Práctica No. 6 Proyecto Final

Departamento Académico de Sistemas Digitales
Instituto Tecnológico Autónomo de México
Primavera 2019

JEAN PAUL VIRUEÑA ITURRIAGA 155265
FABIÁN ORDUÑA FERREIRA 159001

Abstract - En esta práctica se diseñó un robot que dándole instrucciones sea capaz de llegar a un objetivo mientras evita obstáculos. Para esto fue necesario subscribirnos con ROS a un sistema de visión que nos permita ver el ambiente de la prueba y calcular una ruta. Teniendo la ruta, se publicaban las instrucciones al robot. Este reporte tiene como objetivo poner en evidencia cómo se aplicaron y usaron los conocimientos y las herramientas aprendidas a lo largo del semestre durante el laboratorio.

I. INTRODUCCIÓN

En la actualidad, la tecnología es un recurso que tenemos para optimizar nuestras actividades diarias. Los robots son un gran ejemplo de esto ya que se han utilizado desde el siglo pasado para realizar tareas de forma automatizada que el ser humano puede realizar. Se pueden encontrar en todos lados ya que se pueden encontrar en entornos de manufactura, de ensamblaje, embalaje, transporte, todo tipo de exploración, cirugías, armamento, producción industrial [0] por decir algunos ejemplos. Con el paso del tiempo, la robótica ha mejorado considerablemente haciendo que las funciones de los robots sean más rápidas, más precisas y más eficientes.

Es por eso que el objetivo de esta práctica fue implementar un sistema mecatrónico a partir de los conocimientos y el uso de la herramienta adquiridos a lo largo de todo el semestre.

Para esto se diseñó un robot que a partir de un sistema de visión y dándole instrucciones sea capaz de llegar a un objetivo mientras evita hacer contacto con obstáculos.

Dicho diseño fue se construyó a partir de una patineta, dos ruedas, dos motores de corriente directa, dos optointerruptores, un sistema de engranes, una tarjeta Arduino y una terminal con Linux y ROS.

Se configuró la parte electrónica del robot conectando un circuito eléctrico que conectaban la tarjeta Arduino a los motores, a los encoders y a los puentes en H. Se le cargó un programa que permita a Arduino recibir moverse en función de las instrucciones que recibía mediante un XBEE.

Otro XBEE se conectó a la terminal donde se corría un archivo que se subscribió a sistema de visión, calculaba la ruta para ir al objetivo a partir de las posiciones de los obstáculos, el robot y el objetivo y publicaba las instrucciones que tenía que seguir el Arduino.

Al mismo tiempo se implementó dentro del software un controlador PID para poder calcular la velocidad del de los dos motores del robot y así tener un mejor desplazamiento.

Este reporte está organizado en distintas secciones, comenzando con un marco teórico, en el que se presenta la información relevante y de importancia para una mejor comprensión de lo que aquí se aborda; seguido del desarrollo, donde se presenta la manera en que se trabajó para la obtención de los resultados; la sección de resultados, en la que se presenta un análisis de lo obtenido y la sección de conclusiones.

II. MARCO TEÓRICO

Los sistemas embebidos son sistemas basados en microcontroladores o microprocesadores los cuales son diseñados para realizar una tarea en específico; es decir, para realizar operaciones específicas, comúnmente simples y en tiempo real. En nuestro caso tomamos una Arduino MEGA 2560 para realizar nuestro robot, dicha tarjeta es un microcontrolador con 54 pines de entrada y salida digital y 16 de entrada y salida análoga, fue diseñada para realizar proyectos de robótica por

Arduino, una plataforma de código abierto que se dedica a fabricar hardware y software [1 y 2].

A nuestro Arduino se le conectaron dos motores de corriente directa para que el robot pueda desplazarse, estos motores son dispositivos que transforman corriente directa en energía mecánica creando movimiento rotatorio [3, 4 y 5]. Con la ayuda de un puente en H (Figura 1), un circuito electrónico que transmite corriente directa, se le da la dirección al motor o se le indica si debe de frenar de manera pasiva o activa [6].

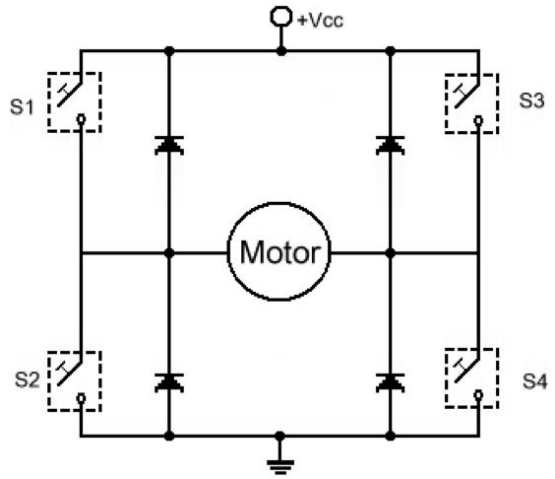

Figura 1 - Diagrama de un puente en H

Para controlar la velocidad del robot se usó un control PID, el cual es un dispositivo que permite controlar un sistema en lazo cerrado. Este último decide la velocidad del robot en función de las entradas y las salidas [7], para que alcance el estado de salida deseado a partir de una función de error del sistema al que se le aplica el control (ver Figura 2) [8]. Tiene tres componentes:

- Control Proporcional: Se encarga de multiplicar la señal de error por una constante *kp*.
- Control Integral: Se encarga de sumar errores para que la acción integral *ki*sea cada vez mayor.
- Control Derivativo: Se encarga de disminuir la velocidad con la que se acerca a la señal deseada mediante la acción derivativa *kd*

Las ecuación del control PID es la siguiente:

$$u(t) = kpe(t)+0tkie(t')dt'+kdde(t)dt$$

Donde *e(t)* es la función del error.

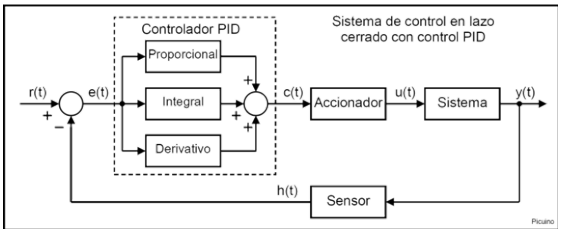

Figura 2 - Esquema de un controlador PID

Para calcular la velocidad de entrada del PID, fue necesario implementar un encoder con la ayuda de un optointerruptor. Un encoder es un transductor que ayuda a calcular la posición o velocidad angular de un eje [9]. Un optointerruptor es un sensor que emite un haz de luz hacia un fotodetector e indica si hay algo obstruyendo la emisión [10], a cada uno de los motores.

Como era necesario que nuestro robot se comunicara con con el sistema de visión y recibir instrucciones, se le conectó un XBEE, una tarjeta que es capaz de establecer un medio inalámbrico [11] usando un estándar para comunicación inalámbrica para redes de área personal, de bajo consumo que emplea señales de radio como medio, llamado Zigbee [12].

Una vez establecida la comunicación, se utilizó ROS para poder ver a nuestro robot, el sistema de visión y la terminal como un nodos y poder recibir y enviar datos e instrucciones.

Robot Operating System (ROS) es un framework que provee librerías, herramientas y convenciones para desarrollar software para robots. ROS contiene funciones para crear robustos y complejos comportamientos para plataformas robóticas [13 y 14]. Dentro de ROS está roscore, una colección de nodos y programas que contiene prerrequisitos de un sistema basado en ROS. Roscore es ejecutado

desde la terminal de una computadora que puedan correr los nodos de ROS dentro de la computadora [15].

Un nodo en ROS es un término utilizado para los ejecutables que están conectados a una red, también llamada grafo, de ROS para comunicar con otros nodos usando tópicos, usualmente un sistema de control de los robots está compuesto de diferentes tópicos [16]. Los tópicos (o topics) son buses, sistemas digitales que transportan información entre dos componentes de un sistema computacional, con un nombre para intercambiar mensajes de manera unidireccional. Para que dos o más nodos se puedan comunicar entre ellos, es necesario que ambos anuncien su tópico y su tipo de datos (todos los mensajes de un tópicos tienen que tener el mismo tipo de dato) [17].

En esta práctica se usaron dos tipos de nodo: publishers y subscribers. Los nodos publisher o publicadores se encargan de enviar mensajes en broadcast de un tipo de dato a un tópico en particular [18]. Los nodos subscriber o suscriptores se encargan de esperar mensajes de cierto tipo de dato a un tópico [19]. Para poder realizar la comunicaciones entre nodos es necesario representarlos en ROS, para eso se utiliza la clase NodeHandle [16 y 19].

III. DESARROLLO

Para cumplir con los objetivos del proyecto final realizamos distintos ejercicios para poder implementar un robot que fuera capaz de moverse de un punto inicial a un destino. En este proyecto se incluye uso de PID, uso de encoders, uso de comunicación usando Xbees, uso de Arduino y ROS para que dado un sistema de visión el robot pudiera llegar a su destino aunque hubiera obstáculos en el camino . Para esto fue necesario hacer uso de herramientas estudiadas a lo largo del curso.

Para la primera parte, la construcción del robot, empleamos un soporte al que le colocamos dos llantas a los costados, mismas que estaban conectadas a un motor para que el robot pudiera moverse. Sobre las llantas colocamos dos optointerruptores empleados para ayudar a la implementación del PID. Asimismo, sobre el soporte colocamos un arduino y una protoboard. La protoboard conectada al arduino, a los motores junto con un puente en H, a los optointerruptores para implementar encoders. El arduino lo conectamos también con un Xbee que posteriormente ayudaría a establecer comunicación. El robot, de la parte física, terminado se puede apreciar en la Figura 3.

La segunda parte consistió en probar los optointerruptores para poder emplearlos posteriormente y construir el PID. Una vez conectados a la protoboard y ésta al arduino, realizamos un código en c++ que nos ayudó a construir los encoder. Estos nos ayudaron para implementar el PID, con código en c++, en los motores de tal forma que la velocidad y los movimientos esperados se vieran reflejados de manera correcta en el robot.

Posteriormente, implementamos la comunicación de los Xbees, por un lado el que estaba conectado al arduino y en otro lado el que estaba conectado a la computadora. Primero empleamos XCTU para configurar y realizar pruebas de comunicación entre los dispositivos. Posteriormente, hicimos uso de la consola para configurarlos. Una vez que la comunicación fue exitosa, desarrollamos un código en c++ para que dada un mensaje recibido de la comunicación, el robot moviera los motores en la dirección que se solicitara.

Después, dado que el robot necesitaba llegar de un punto a otro y que podían presentarse obstáculos entre el punto de origen y destino, fue indispensable realizar un algoritmo que dado el punto de origen, el destino y ciertos obstáculos trazara una ruta que permitiera cumplir su cometido, llegar al destino sin chocar con los obstáculos. Para ello, fue necesario realizar un análisis que tomara en cuenta ciertas restricciones, como el tamaño del campo y el espacio que los obstáculos ocupaban. También realizamos el análisis de cómo debíamos tomar los distintos ángulos que el visor nos devolvería para que el robot rotara a cierta posición y los movimientos fueran precisos.

Paralelamente, se tomaron los archivos iniciales proporcionados por el profesor en el repositorio https://github.com/garygra/PM_proyecto_final.
Con ayuda de estos archivos procedimos a

construir los archivos que nos permitieran hacer el intercambio de información entre los Xbee, dado el sistema de visión. En este caso, empleamos Ros para poder llevar a cabo esta tarea, se crearon distintos tópicos que recibían la posición del robot (posición inicial), la posición del destino y las posiciones de los objetivos dado el sistema de visión; también, tópicos que realizaban la comunicación de cómo debía moverse el robot para transferirlo al arduino y usar lo implementado para cumplir la tarea, llegar al destino.

IV. RESULTADOS

En cuanto al armado del robot, tuvimos que cambiar en distintas ocasiones los puentes en H ya que no servían adecuadamente. También tuvimos que cambiar un motor porque no funcionaba y reparar una llanta porque el tornillo que la unía al motor estaba barrido. A pesar de los problemas, fuimos capaces de armar el robot de tal manera que los componentes funcionaran adecuadamente, el resultado se puede apreciar en la Figura 3.

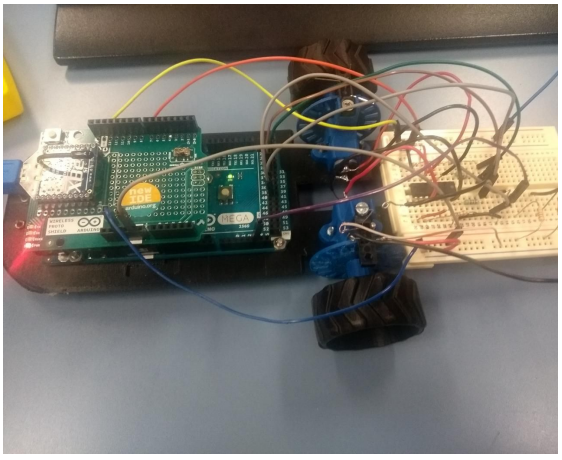

Figura 3 - Robot, parte física terminada.

Cuando probamos los interruptores con el código que escribimos en c++ nos dimos cuenta que uno de los dos no funcionaba correctamente, por lo que tuvimos que cambiarlo y continuamos las pruebas. Una vez que nos cercioramos que funcionaron correctamente realizamos los encoders y dimos paso a programar el PID.

Para la programación del PID, tuvimos que realizar distintas pruebas ya que los parámetros diferenciador, potenciador e integrador debían ser particulares a los motores de tal forma que el ajuste necesario se lograra. Esto tuvo que ajustarse a prueba y error pero logramos implementarlo de tal forma que nuestro robot avanzó correctamente y a la velocidad esperada

Para la comunicación con los Xbees, fue necesario recordar lo que habíamos realizado en una de las práctica anteriores. Logramos establecer comunicación y transmisión de mensajes con ayuda del programa XCTU (ver Figura 4), de tal forma que comprobamos que el movimiento del robot era de acuerdo a lo esperado dados los mensajes.

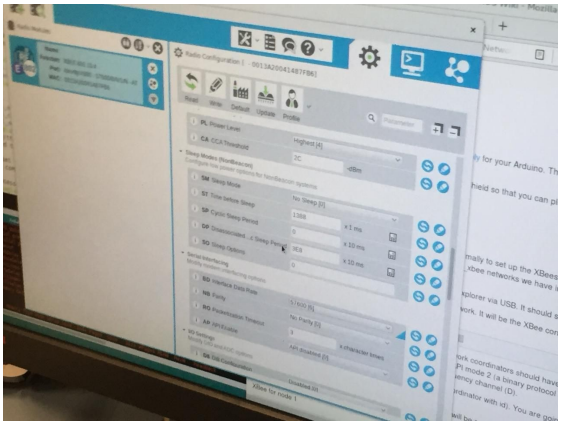

Figura 4 - Configuración de los Xbees usando XCTU.

En cuanto a la ruta que el robot debía seguir dado que varios obstáculos podían estar presentes, desarrollamos un algoritmo que contempla las dimensiones del robot y de los obstáculos. Inicialmente creímos que sería bueno implementar el algoritmo de búsqueda informada A* generando puntos aleatorios en el campo, pero decidimos generar uno distinto, que contempla las distancias entre el robot y los obstáculos dadas rutas tentativas.

Fue desarrollado inicialmente en el lenguaje matlab y posteriormente lo intentamos traducir a python, pero dadas las lógicas de estos lenguajes fue complicado realizarlo. De cualquier forma, en matlab creamos distintas gráficas que permiten visualizar los resultados. En los archivos se puede observar que el resultado dadas las restricciones era el esperado, obtener una ruta que llevara al robot del punto inicial al final sin chocar con los obstáculos.

Por otro lado, cuando se empleó Ros para poder comunicarse con los Xbee, fue necesario analizar el repositorio que nos proporcionó el profesor, adaptar la idea que queríamos y generar los tópicos necesarios creando los archivos de python para poder lograrlo con ayuda de lo que habíamos realizado en prácticas anteriores. En la Figura 5 se puede observar como se tuvo éxito al establecer la comunicación entre Ros y los Xbee, y en la 6 se puede observar cómo se leían los datos de prueba dado cierto visor.

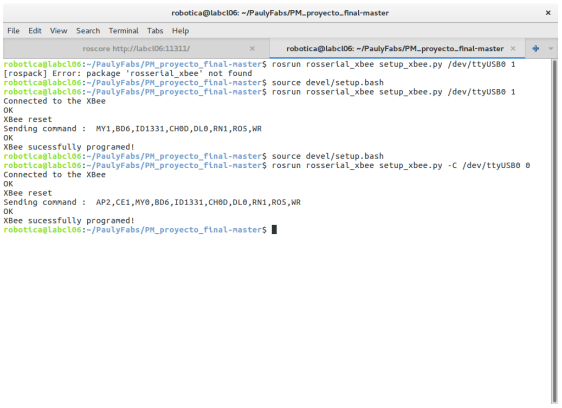

Figura 5 - Conexión exitosa de Xbees con Ros.

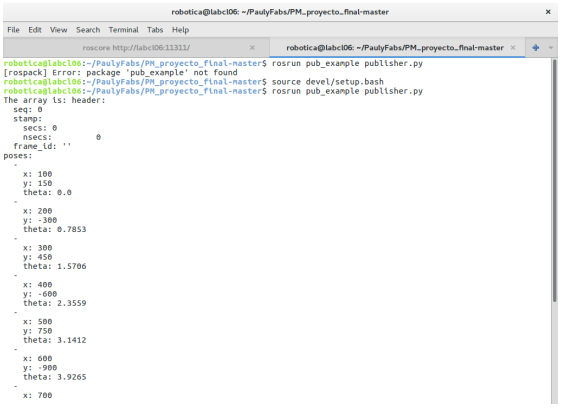

Figura 6 - Despliegue de datos para lectura.

Así que, dado lo que realizamos nuestro robot estaba listo para poder moverse dentro del campo dadas las posiciones del punto final, el destino y los obstáculos. Incluyendo el PID, la conexión entre Xbees y ros, el cálculo de una ruta que sirviera para llegar al destino, el uso y transformación de la información del visor para poder generar los movimientos.

Una vez que tuvimos cada una de estas partes listas y las juntamos, procederíamos a probar como se comportaba el robot, con todo el trabajo por detrás, en el campo de competición. A pesar de que el proyecto compiló adecuadamente en los laboratorios donde estuvimos trabajando, el proyecto no compiló en las computadoras del campo de competición debido a un problema de directorios.

V. CONCLUSIONES

Jean Paul Virueña Iturriaga

Fue muy satisfactorio reforzar lo aprendido durante la práctica. También fue muy interesante poder ver que todos los objetivos de las prácticas pasadas se podían juntar para crear un robot casi desde cero. Fue un poco decepcionante no poder ver a nuestro robot funcionando debido a que no tuvimos tiempo de probarlo con todas las partes unidas, pero me quedo contento al saber que funcionaban por separado y tener la experiencia en conjunto con mi compañero en un ambiente profesional y comunicativo.

Fabián Orduña Ferreira

A lo largo del semestre y de las distintas prácticas logré aprender diversas cosas que fueron aplicadas en esta última práctica, el proyecto final. Además, reforzamos distintos conocimientos. Fue en su momento complicado ya que requerimos mucha paciencia para poder proceder con las distintas tareas que requería el proyecto por los problemas que se presentaron. A pesar que no pudimos probar en vivo debido a problemas de compilación, algo que me llena de orgullo es que pudimos, mi compañero Paul y yo, concluir todas las partes del proyecto de buena forma, dadas las pruebas unitarias que realizamos, en un ambiente cordial y coordinado.

VI. ROLES

Para el desarrollo de las actividades, de las que mostramos previamente los resultados, ambos miembros del equipo colaboramos de forma conjunta. En lo que a este documento concierne, Paul se enfocó en mayor proporción al abstract, a la introducción y al marco teórico, mientras que

Fabián se enfocó más al desarrollo y a los resultados.

VII. FUENTES DE CONSULTA

[0] Wikipedia. "Robot." Wikipedia. April 23, 2019. Accessed April 25, 2019. https://en.wikipedia.org/wiki/Robot.

[1] Arduino. (n.d.). Arduino Mega 2560 Rev3. Retrieved January 31, 2019, from https://store.arduino.cc/usa/arduino-mega-2560-rev3

[2] Arduino. (n.d.). What is Arduino? Retrieved January 31, 2019, from https://www.arduino.cc/en/Guide/Introduction

[3] Chegg Study. "Definition of Dc Motor." Chegg.com. Accessed April 11, 2019. https://www.chegg.com/homework-help/definitions/dc-motor-2.

[4] Engineering 360. "DC Motors Information." Accessed April 11, 2019. https://www.globalspec.com/learnmore/motion_controls/motors/dc_motors.

[5] Geekbot Electronics. "Motores De DC." April 21, 2015. Accessed April 11, 2019. http://www.geekbotelectronics.com/motores-de-dc/.

[6] Wikibooks. (n.d.). Robótica/Puente H. Retrieved March 14, 2019, from https://es.wikibooks.org/wiki/Robótica/Puente_H

[7] Recursostic.educacion.es. "Sistemas De Control." Accessed April 11, 2019. http://recursostic.educacion.es/secundaria/edad/4esotecnologia/quincena11/4quincena11_contenidos_2b.htm.

[8] Picuino. "Controlador PID." Accessed April 11, 2019. https://www.picuino.com/es/arduprog/control-pid.html.

[9] RS Components. "Slotted Optical Switches." Slotted Optical Switches I RS Components. Accessed May 24, 2019. https://uk.rs-online.com/web/c/displays-optoelectronics/optocouplers-photodetectors-photointerrupters/slotted-optical-switches/.

[10] Encoder Product Company. "¿Qué Es Un Encoder?" Encoder. June 16, 1970. Accessed May 24, 2019. http://encoder.com/blog/encoder-basics/que-es-un-encoder/.

[11] Xbee.cl. "¿Qué Es XBee?" XBee.cl - Comunicación Inalámbrica Para Tus Proyectos. Accessed May 24, 2019. https://xbee.cl/que-es-xbee/.

[12] Aprendiendo Arduino. (n.d.). ZigBee/ XBee. Retrieved March 14, 2019, from https://aprendiendoarduino.wordpress.com/tag/xctu/

[13] ROS.org. "Topics." Ros.org. Accessed April 25, 2019. http://wiki.ros.org/Topics.

[14] ROS.org. "About ROS." Ros.org. Accessed April 25, 2019. https://www.ros.org/about-ros/.

[15] ROS.org. "Roscore." Ros.org. Accessed April 25, 2019. http://wiki.ros.org/roscore.

[16] ROS.org. "Nodes." Ros.org. Accessed April 25, 2019. http://wiki.ros.org/Nodes.

[17] ROS.org. "ROS." Ros.org. Accessed April 25, 2019. https://wiki.ros.org/.

[18] Clearpath Robotics. "ROS 101: Creating a Publisher Node." Clearpath Robotics. October 12, 2017. Accessed April 25, 2019. https://www.clearpathrobotics.com/blog/2014/09/ros-101-creating-node/.

[19] ROS.org. "Writing a Simple Publisher and Subscriber (C)." Ros.org. Accessed April 25, 2019. http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber(c).