PRÁCTICA 8:
CONTROL DE ROBOTS MÓVILES

Autor: Pascual Andrés Carrasco Gómez Email: pascargo@inf.upv.es
Autor: Javier Moraga Matoque Email: jamoma1@inf.upv.es

Mecatrónica, Grado en ingeniería informática, Escuela técnica Superior de Ingeniería Informática

Resumen

En esta práctica vamos a mostrar el resultado de la construcción de un robot Lego Mindstorms NXT, una vez construido el robot vamos a comentar las estrategias que hemos escogido para realizar dos competiciones que consisten en un limpiador y en un combate de sumo, finalmente comentaremos el código que implementa dichas estrategias implementado en RobotC.

Palabras Clave: Lego, Mindstorms, NXT, ROBOTC, estrategia, sensores, encoders, actuadores, programación NXT.

1 DESCRIPCIÓN DEL ROBOT

Nombre: Machacasaurio.
En la construcción del robot no nos hemos basado en el montaje básico que nos proporciona el manual del Lego Mindstorms NXT, hemos preferido buscar una configuración con unas características concretas como son principalmente que el robot no tenga mucha altura y que su peso se concentre aproximadamente en el centro, de esta forma obtenemos un centro de gravedad que nos proporciona bastante estabilidad y por ello es bastante difícil que nuestro robot vuelque.
A continuación vamos a mostrar imágenes de diferentes ángulos de nuestro robot indicando sus partes principales y sus características:

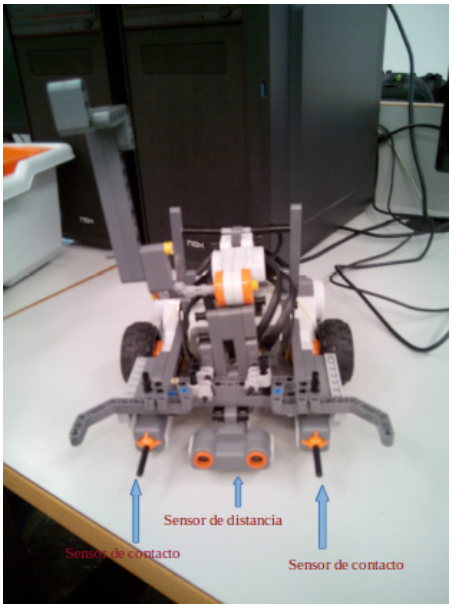

Imagen 1: Sensores en la parte frontal del robot.

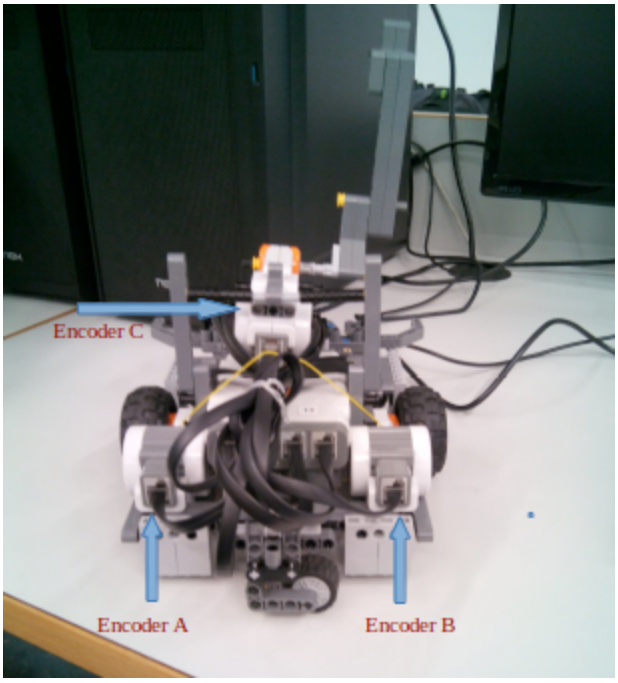

Imagen 2: Encoders del robot.

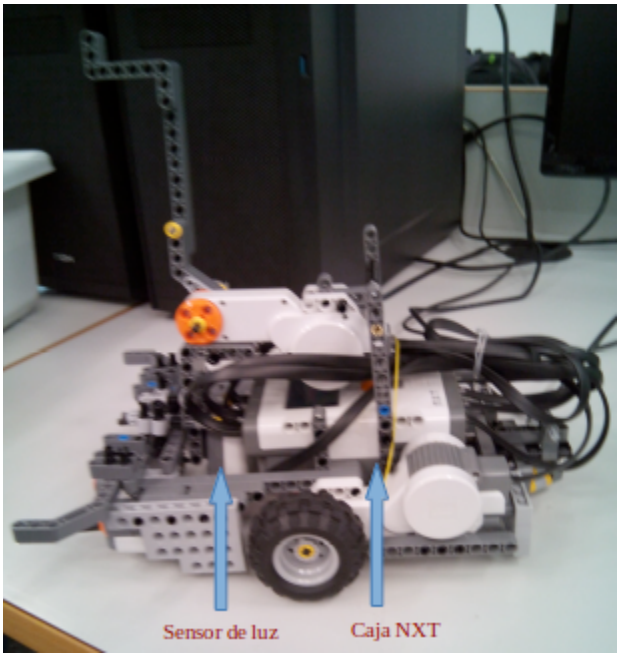

Imagen 3: Parte lateral del robot.

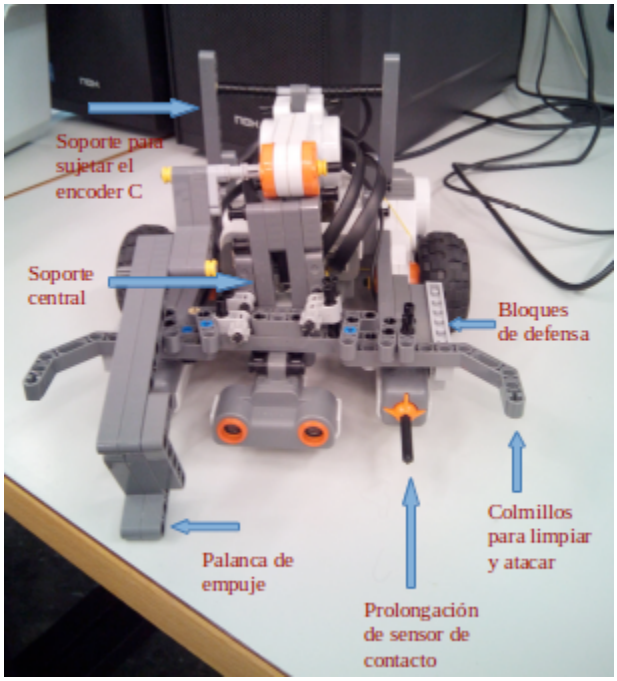

Imagen 4: Características del robot.

NOTA: La construcción del robot es muy robusta, no tuvimos problemas de perdida de piezas en las peleas de la competición de sumo y sujetaba perfectamente las bolas hasta sacarlas fuera del circuito en la competición de robot limpiador.

2 ESTRATEGIAS DESARROLLADAS

Competición robot limpiador

Imagen 5: Configuración del robot para la competición de robot limpiador.

La estrategia en esta competición se centra en como se tiene que mover el robot para intentar visualizar el circuito entero en menos de un minuto ya que no sabemos a priori donde van a estar situadas las bolas.
Nuestra estrategia se basa en primero dar una vuelta de 360º sobre si mismo observando lo que tiene alrededor, si no encuentra ninguna bola mediante el sensor de distancia avanza un periodo de tiempo hacia adelante y vuelve a rotar sobre si mismo, el sensor de distancia esta monitorizando en todo momento y en el momento que detecta una bola va hacia ella y la saca del circuito. Para que recorra todo el circuito es importante tener en cuenta que cuando el robot detecta que se va a salir del circuito mediante el sensor de luz lo que hace es retroceder durante dos segundos, girar 230 grados y avanzar un periodo de tiempo hacia delante, de esta forma el robot recorre todo el circuito. Cuando mostremos el código veremos esto en mas

profundidad, a continuación observamos gráficamente un recorrido para que quede mas claro. NOTA: La flecha roja representa marcha atrás del robot.

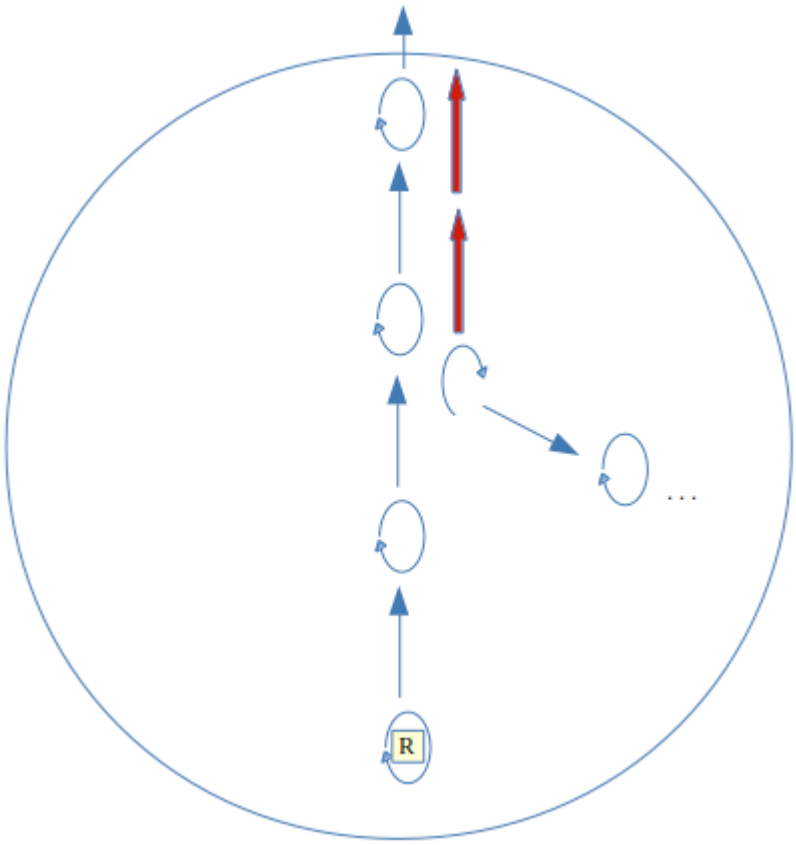

Imagen 6: Traza de la función rastreador.

Competición robot sumo

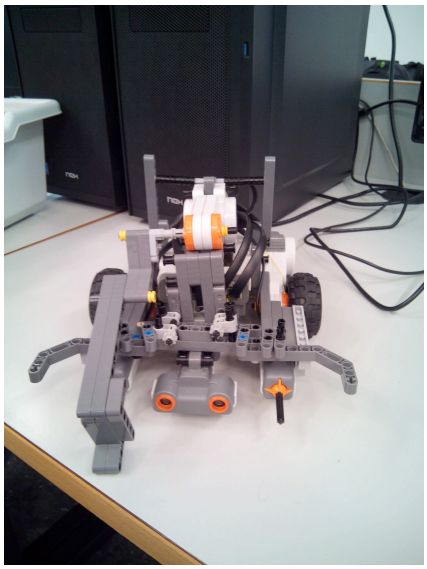

Imagen 7: Configuración del robot para la competición de robot sumo.

La estrategia del robot para la competición de robot sumo se basa en el código implementado para la competición de robot limpiador ya que la función de rastrear la hicimos haciendo pruebas reales y funcionaba muy bien al igual que la de permanecer dentro del circuito, este código añade nuevas funciones que veremos en la parte de descripción de código de la práctica, en resumen busca al robot rival, se aproxima a el hasta tocarlo y en el momento que lo toca ponemos los encoder A y B a la máxima velocidad para que hagan la máxima fuerza de empuje y en el momento que se accionan cualquier de los dos sensores de contacto el encoder C hace de palanca para volcar al robot rival.

3 DESCRIPCIÓN PROGRAMACIÓN DEL ROBOT

En este apartado de la práctica vamos a ver la implementación en ROBOTC de las estrategias comentadas anteriormente, para ello vamos a tener dos secciones una para cada programa .c en las cuales vamos a ver el fragmento de código correspondiente a cada función y una descripción del código.

LIMPIADOR.C

Linea 1-6

```
#pragma config(Sensor, S1,     Luz,                sensorLightActive)
#pragma config(Sensor, S2,     Proximidad,         sensorSONAR)
#pragma config(Sensor, S3,     Contacto,           sensorTouch)
#pragma config(Motor,  motorA,          ,             tmotorNormal, PIDControl, )
#pragma config(Motor,  motorB,          ,             tmotorNormal, PIDControl, )
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
```

Las primeras lineas del código corresponden a las directivas que introduce ROBOTC al insertarle en el menú los sensores y encoders, hemos de comentar que en este programa no vamos a gastar el sensor S3 que corresponde al sensor de contacto pero en el menú lo activamos.

Linea 9-17

```
// Constantes
const tSensors sensorLuz = (tSensors) S1; //sensor Luz
const tSensors sensorProximidad = (tSensors) S2; //sensor Proximidad

// Cabecera de Funciones
void mover();
void sacaBola();
void permanecerDentro();
void rastreador();
```

Las constantes que hemos definido nos permiten trabajar posteriormente para leer los valores que nos proporcionan los sensores, también definimos las cabeceras de las funciones para que no nos salte el compilador si llamamos una función desde otra función sin haberla definido anteriormente.

Linea 22-26

```
// Mover el robot
void mover(int speedA, int speedB){
        motor[motorA]=speedA;
        motor[motorB]=speedB;
}
```

La función mover asigna una velocidad a los encoders A y B que corresponden a las ruedas del robot, dicha velocidad se le pasa a la función por parámetro y nos permite mover el robot ya sea hacia adelante, hacia detrás, girando o parando.

Linea 28-48

```
// Funcion para permanecer dentro del circuito
void permanecerDentro(){
    ClearTimer(T1);
    while(time1[T1]<100){
        mover(0,0);
    }
    ClearTimer(T1);
    while(time1[T1]<2000){
        mover(-85,-85);
    }
    ClearTimer(T1); // Giramos y buscamos bola
    while(SensorValue(sensorLuz)<=50 && time1[T1]<1800){
        mover(40,-40);
        while(SensorValue(sensorProximidad)<50){ sacaBola(); }
    }
    ClearTimer(T1); // Avanzamos y buscamos bola
    while(SensorValue(sensorLuz)<=50 && time1[T1]<1000){
            mover(90,90);
            while(SensorValue(sensorProximidad)<50){ sacaBola(); }
    }
}
```

La función permanecerDentro es de gran importancia ya que si el robot sale fuera del circuito estamos eliminados de la competición. El primer while hace que el robot se detenga durante 100 ms esto hace que las ruedas del robot no pasen de acelerar y desacelerar de repente y hace que el robot cuando detecta el blanco no salga demasiado fuera del circuito para realizar el giro. El segundo hace que el robot retroceda durante 2 s para permanecer dentro del circuito. El tercer while hace que el robot de una vuelta de 230 grados. El último while hace que el robot avance durante un segundo. El robot cuando este girando los 230 grados y avance esta monitorizando el sensor de distancia por si ve una bola y entonces procede a sacarla fuera del circuito.

Linea 51-57

```
// Funcion para tirar una bola fuera del circuito
void sacaBola(){
    while(SensorValue(sensorLuz)<=50){
        mover(90,90);
    }
    if(SensorValue(sensorLuz)>50){ permanecerDentro(); }
}
```

La función sacaBola saca una bola fuera del circuito respetando siempre que el robot este dentro del circuito.

Linea 60-80

```
// Funcion para buscar bolas
void rastreador(){
    while(true){
        while(SensorValue(sensorLuz)<=50){
            while(SensorValue(sensorProximidad)<50){
                sacaBola();
            }
            ClearTimer(T1); // Giramos y buscamos bola
            while(SensorValue(sensorLuz)<=50 && time1[T1]<2700){
                    mover(40,-40);
                    while(SensorValue(sensorProximidad)<50){ sacaBola(); }
            }
            ClearTimer(T1); // Vamos hacia delante y buscando bola
            while(SensorValue(sensorLuz)<=50 && time1[T1]<1300){
                    mover(90,90);
                    while(SensorValue(sensorProximidad)<50){ sacaBola(); }
            }
        }
        permanecerDentro();
    }
}
```

La función rastreador lo que hace es recorrer el circuito para encontrar las bolas a limpiar, siempre que tiene a la vista una bola la saca fuera del circuito, esta continuamente monitorizando el sensor de luz y el sensor de distancia para no salirse y para limpiar, si no esta viendo nada lo que hace es dar una vuelta de 360 grados y avanzar 1.3 s.

Linea 83-89

```
//-------------------------------------------------------
// Tarea Principal
//-------------------------------------------------------
task main(){
    mover(90,90);
    rastreador();
}
```

Esta es la tarea principal que arranca el programa, simplemente le dice que avance al robot y llama a la función rastreador que es la que se estará ejecutando continuamente.

<u>SUMO.C</u>

Linea 1-8

```
#pragma config(Sensor, S1,     Luz,                 sensorLightActive)
#pragma config(Sensor, S2,     Proximidad,          sensorSONAR)
#pragma config(Sensor, S3,     Contacto1,           sensorTouch)
#pragma config(Sensor, S4,     Contacto2,           sensorTouch)
#pragma config(Motor,  motorA,          ,             tmotorNormal, PIDControl, )
#pragma config(Motor,  motorB,          ,             tmotorNormal, PIDControl, )
#pragma config(Motor,  motorC,          ,             tmotorNormal, PIDControl, )
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//
```

Observamos que tenemos mas directivas que en el código LIMPIADOR.c ya que en esta competición utilizamos todos los sensores disponibles y también utilizamos los encoders A, B y C. Estas directivas como ya he comentado anteriormente las introduce ROBOTC al indicarle los sensores y encoders que vamos a utilizar.

Linea 10-21

```
// Constantes
const tSensors sensorLuz = (tSensors) S1; //sensor Luz
const tSensors sensorProximidad = (tSensors) S2; //sensor Proximidad
const tSensors sensorContacto1 = (tSensors)S3; //sensor Contacto
const tSensors sensorContacto2 = (tSensors)S4; //sensor Contacto

// Cabecera de Funciones
void mover();
void permanecerDentro();
void atacar();
void acercarse();
void rastreador();
```

Las constantes que hemos definido nos permiten trabajar posteriormente para leer los valores que nos proporcionan los sensores, también definimos las cabeceras de las funciones para que no nos salte el compilador si llamamos una función desde otra función sin haberla definido anteriormente.

Linea 27-31

```
// Mover el robot
void mover(int speedA, int speedB){
        motor[motorA]=speedA;
        motor[motorB]=speedB;
}
```

La función mover es la misma que en el código LIMPIADOR.c.

Linea 33-53

```
// Funcion para permanecer dentro del circuito
void permanecerDentro(){
    ClearTimer(T1);
    while (time1[T1]<100){
        mover(0,0);
    }
    ClearTimer(T1);
    while (time1[T1]<2000){
        mover(-85,-85);
    }
    ClearTimer(T1); // Giramos y buscamos adversario
    while (SensorValue(sensorLuz) <= 50 && time1[T1]<1800){
        mover(40,-40);
        while (SensorValue(sensorProximidad)<50){ acercarse(); }
    }
    ClearTimer(T1); // Avanzamos y buscamos adversario
    while (SensorValue(sensorLuz) <= 50 && time1[T1]<1000){
        mover(90, 90);
        while (SensorValue(sensorProximidad)<50){ acercarse(); }
    }
}
```

La función permanecerDentro es la misma que en el código LIMPIADOR.c, el motivo de seguir la misma estrategia es que observamos que funcionaba muy bien al realizar pruebas en ambas competiciones, el único cambio que encontramos es que cuando el sensor de distancia encuentra algo llama a la función acercarse en vez de a sacaBola.

Linea 55-70

```
// Funcion para atacar a un adversario cuando lo toca
void atacar(){
    while (SensorValue(sensorLuz) <= 50 && (SensorValue(sensorContacto1) == 1 || SensorValue(sensorContacto2) == 1)){
        mover(100, 100); // Empujamos al rival
        ClearTimer(T1);
        while(time1[T1]<400 && SensorValue(sensorLuz) <= 50){
          motor[motorC] = 100; // Golpeamos arriba
        }
        ClearTimer(T1);
        while(time1[T1]<400 && SensorValue(sensorLuz) <= 50){
          motor[motorC] = -100; // Golpeamos abajo
        }
    }
    motor[motorC] = 0; // Dejamos de golpear
    if (SensorValue(sensorLuz)>50){ permanecerDentro(); } // Permanecemos dentro
}
```

La función atacar lo que hace es comprobar siempre que el robot esta dentro del circuito y al mismo tiempo comprobar que uno o los dos sensores de contacto del robot están pulsados, mientras ocurra esto el robot empuja con los encoders A y B a la máxima velocidad y también acciona la palanca de empuje que esta conectada al encoder C, calculamos midiendo tiempos lo que tarda en hacer una acción de palanca que son 400 ms tanto para levantar la palanca como para volver al estado inicial para poder volver a enganchar al rival, una vez no tenemos contacto con el rival el encoder C se detiene para no gastar bateria.

Linea 72-79

```
// Funcion para aproximarse a un adversario cuando lo ve
void acercarse(){
    while (SensorValue(sensorLuz) <= 50){
        mover(90, 90); // Nos aceracmos hacia el rival
        while(SensorValue(sensorContacto1) == 1 || SensorValue(sensorContacto2) == 1){ atacar(); } // Atacamos al rival
    }
    if (SensorValue(sensorLuz)>50){ permanecerDentro(); } // Permanecemos dentro
}
```

La función acercarse se basa en la función sacaBola del código LIMPIADOR.c, en este caso nos acercamos al rival y cuando lo tocamos con alguno de los sensores de contacto llamamos a la función de atacar.

Linea 81-101

```
// Funcion para buscar rivales
void rastreador(){
    while (true){
        while (SensorValue(sensorLuz) <= 50){
            while (SensorValue(sensorProximidad)<50){
                acercarse();
            }
            ClearTimer(T1); // Giramos y buscamos rival
            while (SensorValue(sensorLuz) <= 50 && time1[T1]<2700){
                mover(40, -40);
                while (SensorValue(sensorProximidad)<50){ acercarse(); }
            }
            ClearTimer(T1); // Vamos hacia delante y buscamos rival
            while (SensorValue(sensorLuz) <= 50 && time1[T1]<1300){
                mover(90, 90);
                while (SensorValue(sensorProximidad)<50){ acercarse(); }
            }
        }
        permanecerDentro();
    }
}
```

La función rastreador funciona es la misma que en el código LIMPIADOR.c, en el caso de detectar algo con el sensor de distancia llamamos a la función de acercarse para que se aproxime al rival hasta que lo toque y pueda atacarlo.

Linea 104-110

```
//-----------------------------------------------------
// Tarea Principal
//-----------------------------------------------------
task main(){
    mover(90,90);
    rastreador();
}
```

La tarea principal activa los motores para que el robot avance y llama a la función rastreador que es la que se ejecuta continuamente llamando a las demás funciones.

4 CONCLUSIONES DE NUESTRO ROBOT EN LAS COMPETICIONES

Competición robot limpiador

Nuestro robot limpio las tres bolas en todos los intentos en menos de un minuto, el robot recorrió todo el circuito llegando incluso a sacar la bola que estaba en el borde del circuito que es la que consideramos como la mas difícil de limpiar. Hemos de tener en cuenta que tener el mismo montaje tanto para la prueba de limpiar como la de sumo dificulta sobretodo la lectura del sensor de distancia ya que nos vimos limitados a que el sensor de distancia estuviese situado en la zona baja del robot para detectar mejor las bolas, perdiendo así un poco de versatilidad a la hora de montarlo. Nuestra solución fue realizar un montaje que fuera híbrido y tuviese un buen comportamiento en ambas competiciones ya que en las dos competiciones el robot tuvo un magnifico resultado.

Competición robot sumo

Nuestro robot en la competición de sumo fue muy robusto, solo nos pudieron ganar porque se dio el caso de que nuestro rival tuvo la suerte de pillarnos por la parte trasera del robot siendo esta la mas débil de nuestra construcción. Respecto a la construcción para el sumo fuimos los únicos que construimos un brazo de empuje que como observamos en la competición fue de gran utilidad ya que permitía levantar los suficiente al rival y de este modo empujarlo mejor e incluso llego a volcar a un robot. Por otra parte el hecho de que nuestro robot tuviera una construcción robusta y un brazo de empuje permitió que en los cara a cara nuestro robot atacase con fuerza y no se limitara a aguantar el empuje del rival o a empujarle hacia fuera.