee el pin 23 en cada ciclo del loop(). Si detecta un pulso alto (HIGH), el contador vuelve a 0 y actualiza el display de inmediato, reiniciando también el temporizador.

4. Función Decodificadora: Se creó la función decoder_7seg(int n), la cual recibe el número actual del contador y, mediante un ciclo for, envía los estados lógicos correctos a los 7 pines de salida de forma masiva.

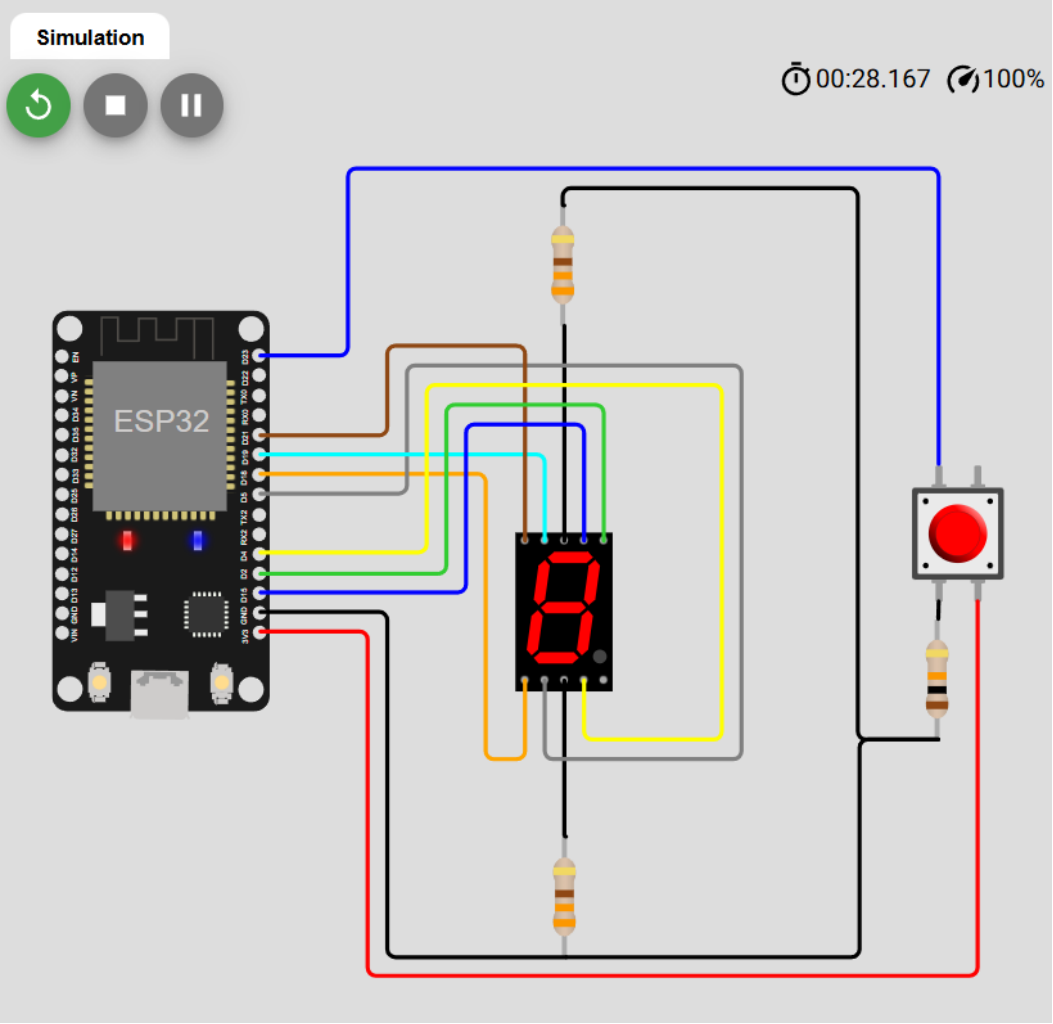

Figura 2. Simulación en Wokwi de un contador hexadecimal con display de 7 segmentos.

5. Resultados

Los dos diseños funcionaron correctamente dentro del simulador. En el primer caso, el LED respondió de inmediato al presionar el pushbutton, lo que permitió comprobar que la entrada y la salida digital estaban configuradas de manera adecuada.

En el segundo caso, el display de siete segmentos mostró la cuenta hexadecimal de forma ordenada. También se comprobó que el botón de reset funcionó correctamente, ya que al presionarlo la cuenta regresaba a cero.

Durante las pruebas se observó que Wokwi facilita mucho la revisión de errores. Si el circuito no funciona, se puede revisar el código, las conexiones o los componentes sin gastar material ni correr el riesgo de dañar algo físico.

También es importante mencionar que, en la versión gratuita, algunas veces la compilación puede tardar un poco debido a la fila de espera. Sin embargo, para prácticas escolares y proyectos de aprendizaje, esta versión es suficiente y cumple bien con su propósito.

6. Conclusiones

Wokwi es una herramienta muy útil para aprender y practicar sistemas embebidos con el ESP32. Su principal ventaja es que permite simular circuitos desde el navegador, sin necesidad de tener todos los componentes físicos desde el inicio.

Los ejemplos desarrollados muestran que esta plataforma ayuda a comprender mejor la relación entre el código y el circuito. En el caso del LED con pushbutton, se pudo observar cómo funciona una entrada y una salida digital. En el contador hexadecimal, se trabajó con varias salidas al mismo tiempo y con una lógica más organizada mediante el uso de millis().

Aunque Wokwi no reemplaza por completo el trabajo en laboratorio, sí sirve como una etapa previa para practicar, probar ideas y corregir errores antes de armar el circuito real. Para estudiantes, esto representa una gran ventaja, porque permite aprender con mayor confianza y sin miedo a dañar componentes.

En general, Wokwi es una buena opción para iniciar en el desarrollo de proyectos con ESP32, especialmente cuando se busca una forma sencilla, accesible y práctica de entender los sistemas embebidos.

Referencias

[1] Espressif Systems. (2025). ESP32 Series Datasheet. Recuperado el 20 de marzo de 2026, de https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[2] Wokwi. (2026). Wokwi – Online Electronics Simulator. Recuperado el 20 de marzo de 2026, de https://wokwi.com