UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e Implementación de Sistema Electrónico de un Robot Esférico

Trabajo de graduación presentado por Pedro Antonio García Morales
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2020

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e Implementación de Sistema Electrónico de un Robot Esférico

Trabajo de graduación presentado por Pedro Antonio García Morales
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2020

Vo.Bo.:

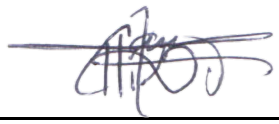
(f) _____

MAEB. Pablo Daniel Mazariegos de la Cerda

Tribunal Examinador:

(f) _____

MAEB. Pablo Daniel Mazariegos de la Cerda



(f) _____

MSc. Carlos Alberto Esquit Hernández

(f) _____

MSc. Jose Eduardo Morales Espinoza

Fecha de aprobación: Guatemala, 17 de junio de 2020.

Prefacio

El presente trabajo de graduación en modalidad de tesis es la culminación de mi preparación académica para el grado de licenciatura. Es por lo que este trabajo reúne gran parte del conocimiento adquirido en estos cinco años y medio de la carrera de Ingeniería Mecatrónica.

El trabajo se desarrolla en la línea de robótica swarm de la Universidad del Valle de Guatemala, la cual tiene como objetivo fomentar la investigación y desarrollo de tecnología innovadora, accesible y de bajo costo.

Índice

Lista de figuras

Lista de cuadros

Resumen

El trabajo consiste en el desarrollo electrónico de un robot esférico con el tamaño adecuado para poder manipularse dentro del Robotarium de la Universidad del Valle de Guatemala.

Para ello el trabajo se dividió en dos módulos principales: la etapa mecánica y la etapa electrónica. Este trabajo trata sobre el desarrollo de la parte electrónica del robot esférico. El sistema electrónico a su vez se divide en módulo de potencia, comunicación WIFI y el algoritmo de control punto-punto.

La etapa de potencia es la encargada de calcular el tiempo de autonomía del robot móvil, al calcular los datos de potencia máxima que consume el circuito y teniendo la capacidad de carga de la batería. Para la comunicación WIFI se implementará una conexión entre una interfaz de control usando una computadora y el módulo WIFI ESP8266. El robot esférico se manipulara dentro de una mesa de pruebas de experimentación, Robotarium, para lo cual se diseñará e implementará un algoritmo de control punto-punto con el que el agente móvil será capaz de tener una locomoción eficiente.

Uno de los propósitos principales es desarrollar esta agente, robot esférico, de modo que sea lo más económicamente posible y de fácil ensamblaje.

Abstract

The work consists in the electronic development of a spherical robot with the appropriate size to be able to manipulate inside the robotat of the Universidad del Valle de Guatemala.

For this, the work was divided into two main modules: the mechanical stage and the electronic stage. This work is about the development of the electronic part of the spherical robot. The electronic system in turn is divided into power module, WIFI communication and the point-point control algorithm.

The power stage is responsible for calculating the autonomy time of the mobile robot, when calculating the maximum power data consumed by the circuit and having the battery's charge capacity. For WIFI communication, a connection between a control interface using a computer and the ESP8266 WIFI module will be implemented. The spherical robot will be manipulated within an experimental test table, robotat, for which a point-to-point control algorithm will be designed and implemented with which the mobile agent will be able to have efficient locomotion.

One of the main purposes is to develop this agent, spherical robot, so that it is as economically possible and easy to assemble.

CAPÍTULO 1

Introducción

El presente trabajo trata sobre la implementación de una nueva plataforma móvil, que forme parte de la rama de robótica de enjambre de la Universidad del Valle de Guatemala. El objetivo de desarrollo de esta plataforma es poder manipular un agente en el Robotarium, logrando que el desarrollo de dicho agente sea de fácil proceso de fabricación, ensamblaje y bajo costo para que en futuras aplicaciones se pueda manipular más de un agente a la vez.

El desarrollo de la nueva plataforma trata de una geometría de un robot esférico y de forma más específica este trabajo se centra en el diseño del sistema electrónico. Este trabajo se llevó a cabo con la idea de proporcionar un diferente tipo de locomoción de un robot móvil, el cual al contar con una geometría no convencional presentaría variantes tanto en su diseño físico como en la parte de implementación electrónica.

Se definieron tres módulos principales para este trabajo. El primero que tenga la capacidad brindar la potencia necesaria a los componentes y sobre todo a los motores encargados del movimiento del robot móvil, además de contar con una autonomía lo más grande posible para poder implementarse en varias rutinas. Una etapa de comunicación inalámbrica, WIFI para esta aplicación, entre una interfaz de control y el robot móvil.

El diseño de una tarjeta de control, PCB, de dimensiones compactas que permitan el menor tamaño de implementación de la parte esférica y que agrupe los módulos de potencia y comunicación WIFI. Finalmente un algoritmo de control punto punto para realizar pruebas de buena locomoción de la plataforma móvil.

CAPÍTULO 2

Antecedentes

2.1. Reingeniería de megaproyectos fase I

En 2017 se presentó la tesis de licenciatura *Reingeniería de Megaproyectos Fase I: Módulo de Swarm Robotics* [1],desarrollado por Erick Hernández, Johnny Guzmán, José Mérida, Otto Wantland, Vidal Villegas, William Orozco y Rony Ajtún de la Universidad del Valle de Guatemala

Proyecto que contaba con las siguientes caraterísticas:

- Comunicación WIFI meidante módulo ESP8266
- Protocolo de comunicación TCP/IP
- Implementación de baterías de 2500 mAh.
- Uso de 6 sensores ultrasónicos para sensado.

2.2. The Robotarium: a remotely accessible swarm robotics research testbed

En 2017 se publicó un artículo en el sitio web *Robohub: The Robotarium: A remotely accessible swarm robotics research testbed* [2], el cual trataba la idea de la construcción de un Robotarium para robots móviles, así como de diseños de robots a implementar para realizar experimentos. Las caracterísitcas principales son:

- Minimizar costo y complejidad de una gran colección de robots.
- Interacción intuitiva entre Robotarium y una interfaz web para envío y recepción de datos.
- Medidas de seguridad para proteger el Robotarium de daños y prevención de colisiones.
- Diseño de robots compactos para manipulación efectiva.

2.3. Diseñar e implementar una red de comunicación inalámbrica para la experimentación en robótica de enjambre

En 2018 se presentó la tésis de licenciatura de *Diseñar e implementar una red de comunicación inalámbrica para la experimentación en robótica de enjambre* [3] trabajo desarrollado por Marlon Castillo en la Universidad del Valle de Guatemala.

El anterior trabajo muestra la selección e implementación de una red WIFI como tipo de comunicación entre un robot móvil y una plataforma de pruebas, esta red de comunicación tiene las siguientes características:

- Protocolo de comunicación TCP/IP.
- Comunicación inalámbrica mediate el módulo WIFI ESP8266.
- Programación de Sockets en C++.

En el trabajo se plantean direntes formas de comunicación inalámbrica (Bluetooth, WIFI, LTE), eligiendo la comunicación WIFI como red inalámbrica a implementar, además se seleccionó el modulo ESP8266 para la comunicación WIFI. Módulo al cual se le hicieron modificaciones como la actualización del Firmware con el objetivo de mejorar las características de funcionamiento y transmisión de datos. Además de la implementación de Sockets TCP para implementar una comunicación estándar entre los robots móviles y la plataformas de pruebas de la Unviersiad del Valle de Gautemala.

2.4. Diseño e implementación de una nueva plataforma móvil para aplicaciones en robótica de enjambre

En 2018 se presentó la tesis de licenciatura de *Diseño e implementación de una nueva plataforma móvil para aplicaciones en robótica de enjambre* [4] trabajo desarrollado por Javier Lima Cordón en la Universidad del Valle de Guatemala.

El trabajo consistió en un diseño electrónico y mecánico más compacto respecto a versiones anteriores[1], Cosnta de comunicación WiFi con un ordenador del que recibe instrucciones y un agente móvil que las ejecuta. El trabajo cuenta con las siguientes características:

- Detección de objetos alrededor de la plataforma móvil,
- Movilización en superficie,
- Monitoreo del desplazamiento, velocidad y dirección.
- Batería recargable con autonomía aproximada de 2 horas.
- Comunicación inalámbrica mediate el módulo WIFI ESP8266.d
- MATLAB como interfaz de control de robot.

CAPÍTULO 3

Justificación

El campo de la robótica es uno de los que presentan más interés en la actualidad por la versatilidad de áreas en las que se puede implementar. El caso de los robots móviles tiene por lo tanto diversas áreas de aplicación, desde automatización de procesos industriales hasta ambientes académicos.

En Guatemala el desarrollo de este tipo de tecnologías es prácticamente nulo por lo que el desarrollo de plataformas robóticas de bajo costo presenta un gran impacto en la actualidad, plataformas que comiencen como áreas de investigación y luego puedan seguir desarrollándose hasta llegar a tener una implementación en el mundo comercial o industrial [5].

Este trabajo se basa en implementar una red WIFI de comunicación, además de un algoritmo de control punto a punto para la ayuda que pueden generar los robots esféricos como elemento de investigación dentro de una plataforma de pruebas, ya que pueden ser modelados como átomos, moléculas o partículas de interés, con los que se puede replicar algún movimiento de la naturaleza o alguna simulación deseada [6].

El trabajo busca implementar un tipo de robot que sea económico y por eso la implementación del módulo WIFI ESP8266 es esencial ya que en la actualidad es usado por su bajo costo y fácil implementación en accesorios de IoT (Internet of things).

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar e implementar el sistema electrónico y de control para la locomoción de un robot esférico.

4.2. Objetivos específicos

- Diseñar el sistema de potencia para la alimentación de los motores y la placa de control del robot esférico.
- Establecer una conexión entre la interfaz de control y el robot móvil mediante el módulo WiFi ESP8266.
- Diseñar una PCB para el control del sistema electrónico.
- Diseñar un algoritmo de control punto a punto para la locomoción del robot esférico.

CAPÍTULO 5

Alcance

Los proyectos de la línea *swarm* incluyen diferentes tipos de locomoción que pueden tener robots móviles por ejemplo diferencial, esférico, omnidireccional los cuales pueden tener diferente cantidad de patas o llantas. En el departamento de Ingeniería Electrónica-Mecatrónica se dispone de una plataforma de pruebas *Robotarium* en dónde se realizarán las pruebas correspondientes. La plataforma tiene medidas de $93.5\,x\,133.5$ cm, por lo que el tamaño del robot esférico debe ser lo más compacto posible.

El presente trabajo es una continuación de la línea de robótica de enjambre *swarm* que se desarrolla en la Universidad del Valle de Guatemala [3], [4]. El proyecto completo trata del desarrollo de un robot esférico, el cual se compone de diseño mecánico y diseño electrónico. El desarrollo del diseño mecánico incluye: el diseño de la base para los motores, placa electrónica y esfera del robot y la parte de visión de computadora. La parte de diseño mecánico se desarrollará en el trabajo de graduación de Antonio Ixtecoc. El diseño electrónico incluye: módulo de potencia, módulo de conexión WiFi y algoritmo de control punto punto se desarrollará en este trabajo.

Al finalizar el presente trabajo el módulo electrónico deberá tener un sistema de potencia capaz de otorgar la suficiente autonomía para el desempeño óptimo de los motores DC y la placa de control, PCB. Deberá tener una conexión WIFI estable entre la interfaz de control, Python, y la placa de control, esta conexión deberá permitir el uso de diferentes operaciones como elegir el punto de destino del robot esférico o controlar la velocidad del mismo. Además deberá tener un algoritmo de control punto punto funcional, capaz de mover el robot móvil entre el punto inicial y el punto de destino.

El proyecto completo, la integración de la parte mecánica, desarrollada en el trabajo de graduación de Antonio Ixtecoc, y la parte electrónica, desarrollada en este trabajo, deberá implementar el algoritmo de control punto punto por medio de la visión de computadora realizada también dentro del trabajo del diseño mecánico. Todo esto dentro del *Robotarium* de la Universidad del Valle de Guatemala.

CAPÍTULO 6

Marco teórico

6.1. Robot móvil

La robótica móvil se ha desarrollado en conjunto con los avances tecnológicos recientes, lo cual ha generado el desarrollo de muchas aplicaciones en la sociedad.
Los robots hoy en día son en su mayoría excesivamente costosos y su función está muy especializada, por lo que los estudios actuales se enfocan en conseguir que sean de aplicaciones generales y más económicas [7].

Los robots se clasifican de la siguiente manera en función de su aplicación y características:

- Robot móvil.
- Robot industrial.
- Robot humanoide.
- Robot inteligente.
- Robot de servicios.

6.1.1. Robot móvil con ruedas

Los vehículos con ruedas son la solución más simple para conseguir la movilidad en terrenos suficientemente duros y libres de obstáculos, permitiendo conseguir velocidades relativamente altas.

Los robots móviles emplean diferentes tipos de locomoción mediante ruedas que les confieren características y propiedades diferentes respecto a la eficiencia energética, dimensiones, cargas útiles y maniobrabilidad. La mayor maniobrabilidad se consigue en vehículos omnidireccionales. Un vehículo omnidireccional en el plano es capaz de trasladarse simultánea e independientemente en cada eje del sistema de coordenadas. y rotar según el eje perpendicular [8].

Adicionalmente, existen una o más ruedas para soporte. Esta configuración es la más frecuente en robots para interiores de pequeño tamaño.

Tracción diferencial

Este tipo de direccionamiento viene dado por la diferencia de velocidades de las ruedas laterales. La tracción se consigue también con estas mismas ruedas. Dos ruedas montadas en un único eje son independientemente propulsadas y controladas proporcionando ambas tracción y direccionamiento. La combinación del movimiento de las dos ruedas provoca el movimiento alrededor del centro de masa. Este sistema es muy útil si consideramos la habilidad del movimiento del móvil. presentando la posibilidad de cambiar su orientación sin movimientos de traslación, lo que podríamos llamar cambio de spin. Las variables de control de este sistema son las velocidades angulares de las ruedas izquierda y derecha, Los diferentes modelos cinemáticos existentes proporcionan trayectorias perfectamente definidas, y con ello obtenemos el posicionamiento deseado [8].

Figura 1: Base de la tracción de un robot diferencial.

6.1.2. Posición y orientación de un robot móvil en el plano.

Para poder describir de forma conveniente las posiciones y orientaciones de los robots móviles en el espacio, utilizaremos los movimientos de rotación y traslación de forma tal que podremos conocer la posición y la orientación del robot en el plano realizando operaciones elementales. Para conocer el movimiento del móvil es necesario conocer la posición y la orientación inicial y final, este camino nos introduce en el estudio de la cinemática de móvil [8].

Partiendo de un marco de referencia inercial A que representa las coordenadas universales del sistema, la expresión de las coordenadas de un punto situado en P respecto al eje de coordenadas A se puede expresar mediante un vector de posición de la siguiente manera.

$$\mathbf{P_T} = \begin{bmatrix} P_x \\ P_y \end{bmatrix} \tag{1}$$

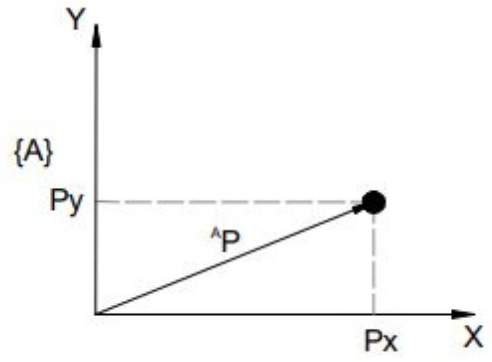


Figura 2: Posición coordenadas cartesianas

Esta forma de expresar las coordenadas se denominan coordenada cartesianas. También podemos utilizar coordenadas polares, de forma que el punto P se puede expresar como la distancia al origen de coordenadas r y el ángulo θ que forma el vector r con el eje x [8]

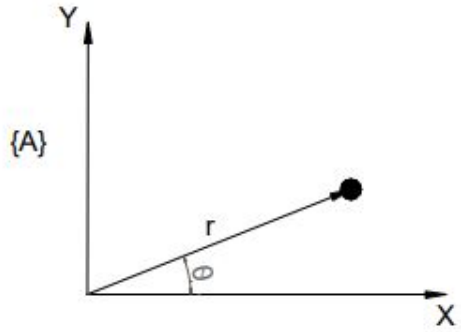


Figura 3: Posición y orientación coordenadas polares

6.2. WIFI

Wi-Fi (Wireless Fidelity) por sus siglas en inglés. Es un tipo de comunicación inalámbrica estandar entre dispositivos. Es decir es un tipo de tecnología que permite que una gran variedad de equipos informáticos (computadoras, impresoras, cámaras, etc.) puedan comunicarse sin la necesidad de utilizar cables [9].

6.2.1. Componentes de una red WIFI

Una red WIFI puede estar compuesta por dos computadoras o por miles de ellas. Para poder comunicarse de forma inalámbrica necesita disponer de un dispositivo que se conoce como *Adaptador de Red*, De forma general a los equipos que forman parte de una red se les conoce como *Terminales* [9].

A parte de los adaptadores de red las redes WIFI pueden disponer de unos equipos conocidos como *Access Point (AP)*, que es una estación base utilizada para gestionar las comunicaciones entre los distintos terminales de la red. Los AP funcionan de forma autónoma, sin necesidad de estar conectados directamente a ningún ordenador [9].
Tanto los terminales como los AP se les conoce por el nombre general de *Estación*. Las estaciones se comunican entre sí ya que usan la misma banda de frecuencia y siguen el mismo conjunto de protocolos [9].

El conjunto de protocolos instalados en las estaciones se compone de dos grupos: uno se ocupa de garantizar la comunicación inalámbrica entre las estaciones (protocolos WIFI), mientras que el otro se ocupa del intercambio de información entre los terminales (protocolos TCP/IP) [9].

6.2.2. Estándar WIFI

El estándar que sigue WIFI para su implementación es el de la *IEEE (Institute of Electrical and Electronics Engineers)*. El estándar WIFI está recogido en la norma IEE 802.11b. Esta norma describe los detalles técnicos para establecer comunicaciones de datos de forma inalámbrica a auna velocidad máxima de $11\,Mbps$. Sin embargo la tecnología de comunicaciones inalámbricas ha seguido desarrollándose, lo que ha permitido transmitir datos a velocidades superiores a $11\,Mbps$ [9].

- **IEEE 802.11b**. Es la norma original que perminte transmisiones de $11\,Mbps$ utilizando la banda de frecuencias de $2.4\,GHz$.
- **IEEE 802.11a**. Esta norma no usa la banda de los $2.4\,GHz$ sino la de $5\,GHz$ con lo que conisgue velociades de transmision de $54\,Mbps$.
- **IEEE 802.11g**. Esta norma surgió de la necesidad de aumentar la velocidad de transmisión sin renunciar a la banda de $2.4\,GHz$. Permite alcanzar velociades de transmisión de $54\,Mbps$
- **IEEE 802.11n**. Con esta norma se consiguen velociades de transmisión de $300\,Mbps$, además resulta ser compatible con todos los estándares anteriores (a, b y g). La característica externa más destacable es que incorpora varias antenas para poder utilizar varios canales simultaneamente.

Sin embargo el modo más común de esta tecnología es el de infraestructura, en el que una estación base funciona como un punto central de conexión con una unión física a la red a

través de un cable de fibra óptica o similar. La estación base se encarga de asignar direcciones IP identificadoras a cada equipo conectado a la red y ayuda a entablar la conexión entre los equipos con la red. [4]

6.2.3. Ventajas de WIFI

- Facilidad de comunicar dispositivos portátiles.
- Facilidad de configuración y reorganización.
- Facilidad de establecer comunicación punto a punto vía radio.

6.3. Modos de operación de redes inalámbricas

6.3.1. Modo ad-hoc (IBSS)

El término se puede traducir como *para esto*, pero se usa comunmente para describir eventos o situaciones improvisadas y a menudo espontáneas [10].
Es un método para que los clientes inalámbricos puedan establecer una comunicación directa entre sí. Al permitir que los clientes inalámbricos operen en modo ad-hoc no es necesario establecer un punto de acceso central. Todos los nodos de una red ad-hoc se pueden comunicar directamente con otros clientes [10].

6.3.2. Modo infraestructura (BSS)

Contrario al modo ad-hoc el modo infraestructura presenta un punto de acceso central. Para conectar varios puntos de acceso y clientes inalámbricos todos deben configurarse con el mismo SSID. Para asegurar que se maximice la capacidad de la red no se debe configurar el mismo canal en todos los puntos de acceso que se encuentran en la misma área física.
En redes IEEE 802.11 el modo infraestructura es conocido como *Basic Service Set (BSS)*, que también se conoce como maestro-cliente [10].

6.4. Protocolo TCP

TCP (Transmission-Control-Protocol o Protocolo de control de transmisión) es uno de los protocolos más fundamentales de internet. Muchos programas dentro de una red de datos compuesta por computadoras pueden usar TCP para crear conexiones entre ellos a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados a su destino sin errores y en el mismo orden en que se transmitieron [11].

TCP da soporte a muchas de las aplicaciones más populares de internet incluidas:

- HTTP (Hypertext Transfer Protocol)
- SMTP (Simple Mail Transfer Protocol)
- SSH (Secure SHell)
- FTP (File Transfer Protocol)

TCP es un protocolo transporte orientado por conexión que envía datos como un flujo de bytes sin estructura. Usando los números de secuencia y los mensajes de reconocimiento.

El protocolo TCP puede retransmitir los datos hasta que o se alcance una condición del descanso o hasta que se haya alcanzado la entrega exitosa.
TCP también puede reconocer mensajes duplicados y los descartará adecuadamente.

Si el ordenador de envío está transmitiendo demasiado rápido para la computadora de recepción, el TCP puede emplear los mecanismos de control de flujo para reducir la Transferencia de datos. TCP también comunica la información de entrega a los protocolos de la capa superiores y a las aplicaciones que soporta. [12]

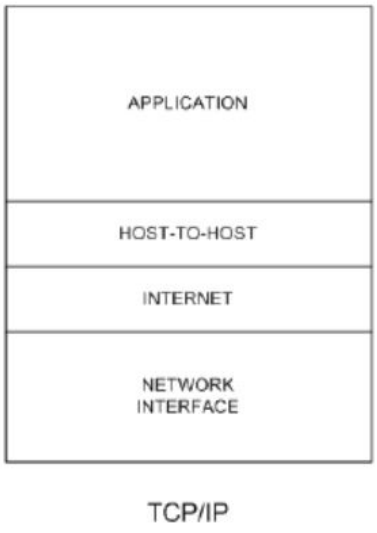


Figura 4: Capas del protocolo TCP

6.4.1. Capa Host-Red

Engloba las funciones de la capa física y la capa de enlace del modelo OSI. Es capaz de conectar el host a la red por medio de algún protocolo que permita envier paquetes IP. Para el modelo TCP/IP se comporta como una *caja negra* [11].

6.4.2. Capa Internet-IP

Su papel es equivalente al desempeñado por la capa de red en el modelo OSI. Se ocupa de encaminar los paquetes de la forma más conveniente para que lleguen a su destino y de evitar que se produzcan situaciones de congestión en los nodos intermedios [11].

La capa de internet da únicamente un servicio de conmutación de paquetes no orientado a conexión. Los paquetes pueden llegar desordenados a su destino, en cuyo caso es responsabilidad de las capas superiores en el nodo receptor la reordenación para que sean presentados al usuario de forma adecuada [11].

La capa de internet define aquí un formato de paquete y un protocolo llamado IP (Internet Protocol), que se considera protocolo *oficial* de la arquitectura, siendo el más popular de todos [11].

6.4.3. Capa de transporte

Consiste en permitir la comunicación *host* a *host* en la red. En esta capa los paquetes son llamados *segmentos* [11].
Aquí se definen dos protocolos:

- **TCP** (Transmission Control Protocol): Ofrece un servicio fiable para que los paquetes lleguen ordenados y sin errores. TCP se ocupa también del control de flujo *host* a *host* para evitar que por ejemplo un *host* rápido sature a un receptor más lento.
- **UDP** (User Datagram Protocol): Da un servicio no orientado a conexión y no confiable. No realiza control de errores ni de flujo, por ejemplo en transmisiones de voz y vídeo en tiempo real.

6.4.4. Capa de aplicación

Esta capa desarrolla las funciones de la capa de sesión del modelo OSI. La experiencia ha demostrado que las capas de sesión y presentación son de poca utilidad, debido a su escaso contenido [11].

La capa de aplicación contiene todos los protocolos de alto nivel que se utilizan para ofrecer servicios a los usuarios [11]. Entre estos podemos mencionar tanto los *tradicionales* que existen desde que se creó el TCP/IP:

- Terminal Virtual (TelNet)
- Transferencia de ficheros (FTP)
- Correo Electrónico (SMTP)

- Servidor de nombres (DNS)
- Servicio de news (NNTP)
- Web (HTTP), Gopher, etc.

6.5. Socket de programación

Un socket (enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets (API, application programming interface) [13].

Un socket es también una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un ordenador particular en Internet) y un número de puerto (el número que identifica una aplicación de Internet particular, como FTP, Gopher, o WWW) [13].

6.5.1. Terminología

- Un socket es un tipo especial de manejador de fichero que utiliza un proceso para pedir servicios de red al sistema operativo.
- Una dirección de socket es la tripleta: protocolo, dirección-local, proceso-local

Por ejemplo, en la familia TCP/IP: *tcp, 193.44.234.3, 12345*

- Una conversación es el enlace de comunicación entre dos procesos.
- Una asociación es la quíntupla que especifica completamente los dos procesos que comprende una conexión.

Protocolo, dirección-local, proceso-local, dirección-externa, proceso-externo

Por ejemplo, en la familia TCP/IP, : *tcp, 193.44.234.3, 1500, 193.44.234.5, 21*

Una media-asociación es: Protocolo, dirección-local, proceso-local o protocolo, dirección-externa, proceso-externo, que especifica cada mitad de una conexión.

La media-asociación se denomina también socket o dirección de transporte. Esto es, un socket es un punto terminal para comunicación que puede nombrarse y direccionarse en una red [13].

6.6. Comunicación serial USART

La comunicación USART *Univsersal Synchronous Asynchronous Reciver/Transmitter*, por sus siglas en inglés, puede ser configurado para establecer una comunicación asíncrona bidireccional simultánea, (*full duplex*), o síncrona, con transmisión de señal de reloj, bidireccional no simultánea (*half duplex*).

6.6.1. UART

Se usa para transmisión de datos asíncronos entre un equipo que funciona como terminal de datos y otro como equipo de comunicación. No hay sincronización de la información transferida entre los dos equipos [14]. Las principales funciones de UART son:

- Conversión de datos, de serie a paralelo y de paralelo a serie.
- Detectar errores con bits de paridad.
- Insertar bits de arranque y paro (start bit, stop bit).

Antes de transferir datos se debe configurarse la naturaleza de los datos, cantidad de bits de los datos, paridad y la cantidad de bits de la información [14].

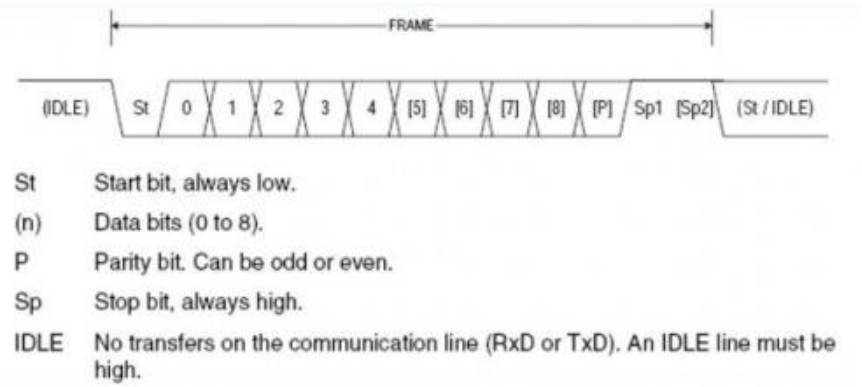


Figura 5: Parámetros comunicación UART

6.7. Módulo WIFI ESP8266

ESP8266 es un puente de puerto serie a WiFi, incluye un microcontrolador para manejar el protocolo TCP/IP y el software necesario para la conexión 802.11, la mayoría de modelos dispone de entradas/salidas (I/O) digitales y algunos modelos una entrada analógica al igual que otros microcontroladores, su punto fuerte es disponer de acceso WIFI y por su bajo precio el chip ESP8266 parece destinado a dar un gran empujón a lo que se ha llamado Internet de las cosas. [15]

Este módulo se comunica con comandos AT a través de un puerto serial. Permite la comunicación con otros dispositivos a través de sus GPIOs y un puerto UART. Tiene integrada una antena, amplificador de potencia, amplificador de recepción de bajo ruido, filtros y módulos de administración de energía. [3]

6.8. Microcontrolador

Los microcontroladores son un sistema de micromcomputadora completo. Es decir, en un solo circuito integrado contiene el microprocesador, memoria de datos, memoria de programa y las unidades de entrada/salida. Debido a su tamaño es barato y fácil de manipular por lo que es ideal para muchas aplicaciones de propósito específico.

Un microcontrolador es llamado un sistema cerrado ya que no se pueden cambiar componentes como microprocesador, capacidad de memoria, periféricos, etc [16].
Los microcontroladores son cada vez más utilizados debido a sus ventajas y facilidad de deseñar circuitos.

6.8.1. PIC16F887

El pic16f887 es fabriado por Microchip, el cual cuenta con las siguientes características básicas [17]:

- Arquitectura RISC: solo 35 instrucciones
- Frecuencia de operación 0-20MHz
- Voltaje de alimentación de 2-5V
- Ahorro de energía modo sleep
- 35 pines de entrada/salida
- Módulo PWM incorporado
- Módulo USART mejorado
- Memoria ROM de 8kb con tecnología FLASH

6.9. Modulación por ancho de pulso (PWM)

La modulación por ancho de pulso, (*Pulse width modulation*), es una de las estrategias más usadas para controlar señales AC o convertidores electrónicos de potencia. En esta técnica el ciclo de trabajo (*duty cycle*) de la señal cuadrada puede tomar una alta frecuencia para lograr un objetivo promedio de baja frecuencia de voltaje o corriente de salida [18].

Teoría de modulación es un área importante de investigación en Electrónica de Potencia por más de tres décadas y continúa teniendo atención e interés considerable. La modulación crea un tren de pulsos que tiene el mismo promedio fundamenta de voltios-segundos como referencia de la señal en cualquier instante [18].

El principal objetivo del PWM es calcular la frecuencia de la señal cuadrada para lograr el nivel promedio deseado de voltaje o corriente de salida. El segundo objetivo de la modulación es determinar la forma más efectiva de ordenar dicha frecuencia para minimizar los armónicos no deseados que puedan distorsionar la señal [18].

6.9.1. Única modulación de ancho de pulso

En una señal de único PWM la variación del ancho de pulso controla el voltaje de salida y solo hay un pulso por cada medio ciclo de trabajo. La frecuencia de referencia de la señal determina la frecuencia fundamental del voltaje de salida.

La ventaja de esta técnica es que incluso los armónicos están ausentes debido a la simetría del voltaje de salida a lo largo del eje x y los *enésimos* armónicos pueden ser del voltaje de salida si el pulso con es igual a $\frac{2\pi}{n}$. Sin embargo, una desventaja es que la salida de voltaje introduce una gran cantidad de armónicos que a una baja frecuencia de salida de voltaje, la distorsión puede ser significativa [18]

6.10. Controlador proporcional integral derivativo PID

El controlador PID, *proporcional-integral-derivativo*, responde al problema de rastreo en el que se busca que el valor de la señal de error tienda a cero. En su versión más simple presenta la siguiente forma:

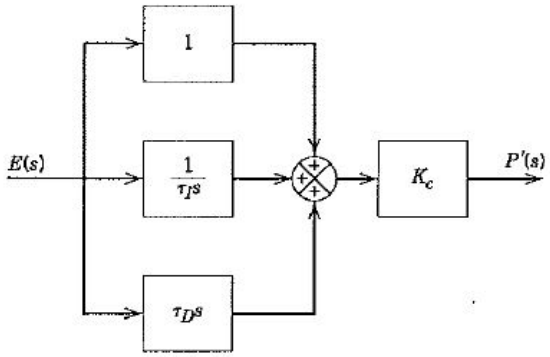


Figura 6: Diagrama de bloques de controlador PID

A continuación se detalla el aporte de cada controlador al sistema.

6.10.1. Controlador proporcional

El objetivo es reducir el error a cero, donde:

$$e(t) = y_{sp}(t) - y_m(t) \tag{2}$$

En donde:

$$\begin{aligned} e(t) &= Señal\,de\,error \\ y_{sp}(t) &= Referencia \\ y_m(t) &= Valor\,medido\,de\,la\,variable\,de\,control \end{aligned}$$

Para un control proporcional la salida controlada es proporcional a la señal de error.

$$p(t) = \bar{p} + K_c\, e(t) \tag{3}$$

En donde:

$$\begin{aligned} p(t) &= Salida\,controlada \\ \bar{p} &= Valor\,en\,estado\,estable \\ K_c &= Constante\,de\,ganancia \end{aligned}$$

El concepto detrás del controlador proporcional es el siguiente:

- La ganancia del controlador puede ser ajustada para que la salida del controlador cambie de forma sensible como se desee a las desviaciones entre la referencia y la salida de la variable controlada [19].
- La magnitud de K_c puede elegirse de modo que la salida controlada incremente o decremente[19].

En controladores proporcionales $\bar{p}$ puede ser ajustada, lo que se conoce como reinicio manual. porque la salida del controlador es igual a $\bar{p}$ cuando el error es cero [19].

6.10.2. Controlador integral

La salida del controlador integral depende de la integral de la señal de error a través del tiempo.

$$p(t) = \bar{p} + \frac{1}{\tau_I} \int_0^t e(t^*) dt^* \tag{4}$$

En donde τ_I es un parámetro ajustable referido al tiempo de la integral, tiene unidades de tiempo. En el pasado, la acción del control integral hace referencia al error flotante. La

acción integral es extensamente usada ya que provee una importante y práctica ventaja, elimina el *offset*, error en estado estacionario [19].

A pesar de que elimina el *offset* que es usualmente un importante objetivo de control. El controlador integral extrañamente es usado solo ya que una pequeña acción de control toma lugar hasta que la señal de error ha persistido por un tiempo.

En contraste, la acción del controlador proporcional toma acción inmediata tan pronto como el error es detectado. Consecuentemente, la acción del control integral es usa conjuntamente con el control proporcional, a lo que se conoce como *controlador Proporcional Integral (PI)* [19].

$$p(t) = \bar{p} + K_c \left(e(t) + \int_0^t e(t^*) dt^* \right) \tag{5}$$

La respectiva función de transferencia del controlador PI es:

$$\frac{P'(s)}{E(s)} = K_c \left(1 + \frac{1}{\tau_I \, s} \right) = K_c \left(\frac{\tau_I \, s + 1}{\tau_I \, s} \right) \tag{6}$$

6.10.3. Controlador derivativo

La acción del control derivativo es predecir el comportamiento futuro de la señal de error considerando su razón de cambio. En el pasado, la acción del control derivativo hace referencia a la razón de acción o control anticipado.

La estrategia anticipatoria debido a la experiencia de un operador puede ser incorporada en controladores automáticos haciendo que la salida del control sea proporcional a la razón de cambio de la señal de error o la variable controlada [19].

$$p(t) = \bar{p} + \tau_D \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{7}$$

En donde τ_D, el tiempo derivativo, tiene unidades de tiempo. Note que la salida del controlador es igual a la salida nominal del valor de $\bar{p}$ tan grande como el error constante. Consecuentemente, el control derivativo nunca es implementado solo, se usa junto con el control proporcional o proporcional-integral [19].

La función de transferencia de un controlador ideal PD es:

$$\frac{P'(s)}{E(s)} = K_c \left(1 + \tau_D s \right) \tag{8}$$

6.10.4. Controlador PID

Ahora consideraremos la combinación de los controladores Porporcional, Integral, Derivativo como controlador PID. El cual corresponde a la siguiente forma:

$$p(t) = \bar{p} + K_c \left[e(t) + \frac{1}{\tau_I} \int_0^t e(t^*) dt^* + \tau_D \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right] \tag{9}$$

Que corresponde a la siguiente función de transferencia

$$\frac{P'(s)}{E(s)} = K_c \left[1 + \frac{1}{\tau_I \, s} + \tau_D \, s \right] \tag{10}$$

6.10.5. Control PID de plantas

Si se puede obtener un modelo matemático de la planta, es posible aplicar diversas técnicas de diseño con el fin de determinar los parámetros del controlador que cumpla las especificaciones del transitorio y del estado estacionario del sistema en lazo cerrado. Sin embargo, si la planta es tan complicada que no es fácil obtener su modelo matemático, tampoco es posible un método analítico para el diseño de un controlador PID. En este caso, se debe recurrir a procedimientos experimentales para la sintonía de los controladores PID [20].

El proceso de seleccionar los parámetros del controlador que cumplan con las especificaciones de comportamiento dadas se conoce como sintonía del controlador. Ziegler y Nichols sugirieron reglas para sintonizar los controladores PID (esto significa dar valores a K_p, T_I y T_D) basándose en las respuestas escalón experimentales o en el valor de K_p que produce estabilidad marginal cuando sólo se usa la acción de control proporcional [20].

6.10.6. Reglas de Ziegler-Nichols para sintonizar controladores PID

Ziegler y Nichols propusieron reglas para determinar los valores de la ganancia proporcional K_p, del tiempo integral T_I y del tiempo derivativo T_D, basándose en las características de respuesta transitoria de una planta dada. Tal determinación de los parámetros de los controladores PID o sintonía de controladores PID la pueden realizar los ingenieros mediante experimentos sobre la planta [20]

Hay dos métodos denominados reglas de sintonía de Ziegler-Nichols. A continuación se hace una breve presentación de estos dos métodos.

Primer método

En el primer método, la respuesta de la planta a una entrada escalón unitario se obtiene de manera experimental. Si la planta no contiene integradores ni polos dominantes complejos conjugados, la curva de respuesta escalón unitario puede tener forma de S,

Este método se puede aplicar si la respuesta muestra una curva con forma de S. Tales curvas de respuesta escalón se pueden generar experimentalmente o a partir de una simulación dinámica de la planta. La curva con forma de S se caracteriza por dos parámetros: el tiempo de retardo L y la constante de tiempo T. El tiempo de retardo y la constante de tiempo se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de S y determinando las intersecciones de esta tangente con el eje del tiempo y con la línea $c(t) = K$ [20].

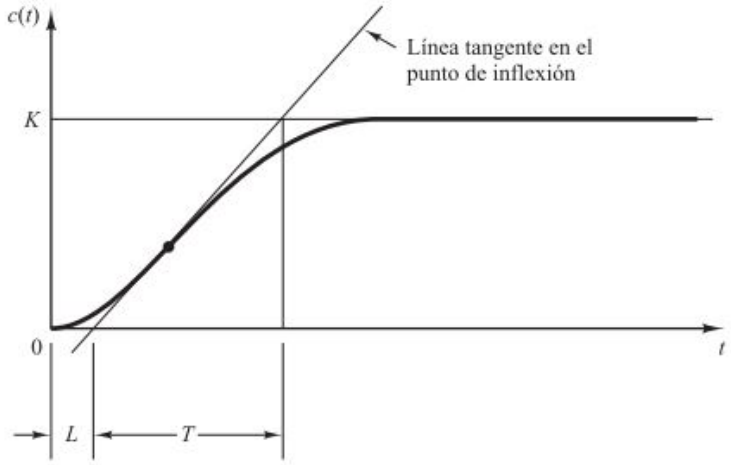


Figura 7: Curva de respuesta en forma de S

Tipo de controlador	K_P	K_I	K_D
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2\,L$	$0.5\,L$

Cuadro 1: Regla de Ziegler-Nichols basada en respuesta de la planta al escalón unitario

Segundo método

En el segundo método, primero se fija $T_I = \infty$ y $T_D = 0$. Usando solo la acción de control proporcional, se incrementa K_p desde 0 hasta un valor crítico K_{cr}, en donde la salida presente oscilaciones sostenidas. Si la salida no presenta oscilaciones sostenidas para cualquier valor que pueda tomar K_p, entonces este método no se puede aplicar. Así, la ganancia crítica K_{cr} y el periodo P_{cr} correspondiente se determinan experimentalmente

Tipo de controlador	K_P	K_I	K_D
P	$0.5\,K_{cr}$	∞	0
PI	$0.45\,K_{cr}$	$\frac{1}{1.2}\,P_{cr}$	0
PID	$0.6\,K_{cr}$	$0.5\,P_{cr}$	$0.125\,P_{cr}$

Cuadro 2: Regla de Ziegler-Nichols basada en la ganancia Kcr y Período Pcr

6.10.7. Diseño sin modelo de controladores PID

- Pueden ajustarse de forma heurística, es decir, no óptima ni perfecta pero "suficientemente buenaz práctica.
- No necesariamente necesita un modelo de planta exacto (o alguno como tal) para poder ajustarse, lo cual ocurre usualmente en la práctica.

Uno de los métodos empleados para ajustar un controlador PID cuando se desconoce la planta del sistema, consiste en utilizar una tabla como la que se muestra a continuación para ajustar de forma *manual* las constantes PID hasta hallar una combinación que produzca la respuesta transitoria deseada.

Parámetro	**Tiempo de subida**	**Sobre elevación**	**Tiempo de Asentamiento**	**Error estado estable**	**Estabilidad**
k_P	Decrece	Crece	Cambio pequeno	Decrece	Degrada
k_I	Decrece	Crece	Crece	Elimina	Degrada
k_D	Cambio menor	Decrece	Decrece	Sin efecto	Mejora si K_D pequeno

Cuadro 3: Respuesta al incremento de constantes del controlador PID

6.10.8. Control digital

Un controlador puede ser implementado de forma analógica y de forma digital. Cuando un controlador es implementado en un sistema computarizado, las entradas y salidas analógicas requieren de cierto tiempo de muestreo. Esto supone una limitación comparada con los controladores analógicos, los cuales no aportan ese atraso en el lazo de control [21].

A pesar de esta limitación, los controladores digitales son muy populares en la práctica debido a su facilidad de implementación gracias a la programación. Dependiendo de la forma en la que se implementa el control, la referencia de un lazo cerrado puede ser programada, lo cual permite disponibilidad para cambios al momento de la operación [21].

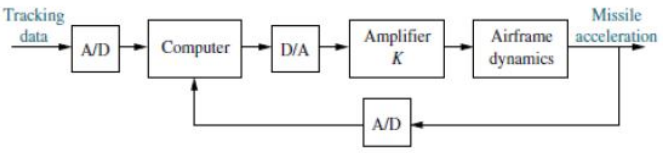


Figura 8: Implementación de un controlador digital

Otro aspecto importante a tomar en cuenta para la implementación digital de un sistema de control es la transformación necesaria para pasar de sistemas continuos en el tiempo a sistemas discretos. Esto es posible gracias a la transformada Z. Existen diferentes aproximaciones para implementar controladores diseñados en el dominio de la frecuencia. Algunos de estos son Forward Euler, Backward Euler y Tustin [21].

CAPÍTULO 7

Metodología

El desarrollo del sistema electrónico del robot esférico se divide en tres módulos principales:

- Módulo de potencia.
- Módulo de comunicación WIFI.
- Algoritmo de control punto punto.

7.1. Módulo de potencia

El módulo de potencia es el encargado de suministrar el voltaje y corriente necesarios para el funcionamiento óptimo del robot móvil. Por restricciones de diseño mecánico, de medidas compactas, se dispone de dos baterías de LiPo de $3.7\,V - 700\,mAh.$ con medidas de $34\,mm\,x\,25\,mm\,x\,7.5\,mm.$

Como componentes principales se tomaran el PIC16F887 como procesador de datos, módulo WIFI ESP8266 como encargado de comunicación WIFI entre el robot móvil y la interfaz digital de control, transistores 2N3904 como drivers de motores, regulador de 3.3V para el funcionamiento del módulo WIFI. La elección de usar un microcontrolador además del módulo WIFI es para realizar la configuración inicial del módulo a través del microcontrolador, por el puerto serial. En lugar de tener que conectar el módulo a una computadora para realizar dicha configuración inicial.

Además de LEDs, resitencias y capacitores varios según se necesiten para la implementación de los circuitos.

Esto con la finalidad de calcular la potencia que demandará el circuito y poder calcular el tiempo de autonomía que la batería es capaz de proporcionar. Para esta etapa se realizaran pruebas en una versión del circuito en protoboard. También se tomara en cuenta adquirir una batería de mayor capacidad de mAh y que sus dimensiones físicas no varíen demasiado respecto a la batería actual.

7.2. Módulo de comunicación WIFI

Las pruebas iniciales del circuito de comunicación WIFI se llevará a cabo en protoboard, con el fin de poder hacer cambios en los componentes y evaluar su funcionamiento previo a la fabricación de la tarjeta de control, PCB, que por restricciones de diseño se deberán usar únicamente componentes de superficie.

Se usará el módulo WIFI ESP8266 para la implementación de la red WIFI, ya que este módulo presenta características como: comunicación mediante el protocolo TCP/IP o amplia documentación para sus diferentes configuraciones. El protocolo TCP/IP por su parte también cuenta con documentación en diferentes entornos de programación por lo que se implementará para la conexión entre el interfaz de control y el robot móvil.

Para la primera prueba de comunicación se realizarán pruebas al robot diferencial en lazo abierto, es decir sin la base esférica ni el sistema de control ni la visión de computadora. Se enviará al robot un valor entre 0 y 100 que representa el porcentaje de velocidad que pueden tener los motores.

La comunicación tendrá la secuencia de comenzar en la computadora con el valor del porcentaje transformado a la resolución de PWM que tiene el pic16f887, este dato se envía al módulo WIFI y este respectivamente al pic16F887, que es el encargado de fijar el valor de PWM en cada uno de los canales que van hacia los motores.

Para esta etapa se tomara en cuenta trabajos de graduación previamente realizados en la línea de swarm de la Unviersidad del Valle de Guatemala, [4], [3], en donde se encontrará documentación relacionada con la estructura de comunicación entre una interfaz de control y el módulo WIFI. También se implementarán librerías relacionadas con comandos para el envío de datos.

Las primeras pruebas se realizaran en MATLAB como interfaz de control, en donde se encuentran las librerías ya implementadas. Luego se trasladará el protocolo de comunicación y las librerías implementadas a Python 2.7. Programa desde el que se realizará la parte de visión de computadora y el posterior algoritmo de control punto punto.

Finalmente para esta etapa se integrará la visión de computadora con el robot diferencial, sin la base esférica, para observar las trayectorias que puede realizar el robot diferencial y el comportamiento que tiene.

7.3. Algoritmo de control punto punto

El diseño del algoritmo de control punto punto es la última fase del proyecto completo del robot esférico. Ya que para llegar a esta fase es necesario tener la parte de potencia y de la comunicación WIFI, en la tarjeta de control PCB, integrada con las bases del diseño mecánico y la visión de computadora.

Las pruebas respectivas para la validación del algoritmo se llevarán a cabo en la plataforma de pruebas, Robotarium, del departamento de Ingeniería Electrónica-Mecatrónica de la Universidad del Valle de Guatemala.

El robot partirá de una posición inicial dentro del Robotarium y se le asignará una posición de destino desde la interfaz de control. Se tomará la información obtenida por la cámara para determinar la posición en la que el robot se encuentra en cada momento y compararla con la posición de destino. Esto con el fin de obtener una señal de error y en base al algoritmo asignar velocidades iguales o diferentes, según sea el caso, a cada llanta del robot móvil.

CAPÍTULO 8

Etapa de potencia

Debido a que una de las características de la parte física del robot esférico es que debe ser de tamaño compacto, la PCB debía tener un tamaño de aproximadamente $4\,cm$ por lado. por lo que se eligieron componentes de superficie, smd, que además de tener el tamaño compacto deseado, en algunos casos reduce el consumo de potencia promedio del componente.

8.1. Selección de componentes

8.1.1. Módulo WIFI

Para la implementación de la red inalámbrica se seleccionó el Módulo WIFI ESP8266 debido a los buenos resultados obtenidos en trabajos previos [3], [4]. Las características de este módulo son las siguientes:

- Voltaje de operación 2.7 - 3.6 V
- Interfaz periférica UART, HSPI, I^2C.
- Modo WIFI: Servidor, Cliente, Cliente-Servidor.

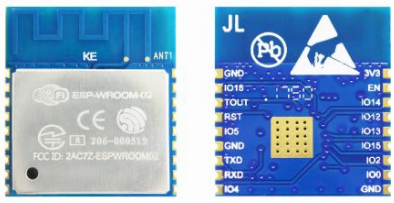


Figura 9: Módulo ESP8266 versión SMD

8.1.2. Microcontrolador

Debido a que el módulo únicamente se encarga de recibir y mandar datos se seleccionó el pic16f887 como procesador de la demás parte del circuito como el control de los motores DC mediante los canales PWM que incorpora, así como de indicadores del estado de proceso del funcionamiento del progrma según la fase en la que se encuentre. Dentro de las ventajas del pic16f887 se encuentra:

- Fácil acceso local
- Módulo USART, I^2C.
- Dosanales PWM
- Varios pines de entrada, salida.

Figura 10: PIC16F887 versión SMD

8.1.3. Batería de LiPo

Se seleccionó una batería recargable de Li-Po (Lithium Polymer) de 3.7V y 700 mAh con medidas de $34mm\,x\,25mm\,x\,7.5mm$, la cual necesita un valor mínimo de 4.2 V para su carga.

La elección de esta batería además de cumplir con requisito en dimensiones y voltaje entrega fue debido al buen resultado en trabajo previamente realizados [3].

Figura 11: Bateía de LiPo 3.7V 700 mAh

8.1.4. Drivers de motores

Debido a que la velocidad de los motores se controlan mediante los canales PWM del pic16f887, es necesario usar drivers para los motores. Esto debido a que los microcontroladores son dispositivos de voltaje y no de corriente. Cada pin provee $5\,V$ pero únicamente una corriente máxima de salida de $20\,mA$.

Se seleccionaron los transistores 2N3904 como drivers de motores. Como parte adicional del driver se necesita un diodo 1n4001. Esta configuración se conoce como *Flyback diode* o *Diodo de retorno*, el cual al estar conectado en paralelo a un inductor y con polaridad inversa a la fuente de voltaje evita los picos de voltaje que ocurren cuando se desconecta la fuente de voltaje.

(a) Transistor 2n3904 smd

(b) Diodo 1n4001 smd

Figura 12: Componentes de driver

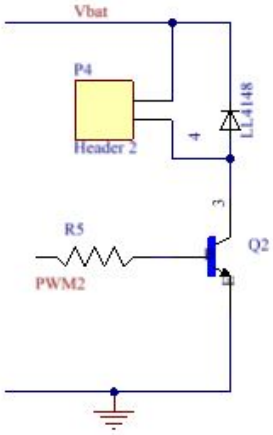


Figura 13: Driver para motor DC

8.1.5. Regulador de voltaje

Debido a que el módulo WIFI ESP8266 funcionan en un rango entre 3 y 3.6 V es necesario incluir dentro del circuito un regulador para alimentar el módulo WIFI. El regulador seleccionado fue el MIC5205 de microchip, el cual es capaz de dar 3.3 V de salida.

Figura 14: Mic 5205 con voltaje de salida de 3.3V

8.1.6. Cargador de batería

Para lograr el objetivo de tener un prototipo compacto y que no sea necesario desarmarlo por completo cada vez que se requiera cargar la batería, a la PCB se le colocará un cargador de batería para cumplir con dicha tarea.

El cargador de batería tiene un voltaje de salida de 4.17 V, valor que permite cargar adecuadamente la batería de LiPo sin dañar significativamente sus ciclos totales de carga.

Figura 15: Cargador de batería

8.1.7. Motores DC

Como actuadores para las llantas de los motores se eligieron los motores RB-DFR-637, los cuales tienen las siguientes características:

- Valores nominales 6 V - 96rpm
- Par de parada $<$ 1 kg $\cdot$ cm
- Corriente máxima 100 mA
- Rango de voltaje 3-9 V
- Peso 4 g

- Cuenta con propia caja de reducción.

Figura 16: Motores DC

8.2. Costo estimado módulo electrónico

Para establecer un costo de la fabricación de la parte electrónica del prototipo, la PCB y los motores DC se tomaron los siguientes valores:

Cantidad	**Componente**	**Precio $**
1	Pic16f887	5.00
1	ESP8266	13.00
1	MIC5205	1.00
1	LD1117	0.75
1	MCP73831	1.00
1	Conector Mircro USB	0.65
4	Diodos 1n4001	0.25
5	2n3904	1.00
6	LEDs	0.25
6	Capacitores	0.25
12	Resistencias	0.50
1	Fabricación PCB	10.00
4	Motores DC	20.80
2	Baterías Li-Po	17.00
	TOTAL	71.45

Cuadro 4: Costo de componentes de PCB y motores DC

Hay que notar que los componentes más caros son los motores DC, mientras que el costo del resto de los componentes electrónicos es de aproximadamente 24 $ lo que equivale $Q\,190$.

A esto hay que agregar que se tomó en cuenta el precio unitario de cada componente, sin tomar en cuenta otros factores como disponibilidad local, cantidad mínima de pedido (MCP73831, MIC5205) o envío del pedido (MCP73831, MIC5205, ESP8266).

De la misma forma el precio de la fabricación de la PCB se cotizó en dos sitios web, obteniendo un aproximado de 50 $ por 5 placas, por lo que en el Cuadro 4 solo se coloca el valor de una PCB. Las cotizaciones se muestran en la sección de Anexos.

8.3. Cálculo de potencia

Luego de tener los componentes definidos se realizó el cálculo de potencia estimada que consumirán los principales componentes, basándose en datos de hojas de datos o especificaciones de proveedores. Para este cálculo estimado se tomo el caso más crítico que es cuando se encuentran todos los componentes funcionando a su máxima capacidad.

Cantidad	**Componente**	**Potencia** (mW)
1	pic16f887	500
1	ESP8266	560
1	MIC5205	455
5	2n3904	1000
5	LED	500
4	1n4001	400
4	Motores DC	400
TOTAL		3815 mW

Cuadro 5: Potencia disipada de componentes principales

Como se observa en el Cuadro 5 la potencia que requieren los componentes principales del circuito es de 3815 mW.

8.4. Autonomía

El propósito de calcular el consumo de potencia del circuito es que conjuntamente con los datos de la batería podemos encontrar el tiempo de uso funcional que esta es capaz de dar al circuito.

Debido a los diferentes valores de voltaje de operación de componentes como el pic16f887, el módulo WIFI ESP8266 o los motores DC, con el objetivo de tener mejores márgenes de operación para cada componente se usaron dos baterías en serie de LiPo 8.1.3.

Las dos baterías son capaces de suministrar una carga de $5180\,mWh$ ya que al estar en serie las baterías el voltaje se suma para un total de $7.4\,V$ mientras que la corriente permanece igual $700\,mAh$.

Teniendo la ecuación de capacidad de carga de una batería:

$$C[mWh] = P[mW] \cdot t[h] \tag{11}$$

De la ecuación anterior despejando para el tiempo y sustituyendo los valores calculados de carga y consumo de potencia se tiene una autonomía aproximada de:

$$t = 1.36\,h \equiv 82\,min \tag{12}$$

El resultado nos indica que la batería es capaz de dar 82 minutos de uso continuo, Como dato adicional hay que tener en cuenta que durante este período de tiempo la batería funciona como una fuente de voltaje que otorga $7.4\,V$ y $1.2\,A$ como máxima corriente de salida.

8.4.1. Cálculos experimentales de autonomía

Una medición más acertada fue observar el consumo de corriente cuando los motores DC estaban en su mínimo y máximo punto de operación, con $7.4\,V$ como alimentación del circuito. Se midió dicha corriente con un multímetro dando como medidas de 97 y $180\,mA$ para los valores mínimo y máximo respectivamente.

Se observó también el consumo de corriente con el amperímetro integrado en las fuentes de voltaje de los laboratorios de la Universidad del Valle de Guatemala con lo que se obtuvieron valores de 82 y $170\,mA$ mínimo y máximo respectivamente.

Tomando el promedio de los valores mínimo y máximo de corriente junto con el voltaje de alimentación se puede determinar un rango de valores experimentales de potencia del circuito.

$$min = 670\,mW \quad ; max = 1300\,mW \tag{13}$$

Si bien estos valores son mucho menores al total del Cuadro 5 se debe tomar en cuenta que no todos los componentes se encuentran operando en su máximo potencial como se detalló en ese Cuadro. Además que se tomó como punto de prueba una posición en dónde las llantas del robot móvil no estaba en contacto con ninguna superficie por lo que podían girar libremente y por lo tanto necesitaban menos torque para poder moverse.

Debido a que el valor de carga de la batería no cambia respecto al de la sección anterior (8.4) se puede calcular un intervalo para la autonomía de la batería

$$min = 4\,h \quad ; max = 7.73\,h \tag{14}$$

CAPÍTULO 9

Establecimiento de red WIFI

9.1. Esquemático implementado

Para la implementación física del la red WIFI como primer punto se construyó el circuito en protoboard, con el fin de comprobar su correcto funcionamiento previo a la fabricación de la PCB. Luego de comprobar su correcto funcionamiento se procedió a diseñar el PCB para la fabricación de la tarjeta de control. Entre el circuito utilizado en protoboard y en PCB solo cambia la versión del módulo ESP8266 y la respectiva conexión de los pines. En el protoboard se usó la versión ESP-01 que consta de 8 pines mientras que en la PCB se usó la versión ESP-12 que consta de 19 pines. En ambos casos ses usó el firmware de fábrica que traen los módulos WIFI.

A continuación se detallan los bloques más importantes del circuito implementado en la PCB.

9.1.1. Cargador de batería

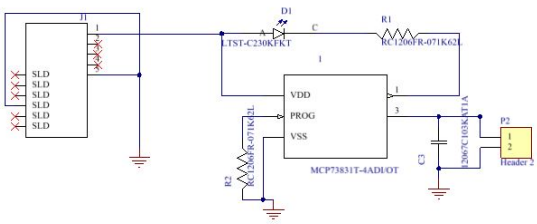


Figura 17: Circuito de cargador de batería

Esta parte del circuito es la encargada de cargar la batería de LiPo, consta del conector Micro-USB el cual se puede conectar a un cargador de celular, a una batería externa o al puerto USB de una computadora.

Para la conexión del cargador de batería se usó la siguiente configuración:

- VDD: Voltaje de alimentación, 6V recomendado.
- V_{Bat}: Conectar a terminal positiva de la batería y a un capacitor mínimo de $4.7\,\mu F$ para asegurar estabilidad en el voltaje de salida.
- PROG: Una resistencia entre PROG y V_{ss} activa condiciones de preacondicionamiento, carga rápida y terminación. El controlador de gestión de carga se puede deshabilitar al no conectar nada a PROG.
- STAT: Es una salida para conectar a un LED de indicador de carga. STAT es una salida lógica de tres estados en el MCP73831.

9.1.2. Conexión módulo WIFI

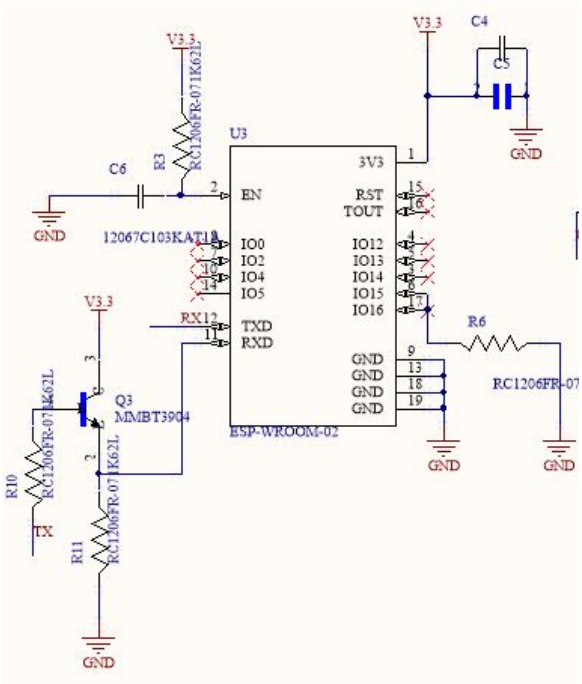


Figura 18: Conexión del módulo ESP8266 versión smd

Esta figura muestra la forma de conectar el módulo ESP8266-12, versión SMD. Cabe destacar que el módulo WIFI tiene la opción de actualizar el firmware que trae de fábrica, esto mediante el pin número 9, $IO0$ según la hoja de datos. Sin embargo para la implementación en la PCB no se colocó esta opción, solo en el circuito probado en protoboard. Para el módulo versión ESP-12 se trabajó con la versión base de firmware y se utilizó en la siguiente configuración:

- Pin 1 - 3.3 V: Conectado a alimentación
- Pin 2 - EN: Enable, conectado a uno lógico mediante una resistencia.
- Pin 6 - IO15: Conectado con un *pull down*.
- Pin 9 - GND: Conectado a GND
- Pin 11 - RXD: UART0_RXD, recive datos en comunicación serial
- Pin 12 - TXD: UART0_TXD, transmite datos en comunicación serial
- Pin 13 - GND: Conectado a GND
- Pin 18 - GND: Conectado a GND
- Pin 19 - GND: Conectado a GND

Debido a que el voltaje de alimentación del circuito general era mayor a los 3.3 V de alimentación del módulo se colocó un acople de voltaje, mediante un transistor, entre el pin TX del pic16f887 y el pin RX del módulo ESP8266.

El módulo WIFI es el encarado de conectarse a la misma red que la interfaz de control. Se configura mediante la recepción de comandos *AT* y toma una dirección IP aleatoria de la red, Una vez establecida la conexión con la interfaz de control el módulo ESP8266 recibe los datos vía WIFI y los envía al pic16f887 por medio del módulo UART.

Conexión PIC16F887

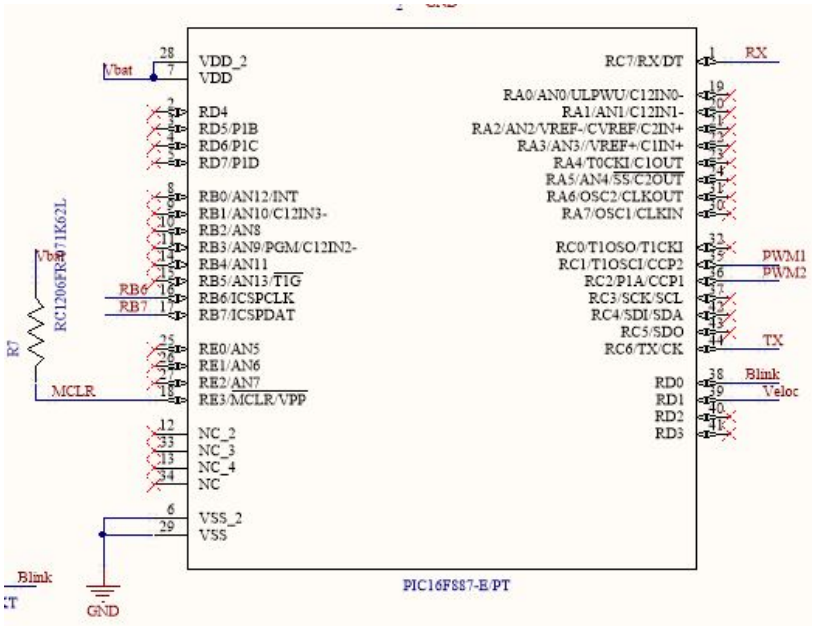


Figura 19: Conexiones de pic16f887

Desde la interfaz de control se envía un valor de velocidad entre 0 y la velocidad máxima alcanzada por el robot móvil. El pic16f887 recibe estos datos del módulo WIFI mediante comunicación serial (UART) y al convertir estos datos a la resolución de la señal de PWM que genera, estos valores se fijan en cada uno de los canales PWM para la velocidad de cada motor DC. Los pines usados por el pic16f887 fueron:

- Pin 1 - RX: Recive los datos de la comunicación serial.
- Pin 6 - Vss: Conectada a GND.
- Pin 7 - Vdd: Voltaje de alimentación.
- Pin 16 - RB6: ICSPCLK pin usado en la programación del pic.
- Pin 17 - RB7: ICSDAT pin usado en la programación del pic.
- Pin 18 - MCLR: Conexión *pull up* y usado en la programación del pic.
- Pin 28 - vdd: Voltaje de alimentación.
- Pin 29 - Vss: Conectada a GND.
- Pin 35 - CCP2: Salida del canal 2 de PWM.
- Pin 36 - CCP1: Salida del canal 1 de PWM.
- Pin 38 - RD0: Conectado a un LED que oscila cuando se realiza la configuración del módulo WIFI.
- Pin 39 - RD1: Conectado a un LED que enciende junto con el circuito.
- Pin 40 - RD2: Conectado a un LED que enciende para saber la IP a la que el módulo se conectó.

9.1.3. Programación PIC16F887

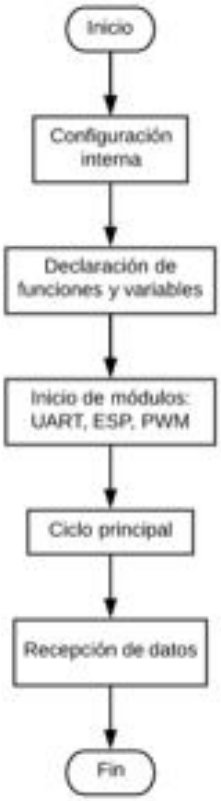


Figura 20: Diagrama de flujo del código del pic16f887

El diagrama de flujo corresponde a la acción del programa implementado en el pic16f887. El inicio corresponde a la declaración de la palabra de configuración, importar librerías de interés y declaración de funciones y variables usadas dentro de los módulos. Así como configuración respecto a la frecuencia del oscilador, interrupciones de diferentes módulos y pines como entradas o salidas. Luego pasa a configurar los módulos principales:

- Start_PWM(): En este módulo se habilitan los pines CCP como salidas de PWM, Así mismo se configuran parámetros del Timer 2 necesarios para la implementación del módulo.
- Start_UART(): En este módulo se habilita el puerto serial, se configura el pin RX como entrda y TX como salida, habilita transmisión y recepción de 8 bits. Se configura el baudrate de alta velocidad y 16 bits. Con lo que se calcula el valor n que permita un baudrate de 115200 mediante la siguiente fórmula:

$$BaudRate = \frac{Fosc}{[4(n+1)]} \tag{15}$$

Despejando para n

$$n = \left[\frac{Fosc}{4 \cdot BaudRate}\right] - 1 \tag{16}$$

Dando como valor $n = 16$, valor a cargar a los registros SPBRG y SPBRGH.

- Start_ESP(): En este módulo se configura desde el pic el módulo WIFI mediante comandos *AT*, la lista de comandos con la respectiva respuesta del módulo son las siguientes:

Comando AT	**Descripción**	**Respuesta**
AT+RST	Reinicia el módulo	ready
ATE0	Desactiva echo serial	OK
AT+CWMODE_CUR=1	Modo cliente	OK
AT+CWJAP_CUR='user','pass'	Conecta a red WIFI	OK
AT+CIPMUX=1	Multiple conexión	OK
AT+CIPSERVER=1,'puerto'	Conexión en puerto designado	OK

Cuadro 6: Lista de comandos AT para configurar módulo wIFI

En donde *user* es el nombre de la red WIFI, *pass* es la contraseña de la red WIFI y *puerto* es el puerto TCP mediante el que se va a establecer la conexión.

9.1.4. Conexión con MATLAB

Para la primera versión de conexión la interfaz de control WIFI fue MATLAB. Se implementó parte de la librería previamente desarrollada *SwarmLib* [4]. Debido a que para este caso no era necesario implementar todas las funciones, se tomaron únicamente las siguientes:

- GetIP(Nombre, Puerto) - IP: Obtiene la IP a la que el módulo WIFI se ha conectado automáticamente. Como parámetros recibe el nombre que se le ha asignado al robot móvil y el puerto de comunicación.
- ConexiónTCP(IP, Puerto) - TCP: Instancia la conexión TCP con la IP del robot. Como parámetro recibe la IP y puerto con el que se debe conectar.
- EnviarComando(Com, ObjTCP): Envía instrucción al robot mediante el objeto TCP previamente instanciado.
- SetVelociad(Con, ObjTCP): Se elige la llanta a la que se enviará velocidad, la cantidad de velocidad que se enviará, valor entre 0 y 100, y el objeto TCP al que se enviará la información.
- SetVelocidades(VelIzq, VelDer, ObjTCP): Se ingresa el valor que se quiere colocar a cada llanta, puede ser el mismo valor o uno diferente y el objeto TCP al que se enviará la información.

```
>> t = tcpclient('192.168.1.22', Puerto, 'ConnectTimeout', 1);
>> TCP = ConexionTCP('192.168.1.22', Puerto);
>> SetVelocidades(70,70,TCP);
>> SetVelocidades(40,40,TCP);
>> SetVelocidades(10,90,TCP);
```

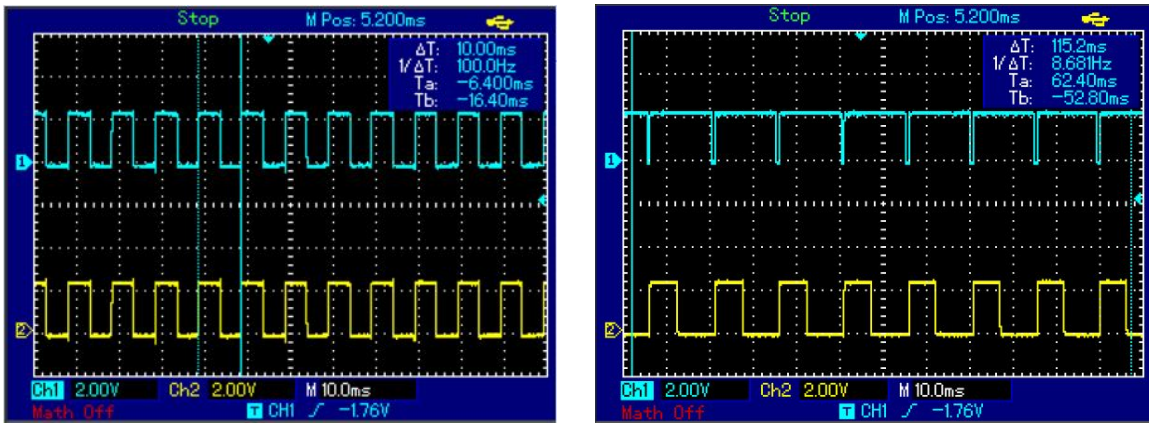
En la figura anterior se observa la estructura que tienen los comandos en la comunicación desde MATLAB, de primero se verifica la conexión tomando en cuenta la IP a la que el módulo se ha conectado de forma automática, el nombre del robot y el puerto mediante el cual se están conectando. Luego se instancia la conexión con la función TCP. El método da como parámetro 1 segundo de espera para la conexión, si el tiempo pasa y no se ha establecido la conexión devuelve un mensaje de error.

Luego se envía la velocidad como porcentaje entre 0 y 100 y el objeto con el que estableció la conexión.

El porcentaje de velocidad dentro de la función SetVelocidades() realiza un cambio de escala del valor ingresado de 0 - 100 a 0 -255, valor de resolución del PWM en el pic16f887. Además el mensaje se codifica de tal forma que siempre se envían tres valores correspondiendo a centenas, decenas y unidades, incluso si el valor ingresado es únicamente de uno o dos dígitos.

Este mensaje se decodifica en el pic16f887 para ingresar el valor correcto al registro que controla el ciclo de trabajo del PWM.

Como prueba de envío correcto de datos en la conexión entre la interfaz de control y protoboard, se observó en el osciloscopio el cambio en el ciclo de trabajo de ambos canales PWM del pic16f887. Obteniendo los siguientes resultados:



(a) Mismo ciclo de trabajo (b) Diferente ciclo de trabajo

Figura 21: Resultados de osciloscopio

El resultado de la implementación se observa en las figuras anteriores. El ciclo de trabajo cambia según el porcentaje de velocidad que se envíe desde MATLAB, en el primer caso se tienen ambas velocidades al 50 % mientras que en el segundo caso el canal 1 del osciloscopio muestra un ciclo de trabajo casi del 100 % mientras que el canal 2 muestra un ciclo de trabajo aproximado de 50 %.

CAPÍTULO 10

Diseño de PCB y ensamble con diseño mecánico

10.1. Versión inicial de PCB

Para la primera versión de PCB se tomo el siguiente esquemático, cuyas partes más importantes se encuentran detalladas en el capítulo anterior.

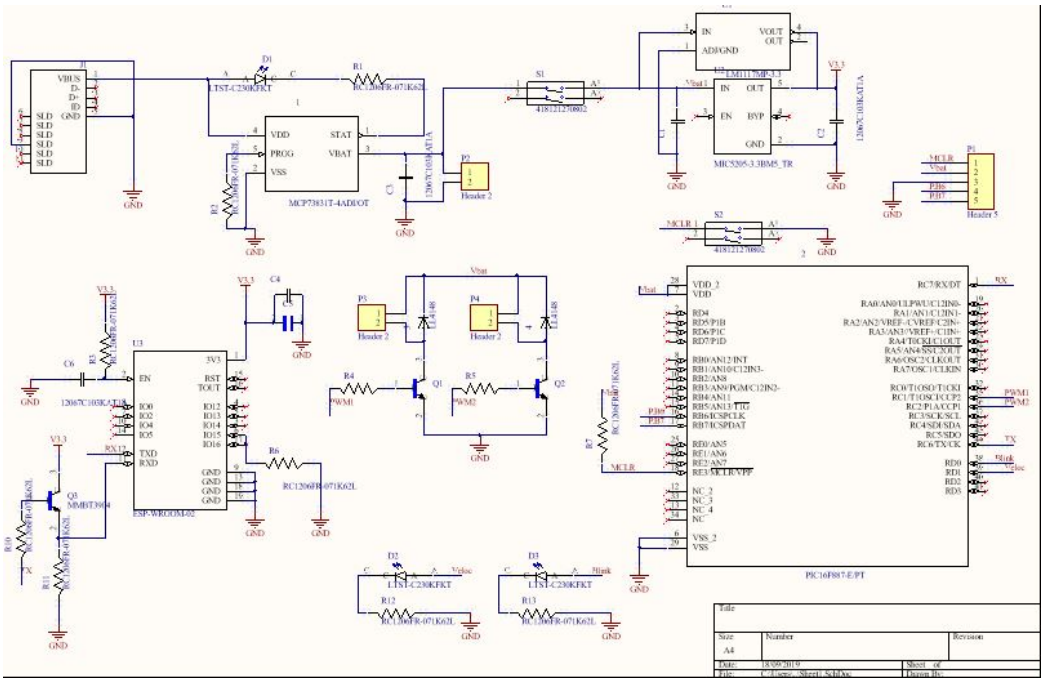
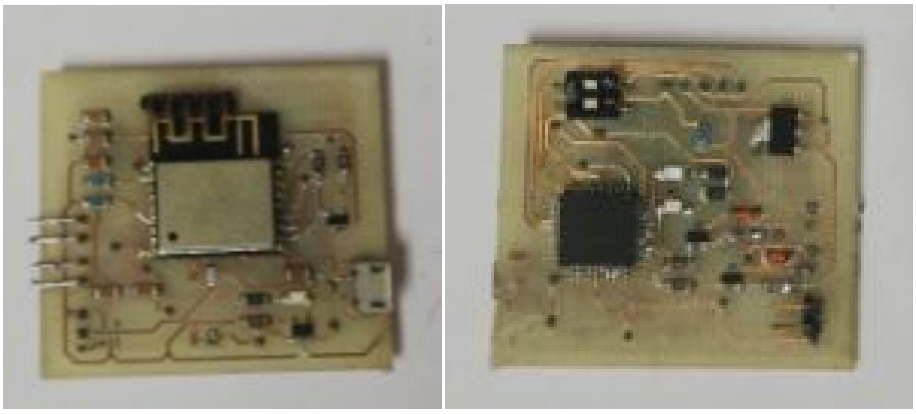


Figura 22: Esquemático implementado

Debido a restricciones del diseño mecánico, el PCB se diseñó en un espacio rectangular con dimensiones de $3.8\,x\,4.2\,cm$ dando como resultado la siguiente PCB.



(a) Cara principal de PCB v1 (b) Cara posterior de PCB v1

Figura 23: Versión 1 de fabricación de PCB

En la cara principal de la PCB se encuentran los siguientes conectores:

- Switch para On/Off del circuito.
- Puerto Micro USB para la carga de la batería.
- Conector para la salida de los dos motores DC.
- Conector para programación del pic.

En la cara posterior de la PCB se encuentra:

- Conector para la batería.
- Switch para resetear el pic en caso de ser necesario.
- Dos LEDs indicadores de secuencias del programa.

10.2. Ensamble inicial con diseño mecánico

Para completar el prototipo de robot esférico se ensambló la PCB con la base para la placa de control y la esfera, desarrolladas en el diseño mecánico.

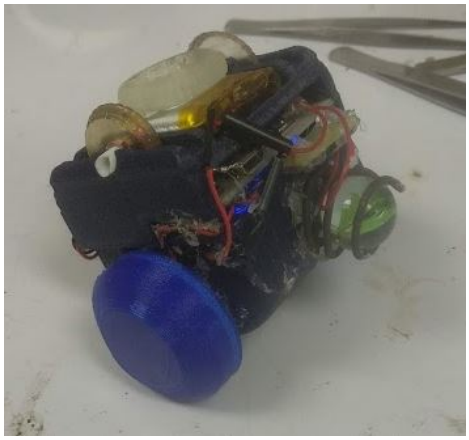
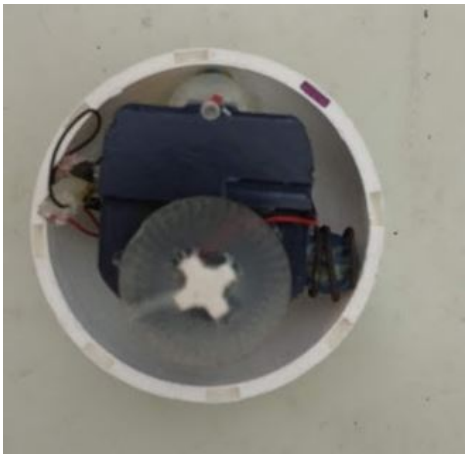
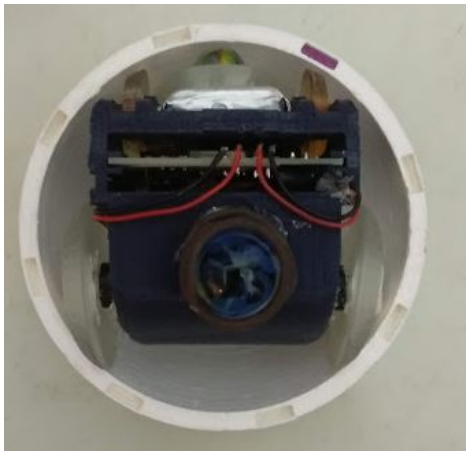


Figura 24: Instalación de módulo electrónico



(a) Posición probada para locomoción



(b) Posición probada para locomoción

Figura 25: Ensamble dentro de la esfera

En la Figura 24 se observa la instalación de la PCB dentro de la base tanto para la placa de control como para los motores DC. En esta base además se observa un tipo de llantas de TPU. Estas llantas son una mezcla entre cónicas y cilíndricas con el objetivo de que la base diferencial tuviera más punto de apoyo al momento de moverse.

En la Figura 25 se observa la forma en que la base del robot diferencial se ajusta a la superficie interna de la esfera. En este caso se observa que las llantas son puramente cónicas ya que el tipo de llantas cónica-cilíndrica tenía el problema de quedar demasiado ajustada dentro de la esfera. El material usado para estas llantas fue resina con el objetivo de generar fricción para lograr la locomoción del robot esférico.

Para una efectiva locomoción la base del robot debe permanecer en la posición original como se muestra en la Figura 25a o 25b. Esto con la finalidad de que el movimiento sea producto del torque generado por los motores. Los puntos de apoyo de la parte superior de la base del robot a la esfera para no perder esta posición. Esto logra que el movimiento lineal que tendría la base diferencial se transmita al robot esférico. Y así el robot esférico mostrar

trayectorias similares a las de un robot diferencial.

Sin embargo al comenzar a hacer pruebas se observaron los siguientes aspectos:

- La base del robot no permanecía en su posición original (25a o 25b), esto causaba que la base se desfasara y por lo tanto no pudiera rotar. Esto debido a la falta de restricción en alguno de los tres ejes (x,y,z), ya que si la esfera generaba un poco más de presión axial los motores no lograban girar. Mientras que en el eje z los soportes superiores no generan suficiente presión para que la base solo deslizara.
- Los motores no son capaces de entregar suficiente torque, ya que ante una no muy elevada fuerza axial dejaban de girar.
- La resolución usada en la impresión 3D no proporciona una superficie del todo *esférica* ya que presentaba una especie de *arrugas* en las partes laterales, más no en las uniones de la esfera.

Pese a que en una configuración específica la esfera lograba movimiento este no era el esperado, ya que la prueba se hizo con la misma velocidad a cada llanta, lo que debería traducirse a un movimiento lineal hacia adelante. Sin embargo la esfera comenzaba a moverse y luego de una pequeña distancia la base del robot se desfasaba lo que hacía que el robot perdiera la dirección de movimiento y por lo tanto tomaba cualquier dirección como trayectoria.

Se observó el movimiento solo de la esfera sin la base del robot, con lo que se detalló que esta tiende a una posición de *estabilidad* en las partes laterales de la superficie. Esto juntamente con las muescas que presentaba en las uniones de ambos hemisferios hacían que la esfera no tuviera un punto pivote fijo de apoyo o que costara mucho encontrarlo, lo que dificultaba la practicidad para comenzar el movimiento.

Debido a que el movimiento era prácticamente nulo y cuando lo tenía no era el esperado, desde el punto de vista de un sistema de control iba a ser prácticamente imposible conseguir controlar el sistema. Ya que el sistema se podría catalogar como *inestable*.

Por estas razones se cambió el diseño mecánico así como parte del sistema electrónico.

10.3. Versión final de PCB

Debido a las razones expuestas en la sección anterior, se cambió ligeramente el esquemático dando como resultado el siguiente circuito:

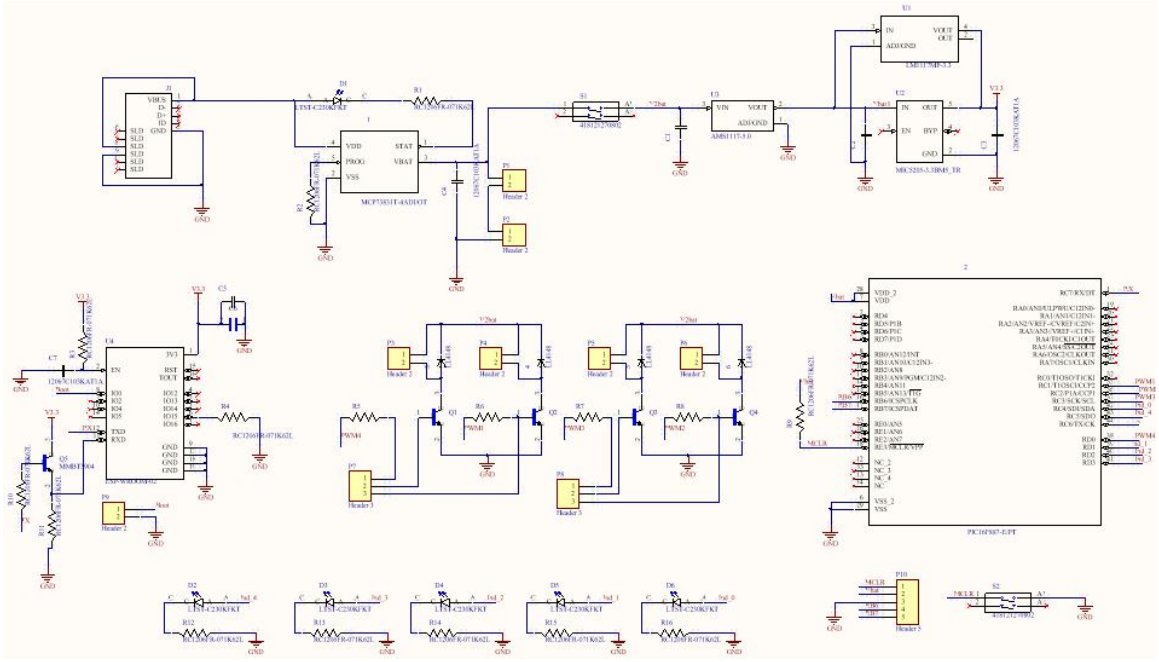


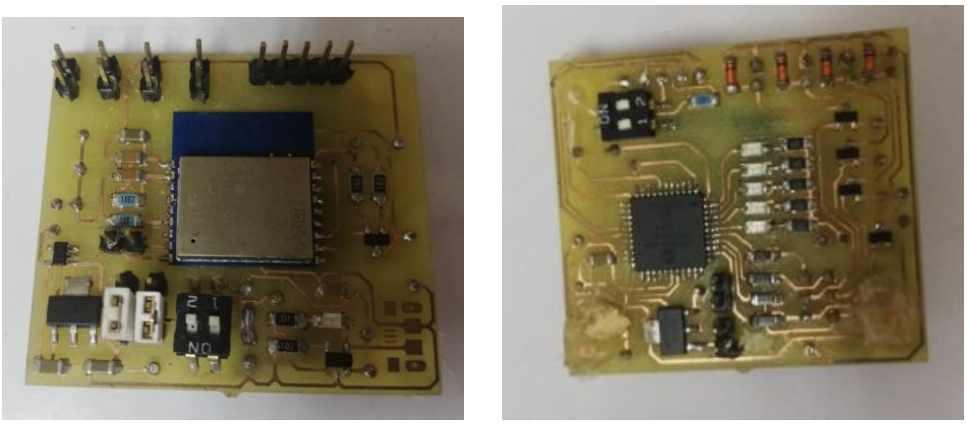
Figura 26: Esquemático completo final

Respecto a la versión inicial los cambios más significativos son:

- El tamaño de la placa aumentó a $4.2\,x\,4.7\,mm$
- Se agregó un regulador extra de 3.3 V.
- Se agregaron dos drivers para motores DC con el fin de tener 4 motores en total.
- Se agregaron 3 LEDs indicadores de secuencias del programa.
- Se agrego un switch para que el módulo WIFI ESP8266 pueda entrar en modo *Boot* y poder configurar su firmware.

Cabe mencionar que los motores DC tienen la opción de funcionar en *pareja*, es decir dos motores conectados al mismo canal PWM o cada uno de forma individual.

El resultado de la PCB final fue el siguiente:



(a) Cara principal de PCB versión final (b) Cara posterior de PCB versión final

Figura 27: Versión final de fabricación de PCB

En la cara principal de la PCB se encuentran los siguientes conectores:

- 1 switch de On/Off del circuito.
- 2 headers para elección de operación de canales PWM.
- 4 conectores para la salida de los dos motores DC.
- 1 conector para programación del pic.

En la cara posterior de la PCB se encuentra:

- 2 conectores para las baterías.
- 1 switch para resetear el pic en caso de ser necesario.
- 5 LEDs indicadores de secuencias del programa.

El circuito electrónico se modificó para solucionar algunos de los problemas del ensamble descritos en la sección anterior. Sin embargo estos cambios responden principalmente a los cambios que se realizaron en el diseño mecánico.

Al circuito original se le agregaron dos drivers para motores DC esto con la finalidad de solucionar en parte el problema de la falta de torque que habían presentado las pruebas iniciales. Estos drivers tienen la opción de funcionar los cuatro de forma independientes, es decir a cada motor DC se le debe asignar una velocidad. También pueden funcionar en parejas, los dos motores del lado derecho funcionan con el mismo canal PWM y lo mismo para los dos motores del lado izquierdo.

10.4. Ensamble final con diseño mecánico

Como parte del diseño mecánico y ante la falta de estabilidad que mostraba el robot diferencial dentro de la esfera se modificó tanto el diseño del robot diferencial como el de la esfera. En este caso la esfera ya no se trata de una impresión 3D sino de un molde de plástico inyectado.

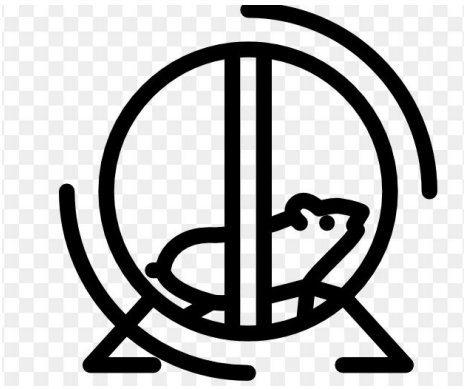
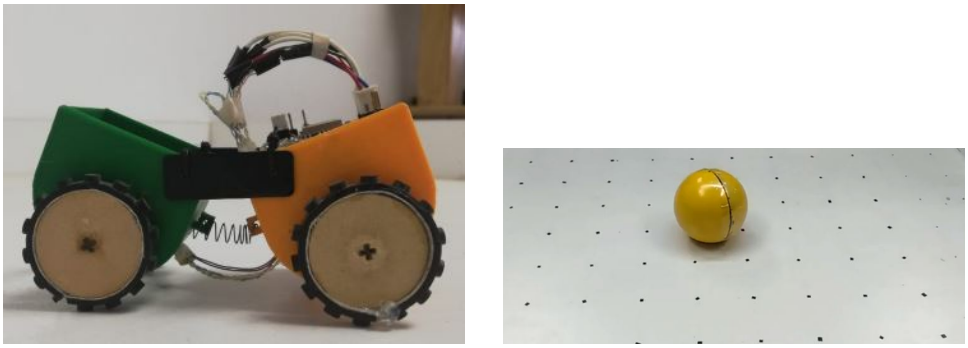


Figura 28: Imagen ilustrativa del movimiento de un hamster en una rueda

El nuevo diseño del robot diferencial se basa en el movimiento de un hamster adentro de una rueda. El hamster al estar en movimiento se mantiene fijo en la parte de abajo de la rueda, mientras que la rueda gira sin ningún tipo de restricción. Aplicar este movimiento al nuevo prototipo de robot sería: el robot diferencial representa al hamster y la esfera representa a la rueda que es la que tiene el movimiento final.

El nuevo robot diferencial se forma como la *unión* de dos robots diferenciales articulados por la parte del medio (Fig. 29a). La articulación tiene como objetivo ajustarse de mejor forma a la superficie interna de la esfera, simulando el movimiento que tendría la columna vertebral del hamster.

Así mismo, las llantas son cilíndricas con una goma rodeando la superficie con la idea de generar más fricción entre la superficie y las llantas.



(a) Robot diferencial (b) Robot diferencial en esfera

Figura 29: Versión final de robot esférico

CAPÍTULO 11

Integración de prototipo

11.1. Socket en Python

Luego de observar el correcto funcionamiento de la comunicación WIFI en MATLAB se procedió a implementar dicha comunicación en Python, ya que en este lenguaje de programación se encontraba desarrollada la visión de computadora.

Para ello se usó el módulo *socket* de Python el cual permite realizar la conexión WIFI mediante protocolo TCP/IP. A continuación se muestra un extracto de la configuración del *socket*:

```
import socket
obj = socket.socket()
ser = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
obj.connect((host, port))
obj.send('u'+vd+vi+'\r\n')
obj.close()
```

Se debe importar el módulo *socket* para poder usar las funciones necesarias para implementar la comunicación. Se instancia un objeto de tipo *socket* como cliente y luego se instancia un objeto tipo servidor que detalla el tipo de conexión a utiliar:

- socket.AF_INET: Hace referencia a conexión mediante IPv4
- socket.SOCK_STREAM: Hace referencia a conexión TCP, la otra opción que posee es conexión UDP.

Para la conexión con el host se indica hacia que dirección IP se va a establecer la comunicación y mediante que puerto. En este punto el *socket* ya se encuentra configurado para la transmisión y recepción de datos. Los tipos de datos que se pueden enviar a través del módulo *socket* son de tipo *string* o *buffer*.
Finalmente para cerrar la conexión se utiliza el comando *.close()*.

11.2. Interfaz de control

Para completar la parte de programación se integró en un documento de Python la parte de visión de computadora junto con la comunicación TCP y el algoritmo de control punto punto.

En el siguiente diagrama de flujo se muestra la lógica de la secuencia que tiene el programa denominado *Robot-Esférico*.

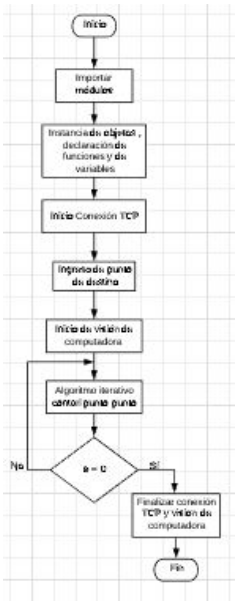


Figura 30: Diagrama de flujo programa de Python

El programa comienza cuando se instancian los objetos para la comunicación TCP/IP y visión de computadora. Luego se establece la comunicación WIFI con la dirección IP y puerto a la que el módulo se ha conectado. Esta dirección IP puede variar por lo que previo a correr el programa será necesario identificar la dirección a la que el módulo se ha conectado y modificarla de ser necesario.

Luego el usuario ingresa una coordenada (x,y) dentro del rango de resolución de la cámara, $600\,x\,450$. Esta coordenada será el punto final de destino que deberá alcanzar el robot esférico. Posteriormente se configura e inicia la etapa de visión de computadora, la cuál por defecto reconoce el centro de masa de colores naranja. Se muestra en una ventana

la cual muestra la circunferencia del *A naranja* e imprime en la consola el valor del centro de masa.

El centro de masa obtenido por medio de la cámara representa la coordenada actual del robot esférico. Este valor será comparado con la coordenada de destino para realizar el algoritmo de control punto punto. El cual al detectar que el error en cuanto a la coordenada actual y la coordenada de destino sea lo suficientemente cercano a cero detendrá el robot móvil en esa posición final.

11.3. Modelado cinemático de robot diferencial

Para el modelado cinemático del robot diferencial se cuenta con un marco de referencia inercial fijo, como se muestra en la siguiente imagen:

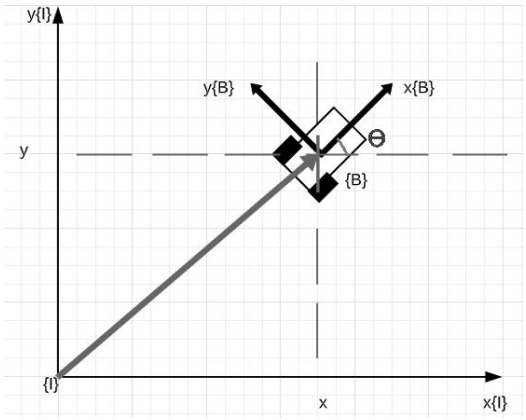


Figura 31: Robot diferencial en plano x-y

En donde podemos observar que necesitamos tres parámetros para identificar la posición del robot x, y, θ. Es decir tenemos la siguiente configuración del robot respecto al marco de referencia inercial. El superíndice I hace referencia al marco inercial global, mientras que el superíndice B hace referencia al marco local del robot diferencial.

$$\mathbf{q} = \xi^I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{17}$$

El movimiento del robot esta determinado por la velocidad que puede alcanzar, es decir:

$$\dot{\mathbf{q}} = \dot{\xi}^I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{18}$$

Sin embargo al observar el marco de referencia local del robot observamos que este solo puede avanzar hacia el frente x y ajustar su dirección, no puede hacer movimientos laterales

y, por lo que se tiene la siguiente configuración:

$$\dot{\xi}^{\mathbf{B}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v \\ 0 \\ \dot{\theta} \end{bmatrix} \tag{19}$$

Con lo que es necesario un jacobiano que permita transformar la configuración del marco de referencia local B al marco de referencia inercial I. Por lo que se comienza definiendo:

$$\mathbf{v}^B = \begin{bmatrix} v \\ 0 \end{bmatrix} \tag{20}$$

Con lo que se puede notar que

$$\mathbf{v}^I = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}^I = R_B^I \, \mathbf{v}^B \tag{21}$$

Si se define a R_B^I como la siguiente matriz de rotación

$$R_B^I = \begin{bmatrix} cos\theta & -sen\theta \\ sen\theta & cos\theta \end{bmatrix} \tag{22}$$

Con esto se puede observar que la velocidad angular tanto en el marco de referencia inercial como en el local es la misma

$$\dot{\theta}^I = \dot{\theta}^B \tag{23}$$

Sabiendo además que

$$\xi = \begin{bmatrix} \mathbf{v} \\ \dot{\theta} \end{bmatrix} \tag{24}$$

Se pueden juntar las expresiones 21 y 23 para obtener

$$\dot{\xi}^I = \begin{bmatrix} \mathbf{v}^I \\ \dot{\theta}^I \end{bmatrix} = \begin{bmatrix} R_B^I \, \mathbf{v}^B \\ \dot{\theta}^B \end{bmatrix} \tag{25}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}^I = \begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ 0 \\ \dot{\theta}^B \end{bmatrix} \tag{26}$$

Al expandir la ecuación 26 se obtienen las ecuaciones de velocidad del robot referenciadas al marco inercial fijo.

$$\begin{aligned} \dot{x} &= v \, cos\theta \\ \dot{y} &= v \, sen\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{27}$$

En la ecuación 27 se puede hacer la sustitución $v = r\,\dot{\phi}$ dando como resultado

$$\begin{aligned} \dot{x} &= r\,\dot{\phi}\,cos\theta \\ \dot{y} &= r\,\dot{\phi}\,sen\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{28}$$

Al analizar cada llanta individual del robot diferencial podemos observar que la velocidad que las define es

$$v_L = \begin{bmatrix} r\,\dot{\phi}_L \\ 0 \end{bmatrix}, v_R = \begin{bmatrix} r\,\dot{\phi}_R \\ 0 \end{bmatrix} \tag{29}$$

En donde observamos que los parámetros que podemos controlar son $\dot{\phi}_L\, y\, \dot{\phi}_R$. Sin embargo hay que encontrar un mapeo entre estas velocidades angulares y el modelo del robot completo.

Para ello se toma el movimiento del robot solo moviéndose la rueda derecha con lo que se obtienen la siguiente velocidad angular y velocidad lineal:

$$\begin{aligned} \omega_R &= \tfrac{v_R}{2\,l} = \tfrac{r\,\dot{\theta}_R}{2\,l} \\ l\,\omega_R &= \tfrac{r\,\dot{\theta}_R}{2} \end{aligned} \tag{30}$$

Para la llanta izquierda se obtiene

$$\begin{aligned} \omega_L &= -\tfrac{v_L}{2\,l} = -\tfrac{r\,\dot{\theta}_L}{2\,l} \\ |l\,\omega_L| &= \tfrac{r\,\dot{\theta}_L}{2} \end{aligned} \tag{31}$$

Con las ecuaciones 30 y 31 podemos obtener las siguientes relaciones

$$\begin{aligned} v &= \tfrac{1}{2} r(\dot{\phi}_R + \dot{\phi}_L) \\ \omega &= \tfrac{1}{2} r(\dot{\phi}_R - \dot{\phi}_L) \end{aligned} \tag{32}$$

Al sustituir la ecuación 32 en la ecuación 28 se obtiene

$$\begin{aligned} \dot{x} &= \tfrac{1}{2} r(\dot{\phi}_R + \dot{\phi}_L)\,cos\theta \\ \dot{y} &= \tfrac{1}{2} r(\dot{\phi}_R + \dot{\phi}_L)\,sen\theta \\ \dot{\theta} &= \tfrac{1}{2\,l} r(\dot{\phi}_R - \dot{\phi}_L) \end{aligned} \tag{33}$$

Con lo que finalmente podemos obtener las ecuaciones de las velocidades que controlan las llantas del robot diferencial.

$$\dot{\phi}_R = \frac{v + \omega\,l}{r} \tag{34}$$

$$\dot{\phi}_L = \frac{v - \omega\,l}{r} \tag{35}$$

Estas ecuaciones toman en cuenta la geometría del robot l, que es la distancia del centro de masa hacia una de las llantas y r es el radio de cada llanta.

Al analizar estas dos ecuaciones nos damos cuenta que los parámetros que varían son la velocidad lineal v y la velocidad angular ω por lo que se tendrá que diseñar un controlador para cada una de estas dos variables.

La velocidad tangencial corresponde al error de posición, mientras que la velocidad angular corresponde al error de orientación. El objetivo del algoritmo de control punto punto es hacer tender ambos errores a cero.

11.4. Plataforma de pruebas *Robotarium*

Luego de haber ensamblado la parte electrónica con el diseño mecánico y de haber establecido la forma de modelar las ecuaciones de velocidades se probó el funcionamiento de la locomoción tanto del robot diferencial como del robot esférico en la plataforma de pruebas para robótica *swarm* de la Universidad del Valle de Guatemala.

Figura 32: Plataforma de pruebas *Robotarium*

El *Robotarium* tiene dimesiones de $133.5\,x\,93.5$ cm para la mesa y $133.5\,x\,128$ cm para el marco que sostiene la cámara digital. Sin embargo el espacio que la cámara es capaz de detectar es de $95\,x\,70$ cm que se traduce en una resolución de $600\,x\,400$ pixeles.

La representación del robot móvil dentro del *Robotarium* es como se observa en la Figura 33

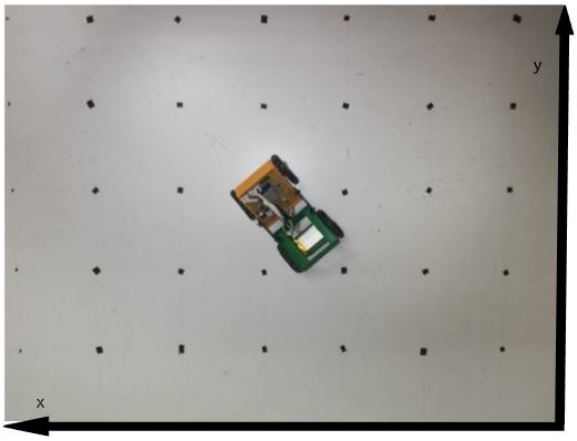


Figura 33: Orientación del robot diferencial dentro del *Robotarium*

CAPÍTULO 12

Algoritmo de control punto punto

12.1. Lazo de control

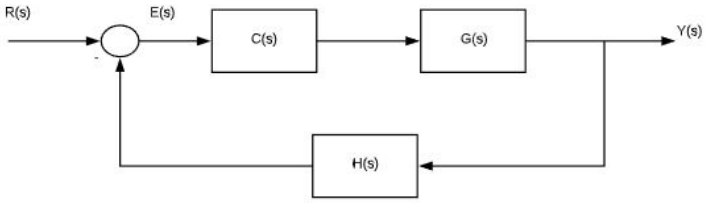


Figura 34: Lazo de control canónico

La Figura 34 muestra el diagrama de bloques de un lazo de control canónico en el dominio de la frecuencia, es decir el diagrama que modela de una forma estandarizada las partes de un sistema LTI con un sistema de control. Para el caso del lazo de control canónico se tiene lo siguiente

- R(s): Señal de referencia que se pretende seguir.
- E(s): Señal de error entre la referencia y la señal de retroalimentación del sensor.
- C(s): Controlador
- G(s): Este bloque representa la planta que se pretende controlar.
- H(s): Representa el sensor que proporciona retroalimentación de la señal de salida del sistema respecto a la referencia.

Para tener una representación más apegada al sistema del robot esférico se diseñó el siguiente diagrama de bloques, en donde:

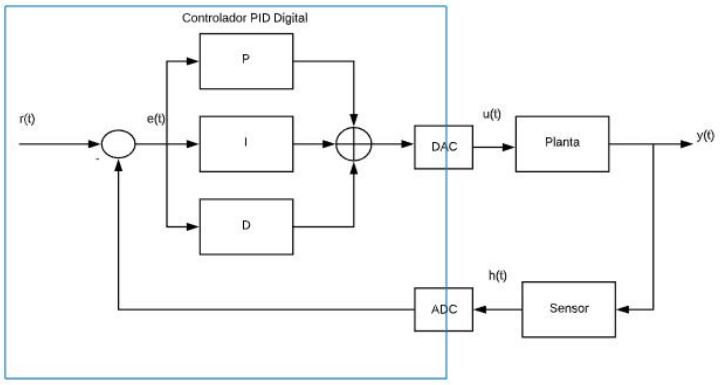


Figura 35: Lazo de control canónico

- r(t): Señal de referencia, es el punto de destino (x, y) que ingresa el usuario en el programa de Python.
- e(t): Señal de error entre el punto actual (señal del sensor) y el punto de referencia (punto de destino)
- Controlador: Controlador PID, o variantes del mismo (P,PI,PD), para llegar al punto de destino (señal de referencia)
- Planta: Robot esférico, el cual se pretende mover hacia un punto de destino (señal de referencia).
- Sensor: Cámara web, es quién proporciona el punto actual del centro de masa del robot esférico.

Cabe destacar que en el diagrama de la Figura 35 lo que se encuentra en el cuadrado azul se desarrollo en el programa de Python junto con la visión de computadora.

12.2. Control PID digital

Debido a que el el sistema de control se debía realizar en Python se procedió a implementar un controlador PID de forma digital.

Para determinar la forma del controlador de forma digital se tomó como base la ecuación del controlador de forma continua.

$$u(t) = K_c \left[e(t) + \frac{1}{\tau_I} \int_0^t e(t^*) dt^* + \tau_D \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right] \tag{36}$$

Los dispositivos digitales no son capaces de replicar señales continuas de forma exacta sino que obtienen una buena aproximación de la misma según el tiempo de muestreo Δt que se tenga de la toma de datos. Por esta razón el tiempo de muestreo se debe de tomar en cuenta para la representación de la integral y derivada de forma discreta.

La parte proporcional (P) del controlador es el error entre la señal de referencia y la señal de retroalimentación del sensor, se define de forma discreta como:

$$e(k\,\Delta t) = e_k \equiv error\,actual \tag{37}$$

Si se la misa forma se define el error anterior como $e((k-1)\Delta t) = e_{k-1}$, podemos definir la derivada de forma discreta como:

$$\dot{e}(t) \approx \frac{e_k - e_{k-1}}{\Delta t} = \frac{e_D}{\Delta t} \tag{38}$$

Con lo que $e_D = e_k - e_{k-1}$.

Para el caso de la integral se tiene el siguiente caso

$$\int_0^t e(\tau)dt \approx E_k\Delta t \equiv \sum_{i=0}^{k} e(i\Delta t)\Delta t + e(k\Delta t)\Delta t = E_{k-1}\Delta t + e_k\Delta t \tag{39}$$

En donde $E_k = E_{k-1} + e_k$ es equivalente a decir que $E_k = E_{k-1} + e_k$

Con estas ecuaciones la forma del controlador PID digital queda de la siguiente forma

$$u(t) \approx u_k = k_p e_k \; + \; k_I E_k \; + \; k_D e_D \tag{40}$$

En este caso las constantes k_p, k_I, k_D absorben el tiempo de muestreo Δt de las ecuaciones anteriores. Finalmente con estas ecuaciones se puede implementar un algoritmo para el controlador PID digital.

12.3. Modelado matemático del sistema de control

Luego de obtener las ecuaciones 34 y 35 es necesario determinar las ecuaciones de velocidad lineal v y velocidad angular w para controlar el movimiento del robot diferencial y poder implementar el algoritmo de control. En la siguiente imagen se muestran los principales parámetros a considerar para encontrar dichas ecuaciones.

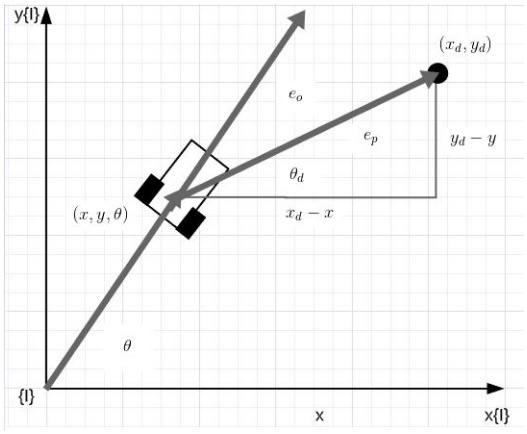


Figura 36: Diagrama del robot diferencial y punto de destino

El error de posición e_p se calcula midiendo la norma de la diferencia entre la posición actual y la posición de destino.

$$e_p = \left\| \begin{bmatrix} x_g \\ y_g \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right\| \tag{41}$$

El ángulo de destino θ_d entre el robot diferencial y el punto de destino se encuentra con la siguiente relación trigonométrica

$$\theta_d = arctan\left(\frac{y_d - y}{x_d - x}\right) \tag{42}$$

Esto sin embargo puede generar problemas con el resultado del ángulo, por lo que en lugar de la función $arctan(x)$ se usa la función $arctan2(x)$.

$$\theta_d = arctan2\left(\frac{y_d - y}{x_d - x}\right) \tag{43}$$

El error de orientación e_o no se puede calcular como la resta de $\theta_d - \theta$ ya que podemos obtener resultados que son múltiplos de π. Por lo que se debe restringir a que este resultado sea únicamente entre $[-\pi, \pi]$, esto se logra de la siguiente forma:

$$e_o = arctan2\left(\frac{sin(\theta_d - \theta)}{cos(\theta_d - \theta)}\right) \tag{44}$$

Una vez establecidas estas ecuaciones podemos implementarlas para el uso de un controlador proporcional P. El cual como su nombre lo indica para la velocidad lineal será proporcional al error de posición y para la velocidad angular al error de orientación.

$$v = k_p\, e_p \tag{45}$$

$$\omega = k_o \, e_o \tag{46}$$

En las ecuaciones 45 y 46 las constantes k_p y k_o son las que se deberán ajustar para lograr el correcto funcionamiento del robot diferencial.

12.4. Conversión de valores de controlador a PWM

En la sección anterior 12.3 se determinaron las ecuaciones y parámetros que definen el sistema de control. Los valores de estas ecuaciones 45, 46 junto con las ecuaciones de la cinemática del robot diferencial 34, 35 determinan el valor de cada llanta del robot diferencial.

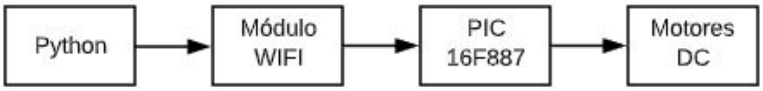


Figura 37: Orden de flujo de información

Sin embargo siguiendo la secuencia de ejecución de información mostrada en la Figura 37 se observa que estos valores no se pueden enviar de forma directa ya que la forma en que el pic controla a los motores DC es mediante sus dos canales PWM, por lo que es necesario hacer una conversión o mapeo de los valores obtenidos en las ecuaciones 34, 35.

Para dicha conversión se usó la siguiente función por partes:

$$PWM_R = \begin{cases} 0 & si \quad v_R < 0 \\ k_R \cdot v_R & si \quad 0 \leq v_R \leq v_{max} \\ v_{max} & si \quad v_R > v_{max} \end{cases} \tag{47}$$

En donde:

- PWM_R: es el valor de ciclo de trabajo que se envía a los motores DC.
- v_R: es el resultado de la ecuación para la velocidad derecha, ecuación 34.
- k_R: es una constante arbitraria para corregir errores mecánicos por pérdida de torque o errores ajenos al resultado de las ecuaciones del sistema de control.
- v_{max}: es el valor máximo de velocidad que puede alcanzar el controlador, con este valor se puede hacer la conversión de velocidad a ciclo de trabajo (PWM). En este punto el valor del ciclo de trabajo del canal PWM es el 100 %

Para el canal PWM que controla el motor DC de la llanta izquierda se siguió el mismo procedimiento que en la ecuación 47. Debido que se dispone de una constante arbitraria k_L esta no necesariamente tiene que tener el mismo valor que la constante k_R. El valor de v_{max} es el mismo para ambas ecuaciones.

12.5. Sistema de control

Para implementar el sistema de control al robot esférico se disponen de las ecuaciones a las que se debe realizar el controlador de velocidad (45, 46). Para estas ecuaciones los errores de posición y orientación se pueden calcular con el desarrollo matemático previamente realizado. Sin embargo no se tiene información de como obtener de forma analítica el valor de las constantes de estas ecuaciones (k_p, k_o). Esto se debe a que se carece de información de la planta como tal.

Debido a esto se procedió a determinar el valor de las constantes de forma experimental aprovechando una de las características más importantes de un controlador PID, poder encontrar el valor de las constantes de forma experimental observando el comportamiento en estado estable. Esto sin necesidad de conocer parcial o totalmente las características de la planta.

Antes de realizar el control al robot esférico se decidió realizar el control únicamente al robot diferencial con el objetivo de encontrar las constantes del controlador de un sistema más estable y conocido respecto a la geometría esférica del nuevo robot. Garantizando así una locomoción esperada del robot diferencial y poder ajustar dichas constantes al comportamiento del robot esférico.

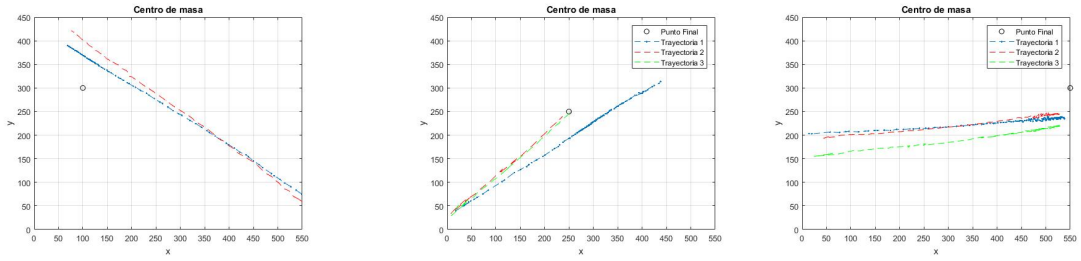
CAPÍTULO 13

Controlador PID a robot diferencial

13.1. Controlador proporcional - P

Se comenzó la implementación del sistema de control únicamente con la parte proporcional en ambos controladores, posición y orientación. Como lo muestran las ecuaciones 45, 46. Debido a lo mencionado anteriormente se comenzó experimentando para encontrar el valor de las constantes k_p, k_o. Se comenzó con valores unitarios para cada una y se fueron incrementando levemente dichos valores hasta obtener resultados significativos para analizar su comportamiento.

Los primeros resultados que se obtuvieron con el controlador P fueron los siguientes:



(a) Controlador P con $k_p = 5.2$, $k_o = 5.7$

(b) Controlador P con $k_p = 5.2$, $k_o = 5.7$

(c) Controlador P con $k_p = 2.5$, $k_o = 2.57$

Figura 38: Trayectorias iniciales con controlador P

En la Figura 38 se puede observar que la trayectoria del centro de masa del robot diferencial es prácticamente recta, esto significa que la componente de velocidad lineal es dominante sobre la componente de velocidad angular. Esto debido a que las cantidades que

opera la componente lineal son valores entre la resolución de la cámara $600\,x\,450$ mientras que los valores de la velocidad angular oscilan entre $[-\pi\,,\,\pi]$, esta diferencia de magnitudes hace que la parte angular (ω) del controlador varíe pero no sea significante en comparación con la parte lineal de la velocidad.

Posterior a esto se aumentó el valor de la constante de velocidad angular con el objetivo de que esta componente fuera más significativa en el resultado final de la velocidad total.

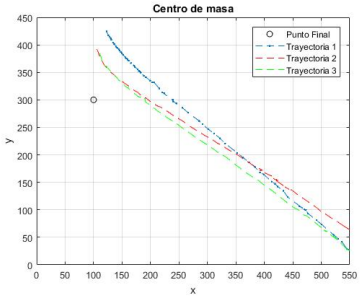
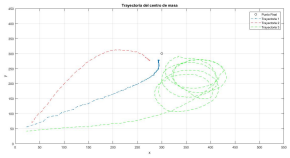
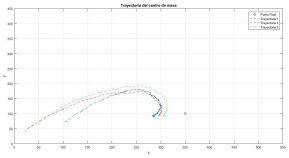


Figura 39: Controlador P con $k_p = 5.2$, $k_o = 15.7$

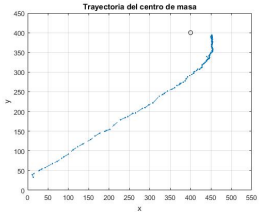
En la Figura 39 se observa que la trayectoria del centro de masa en la parte final ya no es del todo recta, ahora presenta una especie de curva. Sin embargo aún se encuentra muy distante del punto de destino.



(a) Controlador P con $k_p = 2.85$, $k_o = 72.57$



(b) Controlador P con $k_p = 2.85$, $k_o = 72.57$



(c) Controlador P con $k_p = 2.85$, $k_o = 52.57$

Figura 40: Trayectorias iniciales con controlador P, iteración 2

En la Figura 40 se puede observar como al incrementar de forma más significativa el valor de la constante del controlador de error de orientación k_o respecto al del error de posición k_P, la forma de la curva de la trayectoria es mucho más parecida a la esperada por un robot diferencial.

En la Figura 40a se observa que a pesar de dos buenas aproximaciones en las Trayectorias 1 y 2 importa la orientación con la que comienza el robot ya que en la Trayectoria 3 pudo aproximarse en varias ocasiones al punto de destino sin embargo nunca llegó a encontrar estabilidad en el mencionado punto final.

En las Figuras 40b y 40c se observa que las trayectorias son acorde a lo esperadas sin embargo el porcentaje de error respecto al punto de destino es relativamente alto, como lo muestra el Cuadro 7 en donde en la primera columna se muestra la figura a la que hace referencia y el punto de destino. La tercera y cuarta columna muestran la diferencia de

posicionamiento en ambos ejes.

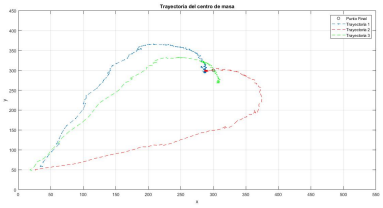
La quinta columna muestra el porcentaje de error entre la magnitud del vector resultante del punto de destino y la magnitud del vector resultante de la última posición que obtuvo el robot diferencial.

Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
40a (300,300)	1	6.07	21.99	4.64
	2	25.14	22.22	7.89
	3	3.15	143.19	20.87
40b (350,100)	1	65.13	9.13	17.85
	2	53.88	5.06	14.57
	3	45.50	6.00	12.45
40c (400,400)	1	52.50	7.89	5.86

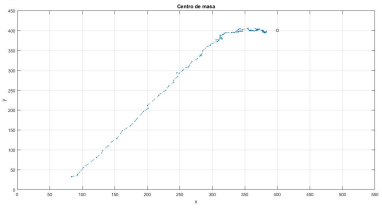
Cuadro 7: Controlador P error de posición, iteración 2

En el Cuadro 7 se observan las medidas que se tomarán para determinar la validez del sistema de control completo (posición más orientación). Esto debido a que, si bien la orientación es importante para determinar los valores de las velocidades de las llantas tanto el modelo cinemático del robot diferencial 11.3 como el modelo matemático del controlador 12.3 y la visión de computadora solo toman en cuenta como referencia el centro de masa del robot móvil. Por lo que el controlador buscará que el centro de masa llegue a al punto final sin importar la orientación *local* del robot.

A pesar de los problemas planteados en la Figura 40 el comportamiento de las trayectorias se acerca al esperado por lo que ajustando levemente la constante del controlador de posición se obtuvieron los mejores resultados para el controlador P mostrados en la siguiente Figura.



(a) Controlador P con $k_p = 2.85$, $k_o = 52.57$



(b) Controlador P con $k_p = 2.85$, $k_o = 72.57$

Figura 41: Trayectorias iniciales con controlador P, iteración 3

Figura P final	Trayectoria	Error en x	Error en y	% Error resultante
41a (300,300)	1	13.24	5.96	3.19
	2	13.50	2.00	2.56
	3	7.07	33.93	4.23
41b (400,400)	1	18.20	4.61	2.84

Cuadro 8: Controlador P error de posición, iteración 3

En el Cuadro 8 se puede observar que a pesar de obtener resultados de porcentaje de error ligeramente menores a los de la segunda iteración se observa que solo el valor del porcentaje no debe ser la única medida de validez del controlador porque el error es relativamente pequeño; sin embargo, al analizar los valores de la diferencia de los ejes (x, y) el error pareciera ser más significativo.

13.2. Casos particulares con el controlador P

13.2.1. Velocidad lineal normalizada

Uno de los problemas más frecuentes al realizar las primeras iteraciones fue notar que en la mayoría de ocasiones cuando el error disminuía debido a que el robot se acercaba al punto de destino, el valor de velocidad del controlador (ecuaciones 34, 35) disminuía al punto de que este valor mapeado al PWM del pic era lo suficientemente pequeño como para no permitir que los motores tuvieran el torque necesario para seguir avanzando.

Esto generaba que el robot en la mayoría de casos no pudiera llegar al punto de destino sino solo aproximarse. Este inconveniente se podía solucionar aumentando la constante de velocidad proporcional al error k_p sin embargo al hacer esto se volvía al problema inicial en donde al robot le costaba hacer giros en la trayectoria.

También se observó que en algunos casos el robot pasaba por el punto de destino sin embargo debido a la velocidad que tenía en ese momento no era capaz de frenar a tiempo. problema que se detallará mejor en la sección 13.2.2

Para solucionar este problema se hizo que la constante del controlador de velocidad lineal k_p fuera función del error de posición $K(\|e\|)$

$$K = \frac{v_o\left(1 - e^{\left(-\alpha\|e\|^2\right)}\right)}{\|e\|} \tag{48}$$

El controlador quedaría de la siguiente forma:

$$\begin{gathered} u = K(\|e\|) \cdot (r - y) \\ u = K(\|e\|) \cdot e \end{gathered} \tag{49}$$

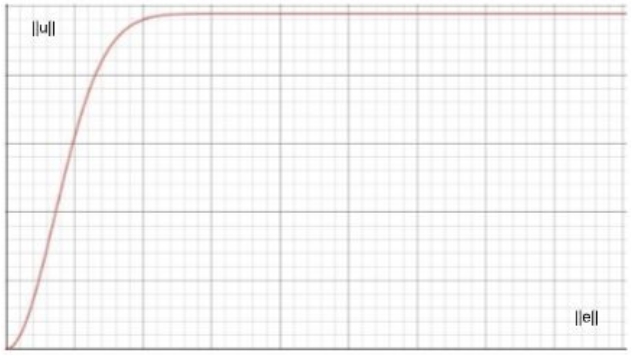
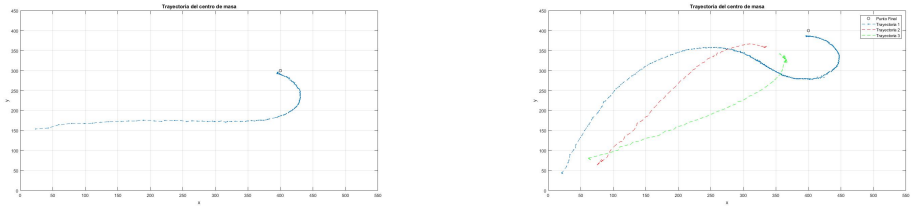


Figura 42: Forma de gráfica de controlador

La ecuación 48 muestra como la constante K es función del error. Así mismo como antes se elegía de forma arbitraria el valor de la constante k_p ahora los valores arbitrarios son $v_o > 0$ y $\alpha > 0$. Esto genera la respuesta que se observa en la Figura 42 en donde el eje x representa la norma del error y el y representa la norma del valor del controlador. La constante v_o controla la amplitud de la señal, en este caso la velocidad constante que alcanza el robot y α controla que tan rápido la velocidad alcanza la amplitud.

13.2.2. Tiempo de desfase

Otro problema que se observó en los primeros casos fue la trayectoria que tomaban ciertas iteraciones, como la Figura 43a y principalmente en la Trayectoria 1 de la Figura 43b se observa como para llegar al punto de destino la trayectoria muestra una curvatura un poco más pronunciada de la habitual. Esto más que la posible falta en ajustar las constantes de los controladores (k_p, k_o) puede significar que el sistema reacciona tarde entre la señal que muestrea (centro de masa de la posición del robot) y la señal que envía (valores de velocidad de ambas llantas). Lo que se traduce como que el sistema presenta un tiempo de *delay* o *lag*.



(a) Controlador P con $k_p = 2.85$, $k_o = 67.57$

(b) Controlador P con $k_p = 1.85$, $k_o = 72.57$

Figura 43: Trayectorias iniciales con controlador P, caso con delay

Al tratarse de un proceso de comunicación complejo que implica el muestreo de la señal de la cámara, la recepción y proceso de múltiples datos en la computadora, el envío de datos mediante WIFI y la posterior comunicación hacia el módulo WIFI y del pic hacia los motores DC es complicado determinar una única fuente dominante de *delay*. Por lo que se debe tomar la suma de todos estos tiempos de retardo como uno solo que afecta a todo el

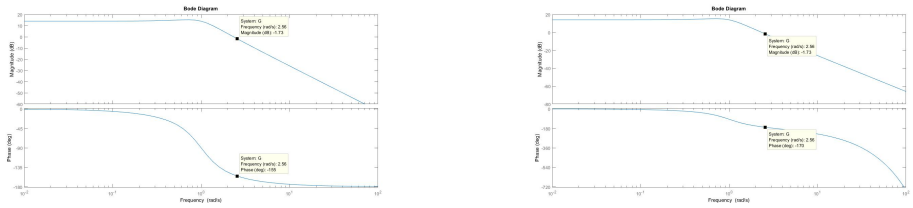
sistema.

Para encontrar una representación del tiempo de retardo se sabe que el *delay* en dominio del tiempo representa una traslación de la señal, lo que se puede modelar como $g(t-\tau)$. Expresión que al transformarla en frecuencia se obtiene:

$$G(s) = e^{-\tau s} \Rightarrow G(jw) = e^{-j\omega\tau} = 1\angle -\omega\tau \tag{50}$$

En la expresión anterior se puede notar que el tiempo de *delay* no modifica la magnitud de la señal solo modifica la fase de la misma.

$$G(s) = k\frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2} \qquad ; G(s) = k\frac{w_n^2}{s^2 + 2\xi w_n s + w_n^2}\, e^{-\tau s} \tag{51}$$



(a) Sistema normal (b) Sistema con desfase

Figura 44: Diagrama de bode de sistema de segundo orden

Tomando como ejemplo un sistema estándar de segundo orden como la función de transferencia del lado izquierdo de la ecuación 51 se obtiene la respuesta en frecuencia de la Figura 44a. Mientras que el mismo sistema con un *delay* de $\tau = 0.1$ obtiene la respuesta de la Figura 44b. Lo importante de esta comparación se encuentra en la gráfica de fase y su interpretación ya que la magnitud como se había visto antes no se modifica por el *delay* del sistema.

Tomando un punto de referencia en el eje de la frecuencia que coincida con una magnitud aproximada de cero dB se tiene un valor de fase mayor para el sistema normal respecto al sistema con *delay*. Además de observar como el *delay* modifica el comportamiento de la curva de fase.

Al analizar los valores de fase con el criterio de estabilidad de Nyquist se tiene un menor margen de fase (MP) para el sistema con *delay* respecto al sistema normal. Lo que significa que un sistema con *delay* tiende a tener un margen menor de estabilidad y si el *delay* es lo suficientemente significativo puede volver al sistema inestable. Esto se debe a la relación que existe entre el margen de fase y el amortiguamiento de un sistema para un sistema de fase mínima.

Debido que el *delay* es síntoma de un sistema que no es de fase mínima se debe tener cuidado con la interpretación del diagrama de bode. Para una mejor conclusión sobre estabilidad se tendría al observar el diagrama de Nyquist. Sin embargo este ejemplo muestra la consecuencia que presenta un sistema con *delay*.

13.2.3. Formas de evitar *Lag*

Para evitar el *delay* de un sistema sea significativo se pueden implementar varias soluciones, dependiendo de la que mejor se adapte al sistema en uso. A continuación se detallan algunas de ellas.

Aproximación de Padé

Una de las principales complicaciones con el *delay* es que no se puede manipular de forma algebraica como los demás términos de una función de transferencia. Por lo que una solución es implementar la aproximación de Padé, la cual se basa en la expansión para la función exponencial de la siguiente forma:

$$e^{-\tau s} = \frac{1 + \sum_{i=1}^{n} \frac{i!(-\tau s)^i}{(2i)!}}{1 + \sum_{i=1}^{n} \frac{i!(\tau s)^i}{(2i)!}} \tag{52}$$

Para sintonización de sistemas de control es frecuente usar una aproximación de primer o segundo orden:

$$e^{-\tau s} = \frac{2 - \tau s}{2 + \tau s} \qquad ; e^{-\tau s} = \frac{\tau^2 s^2 - 8\tau s + 16}{\tau^2 s^2 + 8\tau s + 16} \tag{53}$$

Estas funciones como se puede observar agregan la misma cantidad de ceros y polos según el orden de la aproximación. Esta aproximación del *delay* permite poder manipular la expresión de forma algebraica.

Compensador PD

Un compensador PD tiene la siguiente forma:

$$K\, D(s) = K(1 + T_D\, s) \tag{54}$$

El objetivo de este compensador es mejorar el margen de fase incrementando la fase para frecuencias mayores a $\omega \approx 0.1/T_D$. Con esto tiene un efecto estabilizador. Sin embargo tiene el problema de aumentar la magnitud sin límites con lo que se puede amplificar ruido a altas frecuencias.

Compensador *Lead*

Un compensador *Lead* soluciona el problema la ganancia sin límites del compensador PD y también se implementa para mejorar los márgenes de estabilidad, tiene la siguiente forma:

$$K\, \frac{T\, s + 1}{\alpha\, T\, s + 1} \tag{55}$$

Este compensador tiene un cero en $s = -1/T$ y un polo en $s = 1/(\alpha T)$ con un valor de $0 < \alpha < 1$, se suele tomar un valor de $\alpha = 0.1$. Debido a este valor de α el polo se ubica a la izquierda del cero del compensador.

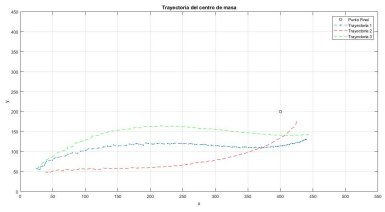
Solución a implementar

Si bien la aproximación de Padé es una forma de encontrar una analogía de forma analítica del *delay* esta necesita conocer el tiempo que la señal se está desfasando τ, valor que en este caso se desconoce por lo que no se implementará esta solución.

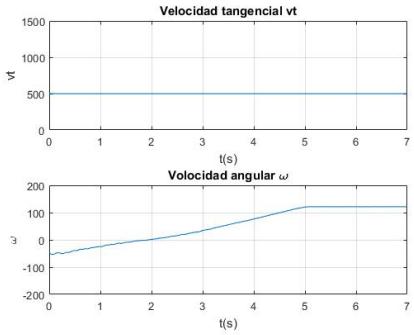
Teóricamente el compensador *Lead* es una mejor forma de solucionar el problema del desfase respecto al compensador PD. Una de las características de un compensador es modificar directamente la dinámica de la planta; sin embargo, en este caso no conocemos a profundidad la dinámica de la misma. A pesar de esto en ambos casos (compensador PD y *Lead*) el margen de mejora de fase depende del ajuste de las constantes K y T esto equivale a ajustar la constante k_D en la parte diferencial del controlador PD o PID.

13.3. Controlador proporcional normalizado - Pn

Al implementar el controlador proporcional normalizado (ecuación 49) se decidió hacerlo solo para el controlador de posición mientras que el controlador de orientación permaneció de la misma forma. Esto debido a que se busca que el robot diferencial tenga una velocidad constante hasta alcanzar el punto de destino mientras que la orientación no se modifique más de lo necesario.



(a) Valores de controladores $V_o = 500$, $\alpha = 0.25$ y $k_o = 67.57$



(b) Gráfica controlador de posición y orientación

Figura 45: Trayectorias controlador Pn

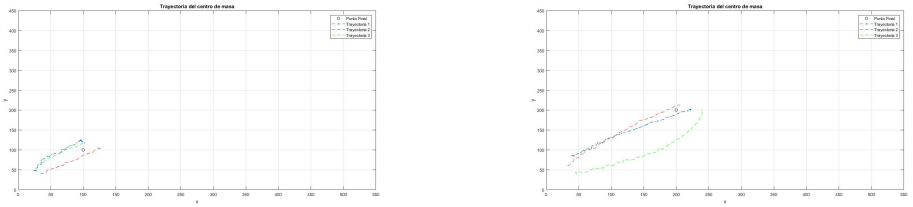
Si bien el objetivo de utilizar el controlador normalizado es reducir lo más posible el error de posición, en la Figura 45a se observa que las trayectorias no se acercan lo esperado al punto de destino. A pesar de esto en la Figura 45b se observa como el controlador de posición alcanza una velocidad constante $V_o = 500$ practicamente desde el inicio de la trayectoria, que es lo esperado por la normalización de la velocidad.

Además de implementar este tipo de controlador normalizado se debe definir una *vecindad* o *frontera* cercana al punto de destino la cual tiene la función de detener al robot móvil cuando se encuentre lo suficientemente cerca del punto de destino. Esto por que en algunas ocasiones puede pasar por el punto final y no detenerse, debido a la velocidad que lleva y al *delay* que presenta el sistema que no le permite reaccionar a tiempo.

Esta *vecindad* se basa en los tres parámetros elegidos para validar el sistema de control:

- Diferencia de posición en x.
- Diferencia de posición en y.
- Porcentaje de error del vector resultante.

Como se observa las diferencias en los ejes x y y forman un cuadrado al rededor del punto final que es la primera condición de error. La segunda condición de error es el porcentaje de error del vector resultante ya que al tratarse de un vector se puede alcanzar la misma magnitud desde dos o más posiciones diferentes. Por lo que ambas condiciones se complementan.



(a) Punto final $(100, 100)$ (b) Punto final $(200, 200)$

Figura 46: Controlador Pn con valores $V_o = 500$, $\alpha = 0.25$ y $k_o = 67.57$

	Trayectoria	**Error en x**	**Error en y**	**% Error resultante**
Mejor Posición	1	13.5	12.5	0.35
	2	7.0	9.5	0.91
	3	10.5	7.5	1.09
Posición final	1	2.00	21.15	10.38
	2	23.50	5.50	14.85
	3	1.50	17.50	9.79

Cuadro 9: Figura 46a, punto final $(100, 100)$

	Trayectoria	Error en x	Error en y	% Error resultante
Mejor Posición	1	7.86	4.66	0.85
	2	6.39	5.71	0.12
	3	29.41	35.95	0.29
Posición Final	1	21.70	0.85	5.77
	2	4.48	13.06	4.41
	3	39.70	0.37	10.47

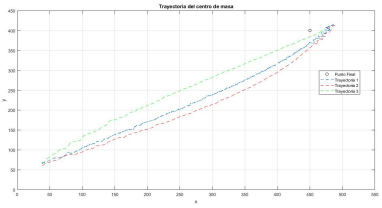
Cuadro 10: Figura 46b, punto final $(200, 200)$

Error en x	Error en y	% Error resultante
7.5	7.5	3.0

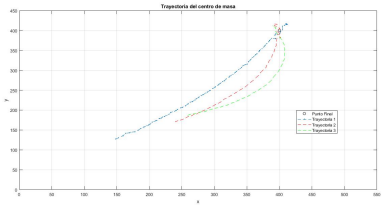
Cuadro 11: Frontera propuesta para error de posición

Los Cuadros 9 y 10 hacen referencia a la Figura 46 en donde se observa la diferencia entre la mejor posición y la posición final que alcanzó el robot diferencial. Como se observa la velocidad normalizada tuvo un efecto positivo en lograr que el robot alcance el punto de destino, sin embargo el robot no es capaz de detenerse en esa posición. Esto se debe al poco rango de error definido por la *frontera* y el *delay* del sistema ya que el robot no responde a tiempo para detenerlo.

Así mismo se puede observar en el Cuadro 9 que la diferencia del error resultante entre la mejor posición y la posición final es significativa. Ya que el robot es capaz de tener una muy buena aproximación, casi todas por debajo del uno por ciento, mientras que para la posición final casi todas se encuentran casi por encima del diez por ciento.



(a) $V_o = 500$, $\alpha = 0.25$ y $k_o = 67.57$. Pf $(450, 400)$



(b) $V_o = 500$, $\alpha = 2.25$ y $k_o = 87.57$. Pf $(400, 400)$

Figura 47: Controlador Pn con mejores resultados

Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
47a (450,400)	1	35.0	14.5	5.96
	2	40.0	14.0	6.54
	3	21.5	5.5	2.11
47b (400,400)	1	10.50	16.50	3.38
	2	7.50	16.19	1.13
	3	10.50	11.50	0.16

Cuadro 12: Errores de posición de la Figura 47

Las trayectorias que se muestran en la Figura 47 muestra resultados de iteraciones que tuvieron mejor comportamiento en alcanzar el punto de destino. Las trayectorias de la Figura 47b muestran una curvatura más habitual al comportamiento de un robot diferencial mientras que las trayectorias de la Figura 47a son más rectas que las esperadas.

Error en x	**Error en y**	**% Error resultante**
15	15	7

Cuadro 13: Segunda frontera propuesta para error

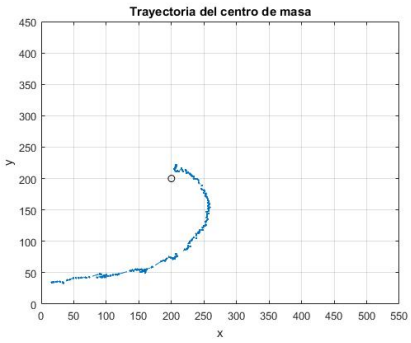
Al analizar los datos del Cuadro 12 se observa que los porcentajes de error del vector resultante disminuyeron respecto a los de las iteraciones anteriores. El uso de las constantes V_o, α y k_o son prácticamente las mismas por lo que el mejor resultado se debe a definir nuevamente la frontera. Para este caso se amplió la zona del rectángulo que delimita la zona de error así como el error del vector resultante. Esto genera que el *freno* del robot reaccione antes de aproximarse lo suficiente al punto de destino y pueda conservar una mejor posición final.

Cabe mencionar que redefinir la frontera implica un error ya que se soluciona el problema tener una buena aproximación respecto al punto de destino, sin embargo el *delay* en el sistema sigue existiendo, por lo que el sistema fácilmente puede convertirse inestable en alguna otra iteración.

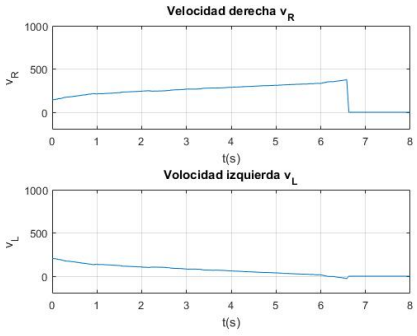
13.4. Controlador PD-n

Luego de haber ajustado el controlador proporcional normalizado (sección 13.3) se siguió a implementar la solución planteada para eliminar el *delay* del sistema. La cual consiste en agregar una parte significativa de control derivativo únicamente al controlador de orientación, dejando el controlador de posición solo con la parte proporcional normalizada.

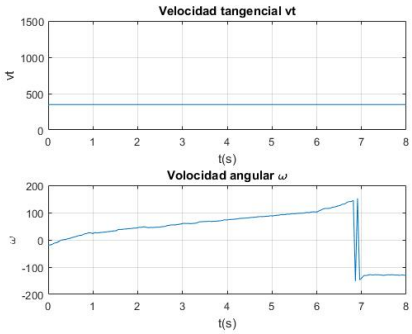
Con lo que se obtuvieron los siguientes resultados:



(a) $V_o = 350$, $\alpha = 0.35$, $kp_o = 47.57$ $kd_o = 0.4757$. Pf $(200, 200)$



(b) Velocidades de llantas derecha e izquierda

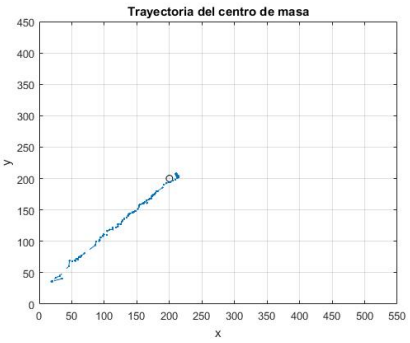


(c) Valores de controlador lineal y de orientación

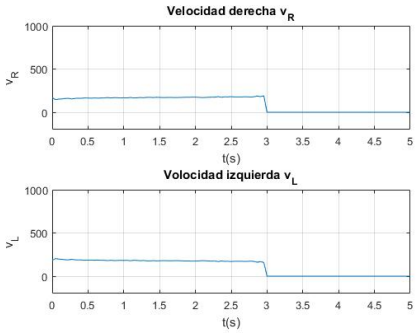
Figura 48: Controlador PD-n iteración 1

La Figura 48a muestra la trayectoria habitual que debería seguir el robot hacia el punto de destino, con esa particular curvatura. Mientras que la Figura 48b muestra los valores del cálculo del controlador para cada una de las llantas. Ambas comienzan con un valor inicial y mientras la llanta derecha v_R incrementa hasta el valor constante de $v_o = 500$, la llanta izquierda disminuye hasta ser prácticamente cero. De esta forma también se explica el giro que realizó el robot diferencial. Ya que a medida que aumentaba la velocidad derecha y disminuía la velocidad izquierda el robot pivotaba para generar esa curva.

En la Figura 48c se observa como el valor de la ecuación de velocidad lineal del controlador es 500 en todo momento lo que concuerda con lo esperado por la normalización de dicho controlador. También se observa como cambia la orientación hasta llegar al segundo 6.5 (aproximadamente) en donde el robot frena y aún hace un pequeño giro sobre su centro de masa cambiando bruscamente la orientación más no la posición del robot diferencial.

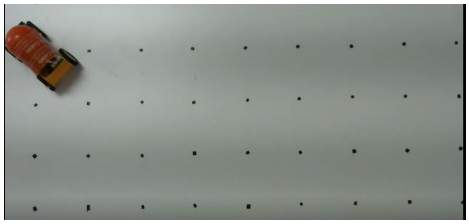


(a) $V_o = 350$, $\alpha = 0.35$, $kp_o = 47.57$ y $kd_o = 5.4757$. Pf $(200, 200)$

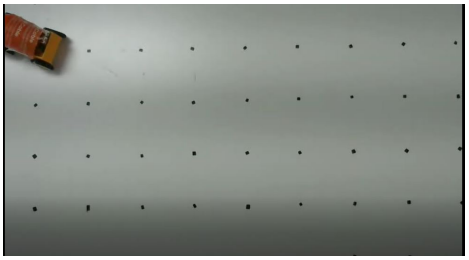


(b) Velocidades de llantas derecha e izquierda

Figura 49: Controlador PD-n iteración 2



(a) Orientación inicial Figura 49



(b) Orientación inicial Figura 48

Figura 50: Orientaciones iniciales de las iteraciones 1 y 2

La Figura 49a muestra una trayectoria más recta respecto a la mostrada en la Figura 48a, esto también se debe a la posición inicial del robot diferencial ya que en este caso el centro de masa estaba mejor direccionado hacia el punto de destino, como se observa en la Figura 50.

En la Figura 49b se observa como la velocidad en ambas llantas es constante lo que coincide con la trayectoria que tomó el robot. Ya que al estar mejor orientado no necesitó asignar más velocidad a una de las dos llantas, en ambas la velocidad fue constante.

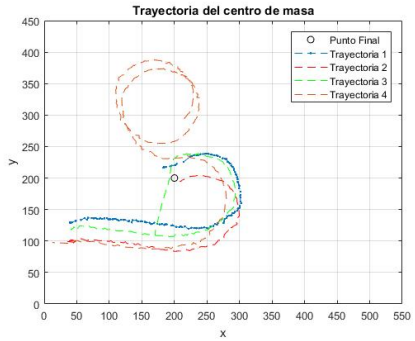
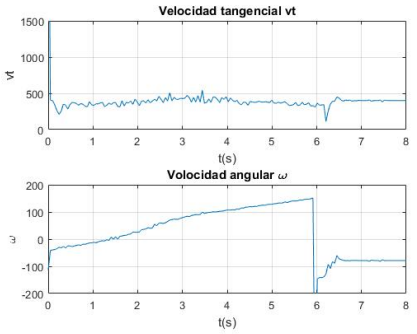
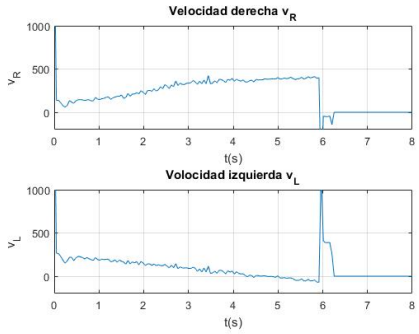


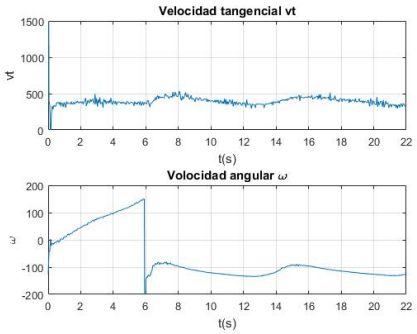
Figura 51: $V_o = 400$, $\alpha = 0.35$, $kp_o = 47.57$ y $kd_o = 85.47$. Pf $(200, 200)$



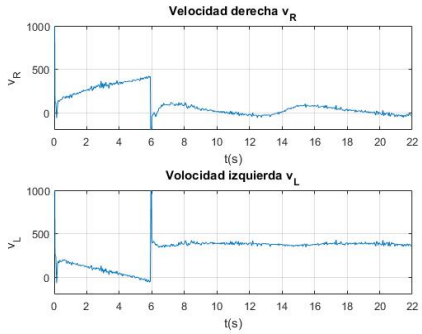
(a) Valores de controladores lineales y de orientación T1



(b) Velocidades de llantas derecha e izquierda T1



(c) Valores de controladores lineales y de orientación T4



(d) Velocidades de llantas derecha e izquierda T4

Figura 52: Controlador PD-n iteración 3

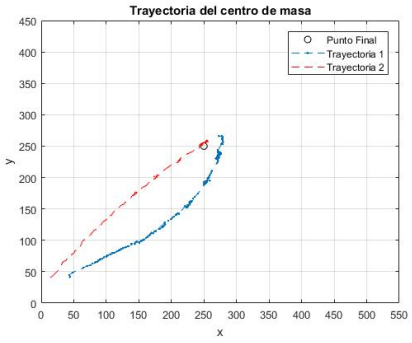
Como se observó en la Figura 50 la orientación inicial del robot es un factor para la trayectoria que sigue hacia el punto de destino. En la Figura 51 la posición inicial del robot diferencial fue paralela al punto de destino y a pesar de usar los mismos valores para las constantes de los controladores se observa que solo en dos de las cuatro trayectorias (T1, T2) el robot se detiene en un punto cercano al punto de destino.

La T3 deja en evidencia el problema de *delay* en el sistema ya que a pesar de lograr una buena aproximación al punto de destino el robot no es capaz de detenerse en este punto y sigue la trayectoria. Con la T4 ocurre algo similar ya que al encontrarse relativamente cerca del punto de destino (posición en que finalizó la T1) el controlador reacciona de forma errónea y dirige al robot a otra dirección.

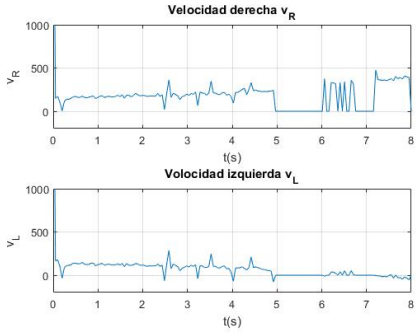
La Figura 52a muestra el resultado del controlador de posición vt, en este caso la velocidad no tiene el valor constante esperado de 500 sin embargo el valor medio es lo suficiente *constante* para que el robot tenga velocidad constante ya que estas variaciones no se perciben al observarlo durante la prueba. En la Figura 52b se muestra como después del segundo 6 el valor de las velocidades es cero ya que el robot se encuentra dentro de la frontera establecida.

Los resultados de la T4 se observan en las Figuras 52c y 52d en donde se observa un cambio brusco de orientación en el segundo 6 lo que concuerda con el giro que toma el robot. Esto también se puede observar con las velocidades de las llantas, ya que la velocidad izquierda luego de disminuir a cero tiene un pico que le hace terminar con velocidad constante

y sumado a que la llanta derecha se mantiene con valores pequeños el robot termina oscilando como se muestra en la gráfica de su trayectoria.



(a) $V_o = 350$, $\alpha = 0.35$, $kp_o = 47.57$ y $kd_o = 85.47$. Pf $(250, 250)$



(b) Velocidades de llantas derecha e izquierda T1

Figura 53: Controlador PD-n iteración 4

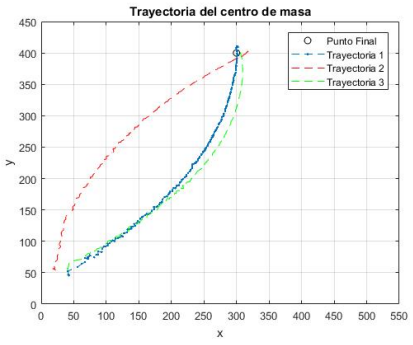
Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
53a (250,250)	1	23.00	16.00	7.81
	2	6.28	6.89	2.63

Cuadro 14: Errores de posición de la Figura 53a

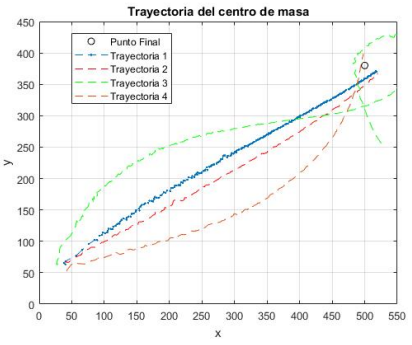
El Cuadro 14 hace referencia a la Figura 53 en dónde se observa que las trayectorias tuvieron buena aproximación al punto de destino. En la Figura 53b se observa como a pesar de alcanzar el punto de destino la T1 hace un pequeño giro sobre su propio eje producto de que la velocidad de la llanta derecha no permaneció en cero, esto sin embargo no afectó el resultado de su posición final.

13.4.1. Controlador PD(nvt)

Al inicio de la sección 13.3 se planteó mantener el controlador de posición solo con la parte proporcional normalizada, sin embargo se analizaron unos casos en los que se aplicó la parte derivativa a ambos controladores. Formando así un controlador PD *completo*



(a) $V_o = 400$, $\alpha = 0.35$, $kd_{vt} = 47.57$, $kp_o = 22.85$ y $kd_o = 85.47$. Pf $(300, 400)$



(b) $V_o = 400$, $\alpha = 0.35$, $kd_{vt} = 47.57$, $kp_o = 22.85$ y $kd_o = 85.47$. Pf $(500, 380)$

Figura 54: Controlador PD-nvt

Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
54a (300,400)	1	2.00	10.43	1.91
	2	17.23	2.09	2.43
	3	5.99	1.13	0.90
54b (500,380)	1	16.70	8.91	1.29
	2	19.55	13.45	1.25
	3	25.68	125.11	6.97
	4	2.21	18.12	1.49

Cuadro 15: Errores de posición de la Figura 54

Para estos resultados se observa en las etiquetas de las Figuras 54a, 54b que se mantuvieron los valores del controlador proporcional de posición. Para el controlador de orientación el peso de la parte derivativa sigue siendo mayor al de la parte proporcional, mientras que para la parte derivativa del controlador de posición (kd_{vt}) se tuvo un valor medio respecto a los valores del controlador de orientación.

Con estos valores de constantes de los controladores los valores de error de posición y de vector resultante no crecieron o disminuyeron significativamente como se muestra en el Cuadro 15. Sin embargo al aumentar más la parte derivativa del controlador de posición (kd_{vt}) la trayectoria del robot era prácticamente en círculos con lo que el sistema se volvía inestable y el error aumentaba significativamente.

13.5. Controlador PID-n

Para completar el controlador se mantuvo la condición planteada al inicio de la sección 13.3 con lo que el controlador de posición permaneció solo con la parte proporcional (Pn) al inicio, mientras que al controlador de orientación se agregó la parte integral formando así un controlador PID.

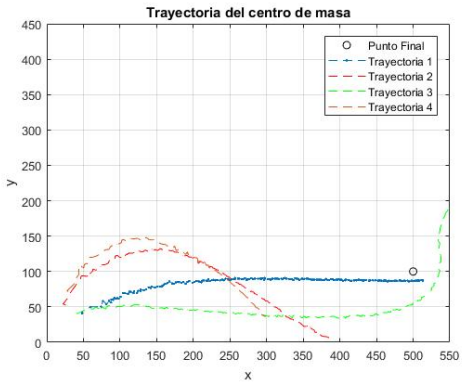


Figura 55: $V_o = 400$, $\alpha = 0.35$, $kd_{vt} = 47.57$, $kp_o = 22.85$, $kd_o = 85.47$ y $kI_o = 2.57$. Pf $(500, 100)$

Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
	1	12.73	12.86	1.99
55	2	109.00	94.50	23.31
(500,100)	3	91.50	113.48	23.33
	4	201.34	63.78	40.98

Cuadro 16: Errores de posición de la Figura 55

Al comenzar a realizar iteraciones del controlador PID se observó que la parte integral al ser la suma de los errores anteriores produce el efecto de saturar leve o fuertemente a los motores DC según sea el valor de la constante integral kI_o, a este problema se le conoce como *windup*. Ya que hace que el valor del controlador w sea cada vez mayor. Esto se observa en las trayectorias de la Figura 55, teniendo en cuenta que el valor de la constante integral es el valor más pequeño de las constantes usadas y el robot solo logró llegar en una ocasión al punto de destino. Siendo estas de las mejores trayectorias logradas con este tipo de controlador. Como se observa los casos de las trayectorias 2 y 4 el robot se sale del área de trabajo haciendo imposible que siga una trayectoria hacia el punto de destino.

Además si esta constante se seguía incrementando (kI_o) la trayectoria del robot era recta o alcanzaba una velocidad a la que no era capaz de realizar cambios de orientación. A esto se suma al problema del *delay* que el sistema sigue presentando.

13.5.1. Técnica anti wind-up: Back calculation

El problema de *windup* es común al introducir la parte integral a un controlador. Existen algunas técnicas para evitarlo, la idea básica es evitar la saturación del actuador. Las ténicas más comúnes son las siguientes:

- Clamping
- Back calculation

La técnica seleccionada fue *Back Calculation* ya que se puede implementar de forma digital, estableciendo límites mínimos y máximos a la salida del controlador para evitar la saturación del actuador. Al aplicar esta técnica se modifica el diagrama de bloques canónico quedando de la siguiente forma:

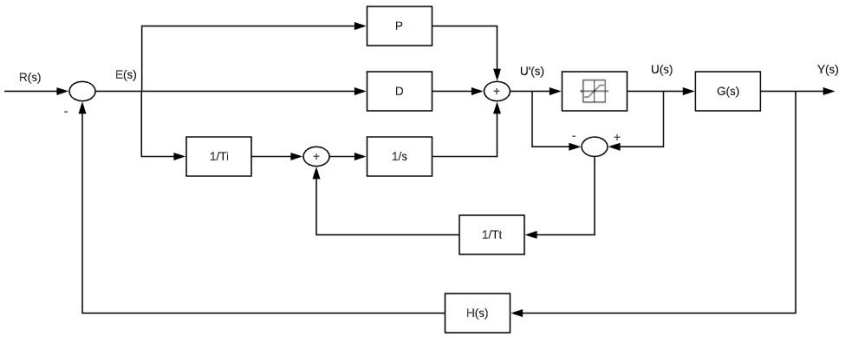


Figura 56: Diagrama de bloques con antiwindup

En la Figura 56 se observa un bloque a la salida del controlador que hace referencia a la saturación del actuador. Se realiza una comparación entre la salida del controlador y el valor que satura al actuador, este valor se multiplica por una constante T_t como efecto para reducir la magnitud del término integral. En esta configuración el término T_t es el mismo que T_i.

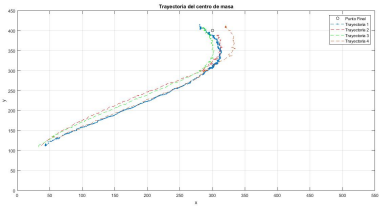
La condición de comparación a la salida del controlador es la siguiente:

$$u = \begin{cases} u_{min} & si \quad u \leq u_{min} \\ u & si \quad u_{min} < u < u_{max} \\ u_{max} & si \quad u \geq u_{max} \end{cases} \tag{56}$$

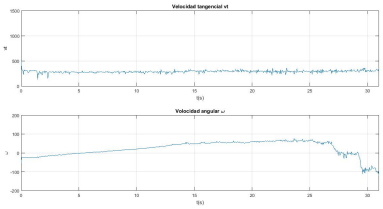
En dónde u_{min} y u_{max} son los límites inferior y superior para la saturación del actuador.

13.6. Resultados controlador PID-n con anti-windup

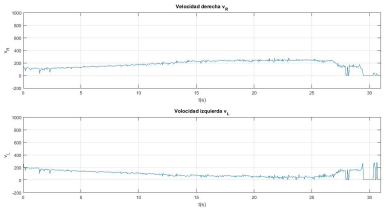
El objetivo de arreglar el problema de saturación del actuador, *Wind up*, es que hasta el momento con la parte proporcional (Pn) se tiene una buena aproximación hacia el punto de destino, la parte derivativa (D) corrige en cierta medida el problema del *delay*. Mientras que la parte integral (I) es la encargada de eliminar el error en estado estacionario y por lo tanto permitir el rastreo de una señal.



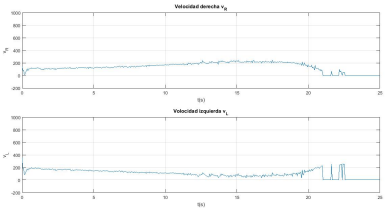
(a) Trayectorias con controlador PID



(b) Valores de controladores lineales y de orientación T1



(c) Velocidades de llantas derecha e izquierda T1



(d) Velocidades de llantas derecha e izquierda T3

Figura 57: Controlador PD-n iteración 1

Los valores del controlador PID-n usados fueron:

Controlador posición				Controlador orientación		
V_o	α	kd_{vt}	ki_{vt}	kp_w	kd_w	ki_{vt}
300	0.35	22.85	3000	67.57	85.57	600.5

Cuadro 17: Valores de controlador PID, iteración 1

Los errores de posición y vector resultante fueron:

Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
57a (300,400)	1	19.41	15.20	0.22
	2	7.99	10.99	0.77
	3	15.01	6.88	0.65
	4	19.54	8.20	3.68

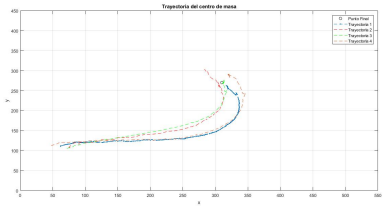
Cuadro 18: Errores de posición de la Figura 57a

En la Figura 57a se observan las trayectorias de la prueba con controlador PID hacia el punto de destino $(300, 400)$. Si bien las trayectorias son cercanas al punto de destino, en el Cuadro 18 observa que la diferencia en x y y es relativamente elevada respecto a la última columna del porcentaje de error del vector resultante.

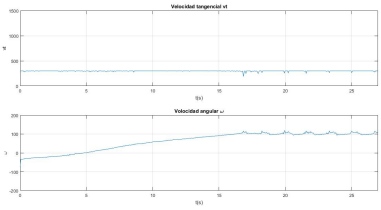
Además en la Figura 57b se observa como el controlador de posición v_t tienen variaciones al rededor del valor 300 producto de la acción integral y derivativa que modifican ligeramente

el valor de la parte proporcional normalizada. También se observa como en la última parte el controlador de orientación reacciona de forma agresiva para corregir la orientación del robot diferencial, en dónde se muestra una vez más el problema de *delay* que muestra el sistema.

El Cuadro 17 muestra los valores de las constantes usadas para ambos controladores, en dónde se observa que se mantiene la parte derivativa alta con el fin de evitar lo más posible el *delay* y las partes integrales mantienen una relación aproximada de 10 : 1 respecto a la parte proporcional.



(a) Trayectorias controlador PID, iteración 2



(b) Controlador de posición y orientación, iteración 2

Figura 58: Controlador PID, iteración 2

Controlador posición				Controlador orientación		
V_o	α	kd_{vt}	ki_{vt}	kp_w	kd_w	ki_w
325	0.35	22.85	0.0	85.67	85.57	0.2

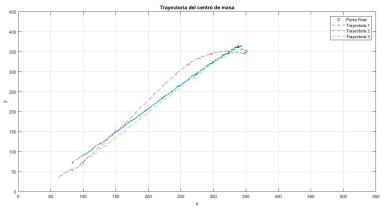
Cuadro 19: Valores de controlador PID, iteración 2

Los errores de posición y vector resultante fueron:

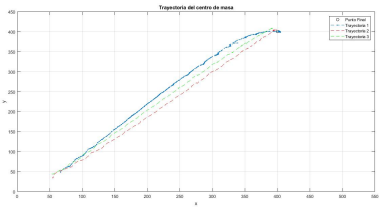
Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
58a (310,270)	1	8.55	9.56	0.09
	2	27.12	32.99	0.83
	3	3.12	5.39	1.43
	4	9.43	19.78	4.91

Cuadro 20: Errores de posición de la Figura 58

En la segunda iteración se probó la eficacia del algoritmo de control sin la parte integral en el controlador de posición como lo muestra el Cuadro 19. El controlador responde de forma similar a los casos mostrados con el controlador PD(nvt) ya que la parte integral es casi nula, manteniendo los valores de las demás constantes. Aún así se muestra como los porcentajes de error son bajos ya que todos tienen valores menores a 5 % .



(a) Trayectorias controlador PID, iteración 3



(b) Trayectorias controlador PID, iteración 4

Figura 59: Controlador PID, iteraciones 3 y 4

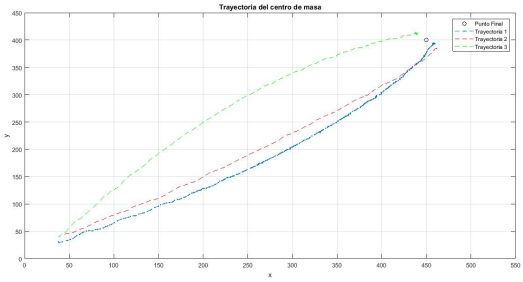
	Controlador posición				Controlador orientación		
	V_o	α	kd_{vt}	ki_{vt}	kp_w	kd_w	ki_w
Fig. 59a	350	0.35	22.85	3000	67.57	45.57	600
Fig. 59b	350	0.35	22.85	3000	87.57	45.57	600

Cuadro 21: Valores de controlador PID, iteraciones 3 y4

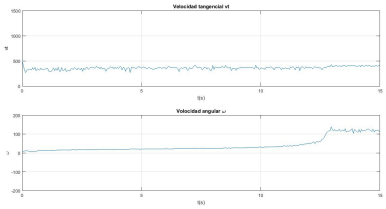
Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
	1	11.48	10.48	2.09
59a	2	2.71	4.81	0.17
(350,350)	3	7.92	6.57	1.17
	1	3.36	0.53	0.14
59b	2	5.81	3.70	0.25
(400,400)	3	8.99	7.63	0.35

Cuadro 22: Errores de posición de la Figura 59

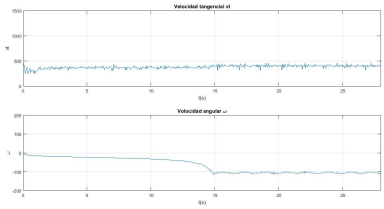
Para la tercera y cuarta iteración (59) se retomaron los valores de los controladores que se usaron en la primera iteración, ya que estos fueron los valores con los que se obtuvieron mejores resultados. Únicamente se varió el valor de Kp_w para lograr que la orientación se alcance de forma más rápida. Los resultados de error como se muestra en el Cuadro 59 son cercanos al punto, sin embargo al no poder definir una frontera más cercana al punto de destino por el *delay* del sistema el error en estado estacionario no se elimina por completo.



(a) Trayectorias controlador PID, iteración 5



(b) Valores de controladores, trayectoria 1



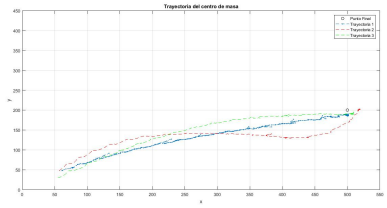
(c) Valores de controladores, trayectoria 2

Figura 60: Controlador PID, iteraciones 5

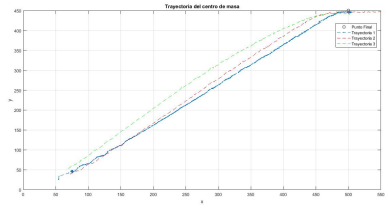
Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
	1	7.50	6.73	0.05
60a	2	11.16	15.03	0.21
(450,400)	3	11.64	13.22	0.24

Cuadro 23: Errores de posición de la Figura 60a

En la Figura 60a se observan las trayectorias de la iteración 5 hacia el punto de destino $(450, 400)$. En las Figuras 60b y 60c se observa como el controlador de posición mantiene un valor prácticamente constante mientras que el de orientación cambia dependiendo de la trayectoria que debía tomar ya que el resultado de sus valores es opuesto.



(a) Trayectorias controlador PID, iteración 6



(b) Trayectorias controlador PID, iteración 7

Figura 61: Controlador PID, iteraciones 6 y 7

Figura P final	Trayectoria	Error en x	Error en y	% Error resultante
61a (500,200)	1	1.51	10.38	0.43
	2	16.70	1.78	3.01
	3	4.34	9.92	0.83
61b (500,450)	1	3.50	3.00	0.0907
	2	0.50	4.00	0.45
	3	1.00	6.91	0.57

Cuadro 24: Errores de posición de la Figura 61

Para las últimas iteraciones (61) se analizaron casos con trayectorias largas ya que partiendo desde el origen hacia puntos cercanos el controlador demostró ser errático mientras que para este tipo de trazos se obtuvieron mejores resultados. Las variables de error se mantienen en lo mostrado en los casos anteriores (24) tendiendo en cuenta que en la Figura 61b para la Trayectoria 2 se tomo en cuenta la mejor posición que tuvo el robot y no la final ya que se sigue mostrando el problema del *delay*, algo que también se aprecia en menor medida en la Trayectoria 2 de la Figura 61a.

13.7. Comparación de trayectorias

Si bien el objetivo del algoritmo de control punto punto es precisamente lograr que el robot móvil alcance un punto de destino existen varias formas de lograr esta tarea. Se puede realizar una trayectoria completa entre los dos puntos (inicial y final) o se puede dividir esta trayectoria en varios puntos y lograr así una trayectoria compuesta por trayectorias más cortas.

Por lo que se pretende determinar con que tipo de trayectoria responde mejor el robot diferencial, si haciendo un recorrido como única trayectoria o siguiendo una secuencia de puntos.

13.7.1. Controlador Pn

Paso de 100

Para el primer caso se dividió la trayectoria en pasos de 100 en ambos ejes. Comenzando en el punto $(100, 100)$ y terminando en el punto $(400, 400)$.

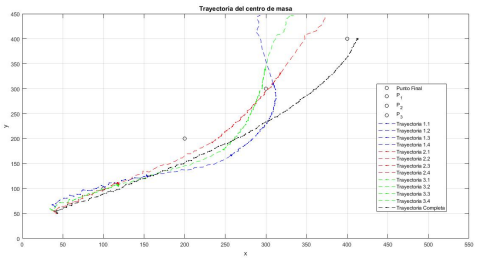


Figura 62: $V_o = 500$, $\alpha = 0.25$, $kp_o = 67.57$. Pf $(400, 400)$ Paso $= 100$

Trayectoria	**Sección**	**Error en x**	**Error en y**	**% Error resultante**
T1	1.1	54.00	25.00	40.25
	1.2	57.52	34.08	8.31
	1.3	8.83	8.98	2.97
	1.4	107.50	47.00	5.57
T2	2.1	17.00	12.00	14.53
	2.2	33.88	7.59	7.08
	2.3	17.83	30.89	8.14
	2.4	27.00	41.00	2.10
T3	3.1	17.67	6.15	12.06
	3.2	48.02	22.03	
	3.3	1.02	38.41	6.43
	3.4	66.00	47.00	1.35

Cuadro 25: Errores de posición de la Figura 62

En la Figura 62 se observa que el comportamiento por trayectorias cortas no tuvo el resultado esperado ya que ninguna trayectoria se acerca al punto de destino. Sin embargo se puede observar que para el punto $(100, 100)$ todas las trayectorias son cercanas a este punto aunque su posición final sea diferente.

A partir de este punto cada trayectoria si bien realiza la curvatura esperada por el modelo diferencial esto generaba que el robot cambiara un poco su orientación con lo que para cada siguiente trayectoria era más complicado el giro que tenía que realizar. Como el caso de la trayectoria 1 que al comenzar el tramo del punto $(300, 300)$ la posición del robot era prácticamente perpendicular al punto de destino. Lo mismo ocurrió en menor medida en las trayectorias 1 y 2.

En cuanto al Cuadro 25 ocurre algo interesante ya que el error resultante demuestra que las primeras trayectorias son las que tiene más porcentaje de error siendo las que más se aproximaron al punto de destino. Esto se debe a que al ser cantidades menores el error es más significativo respecto a cantidades de posiciones mayores. Lo que confirma que la mejor forma de evaluar la eficacia del controlador es tomando en cuenta las tres variables que se muestran en los cuadros de resultados.

Paso de 50

Para el segundo caso se dividió la trayectoria en pasos de 50 en ambos ejes. Comenzando en el punto $(100, 100)$ y terminando en el punto $(400, 400)$.

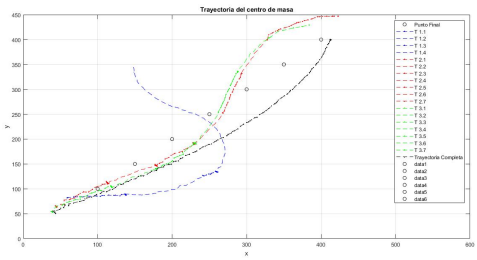


Figura 63: $V_o = 500$, $\alpha = 0.25$, $kp_o = 67.57$. Pf $(400, 400)$ Paso $= 50$

Trayectoria	**Sección**	**Error en x**	**Error en y**	**% Error resultante**
T1	1.4	101.95	94.14	5.96
T2	2.1	13.98	10.44	12.23
	2.2	28.18	2.05	9.18
	2.3	29.77	9.57	5.51
	2.4	17.28	2.77	4.05
	2.5	6.41	31.98	4.46
	2.6	23.03	49.97	4.38
	2.7	23.00	47.00	8.79
T3	3.1	19.67	3.08	11.68
	3.2	27.36	9.98	6.52
	3.3	29.13	7.80	5.74
	3.4	11.90	3.23	3.04
	3.5	12.19	35.96	4.27
	3.6	20.74	53.54	5.22
	3.7	16.00	29.29	1.82

Cuadro 26: Errores de posición de la Figura 63

	Error en x	**Error en y**	**% Error resultante**
Trayectoria completa	13.41	0.19	1.67

Cuadro 27: Errores de posición de la trayectoria completa

Para el caso con paso de 50 se obtuvieron mejores resultados que en el caso anterior. Pese a que la trayectoria sigue sin tener una tendencia uniforme la aproximación a los puntos

finales demostró ser más precisa con este tipo de paso. Para estas trayectorias entre cada puntos el robot seguía cambiando su orientación pero en este caso lo hacía de forma más leve.

Luego de observar los dos casos con diferente tipo de paso y sobre todo comparar los valores finales se observa que los datos del Cuadro 27 son menores en los tres apartados (x, y, resultante) con lo que se puede determinar que para una trayectoria larga el mejor caso es realizarla de forma completa y no por tramos, aunque las trayectorias con paso de 50 no son del todo erróneas.

13.7.2. Controlador PID-n

Trayectorias cortas con paso de 100

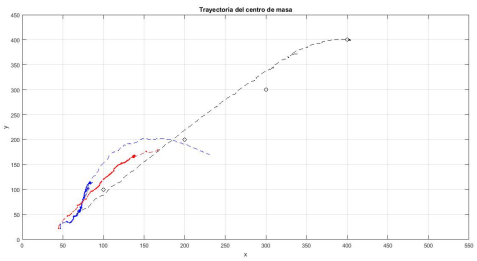


Figura 64: Trayectorias cotas con paso de 100

Trayectoria	**Sección**	**Error en x**	**Error en y**	**% Error resultante**
T1	1.1	16.00	12.50	0.72
	1.2	30.82	30.38	2.75
T2*	2.1	12.00	18.50	2.67
	2.2	31.95	21.89	13.43

Cuadro 28: Errores de posición de la Figura 64

Para la Trayectoria 2 de la Figura 64 se usó el punto inicial $(150, 150)$. Por lo que en el Cuadro 28 la sección 2.1 los valores están referenciados a este punto.

	Error en x	**Error en y**	**% Error resultante**
Trayectoria completa	3.36	0.53	0.14

Cuadro 29: Errores de posición de trayectoria completa la Figura 64

Trayectorias cortas con paso de 200

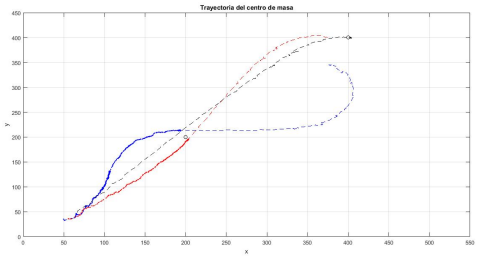


Figura 65: Trayectorias cotas con paso de 200

Trayectoria	**Sección**	**Error en x**	**Error en y**	**% Error resultante**
T1	1.1	8.68	13.5	1.36
	1.2	25.16	57.72	10.26
T2	2.1	2.83	5.17	0.56
	2.2	22.02	1.51	2.91

Cuadro 30: Errores de posición de la Figura 65

	Error en x	**Error en y**	**% Error resultante**
Trayectoria completa	3.36	0.53	0.14

Cuadro 31: Errores de posición de trayectoria completa la Figura 65

Trayectorias cortas con paso de 250

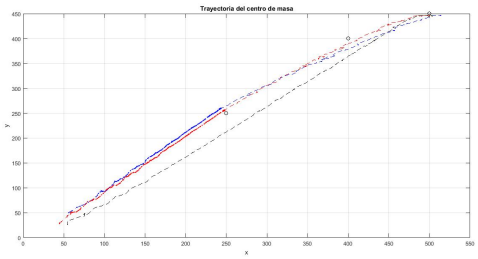


Figura 66: Trayectorias cotas con paso de 250

Trayectoria	**Sección**	**Error en x**	**Error en y**	% **Error resultante**
T1	1.1	6.42	10.42	0.86
	1.2	13.50	3.50	1.16
T2	2.1	4.01	4.50	0.11
	2.2	0.50	3.00	0.35

Cuadro 32: Errores de posición de la Figura 66

	Error en x	**Error en y**	% **Error resultante**
Trayectoria completa	3.50	3.00	0.0907

Cuadro 33: Errores de posición de trayectoria completa la Figura 66

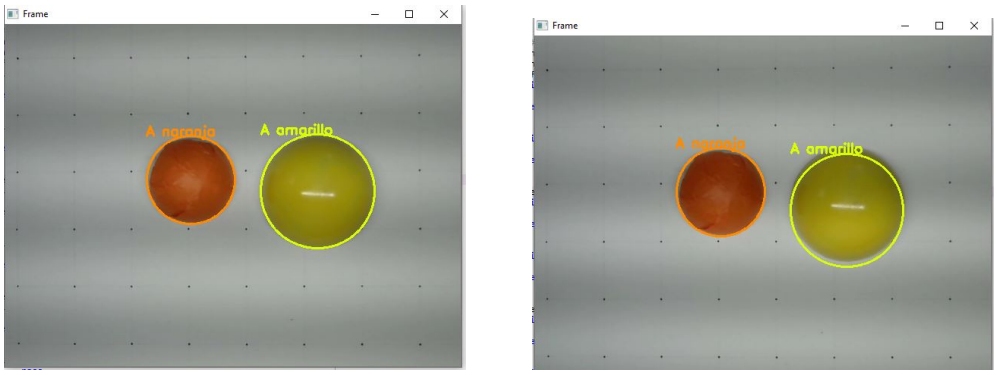
Para el controlador PID se observó el mismo resultado que en la sección 13.7.1 en el cual para trayectorias cortas el controlador no tuvo la reacción esperada. Esto debido a que el controlador direcciona el centro de masa del robot y no toma en consideración la orientación final con la que termina. Por lo que al comenzar una nueva trayectoria comienza con una orientación que le complica el movimiento final.

Sin embargo dividir la trayectoria completa en trayectorias *semi largas* se tuvo mejor resultado que en la sección 13.7.1 ya que el controlador PID tiene una mejor respuesta a trayectorias largas a pesar del *delay* que presenta el sistema.

CAPÍTULO 14

Controlador PID a robot esférico

Habiendo cumplido con el objetivo de sintonizar el algoritmo de control para el robot diferencial se prosiguió a hacer lo mismo para el robot esférico. Sin embargo antes de comenzar con las pruebas se observaron algunas características la nueva geometría del nuevo sistema.



(a) Detección de centro de masa 1 (b) Detección de centro de masa 2

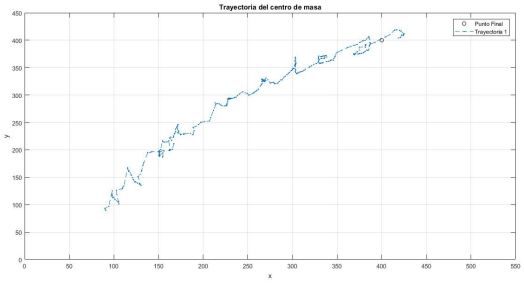
Figura 67: Detección de centro de masa

Se dispone de dos esferas, una de 14.5 y otra de 19.5 cm de diámetro respectivamente. Originalmente amabas de color amarillo. Al analizar la detección de visión de computadora se muestra como el color amarillo original es más sensible a cambios en la detección del color. Como se observa en la Figura 67 la esfera de color naranja permanece constante en la detección del borde mientras que la amarilla por un momento desfasa el contorno que dibuja sobre la superficie detectada.

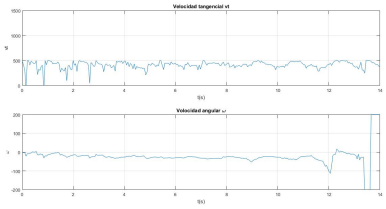
Este desfase en la detección es un problema para el algoritmo de control ya que al ser un

diámetro considerable respecto al área de trabajo hace que la posición del centro de masa cambie repentinamente y el controlador responde de forma agresiva cuando ya se encuentra cercano al punto de destino.

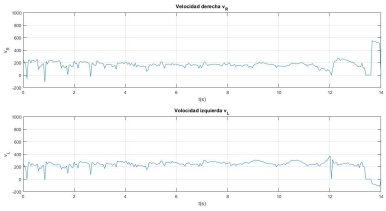
14.1. Esfera de 14.5 cm



(a) Trayectorias controlador PID RE iteración 1



(b) Valores de controladores



(c) Velocidades de llantas

Figura 68: Controlador PID - RE iteración 1

Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
68a (400,400)	1	18.64	4.03	2.85

Cuadro 34: Errores de posición de la Figura 68a

	Controlador posición				Controlador orientación		
	V_o	α	kd_{vt}	ki_{vt}	kp_w	kd_w	ki_w
Fig. 68	500	0.35	22.85	3000	67.57	45.57	600

Cuadro 35: Valores de controlador PID RE iteración 1

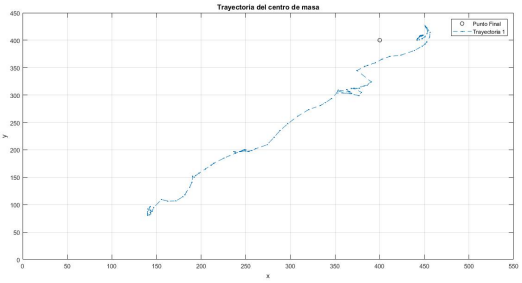
En la Figura 68a se observa la trayectoria de la iteración 1. A diferencia de las trayectorias mostradas en el capítulo anterior (13) no presenta la tendencia de una curva *suave* sino que se mira con mayor distorsión o ruido a lo largo de la misma. Esto se debe a que el robot

esférico no tiene una locomoción constante sino que tiende a balancearse hacia atrás y adelante durante el recorrido de la trayectoria.

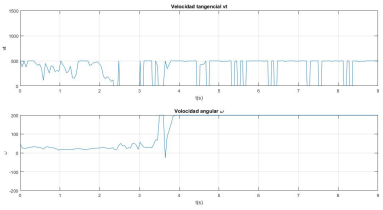
Debido a que la posición interna del robot diferencial estaba orientada en línea recta hacia el punto de destino se observa en las Figuras 68b,69c, que no se tiene demasiado cambio respecto a los resultados anteriores solo con el robot diferencial.

En el Cuadro 34 se observa como para esta primera iteración si bien el error resultante es de 2.85 % la diferencia de posición en x es alta.

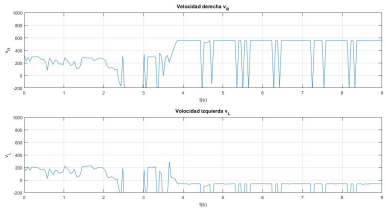
Como valores de controlador se usaron los que aparecen en el Cuadro 35, respecto a los utilizados en la última parte del robot diferencial se aumentó el valor de V_o ya que es la velocidad lineal predominante y ahora el sistema debe tener más torque para mover tanto la base diferencial como la esfera.



(a) Trayectorias controlador PID RE iteración 2



(b) Valores de controladores



(c) Velocidades de llantas

Figura 69: Controlador PID - RE iteración 2

Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
69a (400,400)	1	48.20	8.86	7.24

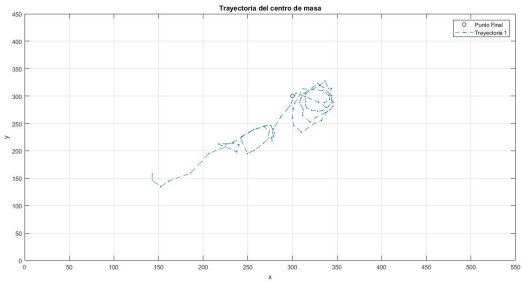
Cuadro 36: Errores de posición de la Figura 69a

Para la iteración 2 se observa que la trayectoria de la prueba tiene un mejor resultado en la continuidad de los puntos, esto se debe a que se hizo el cambio de color del amarillo original a anaranjado, color con el que se tiene una mejor respuesta en la visión de computadora. A pesar de que la trayectoria es más definida el balanceo que realiza el robot esférico por momentos parece desestabilizarlo con lo que se tienen cambios agresivos o varias tomas de

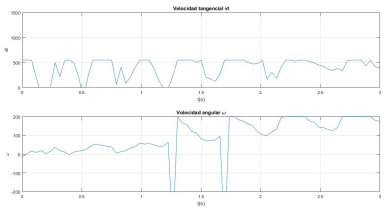
lectura del centro de masa.

Como se muestra en el Cuadro 36 el error final fue aún mayor que la iteración anterior teniendo más error resultante y una diferencia mayor en el eje x. Al usar una esfera con este diámetro puede darse el caso en que el robot diferencial toma una posición en la que queda muy ajustado y por lo tanto los motores nuevamente (sección 10.2) no tienen el torque necesario para mover por completo la esfera.

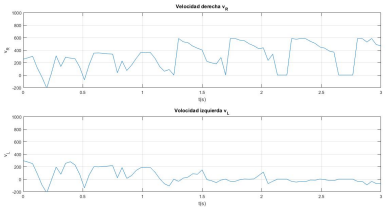
14.2. Esfera de 19.5 cm



(a) Trayectorias controlador PID RE iteración 3



(b) Valores de controladores



(c) Velocidades de llantas

Figura 70: Controlador PID - RE iteración 3

Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
70a (300,300)	1	35.50	12.50	6.14

Cuadro 37: Errores de posición de la Figura 70a

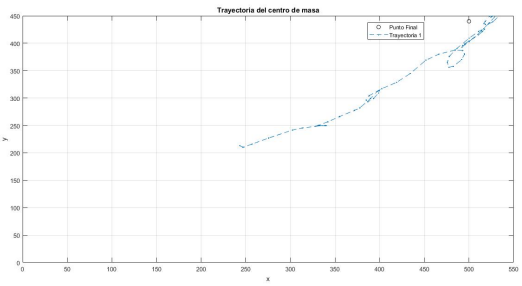
	Controlador posición				Controlador orientación		
	V_o	α	kd_{vt}	ki_{vt}	kp_w	kd_w	ki_w
Fig. 70	400	0.35	22.85	3000	67.57	45.57	600

Cuadro 38: Valores de controlador PID RE iteración 3

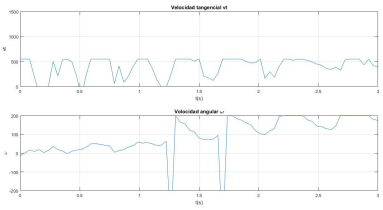
En la Figura 70a se observa la trayectoria de la prueba con la esfera de 19.5 cm la cual al tener el color amarillo muestra ruido en la trayectoria como sucedió en la sección anterior (14.1). A pesar de tener un movimiento casi lineal, los valores de los controladores 70b son inestables ya que reaccionan a cada lectura que se tiene del centro de masa.

En el Cuadro 38 se muestran los valores usados para el controlador PID, en este caso a pesar de que la esfera es más grande se redujo la velocidad del robot V_o ya que al tener más espacio interior no necesita tanto torque para mover la esfera.

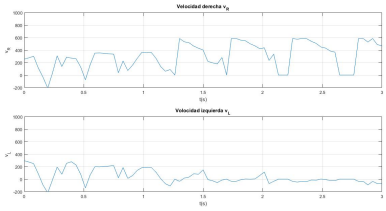
Los valores de error se muestran en el Cuadro 37, en este caso sí son valores alejados de los que se obtuvieron anteriormente. A diferencia de la esfera anterior está al ser más grande permite que el robot diferencial no permanece todo el tiempo en la posición fija esperada (sección 10.4) llegando en algunos casos a tener no solo un desfase en los ejes x y y sino también en el eje z y terminando volteado por completo.



(a) Trayectorias controlador PID RE iteración 4



(b) Valores de controladores



(c) Velocidades de llantas

Figura 71: Controlador PID - RE iteración 4

Figura P final	**Trayectoria**	**Error en x**	**Error en y**	**% Error resultante**
71a (500,440)	1	38.07	29.74	8.53

Cuadro 39: Errores de posición de la Figura 71a

La Figura 71a muestra la trayectoria de esta iteración, para la cual también se cambió el color de la esfera de amarillo a anaranjado. Si bien la trayectoria es más continua, los valores de lectura del centro de masa tienden a tener una respuesta muy agresiva en los controladores como se observa en las Figuras 71b, 71c.

CAPÍTULO 15

Conclusiones

- Los porcentajes de error entre el vector resultante y experimental fue menor al 10 % para el robot esférico y 4 % para el robot diferencial, con lo que se tuvo una buena aproximación al punto de destino
- El consumo de potencia experimental del circuito electrónico resultó ser menor al teórico, por lo que la autonomía de las baterías aumentó de 1.5 a 4 horas.
- El costo de la parte electrónica del robot esférico es de 25$, sin embargo al agregar los motores DC y la fabricación de la PCB el valor aumenta a 75$. Con lo que se convierte en un prototipo de costo medio/alto. Esto sin tomar en cuenta la manufactura de la parte mecánica.
- Para el ensamble inicial con el diseño mecánico se comprobó en las pruebas que los motores proporcionan poco torque, por lo que en el diseño final se agregaron dos motores extra para obtener el torque necesario para el movimiento del robot esférico.
- Con el controlador proporcional normalizado se obtuvo un menor error en estado estacionario respecto al controlador proporcional estándar.
- Para el algoritmo de control se implementó una parte derivativa (D) considerable. Con el objetivo de corregir el problema de *delay* que presenta el sistema. Sin embargo implementar esta técnica no fue capaz de corregir por completo dicho problema.
- Para la parte integral (I) del controlador se implementó de forma digital una técnica de Anti Wind Up para evitar la saturación de los motores DC. Esto con el objetivo de contar con la parte PI de controlador que permite solucionar el problema de rastreo y eliminar el error en estado estacionario.
- La locomoción del robot tuvo mejor respuesta a trayectorias largas respecto a trayectorias cortas en donde el robot se muestra sensible a la orientación inicial que presente para cumplir con la tarea de alcanzar el punto de destino deseado.
- Se comprobó una de las principales características del controlador PID. Poder sintonizarlo de manera manual, logrando un comportamiento acertado.

- La estructura de la esfera demostró no ser lo suficientemente estable para permitir una buena locomoción del robot esférico, lo que dificulta la tarea del algoritmo de control.

CAPÍTULO 16

Recomendaciones

- Con el objetivo de reducir costo en la fabricación del prototipo probar la implementación de la comunicación wifi y el control de los motores DC con el módulo wifi ESP8266-32. Módulo que ya incluye canales PWM.
- Actualizar el Firmware del módulo wifi para evitar la saturación de información. Esto podría ayudar a reducir el *delay* que presenta el sistema.
- Debido a problemas de implementación entre la parte electrónica y la parte mecánica usar con motores DC que proporcionen más torque.
- Usar una cámara web que tenga mayor resolución a la cámara usada (Logitech C270). Así mismo que esta se encuentre a una mayor altura para tener mayor área de trabajo.
- Usar un *encoder* en cada llanta del robot diferencial como sensor extra. Lo que permitiría aplicar de mejor forma una técnica Anti Wind Up además de ayudar en mediciones para corregir *delay* en el sistema.
- Modificar la estructura del robot esférico para que la estabilidad sea mejor, lo que facilite la implementación del algoritmo de control.
- Usar un tamaño de esfera entre 14.5 y 19.5 cm de diámetro para que el robot diferencial tenga un mejor ajuste de la zona interna y pueda tener una mejor locomoción.

CAPÍTULO 17

Bibliografía

[1] R. Ajtún, «Reingeniería de Megaproyectos Fase I-Módulo de Swarm Robotics», Universidad del Valle de Guatemala, 2017.

[2] D. Pickhem, *The Robotarium: A remotely accessible swarm robotics research testbed*, Último acceso 20 de julio de 2019, 2017. dirección: `https://robohub.org/the-robotarium-a-remotely-accessible-swarm-robotics-research-testbed/`.

[3] M. Castillo, «Diseñar e implementar una red de comunicación inalámbrica para la experimentación en robótica de enjambre», Universidad del Valle de Guatemala, 2018.

[4] J. Lima, «Diseño e implementación de una nueva plataforma móvil para aplicaciones en robótica de enjambre», Universidad del Valle de Guatemala, 2018.

[5] P. Miller, «Teoría de los enjambres», *Fundación Dialnet*, vol. 21, n.º 1, págs. 90-111, 2007.

[6] Hisour, *Robot Esférico*, Último acceso 23 de julio de 2019, 2019. dirección: `https://www.hisour.com/es/spherical-robot-42913/`.

[7] L. A. A. Zea, «Diseño e implementación de un robot móvil con control de trayectoria mediante principios odométricos», Universidad Nacional Mayor de San Marcos, 2015.

[8] A. B. Azcón, «Análisis y diseño del control deposición de un robot móvil con tracción diferencial», Universidad Rovira I Virgili, 2003.

[9] J. A. C. Falcón, *WIFI Lo que se necesita conocer*. Grupo RC, 2010.

[10] S. Buettrich y A. Escudero, «Topología e Infraestructur estructura Básica de Redes Inalámbricas», en, 1.ª ed. Alemania: wire.less.dk, 2007, cap. 4.

[11] J. F. Martínez, *Implantación de aplicaciones Web en entornos internet, intranet y extranet*. RA-MA, S.A., 2015.

[12] D. de Ciencias de la computación, «Internet:TCP/IP. Transmisión de datos y redes», https://elvex.ugr.es/decsai/internet/pdf/6%20Internet%20-%20TCPIP.pdf, 2015.

[13] UPV, *Puertos y Sockets*, Último acceso 23 de agosto de 2019, 2010. dirección: `http://personales.upv.es/rmartin/tcpip/cap02s10.html`.

[14] W. Tomasi, *Sistemas de Comunicaciones Electrónicas*. Prentice Hall, 2003.

[15] Anónimo, *Indtroducción al Módulo ESP8266*, Último acceso 18 de julio de 2019, 2015. dirección: http://visystem.ddns.net:7442/ESP8266-modulos/.

[16] V. Rossano, *Electrónica y Microcontroladores PIC*. Users, 2009.

[17] Anónimo, *Características Básicas del PIC16F887*, Último acceso 18 de julio de 2019, 2016. dirección: https://www.mikroe.com/ebooks/microcontroladores-pic-programacion-en-c-con-ejemplos/caracteristicas-basicas-del-pic16f887.

[18] S. K. Peddapelli, *Pulse Width Modulation: Analysis and Performance in Multilevel Inverters*. De Gruyter Oldenbourg, 2007.

[19] M. Duncan, *Process Dynamics and Control*. John Wiley Sons Inc., 2014.

[20] K. Ogata, *Ingeniería de Control Moderna*. Pearson Education, 2010.

[21] N. Nice, *Control Systems Engineering*. John Wiley Sons Inc., 2011.

CAPÍTULO 18

Anexos

18.1. Cotización de PCB

Información básica : PCB estándar

Nº de producto :	W69752ASL1	Estado del pedido :	Requiere verificación
Tiempo de creación :	2020/5/17 02:42:09	Tiempo de preparación :	3-4 días
Tiempo de finalización estimado :	2020-05-20 Zona horaria de China (GMT+8)	Archivo Gerber :	RE_v3.rar

Información de parámetros :

Tipo de placa :	Piezas sueltas	Forma de panel :	
Diseños diferentes en el panel :	1	Tolerar X-out en el panel :	
Dimensiones :	43 x 48 mm	Cantidad :	5
Capa :	2 Capas	Material :	FR-4: TG130
Grosor :	1.6 mm	Min pista/espacio :	8/8mil
Min tamaño agujero :	0.3mm ↑	Máscara de soldadura :	Verde
Serigrafía :	Blanco	Dedos de oro :	No
Acabado superficial :	Ninguno(Cobre desnudo)	"HASL" por "ENIG"	No
Procesamiento de vías :	Vías cubiertas	Acabado de cobre :	2 oz Cu
Pcb extra Número de producto :		Opciones adicionales :	Marcado UL:Ninguno,
Número de PO :		Fabricación :	

Total: **US $51.00**

Figura 72: Cotización de fabricación de PCB en PCBWAY

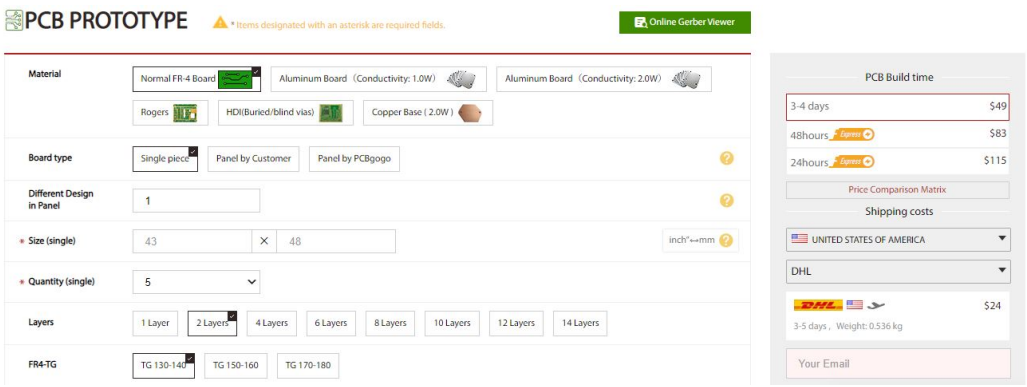
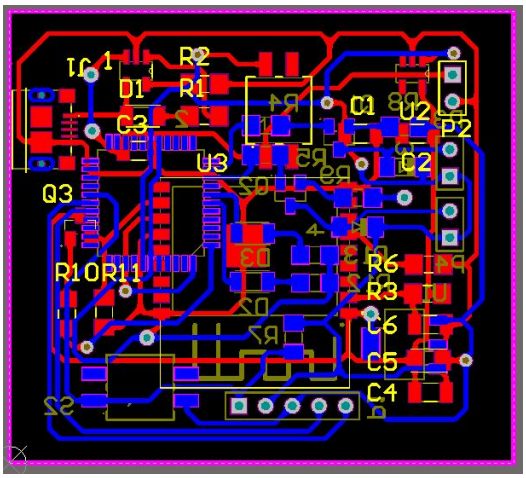
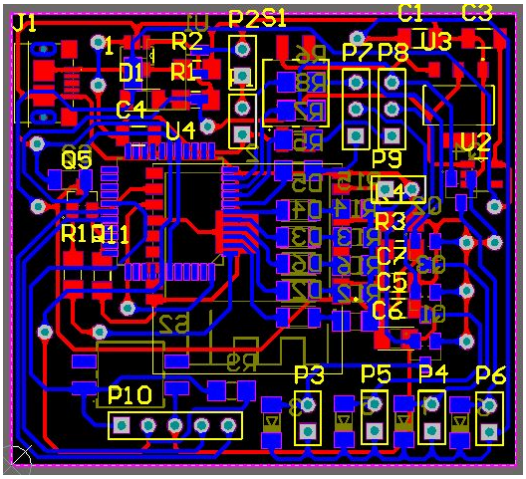


Figura 73: Cotización de fabricación de PCB en PCB PROTOTYPE

18.2. Diseño en Altium Designer de PCB



(a) Diseño de versión inicial de PCB



(b) Diseño versión final de PCB

Figura 74: Diseño de PCB en Altium Designer