Seguimiento 2: Grúa Robótica con servomotores

Angie Marisela García Ríos
Estefany Cuervo Suárez
Juan Felipe Arroyave Zapata
Electrónica digital II

Índice

1. Identificación del proyecto

- **Nombre del proyecto:** Informe de Proyecto - Grúa Robótica usando Micro Python
- **Integrantes del equipo: Angie Marisela García Ríos, Estefany Cuervo Suárez, Juan Felipe Arroyave Zapata**

2. Resumen

El proyecto consiste en la creación de un sistema robótico controlado mediante el microcontrolador ESP32 que permite manejar dos grados de libertad en una grúa de sobremesa. El control manual se realiza por medio de dos potenciómetros que actúan como interfaces analógicas para controlar los servomotores que comandan la rotación de la base y la elevación del brazo principal. Se implementan dos modos automáticos activados por pulsadores que, con software y gestión de interrupciones, ejecutan rutinas de regreso a la posición inicial y secuencias predeterminadas, acompañadas por señales visuales y sonoras. El sistema incorpora indicadores LED y buzzer para retroalimentación al usuario durante el uso.

3. Descripción del hardware

- **ESP32** (modelo con ADC y pines GPIO)
- **Pulsadores conectos a los pines:** 16, 17
- **Led Rojo** conectado al pin 25
- **Led Verde** conectado al pin 26
- **Buzzer** conectado al pin 22
- **Dos ervomotores conectados a los pines:** 18, 19
- **Dos potenciómetros conectados a los pines:** 35, 34
- **Resistencia conectadas a:** leds y buzzer

Diagrama de conexión:

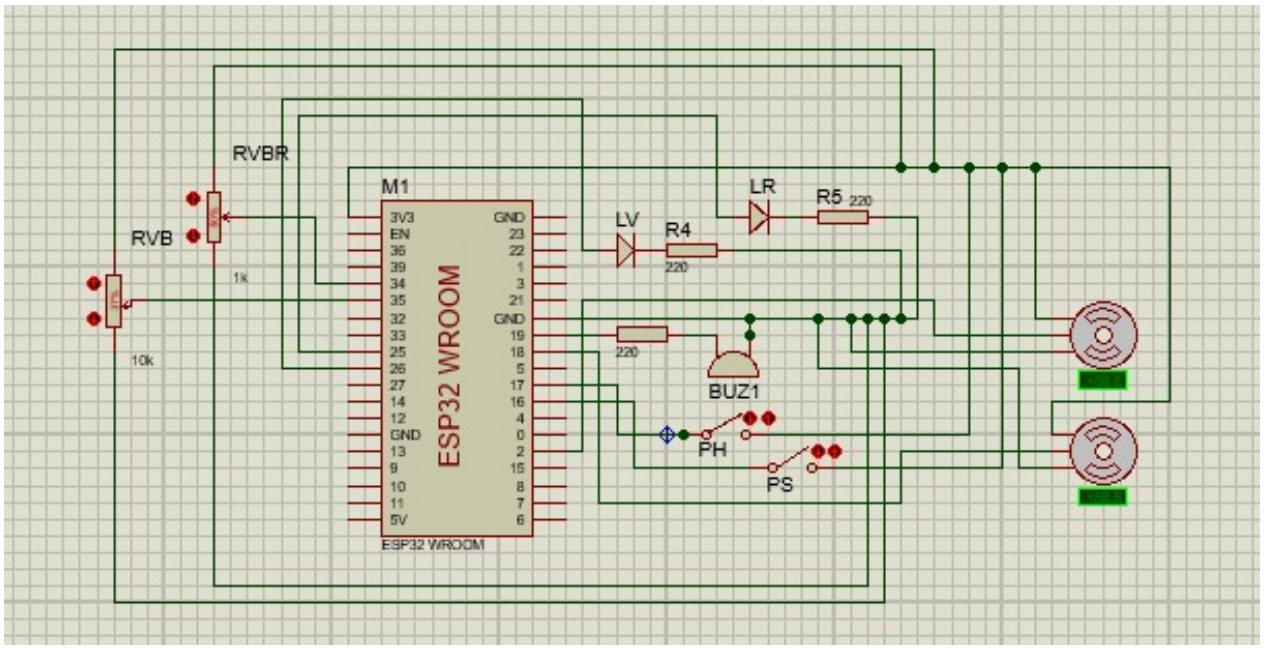
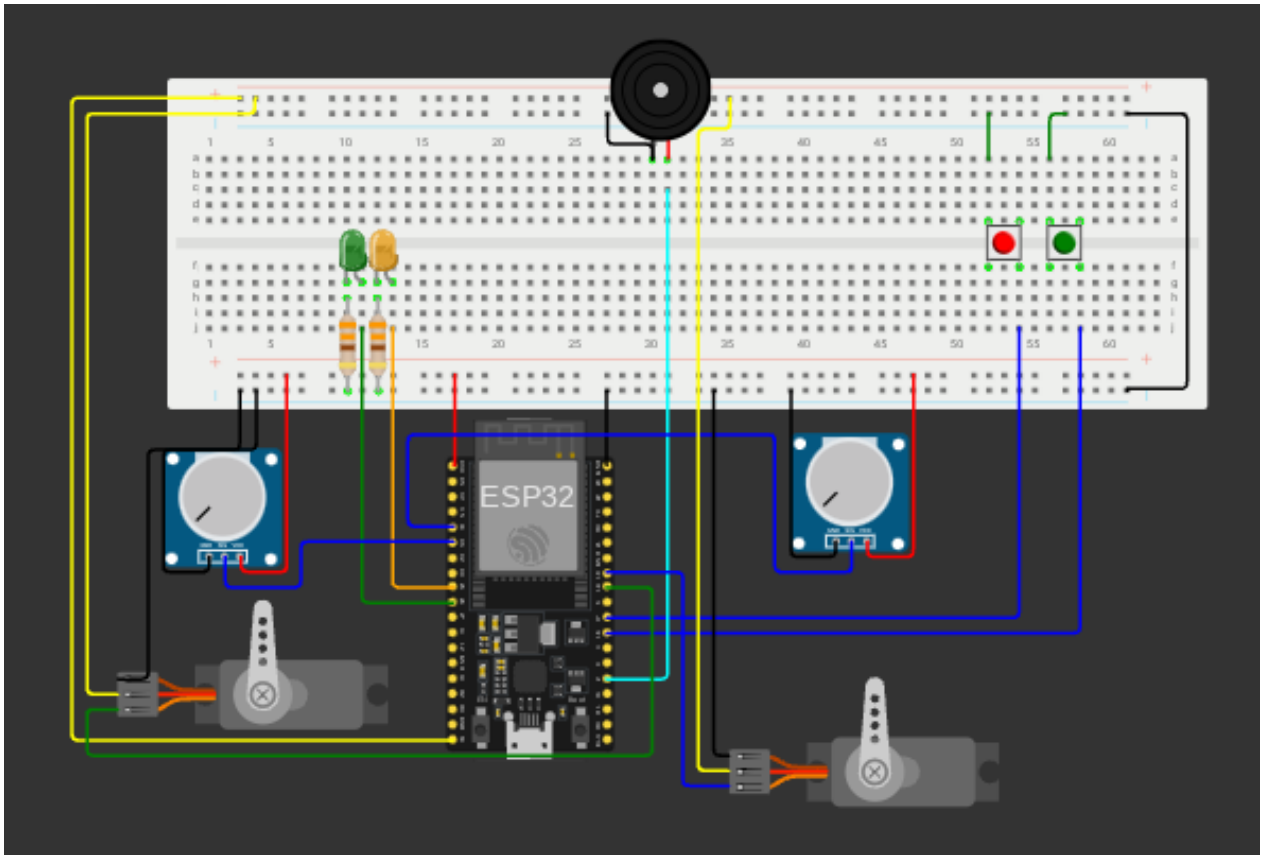


Diagrama de Conexión



Esquema de Wokwi

4. Descripción del software

- **Lenguaje usado:** MicroPython
- **Librerías:** `machine`, `time`
- **IDE:** Thonny y Wokwi
- **Funciones:** Lectura ADC, conversión a PWM, gestión de interrupciones y flags para estados.

Flujo general del programa:

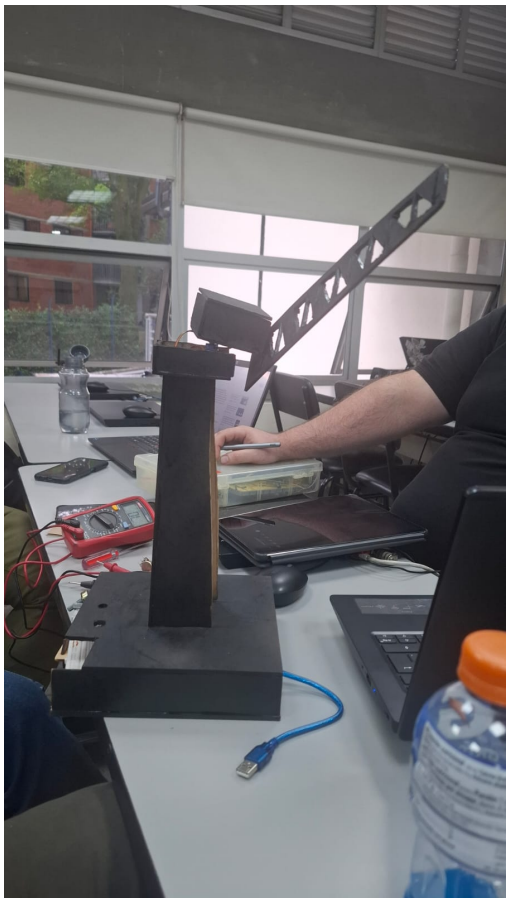
1. Configura pines, variables y estados iniciales.
2. Lee continuamente potenciómetros y detecta pulsadores.
3. Controla servos con señales PWM según entradas o rutinas.
4. Activa LEDs y buzzer durante rutinas automáticas para feedback.
5. Gestiona estados y evita acciones conflictivas con flags.
6. Libera recursos y apaga salidas al finalizar o detener el programa.

5. Estructura del código

- Archivo principal: `main.py`
- No se usan módulos adicionales.
- Código lineal dentro de un ciclo infinito con lectura y decisión en tiempo real.

6. Explicación de funciones principales

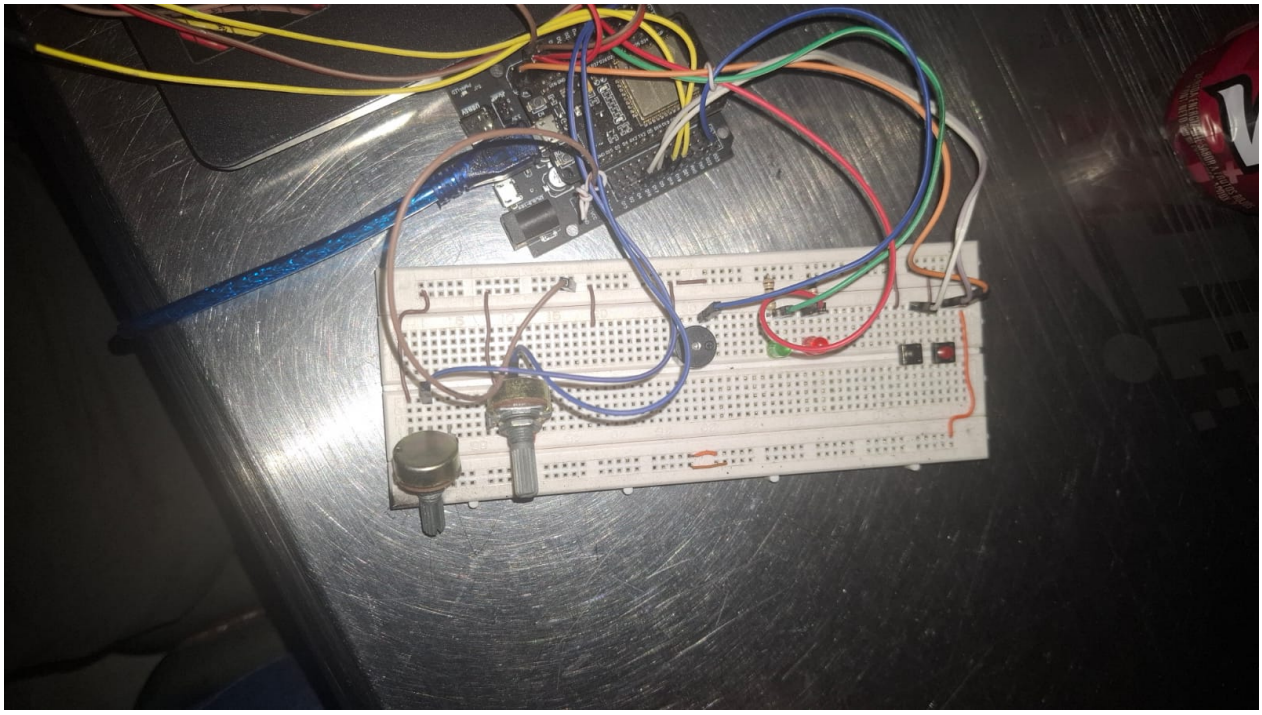
- `adc.a.servo()`: Convierte lectura analógica a ciclo PWM para servo.
- `mover.servo.gradual()`: Movimiento suave entre posiciones servo.
- `rutina.retorno.inicial()`: Lleva servos a posición inicial con indicación sonora y visual.
- `Pin.on() / Pin.off()`:
- Interrupciones con antirrebote para activación de rutinas.



Evidencia visual de la estructura de la grua

Evidencia visual de la estructura de la grua

Evidencia visual de la caja de control y montaje



Evidencia visual del montaje en protoboard

7. Funciones de lógicas

- Control basado en flags para diferenciar modo manual y automático.

- Validación de entradas para evitar ejecución simultánea de rutinas.
- Supervisión de pulsadores para activar rutinas específicas con debouncing.

8. Funciones de retroalimentación

- `beep()`: LED verde encendido en modo manual; rojo en modo automático.
- Buzzer activo durante rutinas automáticas para alertas y confirmaciones.

9. Procedimiento de Prueba

1. Conectar el ESP32 al computador mediante cable USB
2. Cargar el código en el ESP32 usando Thonny IDE
3. Abrir consola serial para visualizar mensajes del sistema
4. Verificar que todos los LEDs estén apagados al inicio
5. Presionar botones y mover potenciometros para confirmar funcionamiento correcto
6.
7. Verificar asignación correcta de puntuaciones en consola
8. Verificar correcto regreso a modo manual y posiciones finales de los servos.

10. Manejo de Errores y Seguridad

10.1. Prevención de Errores

- Antirrebote en software para pulsadores.
- Control estricto de estados con flags que evitan solapamiento.
- Verificación de cambios significativos para mover servos

10.2. Recuperación de Errores

- Manejo de excepciones con limpieza de recursos.
- Mensajes en consola para diagnóstico.
- Reinicio manual en caso de bloqueos.

10.3. Limitaciones y Consideraciones de Seguridad

- Se utilizan resistencias limitadoras para LEDs y buzzer
- Configuración PULL_DOWN en todas las entradas para evitar estados flotantes
- Los pines GPIO del ESP32 tienen protección
- limitada contra cortocircuitos
- Operación en entorno controlado por ausencia de sensores externos.
- Dependencia de alimentación estable y supervisión del circuito.

10.4. Monitorización

- Visualización de estado mediante LEDs y buzzer.
- Indicación de estado del sistema en tiempo real
- Mensajes de depuración e información en consola serial.