Proyecto Final Principios de Mecatrónica

Leslie P. Brenes, Paola Del Hierro, Luis E. Delfín, Diego Villafuerte

Resumen—Presentamos la elaboración de un robot diferencial con dos motores, esto lo realizamos mediante la implementación de rotación diferencial, un Arduino con Xbee y programación en ROS, un control PID. El objetivo final fue realizar un recorrido de un punto A a B mediante el control de las velocidades, este recorrido se realizó con y sin obstáculos.

1. INTRODUCCIÓN

En el semestre, aprendimos sobre el funcionamiento de un microcontrolador, luego nos enseñaron como funcionan los sistemas de percepción y acción, así a partir de un microcontrolador, en nuestro caso Arduino, queremos poder generar un modelo para recibir información a través de sensores, y luego, poder interpretar esa información para entregar un resultado. Aunque ya se habían abordado estas problemáticas en distintas prácticas a lo largo del semestre, es importante poder juntar todas las partes para generar un sistema mecatrónico funcional.

En nuestro proyecto en específico, mediante un carrito compuesto por dos motores independientes en cada rueda, nuestro objetivo fue generar una trayectoria de un punto A a B. Esta trayecteria se realizó de dos formas: con y sin obstáculos.

Aunque parecería una tarea trivial, en realidad poder modelarlo no es tan sencillo: con Arduino recibimos posiciones de forma inalámbrica mediante un XBee y tópicos de ROS, por otro lado para poder generar el movimiento de los motores es necesario implementar un control PID para controlar la velocidad de cada uno, finalmente para poder determinar la lógica del movimiento (velocidad de cada motor) es necesario programar un algoritmo y regresar la información al Arduino.

Esto presenta problemáticas de distinta complejidad, primero poder hacer el correcto montaje de los motores y la placa de Arduino, después poder configurar correctamente los componentes, y finalmente la complejidad de recibir las posiciones necesarias para que mediante un algoritmo se trace la ruta correcta mediante el cálculo de las velocidades en cada uno de los motores.

Otra cosa a considerar es que este sistema es totalmente autónomo, es decir, no requiere de ningún tipo de intervención intermedia por parte de un humano, toda la información se recibe, procesa y envía

En el presente documento, primero presentaremos una breve explicación del funcionamiento de cada componente, posteriormente explicamos como se configuraron y finalmente presentamos los resultados y conclusiones obtenidos.

2. MARCO TEÓRICO

Dado que este proyecto es de alguna forma, la unión de las prácticas anteriores, solo mencionaremos brevemente los dispositivos y conceptos aplicados:

- **Arduino**: Contiene un microcontrolador AVR que utilizaremos para controlar el movimiento de los motores y recibir la señal de las posiciones.
- **XBee**: Es un componente modular que permite la transmisión de información inalámbricamente.
- **Controlador PID**: El controlador PID (Amplificador, Integrador y Derivador) nos permite corregir la señal que se enviará al motor para determinar su velocidad.
- **ROS**: Es un lenguaje de programación para robots, en este caso mediante *rosbags* para obtener la información de las posiciones del carrito, los obstáculos y el destino final.
- **Trazar trayectorias**: Finalmente, para trazar trayectorias, fue importante tener conocimientos básicos de trigonometría y geometría para calcular los parámetros de ángulos y distancias necesarios para determinar la velocidad de cada motor y con ello, generar el movimiento hacia el punto deseado.
- **Pila LIPO**: LIPO significa que es de polímero de litio, la cual es una pila recargable. Esto nos permitió poder usar el Arduino sin cables.

Para la implementación del algoritmo que determina la trayectoria, utilizamos el algoritmo A*, el cual determina la ruta de menor costo entre un nodo origen y objetivo.

3. DESARROLLO

Primero, generamos la configuración física del carrito, el cual consistió en conectar la placa del Arduino en una montura con las ruedas. También en esta etapa, montamos el Xbee.

Para la configuración del Xbee, al igual que en la práctica que realizamos anteriormente, utilizamos el protocolo UART y el programa XCTU. Esto permite que mandemos la comunicación al programa cargado en el Arduino.

En el Arduino configuramos los pines correspondientes a los motores y a las entradas de voltaje para así poder corregir el voltaje que les llega a través de un controlador PID, el cual lo que hace es amplicar, integrar y derivar la señal tal que cuando usemos una pila externa o haya fricción en las ruedas, en poco tiempo se estabilice la señal (en nuestro caso, $\sim$5V) y así, poder controlar la velocidad y movimiento

del carrito (a partir de la velocidad en cada rueda). Es importante mencionar que se requirieron calibraciones del PID diferentes para cada motor.

Una vez calibrado el PID, generamos las instrucciones correspondientes a los movimientos izquierda, derecha, avanzar de frente y parar. Lo que estas instrucciones hacen, es determinar el voltaje objetivo para cada una de las ruedas. Para que el Arduino pueda recibir estas instrucciones desde un dispositivo externo, primero hicimos pruebas con Xbee para que mandara caracteres específicos para cada instrucción, de esta forma pudimos probar el funcionamiento del PID, de la configuración física de los motores y la comunicación entre el XBee receptor y transmisor.

La siguiente etapa del proyecto consistió en poder generar la ruta para llegar del punto A a B. Para esto, requerimos de un sistema de visión el cual envía a través de tópicos en ROS un vector (X, Y, θ) con las coordenadas obtenidas para los obstáculos, el carrito y el objetivo. Estas coordenadas se actualizan constantemente. Además, la información de cada tipo de objeto se envío en *rosbags* y desde una computadora específica que se encuentra en el laboratorio de Robótica, por lo que fue necesario primero hacer la lectura correcta de los datos para poder procesarlos y generar una ruta que se tradujera en instrucciones puntuales para el carrito. En esta sección fue esencial contar con la actualización de la posición del carrito para poder calibrar el movimiento y determinar con precisión si se encontraba en la región objetivo. Para la lectura de los obstáculos, determinamos una lectura de un máximo de 4 obstáculos, una meta, y un robot.

Finalmente, implementamos un algoritmo que regresa un vector de posiciones a partir de la posición actual del carrito, la del destino y la posición de los obstáculos, para esto utilizamos el algoritmo A*, el cual determina la ruta de menor costo entre un nodo origen y objetivo. Para poder utilizar este algoritmo, primero fue importante dividir la región en una cuadrícula (el tamaño lo fuimos cambiando, pero a mayor tamaño de bloque, menor número de nodos) y ya con la cuadrícula lista, descartamos aquellos nodos que contuvieran algún obstáculo. Para poder tener éxito en esto, es importante cosiderar un radio alrededor de un punto, este radio lo consideramos un margen para que no haya colisión entre los objetos. Otra cosa importante a considerar en nuestro algoritmo, es que una vez generado el vector de posiciones a las cuáles deberá dirigirse el carrito, utilizamos una función *ejecutaPlan()* la cual, para cada elemento de la ruta (un arreglo con dos elementos X,Y) determina la distancia a recorrer y el ángulo hacia el cuál deberá girar el carrito. Con estos datos, entra a otras subrutinas. En caso de que el carrito no se encuentre alineado a la dirección a la cual debe dirigirse, entra a una rutina para girar a la izquiera o derecha, dependiendo el ángulo. En estas subrutinas, compara la diferencia entre el ángulo de dirección del carrito y el ángulo al que necesita dirigirse, mientras sea mayor a un margen, manda la instrucción correspondiente a giro izquiera o derecha, dependiendo el caso. Cuando la diferencia entre los ángulos es menor al margen, entonces avanza en línea recta la distancia entre la posición del carrito y el punto al cual dirigirse, luego, enviá la intrucción de paro.

4. RESULTADOS

Primero, utilizamos dos motores DC en ruedas de 42 mm de diámetro, cuya distancia entre ellas es de 115 mm. Estos motores los conectamos a un Arduino Mega 2560. Además para no depender de la corriente de la computadora, utilizamos una batería LIPO de 7V.

Las constantes que utilizamos para el controlador PID fueron:

Constante	Motor 1	Motor 2
K_P	.004	.001
K_I	.005	.005
K_D	.01	.01

Para las instrucciones en el programa Arduino, utilizamos el siguiente switch case con los siguientes valores de voltaje (donde setPoint 1 y 2 corresponden a cada motor):

```
  if(xbee.available()){
        senal = xbee.read();
        switch(senal) {
        case '0' :
          xbee.println("Entre a cero");
          digitalWrite(I2, HIGH);// Arrancamos
          digitalWrite(I1, LOW);
          digitalWrite(I3, HIGH);
// Arrancamos
          digitalWrite(I4, LOW);
          setPoint1 = 2;
          setPoint2 = 2;
          break;
        case '1' :
          digitalWrite(I2, LOW);// Arrancamos
          digitalWrite(I1, HIGH);
          setPoint1 = 2;
          setPoint2 = 2;
          break;
        case '2':
          digitalWrite(I3, LOW);
// Arrancamos
          digitalWrite(I4, HIGH);
          setPoint1 =  2;
          setPoint2 = 2;

          break;
         case '3':
          setPoint1 = 0;
```

Luego, modificacmos el código del ejemplo contenido en el Repositorio proporcionado por el profesor, para la suscripción a los tópicos de ROS, y creamos un código en python para leer los datos y ejecutar el algoritmo.

Utilizamos un tamaño de 200mm para la longitud de los bloques, un margen de 100mm para evitar colisiones.

Finalmente, notamos que la parte que requirió de más creatividad por nuestra parte fue el algoritmo y aunque parecía que poder coordinar las distancias y ángulos que requeríamos, tuvimos muchos problemas de coordinación respecto a las convenciones usadas para la orientación de los ejes X, Y y la medición de los ángulos. También notamos que en el futuro, podríamos intentar generar rutas más suaves usando otro tipo de algoritmos que no dependan de una

malla dada por nosotros sino de la información que está recibiendo el sensor de visión.

5. CONCLUSIONES

Finalmente, podemos resaltar que fusionar cada una de las prácticas en este proyecto requirió de mucho más tiempo del planeado, además, notamos que realizar pruebas en realidad debió haberse realizado desde una etapa más temprana para evitar la práctica de "prueba y errorz planear con más detalle. Sin embargo, también tuvimos problemas de convención respecto a los ejes y los ángulos, lo cual nos enseñó que en un futuro sería conveniente hacer más preguntas antes de comenzar a ejecutar. Como último comentario, este proyecto podría llevarse más allá probando diferentes tipos de aproximaciones para llegar al objetivo, por ejemplo actualizar constantemente la infromación, intentar otro tipo de algoritmos con mallas con menos nodos, con más nodos, intentar que los obstáculos sean dinámicos, mejorar los parámetros del PID para que sean movimientos mucho más precisos, etc.

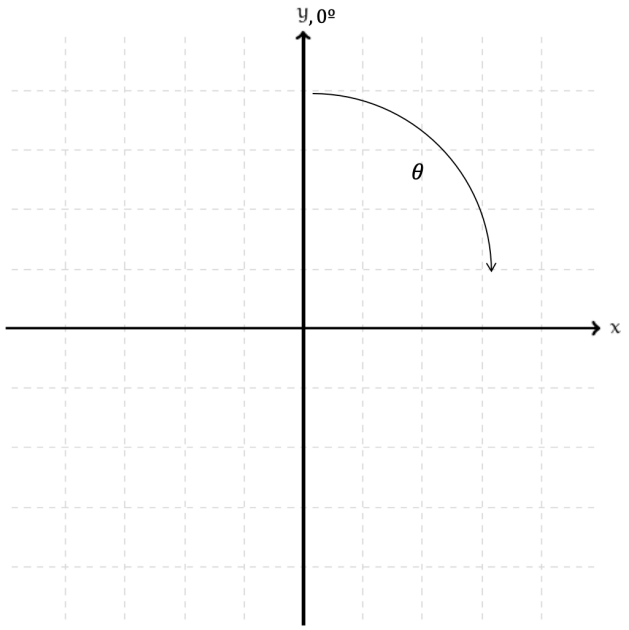

Figura 1. Convención usada

6. FUENTES CONSULTADAS

REFERENCIAS

[1] D. G. Alciatore, *Introduction to Mechatronics and Measurement Systems*. McGraw-Hill Education, 2011.
[2] N. S. Nise, *Control Systems Engineering, 4th Edition*. Wiley, 2003.