Informe practica de laboratorio 8

Universidad Sergio Arboleda
Sistemas Embebidos

Estudiantes:
- Sharay Valentina Galvis Prieto.
- Juan Diego Moreno Cárdenas.

Introducción:

El presente informe describe el desarrollo e implementación de un sistema embebido utilizando el microcontrolador STM32F411. Para este laboratorio, el objetivo principal fue desarrollar un sistema de control de movimiento en línea recta con perfil trapezoidal y cálculo de odometría para un robot tipo Micromouse con actualización periódica, integrando conocimientos de laboratorios anteriores. Además el diseño y la creación de la placa de circuito impreso (PCB) y el respectivo montaje del robot, de tal manera que el robot se desplace de forma autónoma manteniendo trayectoria recta, calculando su posición mediante los encoders de cuadratura que poseen los motores utilizados, y reciba parámetros de movimiento vía USB desde una interfaz hecha en QT Creator por medio del protocolo de comunicación creado en laboratorios anteriores, en donde se le envían los parámetros de Distancia a recorrer (mm), la Velocidad máxima (mm/s) y la Aceleración (mm/s^2) .

Entre los diferentes objetivos que se plantearon se encuentra, el establecer una comunicación USB mediante el puerto COM, para la transmisión de datos, realizar el montaje completo del robot, ubicando la PCB (con el circuito implementado) encima de la base en madera proporcionada por el profesor, de forma en que se encuentre estable. Adicionalmente, se implementaron las baterías de tal manera que se pueda acceder fácilmente a ellas para poder cargarlas. Posteriormente, se incluyó el diseño del perfil trapezoidal de velocidad, el cálculo de la odometría utilizando las ecuaciones del movimiento rectilíneo uniformemente acelerado, se implementó un control feedforward, el uso de un botón físico como disparador de movimiento y por último, la transmisión tanto del valor de la velocidad (mm/s) como de la distancia a la que se espera mantener el sistema de una pared (mm), desde la interfaz gráfica desarrollada en QT Creator. Además, todo el proyecto se reestructuró con el uso de librerías en donde se encuentran todas las funciones utilizadas.

Para cumplir con estos objetivos, se trabajó en la integración de todos los sistemas desarrollados en laboratorios anteriores, se utilizó el mismo circuito implementado que se ha venido construyendo gracias a la elaboración de laboratorios anteriores y se utilizó la misma configuración del reloj, de la comunicación USB y de los diferentes periféricos y módulos utilizados (Timers, ADC, DMA). Todo el desarrollo de este laboratorio cuenta con las funcionalidades que se desarrollaron en laboratorios anteriores. Este laboratorio permitió reforzar conceptos sobre uso de librerías, manejo en el almacenamiento de datos, tipos de variables y funcionamiento de la transmisión de datos.

Materiales utilizados:

- Microcontrolador STM32F411.
- ST-Link V2.
- Cable USB.
- Base ROBOT.
- 2 motorreductores con encoders de cuadratura.
- 2 llantas.
- puente H TB6612FNG.
- PCB.
- Capacitores.
- Resistencias.
- MOSFET 2n7000.
- Led IR emisor.
- Led IR receptor.
- pulsador.
- 2 LEDs indicadores.
- Cables de conexión.

Softwares utilizados:

- Stmcube IDE.
- QT Creator.
- Altium Designer.
- Visual studio code – extensión teleplot.

Procedimiento:

El procedimiento para la implementación del sistema se desarrolló en dos etapas, la primera fue el diseño y fabricación de la PCB para el circuito, y la segunda etapa fue el desarrollo del laboratorio #8.

Etapa de diseño y fabricación de la PCB:

En esta parte del laboratorio se detalla la primera etapa del laboratorio:

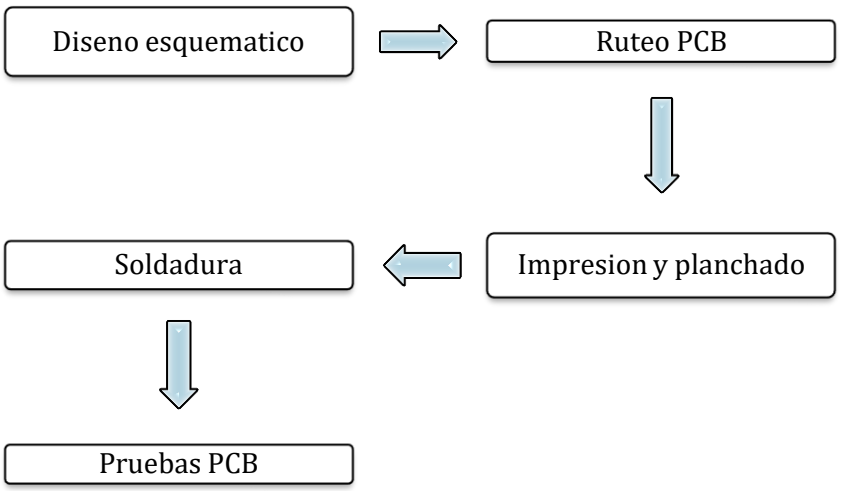

Fig.01. Proceso del desarrollo de la PCB.

Inicialmente, se desarrolló la PCB del sistema en Altium Designer, optando por un diseño de dos capas debido a la gran cantidad de conexiones necesarias entre los componentes. Se realizó el Diagrama esquemático incluido en Anexos, el cual contiene el circuito implementado en la protoboard de laboratorios anteriores y además se agregó lo siguiente:

- Un circuito para la fuente del robot; debido a que las baterías son dos celdas de litio en serie y cargadas al máximo suministran 8.4V a 2000 mA/h, como se puede observar en la *Tabla 01* es necesario regular este voltaje, lo cual se logró por medio de la implementación de un regulador MINI-360, que regula el voltaje de 7.8V a 5V. Además se agregó un regulador LM117DT, con el que se pasa el voltaje de 5V a 3.3V.

Voltaje máximo	8.4 V	4.2 V por celda al 100%
Voltaje nominal total	7.4 V	3.7 V por celda
Voltaje mínimo seguro	6.0 V	3.0 V por celda

Tabla.01. Entrega de voltaje suministrado por las baterías.

Gracias a este circuito implementado, se obtuvieron todos los voltajes necesarios para el funcionamiento de todos los componentes utilizados en el robot. Para facilidad de conexión y desconexión de la batería se implementó una bornera, por el tema de recarga. Y se implementó un swich para el suministro de energía a todo el circuito, que representa el estado de encendido y apagado del robot.

Posteriormente, el voltaje total de las baterías ingresa al pin Vm del puente H TB6612FNG para la alimentación de los motores, y por medio de la conexión de un divisor de voltaje se asegura que a la tarjeta Stm32F411 no se le suministre más de 3.3V como se muestra en la *Fig.01*. Por medio de *Ecu.01,* se calculan los divisores de voltaje para los diferentes estados de carga de la batería como se evidencia en la tabla *Tabla.02* con un capacitor de 10uF a 50V al ADC canal 4 de la Stm32F411, para estabilizar y filtrar la señal de voltaje de batería que será muestreada por el ADC, mejorando la precisión y evitando ruido.

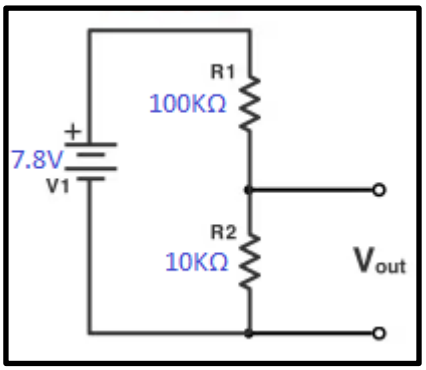

Fig.02. Divisor de voltaje para conexión del ADC canal 4.

$$Vout = V1 \times \frac{R2}{R2 + R1}$$

Ecu.01. Divisor de voltaje.

Voltaje de batería (V1)	*Voltaje enviado al canal 4 ADC (Vout)*
8.4 V (100%)	*0.763*
7.4 V (nominal)	*0.672 V*
6.6 V	*0.600 V*
6.0 V (descargada)	*0.545 V*
5.0 V	*0.455 V*

Tabla.02. Entrega de voltaje suministrado por las baterías.

Estos valores presentados en la *Tabla.02*, indican el valor de voltaje ADC que se le entregará a la tarjeta que están entre 0.5 V y 0.76 V aproximadamente. Lo cual nos asegura que la tarjeta no recibirá voltajes superiores a los 3.3V debido a que los valores están muy por debajo de este valor. Y posteriormente se podría implementar un seguimiento para el nivel de carga de la batería por medio de los valores que se reciben al ADC de la tarjeta.

- Un circuito para el buzzer, debido a que se implementó uno para el circuito que funciona a 5V el conector positivo está conectado a la salida del regulador MINI -360 que suministra 5V, y en el conector negativo del buzzer está conectado al colector de un transistor NPN (MMBT2222A), este actúa como un interruptor electrónico para permitir mayor corriente al buzzer, de tal manera que su funcionamiento es de la siguiente manera:

  la tarjeta envía una señal por medio de una resistencia y el transistor, de tal manera que cuando la señal está en alto (1), la base del transistor se activa, el transistor conduce, por ende el buzzer suena (porque la corriente fluye de +5V a el buzzer y el colector al emisor y por último a GND) y cuando la señal está en bajo (0) simplemente el transistor no conduce, no hay corriente, y el buzzer está apagado.

- Se agregaron dos LED's, los cuales están disponibles para la generación de alarmas visuales.

- Se incorporó un conector de salida UART para el módulo ESP8266-01, facilitando así su integración con el sistema principal. Este módulo permite establecer una comunicación inalámbrica mediante WiFi, abriendo la posibilidad de enviar y recibir datos desde y hacia el robot de forma remota.

- Se implementó en la PCB el pulsador para que el robot reciba instrucciones.

- Además del circuito de los sensores IR implementado en laboratorios anteriores.

Todos los componentes y circuitos mencionados anteriormente se exportaron hacia la PCB y se situaron sobre una plantilla con una plantilla predeterminada por el profesor, la cual tenía la forma y las medidas de la base que ya se tenía en madera. Se continuo con el ruteamiento de todas las pistas de los componentes para el correcto funcionamiento de la PCB, definiendo pistas con un grosor de 0.5mm y un polígono de tierra por ambas caras de la PCB con una separación de 1.2mm, la PCB se diseñó doble capa para mayor facilidad en el ruteo de esta, esto lo podemos observar en la *Fig.02*.

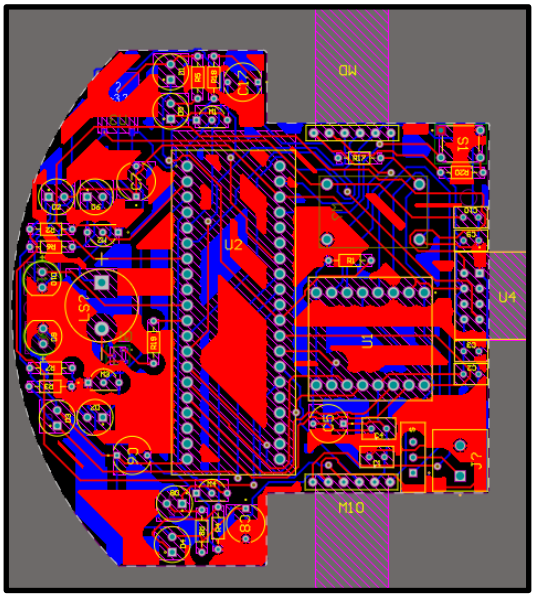

Fig.03. Ruteado de la PCB.

Para la fabricación de la PCB, se imprimieron las pistas tanto de la cara superior, como de la cara inferior, al sacar la impresión se tuvo que aplicar el efecto espejo a la cara superior para la correcta orientación de los componentes sobre la placa. Las impresiones del diseño se realizaron sobre papel couché, el cual permite que el tóner de la impresora láser se adhiera adecuadamente y pueda transferirse de manera uniforme a la placa de cobre mediante calor.

Una vez impresas correctamente las caras del diseño de la placa de circuito impreso sobre papel couche, estas se alinearon y fijaron con cinta enmascarar para asegurar su posición sobre la baquelita virgen. Posteriormente, se aplicó calor con una plancha durante varios minutos, permitiendo que el tóner se transfiriera completamente desde el papel a la superficie de cobre de la baquelita, formando los caminos y pads del circuito. En los casos donde algunas pistas quedaron incompletas o poco definidas, se utilizó un marcador tipo Sharpie para reforzarlas manualmente.

Luego, se preparó una solución de ácido (Cloruro férrico) para sumergir la placa en el ácido y agitar suavemente para favorecer la reacción y eliminar el cobre sobrante. Una vez completada la parte del ácido para la PCB, la placa se retiró del ácido, se enjuagó con abundante agua, y finalmente se limpió y se pulió con una esponja de brillo para retirar los restos de tóner, revelando las pistas de cobre que conformaban la PCB lista para taladrar y soldar los componentes.

Una vez finalizado el proceso de la impresión de la PCB, se procedió a realizar pruebas de continuidad para confirmar que no existieran cortos ni pistas abiertas entre las conexiones del circuito. También se verificó visualmente el ruteo de las pistas, asegurando que los componentes estuvieran correctamente orientados y que no existieran errores en el posicionamiento. Posteriormente, con ayuda del motortool, se realizaron los huecos necesarios para los pines de conexión y los terminales de los componentes, además de los huecos que se encargan de comunicar los caminos de la capa de encima con la de abajo.

Una vez perforada la placa, se procedió a soldar cada uno de los componentes, comenzando por los huecos que comunicaban las pistas de la cara superior con la inferior, luego se procedieron a soldar los componentes de menor tamaño para facilitar el proceso de soldadura de la PCB.

Finalizado el proceso de soldadura de todos los componentes en la PCB, se realizó nuevamente una verificación completa de continuidad, siguiendo cuidadosamente todos los caminos de las pistas para asegurar que no existieran cortocircuitos ni conexiones abiertas. Este chequeo final fue esencial antes de alimentar el sistema.

Posteriormente, se instalaron las baterías y se procedió a montar la PCB en la base de madera y a utilizar los tornillos para su correcto posicionamiento junto a los motores, asegurando una integración mecánica estable. Con el sistema ya ensamblado y alimentado, se realizaron las pruebas iniciales para verificar el correcto funcionamiento del hardware y su comunicación con la interfaz QT.

Etapa de desarrollo del laboratorio #8:

Una vez completado el montaje físico y validado el funcionamiento de la PCB, se procedió con la segunda etapa del laboratorio, el cual consistió en el desarrollo de un sistema de control de distancia que calcula la velocidad deseada en función de la distancia recorrida en tiempo real. Este control está determinado por la distancia desde el inicio del movimiento.

Posteriormente, el robot es desconectado del PC, colocado en el suelo, y queda a la espera de que se presione un pulsador conectado al pin PB10. Esta acción es monitoreada constantemente en el bucle principal de main.c.

El movimiento se activó mediante un pulsador físico (conectado a PB10), que al ser presionado, activó una espera de 5 segundos antes de iniciar el desplazamiento. Este tiempo de retardo es controlado por el Timer 2 (TIM2), que actúa como temporizador general con una resolución de medio milisegundo (frecuencia ajustada mediante el prescaler 4800-1). Durante este tiempo el robot permanece inactivo, asegurando que el usuario tenga tiempo de alejarse del entorno de prueba. Al cumplirse los 5 segundos, el robot comienza el desplazamiento.

La implementación utilizada en el código ejecuta un control directo por pulsos. Se calcula cuántos pulsos del encoder representan la distancia deseada, teniendo en cuenta el diámetro de la rueda ($38.2/2$) y el número de pulsos por revolución ($1431\ pulsos/rev$). Esta estimación convierte la distancia objetivo a un valor entero de pulsos para cada rueda:

$$revoluciones\ =\ distancia/(\pi * (38.2/2))$$

$$pulsos\ objetivo\ =\ revoluciones\ *\ (1431\ pulsos/rev)$$

Durante el movimiento, se utiliza el Timer 3 (TIM3) y el Timer 5 (TIM5) configurados como encoders.

La función actualizar_pose(), ejecutada cada 10 ms mediante TIM4, calcula la posición y orientación del robot en base a los pulsos obtenidos de los encoders conectados a TIM3 y TIM5 para contar los pulsos generados por los motores izquierdo y derecho, respectivamente. La señal de velocidad se mantiene constante hasta que ambos encoders alcanzan el conteo objetivo, momento en el cual los motores se detienen.

Posteriormente se implementó del sistema de odometría, basado en la lectura de los encoders de los motores. Para garantizar un control preciso del movimiento y actualizar la posición del robot.

La odometría se encarga de calcular en tiempo real:

- La posición (x, y)
- La orientación θ
- La velocidad lineal y angular del robot.

Se configuro el sistema para que la lectura de pulsos de los encoders se realice cada 10 ms, mediante una interrupción generada por el Timer 4 (TIM4), con el cual, se mide el número de pulsos de los encoders conectados a los motores izquierdo y derecho.

La distancia recorrida por cada rueda se calcula de la siguiente manera:

$$d_L = \frac{2\pi R \cdot Pulsos_L}{N}$$

Ecu.01. Ecuación para la distancia recorrida por la rueda Izquierda.

$$d_R = \frac{2\pi R \cdot Pulsos_R}{N}$$

Ecu.02. Ecuación para la distancia recorrida por la rueda derecha.

Para lo cual tenemos definido lo siguiente:

- R es el radio de la rueda → 19.1 mm
- N es el número de pulsos por revolución del encoder → 1431

Posteriormente se calculó el desplazamiento lineal del robot, con el promedio de las distancias recorridas por ambas ruedas:

$$\Delta d = \frac{d_L + d_R}{2}$$

Ecu.03. Ecuación para el desplazamiento lineal del robot.

Con el cálculo obtenido de la Ecu.02, se procedió a calcular la velocidad lineal instantánea, de la siguiente manera:

$$v = \frac{\Delta d}{\Delta t}$$

Ecu.04. Ecuación para la velocidad lineal instantánea del robot.

Teniendo en cuenta que la diferencia de tiempo es el intervalo de actualización que se configuro a 10ms.Las variables anteriormente calculadas son almacenadas en la estructura Pose y permiten conocer la posición y velocidad en tiempo real.

Posteriormente se implementó; el control de movimiento desarrollado para el robot el cual se basa en la generación de un perfil de velocidad trapezoidal, el cual permite alcanzar una distancia objetivo de manera controlada, utilizando fases definidas de aceleración, velocidad constante y desaceleración. Este perfil garantiza un movimiento suave y evita cambios bruscos de velocidad que podrían afectar la estabilidad del sistema.

El sistema inicia su funcionamiento al recibir los parámetros de movimiento desde una interfaz Qt vía USB, los cuales incluyen: la distancia a recorrer (mm), la velocidad máxima (mm/s) y la aceleración deseada (mm/s²). Estos datos se almacenan en variables globales, con los valores recibidos se procede a calcular las fases de movimiento:

1. Primero, se calcula el tiempo necesario para alcanzar la velocidad máxima:

$$t_{Aceleracion} = \frac{V_{max}}{a}$$

Ecu.05. Ecuación para el tiempo de aceleración del robot.

2. se determina la distancia necesaria para realizar la aceleración:

$$d_{Aceleracion} = \frac{1}{2} a {t_{Aceleracion}}^2$$

Ecu.05. Ecuación para la distancia de aceleración del robot.

Teniendo en cuenta que la desaceleración es simétrica a la aceleración, la distancia de desaceleración es igual a la distancia de aceleración. Por lo cual el robot necesita:

$$2 \cdot d_{Aceleracion}$$

Ecu.06. Ecuación para la distancia de aceleración y desaceleración del robot.

3. Se compara si la distancia total (D) que es recibida desde Qt, permite mantener una fase de velocidad constante, de tal manera que sea posible acelerar y desacelerar con la distancia que se pretende recorrer. Se establecieron los análisis de los casos posibles:

    - La distancia es suficiente:

    $$D > 2 \cdot d_{Aceleracion}$$

    Lo que significa que hay suficiente espacio para:

Fig.04. Proceso del perfil de velocidad trapezoidal.

Lo que nos indica que la distancia que sobra entre acelerar y frenar es la

distancia de velocidad constante:

$$d_{Constante} = D - 2 \cdot d_{Aceleracion}$$

Ecu.07. Ecuación para la distancia constante del perfil trapezoidal del robot.

- La distancia no es suficiente:

$$D \leq 2 \cdot d_{Aceleracion}$$

Este caso nos indica que el robot no alcanza a llegar a la velocidad máxima antes de tener que empezar a frenar. En este caso se implementa un perfil triangular en el cual el robot se comporta de la siguiente manera:

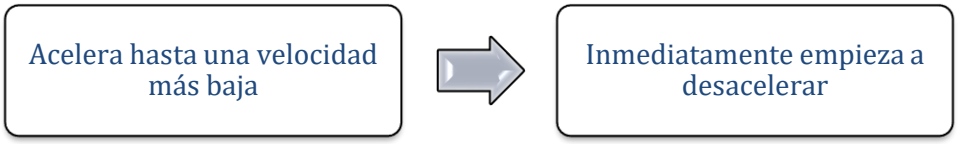

Fig.05. Proceso del perfil de velocidad triangular.

Para corregir la velocidad máxima permitida se calcula de la siguiente manera:

$$V_{Ajustada} = \sqrt{2a \cdot (\frac{D}{2})}$$

Ecu.07. Ecuación para la velocidad ajustada del perfil trapezoidal del robot.

Durante la ejecución de movimiento del robot, ocurre la evaluación constante de la distancia real recorrida para ajustar la velocidad dependiendo de la fase del perfil trapezoidal en que se encuentre el robot en cada momento de su recorrido. El perfil de velocidad trapezoidal cuenta con tres fases:

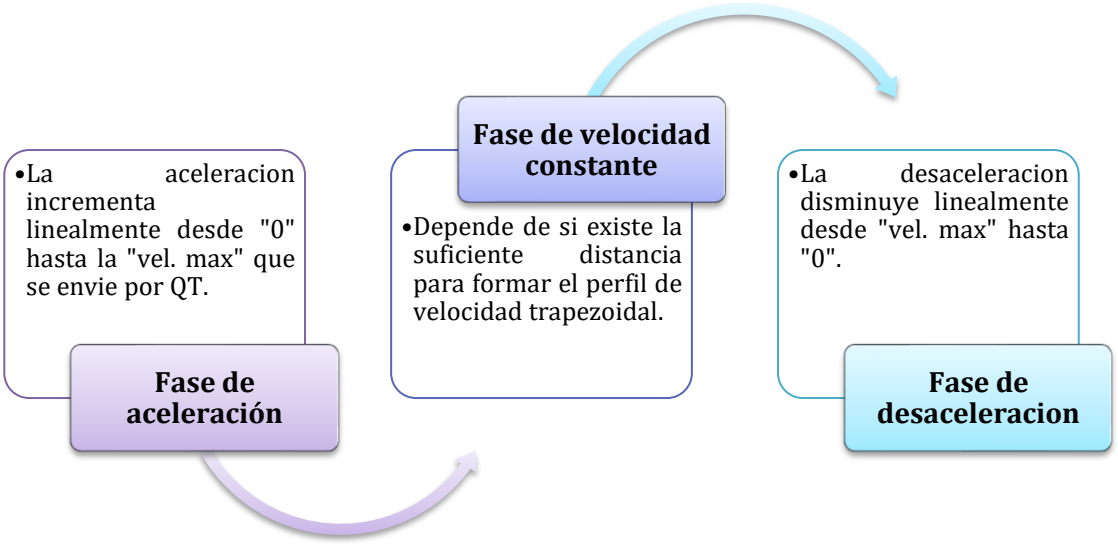

Fig.06. Proceso y definición de las fases del perfil de velocidad trapezoidal.

- Fase de aceleración; ocurre cuando se cumple la condición de que la distancia

actual del robot es menor a la distancia de aceleración ($d_{Actual} < d_{Aceleracion}$), y se calcula la $d_{Relativa}$, que es a la distancia actual que lleva el robot desde que inicio, como se observa en la Ecu.08.

$$d_{Relativa} = d_{Actual} - d_{Inicio}$$

Ecu.08. Ecuación para distancia relativa de la fase de aceleración del perfil trapezoidal del robot.

$$V_{Deseada} = \sqrt{2ad_{Relativa}}$$

Ecu.09. Ecuación para la velocidad deseada de la fase de aceleración del perfil trapezoidal del robot.

- Fase de velocidad constante; ocurre cuando ya se han comparado las distancias, y se determina que la distancia total permite mantener una fase de velocidad constante con la siguiente condición:

$$(d_{Actual} \leq d_{Relativa} < d_{Aceleracion} + d_{constante})$$

En este caso:

$$V_{deseada} = V_{maxima}$$

Ecu.10. Ecuación para la velocidad deseada de la fase de velocidad constante del perfil trapezoidal del robot.

- Fase de desaceleración; ocurre cuando se determina que el robot ya va llegando a su objetivo, y necesita comenzar a reducir su velocidad linealmente por medio de la siguiente condición:

$$(d_{Relativa} \geq d_{Aceleracion} + d_{constante})$$

En este caso se compara la distancia (D) recibida desde Qt menos la distancia que tenia inicialmente, para verificar que ya se llego a la distancia objetivo.

$$D_{relativa} = D - d_{Inicial}$$

Ecu.11. Ecuación para la distancia relativa de la fase de desaceleración del perfil trapezoidal del robot.

$$V_{Deseada} = \sqrt{2a(D_{relativa} - d_{Relativa})}$$

Ecu.12. Ecuación para la velocidad deseada de la fase de desaceleración del perfil trapezoidal del robot.

De esta manera Cuando se alcanza la distancia total D, el sistema ordena detener el movimiento:

$$V_{Deseada} = 0$$

Los cálculos implementados en la función control_distancia() permitieron determinar dinámicamente si el perfil debía ser trapezoidal o triangular, según si la distancia total era suficiente para alcanzar la velocidad máxima. Los PWM se ajustaron conforme a la velocidad deseada en cada instante, utilizando una conversión basada en calibración previa.

Adicionalmente se implementó un control feedforward el cual actúa anticipando el valor del PWM necesario para lograr la velocidad deseada sin esperar a que ocurran errores significativos. El funcionamiento se observa a continuación:

Fig.07. Proceso del control feedforward.

En este caso, el control Feed-Forward se implementa al calcular el valor de PWM necesario para alcanzar una velocidad específica antes de que ocurra el movimiento. Esta conversión de unidades de mm/s a PWM se realiza con la función conversor_mm_s(), la cual se basa en una relación empírica entre velocidad lineal y el ciclo de trabajo del motor. El valor resultante se utiliza como punto de partida para el PWM aplicado, al que posteriormente se le suma una corrección proporcional para compensar cualquier desviación respecto a la velocidad real medida. Esta estrategia logra una respuesta más estable y rápida ante cambios en el cambio de velocidad.

Se utilizo la siguiente formula:

$$PWM = k \cdot V_{deseada} + b$$

Y el método de implementación consiste en un control proporcional de velocidad en el cual el sistema mide la velocidad de cada motor utilizando la odometría y compara con la velocidad deseada. Se aplica una corrección proporcional sobre el PWM para disminuir el error:

$$error = vel\_deseada - vel\_actual$$

$$pwm \mathrel{+}= KP\_VELOCIDAD * error$$

Este método mejora la respuesta del sistema y permite mantener un avance lineal estable y centrado. Adicionalmente, se establece una velocidad mínima inicial $V_{min} = 100\,mm/s$ para evitar bloqueos al inicio del movimiento, cuando la distancia inicial es muy pequeña.

Finalmente el robot se considera que ha alcanzado su destino cuando la distancia recorrida está dentro de un margen de tolerancia de ±3 mm respecto a la distancia objetivo. Al cumplirse esta condición, el PWM se reduce a cero y se detiene el movimiento de los motores. En la Fig.02 se puede ver el funcionamiento general del robot.

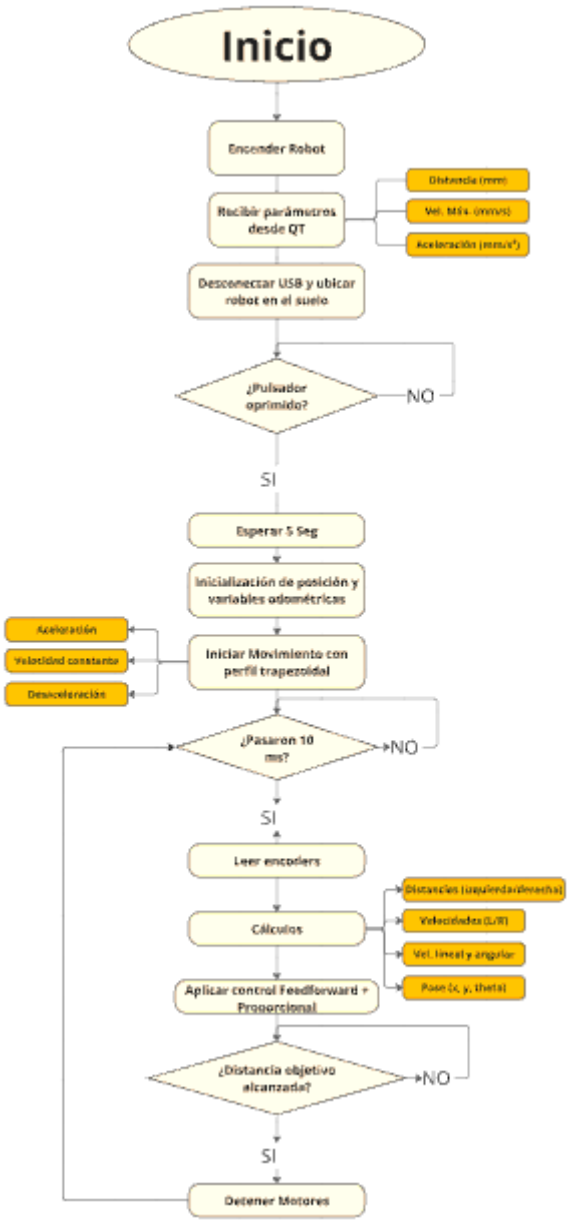

Fig.02. Diagrama general del funcionamiento del robot.

Pruebas y Validación:

- Para validar el sistema, se realizaron pruebas mandando las magnitudes correspondientes para poder evidenciar el perfil trapezoidal y el perfil triangular de la velocidad como se muestran a continuación, gracias a la visualización que se obtuvo por medio del software Visual studio code, en el cual se descargó la extensión teleplot. Y se configuro para obtener la visualización de los datos en tiempo real, de cada prueba.

Prueba	Distancia por recorrer (mm)	Vel. Máxima (mm/s)	Aceleración (mm/s^2)	Grafica (velocidad vs tiempo)
1	50	150	200	
2	500	150	200	
3	100	200	500	
4	300	100	200	

Tabla.03. Pruebas del sistema.

El error fue menor al 1%, demostrando la precisión del sistema de odometría. También se verificó que el tiempo total de desplazamiento coincidiera con los cálculos teóricos del perfil trapezoidal, con un desvío mínimo (< 0.01 s). Además se verifico con ayuda de un metro que la distancia recorrida por el robot fuera exactamente la que se deseaba.

La interfaz Qt permitió graficar en tiempo real la velocidad, posición y orientación del robot, confirmando la estabilidad del sistema en cada fase del movimiento.

Dificultades presentadas:

Durante el desarrollo del laboratorio, enfrentamos varios retos que complicaron el avance del proyecto, pero logramos resolverlos con análisis y apoyo del profesor, de la siguiente manera:

- **Proceso de impresión y transferencia de la PCB:** Una de las mayores dificultades se presentó al momento de imprimir correctamente las capas del diseño de la PCB. Fue necesario realizar varias pruebas para ajustar la calidad y escala de impresión en el papel couche, asegurando que las pistas y pads se definieran con precisión. La alineación de las capas (superior e inferior) también generó inconvenientes, ya que una pequeña desalineación podía afectar la coincidencia de los pads de componentes de doble cara. Para resolverlo, se utilizaron marcas de referencia y cinta enmascarar para fijar las hojas antes de planchar.
- **Planchado y transferencia del tóner:** Durante el planchado, se presentaron problemas para lograr una transferencia uniforme del tóner a la baquelita virgen. En algunos sectores el calor no se distribuyó correctamente, provocando zonas incompletas en las pistas. Fue necesario repetir el proceso de transferencia varias veces y reforzar manualmente algunos caminos con marcador tipo Sharpie.
- **Proceso de ataque químico con ácido:** El revelado con ácido también presentó dificultades, como la necesidad de mantener una agitación constante para que el ácido actuará de manera homogénea sobre toda la superficie. En algunas ocasiones el proceso fue muy lento o irregular, lo que llevó a repeticiones y ajustes en la proporción de la mezcla ácida. Además, se tuvo que controlar cuidadosamente el tiempo de inmersión para evitar que el ácido atacará las pistas deseadas.
- **Verificación de continuidad tras soldadura:** Durante las pruebas de continuidad, se detectaron varios cortocircuitos generados por residuos de cobre no eliminados correctamente durante el proceso químico. Estos residuos entre pistas provocaban cortos en la placa. Para solucionarlos, fue necesario raspar cuidadosamente el área con un cúter, eliminando el exceso de cobre sin dañar las pistas funcionales. Este paso fue fundamental para garantizar la integridad del circuito antes de la alimentación y conexión de los elementos electrónicos.
- **Librerías y orden en el código:** al inicio, el carrito no medía correctamente la distancia recorrida. Tras revisar el código, se identificó que el problema se originaba por un mal uso de variables compartidas entre archivos, específicamente por el uso incorrecto del modificador “extern” sin el acompañamiento de punteros adecuados. Esto impedía que las variables globales se actualizarán correctamente entre módulos, afectando las lecturas de los encoders y la lógica de control de distancia. La solución consistió en redefinir las variables críticas utilizando punteros y referencias bien estructuradas, garantizando su integridad en todo el sistema.

Diagramas:

- *Diagrama de flujo (Archivo anexo ¨Link diagrama de flujo¨).*

- Diagrama esquemático (*Archivo anexo ¨Esquemático PCB¨*).

Conclusiones:

1. La fabricación de la PCB fue una parte fundamental para el robot, ya que permitió integrar todos los circuitos en una sola placa optimizando el espacio y mejorando la estabilidad eléctrica y mecánica del sistema.
2. A pesar de las dificultades en la impresión, alineación, planchado de la placa, se logró obtener una PCB funcional, lo que fue vital para el desarrollo del robot y el laboratorio #8.
3. El uso de Altium Designer facilitó el diseño de la PCB, asegurando una correcta distribución de pistas, cumplimiento de normas establecidas por el profesor y disposición óptima de componentes, incluyendo la ubicación estratégica del pulsador y LEDs.
4. La verificación de continuidad tras la soldadura fue clave para asegurar la operatividad del circuito antes de conectarlo al resto del sistema, debido a que se pudieron corregir algunas pistas que estaban en corto.
5. La implementación del perfil de velocidad trapezoidal permitió que el robot realizará desplazamientos suaves, evitando aceleraciones bruscas que podrían desestabilizarlo.
6. A través del modelo de odometría y una actualización periódica cada 10 ms, se logró mantener una estimación precisa y continua de la posición del robot, y logro llegar a la distancia con los parámetros establecidos y enviados por el puerto COM.
7. Las fallas iniciales en la medición de distancia subrayaron la necesidad de manejar adecuadamente extern y las referencias cruzadas entre archivos para evitar conflictos y asegurar la coherencia de datos.

8. Gracias a las pruebas realizadas se evidencio el optimo funcionamiento del perfil de velocidad trapezoidal y el perfil triangular, como se puede observar en la Tabla 03.