Proyecto Final

Laboratorio de Principios de Mecatrónica

Javier Montiel González

C.U. 159216

Andrea González Cardoso

C.U. 157961

Diego Villalvazo Sulzer

C.U. 155844

Docente: **EDGAR ALEJANDRO GRANADOS OSEGUEDA**

Resumen— El proyecto consistió en implementar un robot diferencial con dos actuadores cuyo propósito era llegar de una posición inicial a una posición final preestablecida. El trayecto que recorrió nuestro robot fue con y sin obstáculos. Para lograr esto, usamos una combinación de ROS (*Robot Operating System*), Arduino y *XBee* como el protocolo de comunicación.

1 Introducción

El propósito de este proyecto fue implementar un sistema mecatrónica utilizando las herramientas vistas a lo largo del semestre. En las prácticas pasadas, tuvimos que implementar protocolos de comunicación, controladores para motores eléctricos, y aprendimos a usar ROS. Todo eso con el propósito de prepararnos para poder elaborar este proyecto final.

El objetivo de este proyecto es construir un robot que fuera de un punto inicial a un punto final, con y sin obstáculos. Para esto, se tuvo que implementar un protocolo de comunicación XBee para que pudiéramos comunicar Arduino, ROS y el robot. ROS se encargaría de recibir la información de la cámaras, y con base en esa información calcular la velocidad de los dos motores para poder ir en la dirección correcta para evitar los obstáculos y dirigirse al punto final. Arduino se encargaría de mandar las señales correctas a cada uno de los motores. El controlador PID permitiría regular el robot para que este alcance el estado de salida deseado, es decir la velocidad deseada en cada motor.

2 Marco Teórico

Un robot es una máquina automática programable capaz de realizar determinadas operaciones de manera autónoma, por ejemplo, moverse, manipular objetos y realizar trabajos a la vez que interacciona con su entorno. Para que un robot pueda interactuar con su entorno, debe tener *sensores* y *actuadores*. Un sensor es dispositivo que capta magnitudes físicas u otras alteraciones en su entorno, y un actuador es un dispositivo mecánico cuya función es proporcionar fuerza para mover o "actuar" otro dispositivo mecánico. En particular, para nuestro robot, el sensor es una cámara que capta la posición del robot. Los actuadores son los dos motores eléctricos en el robot que provocan que se muevan las llantas.

Para lograr que el robot se mueva, utilizamos dos **Motores DC** (por sus iniciales en inglés *Direct Current*). Un motor DC se define como una máquina que convierte energía eléctrica en energía mecánica, provocando un movimiento rotatorio, gracias a la acción de un campo magnético.

Figura 1. Motor DC

ROS (Robot Operating System) es una herramienta para el desarrollo de software principalmente enfocado hacia la robótica. Provee la funcionalidad de un sistema operativo (pero no es únicamente uno, como se podría intuir por el nombre) en un clúster heterogéneo, abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. En general, ROS es una herramienta que facilita la implementación de software para robótica. A grandes rasgos, ROS se puede dividir en tres partes: **nodos, mensajes y tópicos**. Los nodos son procesos que nos permiten hacer alguna acción. Están diseñados para ser módulos independientes que interactúen con otros nodos, mandando o recibiendo mensajes. Algunos nodos proveen información a otros, por ejemplo la información que se adquiere de una cámara o un

sensor, y se dice que el nodo "publica" información. A esa información se le llama tópico.

Por otra parte, los módulos XBee son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo de red llamado IEEE 802.15.4 para crear redes FAST POINT-TO-MULTIPOINT (punto a multipunto); o para redes PEER-TO-PEER (punto a punto). Fueron diseñados para aplicaciones que requieren de un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible. XBee está basado en el protocolo Zigbee. En términos simples, los XBee son módulos inalámbricos fáciles de usar. Zigbee es una alianza y un estándar de redes MESH de eficiencia energética y de costos. XBee utiliza el estándar Zigbee y lo agrega y envuelve en un pequeño empaque elegante.

3 Desarrollo y resultados

3.1 El primer paso fue armar el robot. Los materiales necesarios para armar el robot fueron los siguientes:

Cantidad	Material
1	Base hecha con impresora 3D
2	Llantas hechas con impresora 3D
1	Arduino Mega 2560
1	Arduino wireless proto shield
2	XBee S1
2	Motor DC
2	Hub para motor hechos con impresora 3D
2	Opto Interrupter ITR8102
2	Encoder de 18 orificios
1	Protoboard
1	Batería LIPO

Tabla 1. Materiales para el robot.

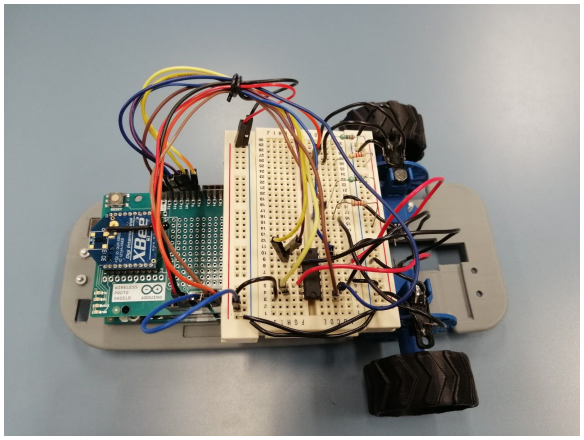

Figura 2. Robot armado

En la figura 2 podemos observar el robot terminado, con todos sus componentes funcionales.

3.2 Una vez armado el robot, hicimos las conexiones al Puente-H para que el motor pudiera ser controlado por el Arduino. Se usó el siguiente diagrama para elaborar el puente H:

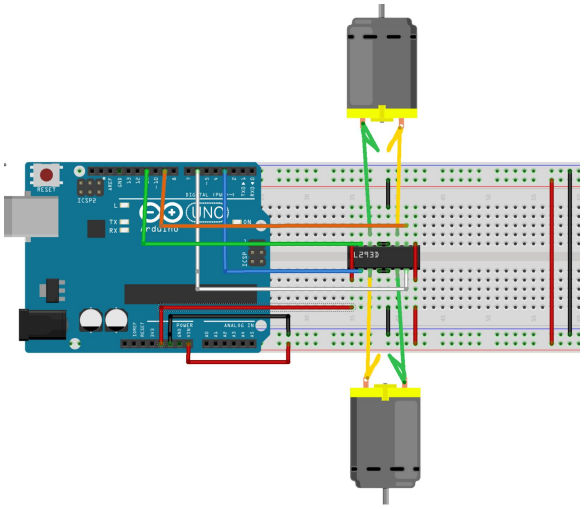

Figura 3. Diagrama de puente H.

Cabe destacar que se alambró de forma casi idéntica, sólo los pines del Arduino fueron asignados de manera diferente. De igual manera, se usó el circuito integrado SN75441ONE en vez del que se muestra en el diagrama, pero tienen un funcionamiento equivalente.

Posteriormente, se hizo la configuración de los optointerruptores.

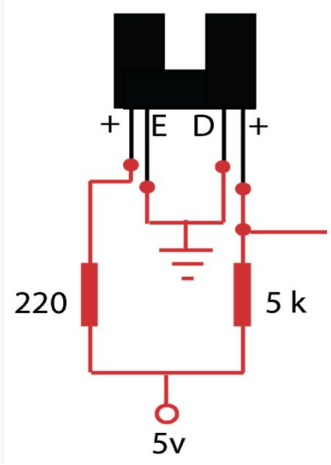

Figura 4. Configuración de los optointerruptores.

En la figura 4 se muestra la configuración de los optointerruptores, que la salida fue alambrada a dos entradas analógicas del Arduino (A0 & A1). El propósito de los optointerruptores es medir la velocidad de las llantas, para que después el controlador PID pudiera estabilizar la velocidad de las llantas.

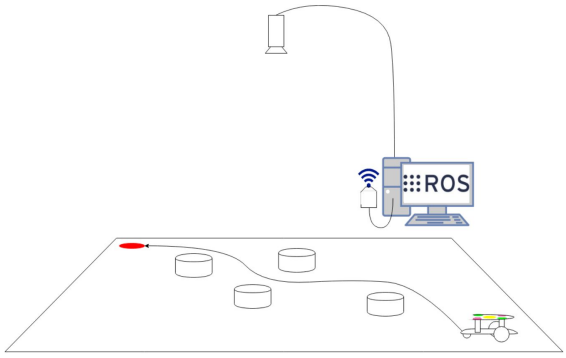

Figura 5. Diagrama de la configuración del experimento.

Como se mencionó en la introducción, el objetivo del proyecto fue que el robot fuera de un punto inicial a un punto final, esquivando cualquier obstáculo entre dichos puntos. Para lograr esto, implementamos el algoritmo A*. Este algoritmo nos regresa una ruta a seguir que después una computadora con ROS interpretaría. La figura 5 es un ejemplo ilustrativo del robot esquivando obstáculos para llegar a la meta (punto rojo).

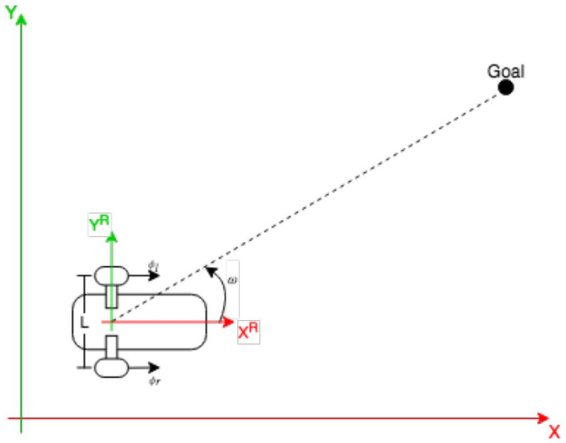

Figura 6. Diagrama que muestra las variables tomadas en cuenta por ROS.

Una vez que tenemos la ruta a seguir, y dependiendo de las coordenadas (X, Y, θ), variaría la velocidad de ambos motores para lograr un movimiento diferencial y modificar la dirección en la que se esté moviendo el robot.

Se implementó un controlador PID para que el movimiento de los motores convergiera y la trayectoria del robot pudiese controlarse de manera más certera.

```
#inc
lude
<ros
.h>
        #include <PID_v1.h>
        #include <std_msgs/String.h>
        #include <SoftwareSerial.h>


        #define E1 13  // Enable Pin
        for motor 2
        #define I1 12      // Control
        pin 1 for motor 2
        #define I2 11      // Control
        pin 2 for motor 2
```

```
#define E2 10  // Enable Pin for motor 1
#define I3 9      // Control pin 1 for motor 1
#define I4 8       // Control pin 2 for motor 1


unsigned long tiempo;
int contM1=0;
int contM2=0;
int valueAntM1 = 0;
int valueAntM2 = 0;
int valueM1=0;
int valueM2=0;


// Direccion (1 adelante, -1 atras)
int dirM1= 1;
int dirM2 = 1;


double velM1=0;
double velM2=0;


//Motor derecho
double kpM2 = .9;
double kiM2 = .7;
double kdM2 = 0.15;


//Motor izquierdo
double kpM1 = .9;
double kiM1 = .6;
double kdM1 = 0.1;


//ErrorM2
double errM2 = 0;
double sumErrorM2= 0;
double errorAntM2 = 0;


//ErrorM1
double errM1 = 0;
double sumErrorM1 = 0;
double errorAntM1 = 0;


double pidM2 = 0;
double pidM1 = 0;


//correccion vel motores
int corrM2 = 0;
int corrM1 = 0;


void setup() {
 //Inicializamos
       for (int i =8 ; i<14 ; i++)
         pinMode(i, OUTPUT);


              //Activamos  los motores
```

```
            digitalWrite(E1, HIGH);
            digitalWrite(E2, HIGH);


                pinMode(A0, INPUT);//Inicializamos la lectura de Motor1
                pinMode(A1, INPUT);//Inicializamos la lectura de Motor2

       Serial.begin(9600);
       tiempo = millis();
}


void loop()
  {


                    valueM1= digitalRead(A0);
                    valueM2= digitalRead(A1);



           if(valueM1==1 && valueAntM1==0){
       contM1= contM1+1;
     }
     valueAntM1=valueM1;


           if(valueM2==1 && valueAntM2==0){
       contM2= contM2+1;
     }
     valueAntM2=valueM2;



           if(millis() >= tiempo+250){


   //PID


     // Error es velocidad deseada - velocidad detectada en orificios/segundo
        errM2 = velM2 - 4*contM2*dirM2;
        errM1 = velM1 - 4*contM1*dirM1;


   // Error integral
   sumErrorM2 += errM2;
   sumErrorM1 += errM1;


   // PID completo
      pidM2 = errM2*kpM2 + (errM2 - errorAntM2)*kdM2 + sumErrorM2*kiM2;
      pidM1 = errM1*kpM1 + (errM1 - errorAntM1)*kdM1 + sumErrorM1*kiM1;


   // Actualizar error previo
   errorAntM2= errM2;
```

```
    errorAntM1= errM1;


    // Ajustar salida
     corrM2  =  min(abs(pidM2),
255);
      corrM1=  min(abs(pidM1),
255);

    if (pidM2 >= 0){
      analogWrite(I1, corrM2);
      analogWrite(I2, 0);
      dirM2 = 1;
    }else{
      analogWrite(I1, 0);
      analogWrite(I2, corrM2);
      dirM2 = -1;
    }

    if (pidM1 >= 0){
      analogWrite(I4, corrM1);
      analogWrite(I3, 0);
      dirM1 = 1;
    }else{
      analogWrite(I4, 0);
      analogWrite(I3, corrM1);
      dirM1 = -1;
    }


      contM1=0;
      contM2=0;
                tiempo= millis();
            }
        }
```

Para que se pudieran comunicar los sensores y los actuadores, es decir, para que nuestro programa en Arduino pudiera saber la posición del robot y pudiera mandar, mandar la información a ROS para que calcule la mejor ruta al destino, y le comunique eso a los motores, tuvimos que implementar un protocolo de comunicación XBee. Desde la computadora con Linux corriendo ROS, se conectaría un XBee, y otro en el robot. Entonces, por medio de ROS, usando nodos, mensajes y tópicos se lograrían comunicar para que el robot pudiese hacer los ajustes correspondientes en la velocidad y dirección para llegar a la meta.

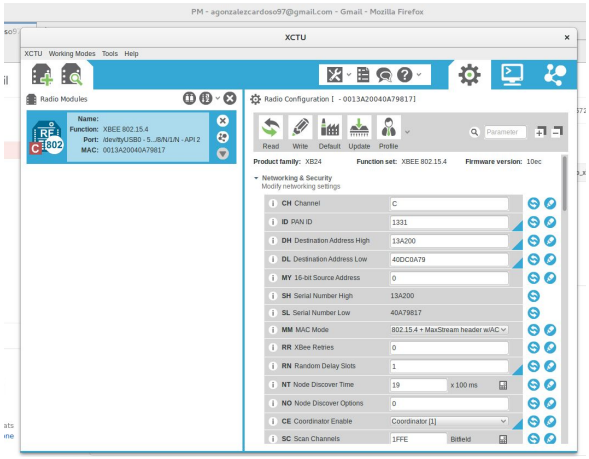

Figura 7. Configuración de un Xbee.

En la figura 7 se muestra la configuración que se implementó en los dos XBees para que se pudiesen comunicar.

A pesar del gran esfuerzo y el tiempo dedicado al proyecto, no se logró implementar la funcionalidad completa del robot. ROS es una gran herramienta para sistemas distribuidos, y nuestro robot se podría pensar como uno, ya que cada nodo en ROS se encargaba de diferentes partes de la funcionalidad. Todos los nodos por separado funcionaron, pero a la hora de unirlos e interconectar cada uno de ellos, tuvimos conflictos. El robot no siempre se comportaba de la manera

que debía, y la conexión de los XBees fue muy deficiente, ya que no siempre se conectaba al XBee que debía, ya que en los laboratorios había mucha gente trabajando con más XBees, entonces existía un grado alto de interferencia.

4 Conclusiones

Andrea: El proceso, construcción y realización de este proyecto constituyó todo un reto para este curso. Fue un proyecto que juntó todos los temas y conocimientos que se fueron adquiriendo a lo largo del curso, además de que promovió la búsqueda de temas nuevos e información extra para cualquier imprevisto encontrado en el camino. A pesar de no poder ejecutar el resultado final del proyecto, creo que construir lo que logramos fue un trabajo que requirió mucha dedicación.

Como proyecto final, creo que es muy completo, porque como mencioné anteriormente, requiere la consolidación y dominación de todos los temas vistos en prácticas anteriores. Sin embargo fueron los detalles extra, los imprevistos a la hora de ejecución (principalmente la conexión entre los XBee), y la unión de todos los elementos lo que evitaron que se completara el proyecto. Sin embargo la experiencia y el conocimiento aprendido, así como una gran experiencia de trabajo en equipo hicieron que este proyecto fuera muy interesante.

Diego: A lo largo de esta práctica, necesitamos juntar todos los conocimientos y herramientas que aprendimos a lo largo del semestre, tanto en la teoría como el laboratorio.
Yo creo que fue un trabajo muy completo, ya que armamos un robot desde cero, tuvimos que programar en muchos lenguajes de programación diferente, y de hecho es la primera vez en el laboratorio que juntamos tantas cosas y conceptos para lograr un sistema complejo.
Siento que como un último trabajo para esta materia, refuerza todos los conocimientos que aprendimos, ya que para poder llevarlos a la práctica es necesario entender bien los conceptos.

Javier: En este proyecto se utilizaron todos los conocimientos adquiridos a lo largo del semestre. Por otro lado, no se logró conseguir la implementación deseada. Entre las causas que llevaron a esto fue que varias de las prácticas no fueron concluidas. Así pues, solo se lograron implementar las funcionalidades de los motores diferenciales y el PID para el robot. Nunca se logró la comunicación con los Xbee, lo que consumió bastante tiempo y al final no se pudo concluir con lo demás. Creo que hizo falta una mejor distribución de las tareas en el equipo y considero que se pudo haber realizado un mejor proyecto.

Bibliografía

[1]. https://es.wikipedia.org/wiki/Sistema_Operativo_Rob%C3%B3tico

[2]. https://www.oreilly.com/library/view/ros-robotics-by/9781782175193/ch01s07.html

[3]. https://learning.oreilly.com/library/view/ros-robotics-by/9781782175193/ch01s06.html

[4]. https://es.wikipedia.org/wiki/Motor_de_corriente_continua