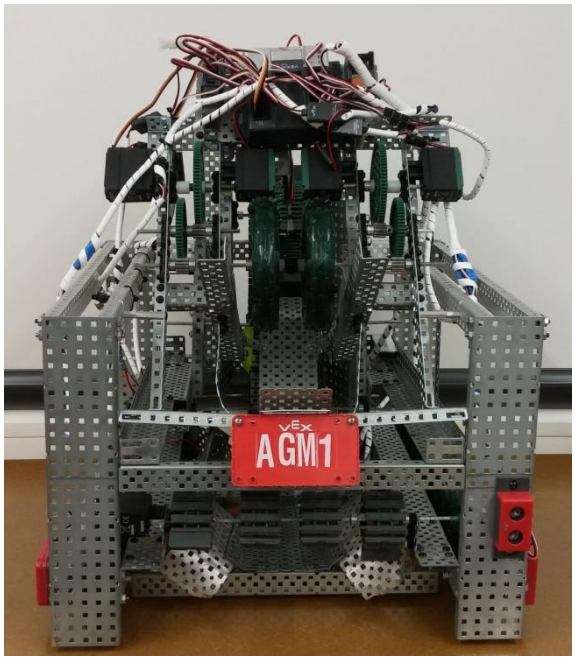
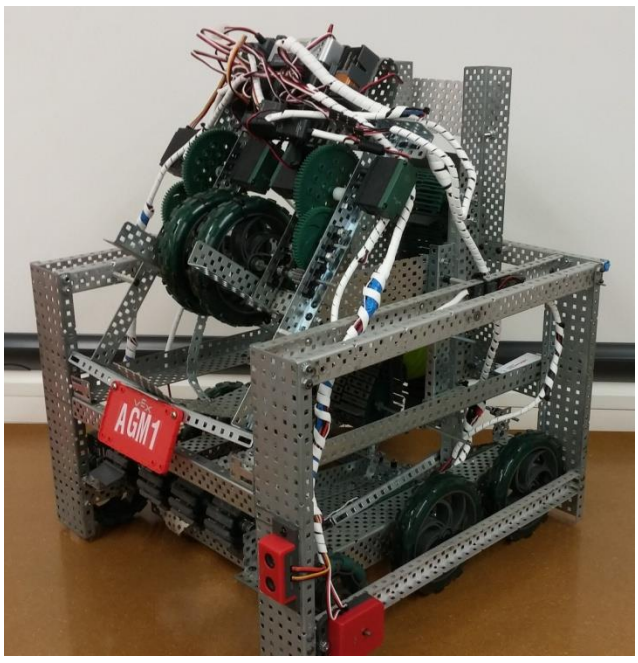
13/03/2016

VEX ROBOTICS – U COMPETITION SPAIN

Joaquín González Espínola , Arnau Liesa Montfort y Xavier Blanquer Natividad, Jose Luís Peña



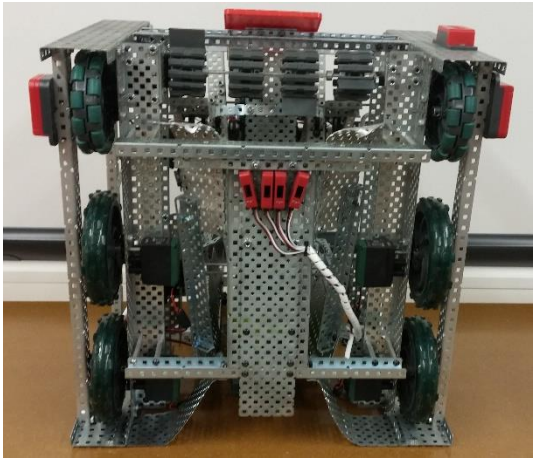


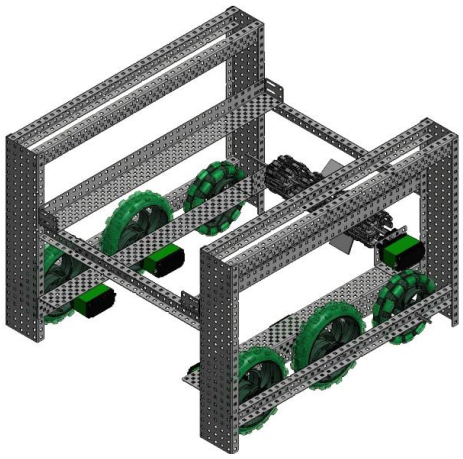
Introducción

La base

El robot que presentamos en la competición nacional ha sido diseñado desde cero. Hemos usado la herramienta Autodesk Inventor 2014.

En primer lugar se diseñó la base y se probaron los diferentes tipos de ruedas, de las cuales optamos por cuatro ruedas motrices en la parte trasera y dos ruedas omnidireccionales en la parte delantera. En cuanto a las ruedas motrices, las dos más atrasadas son de 5" y las dos más adelantadas son de 4".

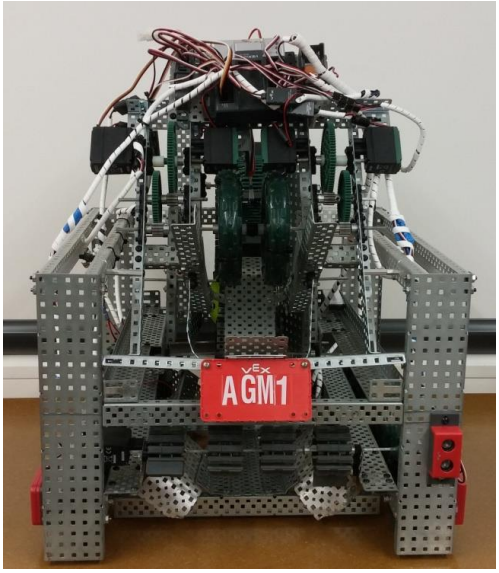


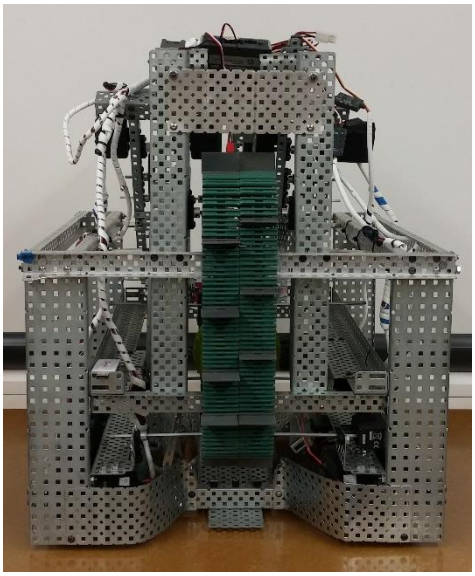


El chasis escogido es de tipo cuadrado mayoritariamente en aluminio haciendo que sea más ligero.

Recogedor de pelotas

Se ha realizado un recogedor de pelotas hibrido, que recoge pelotas por delante y a la vez puede recogerlas por detrás. Permite recoger las pelotas que están debajo a mucha velocidad y una vez dentro impide que estas puedas salir.



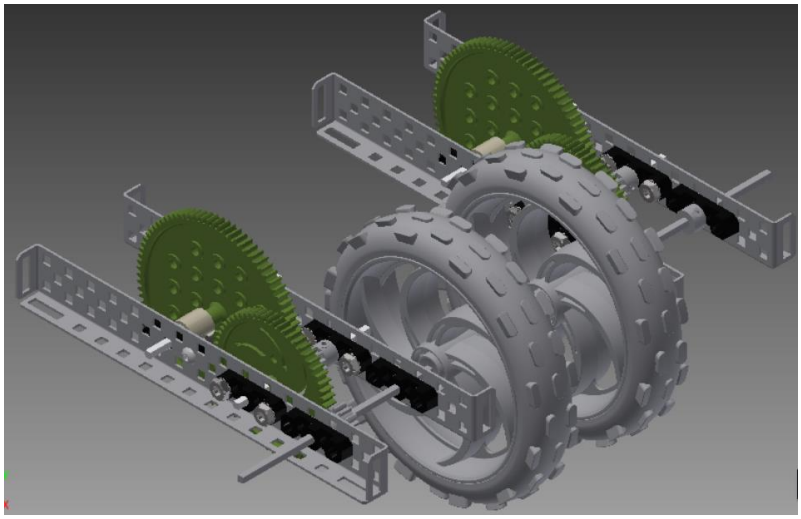


Lanzador

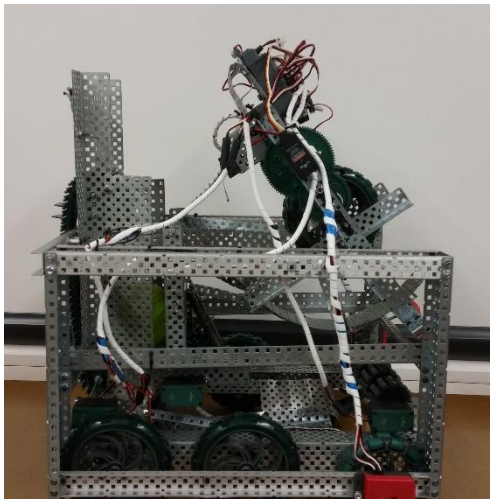
Para nuestro lanzador se han utilizado diferentes configuraciones para poder lanzar las pelotas desde base.

Observamos mejores resultados al usar una secuencia de engranajes 7:1*5:1 proporcionando, en teoría, hasta 3500 revoluciones por minuto. Esta impulsado por 4 motores de 100 rpm y utiliza dos ruedas de 5" que hace que las pelotas lleguen con facilidad y con un ángulo adecuado para poder encestar.

Se realizado el diseño en Autodesk Inventor 2014 y se ajustó a la realidad para que hubiera el mínimo rozamiento:



Para controlar la velocidad del lanzador se ha colocado un encoder integrado en uno de los motores. Durante las pruebas, también hemos colocado encoders en las ruedas de 5" del lanzador (las velocidades necesarias para encestar en el campo son de 22 a 29 vueltas por segundo).



Sensores y programación

Los sensores que hemos usado son: 3 encoders, 4 seguidores de línea, 1 sónar y un final de carrera.

Empezaremos a hablar de la estrategia y el porqué de cada sensor. En la parte autónoma optamos a 45 segundos los cuales podemos usar sensores, en nuestro caso, la estrategia consiste en lanzar las cuatro pelotas de precarga desde base, localizar la línea e ir directos a coger el montón más cercano, siendo este el que encarando el robot tiraremos. Continuamos siguiendo la línea hasta llegar al montón más cercano a la red el cual una vez cogido retrocederemos para tirar las pelotas de nuevo. A partir de aquí seguiremos la línea hacia atrás hasta encontrar el centro del campo que és donde se cruzan las líneas, y es allí donde usaremos los encoders puestos en las ruedas, para determinar el giro correcto y movernos hacia atrás para coger los montones que hay entre las dos bases, una vez cogidos volveremos a encarar el robot a canasta y tiraremos por última vez.

Tenemos el tiempo justo para hacer esta estratégia y lo logramos gracias la precisión que nos da el conjunto de todos los sensores.

Primero hablaremos de los encoders:

Los usamos para determinar las revoluciones ideales del lanzador desde cada posición y conseguir los giros perfectos.

El código usado es el siguiente:

```
#pragma config(Sensor, dgtl3,  rightEncoder,   sensorQuadEncoder)

#pragma config(Sensor, dgtl5,  leftEncoder,    sensorQuadEncoder)

#pragma config(Sensor, I2C_1,  L4MEncoder,     sensorQuadEncoderOnI2CPort,    ,
AutoAssign)

// funcion que hace que el lanzador vaya a una velocidad

int ajustlanzador()

{

        // Ajustamos la velocidad para iniciar lanzamientos

        int gradosseg=0;

        int error=0;



 SensorValue[L4MEncoder] = 0;


 wait1Msec(500);
```

```
        gradosseg=SensorValue[L4MEncoder];



        if(320<gradosseg) error=-2;
        else if(300<gradosseg<321) error=0;
        else if(gradosseg<301) error=2;

        return error;
}
// funcion para realizar un giro a la izquierda de X grados
void giro_derecha(int grados)
{
        SensorValue[leftEncoder] = 0;
        while(SensorValue[leftEncoder] < grados)
        {
                motor[backRightMotor]  = -50;
                motor[backLeftMotor]  = 50;
        }

        motor[backRightMotor]  = 0;
        motor[backLeftMotor]  = 0;
}
```

Las dos funciones anteriores son dos que usamos con los encoders. La primera nos gradua la velocidad del lanzador, su funcionamiento consiste en decirle las revoluciones que queremos que gire el lanzador y así los motores suben su velocidad o la bajan. Para ello hemos calculado la revoluciones en cada punto de lanzamiento.

Por otra parte tenemos la función que usamos para girar a la izquierda, siguiendo un procedimiento similar, en este caso el robot girara hasta completar los grados estipulados, también calculados anteriormente.

Ahora pasamos al sensor sónar:

El sónar nos ha facilitado la tarea de saber exactamente donde está el robot. Nos basamos a decir esto ya que en nuestra programación nuestro robot siempre está apuntando a la red cual cosa podemos detectar el tubo de PVC de la zona baja de puntuación.

Para *su programación simplemente hemos usado el siguiente código:*

```
#pragma config(Sensor, dgtl11,  sonarSensor,    sensorSONAR_cm)

while(SensorValue(sonarSensor) > 25 || SensorValue(sonarSensor) == -1)

{

}
```

Explicamos el siguiente código. El sensor sónar tenemos puesto que nos detecte en centímetros, por lo tanto en esta sentencia WHILE(cuando) le estamos declarando dos cosas, la primera es que nos adquiera el valor del sónar y nos haga el contenido del while siempre que el valor del sónar sea mayor a 25cm, por otra parte, cuando el sónar no recibe la onda enviada, su resultado es -1, es por eso que le hemos declarado que si su valor es -1 no continúe dentro del while ya que normalmente tiene micro cortes.

Seguimos con el seguidor de línea:

Para aprovechar el máximo el tiempo autónomo, buscamos que el robot pudiera tirar pelotas mientras sigue la línea, es por eso que queremos que aparte de seguir la línea lo haga de la forma más recta posible. Calculamos la separación entre seguidores y el ancho de la línea y vimos que puede detectar dos sensores a la vez de forma muy justa, así cuando nuestros seguidores de línea centrales detecten línea la precisión es absoluta.

Ahora entraremos a su programación, para ellos nos hemos basado en el concepto de lógica difusa. Consiste en enseñar al robot todas las opciones posibles. Después de haber estudiado su comportamiento vimos 9 posibilidades, la cual cosa la hemos declarado en una misma función.

El código es el siguiente:

```
#pragma config(Sensor, in1,    lineFollowerRIGHT, sensorLineFollower)

#pragma config(Sensor, in2,    lineFollowerCENTERR, sensorLineFollower)

#pragma config(Sensor, in3,    lineFollowerCENTERL, sensorLineFollower)

#pragma config(Sensor, in4,    lineFollowerLEFT, sensorLineFollower)

void siguelinea()

{


    int threshold = 600;     /* found by taking a reading on both DARK and LIGHT    */
```

```
	/* surfaces, adding them together, then dividing by 2. */


	while(SensorValue(sonarSensor) > 25 || SensorValue(sonarSensor) == -1)
	{
		int right = (SensorValue(lineFollowerRIGHT));
		int centerr = (SensorValue(lineFollowerCENTERR));
		int centerl = (SensorValue(lineFollowerCENTERL));
		int left = (SensorValue(lineFollowerLEFT));


		// RIGHT sensor sees dark:
		if(  right>threshold && centerr>threshold && centerl>threshold &&
left<threshold )
		{


			// counter-steer right:
			motor[backLeftMotor]  = 0;
			motor[backRightMotor] = 60;


		}
		// CENTER sensor sees dark:
		if(  right>threshold && centerr>threshold && centerl<threshold &&
left<threshold )
		{
			// go straight
			motor[backLeftMotor]  = 0;
			motor[backRightMotor] = 60;
		}
		if(  right>threshold && centerr>threshold && centerl<threshold &&
left>threshold )
		{
```

```
                    // go straight
                    motor[backLeftMotor]  = 40;
                    motor[backRightMotor] = 60;
                }
                if(  right>threshold && centerr<threshold && centerl<threshold &&
left>threshold )
                {
                    // go straight
                    motor[backLeftMotor]  = 60;
                    motor[backRightMotor] = 60;
                }
                if(  right>threshold && centerr<threshold && centerl>threshold &&
left>threshold )
                {
                    // counter-steer left:
                    motor[backLeftMotor]  = 60;
                    motor[backRightMotor] = 40;
                }
                // LEFT sensor sees dark:
                if(  right<threshold && centerr<threshold && centerl>threshold &&
left>threshold )
                {
                    // counter-steer left:
                    motor[backLeftMotor]  = 60;
                    motor[backRightMotor] = 0;
                }
                if(  right<threshold && centerr>threshold && centerl>threshold &&
left>threshold )
                {
```

```
                motor[backLeftMotor]  = 60;
                motor[backRightMotor] = 0;
            }
            if(  right<threshold && centerr<threshold && centerl<threshold &&
left<threshold )
            {
                wait1Msec(500);
                motor[backLeftMotor]  =0;
                motor[backRightMotor] = 0;
                lanzador(70);
                wait1Msec(500);
            }

            if(  right>threshold && centerr>threshold && centerl>threshold &&
left>threshold )
            {
                motor[backLeftMotor]  = -30;
              motor[backRightMotor] = 50;
            }
        }
                motor[backLeftMotor]  =0;
                motor[backRightMotor] = 0;
   lanzador(75);

}
```

Como se puede observar, tenemos las 9 posibilidades diferenciadas en los distintos IF.

Para empezar a explicar el código, hemos puesto el valor de los sensores en variables para así trabajar de una forma más cómoda.

Después de ello, hemos tenido que crear un threshold (umbral) para interpretar el funcionamiento de seguidor de línea. El valor de sensor al detectar blanco es de 180, y cuando detecta negro es de 1700,

por lo tanto hemos cogido un número aproximado a la mitad para hacer la distinción de blanco o negro.

El funcionamiento de los diferentes IF consiste en decirle si el sensor está por encima del umbral (detecta negro) o por debajo (detecta blanco).

Por último explicamos el final de carrera:

Podríamos decir que es uno de los sensores que más nos ha gustado poner. Su función es la de ganar milésimas de segundo en cada lanzamiento de bolas.

Hemos visto que todos los lanzadores que hemos probado tienen un tiempo para poder recuperar la velocidad deseada, es por eso que con este sensor, lo que hacemos es detectar la pelota justo antes de lanzarla así dándole un impulso extra que nos evita el tiempo de perdida de potencia.

Conclusiones

Primero de todo queremos agradecer la oportunidad que se nos da a los estudiantes con competiciones a nivel mundial, nos ha ayudado a completar gran parte de los estudios pudiendo ver, tocar y montar un robot.

Llegar hasta el día de la competición ha sido un trabajo duro. Esta es la segunda participación de algunos miembros del equipo, desde el principio del proyecto han compartido todo su conocimiento adquirido para que todos estemos al mismo nivel tanto de montaje como de programación. A partir de aquí, nos documentamos sobre los objetivos y normas de la competición e hicimos una lluvia de ideas para poner en común nuestros conocimientos y poder crear un robot competitivo.

Cuando acabamos de decidir nuestro primer diseño competitivo, separando las partes principales, el movimiento, la recolección de pelotas y su lanzamiento, empezó el verdadero dolor de cabeza.

Todo y la primera puesta en común, al principio se notaba una gran diferencia entre los integrantes del grupo, nos costó bastante adecuarnos a un nivel cómodo para los tres y es así que nos repartimos las tareas y después no pudimos juntarlas por utilizar diferentes estilos de montajes.

Antes de continuar nos sentamos a hablar y volver a diseñar un robot más a acorde de todos.

Ahora sí que íbamos todos a una y mejoramos el robot exponencialmente hasta llegar a la competición de Barcelona, la cual ganamos por los pelos.

Nos sorprendió mucho el resultado, porque por un descuido nuestro perdimos la final, pero gracias al trabajo bien hecho conseguimos que el robot tuviera la mejor mecánica y programación así logrando el reconocimiento.

Eso nos marcó mucho, no nos gustó la resolución final y es por eso que el lunes siguiente empezamos a hacer un robot, no capaz de ganar, sino capaz de plantar cara en la final mundial, nuestro objetivo es llegar a la final mundial y allí clasificarnos para las eliminatorias del torneo.

El robot es completamente distinto, hemos cambiado los tres pilares más importantes del robot y además hemos empezado a crear un sistema para la elevación del otro robot.

Resumiendo nuestro robot en números, en la parte autónoma conseguimos encestar un total de 12 pelotas de las cuales 3 naranjas y 9 verdes consiguiendo 75 puntos como mejor marca. Después en la parte manual creemos que meteremos dos pelotas naranjas más y un total de 24 pelotas verdes, así consiguiendo como mejor marca 140 puntos y un total de 215 puntos. Recordamos que no optamos a los 50 puntos de elevación de robot.

Estamos muy contentos del resultado final, ahora solo falta competir y conseguir la victoria.