Universidad de Alicante

Investigación y Propuestas Innovadoras de Redes UA para la Mejora Docente

ISBN: 978-84-617-3914-1

Coordinadores

José Daniel Álvarez Teruel
María Teresa Tortosa Ybáñez
Neus Pellín Buades

Universidad de Alicante
Vicerrectorado de Estudios, Formación y Calidad
Instituto de Ciencias de la Educación (ICE)

ISBN: 978-84-617-3914-1

Revisión y maquetación: Neus Pellín Buades

Desarrollo de un robot modular para la enseñanza en el Máster en Automática y Robótica

C. A. Jara Bravo, J. Pomares Baeza, S. T. Puente Méndez, F. Torres Medina, D. Mira Martínez, C. M. Mateo Agulló, J. Pérez Alepuz

Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal

Universidad de Alicante

RESUMEN

Este trabajo presenta el diseño, construcción y programación de un robot modular para el desarrollo tanto de competencias genéricas como específicas, en las enseñanzas de electrónica, control y programación del Master de Automática y Robótica de la Escuela Politécnica Superior de la Universidad de Alicante. En este trabajo se exponen los diferentes módulos propuestos, así como los objetivos de aprendizaje para cada uno de ellos. Uno de los factores más importantes a destacar en el presente estudio es el posible desarrollo de la creatividad y el aprendizaje autónomo. Para ello, se desarrollará especialmente un módulo de comunicación por *bluetooth* que servirá para monitorizar, cambiar y adaptar on-line diversos parámetros de control y potencia del robot. Además, dicha herramienta se ha introducido como parte de la metodología en las asignaturas del Máster de Electromecánica y Sistemas de Control Automático. En esta memoria se mostrarán los distintos resultados obtenidos durante y en la finalización de este trabajo.

Palabras clave: aprendizaje autónomo, experiencias prácticas, programación, control, robot rastreador,

1. INTRODUCCIÓN

1.1 Problema/cuestión.

Con la llegada del Espacio Europeo de Educación Superior (EEES), el docente universitario debe impartir las asignaturas teniendo en cuenta dos características fundamentales (Huber, 2008): la adquisición de competencias y la enseñanza centrada en el estudiante. El estudiante deja de ser un mero espectador como en el modelo tradicional de clases magistrales y se transforma en el actor principal de su aprendizaje. Dentro de este aprendizaje autónomo, las experiencias prácticas y el desarrollo de habilidades centradas en la experiencia adquirida desde la empírica juegan un papel importante. Para ello, se ha realizado este proyecto, que se centra en un robot modular para el desarrollo tanto de competencias genéricas como específicas, en las enseñanzas de electrónica, control y programación del Máster de Automática y Robótica de la Escuela Politécnica Superior de la Universidad de Alicante.

1.2 Propósito.

Este trabajo presenta el diseño, construcción y programación de un robot modular para el desarrollo tanto de competencias genéricas como específicas, en las enseñanzas de electrónica, control y programación del Máster de Automática y Robótica de la Escuela Politécnica Superior de la Universidad de Alicante. Este robot servirá fundamentalmente para el desarrollo de la creatividad y el aprendizaje autónomo del alumno. Además, dicha herramienta se ha introducido como parte de la metodología en las asignaturas del Máster de Electromecánica y Sistemas de Control Automático, de las que mostraremos sus fichas modificadas.

El robot modular se ha diseñado y desarrollado pasando cada una de las siguientes fases:

- Programación de un microcontrolador Arduino (Arduino, 2014) donde se encuentran conectados una serie de sensores lumínicos puestos en la base del robot, con los que podremos dirigir el robot por una línea negra.
- Monitorización de velocidad, fuerzas y errores del robot. Esta información será enviada por el adaptador *bluetooth* incorporado al robot y recibidos por el PC. Estos datos se podrán analizar y mostrar por pantalla en forma de graficas generadas por el programa creado.

- Ajustes de los parámetros anteriores para mejorar el guiado del robot. Envió de datos al robot por el *bluetooth* del PC utilizando el programa creado. Toda la información es codificada, creando así un nuevo protocolo de comunicación entre el robot y el PC.

Este proyecto, con las características indicadas anteriormente, debería implicar una mejora en el proceso de aprendizaje del alumno al ser éste activo, auto-dirigido, constructivo y situado (Shuell, 1986). Además, el papel del profesor en este caso cambia del tradicional transmisor de conocimientos al nuevo tutor del estudiante, que guía su aprendizaje y le ayuda a resolver sus dudas. En esta memoria se exponen los diferentes módulos propuestos en el robot, así como los objetivos de aprendizaje para cada uno de ellos.

2. METODOLOGÍA

2.1. Descripción del contexto y de los participantes

Para el desarrollo de la red docente se han realizado reuniones periódicas conjuntas con el objetivo de coordinar el desarrollo y fijar los principales objetivos a cubrir. Todo el profesorado participante en este proyecto imparte docencia en el Máster Universitario en Automática y Robótica de la Universidad de Alicante. Para hacer efectiva la colaboración entre todos los miembros del proyecto se han establecido 2 grupos de trabajo. Un primer grupo se encargará de desarrollar el robot modular (sensorización, montaje, control, potencia, módulo de monitorización, etc), mientras que el segundo grupo desarrollará las estrategias de auto-aprendizaje, auto-evaluación y aprendizaje tutorizado por el profesor haciendo uso del robot modular.

2.2. Materiales e instrumentos

En este apartado vamos a describir los materiales e instrumentos del proyecto, empezando por la base del proyecto, como las características del *shield* donde está montado el microcontrolador Arduino. Además, se va a explicar el funcionamiento de los sensores que hacen posible la lectura de las líneas del suelo, qué tipo de datos recogen y cómo podemos trabajar con ellos posteriormente. También se mostrarán las características de los motores traseros, que hacen posible el movimiento del robot. Estos motores serán de corriente continua y vendrán provistos de un *encoder*, de donde se

podrá obtener la velocidad. La comunicación la haremos con el modulo *bluetooth*, HC-05, fabricado especialmente para placas Arduino. Utiliza una comunicación serie, con la que crearemos nuestro propio protocolo, sencillo y fácil de utilizar. Otro sensor que vamos a anclar al *shield*, es un giroscopio tipo l3g4200d. Con el obtendremos datos de fuerzas generadas en la toma de curvas.

2.2.1. *Shield* robot base.

Para realizar mucho mejor nuestro trabajo, recurrimos a una placa compacta, en la que se conecta todo nuestro hardware. En la figura 1 se observar con detenimiento los componentes de los que forma parte.

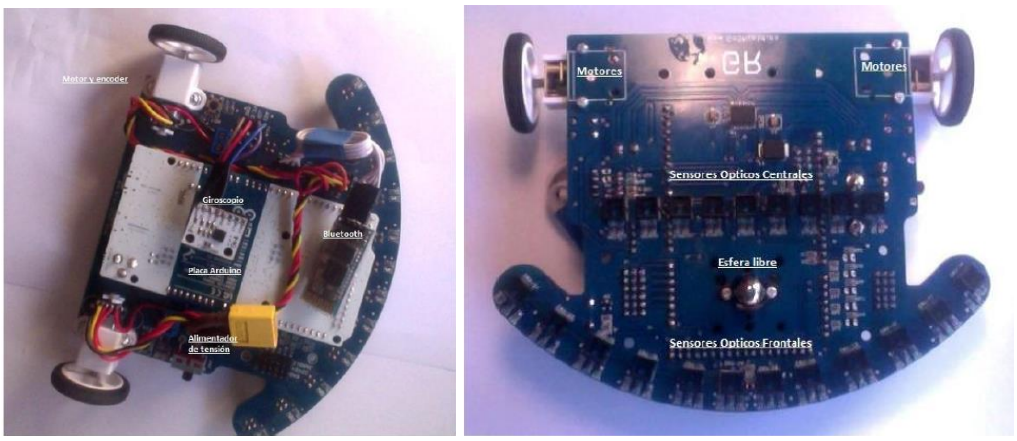

Figura 1. Robot con todos sus componentes

Placa Arduino Due

El Arduino Due es un microcontrolador basado en el Atmel SAM3X8E ARM Cortex-M3 CPU. Es el primer Arduino con 32 bit en el procesador ARM y funcionando a una frecuencia de 84MHz. Posee 54 salidas y entradas digitales, de las cuales 12 se pueden utilizar como salidas PWM, 12 como entradas analógicas, 4 son para el puerto serie. Tiene además dos conversores digital a analógico, 2 TWI, un botón de reset, etc. Es muy importante saber que estas placas trabajan a 3.3V a diferencia de las demás que funcionan con una tensión de 5V. La placa contiene todo lo necesario para la utilización del microcontrolador, simplemente tenemos que conectarlo a un ordenador con USB. Es compatible con todos los *shield* de Arduino que puedan trabajar a 3.3V.

Figura 2. Placa Arduino Due

Comunicación

El Arduino Due tiene una serie de facilidades para la comunicación con ordenadores, otros Arduinos o microcontroladores de otros fabricantes, incluso con diferentes dispositivos como tablets o cámaras. El SAM3X ofrece un hardware UART y tres USARTs para la comunicación TTL serie. El puerto de programación conectado directamente al ATmega16U2, proporciona un puerto virtual para conectarlo con el PC. De serie, los pines RX0 y TX0 proporcionan también una comunicación serie a USB para programar la placa. Además, el propio software de Arduino incluye un monitor de serie que permite el paso de datos básicos entre el PC y la placa y gracias a los LED RX y TX de la placa, podemos observar si se están transmitiendo datos o no, ya que el parpadeo nos lo indica (no en la comunicación serie utilizando los pines 0 y 1). Es aquí donde conectaremos el nuestro modulo *bluetooth* para poder controlarlo a distancia desde nuestro ordenador. El puerto USB nativo, conectado al SAM3X, permite la comunicación serie a través del USB, proporcionando una la conexión entre el Monitor serie y otras aplicaciones del PC. También permite el acople de periféricos, como un ratón o un teclado.

Programación

El Arduino Due puede ser programado con utilizando el software propio de la marca. Para enviar el programa generado es recomendable utilizar el puerto USB de programación, debido a la forma en que el microcontrolador maneja el borrado de la memoria Flash, necesario para la subida del código.

- Puerto de Programación: para usar este puerto necesitamos decírselo al IDE de Arduino. “Tools/Board/Arduino Due (Programming Port)”. El puerto de programación utiliza el 16U2 como un chip de USB serie conectado a la primera UART del SAM3X

(RX0 y TX0). El 16U2 tiene dos clavijas conectados al reset y borra los pins del SAM3X. La apertura y el cierre del puerto de programación, conectado a 1200bps, desencadena un procedimiento de "borrado duro" del chip SAM3X, antes de comunicarse con la UART, siendo más fiable que el "borrado suave" que se produce en el puerto nativo. Por lo tanto podemos decir que es el puerto recomendado para la programación, como ya hemos citado antes.

- Puerto Nativo: Para utilizar este puerto, hay que seleccionar "Arduino Due (puerto USB nativo)" en el IDE. El puerto USB nativo está conectado directamente a la SAM3X. La apertura y el cierre de este puerto, desencadena un procedimiento de borrado suave, donde la memoria flash se borra y la placa se reanudará con el gestor de arranque. Si el micro cayó por alguna razón, es probable que el procedimiento de borrado suave no funcione. Abrir y cerrar el puerto nativo a una velocidad de transmisión diferente no restablecerá el SAM3X.

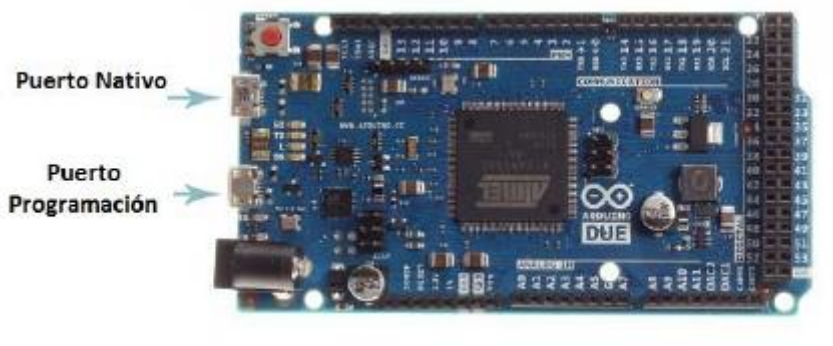

Figura 3. Puertas de entrada al Arduino Due

2.2.2. Sensores ópticos CNY70.

El sensor CNY70 es uno de los sensores más empleados en micro-robótica, dada su economía y sus variadas aplicaciones prácticas. Habitualmente se usa siempre que se desea que el robot móvil siga un camino marcado por una raya en el suelo, este enviará al microcontrolador toda la información necesaria para que sepa si ambos están sobre la raya, uno de ellos o por el contrario, los dos están sobre el fondo. También pueden usarse en la detección de obstáculos, siempre que las superficies de los mismos sean reflectantes.

Figura 4. Sensor óptico CNY70

2.2.3. Motores de corriente continua y encoder.

Para este proyecto vamos a utilizar dos motores de corriente continua de altas prestaciones, con una reductora de 30:1 para conseguir más potencia en el robot y reduciendo su velocidad siendo así más manejable. Además posee un eje extendido que anclaremos a las ruedas. Situados en la parte de atrás del rastreador, son los encargados de su movimiento. Únicamente necesitan un punto de apoyo en el centro del bajo del robot. Una esfera que se mueve libremente es nuestra solución. Estos motores tienen la suficiente potencia para moverlo a una velocidad bastante alta, no obstante normalmente no los ponemos a su velocidad máxima. En el programa de control tenemos que evitar que se le envíe una velocidad superior a la permitida, en nuestro caso el máximo es de 122.

Figura 5. Motor DC 30:1 con encoder

Especificaciones técnicas

- Reductora: 30:1
- Tensión funcionamiento: 3 a 9 VDC. (Se recomienda 6V, mayores voltajes pueden reducir la vida del motor).
- RPM: 1000 RPM
- Corriente en bloqueo: 1.6A.
- Corriente en vacio: 100mA.
- Dimensiones: 26 x 10 x 12mm
- Diámetro del eje: 3mm de diámetro, con ranura de bloqueo. Eje trasero extendido.
- Fuerza (Par máximo): 1,1 kg-cm.

Encoder

Estos motores poseen *encoders* ópticos, que se encargan de obtener la velocidad a la que gira el extremo del motor. Son utilizados en el programa para monitorizar las

velocidades tanto de la rueda izquierda como la de la derecha, obteniendo la velocidad lineal del robot en cada instante de tiempo.

2.2.4. Módulo *bluetooth* HC05.

Se trata un pequeño modulo *bluetooth* de apenas 30 mm de largo, unido a su poco consumo de energía, lo hace perfecto para aplicaciones en la micro-robótica. Utiliza la comunicación serie para establecer un canal por donde poder enviar y recibir información. Esta información se envía con una velocidad de 115200 baudios. Crearemos un protocolo especial con una serie de parámetros que utilizaremos para que los dos programas, el del Arduino y el interfaz del PC, puedan comunicarse sin ningún tipo de problema. Los pines del bluetooth los conectaremos a las entradas de "serie" que tiene el Arduino. Con esto funcionaría igual que el serial monitor del IDE de Arduino.

Figura 6. Modulo Bluetooth hc-05

Conexión del módulo *bluetooth* al *shield*

El modulo se conectaría a los pines TX, RX, voltaje y tierra. Estos primeros son los encargados de enviar y recibir toda la información. A continuación mostraremos una captura con las conexiones.

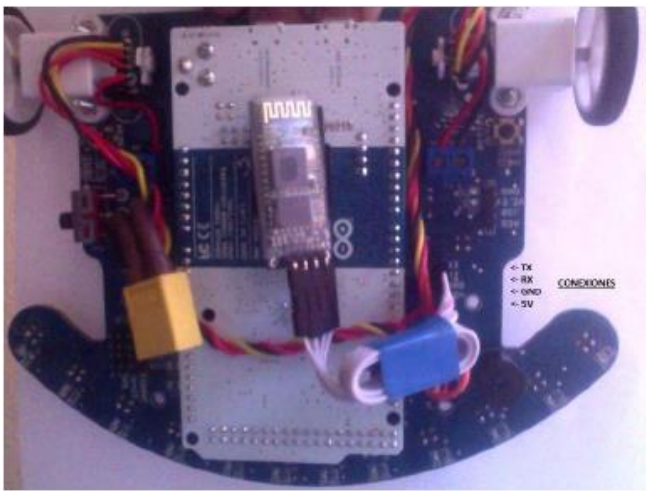

Figura 7. Conexión módulo *bluetooth* al robot modular.

Protocolo usado

Arduino→PC

	Identificador	Valor	Fin de trama
Vel Motor Der	"D"	Entero	"@"
Vel Motor Izq	"I"	Entero	"@"
Error	"E"	Entero	"@"
Giroscopio	"G"	Entero	"@"

PC→Arduino

	Identificador	Valor	Fin de trama
Vel Inicial	"V"	Entero	"@"
Proporcional	"P"	Entero	"@"
Derivativa	"D"	Entero	"@"
Inicio	"I"		"@"
Parada	"S"		"@"
Calibrado	"C"		"@"
Activar Env Datos	"E"		"@"
Parar Env datos	"N"		"@"

2.2.4. Giroscopio L3G4200D.

Los giroscopios, o girómetros, son dispositivos que miden o mantienen el movimiento de rotación, la velocidad angular. Las unidades de velocidad angular se miden en grados por segundo (° / s) o revoluciones por segundo (RPS). Los giroscopios se utilizan a menudo en los objetos que no están girando muy rápido sobre sí mismos. En su lugar giran unos pocos grados en cada eje. Mediante la detección de estos pequeños cambios los giroscopios ayudan a estabilizar el vuelo de la aeronave. Además, hay que tener en cuenta que la aceleración o la velocidad lineal de la aeronave no afecta a la medición del giróscopo. Los giroscopios sólo miden la velocidad angular.

Figura 8. Giroscopio L3G4200D

Ejes

Este giroscopio de 3 ejes: puede medir la rotación en torno a tres ejes: X, Y , y Z . Algunos giroscopios vienen en variedades de eje simple y doble, pero el giroscopio de tres ejes en un solo chip son cada vez más pequeño, menos costoso y más popular. Para nuestro trabajo, al trabajar en un plano 2D solo haremos uso de uno de los dos ejes. Colocando el chip de forma horizontal, el único eje que proporciona información es este mismo.

Conexión

Giroscopio con una interfaz digital por lo general usan el SPI o protocolos de comunicación I2C, en nuestro caso utilizaremos este último. El uso de estas interfaces permite una conexión fácil a un microcontrolador. Una de las limitaciones de una interfaz digital es la velocidad de muestreo máxima, trabajando a una frecuencia de muestreo máximo de 400 Hz. Para incluirlo dentro de nuestro programa, nos haremos uso de su librería (L3G), que implementa las funciones necesarias para obtener los datos de velocidad angular de cada eje. Se modificó la librería, puesto que el *shield* con el que trabajamos, tenía únicamente la comunicación de I2C secundaria activada (Wire1). Matizamos que la placa DUE posee dos vías de comunicación I2C. En nuestro caso, lo conectaremos a los pines SDA, SCL, voltaje y tierra. Internamente el *shield* está diseñado para hacer de puente con el Arduino.

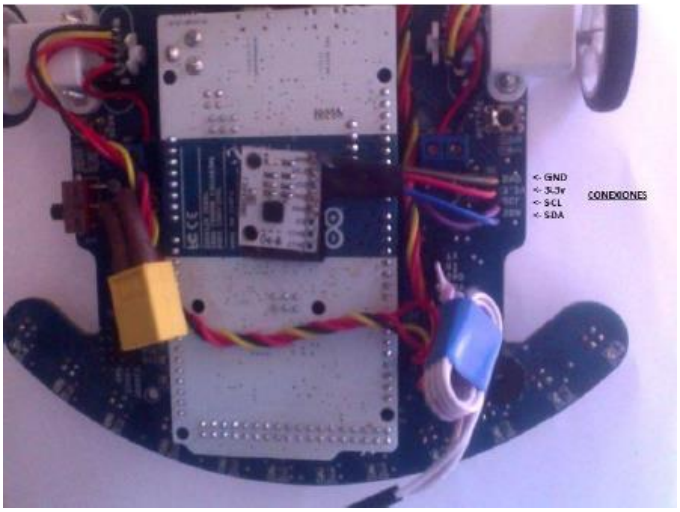

Figura 9. Conexión del giroscopio al robot modular.

2.3. Procedimientos: programación del robot modular

En esta parte, se va a comentar toda la parte de programación del Arduino para acondicionar el robot modular para las prácticas. Todo el programa irá metido dentro del micro Arduino y constará de una librería propia implementada, que se encargará de administrar todo el proceso. Esta librería estará programada utilizando C++ y constará de una serie de funciones propias para controlar el robot. Para la interfaz, nos ayudaremos del entorno *Microsoft Visual Studio*. Con el diseñaremos todas las ventanas interactivas que nos ayudaran en la monitorización y el ajuste del robot. Utilizará la comunicación *bluetooth* serie del ordenador para enviar y recibir dicha información.

2.3.1. Clase de control del robot.

Esta clase está creada desde cero, apoyándonos en ejemplos de una librería que se implementó para este robot. Esta librería tenía varias funciones muy básicas de control, pero que apenas nos podrían servir de mucha ayuda, no obstante comenzamos con esta base. Elegimos administrar todo el código en una clase, porque es más fácil administrarla a posteriori. El código se encontrará más limpio y para futuras modificaciones resultará beneficioso. Queríamos que las funciones de la clase tuvieran el menor código posible, de esta manera su depuración seria cómoda. Además el programa principal del Arduino sería muy simple, únicamente tendría las interrupciones de los *encoder*, imposibles de implementar en una clase, y las declaraciones de objetos fuera de nuestra clase. Esto último hace referencia a la utilización de la librería que controla el giroscopio. Todo este código estaría guardado en el Arduino y se ejecutaría nada más encenderlo. A continuación mostraremos las diferentes funciones implementadas para el funcionamiento del robot.

- *CRobot (void)*

Constructor. Llama a la función inicio.

- *void inicio ()*

Inicializa todos los vectores y variables del programa.

- *void MotorVelIzq (unsigned int vel)*

Coloca una velocidad al motor izquierdo.

- *void MotorVelDer (unsigned int vel)*

Coloca una velocidad al motor derecho.

- *void MotorVel (int VelI, int VelD)*

Actualiza las velocidades de los dos motores.

 void MotorPonerVel (unsigned int vel)

Actualiza la velocidad inicial del robot.

 int ObtenerMotorVelIzq ()

Obtiene la velocidad actual del motor izquierdo.

 int ObtenerMotorVelDer ()

Obtiene la velocidad actual del motor derecho.

 void IncEncoderIzq ()

Incrementa encoder del motor izquierdo.

 void IncEncoderDer ()

Incrementa encoder del motor derecho.

 bool TiempoEncoder ()

Devuelve true cuando han pasado 100 ms.

 void SensorCentralLeer ()

Funcion que obtiene los valores de los sensores centrales.

 void SensorFrontalLeer ()

Función que obtiene los valores de los sensores frontales.

 void SensorFrontalCalcular ()

Calcula la distancia de la línea al centro del robot.

 void LedFrontalApagar (const int numLed)

Enciende el LED que queramos.

 void LedFrontalEncender (const int numLed)

Apagar el LED que queramos.

 void LedFrontal ()

Enciende los LED, mostrándonos donde se encuentra la línea a seguir.

 void LedFrontalFantastico ()

Función auxiliar que enciende y apaga los LED haciendo el efecto del coche fantástico.

 void Calibrar ()

Calibración. Utilizada para obtener el máximo y el mínimo de los sensores frontales. Esto se emplea para filtrar el resultado y realizar un mejor análisis.

 void ControladorRectas (int Distancia)

Calibración. Auxiliar para aumentar la velocidad en las rectas.

 void Estabilizar ()

Función principal que manda la información necesaria a los motores para corregir el error (la distancia del centro a la línea)

 void ControlPD (int P, int D)

Inicializa las constantes de proporcionalidad y derivativa.

 void AnalizarCamino ()

Obtenemos para donde tenemos que girar en la próxima intersección.

 void GirarCamino ()

Elimina los caminos por los que el robot no debe girar. Deshabilita los sensores que no están cerca de los caminos incorrectos. Haciendo creer al robot que solo existe una línea.

 void EnviarDatos ()

Enviar todos los datos por el puerto serie. Luego estos datos son leídos por el programa interfaz.

 void RecibirDatos ()

Recibe los datos que manda el programa interfaz para modificar las variables internas.

 void ModificarVariables ()

Modifica las variables internas, según los datos mandados por el programa interfaz.

 void EnviarDatosGiro (L3G _gyro)

Guarda el objeto gyro dentro de esta clase. Es llamado por la función principal main().

 void Rastreador ()

Función principal que recoge todos las funciones para que el robot pueda seguir una línea y mandar y recibir datos.

2.3.2. Interfaz gráfica .NET

Se trata de un programa diseñado con *Microsoft Visual Studio* por lo tanto apto para cualquier sistema Windows. Posee todas las ventanas, gráficas y formularios necesarios para monitorizar y controlar el robot rastreador. Mediante la conexión serie *bluetooth* podemos comunicarnos fácilmente con él e intercambiar información. El programa consta de varias pestañas en las que podemos navegar libremente. Con esto conseguimos empaquetar todo el contenido en un solo programa, mejorando la estética y su manejo. Únicamente tendremos ventanas emergentes que activaremos para observar el comportamiento del robot mediante gráficas. Estas gráficas mostrarán toda la información que podamos obtener del robot. A continuación se explicará más

detenidamente el funcionamiento de dicho programa, analizaremos las funcionalidades de cada pestaña y se mostrará cómo realizar un correcto manejo del mismo.

Pestaña de inicio

En esta pestaña de inicio podemos elegir que puerto serial utilizar para la comunicación. Tenemos una lista de los posibles puertos a elegir. Teniendo en cuenta que solo lo podemos conectar vía *bluetooth* o vía USB, solo nos aparecerán dos. En general nos mostraría todos los puertos compatibles con el bus serial. Marcamos el que queramos, normalmente utilizaremos el puerto *bluetooth*, y pulsaríamos el botón de Aceptar. Esperamos un instante y en la zona de información nos aparece que el robot se encuentra conectado con el nombre del puerto elegido.

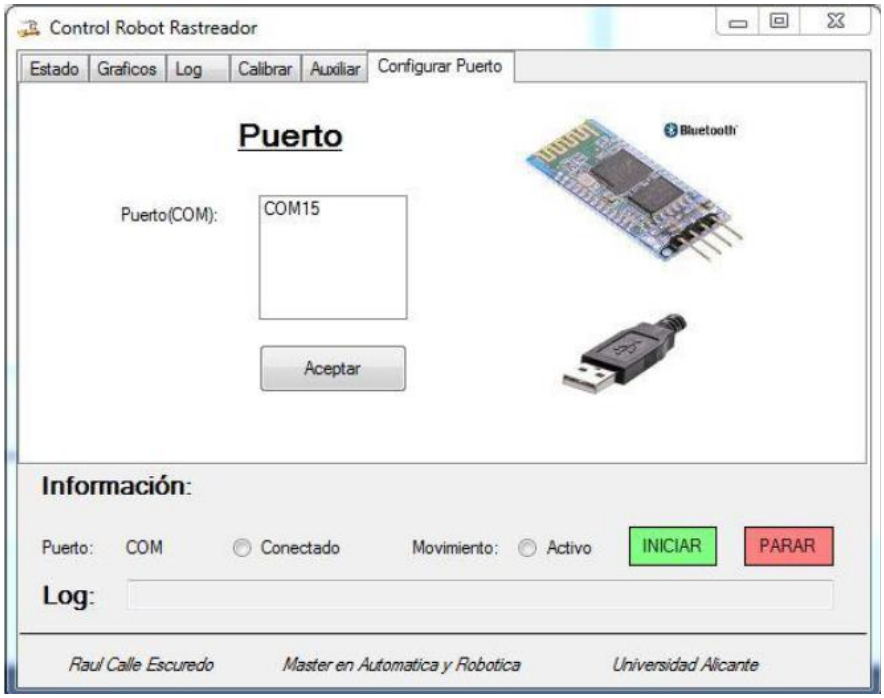

Figura 10. Interfaz: pestaña de configuración del puerto.

Pestaña de estado

Esta pestaña se encarga de enviar información al robot. Estos datos son: la velocidad inicial, la constante de proporcionalidad y derivativa. Podemos observar que el formulario donde se encuentran la información está bloqueado, esto es por seguridad, para poder modificar la información tenemos que darle al botón de "Modificar" y automáticamente de desbloquea pudiendo introducir el valor deseado. Si esperamos

posicionados en el recuadro nos salta una ayuda, con los límites superior e inferior (Velocidad: 10-120).

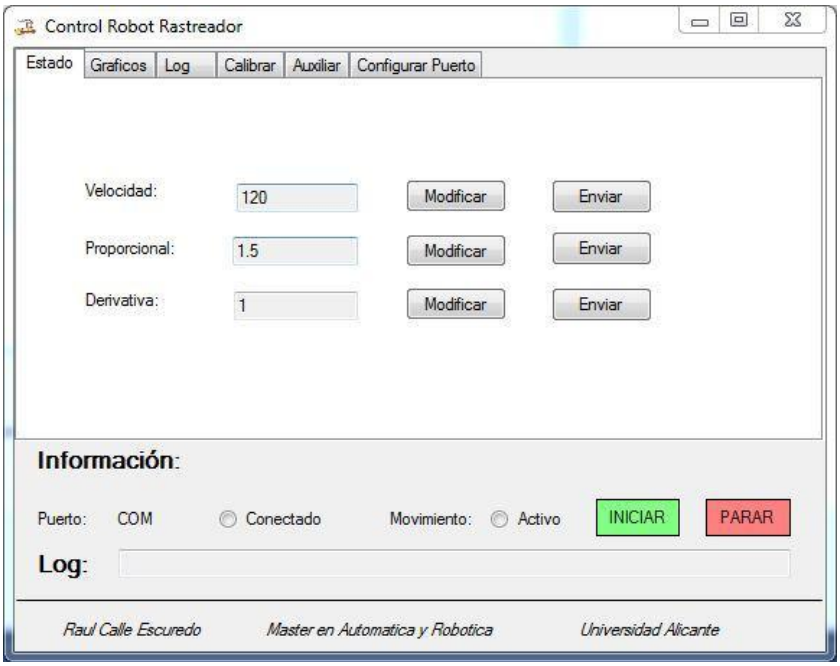

Figura 11. Interfaz: pestaña de estado.

Pestaña de Calibrar

Esta parte es muy importante ya que necesitamos un reconocimiento lumínico previo.

Como trabajamos con sensores ópticos necesitamos saber la cantidad de luz que tenemos en el circuito. A mayor luz mejor responden los sensores, y en la programación no necesitamos filtrar tanto el resultado. Cuando clicamos en el botón de calibrar, el programa envía una señal al robot seguida de una serie de datos, como la velocidad, la proporcional y la derivativa (del control PD). Automáticamente el robot realiza un giro estático de derecha a izquierda y viceversa, para que todos los sensores capten la línea negra del circuito. Obtiene su valor más alto de luminosidad que correspondería con la línea negra, y el valor más pequeño, asociado al color blanco de la pista. Toda esta información se mostrará por pantalla después del proceso de calibrado.

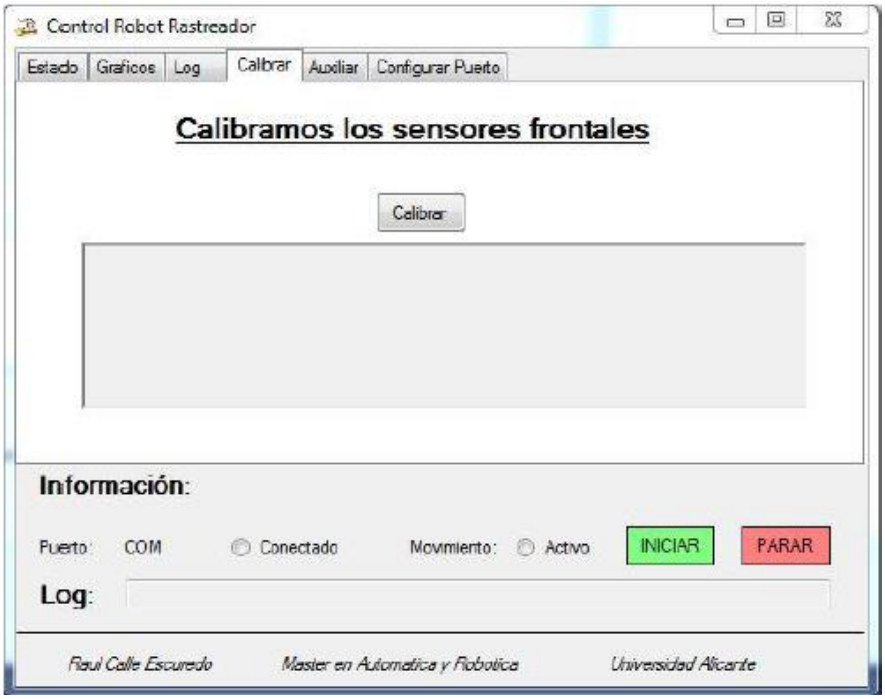

Figura 12. Interfaz: pestaña de calibración

Pestaña de Gráficos

Esta pestaña es la más completa de todas. En ella podemos obtener toda la información que proviene del robot, plasmada en unas gráficas. Estos datos únicamente son mostrados cuando el robot se encuentra en movimiento.

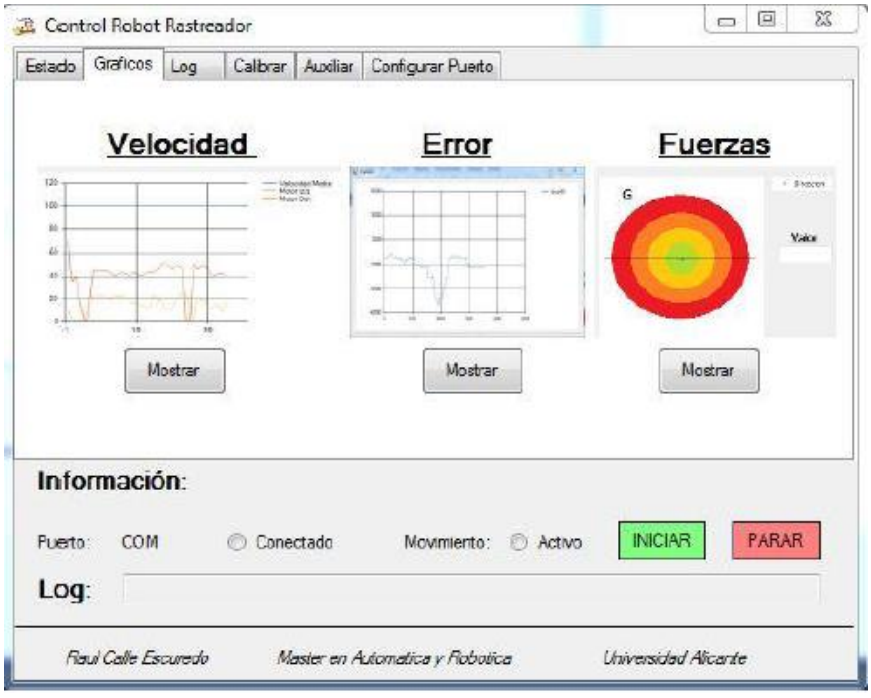

Figura 13. Interfaz: pestaña de gráficos

3. RESULTADOS

La parte de resultados se compone, por un lado, de la parte educativa donde se han modificado las guías docentes de las asignaturas de Electromecánica y Sistemas de Control Automático para incorporar esta herramienta, y por otro lado, la herramienta resultante y su funcionalidad después de su desarrollo y programación.

3.2.1. Modificación guías docentes.

En este apartado se presentan las guías docentes de las asignaturas de Electromecánica y Sistemas de Control Automático, que se han modificado para la incorporación de este robot modular, dentro de su Metodología.

Electromecánica (37804)

Actividad	Horas Presenciales /Horas No presenciales	Metodología
Clase de teoría	37,5 P	Constarán de lecciones magistrales, debates sobre aspectos del temario y de los seminarios, y de resolución de problemas con participación de los alumnos. Se considerarán debates virtuales soportados por alguna plataforma de b-Learning. Las lecciones magistrales se impartirán mediante presentaciones multimedia, mostrando casos de análisis reales sobre un robot rastreador desarrollado y diseño con herramientas de software adecuadas.
Laboratorios virtuales		Mediante el uso de laboratorios virtuales, se podrán mostrar ejemplos de sistemas reales simulados. De esta manera el alumno podrá experimentar de una manera virtual con un sistema real y poder realizar un aprendizaje constructivo a través del cambio de parámetros del sistema simulado. Además, el empleo de los laboratorios virtuales permite una vía de auto-aprendizaje en horario no presencial para los alumnos.
Práctica de Laboratorio	22,5 P	En ellas los alumnos realizarán el diseño y programación de sistemas reales usando software de análisis, entornos de programación, y laboratorios virtuales y/o remotos. En concreto, se emplearán dos: una maqueta que emula el comportamiento de un sistema de bombeo y un robot modular sigue-líneas.

Sistemas de Control Automático (37802)

Actividad	Horas Presenciales /Horas No presenciales	Metodología
Clase de teoría	30P/55NP	Constarán de lecciones magistrales, debates sobre aspectos del temario y de los seminarios, y de resolución de problemas. Las lecciones magistrales se impartirán usando presentaciones multimedia, resolviendo casos de análisis y diseños con herramientas de software adecuadas, y con un uso importante de laboratorios virtuales y sistemas reales de control, tales como un robot rastreador.
Tutorías grupales	11P/2,5NP	Se realizaran tutorías a grupos de alumnos para atender dudas sobre temas específicos y desarrollar trabajos o proyectos en equipo.
Seminarios	4P/2,5NP	Participarán ponentes que expongan aplicaciones prácticas reales de temas abordados en la asignatura. Los alumnos deberán realizar informes sobre las ponencias, y estas serán analizadas y discutidas en las clases de teoría.
Práctica de Laboratorio	15P/30NP	Los alumnos realizarán un proyecto consistente en la implementación y control de controladores tipo PID sobre un sistema de bombeo y de un robot rastreador. Los alumnos podrán probar en tiempo real los distintos parámetros de ajuste PID sobre los sistemas reales y poder visualizar el comportamiento del sistema real con diferentes controladores

3.2.2. Robot modular: funcionalidades educativas del sistema

Calibración

Se va a mostrar el resultado de calibrar el robot según la cantidad lumínica que exista en la sala. La información captada por los sensores varía según la cantidad de luz. Probaremos la calibración en interiores con bastante luz y con poca luz.

- Interiores (Luminosidad alta):

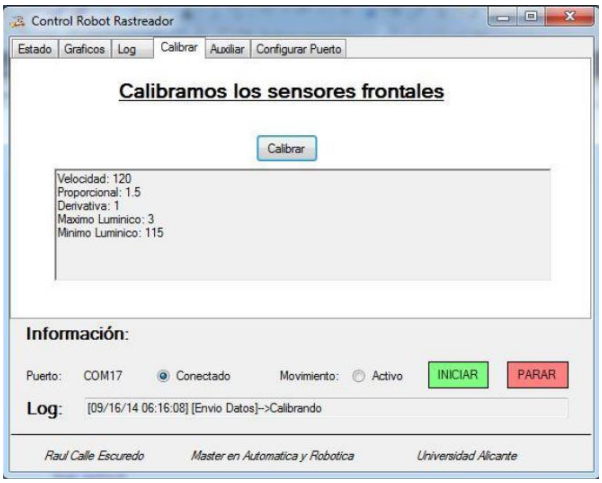

Figura 14. Calibración I

- Interiores (Luminosidad baja):

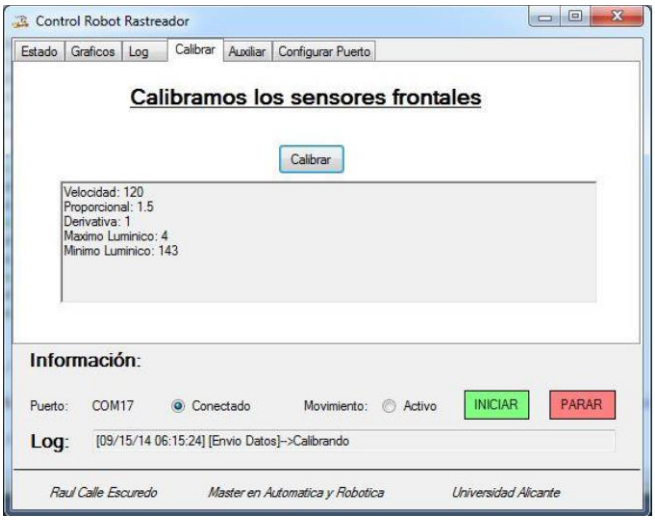

Figura 15. Calibración II

Podemos observar que el valor mínimo y máximo captado por los sensores varía. En la primera prueba, vemos como el máximo ronda los 100, mientras que en la segunda está cerca de los 150. Con el mínimo no vemos apenas diferencia, ya que el blanco es menos sensible a la cantidad de luz. Matizamos que el máximo correspondería a la línea negra y mínimo a la zona blanca de la pista.

Cambio de velocidad

En este apartado analizaremos las gráficas, a medida que modificamos la velocidad de los motores. Al aumentar la velocidad, ocurren varias cosas: Lo que observamos a primera vista es que las gráficas de velocidades en los motores, del error y de los datos

del giroscopio, se hacen más variantes, es decir, sufren más oscilaciones y se producen picos más altos.

- Gráfica de velocidad:

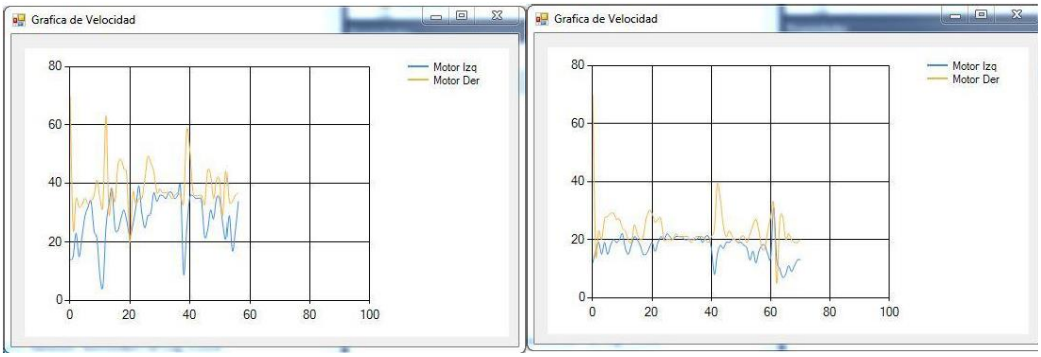

Figura 16. Gráficas de distintas velocidades

- Gráfica de error PD:

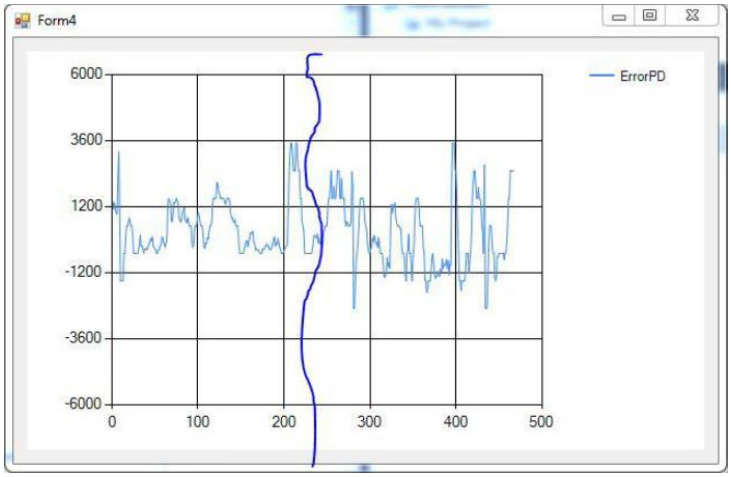

Figura 17. Gráfica de error PD

- Giroscopio: Mostramos los valores más altos que se generan según las distintas velocidades.

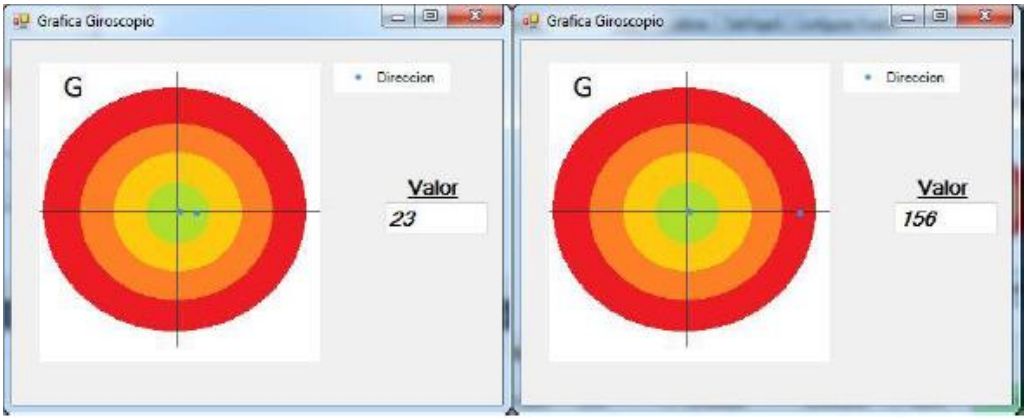

Figura 18. Giroscopio con velocidad 60 y 120.

Cambios en el control PD

Para controlar el robot, y que siga las líneas de manera correcta sin desviarse del camino, en necesario la utilización del control PD. Como hemos visto antes, la interfaz posee una pestaña en la que podemos enviarle las constantes de proporcionalidad y derivativa, de forma instantánea y ver los resultados en la gráfica de error. Para esta prueba vamos a realizar diversas modificaciones en las constantes y ver el resultado en dichas gráficas. Hay que añadir que la resolución de las pruebas no es muy buena ya que tenemos que limitar el envió de datos por el puerto. Esto es debido a que solo tenemos una velocidad relativamente baja de comunicación. Se mostrarán una serie de gráficas en las que aparecerán los errores según los valores de las constantes, junto con un texto que explica lo que le ocurre al robot en cada una de ellas.

- Proporcional (1) Derivativa (1): El sistema corrige el error lentamente aunque el robot consigue seguir la línea, si se encuentra con una curva muy pronunciada se sale del circuito.

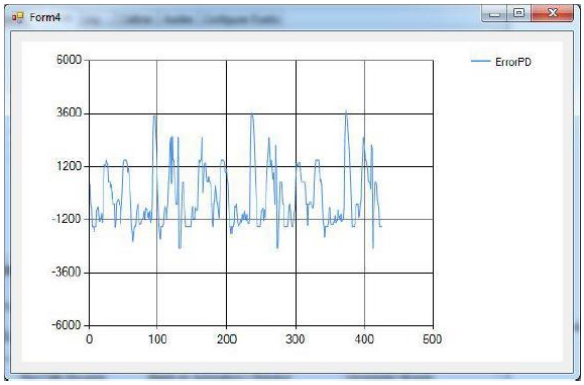

Figura 19. Gráfica error PD. P(1) D(1)

- Proporcional (1.5) Derivativa (1): Al ir aumentando la proporcional vemos como el robot corrige más rápido el error, los picos son más finos. El problema es que se genera mucho movimiento oscilante. Si nos fijamos en los puntos más oscuros de la gráfica.

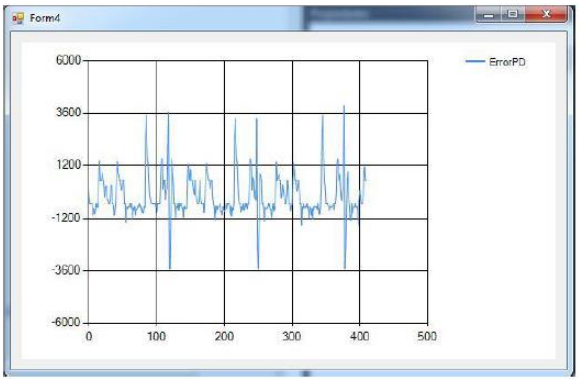

Figura 20. Gráfica error PD. P(1.5) D(1)

- Proporcional (1.5) Derivativa (1.5): Para reducir este movimiento oscilante, aumentamos la derivada. Con esto conseguimos disminuir este movimiento extraño. Si nos fijamos bien vemos que los picos no son tan altos (curvas) y la gráfica está más clara, no hay tantas oscilaciones en los puntos después de las curvas.

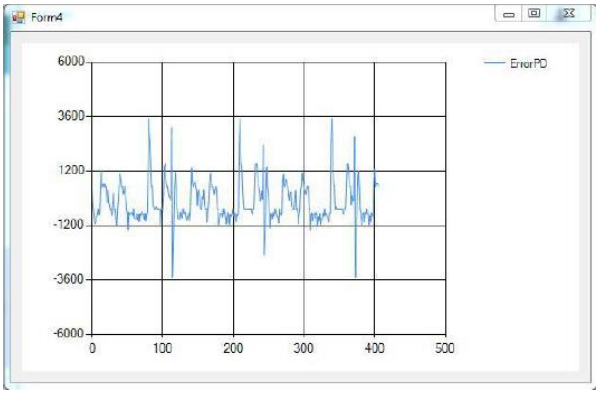

Figura 22. Gráfica error PD. P(1.5) D(1.5)

4. CONCLUSIONES

En este artículo se ha descrito el diseño y realización de un robot modular para el desarrollo tanto de competencias genéricas como específicas, en las enseñanzas de electrónica, control y programación Master de Automática y Robótica de la Escuela Politécnica Superior de la Universidad de Alicante. La principal característica que se pretende con este proyecto es permitir al alumno de una serie de experimentos reales para poder testar diferentes controladores y ver el comportamiento en tiempo real del sistema o robot modular.

El uso de la herramienta propuesta permite reforzar el aprendizaje activo del estudiante ya que debe realizar actividades y ejercicios con el objetivo de comprender

los conocimientos adquiridos. De esta manera, el estudiante interioriza los conceptos y comprende su alcance al aplicarlos a un entorno realista.

5. DIFICULTADES ENCONTRADAS

Dado que los alumnos en sus estudios previos no adquieren los suficientes conocimientos en el ámbito del control, las mayores dificultades que se encontraron fue el hecho de tener que explicar desde el inicio toda la teoría de control. Dado que esta plataforma tiene una gran parte de control, dicha dificultad se redujo ya que los alumnos asimilaban mejor los conceptos de los controladores P, D y PD con el uso de ejemplos prácticos y fácilmente visibles mediante el robot modular desarrollado.

6. PROPUESTAS DE MEJORA Y PREVISIÓN DE CONTINUIDAD

Como propuesta de mejora se plantea el uso de reconocer y guardar el camino, como a su vez la capacidad de tomar decisiones de giro según las marcas dispuestas para ello.

7. REFERENCIAS BIBLIOGRÁFICAS

Arduino, 2014. Especificaciones técnicas de la placa Arduino Due. Online: http://arduino.cc/en/Main/arduinoBoardDue

Huber, G.L. (2008). Active Learning and Methods of teaching. *Revista de Educación, 2008, 59-81.* On-line: http://www.revistaeducacion.mec.es/re2008/re2008_04.pdf

Shuell, T.J. (1986). Cognitive Conceptions of Learning. *Review of Educational Research, 56(4), 411-436.*