Diseño de calculadora didáctica con displays de 7 segmentos

Rojas Pérez José Gabriel [1][0009-0005-1227-7335], Lozano Cruz José Ángel [2][0009-0009-3034-924X] y Jimenez Bonilla Giovani [3][0009-0009-7064-477X]

[1] Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, Puebla 72570, México.

Abstract.
Este proyecto presenta el diseño e implementación de una calculadora didáctica que integra hardware y software mediante una arquitectura cliente-servidor. El sistema utiliza un microcontrolador ESP32 para controlar dos displays físicos de 7 segmentos, los cuales muestran resultados obtenidos a partir de operaciones aritméticas básicas (suma, resta, multiplicación y división), así como funciones de conteo ascendente, descendente, reinicio y detención. La interacción con el usuario se realiza a través de una interfaz gráfica desarrollada en Python, que envía peticiones HTTP al ESP32 para su procesamiento. El desarrollo del hardware incluyó la construcción manual de los displays, el cableado de LEDs y la asignación de puertos en el microcontrolador. Las pruebas confirmaron un correcto funcionamiento en la comunicación, validación de errores y consistencia entre el display virtual y físico. Este trabajo busca fomentar la comprensión de los principios electrónicos y de comunicación que subyacen en sistemas integrados, ofreciendo un enfoque educativo y práctico

Palabras clave:
Calculadora digital, ESP32, Displays , Cliente-servidor, Interfaz, Sistemas embebidos, Comunicación hardware-softwar**e.**

1 Introducción

En los tiempos modernos siempre se ha buscado la eficiencia y la optimización de herramientas para nuestro día a día, en este caso en el tema principal de nuestro proyecto, la calculadora es una herramienta que ha estado a lo largo de la historia humana iniciando mecánicamente hasta nuestra era que ya es digital gracias a la invención de los chip, hoy en día el uso más común de estas calculadoras son digitales con interfaces graficas, hay de distintos tipos de calculadoras hoy en día están desde las más básicas que pueden hacer operaciones simples como sumar, multiplicar, dividir y restar , hasta más especializadas dependiendo de los usos y necesidades de la personas.

Hoy en día, la mayoría de las personas utiliza calculadoras digitales integradas en smartphones sin considerar los principios electrónicos y de comunicación que las

sustentan. Proyectos básicos basados en matrices de LEDs y microcontroladores reviven el interés por entender estos fundamentos. Este trasfondo sitúa el valor educativo de recrear una calculadora con hardware accesible y software libre, que sirva de puente entre teoría y práctica.

A pesar de la abundancia de información acerca de este presente proyecto como tutoriales, información acerca de los microcontroladores, leds, cables. La comercialización de displays de leds ya hechos con funciones diferentes y económicos, pocos profundizan en la arquitectura por capaz que se utiliza, la interacción con el hardware, el uso de la herramienta multifuncional (esp 32). Analizar y profundizar en estos contextos nos permite entender a detalle el ciclo y el flujo de los datos desde que se da el clic en la interfaz, hasta donde se muestra el resultado en el display, en este caso en la calculadora.
En el presente proyecto se propone desarrollar una calculadora que muestre resultados en los dos displays, se profundizara más adelante, procederé a describir generalmente como están conformados y su distribución, esto para lograr un mayor entendimiento general en dado caso que no se tenga conocimiento técnico. Son siete segmentos de leds soldados entre si con un cable, controlado por un esp 32 donde la interfaz o GUI nos permitirá ingresar operaciones como suma, resta, multiplicación y división, así como iniciar conteos ascendentes, descendentes o detener y reiniciar la pantalla.

Finalmente, para concluir esta introducción cabe destacar el uso de diferentes disciplinas, la conexión y el uso de la arquitectura en capaz que es la mejor herramienta de optimización y organización que hemos estado aplicando con la finalidad de garantizar modularidad y escalabilidad en el proyecto.

2. Estado del arte

En base con nuestro proyecto de la implementación de una calculadora buscamos proyectos que abarquen el enfoque de lo que hemos estado realizando en clase, por lo tanto, revisando proyectos encontramos que: una de las formas más eficientes para controlar un display de siete segmentos sin utilizar todos los pines de un microcontrolador es mediante un registro de desplazamiento como el 74HC595 (ACM kits, 2020). El proyecto consiste en construir una calculadora simple controlada por una placa Arduino. Este sistema recibe los números y las operaciones a través de un teclado matricial (keypad) y muestra los resultados en un display de siete segmentos. Para hacer el control del display más eficiente y ahorrar pines de conexión en el Arduino, se utiliza un chip clave, el 74HC595.

Posteriormente, se revisó un video de YouTube que pertenecía al canal Muy Fácil De Hacer, este video fue realizado en 2016, que fue la base en la que nos inspiramos en el diseño principal de nuestro proyecto, el proceso comienza construyendo el cuerpo de los siete segmentos con una plantilla de cartón, para luego instalar luces LED en su interior. A continuación, se realiza el trabajo eléctrico, soldando resistencias a los LEDs y conectando cables para controlar cada segmento de forma independiente. Finalmente,

se sella y se le da un acabado estético al display para difuminar la luz, y se comprueba su correcto funcionamiento conectándolo a una placa Arduino.

Finalmente, la construcción de calculadoras utilizando microcontroladores y periféricos como teclados matriciales y displays de siete segmentos es un proyecto recurrente en la comunidad de electrónica y sistemas embebidos. Un ejemplo de esto es el trabajo documentado por Franller Vásquez (2020), donde se presenta una calculadora funcional que realiza operaciones aritméticas básicas y muestra los resultados en displays.

Tabla 1. Comparación de trabajos relacionados con la calculadora.

Autor	Año	Descripción del Proyecto	Aporte Principal	Relación con el Proyecto
ACM kits	2020	Creación de una calculadora simple con Arduino, teclado matricial y display de 7 segmentos.	Uso del chip 74HC595 (registro de desplazamiento) para controlar el display de forma eficiente y ahorrar pines.	Aporta una técnica clave de hardware para optimizar el control del display.
Muy Fácil De Hacer	2016	Tutorial para construir un display casero de 7 segmentos usando cartón, LEDs y resistencias.	Se enfoca en la construcción física y "DIY" del componente de visualización, no en la lógica de la calculadora.	Sirvió como inspiración principal para el diseño y la construcción física del display.
Franller Vasquez	2020	Presentación de una calculadora funcional que realiza operaciones aritméticas básicas con teclado y displays.	Valida el concepto de la calculadora con estos periféricos como un proyecto recurrente y viable en la comunidad.	Confirma la relevancia del proyecto y sirve como un ejemplo de implementación completa.

3. Metodología

Este proyecto consiste en la implementación de una calculadora que, a través de una interfaz gráfica realizada en Python, permite efectuar operaciones básicas como suma, resta, multiplicación y división, mostrando de manera clara los resultados obtenidos. Además, el proyecto incorpora la representación de los resultados en dos displays de 7 segmentos. Estos se controlan mediante una matriz de codificación que traduce cada número en el encendido de los segmentos correspondientes, logrando una salida física que complementa la visualización digital en pantalla. Así logrando la interacción de un sistema que contiene un software y un hardware que nos permite interactuar de manera gráfica con el usuario.

3.1. Desarrollo de display de 7 segmentos

Para realizar este proyecto nos basamos en el diseño propuesto por el canal de Muy Fácil De Hacer, donde comenzamos creando una plantilla de un display de 7 segmentos que obtuvimos a través de internet en la cual a través de un papel cascarón plasmamos el diseño.

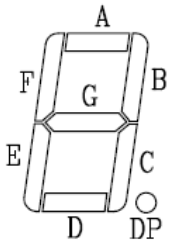

Fig. 1. *Plantilla de un display de 7 segmentos*

Para permitir el funcionamiento de los LEDs en cada segmento, se realizaron dos orificios en la parte central de cada uno. Estos orificios sirven para insertar las patillas positiva y negativa del LED, lo que facilita tanto su conexión como la medición de los espacios necesarios para el correcto acomodo de los terminales negativos. De esta manera, se logra una distribución ordenada que garantiza la conducción eléctrica adecuada y la correcta visualización de los números en el display.

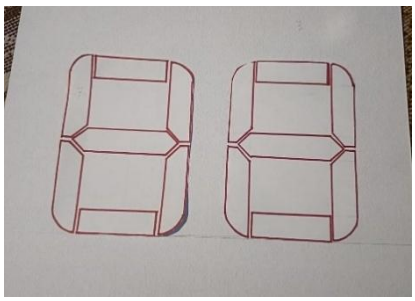

Fig. 2. *Vista superior del trazado de dos displays de 7 segmentos antes de la colocación de LEDs.*

Ya colocados de manera correcta, comenzamos a soldar los puntos negativos a una salida de tierra que posteriormente se va a conectar al GND, al igual que se soldaron las salidas positivas a cada cable macho donde posteriormente con ayuda de cables hembras íbamos a conectarlos a cada pin de nuestro ESP32.

3.1.1. Asignación de puertos

Para lograr que nuestra placa ESP32 pueda identificar que puertos van a ser parte de nuestras decenas (primer display) y nuestras unidades (segundo display), asignamos los siguientes puertos en base a sus segmentos, al igual que por cada segmento definimos un color para no confundirnos a la hora de colocarlos en los puertos y que los segmentos se mal acomoden.

Tabla 2. Asignación de puertos en base a sus segmentos y su color.

Display	a	b	c	d	e	f	g
Decenas	13	14	27	32	16	33	17
Unidades	23	22	21	19	18	4	5

3.1.2. Colocación de puertos

En base con la fig.1 empezamos a colocar los cables hembra en base a su puerto correspondiente con la ayuda de la tabla 2, para posteriormente hacer las pruebas del conteo ascendente y descendente.

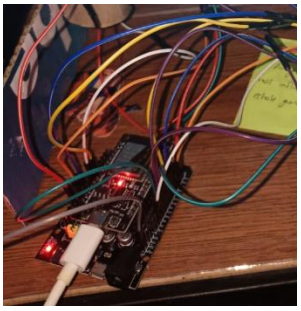

***Fig. 3**. Montaje del microcontrolador ESP32 con las conexiones hacia los displays de 7 segmentos.*

3.1.3. Display de 7 segmentos

Después de finalizar el armado de la estructura y la conexión del cableado, se obtuvo el primer diseño físico del display de 7 segmentos. Para simular la apariencia de un display digital, se aplicó pintura negra sobre la superficie, lo que permitió darle un aspecto más realista. Posteriormente, se recortaron figuras en hojas de opalina que representaban cada segmento, creando la sensación de que cada uno contaba con una pared o una caja independiente. Finalmente, se colocó una capa de papel en la parte superior para cubrir los acabados y lograr un diseño más uniforme y estético, resaltando la forma de los números que se mostrarán en el prototipo.

***Fig. 4**. Construcción manual de los displays de 7 segmentos listos para la instalación de los componentes electrónicos.*

Fig. 5. *Vista del prototipo ensamblado con carcasa protectora y conexiones hacia el microcontrolador.*

Una vez concluido el prototipo, se obtuvo una versión preliminar funcional en la que fue posible visualizar con claridad los números en el display, gracias a la correcta programación de los segmentos. Esta etapa permitió comprobar el adecuado encendido de los LEDs, descartando posibles fallas de conexión o errores en la ubicación de los componentes.

Fig. 6*. Visualización del número 25 en el prototipo del display de 7 segmentos.*

3.2. Desarrollo de la interfaz de la calculadora

De primera mano definimos los estilos globales de la interfaz del display el color de fondo, el color de texto y los estilos de letra que se usaran como se muestra en la imagen

Fig. 7. *Interfaz de la calculadora en Python.*

Esto se define al inicio de la programación para que no se pierda el formato.

```
BG_COLOR = "#f4f4f4"
BTN_COLOR = "#4CAF50"
BTN_COLOR_HOVER = "#45a049"
BTN_TEXT_COLOR = "white"
FONT_MAIN = ("Segoe UI", 12)
FONT_DISPLAY = ("Courier", 28, "bold")
```

Fig. 8. *Definición de los colores de los botones.*

Botón de entradas de valores:

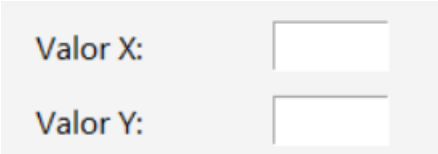

Fig. 9. *Botones de entrada para valores.*

En este caso se define la ubicación de los botones el color como se muestra en la fig. 10.

```
tk.Label(root, text="Valor X:", bg=BG_COLOR, font=FONT_MAIN).grid(row=0, column=0, padx=5, pady=5, sticky="e")
tk.Entry(root, textvariable=self.var_x, width=6, font=FONT_MAIN).grid(row=0, column=1, pady=5)

tk.Label(root, text="Valor Y:", bg=BG_COLOR, font=FONT_MAIN).grid(row=1, column=0, padx=5, pady=5, sticky="e")
tk.Entry(root, textvariable=self.var_y, width=6, font=FONT_MAIN).grid(row=1, column=1, pady=5)
```

Fig. 10. *Ubicación de botones en la interfaz.*

Display:
Se agrega un pequeño display virtual que simula el resultado que mostrará el display físico.

Fig. 11. *Display virtual que muestra los resultados.*

En la fig. 12 se muestra la configuración de los botones dependiendo sus clases, colores, posición.

```
tk.Label(root, text="Display:", bg=BG_COLOR, font=FONT_MAIN).grid(row=2, column=0, padx=5, pady=5, sticky="e")
tk.Label(root, textvariable=self.display_val, width=5, bg="black", fg="lime",
         font=FONT_DISPLAY, relief="sunken").grid(row=2, column=1, pady=5)
```

Fig. 12. *Configuración de botones.*

3.2.1. Botones de operaciones

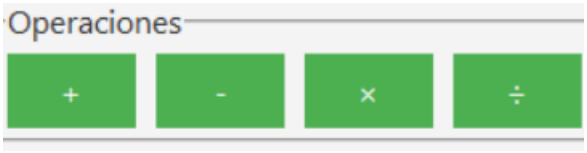

Fig. 13. *Diseño de la interfaz de los botones.*

En este bloque de la interfaz se tiene las opciones básicas que se necesitan para el display donde se dividió como se muestra en la fig. 14.

```
ops = tk.LabelFrame(root, text="Operaciones", bg=BG_COLOR, font=FONT_MAIN, fg="#333")
ops.grid(row=3, column=0, columnspan=2, pady=10)

self._crear_boton(ops, "+", self.sumar, 0, 0)
self._crear_boton(ops, "-", self.restar, 0, 1)
self._crear_boton(ops, "×", self.multiplicar, 0, 2)
self._crear_boton(ops, "÷", self.dividir, 0, 3)
```

Fig. 14. *Definición de las operaciones aritméticas.*

Botón sumar

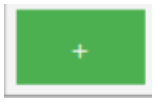

Fig. 15. *Botón sumar.*

Al hacer clic, ejecuta la función sumar ().
Esta función llama a leer_operandos () para obtener y validar los valores X e Y (o usar el último resultado guardado en memoria si un campo está vacío).
Si los valores son correctos, envía una petición HTTP al ESP32 con el endpoint /sumar y los parámetros x y y.
El ESP32 realiza la suma y devuelve el resultado, que se muestra en el display virtual (display_val) y se guarda en memoria.

Botón restar

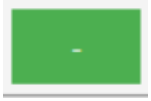

Fig. 16. *Botón restar.*

Ejecuta restar (), que obtiene y valida los operandos y el resultado se muestra en el display y se guarda en memoria.

Botón multiplicación

Fig. 17. *Botón multiplicar.*

Este botón ejecuta multiplicar (), que obtiene y valida los operandos y el resultado se muestra en el display y se guarda en memoria.

Botón división

Fig. 18. *Botón división.*

Este botón ejecuta dividir (), que obtiene y valida los operandos antes de enviar la petición, verifica que el divisor (Y) no sea cero, el resultado se muestra en el display y se guarda en memoria.

3.2.2. Lógica del bloque de operaciones

```
def sumar(self):
    x, y = self.leer_operandos()
    if x is not None:
        self.enviar("sumar", {"x": x, "y": y})

def restar(self):
    x, y = self.leer_operandos()
    if x is not None:
        self.enviar("restar", {"x": x, "y": y})

def multiplicar(self):
    x, y = self.leer_operandos()
    if x is not None:
        self.enviar("multiplicar", {"x": x, "y": y})

def dividir(self):
    x, y = self.leer_operandos()
    if x is not None:
        if y == 0:
            messagebox.showwarning("Operación inválida", "No se puede dividir entre cero.")
        else:
            self.enviar("dividir", {"x": x, "y": y})
```

Fig. 19. *Lógica de programación para las funciones aritméticas.*

En este caso al llamar a las funciones ya encapsuladas en cada botón lo que se requieren son los valores x, y, que se tienen que dar por el usuario y esto a su vez manda la petición al esp 32 y verifica en caso de que haya algún error.

3.2.3. Bloque de botones de control

Fig. 20. *Interfaz de control del display.*

En este bloque se dividen los controles que se harán en el display en la fig. 21 se define su tamaño nombre y las funciones que van a llamar.

```
ctrl = tk.LabelFrame(root, text="Control", bg=BG_COLOR, font=FONT_MAIN, fg="#333")
ctrl.grid(row=4, column=0, columnspan=2, pady=10)

self._crear_boton(ctrl, "Ascendente", lambda: self.enviar("ascendente"), 0, 0, ancho=12)
self._crear_boton(ctrl, "Descendente", lambda: self.enviar("descendente"), 0, 1, ancho=12)
self._crear_boton(ctrl, "Detener", lambda: self.enviar("detener"), 1, 0, ancho=12)
self._crear_boton(ctrl, "Reset", lambda: self.enviar("reset"), 1, 1, ancho=12)
```

Fig. 21. *Implementación de los botones de control.*

Donde:

- El botón ascendente: envía una petición HTTP al ESP32 con el endpoint /ascendente. El ESP32 inicia un conteo ascendente en el display físico y devuelve el valor actual al display virtual.

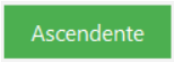

Fig. 22. *Botón de conteo ascendente.*

- Botón Descendente: el botón descendente envía una petición HTTP al ESP32 con el endpoint el ESP32 inicia un conteo descendente en el display físico y devuelve el valor actual al display virtual.

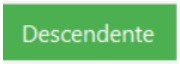

Fig. 23. *Botón de conteo descendente.*

- Botón detener: el botón detener envía una petición HTTP al ESP32 con el endpoint el ESP32 detiene cualquier conteo activo y mantiene el último valor mostrado.

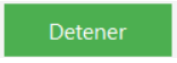

Fig. 24. *Botón para detener el conteo.*

- Botón Reset: el botón envía una petición HTTP al ESP32 con el endpoint y el ESP32 reinicia el display físico a “00” y actualiza el display virtual.

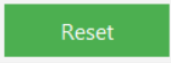

Fig. 25. *Botón para reiniciar conteo.*

3.3. Diagrama de flujo del proyecto

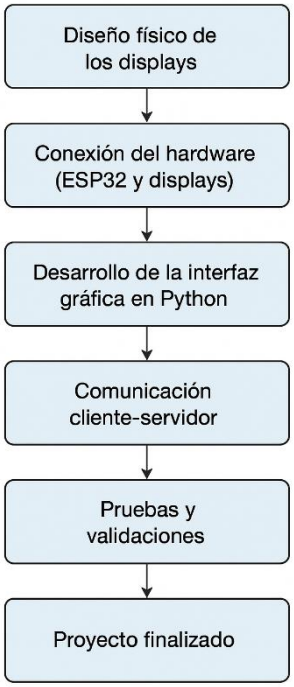

Fig. 26. *Diagrama de flujo de la comunicación entre la calculadora y los displays.*

El proyecto siguió un ciclo de implementación con las siguientes fases:

1. **Diseño físico de los displays**. Creación de la plantilla, perforación, colocación de LEDs y conexión de terminales
   .
2. **Conexión del hardware (ESP32 y displays)** – Asignación de puertos para decenas y unidades, cableado y soldadura de LEDs.

3. **Desarrollo de la interfaz gráfica en Python** – Definición de botones, colores, display virtual y lógica de operaciones.

4. **Comunicación cliente-servidor** – La interfaz envía peticiones HTTP al ESP32, que procesa las operaciones y responde.

5. **Pruebas y validaciones** – Se verificó conteo ascendente/descendente, operaciones aritméticas y manejo de errores.

3.4. Diagrama cliente servidor.

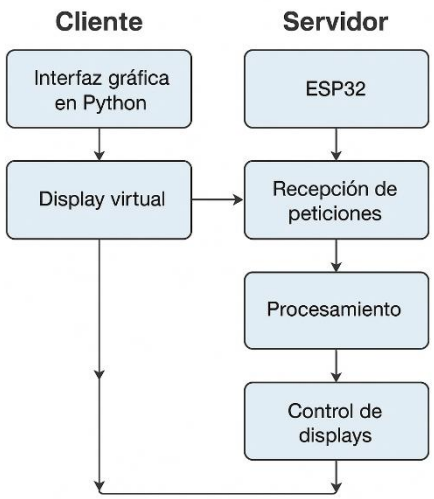

Fig. 27. *Diagrama de flujo Cliente – Servidor.*

El sistema funciona bajo una arquitectura **cliente-servidor**:

- **Cliente:**
  - La **interfaz gráfica en Python** que recibe datos del usuario.
  - Muestra resultados en un display virtual.
  - Envía peticiones HTTP al microcontrolador ESP32.
- **Servidor (ESP32):**
  - Recibe las peticiones (sumar, restar, multiplicar, dividir, ascendente, descendente, detener, reset).
  - Procesa los cálculos y controla los **displays físicos de 7 segmentos**.
  - Devuelve el resultado al cliente (interfaz).

3.5. Diagrama de implementación de hardware y software.

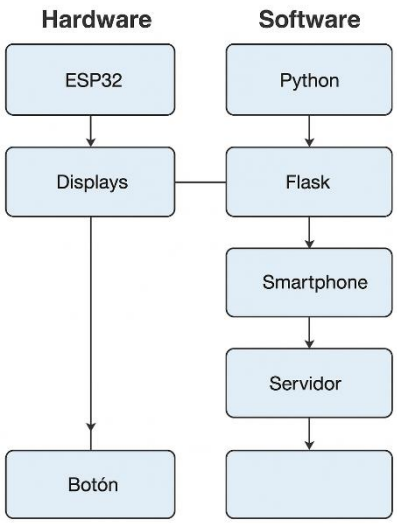

Fig. 28. *Diagrama de flujo de la implementación de hardware y software.*

Hardware:

- **Microcontrolador:** ESP32.
- **Displays físicos:** Dos de 7 segmentos construidos con LEDs, resistencias y cableado.
- **Conexiones:** Cableado entre segmentos y ESP32 con asignación de puertos para decenas y unidades.
- **Carcasa protectora:** Para mejorar estética y resistencia del prototipo.

Software:

- **Cliente (Python):**
  - Interfaz gráfica con botones para operaciones básicas.
  - Display virtual que refleja el resultado.
  - Manejo de errores (división entre 0, fuera de rango).
- **Servidor (ESP32):**
  - Lógica de encendido/apagado de segmentos.
  - Recepción de peticiones HTTP.

- Ejecución de operaciones matemáticas y control del display físico.

4. Pruebas

Como primera prueba se empezó el conteo ascendente dado en el bloque de control se observará en la figura 29 y figura 30.

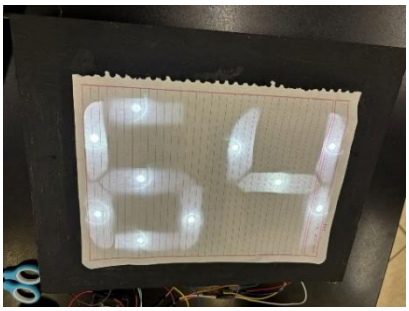

Fig. 30. *Conteo ascendente con número 64.*

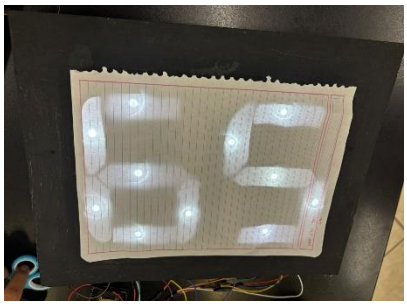

Fig. 29. *Conteo ascendente con número 65.*

Ahora se probará el control de conteo descendente, se podrán observar en la figura 31 y 32.

Fig. 31. *Conteo descendente con número 86.*

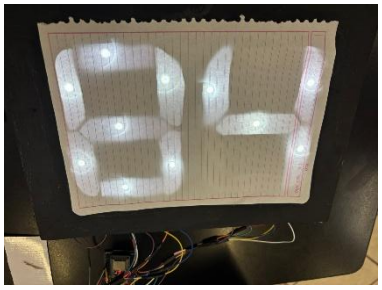

Fig. 33. *Conteo descendente con número 84.*

Siguiendo en los bloques de control se probarán el de reset que lleva el display a un estado de 00, se observa en la figura 33 y figura 34.

Fig. 32. *Interfaz con display virtual en 00.*

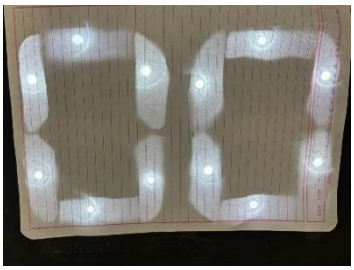

Fig. 34. *Display físico en 00.*

Ahora se probarán con los botones de las operaciones suma resta y división en donde también se juntará con el uso de la memoria, es decir se tendrán que guardar los resultados. Las operaciones que se harán serán una suma de 5 + 5 después una resta de –3 una multiplicación por dos y al último una división entre 2. Esto se observará en cada imagen de cada paso:

1. Suma.

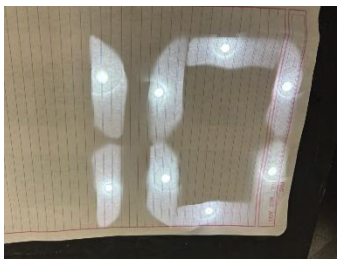

Fig. 35. *Suma de 5 + 5.*

2. Resta.

Fig. 36. *Resta de 10 -3.*

3. Multiplicación

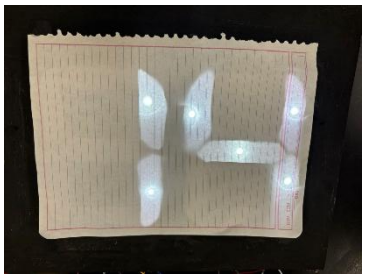

Fig. 37*. Multiplicación de 7 x 2.*

4. División

Fig.38. *División de 7 / 2.*

5. Resultados de operaciones invalidas. En este caso como es un display de dos dígitos no se puede obtener resultados fuera de 0 y 99, división entre 0 por lo tanto la interfaz gráfica lanzara un error como se ve en la figura 39 y en la figura 40 el error de que no se puede dividir entre 0.

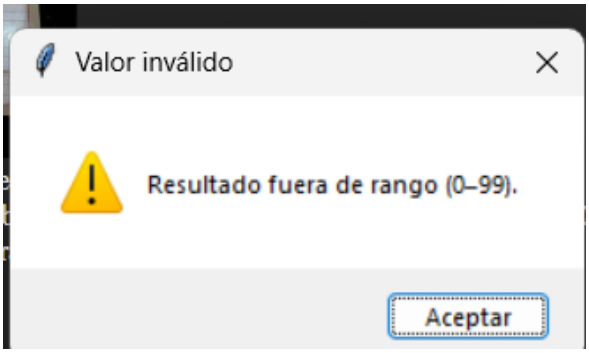

Fig.39. *Alerta de fuera de rango.*

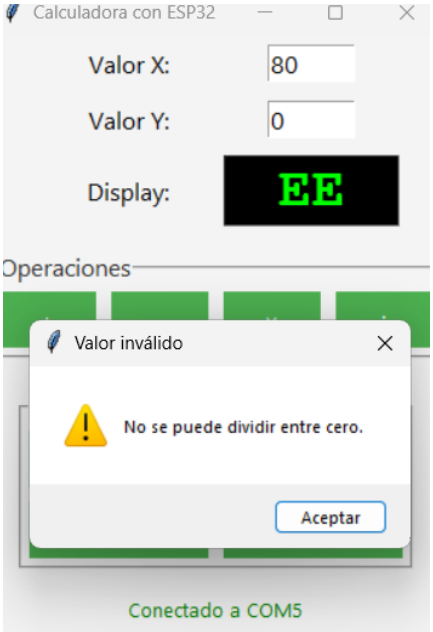

Fig.40. *Alerta de error al dividir entre 0.*

5. Resultados

5.1. Resultados obtenidos

Durante las pruebas funcionales se verificó el correcto desempeño de los dos módulos principales de la interfaz:

5.1.2. Bloque de Operaciones

1. Se ingresaron diferentes combinaciones de valores dentro del rango permitido (0–99).
2. Las operaciones de suma, resta, multiplicación y división arrojaron resultados correctos y coherentes con los cálculos esperados.
3. En casos de división entre cero o resultados fuera de rango, la interfaz mostró el mensaje de error “EE” y el ESP32 replicó la indicación en el display físico, cumpliendo con la validación establecida.

5.2.2. Bloque de Control

Los comandos **Ascendente**, **Descendente**, **Detener** y **Reset** fueron enviados correctamente al ESP32.

El display físico respondió de forma correcta las siguientes opciones:

1. **Ascendente**: incremento secuencial del valor mostrado.
2. **Descendente**: decremento secuencial.
3. **Detener**: pausa del conteo manteniendo el valor actual.
4. **Reset**: reinicio del display a “00”.

6. Conclusiones

El proyecto permitió implementar una calculadora digital que integra una interfaz gráfica en Python con un sistema físico de dos displays de 7 segmentos controlados por un microcontrolador ESP32. La correcta codificación de los segmentos y la asignación de puertos garantizó que cada número pudiera representarse con precisión mediante el encendido de los LEDs correspondientes. El diseño físico de los displays, elaborado con plantillas, perforaciones y conexiones soldadas, junto con la programación de operaciones aritméticas básicas, logró un funcionamiento estable y confiable.

El ensamblaje final, complementado con acabados estéticos y una carcasa protectora, permitió obtener un prototipo funcional que muestra de forma clara los resultados en pantalla y en los displays. Las pruebas realizadas confirmaron el adecuado flujo de datos desde la interfaz gráfica hasta la salida física, descartando errores de conexión y validando la lógica de control implementada. De esta manera, el proyecto demostró la viabilidad de integrar software y hardware en un sistema sencillo, funcional y escalable.

Referencias

1. Equipos ACM. (2020, 14 de agosto). *CALCULADORA BÁSICA CON DISPLAY DE SIETE SEGMENTOS Y KEYPAD||LIBRERIA KEYPAD.H||74HC595||ARDUINO* [Video]. YouTube. https://www.youtube.com/watch?v=fg67hOIgMPY

2. Muy Fácil De Hacer. (2016, 8 de marzo). *Cómo Hacer un Display de 7 Segmentos* [Video]. YouTube. https://www.youtube.com/watch?v=xwL4EoiB8II

3. Vasquez, F. (2020, 2 de mayo). CALCULADORA CON TECLADO MATRICIAL Y SIMPLES DISPLAYS [Video]. YouTube. https://www.youtube.com/watch?v=hHyMI6kAz3Y