Desarrollo e Implementación de un Robot Móvil con Ruedas

Emanuele Donato

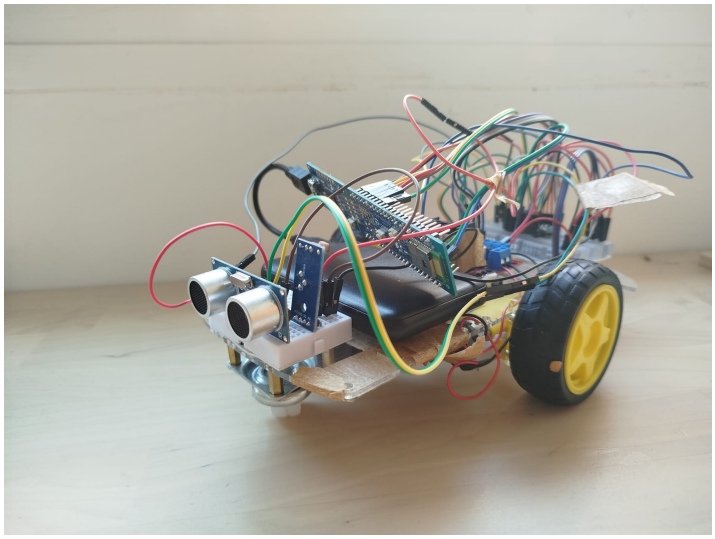
June 11, 2024

Abstract

En este informe se aborda el problema de la regulación de un robot móvil del tipo uniciclo. Considerando las ecuaciones cinemáticas expresadas en la ecuación:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \omega \end{cases}$$

se propone el diseño de un control de retroalimentación basado en el estado (x, y, θ), asumiendo que este estado es conocido. El objetivo es guiar al uniciclo hasta una posición deseada (x_d, y_d) en el plano, sin especificar la orientación final. Este tipo de problema se conoce como regulación parcial, ya que el término "parcial" se refiere a que solo se especifica la posición final deseada y no la orientación. La solución propuesta busca garantizar que el uniciclo alcance la posición objetivo de manera eficiente y precisa.



1 Modelo del Uniciclo

El uniciclo es un tipo de robot móvil caracterizado por tener una rueda principal que proporciona el movimiento y una rueda pasiva que ayuda en el equilibrio. Las ecuaciones cinemáticas que describen su movimiento son fundamentales para entender su control y navegación.

1.1 Ecuaciones Cinemáticas

El modelo cinemático del uniciclo se puede expresar mediante las siguientes ecuaciones:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \omega \end{cases}$$

donde: - x e y representan la posición del uniciclo en el plano. - θ es la orientación del uniciclo. - v es la velocidad longitudinal. - ω es la velocidad angular.

1.2 Control del Movimiento

Para controlar el movimiento del uniciclo y llevarlo a una posición deseada (x_d, y_d) sin especificar la orientación final, se utilizan leyes de control basadas en la retroalimentación del estado. La velocidad longitudinal v y la velocidad angular ω son las variables de control.

El control proporcional en las velocidades se define de la siguiente manera:

$$v = K_v \cdot e_p \cos(\varphi)$$
$$\omega = K_\omega \cdot (\theta_d - \theta)$$

donde: - K_v y K_ω son constantes positivas de ganancia. - $e_p = \sqrt{(x_d - x)^2 + (y_d - y)^2}$ es la distancia al objetivo. - $\varphi = \theta - \theta_d$ es el ángulo entre la orientación del uniciclo y la dirección hacia el objetivo. - $\theta_d = \text{atan2}(y_d - y, x_d - x)$ es la orientación deseada hacia el objetivo.

1.3 Análisis de la Convergencia

Para demostrar la convergencia del control, se puede utilizar una función de Lyapunov definida como:

$$V(x, y) = \frac{1}{2} e_p^2$$

La derivada de V a lo largo de las trayectorias del sistema es:

$$\dot{V} = -K_v[(x_d - x)\cos(\theta) + (y_d - y)\sin(\theta)]^2$$

Dado que $\dot{V} \leq 0$, e_p es limitado y converge a cero, lo que asegura que el uniciclo alcanzará la posición deseada (x_d, y_d).

1.4 Discretización del Modelo

Para la implementación práctica del control, es útil discretizar el modelo cinemático del uniciclo. Utilizando una aproximación de Euler, las ecuaciones discretizadas son:

$$\begin{cases} x_{k+1} = x_k + v_k \cos(\theta_k)\Delta t \\ y_{k+1} = y_k + v_k \sin(\theta_k)\Delta t \\ \theta_{k+1} = \theta_k + \omega_k \Delta t \end{cases}$$

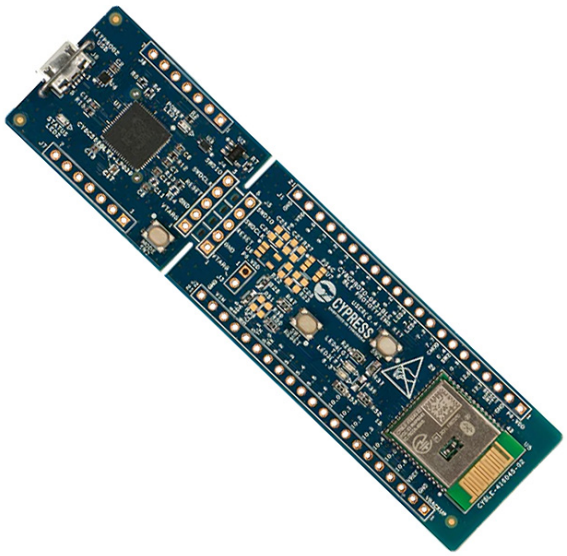
donde Δt es el intervalo de discretización. Esta discretización introduce un error de aproximación que depende del tamaño de Δt. Sin embargo, en la práctica, se pueden utilizar medidas directas (aunque ruidosas) de los desplazamientos efectivos de las ruedas en cada intervalo de discretización, lo que mejora la precisión.

El error acumulado en la reconstrucción odométrica se puede minimizar utilizando sensores adicionales y algoritmos de estimación que combinen múltiples fuentes de datos para corregir las estimaciones de posición y orientación.

2 PSoC® 6 CYBLE-416045-02 board

La plataforma de hardware utilizada es la placa: CYBLE-416045-02.

- **Microcontrolador:** PSoC 6 MCU
- **Arquitectura:** ARM Cortex-M4 y ARM Cortex-M0+
- **Frecuencia de reloj:** hasta 150 MHz (Cortex-M4) y 100 MHz (Cortex-M0+)
- **RAM:** 288 KB



- **ROM:** 1 MB de Flash
- **PIN:** 116
- **Oscilador de cristal:** 32.768 kHz y 24 MHz
- **Interfaz Bluetooth:** Bluetooth 5.0 Low Energy (BLE)
- **Consumo de energía:**
  - Corriente TX: 5.7 mA (radio solamente, 0 dBm)
  - Corriente RX: 6.7 mA (radio solamente)
- **Interfaces:**
  - USB 2.0 Full Speed
  - I2C
  - SPI
  - UART
  - CAN
  - ADC
  - DAC
  - GPIO
- **Capacidades de I/O programables:** Hasta 36 GPIOs con modos de conducción programables
- **Seguridad:**
  - Arranque seguro con autenticación de hardware
  - Aceleradores criptográficos para AES, 3DES, RSA y ECC
  - Generador de números aleatorios verdaderos (TRNG)
- **Sensado capacitivo:**
  - CSD (Capacitive Sigma-Delta) para detección de proximidad y tolerancia a líquidos
  - Sensado de capacitancia mutua con CSX
- **Alimentación:**

  - 5 V (USB)
  - 1.8 V a 3.3 V (VTARG)
- **Indicadores LED:**
  - Dos LEDs para retroalimentación del usuario
  - LED de estado KitProg2
- **Botones:**
  - Un botón de usuario
  - Un botón de reinicio
  - Un botón de modo KitProg2

3 Hardware

3.1 L298N H-bridge

El módulo L298N admite una entrada de tensión (*Vin*) entre 3V y 35V, con el pin de conexión a tierra (*GND*) justo al lado. La conexión de *V lógico* puede operar de dos maneras:

- Si el jumper del regulador está cerrado, se activa el regulador de tensión del L298N, proporcionando una salida de 5V en *V lógico*, que se puede utilizar para alimentar una placa Arduino.
- Si el jumper del regulador está abierto, se desactiva el regulador y se necesita alimentar la parte lógica del módulo con una tensión de 5V en *V lógico* para que funcione.

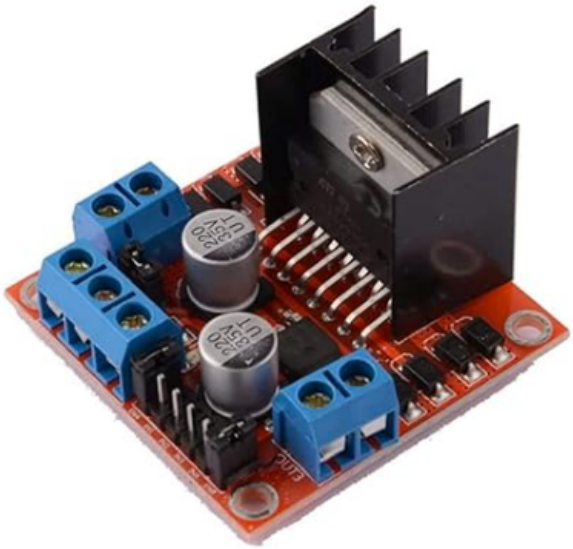


Figure 1: módulo L298N

3.1.1 Control de Motores DC

Las salidas para los motores A y B proporcionan la energía para mover los motores. Es importante tener en cuenta la polaridad al conectarlos para asegurar que los motores giren en la dirección deseada. Los pines *IN1* e *IN2* controlan el sentido de giro del motor A, mientras que los pines *IN3* e *IN4* hacen lo mismo para el motor B. El control del sentido de giro funciona de la siguiente manera:

- *IN1* en HIGH e *IN2* en LOW: el motor A gira en un sentido.
- *IN1* en LOW e *IN2* en HIGH: el motor A gira en el sentido opuesto.
- El mismo principio se aplica a los pines *IN3* e *IN4* para el motor B.

Para controlar la velocidad de los motores, se deben quitar los jumpers y usar los pines *ENA* y *ENB*, conectándolos a salidas PWM. Un valor PWM controla la velocidad de giro. Si los jumpers están colocados, los motores girarán siempre a la misma velocidad.

3.2 TXB0108 8-bit Noninverting Translator

3.2.1 Conexiones y Funcionamiento

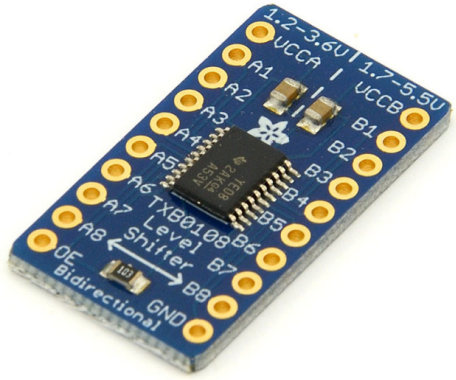


Figure 2: Diagrama del TXB0108

El traductor no inversor de 8 bits TXB0108 utiliza dos fuentes de alimentación configurables por separado. El puerto A está diseñado para seguir a *VCCA*. *VCCA* acepta cualquier voltaje de alimentación de 1.2 V a 3.6 V. El puerto B está diseñado para seguir a *VCCB*. *VCCB* acepta cualquier voltaje de alimentación de 1.65 V a 5.5 V. Esto permite una traducción bidireccional universal de bajo voltaje entre cualquiera de los nodos de voltaje de 1.2 V, 1.5 V, 1.8 V, 2.5 V, 3.3 V y 5 V. *VCCA* no debe exceder *VCCB*.

Cuando la entrada de habilitación de salida (OE) está en bajo, todas las salidas se colocan en estado de alta impedancia.

El TXB0108 está diseñado de modo que el circuito de entrada OE se suministra mediante *VCCA*.

Este dispositivo está completamente especificado para aplicaciones de apagado parcial utilizando *Ioff*. El circuito *Ioff* deshabilita las salidas, evitando el flujo de corriente dañina a través del dispositivo cuando está apagado.

3.3 WayinTop Sensor Fotosensible

3.3.1 Características y Funcionamiento

El *WayinTop Sensor Fotosensible* opera con un voltaje de 3.3 a 5V y proporciona una salida digital de conmutación (0 y 1). Su PCB mide aproximadamente 3.2 cm x 1.4 cm y tiene un pin header de 4 pines.

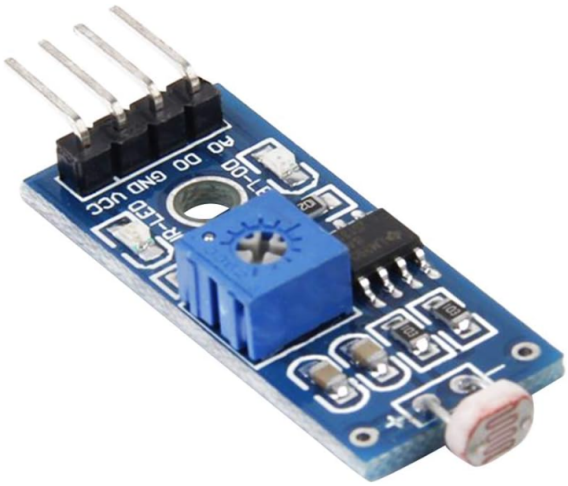


Figure 3: WayinTop Sensor Fotosensible

Este sensor es altamente sensible a la luz ambiental y se utiliza para detectar cambios en la intensidad lumínica. En condiciones de baja intensidad de luz, el módulo emite un nivel alto. Cuando la luz supera el umbral establecido, la salida cambia a nivel bajo. La salida de drenaje abierto (OD) se puede conectar a un microcontrolador para monitorear estos cambios o activar un módulo de relé como interruptor de luz.

La salida analógica (AO) del sensor permite obtener mediciones precisas de la luz ambiental a través de la conversión AD. Para su operación, el pin *VCC* se conecta a una fuente de 3.3 a 5V, el *GND* a tierra, y el *DO* proporciona la señal de conmutación TTL.

3.4 Módulo de Rango Ultrasónico HC-SR04

3.4.1 Características y Funcionamiento

Figure 4: Módulo de Rango Ultrasónico HC-SR04

El *Módulo de Rango Ultrasónico HC-SR04* proporciona una medición sin contacto en el rango de 2 cm a 400 cm, con una precisión de hasta 3 mm. Este módulo incluye transmisores ultrasónicos, un receptor y un circuito de control.

El principio básico de funcionamiento es el siguiente:

1. Se utiliza una señal IO de disparo de al menos 10 microsegundos de nivel alto.
2. El módulo envía automáticamente ocho pulsos de 40 kHz y detecta si hay una señal de pulso de retorno.
3. Si hay una señal de retorno, el tiempo de duración de la salida de nivel alto IO es el tiempo desde el envío del ultrasonido hasta su retorno.

La distancia se calcula con la fórmula:

$$\text{Distancia} = \left(\frac{\text{tiempo de nivel alto} \times \text{velocidad del sonido (340 m/s)}}{2} \right)$$

Parámetros Eléctricos:

- Voltaje de trabajo: DC 5V
- Corriente de trabajo: 15 mA
- Frecuencia de trabajo: 40 Hz
- Rango máximo: 4 m
- Rango mínimo: 2 cm
- Ángulo de medición: 15 grados
- Señal de entrada de disparo: pulso TTL de 10 microsegundos
- Señal de salida de eco: señal de nivel TTL proporcional a la distancia
- Dimensiones: 45 mm x 20 mm x 15 mm

Para su funcionamiento, el módulo se conecta de la siguiente manera:

- **VCC:** Conectar a la fuente de alimentación de 5V
- **Trig:** Entrada de pulso de disparo
- **Echo:** Salida de pulso de eco
- **GND:** Conectar a tierra

3.5 Youmile Encoder Speed

3.5.1 Características y Funcionamiento

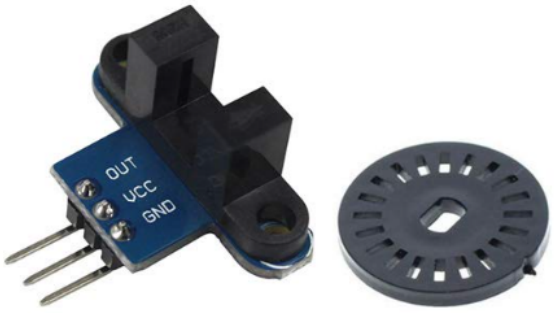


Figure 5: Youmile Encoder Speed

El *Youmile Encoder Speed* es un dispositivo que utiliza una rueda dentada con agujeros por donde pasa una señal láser. Al contar las interrupciones del láser, se puede medir la rotación de la rueda. Este encoder presenta las siguientes especificaciones:

- **Voltaje de operación:** 3.3V a 5V
- **Dimensiones:** 26.8 mm x 15 mm x 18.7 mm
- **Tamaño de los agujeros de montaje:** 3 mm

- **Ancho de la ranura:** 6 mm

Para su funcionamiento con un microcontrolador (MCU):

- **VCC:** 3.3V a 5V
- **GND:** Conectar a tierra
- **DOUT:** Conectar a una entrada digital del MCU

El Youmile Encoder Speed funciona midiendo las interrupciones del haz láser causadas por los agujeros en la rueda dentada. Cada interrupción corresponde a un pulso digital en la salida ($DOUT$) que puede ser leído por el microcontrolador.

Dado el número limitado de agujeros en la rueda, el tiempo de muestreo se ha limitado a 0.2 segundos, lo que corresponde a una frecuencia de muestreo de 5 Hz.

3.6 Motor DC

3.6.1 Características y Funcionamiento

Figure 6: Motor DC

El *Motor DC* presenta las siguientes características:

- **Voltaje nominal:** 3 6 V
- **Corriente continua sin carga:** 150 mA +/- 10
- **Min. Velocidad de funcionamiento (3 V):** 90 +/- 10
- **Min. Velocidad de funcionamiento (6 V):** 200 +/- 10
- **Par de torsión:** 0,15 Nm 0,60 Nm
- **Par de bloqueo (6V):** 0.8 kg.cm
- **Relación de engranajes:** 1:48
- **Material:** Plástico
- **Dimensiones del cuerpo:** 70 x 22 x 18 mm
- **Peso del producto:** 29 g / 1.02 oz
- **Ruido:** ¡ 65 dB

3.6.2 Velocidad Máxima y Control PWM

La velocidad máxima del motor se ha calculado en aproximadamente 0.5 m/s a vacío. Al controlar el motor mediante PWM, se observó que por debajo de un duty cycle del 25

Aunque la relación entre el duty cycle y la velocidad no es lineal, se ha considerado lineal entre los puntos de muestra de los duty cycles para simplificar el control.

Esta aproximación permite un control efectivo de la velocidad del motor ajustando el duty cycle del PWM, facilitando la implementación en aplicaciones prácticas.

4 Software

La implementación de software del robot móvil está estructurada para gestionar de manera eficiente los diversos componentes que permiten su funcionamiento autónomo. El software está dividido en varias librerías especializadas, cada una encargada de manejar diferentes aspectos del robot. Las principales librerías incluyen: `control.h`, `encoder.h`, `HC-SR04.h`, `light_sensor.h`, y `odometry.h`.

4.1 Control de Velocidad y Dirección

La librería `control.h` contiene las funciones necesarias para el control de velocidad y dirección de las ruedas del robot. La función principal `set_speed()` ajusta las velocidades de las ruedas utilizando señales PWM generadas por el microcontrolador. Dependiendo de la velocidad deseada, se determina la dirección de rotación (adelante o atrás) y se ajusta el duty cycle de la señal PWM en consecuencia.

4.2 Medición de Velocidad con Encoders

La librería `encoder.h` gestiona la medición de la velocidad de las ruedas mediante encoders. Estos encoders utilizan una rueda dentada con agujeros y un sensor láser para detectar el movimiento. Las interrupciones generadas por los pulsos del encoder son manejadas por las funciones en esta librería, que calculan la velocidad angular de cada rueda. La función `encoder_speed()` aplica un filtro de media móvil exponencial para suavizar las lecturas de velocidad.

4.3 Detección de Obstáculos con Sensor de Ultrasonidos

La librería `HC-SR04.h` se encarga de la detección de obstáculos mediante un sensor de ultrasonidos HC-SR04. Esta librería incluye manejadores de interrupciones que procesan las señales de eco del sensor, calculando la distancia a los objetos y aplicando un filtro de suavizado para mejorar la precisión.

4.4 Sensor de Luz para Adaptaciones Ambientales

La librería light_sensor.h integra un sensor de luz para adaptar el comportamiento del robot según las condiciones de iluminación del entorno. Este sensor mide la intensidad de la luz ambiental y envía los datos al microcontrolador. Las funciones en esta librería procesan estas lecturas para ajustar parámetros operativos del robot, asegurando un funcionamiento eficiente en diversas condiciones de luz. En ausencia de luz, el sensor detiene el robot automáticamente para evitar movimientos inseguros.

4.5 Odómetro y Navegación

La librería `odometry.h` es responsable de calcular la posición del robot en función de las velocidades de las ruedas y el tiempo transcurrido. Utilizando un modelo cinemático diferencial, las funciones en esta librería actualizan las coordenadas x e y y la orientación θ del robot. También se calcula el error de posicionamiento respecto a un objetivo deseado y se desactiva el control cuando el robot alcanza el objetivo con suficiente precisión.

4.6 Encoder

4.6.1 Descripción del Esquemático

El esquemático mostrado en la Figura 7 se compone de un contador de temporizador, un reloj y dos entradas de interrupción de encoders. A continuación se describe cada componente y su función en el sistema.

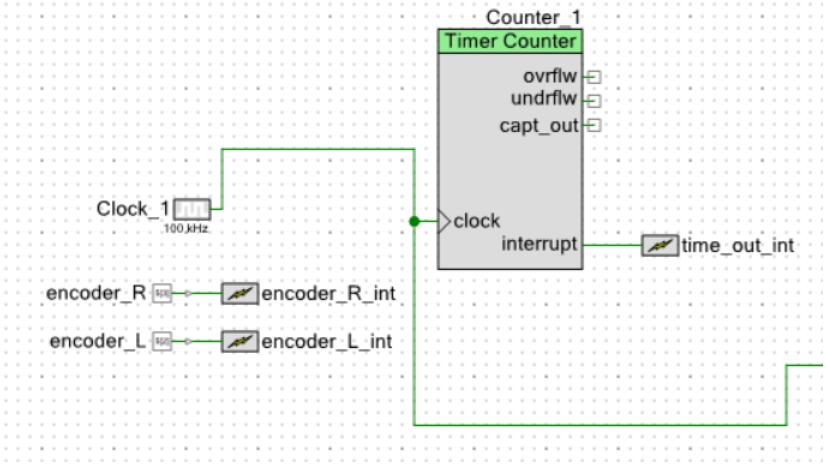


Figure 7: Esquemático del sistema de control del robot móvil.

`Reloj`

El componente `Clock_1` genera una señal de reloj con una frecuencia de 100 kHz. Esta señal de reloj se utiliza para alimentar el `Counter_1`, proporcionando una base de tiempo para su funcionamiento.

`Contador de Temporizador`

El `Counter_1` es un temporizador configurado para contar los pulsos del `Clock_1`. Este contador genera una interrupción (`time_out_int`) cuando alcanza un período predefinido. En este caso, el período del contador está configurado a 2500 microsegundos, lo que resulta en una frecuencia de 0.005 kHz.

`Entradas de Interrupción de Encoders`

El esquema incluye dos entradas de interrupción, `encoder_R_int` y `encoder_L_int`, que están conectadas a los encoders derecho e izquierdo del sistema respectivamente. Estas interrupciones se utilizan para contar los pulsos generados por los encoders cuando las ruedas del robot giran.

4.6.2 Código de Manejo de Interrupciones

El siguiente código se utiliza para manejar las interrupciones generadas por los encoders. Cada vez que ocurre una interrupción, se incrementa un contador de ticks correspondiente (`tick_L` para el encoder izquierdo y `tick_R` para el encoder derecho).

```
void encoder_L_IRQ_Handler(void) {
    NVIC_ClearPendingIRQ(encoder_L_int_cfg.intrSrc);
    ++tick_L;
}

void encoder_R_IRQ_Handler(void) {
    NVIC_ClearPendingIRQ(encoder_R_int_cfg.intrSrc);
    ++tick_R;
}
```

- *Función* `encoder_L_IRQ_Handler`: Esta función se llama cuando se detecta una interrupción en el encoder izquierdo. Limpia la interrupción pendiente y aumenta el contador de ticks izquierdo (`tick_L`).

- *Función* `encoder_R_IRQ_Handler`: Similarmente, esta función se llama cuando se detecta una interrupción en el encoder derecho. Limpia la interrupción pendiente y aumenta el contador de ticks derecho (`tick_R`).

4.7 Control de PWM para el Movimiento de las Ruedas

4.7.1 Descripción del Esquemático

El esquemático mostrado en la Figura 8 se compone de un contador de temporizador, un reloj y dos módulos PWM que controlan las ruedas izquierda y derecha del robot. A continuación se describe cada componente y su función en el sistema.

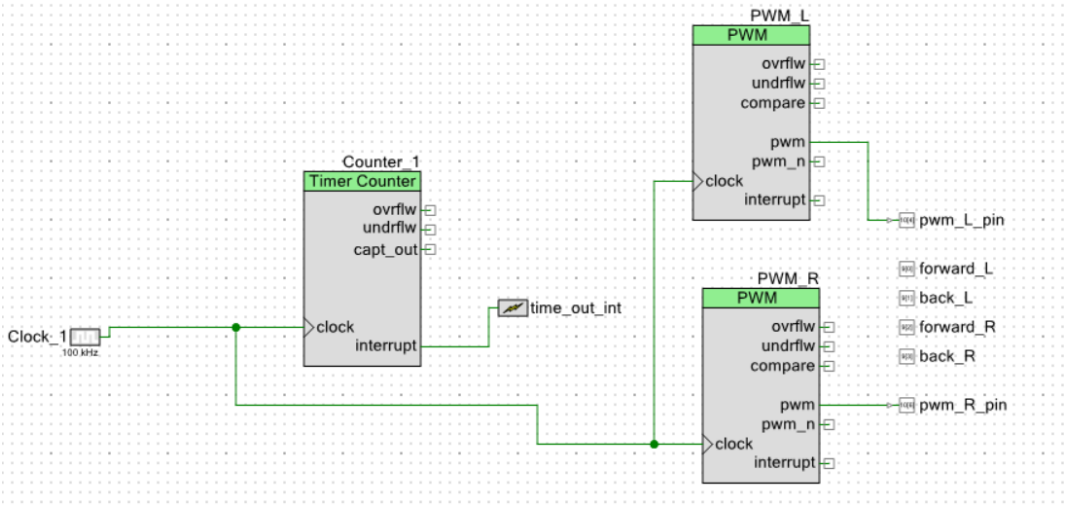


Figure 8: Esquemático del sistema de control del robot móvil.

Reloj

El componente `Clock_1` genera una señal de reloj con una frecuencia de 100 kHz. Esta señal de reloj se utiliza para alimentar el `Counter_1` y los módulos `PWM_L` y `PWM_R`, proporcionando una base de tiempo para su funcionamiento.

Contador de Temporizador

El `Counter_1` es un temporizador configurado para contar los pulsos del `Clock_1`. Este contador genera una interrupción (`time_out_int`) cuando alcanza un período predefinido, utilizado para la realizacion del programa principal.

Módulos PWM

Los módulos `PWM_L` y `PWM_R` controlan la velocidad de las ruedas izquierda y derecha respectivamente. Estos módulos generan señales PWM cuyo duty cycle se ajusta para controlar la velocidad de los motores de las ruedas.

Pines de Control del Movimiento

El movimiento del robot se controla mediante la configuración de los pines de los módulos PWM y los pines de dirección: - `forward_L` y `back_L`: Controlan la dirección de la rueda izquierda. - `forward_R` y `back_R`: Controlan la dirección de la rueda derecha. - `pwm_L_pin` y `pwm_R_pin`: Salidas PWM que controlan la velocidad de las ruedas izquierda y derecha respectivamente.

4.7.2 Código de Manejo de PWM

La función `set_speed()` es responsable de ajustar el duty cycle de los módulos PWM para cambiar la velocidad de las ruedas. A continuación se muestra el código utilizado para este propósito:

```
void set_speed(float v_R_des, float v_L_des) {
    int level_R = _calcola_velocita(v_R_des);
    int level_L = _calcola_velocita(v_L_des);

    uint32_t status = Cy_SysLib_EnterCriticalSection();
    _setSpeed_direction(v_L_des, v_R_des);
    PWM_L_SetCompare0(_hash_function_PWM(level_L));
    PWM_R_SetCompare0(_hash_function_PWM(level_R));
```

```
    Cy_SysLib_ExitCriticalSection(status);
}
```

- *Función* ***set_speed***: Esta función ajusta la velocidad de las ruedas calculando el nivel de velocidad deseado y configurando el duty cycle correspondiente en los módulos PWM. Utiliza la función `_calcola_velocita` para convertir la velocidad deseada en un nivel de PWM y la función `_hash_function_PWM` para calcular el valor del duty cycle. Luego, establece la dirección del movimiento y ajusta los valores de comparación (*set compare*) de los módulos PWM para reflejar los cambios en el duty cycle.

Este diseño permite un control preciso y eficiente de las velocidades de las ruedas, ajustando dinámicamente los parámetros del PWM según las necesidades de navegación del robot.

4.8 Sensor de Luminosidad

4.8.1 Descripción del Esquemático

El esquemático mostrado en la Figura 9 se compone de un contador de temporizador, un reloj y un módulo ADC que mide la intensidad de la luz. A continuación se describe cada componente y su función en el sistema.

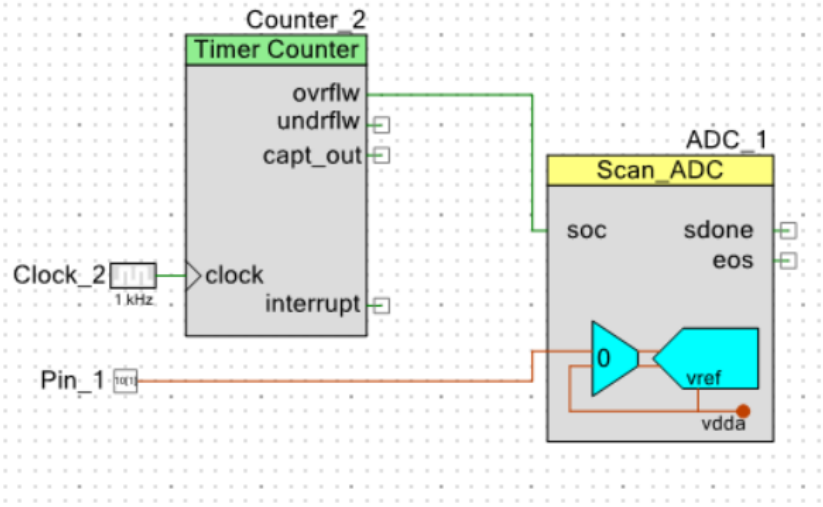


Figure 9: Esquemático del sistema de medición de luminosidad del robot móvil.

Reloj

El componente `Clock_2` genera una señal de reloj con una frecuencia de 1 kHz. Esta señal de reloj se utiliza para alimentar el `Counter_2`, proporcionando una base de tiempo para su funcionamiento.

Contador de Temporizador

El `Counter_2` es un temporizador configurado para contar los pulsos del `Clock_2`. Este contador manda un signal cuando alcanza un período predefinido tramite un signal de overflow. En este caso, el contador sincroniza las mediciones del ADC.

ADC (Convertidor Analógico-Digital)

El módulo `ADC_1` se utiliza para medir la intensidad de la luz detectada por el sensor conectado al pin `Pin_1`. El ADC convierte la señal analógica del sensor en valores digitales que pueden ser procesados por el microcontrolador. El límite para generar una interrupción se ha establecido en 2.4V.

4.8.2 Código de Manejo del Sensor de Luminosidad

El siguiente código se utiliza para inicializar y manejar las interrupciones generadas por el ADC cuando la intensidad de la luz supera el límite establecido. La variable `enable` se utiliza para detener el robot cuando la luz es insuficiente.

```
void LightSensor_Init(void) {
```

```
    ADC_1_StartEx(ADC_1_my_ISR);
    ADC_1_SetEosMask(0);
    ADC_1_SetLimitMask(1);
}

void ADC_1_my_ISR(void) {
    // Pulisce l'interrupt in sospeso
    //NVIC_ClearPendingIRQ(ADC_1_IRQ_cfg.intrSrc);

    // Inizializza il campione a 0
    sample_uV = 0;

    // Ottiene il risultato grezzo dell'ADC
    int16_t raw_counts = ADC_1_GetResult16(0);

    // Converte il risultato grezzo in conti
    int16_t counts = Cy_SAR_RawCounts2Counts(ADC_1_SAR__HW, 0, raw_counts);

    // Converte i conti in microvolt
    sample_uV = ADC_1_CountsTo_uVolts(0, counts);
    enable = 0;
    // Ottiene lo stato dell'interrupt
    uint32_t intr_status = Cy_SAR_GetInterruptStatus(ADC_1_SAR__HW);

    // Pulisce l'interrupt di range
    Cy_SAR_ClearRangeInterrupt(ADC_1_SAR__HW, intr_status);
}
```

- *Función* `LightSensor_Init`: Esta función inicializa el módulo ADC y configura la masquera por el límite superior para generar una interrupción cuando la señal analógica del sensor de luz supere los 2.4V y desabilitar la interrucion Eos.

- *Función* `ADC_1_my_ISR`: Esta es la rutina de servicio de interrupción personalizada para manejar las interrupciones del ADC. Limpia la interrupción pendiente, lee y convierte el valor analógico a digital, y si la intensidad de la luz es insuficiente, la variable `enable` se establece en 0, deteniendo el robot.

Este diseño permite que el robot responda a las condiciones de iluminación del entorno, deteniéndose automáticamente cuando la luz es insuficiente para operar de manera segura.

4.9 Sensor de Ultrasonidos

4.9.1 Descripción del Esquemático

El esquemático mostrado en la Figura 10 se compone de un contador de temporizador, un reloj y los pines de entrada y salida del sensor de ultrasonidos. A continuación se describe cada componente y su función en el sistema.

Relojes

El esquema utiliza dos relojes: - `Clock_3` genera una señal de reloj con una frecuencia de 2 MHz para el `Counter_3`. - `Clock_4` genera una señal de reloj con una frecuencia de 1 MHz para el `Counter_4`.

Contadores de Temporizador

- `Counter_3`: Este temporizador cuenta los pulsos del `Clock_3` y genera interrupciones utilizadas para medir el tiempo de vuelo de la señal ultrasónica. - `Counter_4`: se utiliza para medir el tiempo de respuesta del eco.

Pines de Control del Sensor de Ultrasonidos

- `echo`: Pin de entrada que recibe el eco de la señal ultrasónica. - `trigger`: Pin de salida que envía pulsos para disparar la señal ultrasónica.

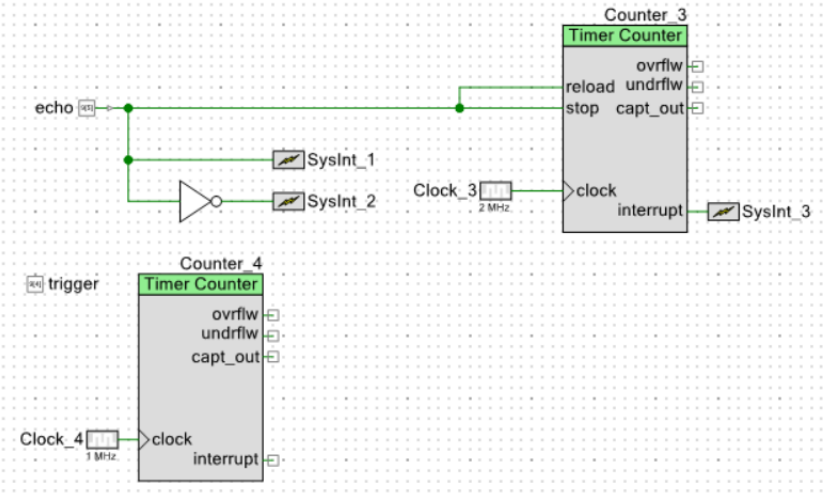


Figure 10: Esquemático del sistema de medición de distancia con el sensor de ultrasonidos del robot móvil.

4.9.2 Código de Manejo del Sensor de Ultrasonidos

El siguiente código se utiliza para manejar las interrupciones generadas por el sensor de ultrasonidos y calcular la distancia basada en el tiempo de vuelo de la señal.

```
void echo_sub_Handler() {
    // Cancella l'interruzione pendente
    NVIC_ClearPendingIRQ(SysInt_1_cfg.intrSrc);

    // Entra nella sezione critica
    uint32_t status = Cy_SysLib_EnterCriticalSection();
    echo_bajo_called = false;
    // Imposta il contatore a 65535
    Counter_4_SetCounter(65535u);

    // Esci dalla sezione critica
    Cy_SysLib_ExitCriticalSection(status);
}

void echo_bajo_Handler() {
    // Cancella l'interruzione pendente
    NVIC_ClearPendingIRQ(SysInt_2_cfg.intrSrc);

    // Entra nella sezione critica
    uint32_t status = Cy_SysLib_EnterCriticalSection();

    // Calcola il valore del contatore
    value_counter = 65535u - Counter_4_GetCounter();

    // Calcola la distanza
    distancia = value_counter / 58.0;

    // Applica un filtro di smoothing
    filtered_distance = alpha * distancia + (1 - alpha) * filtered_distance;

    if (filtered_distance <= barriera && filtered_distance > 3.0) {
        Cy_SCB_UART_PutString(UART_1_HW, "\n\nTROPPO VICINO\n\n");
```

```
        enable_ultrasonic_sensor = 0;
    } else {
        sprintf(imp, "%f cm\n", filtered_distance);
        Cy_SCB_UART_PutString(UART_1_HW, imp);
        enable_ultrasonic_sensor = 1;
    }

    // Riavvia il contatore 3
    Counter_3_Start();

    // Segnala che echo_bajo_Handler è stato chiamato
    echo_bajo_called = true;

    // Esci dalla sezione critica
    Cy_SysLib_ExitCriticalSection(status);
}

void SysInt_3_IRQHandler(void) {
    // Cancella l'interruzione del contatore 3
    Counter_3_ClearInterrupt(Counter_3_config.interruptSources);
    uint32_t status = Cy_SysLib_EnterCriticalSection();
    // Controlla lo stato del segnale echo
    if (Cy_GPIO_Read(echo_PORT, echo_NUM) == 0) {
        if (var_aus_trigger == 1) {
            Cy_GPIO_Write(trigger_PORT, trigger_NUM, 1);
            var_aus_trigger = 0;
            Counter_3_SetCounter(1u);
        } else {
            Cy_GPIO_Write(trigger_PORT, trigger_NUM, 0);
            var_aus_trigger = 1;
            Counter_3_SetCounter(19u);
        }
    }
    Cy_SysLib_ExitCriticalSection(status);
}
```

Problemas y Soluciones Intentadas

Durante las pruebas, se observó un comportamiento incorrecto del sensor de ultrasonidos mientras el robot estaba en movimiento. Este problema podría deberse a interferencias de hardware o a una implementación incorrecta del software. Se realizaron varios intentos para corregir este problema, incluyendo:

- Revisión del sistema de interrupciones y manejo de secciones críticas.
- Adición de filtros de software para reducir el ruido.
- Implementación de filtros de hardware con condensadores (no presente en la versión final).
- Adición de variables de control para asegurar la correcta ejecución del flujo del código (no presente en la versi on final).
- Simplificación de la rutina de interrupción para una gestión más rápida del ISR (no presente en la versi on final).

Sin embargo, ninguno de estos intentos logró resolver completamente el problema, lo que sugiere la necesidad de una revisión más exhaustiva del diseño de hardware y software.

4.10 Código Principal

El código principal del robot móvil incluye la configuración de interrupciones, la inicialización de periféricos y el bucle principal que gestiona el control del robot. A continuación se describen las principales partes del código:

4.10.1 Manejador de Interrupciones del Temporizador

La función `SysTime_out_IRQHandler` maneja las interrupciones generadas por el temporizador `Counter_1`. Esta función se encarga de limpiar la interrupción, invertir el estado de los LEDs y establecer un flag que indica que es momento de ejecutar el control del robot.

```
void SysTime_out_IRQHandler(void) {
    Counter_1_ClearInterrupt(Counter_1_config.interruptSources);
    control_flag = true; // Imposta il flag per segnalare che è tempo di eseguire il controllo
    Cy_GPIO_Inv(GREEN_LED_PORT, GREEN_LED_NUM);
    Cy_GPIO_Inv(RED_LED_PORT, RED_LED_NUM);
}
```

4.10.2 Función Principal

La función `main` es el núcleo del programa y se encarga de la configuración inicial del sistema, la inicialización de periféricos, y la gestión del bucle principal del control del robot.

```
int main(void) {
    __enable_irq(); /* Enable global interrupts. */

    UART_1_Start();
    Cy_SysInt_Init(&time_out_int_cfg, SysTime_out_IRQHandler);
    Cy_SysInt_Init(&encoder_L_int_cfg, encoder_L_IRQ_Handler);
    Cy_SysInt_Init(&encoder_R_int_cfg, encoder_R_IRQ_Handler);

    NVIC_ClearPendingIRQ(time_out_int_cfg.intrSrc);
    NVIC_EnableIRQ(time_out_int_cfg.intrSrc);

    NVIC_ClearPendingIRQ(encoder_L_int_cfg.intrSrc);
    NVIC_EnableIRQ(encoder_L_int_cfg.intrSrc);

    NVIC_ClearPendingIRQ(encoder_R_int_cfg.intrSrc);
    NVIC_EnableIRQ(encoder_R_int_cfg.intrSrc);

    Cy_SysInt_Init(&BtnIsr_cfg, BtnIsr_handler);
    NVIC_ClearPendingIRQ(BtnIsr_cfg.intrSrc);
    NVIC_EnableIRQ(BtnIsr_cfg.intrSrc);

    PWM_L_SetCompare0(0);
    PWM_R_SetCompare0(0);

    LightSensor_Init();
    Counter_2_Start();

    // Configuración del sensor ultrasónico
    Cy_SysInt_Init(&SysInt_1_cfg, echo_sub_Handler);
    Cy_SysInt_Init(&SysInt_2_cfg, echo_bajo_Handler);

    NVIC_ClearPendingIRQ(SysInt_1_cfg.intrSrc);
    NVIC_EnableIRQ(SysInt_1_cfg.intrSrc);
```

```
    NVIC_ClearPendingIRQ(SysInt_2_cfg.intrSrc);
    NVIC_EnableIRQ(SysInt_2_cfg.intrSrc);

    Counter_4_Enable();
    Counter_4_Start();

    Cy_SysInt_Init(&SysInt_3_cfg, SysInt_3_IRQHandler);
    NVIC_ClearPendingIRQ(SysInt_3_cfg.intrSrc);
    NVIC_EnableIRQ(SysInt_3_cfg.intrSrc);
    Counter_3_Start();

    // Inicio de los módulos PWM
    PWM_L_Start();
    PWM_R_Start();

    for (;;) {
        // Bucle principal
        if (control_flag) {
            control_flag = false; // Resetta il flag
            if (enable_ultrasonic_sensor == 1 && enable == 1) {
                uint32_t status = Cy_SysLib_EnterCriticalSection();
                encoder_speed();
                Cy_SysLib_ExitCriticalSection(status);



                status = Cy_SysLib_EnterCriticalSection();
                Odometry();
                Cy_SysLib_ExitCriticalSection(status);



                status = Cy_SysLib_EnterCriticalSection();
                Control_P();
                Cy_SysLib_ExitCriticalSection(status);



                status = Cy_SysLib_EnterCriticalSection();
                set_speed(v_R_des, v_L_des);
                Cy_SysLib_ExitCriticalSection(status);
            }
        }
    }
}
```

Inicialización del Sistema

- Habilitación de las interrupciones globales.
- Inicialización de la UART para comunicación.
- Configuración de las interrupciones para el temporizador y los encoders.
- Inicialización del sensor de luz y los contadores.
- Configuración de las interrupciones para el sensor ultrasónico.

Bucle Principal

- El bucle principal verifica el `control_flag` para determinar cuándo ejecutar el control del robot.
- Si las condiciones son adecuadas (sensores habilitados y bandera de control activada), se realizan las siguientes acciones en orden:
  - Lectura de los encoders para obtener la velocidad de las ruedas.
  - Actualización de la odometría para determinar la posición del robot.
  - Ejecución del controlador proporcional para ajustar la velocidad y dirección del robot.
  - Configuración de los PWM para ajustar las velocidades de las ruedas.

Este diseño modular y estructurado permite que el robot funcione de manera autónoma.

5 Asignación de Pines

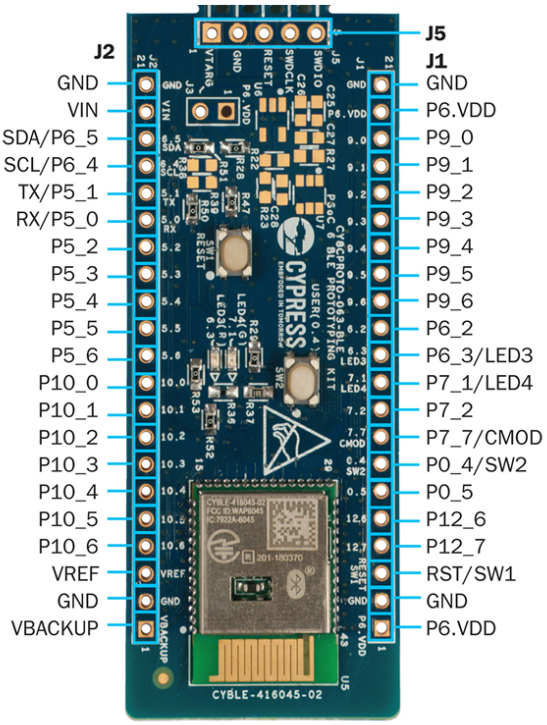


Figure 11: Pins

La configuración de los pines es crucial para el correcto funcionamiento del sistema del robot móvil. A continuación se detalla la asignación de pines utilizada en el proyecto, especificando la función de cada pin y el puerto correspondiente.

- **UART_1_rx**: **Port**: P5[0]
- **UART_1_tx**: **Port**: P5[1]
- **back_L**: **Port**: P9[1]
- **back_R**: **Port**: P9[3]
- **echo**: **Port**: P5[5]
- **encoder_L**: **Port**: P5[2]
- **encoder_R**: **Port**: P5[3]

- **forward_L**: **Port**: P9[0]
- **forward_R**: **Port**: P9[2]
- **GREEN_LED**: **Port**: P7[1]
- **Pin_1**: **Port**: P10[1]
- **pwm_L_pin**: **Port**: P10[4]
- **pwm_R_pin**: **Port**: P10[6]
- **RED_LED**: **Port**: P6[3]
- **trigger**: **Port**: P9[4]
- **UserBtn**: **Port**: P0[4]

Cada pin tiene una función específica que se detalla a continuación:

- **UART_1_rx** y **UART_1_tx**: Pines utilizados para la comunicación UART.
- **back_L** y **back_R**: Controlan el movimiento hacia atrás de las ruedas izquierda y derecha.
- **echo**: Pin de entrada para el eco del sensor de ultrasonidos.
- **encoder_L** y **encoder_R**: Pines de entrada para los encoders de las ruedas izquierda y derecha.
- **forward_L** y **forward_R**: Controlan el movimiento hacia adelante de las ruedas izquierda y derecha.
- **GREEN_LED** y **RED_LED**: Pines de control para los LEDs verde y rojo, utilizados para indicar el estado del sistema.
- **Pin_1**: Pin de entrada para el sensor de luz.
- **pwm_L_pin** y **pwm_R_pin**: Pines de salida PWM para controlar la velocidad de las ruedas izquierda y derecha.
- **trigger**: Pin de salida para disparar el sensor de ultrasonidos.
- **UserBtn**: Pin de entrada para un botón de usuario.

6 Conclusiones y Propuestas de Mejora

En este proyecto, se ha diseñado y desarrollado un robot móvil con capacidades de navegación autónoma utilizando diversos sensores e interruptores para controlar su movimiento. A lo largo del proceso, se han enfrentado y resuelto varios desafíos relacionados con la integración de hardware y software, la gestión de interrupciones y el filtrado de señales.

El sistema actual demuestra una buena funcionalidad básica, permitiendo al robot moverse y responder a las entradas de los sensores de manera efectiva. Sin embargo, se identificaron algunos problemas persistentes, especialmente relacionados con el sensor de ultrasonidos, que no siempre se comporta de manera confiable cuando el robot está en movimiento. Esto sugiere la necesidad de una mayor robustez en el diseño del sistema.

6.1 Propuestas de Mejora

Para mejorar la precisión, la fiabilidad y la capacidad del robot, se proponen las siguientes mejoras:

- **Introducción de Sensores Adicionales para la Odometría**: Integrar sensores adicionales como encoders de alta resolución y sensores de giroscopio y acelerómetro (IMU) para mejorar la precisión de la odometría del robot. Esto permitirá una estimación más precisa de la posición y la orientación del robot.
- **Filtro de Kalman**: Implementar un filtro de Kalman para la fusión de datos de múltiples sensores. Este filtro permitirá una mejor estimación del estado del robot (posición, velocidad, etc.) al combinar las lecturas de diferentes sensores y reducir el impacto del ruido en las mediciones.
- **Cámara y Visión por Computador**: Añadir una cámara al sistema del robot para implementar algoritmos de visión por computador. Esto permitirá al robot detectar y evitar obstáculos, así como realizar tareas más complejas como el seguimiento de objetos y la navegación basada en características visuales.
- **Algoritmo de Control Predictivo Basado en Modelos (MPC)**: Implementar un algoritmo de control predictivo basado en modelos (MPC) para mejorar la capacidad del robot de seguir trayectorias y responder a cambios en el entorno de manera más eficiente. El MPC permite predecir el comportamiento futuro del robot y optimizar su trayectoria en tiempo real.
- **Integración con ROS2**: Migrar el sistema de control y gestión del robot a ROS2 (Robot Operating System 2). ROS2 proporciona una infraestructura robusta y escalable para el desarrollo de aplicaciones robóticas, facilitando la comunicación entre diferentes componentes del sistema, la integración de nuevos sensores y actuadores, y la implementación de algoritmos avanzados de control y planificación.

Estas mejoras no solo aumentarán la precisión y la fiabilidad del sistema actual, sino que también abrirán nuevas posibilidades para aplicaciones más avanzadas y complejas, permitiendo que el robot móvil sea utilizado en una variedad de escenarios y tareas más exigentes.

En resumen, el proyecto ha logrado sus objetivos iniciales, pero aún hay un amplio margen para la mejora y expansión del sistema. Con las propuestas mencionadas, el robot móvil puede evolucionar hacia una plataforma de investigación y desarrollo más potente y versátil.