Informe De Laboratorio 2

Diego Andrés Mutis Muñoz, Mateo Hoyos Mesa

Laboratorio de electrónica Digital 3
Departamento de ingeniería y de Telecomunicaciones
Universidad de Antioquia
2024

Resumen

El proyecto comenzó configurando el entorno de desarrollo y ejecutando el primer programa "blink" con la Raspberry Pi Pico W. Luego, se llevó a cabo un análisis del sistema mediante la creación de un diagrama de flujo y un circuito esquemático. Posteriormente, se ensambló el circuito y se desarrolló el código. Se realizaron las primeras pruebas, que incluyeron la lectura de pulsadores, la secuencia de encendido de LEDs y la visualización del tiempo en los displays. Finalmente, se programó el juego de acuerdo con las especificaciones requeridas.

Palabras claves: *RP2040, GPIO, Raspberry Pi Pico W.*

Introducción

El juego de reacción simple es un proyecto interactivo que desafía a los participantes a presionar un botón tan rápido como puedan después de que se encienda uno de tres LEDs. Esta actividad introduce a la programación de microcontroladores, el uso de GPIOs y los conceptos básicos de medición de tiempo en sistemas embebidos de manera divertida y sencilla.

El juego cuenta con tres LEDs de colores diferentes, cada uno con un pulsador asociado, y un cuarto pulsador llamado botón de START. Al iniciar el juego con el botón de START, los LEDs realizan una secuencia de encendido y apagado. Después de un período de espera aleatorio, uno de los LEDs se enciende de forma aleatoria y el jugador debe presionar el pulsador correspondiente lo más rápido posible. Si acierta, se muestra el tiempo de reacción en Displays de 7 segmentos y el juego vuelve al estado inicial. Si no presiona el pulsador a tiempo, se penaliza con 1 segundo. Presionar pulsadores incorrectos no detiene el conteo del tiempo, pero genera una penalidad de 1 segundo.

Marco teórico

De acuerdo con la documentación de la Raspberry Pi Pico W, tanto el voltaje de salida como el funcionamiento de los GPIO son de 3.3 V. [1]

Se utilizaron pulsadores en modo Pull Down, de manera que, al ser presionados, se garantiza un nivel lógico alto a la entrada. Según la documentación, cualquier voltaje mayor a 1.8V se considera un nivel alto. El valor típico de la resistencia utilizado es de 1K. Para los tres leds de diferentes colores se utilizó una resistencia de 330.

El display de siete segmentos consta de ocho Leds, siete de los cuales representan los segmentos "a", "b", "c", "d", "e", "f", "g" y "dp", mientras que uno adicional representa el punto decimal. El control de los cuatro Displays se realiza mediante la técnica de multiplexación. Este método implica encender un solo display, mostrar el número correspondiente y luego apagarlo, repitiendo este proceso con el siguiente display a una alta velocidad. Esto crea la ilusión de que todos los Displays están encendidos simultáneamente. En realidad, se realiza una secuencia de multiplexación, controlando el ánodo o cátodo común del display.

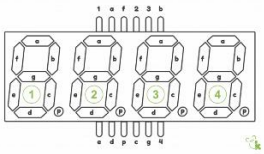

Figura 1. Display7x4Digitos. [2]

El consumo típico de cada segmento del display está entre 10 a 20mA.

$$R_{segmento} = \frac{3.3v}{10mA} = 330\ \Omega$$

Se usaron Displays de siete segmentos de cato común. Para controlar cada display se usarán transistores BJT. Para un transistor 2n2222 se tiene un $\beta = 100\ y\ I_c = 200mA$.

$$I_B = \frac{200mA}{100} = 2mA$$

Como medida de protección, se recomienda duplicar la corriente de base. A la final se utilizo resistencia de 1K para la base del transistor.

$$R_{Base} = \frac{3.3 - 0.7}{4mA} = 650$$

Procedimiento experimental y resultados

En primer lugar, se realizó un análisis del sistema, siguiendo la recomendación del profesor, para determinar las entradas y salidas requeridas en el juego. Teniendo en cuenta este análisis se realizó un diagrama del montaje físico como se muestra en la **Figura 2.** Además de los componentes requeridos mostrados en la **Tabla 1.** Inicialmente, se consideró la posibilidad de utilizar un decodificador como el 7448. Sin embargo, debido a la alimentación de 3.3V, se optó por conectar directamente los ocho segmentos al microcontrolador mediante resistencias. Teniendo así 4 entras y 15 salidas, para un total de 19 pines de la *Raspberry Pi Pico W.*

Entradas:

1. Botón START
2. Botón rojo
3. Botón azul
4. Botón verde

Salidas:

1. Segmento "a"
2. Segmento "b"
3. Segmento "c"
4. Segmento "d"
5. Segmento "e"
6. Segmento "f"
7. Segmento "g"
8. Segmento "dp"
9. Display 1
10. Display 2
11. Display 3
12. Display 4
13. LED rojo
14. LED azul
15. LED verde

.

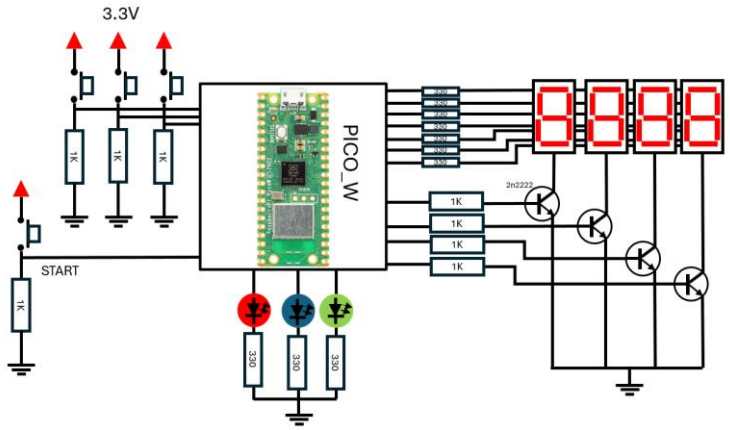

Figura 2. Circuito esquemático.

Tabla 1. Componentes del circuito.

Cantidad	Componente
1	Raspberry Pico W
4	Pulsadores
1	Paquete de resistencia de 1K
1	Paquete de resistencia de 330
1	Paquete de LEDs de colores
4	Transistor 2n2222
1	Display 7 segmentos de 1 digito
1	Display 7 segmentos de 3 digito

Para iniciar con la programación del sistema, primero se instaló el entorno de desarrollo, con el enlace de descarga que se encuentra en la documentación de la Raspberry PI Pico. Esta instalación nos presentó un problema a la hora de realizar la compilación en la tarjeta Pico W, se logró solucionar con la configuración correcta del *CMakeLists* colocando "$set(PICO\_BOARD\ pico\_w)$" en lugar de "$set(PICO\_BOARD\ pico)$".

- Instalación del Pico - Visual Studio Code.
- Configuración del Visual Studio Code para la Raspberry Pi Pico W.
- Se corrió del Blink para la Raspberry Pi Pico W.
- Se descargo el ejemplo del Moodle, para visualizar por consola. (se usó el programa Putty)

Con el Visual Studio Code instalado correctamente se hacen una serie de ejemplos como:

- Leer un pin para prender y apagar un led.
- Mirar valores utilizando solamente un display 7-segmentos
- Multiplexacion de los displays.
- Lectura de pulsadores.

Se empozó a trabajar con la placa raspberry pi pico w y como una de las formas de aprender a programar microcontroladores es mediante la observación de y entendimiento de ejemplos, se realizó la réplica de una serie de estos que nos podían ayudar a entender el funcionamiento de componente importantes que eran necesarios para logar el funcionamiento de este proyecto.

El primer ejemplo realizado fue el de establecer un pin como salida para controlar la intermitencia de un LED. En primera instancia se utilizó la función sleepms(1000) para encender y apagar el LED, pero esta función lo que hace es mandar a dormir el procesador y más adelante nos dimos cuenta que esta forma de trabajar con retados era completamente inservible para la implantación de juego, esto debido a que si mandábamos a dormir el procesador todos los procesos se veían afectados y por ejemplo los displays dejaban de funcionar y se apagaban. Este problema lo solucionamos haciendo uso de timers y atendiendo las diferentes etapas de flujo mediante interrupciones.

Luego, aprendimos a utilizar un solo display 7-segmentos para realizar conteos de 0 a 9 en segundos como se muestra en la **Figura 3**.

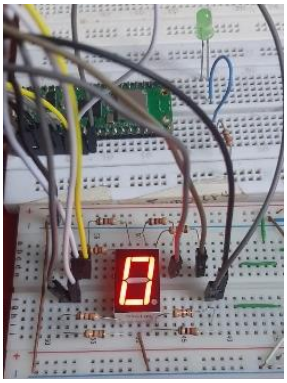

Figura 3. Encendido Displays 7 segmentos

Sin embargo, esto no era suficiente porque necesitábamos visualizar números en 4 displays al mismo tiempo, esto nos llevó a investigar sobre la multiplexación y poner en marcha este método para cumplir con los requerimientos como se observa en la **Figura 4**.

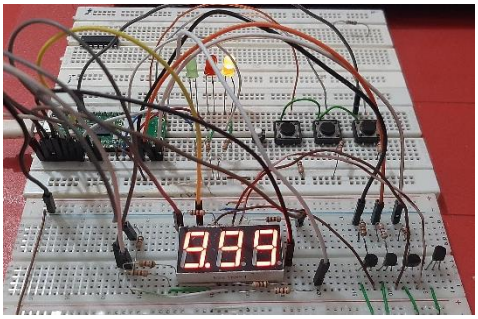

Figura 4. Encendido Displays 7 segmentos

Y como última parte de la realización de ejemplos y pruebas, aprendimos a utilizar botones y capturar sus estados. En esta parte tuvimos un problema y era que al presionar el botón el programa hacia lo que queríamos, pero una vez se dejaba de presionar el flujo de programa cambiaba radicalmente porque la condición ya no se cumplía. Este problema se solucionó mediante la activación de banderas justo en el momento que se leía el estado del botón.

Luego, se elaboró el diagrama de flujo mostrado en la **Figura 5** según la descripción que se pide para el juego y poder empezar a realizar el código en el lenguaje C. Este nos ayudó en el desarrollo del código posteriormente.

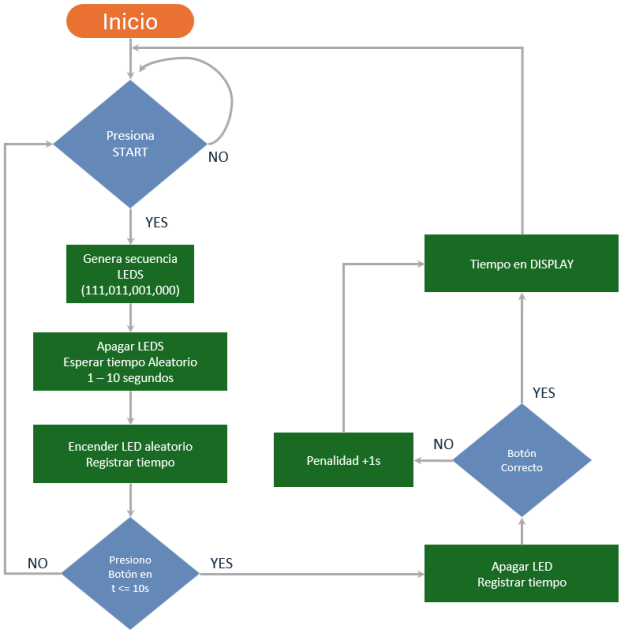

Figura 5. Diagrama de flujo

Conclusiones

- Uno de los principales retos en este proyecto estuvo relacionado con mantener el procesador funcionando en todo momento, es decir, al inicio estábamos utilizando la función sleepms() para realizar cualquier pausa que necesitamos, pero no éramos consientes del error tan grande estamos cometiendo ya que esta función manda a dormir el procesador y todos los procesos se miran afectados traduciéndose en un mal funcionamiento del flujo del programa. Mas halla de todo esto, se puedo lograr el correcto funcionamiento utilizando timers para la activación de interrupciones y logrando que el procesador las atienda exactamente en ese instante. Lo anterior es una herramienta muy poderosa para poder mantener el procesador en constante operación.
- No se han incluido características adicionales en el juego, pero se han considerado posibles mejoras para futuras actualizaciones. Por ejemplo, se podrían agregar niveles de dificultad y diferentes modos de juego, como un modo de memorización de secuencias. Además, se podría implementar sonido y un sistema de puntuación, que se mostraría después de cada intento. También se podría considerar el uso de una pantalla LCD en lugar de Displays de 7 segmentos para una mejor visualización.
- La medición de tiempos de reacción en tiempo real fue esencial durante el desarrollo de la práctica. Nos permitió adquirir habilidades en el manejo del tiempo y en la medición de intervalos precisos utilizando el microcontrolador. Esta introducción a los conceptos de tiempo real nos brindó una comprensión más profunda sobre cómo los sistemas embebidos responden a eventos externos.
- En general, la Raspberry Pi Pico se destaca como una placa de desarrollo de alto rendimiento y bajo costo. Durante el proceso, se profundizó en el conocimiento sobre la programación de la Raspberry Pi Pico y la interacción con los GPIOs para controlar las entradas y salidas.

Bibliografía

[1] Raspberry Pi Foundation. (2024, marzo). Documentación del microcontrolador Raspberry Pi Pico. Recuperado de https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html

[2] Castro, S. (2024, marzo). Multiplexar Display 7 Segmentos con Arduino. Control Automático Educación. Recuperado de https://controlautomaticoeducacion.com/arduino/multiplexar-display-7-segmentos/