PROYECTO FINAL

Daniela Arely Morales Hernández 142976
Stephanie Lizeth Malvaes Diaz 135515

Resumen

Existen múltiples situaciones en las cuales un robot requiere seguir ciertas trayectorias. Para abordar el problema de que nuestro robot siga trayectorias usamos: dos motores de corriente directa, opto-interruptores, un Arduino, controles PID, un sistema de visión y un programa en ROS. Con la información provista por el sistema de visión, se logró calcular una ruta del inicio al fin y los motores con el control y los opto-interruptores siguen dicha ruta.

1. INTRODUCCIÓN

Los robots autónomos se vuelven cada vez una realidad más cercana. Un robot autónomo es aquel que no requiere una aportación humana durante su operación para ejecutar su función. Dentro de estos robots autónomos existen aquellos que se desplazan de un lugar a otro evitando obstáculos.

Por ejemplo, parte del trabajo de Eduardo Morales en el Instituto Nacional de Astrofísica, Óptica y Electrónica consiste en un robot autónomo que interactúa con no expertos. A este robot se le pide que obtenga un objeto y el robot lo busca dentro de una serie de cuartos usando un sistema de visión. Genera una ruta óptima con base en una

heurística de distancia a los cuartos, tamaño de los cuartos y la probabilidad de encontrar el objeto deseado en el cuarto.

El robot puede agarrar objetos por medio de una mano operada con motores que igualmente requieren control. Este robot en particular se piensa que sea parte de la vida diaria en el hogar.

Otra empresa que está a la vanguardia de lo que concierne robots autónomos es Tesla. Su concepto de un vehículo autónomo incorpora elementos similares a los que se presentan en nuestro problema. Por medio de sensores ultrasónicos y cámaras, se crea una imagen del entorno.

Con base en la información recolectada se determina si el vehículo debe acelerar, frenar, mantener la velocidad constante o girar en alguna dirección. Este tipo de tecnologías es capaz de alertar al conductor momentos antes de que ocurra un accidente y evitarlo. El trazado de rutas, el movimiento del vehículo y la necesidad no solo de seguir un carril, sino evitar colisiones con otros vehículos y seguir los señalamientos exaltan el nivel al que puede llegar nuestra problemática inicial.

Nuestro objetivo entonces es: dado un vehículo con dos motores y un Arduino, ser capaces de que de manera autónoma reconozca un punto objetivo y siga una ruta hasta él, evitando obstáculos fijos a lo largo de su ruta.

A continuación, en el Marco Teórico, se esbozan los conceptos principales utilizados en la solución a la problemática. En el Desarrollo se explica la solución de manera

explícita, detallando lo que se llevó a cabo en el experimento.

Los Resultados evalúan el éxito del proceso descrito en el Desarrollo junto con sus limitantes. La sección de conclusiones incluye una aportación individual de cada miembro del equipo que resalta los aspectos principales del experimento. En Rol se enuncia la participación que tuvo cada miembro del equipo en el desarrollo de la solución.

Finalmente, en Fuentes Consultadas se exhiben las fuentes citadas de manera directa, indirecta o que aportaron los conocimientos para el desarrollo del experimento.

2. MARCO TEÓRICO

Un robot autónomo que siga una ruta tiene varios componentes, los cuales se pueden dividir en dos categorías principales: la parte de la lógica, que se encarga de trazar la ruta y que el robot la siga; y la parte del control, es decir, lo que se encarga de regular la velocidad de los motores, así como de seguir instrucciones acerca del movimiento del robot.

En este caso se hizo un robot con tracción diferencial. La tracción diferencial se caracteriza por el uso de dos motores independientes para controlar las llantas de cada lado, a diferencia de la tracción de un coche donde la rotación de las llantas fronteras o traseras es igual en ambos lados.

La Figura 1 muestra un diagrama de un robot simple con tracción diferencial. Este robot se caracteriza por el radio de cada rueda y la distancia entre las llantas [1].

El uso de motores independientes para cada llanta permite al robot tener un mayor rango de movimientos. Por ejemplo, además de poder moverse hacia adelante y atrás girando las llantas a la misma velocidad, puede también controlar la velocidad angular del robot haciendo diferir la velocidad de los motores, e inclusive puede rotar en su propio eje cuando hace girar los motores en velocidades opuestas. En otras palabras, la tracción diferencial permite un mayor control sobre la velocidad y la velocidad angular.

Específicamente, la relación entre la velocidad de cada rueda (˙l; ˙r) y la velocidad v con velocidad angular es:

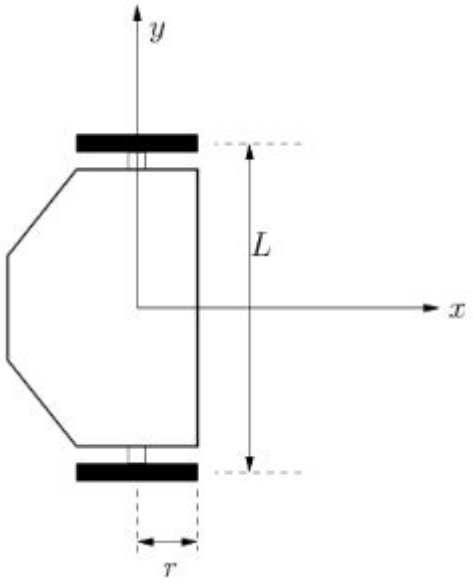


Figura 1: Diagrama de un robot diferencial

A pesar de estas ventajas, la tracción diferencial trae consigo un gran problema. Debido al uso de motores independientes, es necesario tener un buen control no solo individual, sino sobre todos los motores en conjunto. Es decir, una tracción diferencial gana

precisión sobre el movimiento del robot a costo de mayor precisión en el control de los robots.

Nuestro vehículo está conformado por dos motores de corriente directa (motores DC) y un Arduino. Este último es una plataforma de hardware y software libre, basada en una placa con un microcontrolador y un entorno (IDE) de desarollo [2].

Los motores DC son dispositivos que convierten la energía eléctrica en mecánica, provocando un movimiento rotatorio y tienen dos partes: un estator y un rotor. El primero da soporte mecánico al motor y es donde se encuentran los polos y el segundo, generalmente es de forma cilíndrica el cual se polariza por medio de una corriente la cual le llega mediante dos escobillas [3]. Para controlar cómo fluye el voltaje a través de los motores se utilizó un puente H y para controlar la velocidad se implementó un control PID (Proportional Integrative Derivative).

Un puente H es un circuito electrónico que permite que un motor eléctrico DC gire a la izquierda, a la derecha, entre en freno pasivo y en freno activo. Este circuito se construye con 4 interruptores. Así, dependiendo de cuáles interruptores estén abiertos/cerrados el motor se mueve en una dirección o se frena. En la Fig. 2 se puede ver el diagrama de un puente H y su funcionamiento.

Los controles PID funcionan con base en una retroalimentación de la señal de salida, conocida como señal de error, e(t). Esta señal se calcula con la diferencia de la velocidad deseada y la velocidad actual del motor. Posteriormente se alimenta al controlador, el cual emite una señal de control al motor, u(t), para intentar alcanzar el valor deseado [4]. Como su nombre lo indica un controlador PID está compuesto de tres partes: la proporcional, la diferencial y la integral (Fig. 3).

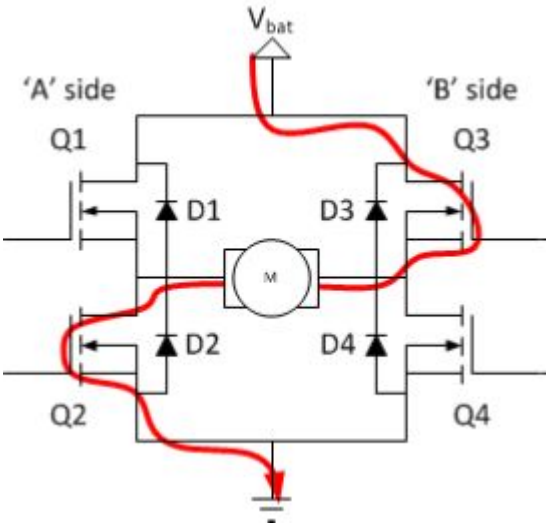


Figura 2: Diagrama de un puente H

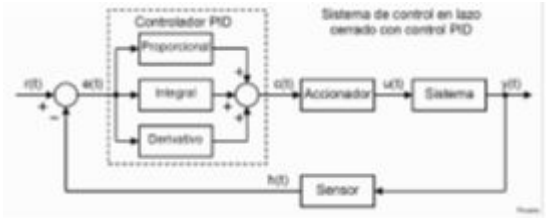


Figura 3: Diagrama de un controlador PID

La acción de control proporcional tiene como objetivo minimizar el error del sistema y da una salida proporcional a la señal de error. La acción de control integral tiene como objetivo reducir el error del sistema en régimen permanente.

Es un control más lento basado en la señal de error acumulada. Por último la acción de control derivativa tiene como objetivo aumentar la estabilidad del sistema por medio del control de la

rapidez con que cambia la señal del motor.

De esta manera la ecuación de la señal de control está dada por:

$u(t) = Kpe(t) + Ki + Kd$

Para poder determinar la velocidad a la que corren los motores se utilizaron dos optointerruptores, uno para cada rueda. Un optointerruptor es un sensor en forma de ”U”:

En uno de los extremos está un diodo emisor de infrarrojos, y en el otro hay un fototransistor que recibe la señal. Este sensor detecta cuando un objeto para por la ranura pues se interrumpe el rayo de luz infrarroja (Fig. 4) [6]. A diferencia de los interruptores mecánicos, los optointerruptores son interruptores sin contacto, lo que mejora la confiabilidad al evitar que el sensor se desgaste debido a la abrasión(contacto) [7].

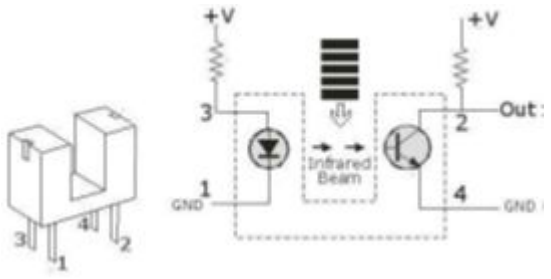


Figura 4: Diagrama de un optointerruptor

Tanto los obstáculos, como la meta y la posición del robot eran publicados por un sistema de visión a diferentes nodos

Figura 5: Sistema de publicación-inscripción de ROS.

Estos se procesan y se calculaba la ruta utilizando un algoritmo de búsqueda. Se realizó una comunicación indirecta entre el Arduino y la computadora que procesa la trayectoria por medio de XBee.

ROS (Robot Operating System), es un meta sistema operativo para robots, es decir, es un intermedio entre un sistema operativo y un middleware. Tiene como objetivo principal facilitar la creación de comportamientos de robots que sean complejos y robustos entre diversas plataformas [8]. ROS está basado en una arquitectura peer-to-peer, acoplada a un sistema de almacenamiento en búfer y un sistema de búsqueda (master), los cuales permiten que cualesquiera dos componentes se comuniquen entre ellos de manera síncrona o asíncrona. La comunicación asíncrona se realiza por medio de un sistema de publicación-inscripción en el que se publican mensajes, que pueden ser de distintos tipos, a diferentes tópicos (Fig. 5)

3. DESARROLLO

3.1. Trayectorias

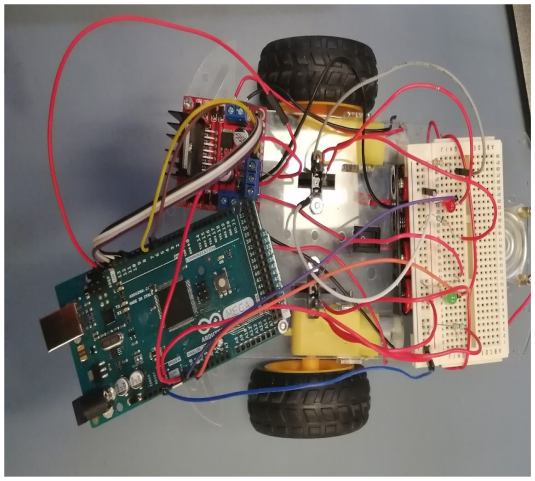
Se implementó mediante el programa de arduino PID, mediante cálculos para hacer las vueltas y líneas rectas, para el círculo y el cuadrado.

3.2. ROS

Se creó un workspace, directorio y archivos de python con la información del escucha y el escritor. Se presentaron dificultades para

3.3. Robot

Para controlar cada motor del robot, se utilizó un puente H conectado a una batería de 7.4 V. Luego, se agregó un encoder con un opto-interruptor al eje del motor para poder medir su velocidad. Todos estos componentes se conectaron al ATmega. Desde ahí, se corrió un programa que contaba los ticks de cada interruptor y con esto calculaba la velocidad del motor cada 250 ms. Después, comparaba la velocidad contra el objetivo, y, usando un control PID, determinaba el voltaje necesario para el motor. Este voltaje se conseguía utilizando un PWM al control del puente H, reduciendo el voltaje original de 7.4 V al determinado por el ciclo de trabajo de la señal. Finalmente, el Arduino se comunicaba con el sistema de ROS que determinaba la ruta a través de un XBee. Al recibir un mensaje del sistema central, el Arduino actualiza la velocidad objetivo de cada motor, así como la dirección en que este debía girar.



La Figura 13 muestra el robot diferencial completo.

4. RESULTADOS

4.1. Trayectorias

Se realizó la trayectoria mediante el programa PID en arduino del círculo y el cuadrado. No obtuvimos tanta precisión, pero gran parte de las veces si siguió las figuras esperadas.

El principal problema durante el desarrollo de las trayectorias es pasamos de intentar hacerlo con python suponiendo que nuestra comunicación Xbee resultaría.

Al darnos por vencidas en conseguir esa parte del proyecto procedimos a desarrollar en PID en arduino, el cual se encuentra en el repositorio.

Con el PID se tuvo problemas para obtener la lectura de los encoders ya que no lograba recibir la lectura de las vueltas.

Una vez que terminamos de configurar el PID no logramos implementar la trayectoria ya que el valor recibido no era actualizado en los métodos usados.

Para seudo-lograr que el robot recorriera las trayectorias usamos algunos métodos que activan los motores y modifican la velocidad de cada uno, así como uso de delay para definir la trayectoria, igualmente la rutina de arduino se encuentra en el repositorio.

4.2. ROS

Se siguieron los tutoriales de ROS, creación de workspace, de directorio, escucha y escritor en dos programas de python, no se concluyó por

problemas de software, ya que había problema con un archivo de la instalación de ROS.

4.3. Robot

El Cuadro 1 muestra la configuración de los pines del ATmega 2560 utilizado en la construcción del robot.

Además, para alimentar el Arduino y los motores DC se utilizaron pilas. Tuvimos problemas con la configuración del Xbee, con el cual se buscaba una comunicación entre el ATmega 2560 y la PC que determinaba la velocidad deseada de cada motor.

La PC enviaba, en orden, la velocidad del motor derecho en ticks/seg, la dirección del motor derecho (1 hacia adelante, 0 hacia atrás), la velocidad del motor izquierdo y la dirección del motor izquierdo. Cada uno de estos elementos se enviaba como un entero de 8 bits sin signo.

Para los controles PID, se obtuvieron las siguientes constantes:

- klp= 0;9
- krp= 0;7
- kli = 0;6
- kri = 0;7
- kld= 0;1
- krd= 0;15

Los pines usados son los siguientes:

13 Output Motor Derecho -0

12 Output Motor Derecho -1

10 Output Motor Izquierdo -0

11 Output Motor Izquierdo -1

5. CONCLUSIONES

- Arely

El uso de tracción diferencial en el robot representó un desafío debido a la necesidad de tener un control con comportamiento similar en ambos motores.

Durante la práctica se presentaron varios retos, el principal fue la falta de una batería que diera un voltaje constante debido a que inicialmente usamos las que venía con el robot.

Una vez que nos percatamos de que debido la pila de las baterías se tenían que cambiar los delays fue un constante cambio que al funcionar, el segundo intento no llevaba la misma potencia por lo que no funcionaba de la misma manera.

Logramos entender el concepto de la mayoría de los factores que requería la práctica pero por cuestiones de falta de tiempo para dominar algunos entornos y los materiales inadecuados(refiriéndose a la pila) nos dificulto mucho el concluir de manera aceptable.

-Stephanie

Este proyecto nos permitió darnos cuenta que pasar de la teoría a la práctica no siempre es fácil pues los componentes físicos casi nunca se comportan de manera ideal. Esto se debe de tener en cuenta al momento de crear los algoritmos para poder manejar fallos de mejor manera.

La liga del repositorio es la siguiente:

https://github.com/DArely2488/PM_ProyectoFinal

6. FUENTES CONSULTADAS

[1] S. M. LaValle, “Planning algorithms,” accesed 2019-05-22. [Online]. Available: http://planning.cs.uiuc.edu/node659.html

[2] Sparkfun, “What is an arduino?” accessed 2019-05-22. [Online]. Available:https://learn.sparkfun.com/tutorials/ what-is-an-arduino/all

[3] Geekbot Electronics, “Motores de dc,” accessed 2019-05-21. [Online]. Available: http://www.geekbotelectronics.com/ motores-de-dc/

[4] L. Vélez de Guevara, “Sistemas de control de lazo cerrado,” accessed 2019-05-21. [Online]. Available: https://makinandovelez.wordpress.com/2018/02/15/sistemas-de-control-de-lazo-cerrado/

[5] V. Mazzone, “Controladores pid,” accessed 2019-04-11.[Online]. Available: http://www.eng.newcastle.edu.au/jhb519/teaching/caut1/Apuntes/PID.pdf

[6] L. Llamas, “Hacer un encoder óptico con un optointerruptor y arduino,” accessed 2019-05-22. [Online]. Available: https://www.luisllamas.es/usar-un-optointerruptor-con-arduino/

[7] ROHM Semiconductor, “What is a photointerrupter?” accessed 2019-05-22. [Online]. Available: https://www.rohm.com/electronics-basics/photointerrupters/what-is-a-photointerrupter

[8] Open Source Robotics Foundation, “About ros,” accessed 2019-05-22. [Online]. Available: http://www.ros.org/about-ros/

[9] U. Endriss, “Search techniques for artificial intelligence,” accessed 2019-05-22. [Online]. Available: http://www.cs.ubbcluj.ro/csatol/logfunk/prolog/slides/7-search.pdf

[10] XBee.cl, “¿qué es xbee?” accessed 2019-05-20. [Online]. Available:https://xbee.cl/que-es-xbee/