Reporte de Proyecto: Sistema de Control de silla de ruedas movil

Autor: Eduardo Gonzalez Gonzalez

Resumen Ejecutivo

Este reporte presenta un sistema de control para una silla de ruedas móvil.. El proyecto se centra en la creación de un código Python funcional que permite la navegación autónoma y el control manual de un robot, integrando diversas tecnologías como conectividad WiFi, lectura de RFID para puntos de control, control preciso de motores, y detección de obstáculos mediante sensores ultrasónicos. Una característica distintiva de este desarrollo es la implementación de un servidor web embebido, que facilita la interacción remota y el monitoreo del robot a través de una interfaz intuitiva. El diseño del código prioriza la funcionalidad y la eficiencia, demostrando una sólida comprensión de la robótica, la programación de sistemas embebidos y la interacción hardware software.

1. Introducción

En el campo de la robótica móvil, la capacidad de un robot para navegar de forma autónoma en un entorno predefinido, mientras ofrece opciones de control manual, es fundamental para diversas aplicaciones, desde la logística hasta la asistencia personal. Este proyecto aborda la creación de un sistema de control integral para un robot móvil, con un énfasis particular en la funcionalidad y la eficiencia del código. Como diseñador principal de este sistema, mi objetivo fue desarrollar una solución que no solo fuera técnicamente robusta, sino también práctica y fácil de operar, incluso en entornos con recursos computacionales limitados. La integración de múltiples sensores y actuadores, junto con una interfaz de usuario remota, fue un desafío clave que se abordó mediante un diseño de software modular y bien estructurado.

El presente documento detalla la arquitectura del código, las funcionalidades implementadas, los componentes de hardware controlados y las decisiones de diseño que permitieron la creación de un sistema operativo y adaptable. Se destacará cómo cada módulo del código contribuye a la funcionalidad general del robot, permitiendo
una interacción fluida entre el software y el hardware, y cómo la implementación de un

servidor web embebido transforma la forma en que los usuarios pueden interactuar con el robot.

2. Arquitectura y Componentes del Código

El código Python desarrollado para este proyecto sigue una arquitectura modular, lo que facilita su comprensión, mantenimiento y futuras expansiones. Cada sección del código está dedicada a una funcionalidad específica, permitiendo una clara separación de responsabilidades y una gestión eficiente de los recursos del sistema.

2.1. Módulos y Librerías Utilizadas

Para la implementación de las diversas funcionalidades, se hizo uso de varias librerías y módulos, tanto estándar de Python como específicos para el entorno de MicroPython en el que opera el robot:

- **`dcmotor`** : Esta librería, probablemente personalizada para el proyecto, es fundamental para el control de los motores de corriente directa (DC). Proporciona una abstracción que simplifica la interacción con los pines de control y las señales PWM necesarias para el movimiento del robot.
- **`machine`** : Un módulo esencial en MicroPython, `machine` permite la interacción directa con el hardware del microcontrolador. Se utilizó para configurar los pines de entrada/salida (`Pin`), generar señales de modulación por ancho de pulso (`PWM`) para el control de velocidad de los motores y el parlante, y para la comunicación SPI (`SPI`) con el módulo RFID.
- **`time`** : Este módulo se empleó para funciones relacionadas con la temporización, como `sleep` para introducir retardos en la ejecución y `ticks_ms` para mediciones de tiempo precisas, cruciales en la lógica de lectura RFID y control de movimiento.
- **`mfrc522`** : Una librería específica para interactuar con el módulo lector de RFID MFRC522. Permite la detección y lectura de etiquetas RFID, lo que es vital para la navegación basada en puntos de control.
- **`socket`** : Utilizado para la implementación del servidor web. Este módulo proporciona las herramientas necesarias para establecer conexiones de red, enviar

y recibir datos, y gestionar las solicitudes HTTP.

- `network` : Este módulo de MicroPython es responsable de la gestión de la conectividad de red, permitiendo al robot conectarse a una red WiFi y obtener una dirección IP para la comunicación remota.
- `_thread` : Para habilitar la concurrencia en el sistema, `_thread` se utilizó para ejecutar el servidor web en un hilo separado. Esto asegura que la interfaz web pueda responder a las solicitudes del usuario sin interrumpir el bucle principal de control del robot.
- `hcsr04` : Probablemente una librería personalizada para el sensor ultrasónico HC SR04. Facilita la medición de distancias, lo que es crucial para la detección de obstáculos y la navegación segura.

2.2. Componentes de Hardware Controlados

El código fue diseñado para interactuar y controlar una serie de componentes de hardware, cada uno desempeñando un papel vital en la funcionalidad del robot:

- **LEDs:** Se configuraron tres LEDs (`led_cx` , `led_r` , `led_v`) para proporcionar indicadores visuales del estado del sistema, como la conectividad WiFi, la lectura
de RFID o el estado general del robot.
- **Parlante (Speaker):** Controlado mediante PWM, el parlante emite tonos audibles para proporcionar retroalimentación al usuario sobre el estado de las operaciones (inicio, éxito, error).
- **Sensores Ultrasónicos (HCSR04):** Un sensor `ultra_front` se implementó para detectar obstáculos en el camino del robot, permitiendo una respuesta proactiva para evitar colisiones.
- **Motores DC:** Dos motores de corriente directa, controlados por un driver DRV8833, son los encargados del movimiento del robot. El código gestiona sus pines de control y las señales PWM para regular la velocidad y la dirección.
- **Módulo RFID (MFRC522):** Este módulo permite al robot identificar su posición en el entorno mediante la lectura de etiquetas RFID estratégicamente ubicadas,

que actúan como puntos de control.

3. Funcionalidades Implementadas

El código desarrollado integra una serie de funcionalidades clave que dotan al robot de capacidades avanzadas de navegación y control.

3.1. Gestión de Audio y Conectividad

- **Funciones de Audio:** Se implementaron funciones (`play_tone` , `play_startup` , `play_success` , `play_error`) para generar diferentes tonos audibles, proporcionando una retroalimentación clara al usuario sobre el estado del robot y las operaciones realizadas.
- **Conexión WiFi:** La función `connect_wifi()` se encarga de establecer una conexión persistente a una red WiFi predefinida. Esto es fundamental para la comunicación remota y el control del robot a través de la interfaz web.

3.2. Detección de Obstáculos y Control de Motores

- **Sensores Ultrasónicos:** La función `get_distance_filtered()` obtiene lecturas de distancia del sensor ultrasónico, aplicando un filtro para mejorar la precisión. La función `check_obstacles()` utiliza estas lecturas para detectar la presencia de obstáculos frontales. Si se detecta un obstáculo a una distancia crítica, el robot se detiene, emite una alerta sonora y espera a que el camino se despeje antes de reanudar su movimiento. Esta lógica es crucial para la seguridad y la autonomía del robot.
- **Control de Motores:** Se definieron funciones detalladas para el control de los motores, incluyendo `move_forward()` , `move_backward()` , `stop()` , `turn_left()` , `turn_right()` , `turn_180()` , y `detener()` . Estas funciones permiten un control preciso sobre la dirección y la velocidad del robot. Además, se implementaron funciones específicas para el control manual (`manual_forward` , `manual_left` , etc.), que se activan a través de la interfaz web.

3.3. Sistema de Navegación RFID

- **Lectura RFID:** La función `read_rfid()` se encarga de interactuar con el módulo

RFID para detectar y leer etiquetas. Se implementó un mecanismo de retardo para
evitar lecturas repetidas y se proporciona retroalimentación visual (LED rojo) y auditiva al detectar una etiqueta. Las etiquetas RFID se mapean a puntos de control (`rfid_checkpoints`) en el entorno del robot.

- **Navegación Autónoma:** La función `navigate_to_destination()` es el corazón del sistema de navegación autónoma. Utiliza la función `determine_route()` para calcular la secuencia de movimientos y puntos de control necesarios para llegar a un destino específico. El robot avanza a través de la ruta, deteniéndose en los puntos de control intermedios y realizando las acciones de giro necesarias. La detección de obstáculos se integra en este proceso para asegurar una navegación segura.

3.4. Servidor Web y Control Remoto

- **Interfaz Web:** La función `web_server()` implementa un servidor web embebido que permite el control y monitoreo remoto del robot a través de una interfaz HTML simple. Esta interfaz muestra el punto de control actual del robot, su destino y el modo de operación (manual o autónomo).
- **Comandos Remotos:** A través de la interfaz web, los usuarios pueden establecer un destino para la navegación autónoma (`/go?dest=`) o enviar comandos de movimiento manual (`/manual?cmd=`). El servidor web procesa estas solicitudes y activa las funciones correspondientes en el robot. La ejecución del servidor web en un hilo separado (`_thread.start_new_thread(web_server, ())`) asegura que la interfaz sea responsiva sin bloquear la lógica de control principal del robot.

4. Diseño Funcional y Aportación.

Como diseñador principal de este código, mi enfoque se centró en la creación de un sistema que no solo fuera técnicamente viable, sino que también ofreciera una funcionalidad robusta y una experiencia de usuario intuitiva. Cada línea de código y cada decisión de diseño fueron tomadas con el objetivo de asegurar que el robot operara de manera eficiente y confiable en su entorno. Los siguientes puntos destacan mi contribución clave al diseño funcional de este proyecto:

- **Integración de Sistemas Complejos:** Fui responsable de la integración exitosa

de múltiples subsistemas (sensores, actuadores, comunicación inalámbrica, interfaz
de usuario) en una solución cohesiva. Esto implicó la selección adecuada de hardware y librerías, así como la implementación de protocolos de comunicación eficientes entre ellos.

- **Lógica de Control de Navegación:** Diseñé la lógica de navegación autónoma, incluyendo la definición de rutas, el manejo de puntos de control RFID y la implementación de algoritmos para la toma de decisiones en tiempo real. La
capacidad del robot para calcular y seguir rutas, así como para reaccionar a obstáculos, es un testimonio directo de este diseño.
- **Robustez y Manejo de Excepciones:** Anticipé y abordé posibles escenarios de fallo, como la detección de obstáculos, implementando mecanismos de respuesta que garantizan la seguridad y la continuidad operativa del robot. La lógica de detención y espera ante obstáculos es un ejemplo claro de este enfoque en la robustez.

Diseño de Interfaz de Usuario: Concebí y desarrollé la interfaz web, priorizando la simplicidad y la facilidad de uso. Esta interfaz permite a cualquier usuario controlar y monitorear el robot de forma remota, sin necesidad de conocimientos técnicos avanzados, lo que amplía significativamente la aplicabilidad del sistema. • **Optimización para Entornos Embebidos:** Dada la naturaleza de los microcontroladores, optimicé el código para asegurar un rendimiento eficiente con recursos limitados. Esto incluyó la gestión cuidadosa de la memoria y el procesamiento, así como la implementación de concurrencia básica para mejorar la responsividad del sistema.

En resumen, este proyecto es una demostración de mi capacidad para diseñar y desarrollar soluciones de software complejas para sistemas embebidos, con un enfoque en la funcionalidad, la fiabilidad y la usabilidad. El código no solo cumple con los requisitos técnicos, sino que también refleja una comprensión profunda de los desafíos prácticos en el campo de la robótica.

5. Conclusiones

El sistema de control de robot móvil usando de modelo una silla de ruedas El sistema de control de un robot móvil, basado en el modelo de una silla de ruedas, representa un logro significativo en la integración de hardware y software para crear una solución

funcional y versátil. La capacidad del robot para navegar de forma autónoma, detectar obstáculos y ser controlado remotamente a través de una interfaz web demuestra la efectividad del diseño del código.

La implementación de funcionalidades como la retroalimentación auditiva y visual, junto con una robusta lógica de navegación basada en tecnología RFID, contribuye a un sistema completo, intuitivo y fácil de usar. Este proyecto valida mis habilidades para diseñar y desarrollar sistemas embebidos complejos, abarcando desde la concepción de la lógica de control hasta la implementación de la interfaz de usuario.

Aunque mi responsabilidad se centró únicamente en el desarrollo del código, al tratarse de una compra anónima, desconozco qué se logró finalmente con el producto. Sin embargo, el desarrollo de este sistema me permitió consolidar conocimientos clave y demostrar competencias técnicas aplicadas en un entorno real

6. Anexos

Anexo partes del codigo

```
led_cx = Pin(2, Pin.OUT)  # LED integrado de la placa (WiFi)
led_r = Pin(17, Pin.OUT)  # LED rojo (GPIO5)
led_v = Pin(16, Pin.OUT)  # LED verde (GPIO17)

# Variables de estado del sistema
running = True
current_checkpoint = None
destination = None
manual_mode = False
```

```
def check_obstacles():
    dist_front = get_distance_filtered(ultra_front)

    if dist_front <= 30:
        print("¡Obstáculo detectado! Deteniendo...")
        play_error()
        detener()

        # Esperar a que se despeje
        while True:
            dist = get_distance_filtered(ultra_front)
            if dist > 30:
                print("Camino despejado, reanudando...")
                play_tone(800, 100)
```

```python
def web_server():
    
    global destination, manual_mode, current_checkpoint

    addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
    s = socket.socket()
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.bind(addr)
    s.listen(1)
    print("Servidor web iniciado en 0.0.0.0:80")
```