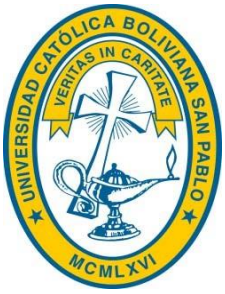


Universidad Católica Boliviana “San Pablo”

Facultad de Ingeniería

DIPLOMADO INTERNET OG THINGS

(3RA VERSIÓN)

INFORME PROYECTO FINAL

MODULO 3: SISTEMAS EMBEBIDOS

Realizado por:

RAUL RUBEN PACHECO SANDOVAL

2025

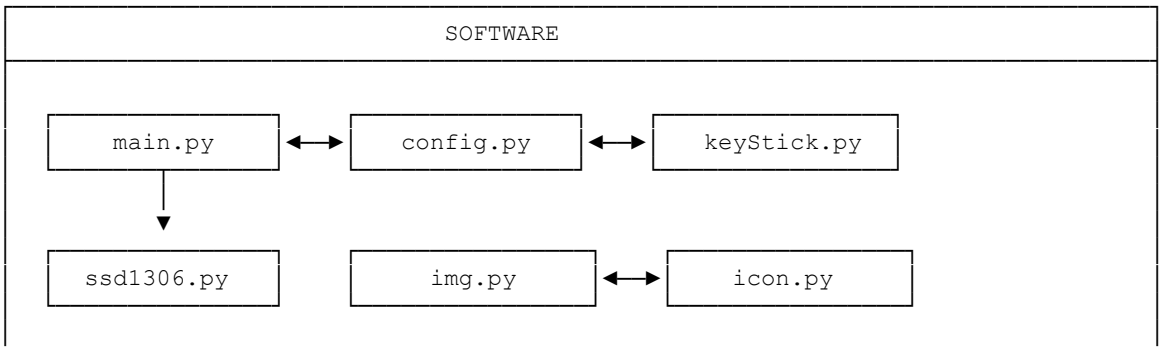
CONTENIDO

INFORME DEL PROYECTO FINAL

Este informe detalla la estructura y funcionalidad de un programa diseñado para un dispositivo MicroPython, presumiblemente una Raspberry Pi Pico con una pantalla OLED y un joystick. El sistema opera como un centro de actividades multifuncional, ofreciendo diversas "modos" o aplicaciones, incluyendo reloj digital y analógico, cronómetro, calculadora gráfica, calendario, juego de Sokoban y sensor de temperatura.

DIAGRAMA DE BLOQUES

RASPBERRY PI PICO				
PERIFÉRICOS	ENTRADAS	SALIDAS	INTERFACES	COMUNICACIÓN
Pantalla OLED (SSD1306)			I2C: - GPIO4(SDA) - GPIO5(SCL)	
Joystick Analógico	GPIO26(ADC0) (Eje X) GPIO27(ADC1) (Eje Y) GPIO15 (Botón)		ADC: - Eje X - Eje Y Digital In: - Botón	
Buzzer		GPIO14(PWM)	PWM: - Sonido	
Sensor Ultrasónico (HC-SR04)	GPIO8(Echo)	GPIO7(Trig)	Digital: - Trigger - Echo	
LED RGB		GPIO16(PWM) (Rojo) GPIO17(PWM) (Verde) GPIO18(PWM) (Azul)	PWM: - Rojo - Verde - Azul	
Sensor Temperatura (DHT22)	GPIO19(ADC4)		ADC: - Temp	



MÓDULOS DE APLICACIÓN			
modo_reloj.py	modo_crono.py	modo_calculadora	modo_calendario
modo_reloj_ analogico.py	modo_sokoban.py	modo_ temperatura.py	

Explicación del diagrama:

Bloque de Hardware (Raspberry Pi Pico):

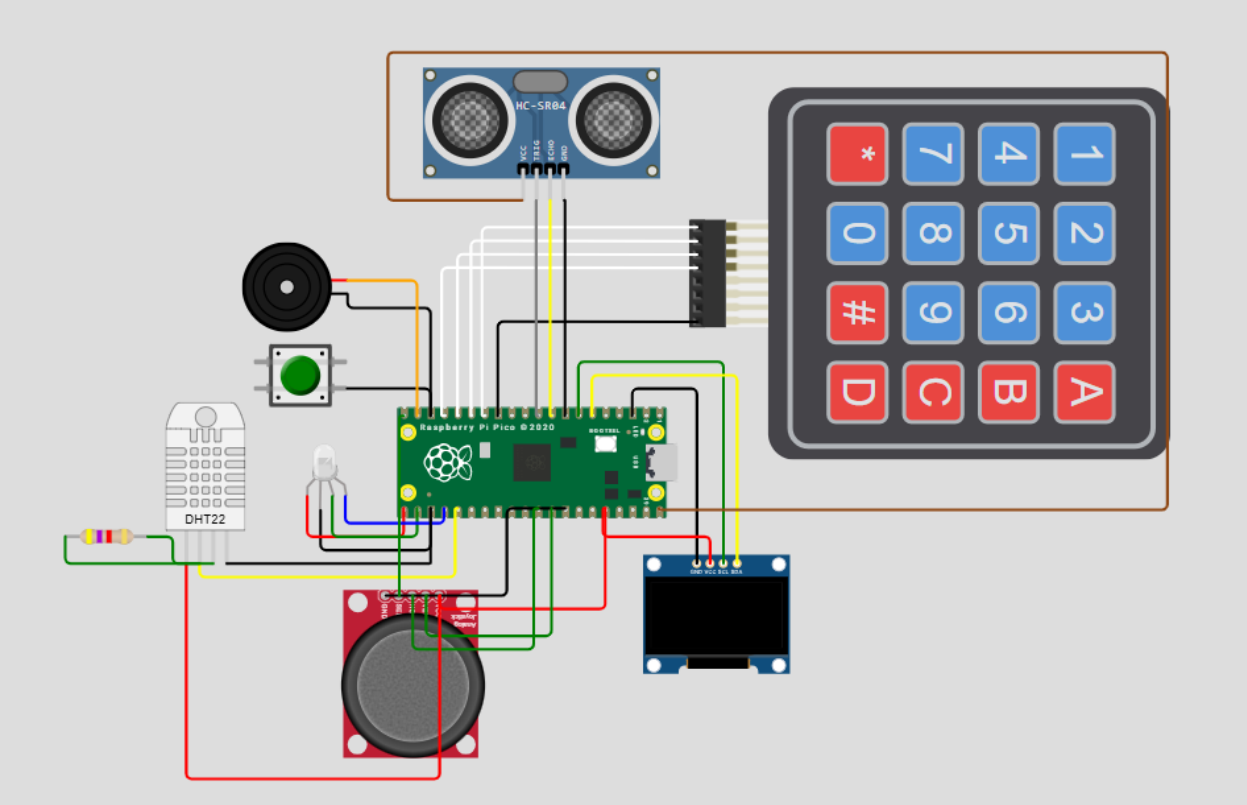
1. **Periféricos**: Todos los componentes conectados
2. **Entradas**: Pines configurados como entradas (joystick, botón, sensor ultrasónico)
3. **Salidas**: Pines configurados como salidas (buzzer, LED RGB)
4. **Interfaces**: Tipo de comunicación utilizada (I2C, ADC, PWM, etc.)
5. **Comunicación**: Reservado para posibles interfaces de comunicación adicionales

Bloque de Software:

1. **Núcleo del sistema**:
    - main.py: Coordina todo el sistema
    - config.py: Configuración centralizada
    - keyStick.py: Gestión del joystick y botón
2. **Drivers y recursos**:
    - ssd1306.py: Driver para la pantalla OLED
    - img.py e icon.py: Recursos gráficos
3. **Módulos de aplicación**:
    - 8 módulos que implementan las diferentes funcionalidades
    - Cada uno sigue el mismo patrón de interfaz con el sistema principal

El diagrama muestra cómo todos los componentes físicos se conectan a la Raspberry Pi Pico a través de diferentes interfaces, y cómo el software se organiza en capas, con el sistema principal coordinando todo y los módulos de aplicación implementando funcionalidades específicas.

ESQUEMA ELECTRÓNICO



Componente	Pines Raspberry Pi Pico	Descripción
Pantalla OLED I2C		
SDA	GPIO4	Pin de datos I2C (Serial Data Line)
SCL	GPIO5	Pin de reloj I2C (Serial Clock Line)
Dirección I2C	0x3C	Dirección I2C del módulo OLED SSD1306
Joystick Analógico		
Eje X	GPIO26 (ADC0)	Entrada analógica para el eje X
Eje Y	GPIO27 (ADC1)	Entrada analógica para el eje Y
Buzzer	GPIO14	Salida PWM para el control del buzzer
Botón	GPIO15	Entrada digital para el botón

Sensor Ultrasónico (HC-SR04)		
TRIG	GPIO7	Pin de disparo (Trigger)
ECHO	GPIO8	Pin de eco (Echo)
LED RGB		Pines PWM para control individual de cada color
Rojo	GPIO16 (PWM)	Canal PWM para el color Rojo
Verde	GPIO17 (PWM)	Canal PWM para el color Verde
Azul	GPIO18 (PWM)	Canal PWM para el color Azul
Sensor de Temperatura	**GPIO19 (ADC4)**	**Acceso al sensor de temperatura DHT22**

EXPLICACION DEL CODIGO

Resumen del Sistema Principal (main.py)

El archivo main.py actúa como el orquestador principal del sistema. Es responsable de la inicialización del hardware, la gestión del menú de selección de modos y la ejecución de cada modo específico.

Funcionalidades Clave:

- Inicialización de Hardware: Configura la pantalla OLED (SSD1306 vía I2C) y el buzzer (PWM). Utiliza el archivo config.py para obtener los pines y parámetros de configuración.
- Gestión de Modos: Define una lista de "modos" disponibles, cada uno con un nombre, el módulo Python asociado y un icono.
- Menú Interactivo: Presenta un menú en la pantalla OLED que permite al usuario navegar entre los diferentes modos utilizando el joystick.
- Carga Dinámica de Módulos: Importa y ejecuta dinámicamente el módulo correspondiente al modo seleccionado por el usuario. Esto permite una estructura modular y extensible.
- Manejo de Errores: Incluye un mecanismo básico para capturar y mostrar errores si un módulo falla al cargar o ejecutar.
- Funciones Compartidas: Pasa objetos y funciones comunes (como oled, get_direction del joystick y play_sound del buzzer) a los módulos de cada modo, lo que promueve la reutilización de código y evita la reinicialización de hardware.

- Pantallas de Inicio: Muestra pantallas de bienvenida y carga al iniciar el dispositivo.

Componentes Principales:

- **main.py: Lógica central, inicialización, menú y despacho de modos.**
- **config.py: Archivo de configuración de hardware (pines, umbrales, etc.).**
- **keyStick.py: Módulo para la lectura del joystick y detección de eventos (direcciones y doble clic).**
- **ssd1306.py: Driver para la pantalla OLED.**
- **img.py: Almacena datos de imágenes (fondo, logo) utilizadas en la interfaz.**
- **icon.py: Almacena datos de iconos para cada modo en el menú.**

ARCHIVOS DE CONFIGURACIÓN Y UTILIDADES

config.py

Este archivo centraliza todas las configuraciones de hardware y parámetros operativos del dispositivo. Su propósito es facilitar la modificación de pines, umbrales y otras constantes sin necesidad de alterar la lógica principal del programa.

Detalles:

- Pantalla OLED (I2C): Define la ID de la interfaz I2C, pines SDA/SCL, dimensiones de la pantalla (ancho y alto) y dirección I2C.
- Joystick (Analógico): Configura los pines ADC para los ejes X e Y del joystick, así como umbrales de lectura para detectar movimientos en diferentes direcciones y un valor de centro para calibración.
- Buzzer: Especifica el pin del buzzer, su frecuencia base y ciclo de trabajo para la generación de sonido.
- Botón: Define el pin del botón, utilizado principalmente para la selección o confirmación.
- Sensor Ultrasónico (HC-SR04): Configura los pines ECHO y TRIG para un sensor de distancia.
- LED RGB: Define los pines PWM para los canales rojo, verde y azul de un LED RGB.

- Conexión WiFi: Incluye placeholders para el SSID y la contraseña de la red WiFi, necesarios para funcionalidades que requieran conexión a internet (como la sincronización NTP).

keyStick.py

Módulo dedicado al procesamiento de la entrada del joystick y el botón.

Detalles:

- Inicialización: Configura los pines ADC para el joystick y el pin de entrada para el botón.
- Detección de Dirección: Lee los valores analógicos del joystick y los compara con los umbrales definidos en config.py para determinar la dirección (arriba, abajo, izquierda, derecha).
- Detección de Doble Clic: Implementa una lógica para detectar un doble clic en el botón central del joystick, lo cual se utiliza para salir de los modos y regresar al menú principal.
- Debounce: Incluye un retardo para evitar múltiples lecturas del botón debido al rebote mecánico.

icon.py

Contiene los datos de imagen en formato bytearray para los iconos que se muestran en el menú de la pantalla OLED.

Detalles:

- Cada icono es una imagen monocromática de 60x60 píxeles, lo que requiere 450 bytes de datos.
- Se definen iconos para "Reloj digital", "Cronómetro", "Reloj analógico", "Sokoban", "Calculadora Gráf." y "Temperatura".

img.py

Almacena los datos de imagen para elementos gráficos más grandes, como el fondo y el logo.

Detalles:

- image_fondo: Un bytearray que contiene los datos de la imagen de fondo de la pantalla (probablemente 128x64 píxeles).
- logo_imagen(): Una función que devuelve un bytearray para una imagen de logo (probablemente 46x46 píxeles), utilizada en las pantallas de inicio o en la interfaz general.

ssd1306.py

Este es el driver de bajo nivel para la pantalla OLED SSD1306.

Detalles:

- Proporciona la interfaz para comunicarse con la pantalla a través de I2C.
- Implementa funciones para dibujar píxeles, líneas, texto y manipular el framebuffer.
- Es una librería estándar de MicroPython para este tipo de pantallas.

MÓDULOS DE MODOS (APLICACIONES)

Cada uno de estos archivos representa una aplicación o "modo" que el usuario puede seleccionar desde el menú principal. Todos ellos siguen un patrón común: importan utime y posiblemente machine.RTC, y definen una función main(oled, get_direction, play_sound) que es la puerta de entrada a su lógica.

hora_actual.py

Este módulo se encarga de funcionalidades relacionadas con la hora y fecha, incluyendo la sincronización NTP. Aunque el main.py lo tiene comentado inicialmente, su estructura sugiere que está diseñado para:

Detalles:

- Conexión WiFi: Permite al dispositivo conectarse a una red WiFi utilizando las credenciales de config.py.
- Sincronización NTP: Sincroniza la hora del RTC (Real-Time Clock) del dispositivo con un servidor NTP a través de internet.
- Obtención de Fecha/Hora: Recupera la fecha y hora actual del RTC.
- Visualización en OLED: Muestra la fecha y hora formateadas en la pantalla OLED.

modo_reloj.py (Reloj Digital)

Implementa la funcionalidad de un reloj digital con la capacidad de ajustar la hora y fecha.

Detalles:

- RTC: Utiliza el RTC del MicroPython para obtener la hora y fecha.
- Modo Edición: Permite al usuario entrar en un modo de edición para ajustar la hora (horas, minutos) y la fecha (año, mes, día) utilizando el joystick.

- Efecto de Parpadeo: Implementa un efecto de parpadeo para el campo que se está editando, mejorando la usabilidad.
- Visualización: Muestra la hora actual y la fecha en un formato legible en la pantalla OLED.

modo_crono.py (Cronómetro)

Proporciona una función de cronómetro simple.

Detalles:

- Inicio/Parada/Reset: El botón central del joystick inicia y detiene el cronómetro. Para reiniciar, el usuario debe salir y volver a entrar al modo.
- Medición de Tiempo: Utiliza utime.ticks_ms() para medir el tiempo transcurrido con precisión de milisegundos.
- Visualización: Muestra el tiempo transcurrido en formato HH:MM:SS.ms en la pantalla OLED.

modo_calculadora.py (Calculadora Gráfica)

Un modo de calculadora que permite ingresar expresiones y, lo que es más interesante, graficar funciones.

Detalles:

- Entrada de Expresiones: Permite al usuario construir una expresión matemática utilizando un teclado virtual en pantalla navegable con el joystick.
- Soporte de Operadores y Funciones: Admite operadores aritméticos básicos (+, -, *, /, ^) y funciones matemáticas comunes (sin, cos, tan, log, sqrt, abs).
- Gráficos de Funciones: La característica distintiva es la capacidad de graficar expresiones en la pantalla OLED, permitiendo zoom (acercar/alejar) y desplazamiento del gráfico. Esto implica un procesamiento de expresiones y mapeo de coordenadas a píxeles.

modo_calendario.py (Calendario)

Muestra la fecha actual en la pantalla OLED.

Detalles:

- RTC: Obtiene la fecha del RTC.
- Formato de Fecha: Muestra el día de la semana, día, mes y año.
- Número de Semana: Calcula y muestra un número de semana simplificado.

- Navegación Limitada: En la versión proporcionada, solo muestra la fecha actual y permite salir del modo con doble clic.

modo_reloj_analogico.py (Reloj Analógico)

Proporciona una representación visual de un reloj analógico.

Detalles:

- RTC: Utiliza el RTC para obtener la hora, minuto y segundo.
- Dibujo de Reloj: Dibuja un círculo y las manecillas de hora, minuto y segundo en la pantalla OLED, calculando sus posiciones basadas en el tiempo actual.
- Actualización Constante: Actualiza el dibujo del reloj a intervalos regulares para un movimiento fluido de las manecillas.

modo_sokoban.py (Juego Sokoban)

Implementa una versión básica del clásico juego de puzzle Sokoban.

Detalles:

- Tablero de Juego: Define un nivel de juego como una matriz de caracteres representando paredes, cajas, objetivos y al jugador.
- Movimiento: Permite al jugador mover su personaje utilizando el joystick.
- Empujar Cajas: Si el jugador intenta moverse a una casilla con una caja, la caja se moverá si el espacio detrás de ella está libre.
- Condición de Victoria: El juego se gana cuando la caja se mueve a la posición del objetivo.
- Gráficos Simplificados: Utiliza caracteres para representar los elementos del juego en la pantalla OLED.

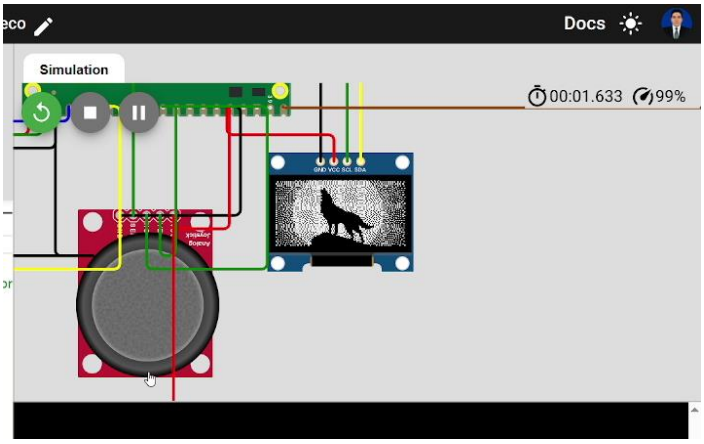
modo_temperatura.py (Sensor de Temperatura)

Muestra la lectura de temperatura, asumiendo el uso del sensor de temperatura interno de la Raspberry Pi Pico (ADC4).

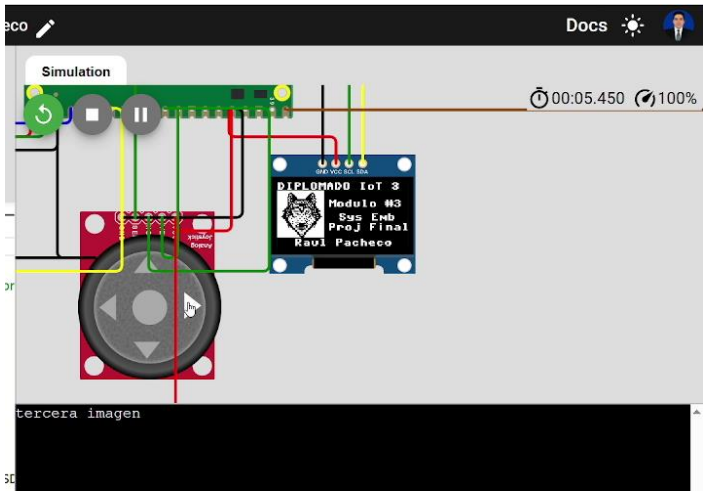
Detalles:

- Lectura ADC: Utiliza el ADC para leer el valor del sensor de temperatura interno.
- Conversión a Celsius: Convierte la lectura del ADC a grados Celsius utilizando una fórmula de calibración.
- Visualización: Muestra la temperatura actual en la pantalla OLED.
- Gráfico Simple: Incluye una función para dibujar un gráfico de barras simple que representa el nivel de temperatura.

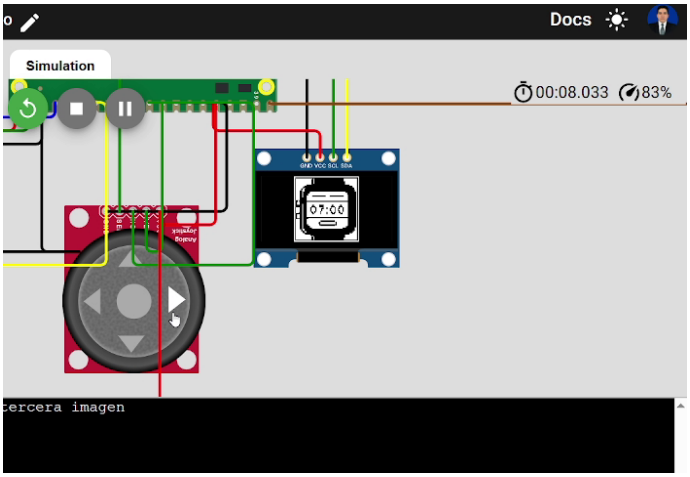
CAPTURAS DE PANTALLA



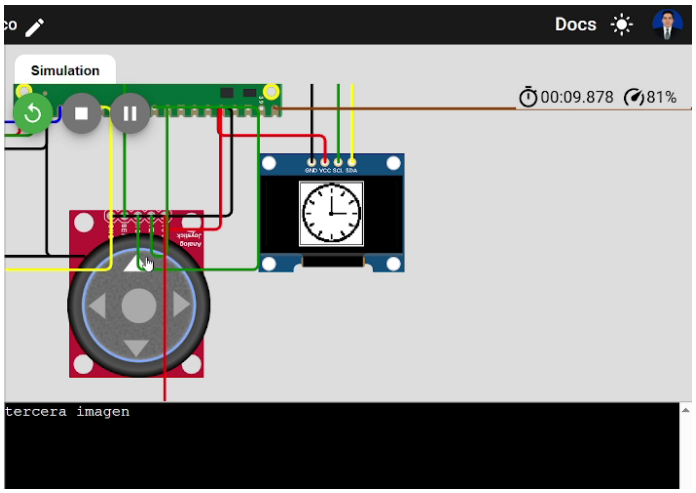
Pantalla de Bienvenida



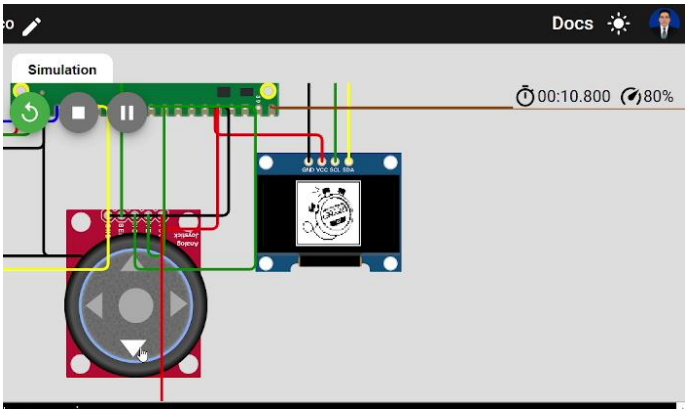
Pantalla de Presentación



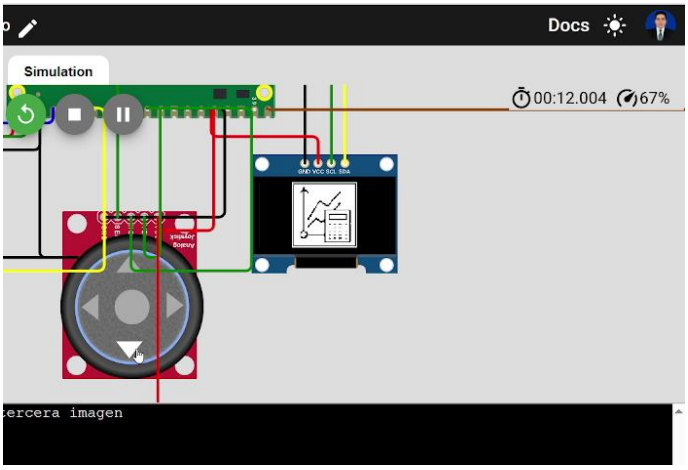
Pantalla Modo Reloj Digital



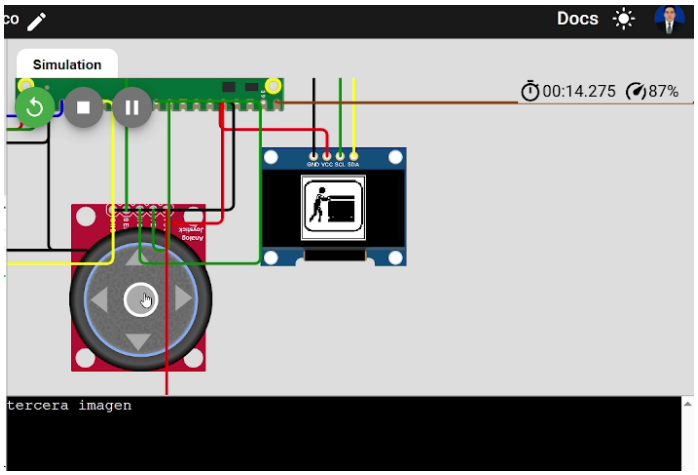
Pantalla Modo Reloj Analoogico



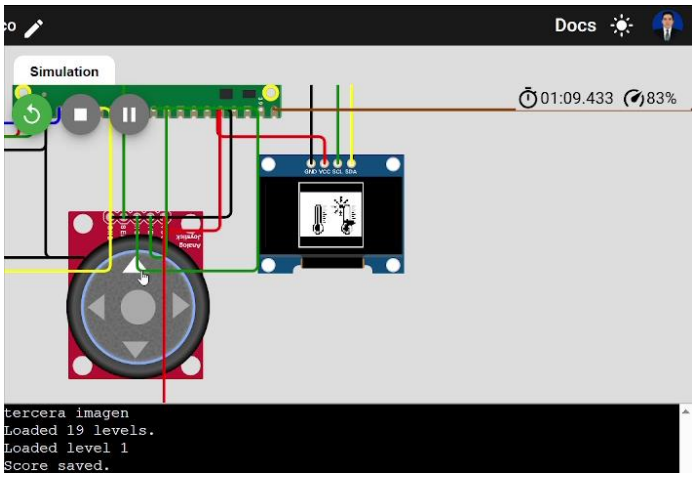
Pantalla Modo Cronometro



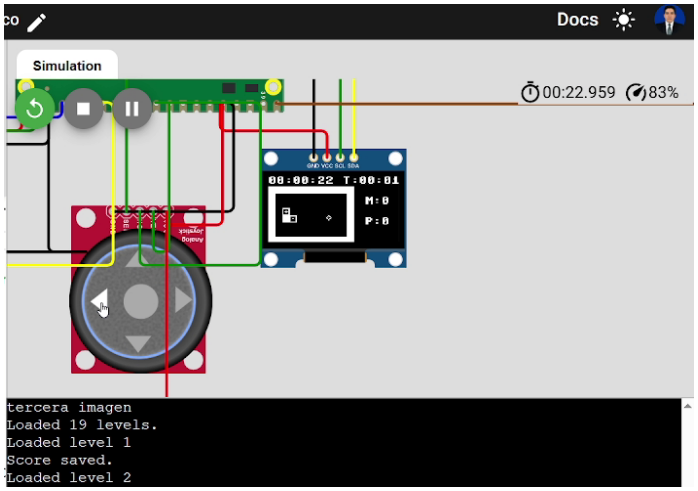
Pantalla de Modo Calculadora Grafica



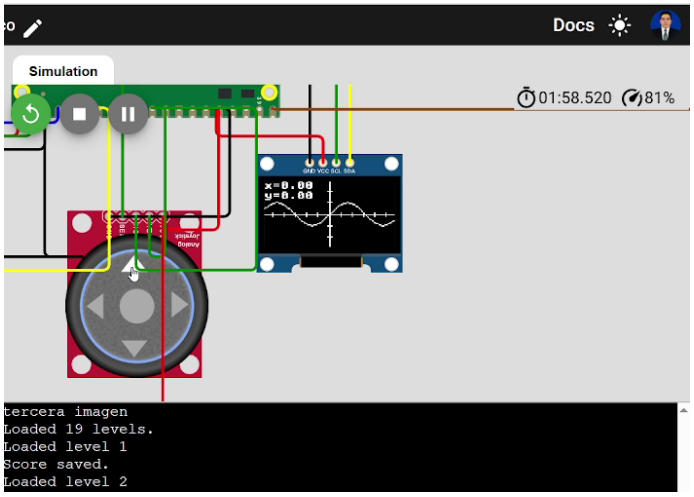
Pantalla Modo Sokoban juego



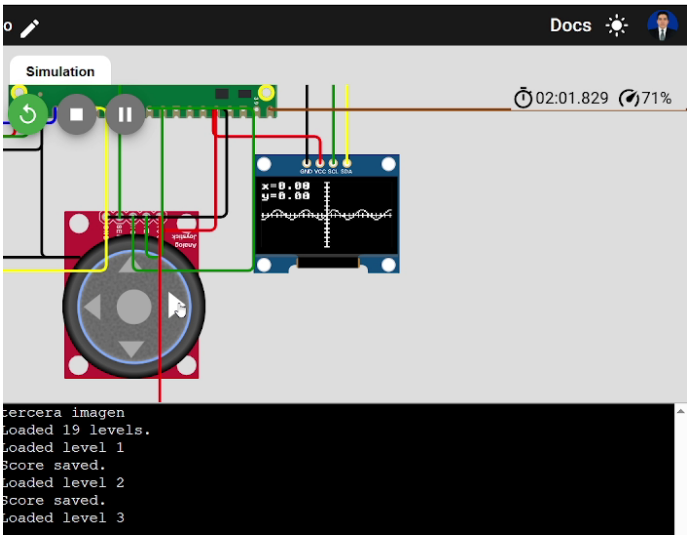
Pantalla Modo Temperatura



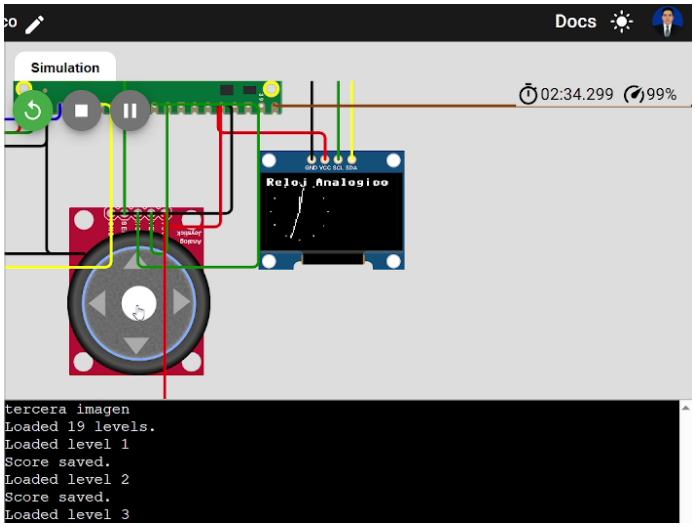
Juego de Sokoban



Pantalla demo de grafico de funciones



Pantalla demo del zoom



Pantalla demo reloj Analogico

PROYECTO EN WOKWI

https://wokwi.com/projects/436129770805324801

Se ira actualizando conforme a cada bugs que se encuentre.

JUSTIFICACIÓN DE DISEÑO:

Enfoque Modular del Sistema MicroPython

Principio Rector: Modularidad

La modularidad ha sido el pilar fundamental en la concepción de este sistema. Se ha buscado dividir la funcionalidad compleja del dispositivo en componentes independientes y autocontenidos, conocidos como "módulos". Cada módulo es responsable de una porción específica del comportamiento del sistema, interactuando con otros módulos a través de interfaces bien definidas.

Ventajas de la Modularidad Implementada

La aplicación rigurosa de la modularidad ofrece múltiples beneficios clave para este proyecto:

- **Reusabilidad del Código:**
  - Componentes como los drivers (ssd1306.py), el manejo del joystick (keyStick.py), y la configuración de pines (config.py) son desarrollados una única vez y utilizados por múltiples módulos de aplicación. Esto reduce la duplicación de código y acelera el desarrollo de nuevas funcionalidades.
  - Las funciones de inicialización de hardware y control de periféricos (como play_sound para el buzzer) son centralizadas en main.py y pasadas como parámetros a los modos, asegurando que cada modo utilice las mismas instancias de hardware y métodos de control.
- **Mantenibilidad Simplificada:**
  - Cada módulo es una unidad autónoma. Si surge un error o se necesita una mejora en la funcionalidad del cronómetro, por ejemplo, solo se necesita modificar modo_crono.py sin afectar la lógica de la calculadora o el reloj analógico.
  - La separación de la configuración de hardware en config.py facilita las adaptaciones a diferentes placas o cambios de pines, sin necesidad de tocar la lógica de las aplicaciones.
- **Escalabilidad y Extensibilidad:**
  - Añadir una nueva funcionalidad (por ejemplo, un juego adicional o un sensor diferente) es tan simple como crear un nuevo archivo modo_nuevo.py, implementando la función main requerida, y agregarlo a la lista modes en main.py. El sistema principal lo cargará y gestionará automáticamente.
  - Los nuevos módulos pueden aprovechar las utilidades y funciones compartidas existentes (oled, get_direction, play_sound) sin tener que reimplementarlas.

- **Facilidad de Depuración y Pruebas:**
    - Al aislar funcionalidades, los errores pueden ser localizados más rápidamente dentro de un módulo específico. Esto reduce el tiempo y el esfuerzo de depuración.
    - Cada módulo puede ser probado de forma independiente, facilitando la verificación de su correcto funcionamiento antes de integrarlo al sistema completo.
- **Colaboración en Equipos (si aplica):**
    - En un escenario de desarrollo en equipo, la modularidad permite que diferentes desarrolladores trabajen en distintos módulos simultáneamente con mínimas colisiones de código y dependencias.
- **Uso Eficiente de Recursos (MicroPython):**
    - Aunque MicroPython permite cierto grado de flexibilidad en la estructura, la modularidad ayuda a organizar el código de manera que el intérprete pueda manejarlo eficientemente, cargando solo lo necesario cuando se requiere, aunque en un entorno de MicroPython la precarga de todos los módulos es común para evitar ralentizaciones durante la ejecución.

Componentes Clave y su Interconexión Modular

La siguiente tabla ilustra la distribución modular de las funcionalidades:

Módulo Principal / Directorio	Responsabilidad Principal	Interacciones Clave
main.py	Orquestador del sistema, menú, carga dinámica de modos, inicialización de periféricos compartidos.	Interactúa con config.py, keyStick.py, ssd1306.py, img.py, icon.py y carga/ejecuta los modo_*.py.
config.py	Definición de constantes de hardware (pines, umbrales), parámetros de red.	Importado por main.py, keyStick.py, hora_actual.py, modo_reloj_analogico.py, modo_sokoban.py.
keyStick.py	Manejo de la entrada del joystick y el botón.	Importado por main.py; usa config.py.
ssd1306.py	Driver de la pantalla OLED.	Importado por main.py y hora_actual.py (y otros módulos si se inicializa localmente, aunque se prefiere pasar la instancia).
img.py	Almacenamiento y provisión de datos de imágenes grandes (fondo, logo).	Importado por main.py.
icon.py	Almacenamiento y provisión de datos de iconos para el menú.	Importado por main.py.

Módulo Principal / Directorio	Responsabilidad Principal	Interacciones Clave
hora_actual.py	Conexión WiFi, sincronización NTP, visualización de hora/fecha.	Importa config.py, network, ntptime.
modo_reloj.py	Reloj digital, ajuste de hora/fecha.	Recibe oled, get_direction, play_sound de main.py.
modo_crono.py	Cronómetro.	Recibe oled, get_direction, play_sound de main.py.
modo_calculadora.py	Calculadora gráfica.	Recibe oled, get_direction, play_sound de main.py.
modo_calendario.py	Visualización de calendario.	Recibe oled, get_direction, play_sound de main.py.
modo_reloj_analogico.py	Reloj analógico.	Recibe oled, get_direction, play_sound de main.py. Importa config.py.
modo_sokoban.py	Implementación del juego Sokoban.	Recibe oled, get_direction, play_sound de main.py. Importa config.py.
modo_temperatura.py	Lectura y visualización de temperatura.	Recibe oled, get_direction, play_sound de main.py.