Sistema avanzado de medicion de reflejos

Luisa Castaño Sepúlveda, Anny Juliana Acosta, Juana Valentina Monsalve
Electrónica digital II

Lunes 15 de Septiembre 2025

Índice

1. Identificación del proyecto

- **Nombre del proyecto:** Sistema avanzado de medicion de reflejos.
- **Integrantes del equipo:** Luisa Castaño Sepúlveda, Anny Juliana Acosta, Juana Valentina Monsalve

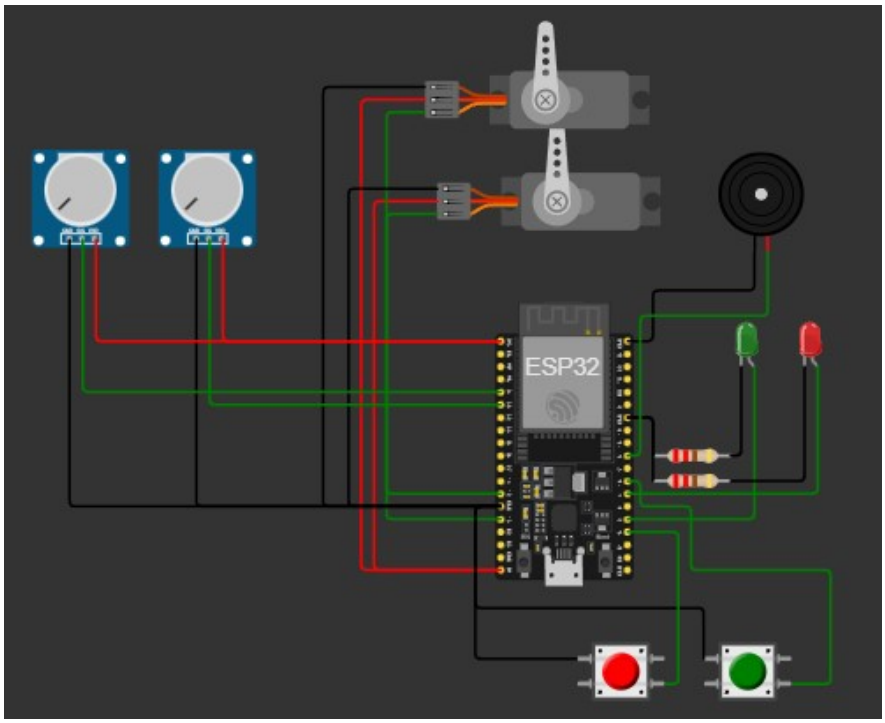
2. Resumen

En esta práctica se diseñó e implementó una grúa a escala controlada por un microcontrolador ESP32. El sistema permite mover la base y el brazo de la grúa en tiempo real mediante potenciómetros, así como ejecutar rutinas automáticas de retorno y secuencia de movimiento para simular tareas repetitivas. Se añadieron indicadores visuales y sonoros (LEDs y buzzer) para mostrar el estado de operación, y se aplicó antirrebote por software para un manejo confiable de las entradas digitales. Este proyecto permitió integrar conceptos de control de PWM, lectura analógica con ADC, manejo de interrupciones y lógica de estados, aplicados a un sistema mecatrónico.

3. Descripción del hardware

- **ESP32:** microcontrolador con salidas PWM y entradas analógicas.
- **Servomotores:**
  - Servo de base (Pin 13): rota la grúa.
  - Servo de brazo (Pin 12): sube y baja el brazo principal.
- **Potenciómetros:** dos potenciómetros conectados a pines 34 y 35 permiten el control proporcional de la posición de los servos.
- **Indicadores:**
  - LED verde en pin 2 para indicar modo manual activo.
  - LED rojo y buzzer en pines 4 y 5 para alertas durante rutinas automáticas.
- **Botones de control:**
  - Botón de retorno (pin 15): activa la rutina de recentralizado de la grúa.
  - Botón de secuencia (pin 16): ejecuta una secuencia de movimientos predefinida.
- **Fuente de alimentación:** 5V estable por USB o regulador externo.

Diagrama de conexión:



Circuito:

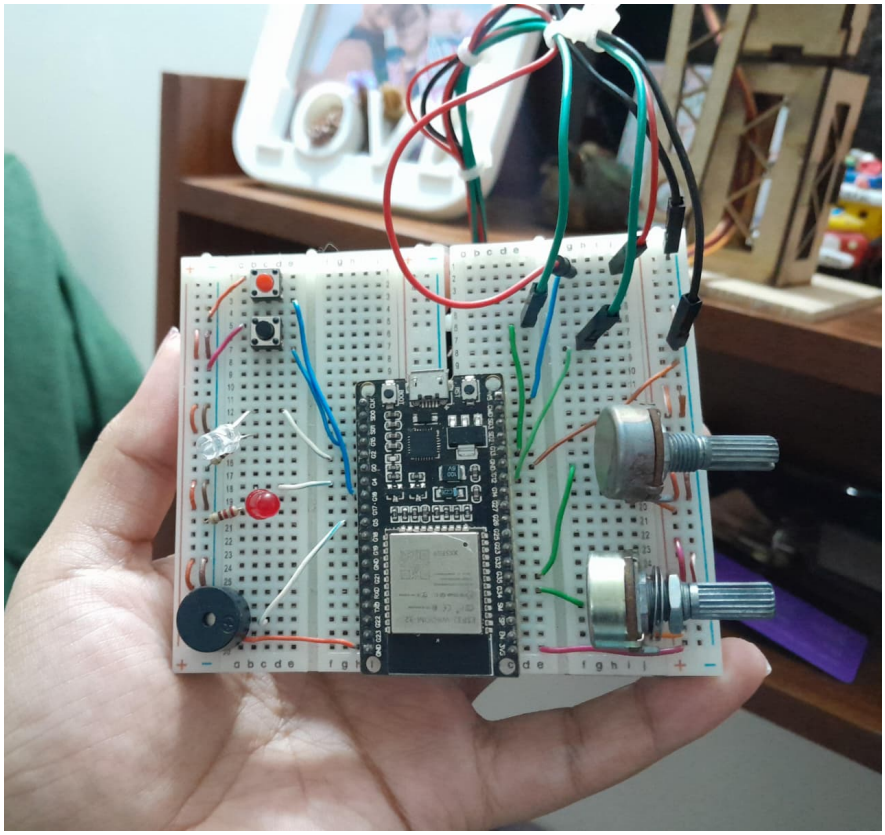
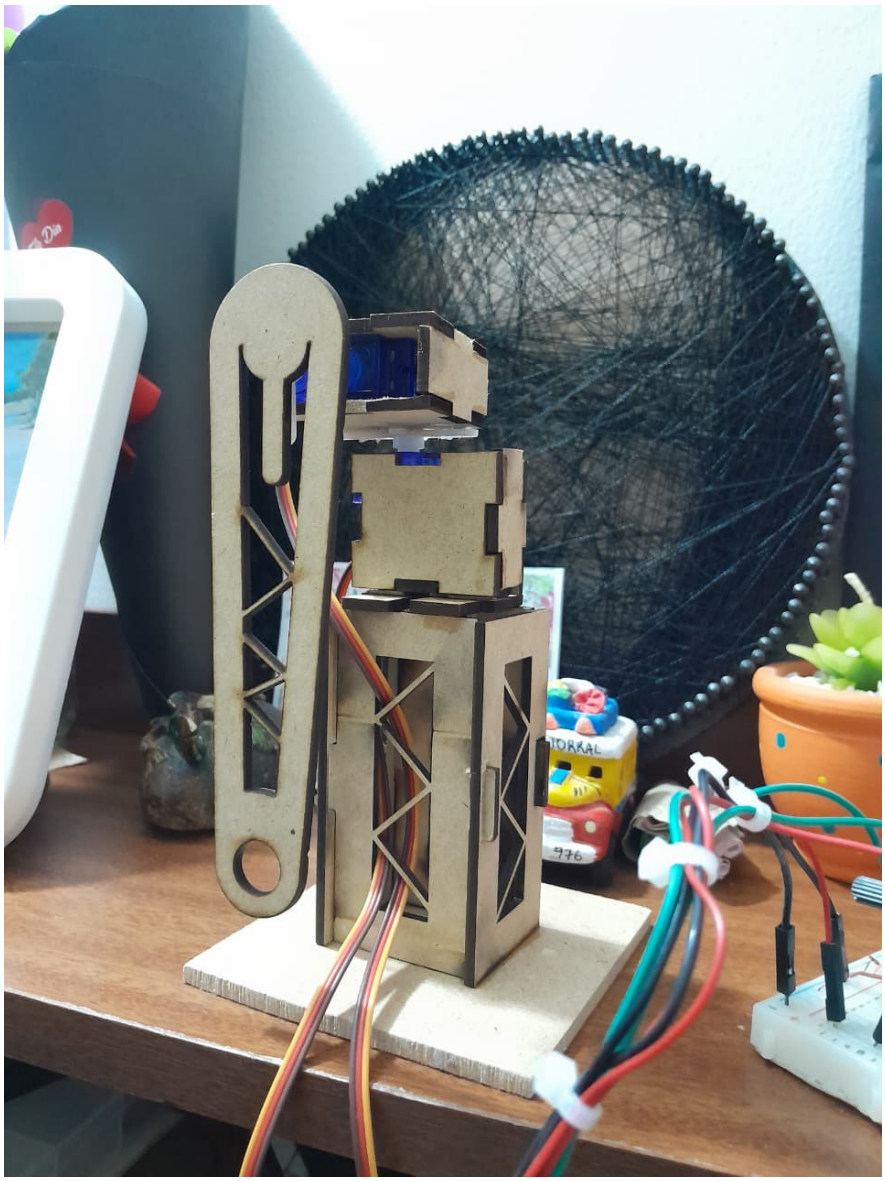


Foto del proyecto:



4. Descripción del software

- **Lenguaje usado:** MicroPython
- **Librerías:** `machine`, `time`
- **IDE:** Thonny

Flujo general del programa:

1. Configuración inicial de pines para PWM, ADC, LEDs, buzzer y botones.
2. Definición de variables globales de estado y posiciones iniciales de la grúa.
3. Configuración de interrupciones con antirrebote para los botones.

4. Posicionamiento inicial de la grúa en estado neutro (base en 90°, brazo en 90°).
5. Ejecución de un bucle principal que:
    - Lee eventos de botones para cambiar el modo de operación.
    - Mueve los servos en tiempo real si está en modo manual.
    - Ejecuta retorno o secuencia si el modo automático está activado.
    - Sincroniza potenciómetros con la posición de la grúa antes de regresar a control manual.

5. Estructura del código

- **Archivo principal:** `main.py`
- Variables globales: `modoManual, modoRetorno, modoSecuencia, posBase, posBrazo, evento`.
- Funciones auxiliares:
    - `moverServo()`: convierte un ángulo en ciclo de trabajo PWM para controlar el servo.
    - `leerPotenciometros()`: convierte valores analógicos en ángulos para los servos.
    - `activarAlarma() / desactivarAlarma()`: controlan LED rojo y buzzer.
    - `rutinaRetorno()`: devuelve la grúa a su posición inicial.
    - `rutinaSecuencia()`: ejecuta movimientos encadenados simulando una operación automática de carga.
- Interrupciones:
    - `interrupcionReset()`: detecta pulsación del botón de retorno.
    - `interrupcionSecuencia()`: detecta pulsación del botón de secuencia.
- Lógica principal:
    1. Evalúa eventos.
    2. Ejecuta movimiento manual o rutinas automáticas según el estado.
    3. Controla indicadores visuales y sonoros.

6. Explicación de funciones principales

- `moverServo(servo, angulo)`: mueve el servo a un ángulo entre 0°–180°.
- `leerPotenciometros()`: lee los potenciómetros y traduce su posición en un valor de ángulo para cada eje.

- `rutinaRetorno()`: mueve la grúa a su posición inicial paso a paso, activando alarma durante el proceso.
- `rutinaSecuencia()`: ejecuta un ciclo de movimientos predefinido y regresa la grúa a su posición inicial, emitiendo alerta visual y sonora.
- `interrupcionReset() / interrupcionSecuencia()`: gestionan el cambio de modo de operación sin rebotes eléctricos.

7. Comunicación

En este proyecto no se implementaron protocolos de comunicación externos, ya que el control se realiza de forma local mediante potenciómetros y pulsadores. Sin embargo, el sistema puede extenderse para incluir diferentes mecanismos de comunicación para monitoreo remoto, automatización y registro de datos.

8. Interfaz de usuario

- **LED verde:** indica que el sistema está en control manual.
- **LED rojo + buzzer:** se activan durante rutinas automáticas como alerta de seguridad.
- **Potenciómetros:** controlan base y brazo de la grúa.
- **Botones:** activan retorno o secuencia automática.

9. Procedimiento de prueba

1. Conectar el ESP32 a la fuente de alimentación o puerto USB.
2. Cargar el archivo `main.py` en el ESP32 desde Thonny.
3. Verificar el movimiento inicial de los servos a 90°.
4. Girar los potenciómetros y comprobar que los movimientos de la grúa correspondan.
5. Presionar el botón de retorno y confirmar que la grúa vuelva a su posición inicial.
6. Presionar el botón de secuencia y verificar el ciclo de movimientos de la grúa.
7. Recentralizar los potenciómetros para volver al modo manual.

10. Manejo de errores y seguridad

- Antirrebote implementado en software para botones.
- Movimientos graduales para evitar golpes mecánicos en la estructura de la grúa.
- Se recomienda no forzar manualmente los servos para evitar daños.
- Alimentar el sistema con una fuente estable para evitar reinicios.
- Agregar protección de sobrecorriente si se usa en cargas más grandes.