



**Escuela Superior de Ingenieros de Sevilla**

---

**PPCarro: Vehículo móvil basado en péndulo invertido**

---

Alberto Prieto  
Antidio Viguria  
Benito José Vela  
Mirko Fiacchini  
Ramón Cano

Trabajo de los Cursos de Doctorado:

Control No Lineal Aplicado  
Sistema de Control No Lineal  
Robótica Industrial

**INDICE GENERAL:****Memoria I:**

DESCRIPCIÓN DE LA ESTRUCTURA MECÁNICA Y DE LOS COMPONENTES HARDWARE.

<b>CAPITULO I. MEMORIA DESCRIPTIVA. ....</b>	<b>7</b>
1.1. MOTIVACION Y OBJETIVOS.....	7
1.2. ESTRUCTURA Y TRACCION. ....	7
1.2.1. CHASIS.....	7
1.2.2. RUEDAS.....	8
1.2.3. ADAPTADOR EJE-RUEDA.....	8
1.2.4. MOTORES.....	9
1.2.5. TORNILLOS DE SUJECIÓN.....	10
1.2.6. BATERIAS.....	10
1.3. ELECTRÓNICA.....	12
1.3.1. MICROCONTROLADOR.....	12
1.3.2. INCLINÓMETRO.....	13
1.3.3. ENCODERS.....	14
1.3.4. CONTROLADORA DE LOS MOTORES.....	15
1.3.6. CAJA DE LA ELECTRÓNICA.....	16
1.3.7. CAJA DE CONTROL DEL VEHÍCULO.....	17
1.4. EQUIPOS AUXILIARES.....	18
1.4.1. CARGADOR DE BATERIAS.....	18
1.4.2. RADIO-MÓDEM.....	18
1.4.3. SISTEMA DE PROGRAMACION DEL MICROCONTROLADOR.....	19
1.4.4. PUÑO DE DIRECCIÓN.....	19
1.5. EQUIPO COMPLETO.....	20
<b>CAPITULO II. PRESUPUESTO.....</b>	<b>21</b>
2.1. ESTRUCTURA Y TRACCION.....	21
2.2. ELECTRONICA.....	21
2.3. EQUIPOS AUXILIARES.....	21
2.4. OTROS CONCEPTOS.....	22
2.6. RESUMEN DEL PRESUPUESTO.....	22
<b>CAPITULO III. PLANOS.....</b>	<b>23</b>
3.1. ESTRUCTURA GENERAL.....	23
3.1.1. MOTORES.....	23
3.1.2. ADAPTADORES EJE-RUEDA.....	23
3.1.3. DESPIECE DE LA ESTRUCTURA.....	23
3.1.2. ADAPTACION DE LOS ENCODERS A LOS MOTORES.....	23

**Memoria II:**

SOFTWARE DESARROLLADO PARA EL MICROCONTROLADOR Y LA APLICACIÓN DE PC.

<b>CAPITULO I. SOFTWARE DEL MICROCONTROLADOR.....</b>	<b>26</b>
1.1. INTRODUCCIÓN.....	26
1.2. ARQUITECTURA DE TINYOS.....	26
1.3. ENTORNO DE DESARROLLO DE TINYOS.....	26
1.4. EL LENGUAJE DE PROGRAMACIÓN NESC.....	27
1.4.1. MODELO DE PROGRAMACIÓN.....	28
1.5. ESTRUCTURA DEL PROGRAMA DESARROLLADO.....	29
1.6. INSTALACIÓN DEL ENTORNO DE DESARROLLO.....	30
1.7. ¿CÓMO SE INTRODUCE UN NUEVO CONTROLADOR?.....	31
1.8. COMPILACIÓN Y CARGA DE UN PROGRAMA.....	31

1.9. DOCUMENTACIÓN ADJUNTA.....	31
<b>CAPITULO II. SOFTWARE DE LA APLICACIÓN DE PC.....</b>	<b>32</b>
2.1. INTRODUCCIÓN.....	32
2.2. OBJETIVOS.....	32
2.3. REQUISITOS MÍNIMOS.....	33
2.3.1. <i>HARDWARE</i> .....	33
2.3.2. <i>SOFTWARE</i> .....	33
2.4. INSTALACIÓN.....	33
2.5. DESINSTALACIÓN.....	33
2.6. CARACTERÍSTICAS Y MANEJO DEL MONITOR.....	34
2.6.1. <i>USO</i> .....	34
2.6.2. <i>PLANTILLAS</i> .....	35
2.6.3. <i>CONFIGURANDO LA COMUNICACIÓN</i> .....	40
2.6.4. <i>CONFIGURANDO LA REPRESENTACIÓN GRÁFICA</i> .....	41
2.6.5. <i>EXPORTANDO DATOS A MATLAB</i> .....	43
2.7. FUNCIONAMIENTO DE LOS ELEMENTOS PRINCIPALES.....	43
2.7.1. <i>INTERFAZ GRÁFICA</i> .....	43
2.7.2. <i>MÓDULO DE COMUNICACIONES</i> .....	43
2.7.3. <i>PARSER XML</i> .....	45
2.7.4. <i>INTERFAZ MATLAB</i> .....	46

**Memoria III:**

DISEÑO, SINTONIZACIÓN Y SIMULACIÓN DE CONTROLADORES.

<b>CAPITULO I. DESCRIPCIÓN Y MODELADO DEL SISTEMA.....</b>	<b>49</b>
1.1 DESCRIPCIÓN DEL SISTEMA.....	49
1.2 MODELO DEL SISTEMA.....	49
1.2.1 <i>VEHÍCULO CON TRACCIÓN DIFERENCIAL</i> .....	49
1.2.2 <i>PÉNDULO INVERTIDO SOBRE BASE MÓVIL</i> .....	50
<b>CAPITULO II. CONTROLADORES. DISEÑO Y SIMULACIÓN.....</b>	<b>51</b>
2.1 CONTROL LQR.....	51
2.1.1 <i>UBICACIÓN DE POLOS CON LQR</i> .....	51
2.2 CONTROLADOR NO LINEAL: ASTOLFI-KALIORA.....	53
2.2.1 <i>LINEALIZACIÓN PARCIAL</i> .....	53
2.2.2 <i>ESTABILIZACIÓN DEL PÉNDULO POR MOLDEO DE LA FUNCIÓN DE ENERGÍA</i> .....	54
2.2.3 <i>ESTABILIZACIÓN DE LA VELOCIDAD</i> .....	55
2.2.4 <i>SINTONIZACIÓN DEL CONTROLADOR NO LINEAL</i> .....	55
2.3 SIMULACIONES.....	56
2.4 EFECTO DE LA SATURACIÓN, TIEMPO DE MUESTREO Y RETRASO.....	60





**Escuela Superior de Ingenieros de Sevilla**

---

**DESCRIPCIÓN DE LA ESTRUCTURA MECÁNICA Y DE LOS  
COMPONENTES HARDWARE**

---

Alberto Prieto  
Antidio Viguria  
Benito José Vela  
Mirko Fiacchini  
Ramón Cano

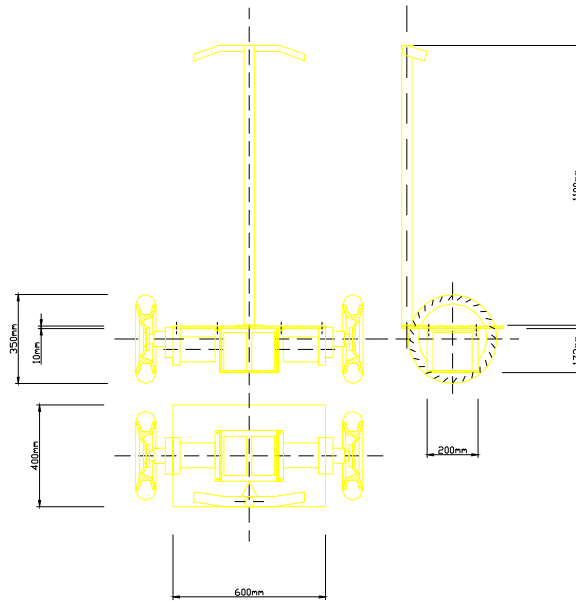
Trabajo de los Cursos de Doctorado:

Control No Lineal Aplicado  
Sistema de Control No Lineal  
Robótica Industrial

**INDICE.**

<b>CAPITULO I. MEMORIA DESCRIPTIVA. ....</b>	<b>7</b>
1.1. MOTIVACION Y OBJETIVOS.....	7
1.2. ESTRUCTURA Y TRACCION. ....	7
1.2.1. CHASIS.....	7
1.2.2. RUEDAS.....	8
1.2.3. ADAPTADOR EJE-RUEDA.....	8
1.2.4. MOTORES.....	9
1.2.5. TORNILLOS DE SUJECIÓN.....	10
1.2.6. BATERIAS.....	10
1.3. ELECTRÓNICA.....	12
1.3.1. MICROCONTROLADOR.....	12
1.3.2. INCLINÓMETRO.....	13
1.3.3. ENCODERS.....	14
1.3.4. CONTROLADORA DE LOS MOTORES.....	15
1.3.6. CAJA DE LA ELECTRÓNICA.....	16
1.3.7. CAJA DE CONTROL DEL VEHÍCULO.....	17
1.4. EQUIPOS AUXILIARES.....	18
1.4.1. CARGADOR DE BATERIAS.....	18
1.4.2. RADIO-MÓDEM.....	18
1.4.3. SISTEMA DE PROGRAMACION DEL MICROCONTROLADOR.....	19
1.4.4. PUÑO DE DIRECCIÓN.....	19
1.5. EQUIPO COMPLETO.....	20
<b>CAPITULO II. PRESUPUESTO.....</b>	<b>21</b>
2.1. ESTRUCTURA Y TRACCION.....	21
2.2. ELECTRONICA.....	21
2.3. EQUIPOS AUXILIARES.....	21
2.4. OTROS CONCEPTOS.....	22
2.6. RESUMEN DEL PRESUPUESTO.....	22
<b>CAPITULO III. PLANOS.....</b>	<b>23</b>
3.1. ESTRUCTURA GENERAL.....	23
3.1.1. MOTORES.....	23
3.1.2. ADAPTADORES EJE-RUEDA.....	23
3.1.3. DESPIECE DE LA ESTRUCTURA.....	23
3.1.2. ADAPTACION DE LOS ENCODERS A LOS MOTORES.....	23





**Figura 2: Chasis.**

### 1.2.2. RUEDAS.

Se utilizarán ruedas neumáticas, con cámara de aire y llanta de acero, de 410mm de diámetro (llanta 16”), utilizadas comúnmente en remolques de automóviles.



**Figura 3: Ruedas.**

Puede encontrarse en:  
[TIENDAS FEUBERT.](#)

### 1.2.3. ADAPTADOR EJE-RUEDA.

Esta pieza es necesaria para unir el eje del motor con la llanta de la rueda. Se une a ambas piezas mediante tornillos. Elegimos el modelo NPC-PH448 de la casa NPC Robotics.



**Figura 4: Adaptador Eje-Rueda.**

Puede encontrarse en:

<http://www.npcrobotics.com/products/viewprod.asp?prod=21&cat=18&mode=gfx>

#### 1.2.4. MOTORES.

Para la tracción de nuestro robot móvil se han elegido motores de corriente continua alimentados a 24Vdc, que llevan incorporados una caja reductora de relación 20:1. Concretamente el modelo NPC-T64 que comercializa la compañía NPC Robotics.



**Figura 5: Motor de DC.**

Puede encontrarse en:

<http://www.npcrobotics.com/products/viewprod.asp?prod=42&cat=20&mode=gfx>

Adjuntamos tabla con características de par, intensidad y potencia consumida a diferentes regímenes de velocidad.

Dynamometer Test Results			
Torque in inch/pounds	Amps	RPM	HP
30	8.6	238	.11
60	12.5	230	.22
90	16.2	225	.32
120	20.0	218	.41
150	23.5	211	.52
180	27.5	206	.62
210	31.6	200	.71
240	35.1	194	.81
270	39.2	187	.89
300	43.1	181	.95 →
825	110	STALL	

Figura 6: Tabla de par / intensidad del Motor de DC.

### 1.2.5. TORNILLOS DE SUJECIÓN.

Se han elegido varios tipos de tornillos para ensamblar las diferentes piezas principales del chasis y motores, que pasamos a relacionar:

- Sujeción de los motores a la placa horizontal (para cada motor):  
2 tornillos 5/16-24-UNF de longitud 25 mm, resistencia 12,9 y cabeza DIN912.
- Sujeción de los adaptadores al eje del motor (para cada motor):  
4 tornillos 5/16-24-UNF de longitud 25 mm, resistencia 12,9 y cabeza DIN912.
- Sujeción de los adaptadores a la llanta de la rueda (para cada rueda)  
4 tornillos 5/16-24-UNF de longitud 16 mm, resistencia 12,9 y cabeza DIN912.
- Sujeción del compartimento de baterías a la placa horizontal:  
4 tornillos Métrica 8mm de longitud 10 mm, resistencia 8,8 y cabeza DIN933.  
Incluye arandela Grower-B.
- Sujeción del manillar (mástil en forma de 'T' a la placa horizontal):  
4 tornillos Métrica 8mm de longitud 16 mm, resistencia 8,8 y cabeza DIN933.  
Incluye arandela Grower DIN127 y tuerca DIN933 de 8mm.

### 1.2.6. BATERIAS.

El sistema de alimentación está compuesto de dos baterías de 12V (conectadas en serie para dar 24Vdc), que alimentarán los motores y la electrónica. Se ha elegido el modelo NPC-B1412 de la casa NPC Robotics, de características:

- 12V, 14Ah.
- Descarga en 3 min. a 90A.
- Intensidad máxima 300<sup>a</sup>.
- Tamaño 6" W x 5.8" H x 3.5" D

Por último, comentar que se ha realizado una estimación de la duración de las baterías en el vehículo y al menos deberían durar una hora.



**Figura 7: Baterías.**

Puede encontrarse en:

<http://www.npcrobotics.com/products/viewprod.asp?prod=17&cat=15&mode=gfx>

La electrónica no se alimenta directamente desde estas baterías, sino que a partir de los 24V que dan éstas y con un simple circuito formado por reguladores de tensión conseguimos las tensiones necesarias (5V y 9V) para los distintos componentes electrónicos.

### 1.3. ELECTRÓNICA.

#### 1.3.1. MICROCONTROLADOR.

El control de nuestro robot móvil se ha implementado usando la placa microcontroladora MAVRIC-IIB (modelo MAV2BST) de la compañía BDMICRO. Esta placa de desarrollo tiene la ventaja de que ya viene lista para usar, es decir, que ya trae su propio circuito de alimentación, su convertidor de tensiones para poder usar los 2 puertos serie con el nivel de tensiones del estándar RS-232, acceso a todos los puertos del microcontrolador e incluso pines especiales para las señales PWM. Por último comentar que se han utilizado los siguientes puertos y entradas y salidas digitales para la comunicación con los distintos dispositivos:

- **Alimentación:** Pines V+Input y GND, entre 5.5V y 15V.
- **Puerto serie 0:** comunicaciones con la placa que controla los 2 motores, mediante un protocolo de comunicaciones que se describe en el documento referente al software desarrollado.
- **Puerto serie 1:** comunicaciones con el PC, este canal de comunicaciones se utiliza para dar comandos al controlador del vehículo (parada, comienzo, parada de emergencia) y para cargar valores de los distintos controladores utilizados y para representar gráficamente datos de distintas variables del vehículo y guardarlas en ficheros compatibles con Matlab para su posterior estudio.
- **Entrada analógica 0:** señal proveniente del inclinómetro, una vez digitalizada la señal analógica se realiza una transformación lineal para pasarla a radianes.
- **Entrada analógica 1:** señal proveniente del puño que indica el sentido de la dirección. Ambas señales analógicas varían entre 0V y 5V.
- **Puerto ISP:** puerto utilizado para programar el microcontrolador y se conecta al programador AVRISP.
- **Salidas digitales:** la salida 1 del puerto B se utiliza para indicarle a la controladora de emergencia que realice una parada de emergencia y las salidas 0 y 2 del puerto B se utilizan para iluminar los LEDs que indican el estado del microcontrolador, estos son:
  - LED siempre encendido: el controlador está parado.
  - LED parpadeando: el controlador está activo y controlando el vehículo.
  - LED apagado: se ha producido una parada de emergencia.
- **Entradas digitales:** la entrada 0 del puerto A está conectada a un pulsador y este pulsador se utiliza por motivos de seguridad. El funcionamiento es el siguiente, cuando el controlador está activado este pulsador debe estar pulsado y si se lleva más de medio segundo sin ser pulsado entonces el controlador realiza una parada de emergencia. También se utilizan las entradas 0 y 1 del puerto D para la lectura de las señales provenientes de los encoders.

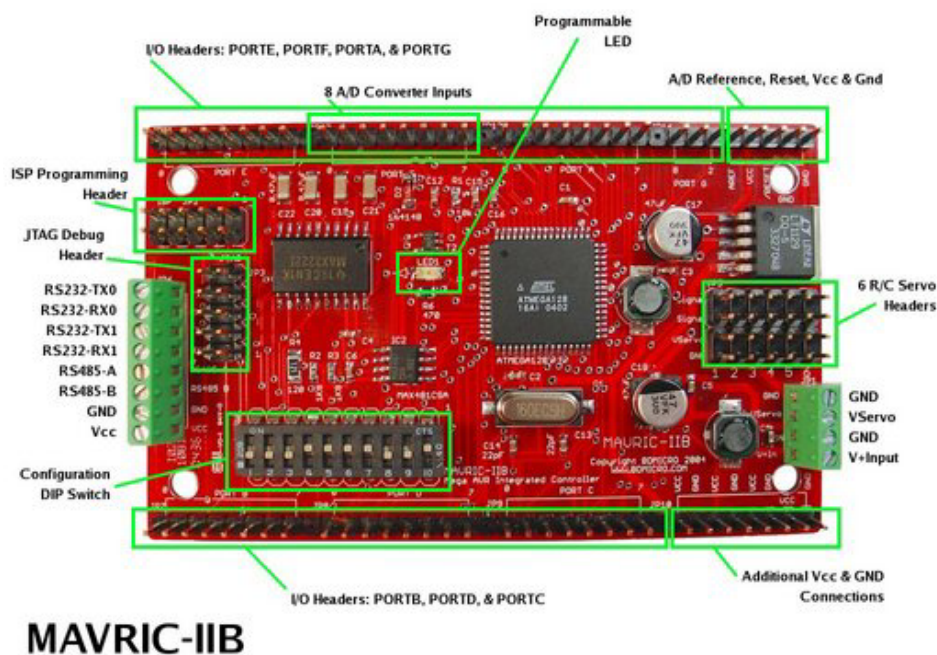


Figura 8: Microcontrolador.

Puede encontrarse en:

<http://www.bdmicro.com/mavric-iib/>

Se adjunta fichero con características técnicas y manual de instrucciones (este documento junto con el resto de manuales puede encontrarse en el directorio *Manuales* del CD adjunto a esta memoria).



"mavric-iib  
(micro).pdf"

### 1.3.2. INCLINÓMETRO.

El Inclinómetro elegido es el modelo FAS-G de la Compañía MicroStrain. Este inclinómetro de 360° tiene una resolución menor de 0.1 grados y una repetibilidad de 0.10 grados. Además podemos comunicarnos con él mediante el puerto serie a una velocidad de 19200bps o mediante una señal analógica (utilizando un convertidor D/A de 12 bits) cuya frecuencia de refresco es la misma que la señal proveniente del puerto serie. En nuestro caso se ha utilizado la señal analógica para medir la orientación del vehículo, ya que teníamos ocupado los 2 puertos series que tiene el microcontrolador (comunicaciones con el PC y con la placa controladora de los motores).



Figura 9: Inclinómetro.

Puede encontrarse en:

<http://www.microstrain.com/fas-g.aspx>

Se adjunta fichero con características técnicas y manual de instrucciones.



"FASG\_usermanual  
(inclinometro).pdf"

### 1.3.3. ENCODERS.

Para medir la velocidad de rotación de cada rueda se ha elegido el encoder modelo E5S de la compañía US Digital. Debido a que sólo estamos interesado en calcular la velocidad de las ruedas y no su posición, se ha utilizado únicamente una de las 2 señales de los encoders y para calcular de la velocidad lo que se hace es contar el número de pulsos de cada encoder por cada ciclo de control.

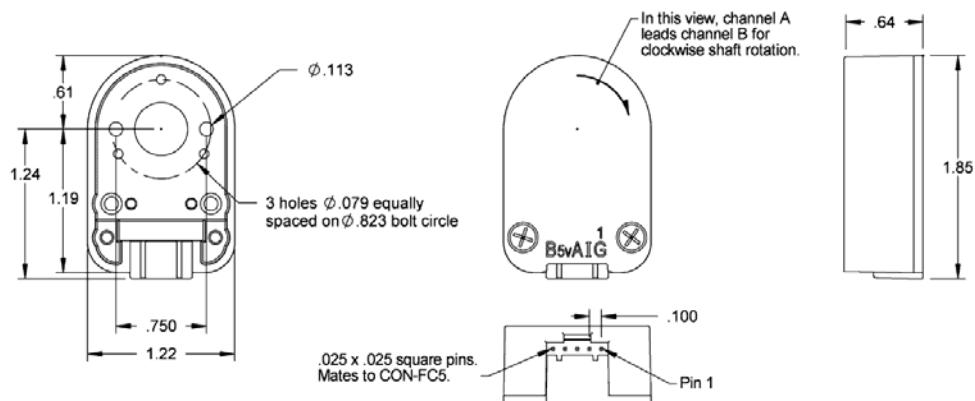


Figura 10: Encoder.

Puede encontrarse en:

<http://www.usdigital.com/products/e5s/>

Hemos añadido un plano donde detallamos cómo adaptar estos encoders a los motores seleccionados. Para ello se diseñó un pequeño soporte en PVC para acoplar el encoder en el interior del hueco trasero del motor. El proceso es el siguiente:

- Desmontar tapa trasera del motor.
- Pegar el casquillo al eje del motor con soldadura fría (el pegamento metálico tardará 12 horas en solidificar totalmente).

- Atornillar el cuerpo del Encoder al soporte de PVC.
- Colocar el soporte de PVC y atornillar a los pernos que asoman en el interior del motor.
- Colocar el disco del encoder, asegurarlo (el encoder trae su propio destornillador con cabeza hallen), montar electrónica y enchufar cable de datos.
- Cerrar tapa del motor.



Figura 11: Montaje del encoder

Adjuntamos hoja de características técnicas del encoder:



"e5s Data Sheet  
(encoder).pdf"

#### 1.3.4. CONTROLADORA DE LOS MOTORES.

Se ha elegido el modelo NPC-AX2550 de la casa RoboteQ, el cual se alimenta a 24V y tiene salida para 2 motores. Las consignas de control se le puede especificar de 3 formas distintas:

- Con señales PWM.
- Con señales analógicas.
- Con el puerto serie, a partir de un protocolo de comunicaciones.

La opción que se ha utilizado ha sido la del puerto serie, por lo que la controladora de los motores convertirá las consignas generadas por el Microcontrolador sobre los dos motores de corriente continua. La velocidad a la que se envían datos es de 9600bps y se utiliza el siguiente protocolo:

- En primer lugar se envían una !A o !B para indicar a cual de los 2 motores nos estamos refiriendo. Y dependiendo de si el carácter está en mayúscula o minúscula, el sentido de la rueda será positivo o negativo.
- En segundo lugar se le da un valor numérico en hexadecimal entre 0 y 7F para indicar la velocidad que se le quiere dar a cada uno de los motores.
- Por último se transmite un retorno de carro.

Finalmente comentar que además de la comunicación serie existe una línea digital para indicar desde el microcontrolador hacia la controladora de los motores, que esta última pase a estado de parada de emergencia.

En el fichero Anexo se adjunta descripción de las características técnicas y guía rápida de instalación y operación.



"ax2550man17-0201  
05 (driver).pdf"



Figura 12: Controlador de Motores.

Puede encontrarse en:

<http://www.npcrobotics.com/products/viewprod.asp?prod=59&cat=24&mode=gfx>

### 1.3.6. CAJA DE LA ELECTRÓNICA.

Se ha construido una caja de aluminio donde se ha instalado gran parte de la electrónica anteriormente comentada. En esta caja tenemos los siguientes componentes:

- Placa del microcontrolador.
- Circuito convertor de tensiones: a partir de los 24V de las baterías obtenemos los 9V y 5V necesarios para la electrónica.
- Programador del microcontrolador.
- Todas las señales de entrada y salida del microcontrolador.



Figura 13: Caja de aluminio con la electrónica

Por otra parte y con respecto a las conexiones, decir que los encoders y el inclinómetro están siempre conectados a la caja de la electrónica y que el resto de conexiones se realizan a partir de 3 conectores DB-9 y un conector utilizado para la alimentación. Cada uno de los conectores DB-9 se utilizan para:

- Conexión con la placa controladora de los motores.
- Conexión con el enlace inalámbrico, el puño que da las consignas de dirección y la caja de control del vehículo instalados ambos en la parte final de la barra vertical.
- Conexión utilizada para la programación del microcontrolador.



Figura 14: Lateral de la caja de la electrónica

### 1.3.7. CAJA DE CONTROL DEL VEHÍCULO.

Esta caja de plástico está instalada sobre el manillar del vehículo y está compuesta por:

- Interruptor que activa o desactiva la alimentación de la placa controladora de motores y por lo tanto de los motores.
- Pulsador utilizado como medida de seguridad, a partir del instante en el que el controlador está activado, si el pulsador lleva más de medio segundo sin estar pulsado, entonces el control se para y el microcontrolador indica a la placa controladora de los motores que realice una parada de emergencia.
- LED rojo que indica el estado del controlador, existen 3 posibles estados:
  - Si el LED está encendido el controlador está parado.
  - Si el LED parpadea, entonces el control está activo.
  - Si el LED está apagado es que se ha producido una parada de emergencia.

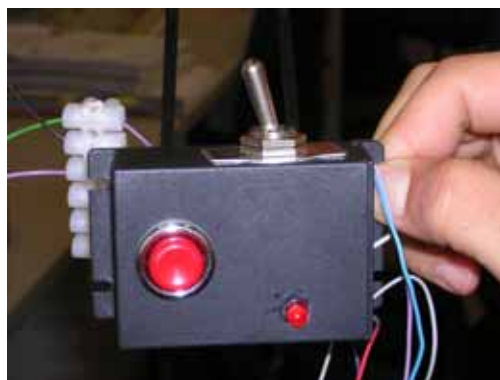


Figura 15: Caja de control del vehículo

## 1.4. EQUIPOS AUXILIARES.

### 1.4.1. CARGADOR DE BATERIAS.

La función de este dispositivo será recargar las dos baterías de 12V del sistema de potencia principal. Se ha elegido el cargador CBP.1000/12 de la marca TRQ. Debido al gran tiempo de carga de las baterías (unas 12 horas), se han comprado 2 cargadores para que se puedan cargar ambas a la vez.



Figura 16: Cargador de baterías.

### 1.4.2. RADIO-MÓDEM.

Para comunicar el sistema de control de nuestro robot con el PC de supervisión, se utilizará una comunicación inalámbrica serie. Se ha elegido el Modelo X9220 Bluetooth de la compañía Warwick Wireless. Esta comunicación inalámbrica es transparente tanto para el PC de supervisión como para el microcontrolador, es decir, los dispositivos se comunican como si hubiera un cable serie entre ellos. Este enlace inalámbrico está basado en tecnología Bluetooth, tiene un alcance de unos 100 metros en espacio abierto y se ha utilizado a una velocidad de 19200bps, aunque puede llegar a 115200bps.



Figura 17: Radio-Módem.

Puede encontrarse en:

<http://www.radiotelemetry.co.uk/WirelessModem.htm>

En el fichero Anexo se adjunta hoja de características técnicas.



"DS156 X9220  
Bluetooth (modem).pdf"

### 1.4.3. SISTEMA DE PROGRAMACION DEL MICROCONTROLADOR.

Para la programación de la placa microcontroladora es necesario utilizar el Programador Modelo AVRISP de la Compañía Atmel. Este programador tiene la ventaja de ser muy barato ya que utiliza el protocolo de programación ISP (In System Programming) sin embargo no permite la depuración de los programas como es el caso de los programadores con interfaz JTAG.



Figura 18: Sistema de Programación.

Puede encontrarse en:

<http://www.digikey.com/scripts/dksearch/dksus.dll?KeywordSearch>

### 1.4.4. PUÑO DE DIRECCIÓN.

Se ha utilizado un puño de aceleración de las motos eléctricas para indicar las referencias en la dirección del vehículo. Este puño está alimentado a 5V y da una señal que varía entre 0V y 5V en función de la rotación de éste.



Figura 19: Puño de dirección

### **1.5. EQUIPO COMPLETO.**

En la siguiente figura aparece todo el equipo, formado por los distintos componentes comentados en este capítulo y montado sobre la plataforma mecánica del vehículo. Además se puede observar un portátil que hace de PC de supervisión con el enlace inalámbrico para comunicarse con el microcontrolador instalado en el vehículo.



**Figura 20: Equipo completo**

**CAPITULO II. PRESUPUESTO.****2.1. ESTRUCTURA Y TRACCION.**

Cantidad	Unidad	Descripción	Precio Unitario	Importe
1	UD	CHASIS compuesto por:	268€/UD	268,00€
0,24	M2	- Placa de Aluminio 400x600x10mm	€/M2	
1,60	ML	- Tubo de Aluminio Ø40mm	€/ML	
8	HR	- Mano de Obra y varios	€/HR	
4	UD	Tornillos Métrica 8mm de longitud 10 mm, resistencia 8,8 y cabeza DIN933, incluyendo arandela Grower DIN127.	0,28€/UD	1,12€
4	UD	Tornillos Métrica 8mm de longitud 16 mm, resistencia 8,8 y cabeza DIN933, incluyendo arandela Grower DIN127 y tuerca DIN933 de 8mm.	0,28€/UD	1,12€
8	UD	Tornillos 5/16-24-UNF de longitud 16 mm, resistencia 12,9 y cabeza DIN912.	0,36€/UD	2,88€
12	UD	Tornillos 5/16-24-UNF de longitud 25 mm, resistencia 12,9 y cabeza DIN912.	0,43€/UD	5,16€
2	UD	Ruedas neumáticas de 410 mm de diámetro con cámara de aire y llanta de acero	24,95€/UD	49,90€
2	UD	Adaptador Eje-Rueda NPC-PH448 de NPC Robotics	20,00€/UD	40,00€
2	UD	Motoreductor DC 24Vdc de relación 20:1 modelo NPC-T64 de NPC Robotics	297,00€/UD	594,00€
2	UD	Batería NPC-B1412 de NPC Robotics	79,00€/UD	158,00€
		<b>Suma Capítulo 2.2</b>		<b>1.120,18€</b>

**2.2. ELECTRONICA.**

Cantidad	Unidad	Descripción	Precio Unitario	Importe
1	UD	Microcontrolador MAVRIC-IIB (modelo MAV2BST) de la compañía BDMICRO	139,00€/UD	139,00€
1	UD	Inclinómetro FAS-G de la Compañía MicroStrain	795,00€/UD	795,00€
2	UD	Encoder E5s de la casa US Digital	51,00€/UD	102,00€
1	UD	Controlador de Motores DC NPC-AX2550 de la casa RoboteQ	499,00€/UD	499,00€
		<b>Suma Capítulo 2.3</b>		<b>1.535,00€</b>

**2.3. EQUIPOS AUXILIARES.**

Cantidad	Unidad	Descripción	Precio Unitario	Importe
2	UD	Cargador de Baterías	35,00€/UD	70,00€
2	UD	Radiomodem X9220 Warwick Wireless	110€/UD	220,00€
1	UD	Sistema de Programación AVRISP de la Compañía Atmel	29,00€/UD	29,00€
1	UD	Cables y componentes electrónicos	40,00€/UD	40,00€
1	UD	Puño para las consignas de dirección	35,00€/UD	35,00€
		<b>Suma Capítulo 2.4</b>		<b>394,00€</b>

**2.4. OTROS CONCEPTOS.**

<b>Cantidad</b>	<b>Unidad</b>	<b>Descripción</b>	<b>Precio Unitario</b>	<b>Importe</b>
48	HR	Hora/Hombre de montaje e instalación	32,00€/HR	1.536,00€
96	HR	Hora/Hombre de programación y puesta en marcha	42,00€/HR	4.032,00€
24	HR	Hora/Hombre de pruebas y control de calidad	52,00€/HR	1.248,00€
1	UD	Redacción de Proyecto y confección de As-Builts	1.500,00€/UD	1.500,00€
		<b>Suma Capítulo 2.5</b>		<b>8.316,00€</b>

**2.6. RESUMEN DEL PRESUPUESTO.**

	<b>Capítulo</b>	<b>Descripción</b>		<b>Importe</b>
	2.1	Estructura y Tracción		1.122,18
	2.2	Electrónica		1.535,00
	2.3	Equipos Auxiliares		394,00
		<b>Total Presupuesto Componentes</b>		<b>3051,18€</b>
	2.4	Otros Conceptos		8.316,00
		<b>Total Presupuesto</b>		<b>11.367,18€</b>

## **CAPITULO III. PLANOS.**

Todos los planos se encuentran en el CD adjunto a la memoria en el directorio /Planos.

### **3.1. ESTRUCTURA GENERAL.**



Estructura.dwg

#### **3.1.1. MOTORES.**



npc-t64(motor).pdf

#### **3.1.2. ADAPTADORES EJE-RUEDA.**



npc-ph448(hub).pdf

#### **3.1.3. DESPIECE DE LA ESTRUCTURA.**



Despiece.dwg

#### **3.1.2. ADAPTACION DE LOS ENCODERS A LOS MOTORES.**



Encoders.dwg



**Escuela Superior de Ingenieros de Sevilla**

---

**SOFTWARE DESARROLLADO PARA EL MICROCONTROLADOR Y LA  
APLICACIÓN DE PC**

---

Alberto Prieto  
Antidio Viguria  
Benito José Vela  
Mirko Fiacchini  
Ramón Cano

Trabajo de los Cursos de Doctorado:

Control No Lineal Aplicado  
Sistema de Control No Lineal  
Robótica Industrial

**INDICE.**

<b>CAPITULO I. SOFTWARE DEL MICROCONTROLADOR.....</b>	<b>26</b>
1.1. INTRODUCCIÓN.....	26
1.2. ARQUITECTURA DE TINYOS .....	26
1.3. ENTORNO DE DESARROLLO DE TINYOS.....	26
1.4. EL LENGUAJE DE PROGRAMACIÓN NESC.....	27
1.4.1. <i>MODELO DE PROGRAMACIÓN</i> .....	28
1.5. ESTRUCTURA DEL PROGRAMA DESARROLLADO.....	29
1.6. INSTALACIÓN DEL ENTORNO DE DESARROLLO.....	30
1.7. ¿CÓMO SE INTRODUCE UN NUEVO CONTROLADOR?.....	31
1.8. COMPILACIÓN Y CARGA DE UN PROGRAMA.....	31
1.9. DOCUMENTACIÓN ADJUNTA.....	31
<b>CAPITULO II. SOFTWARE DE LA APLICACIÓN DE PC.....</b>	<b>32</b>
2.1. INTRODUCCIÓN.....	32
2.2. OBJETIVOS.....	32
2.3. REQUISITOS MÍNIMOS.....	33
2.3.1. <i>HARDWARE</i> .....	33
2.3.2. <i>SOFTWARE</i> .....	33
2.4. INSTALACIÓN.....	33
2.5. DESINSTALACIÓN.....	33
2.6. CARACTERÍSTICAS Y MANEJO DEL MONITOR.....	34
2.6.1. <i>USO</i> .....	34
2.6.2. <i>PLANTILLAS</i> .....	35
2.6.3. <i>CONFIGURANDO LA COMUNICACIÓN</i> .....	40
2.6.4. <i>CONFIGURANDO LA REPRESENTACIÓN GRÁFICA</i> .....	41
2.6.5. <i>EXPORTANDO DATOS A MATLAB</i> .....	43
2.7. FUNCIONAMIENTO DE LOS ELEMENTOS PRINCIPALES.....	43
2.7.1. <i>INTERFAZ GRÁFICA</i> .....	43
2.7.2. <i>MÓDULO DE COMUNICACIONES</i> .....	43
2.7.3. <i>PARSER XML</i> .....	45
2.7.4. <i>INTERFAZ MATLAB</i> .....	46

## CAPITULO I. SOFTWARE DEL MICROCONTROLADOR.

### 1.1. INTRODUCCIÓN

El software del microcontrolador ha sido escrito bajo TinyOS que es un sistema operativo para sistemas embebidos. TinyOS, además tiene la ventaja de ser “open source” y tiene una arquitectura basada en componentes que permite una rápida implementación e innovación mientras que se minimiza el tamaño del código, lo cual es muy importante debido a la gran restricción de cantidad de memoria que existe en los sistemas embebidos. El sistema operativo TinyOS está escrito en NesC, un nuevo lenguaje de programación para programación de aplicaciones estructuradas basado en componentes, cuyas características se explicarán más adelante

Por otra parte la ejecución del sistema está dirigida por eventos, lo cual permite un control fino del consumo energético y una flexibilidad en la ejecución de tareas que es necesaria debido a la naturaleza impredecible de las comunicaciones inalámbricas y las interfaces con el mundo real.

### 1.2. ARQUITECTURA DE TINYOS

Los objetivos del sistema operativo TinyOS son:

- Soportar sistemas embebidos en red, lo cual implica:
  - Poder tener al sistema dormido pero vigilante a estímulos.
  - El ciclo del sistema debe ser: estar dormido lo máximo posible, despertar cuando haya algún estímulo exterior, realizar los cálculos y tareas necesarios y volver a dormir lo más rápido posible.
  - Ser fácilmente escalable y compatible con un gran número de nodos.
- Soporte de los avances tecnológicos, lo cual implica cada vez más pequeño, barato y que consuma menos.

De estos objetivos se concluye que el diseño debe estar basado en un núcleo multi-hilo que sea extremadamente simple y eficiente. Este núcleo tiene 2 tipos estructuras:

- Eventos: pueden interrumpir a las tareas, se ejecutan cuando se produce un evento hardware y deben realizar pequeñas cantidades de cálculos.
- Tareas: no son temporalmente críticas, suelen utilizar mucho mayor tiempo de cálculo que los eventos, además las tareas no se interrumpen entre ellas por lo que la siguiente tarea sólo se ejecuta cuando termina la anterior (esto simplifica el núcleo del sistema al sólo necesitar una única pila).

### 1.3. ENTORNO DE DESARROLLO DE TINYOS

En principio, el entorno de desarrollo se tiene que utilizar bajo Linux y se necesita una serie de paquetes para su utilización entre los que destacan:

- La máquina virtual de Java: para la ejecución de ciertas aplicaciones gráficas como el simulador TinyViz, que se comentará más adelante.
- Avr-gcc: compilador cruzado que compila de C a lenguaje ensamblador para los microprocesadores de 8 bits de Atmel.
- Compilador para Nesc: compilador que pasa del lenguaje NesC a C.

Sin embargo también se puede instalar todo este “entorno de desarrollo” bajo Windows utilizando el emulador Cygwin.

Una vez instalado este entorno, nos damos cuenta que en realidad el sistema operativo TinyOS no es más que un conjunto de ficheros que implementan una serie de funcionalidades que nos abstraen del bajo nivel del microcontrolador a la hora de desarrollar una aplicación. Estos ficheros están ordenados por directorios, las cuales se describen a continuación:

- apps: directorio donde se guardan todas las aplicaciones que trae por defecto el entorno y que pueden servir para entrar en contacto con el sistema. Además debería ser aquí donde guardaríamos nuestras propias aplicaciones.
- doc: directorio de documentación donde se incluye un tutorial introductorio a TinyOS y una serie de ficheros sobre distintos temas. Además el entorno de desarrollo trae una

herramienta de creación automática de documentación para las aplicaciones y será en este directorio donde se guarde esta documentación.

- tools: directorio donde están distintas herramientas de desarrollo como el simulador TinyViz, programas para la comunicación entre el PC y los nodos de la red de sensores, programas de monitorización de la red, etc.
- tos: directorio donde se implementa el núcleo del sistema operativo. Está dividido, a su vez, en diferente directorios:
  - interfaces: en este directorio se guardan los interfaces. Los interfaces definen una especie de API que es implementada por un módulo y que puede ser usada por cualquier aplicación (para más información ver el apartado sobre NesC)
  - lib: directorio donde aparecen librerías listas para ser usadas por cualquier aplicación donde se implementan contadores, colas, protocolos de cifrado (TinySec), sistema embebido de bases de datos (TinyDB).
  - platform: en este directorio aparece la implementación a bajo nivel que es dependiente de la plataforma que se utilice y sobre todo del microcontrolador que se vaya a utilizar.
  - sensorboards: implementación a bajo nivel de las placas de expansión con sensores que existen para las diferentes placas.
  - system: en este directorio está el código del núcleo del sistema y donde se implementan las funciones esenciales para su funcionamiento.
  - types: en este directorio aparecen una serie de ficheros donde se implementan los distintos tipos que se utilizan en este sistema.

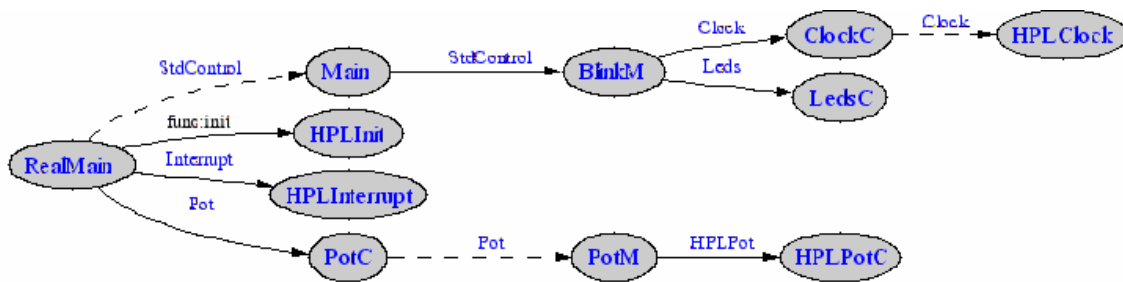


Figura 21: Ejemplo de estructura de una aplicación creado automáticamente por la herramienta de documentación

#### 1.4. EL LENGUAJE DE PROGRAMACIÓN NES C.

Como se dijo anteriormente el código de este sistema operativo está escrito en el lenguaje NesC, este lenguaje soporta el diseño de TinyOS, formado por eventos y tareas. En la siguiente figura se puede observar los pasos de compilación desde la aplicación en NesC hasta el ejecutable.

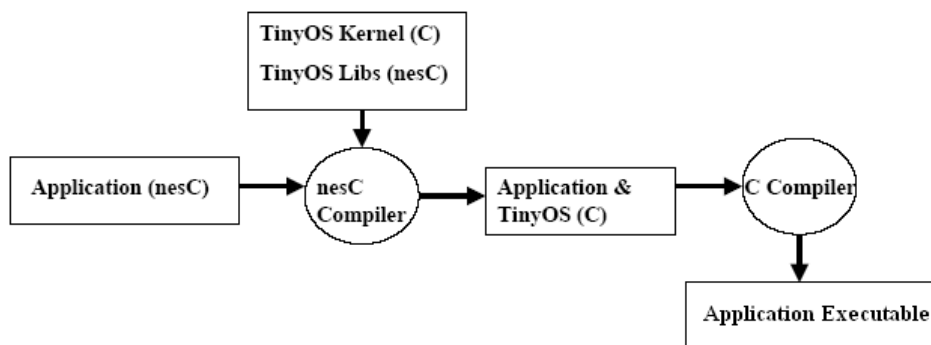


Figura 22: Pasos de compilación de una aplicación en NesC

Como se puede observar realmente el compilador NesC es un parse que traduce del lenguaje NesC a C y después es el código en C el que ya se compila para la máquina donde queremos ejecutar el programa.

Por otra parte este lenguaje tiene una serie de características, las cuales son:

- Todo es estático, es decir, que no hay reserva dinámica de memoria y que no hay funciones a punteros.
- En la compilación realiza un análisis de posibles condiciones de carrera.
- Todas las funciones se compilan "inline".

#### 1.4.1. MODELO DE PROGRAMACIÓN.

En primer lugar comentar que en el código escrito en NesC se organiza en 3 tipos diferentes de ficheros:

- Interfaces:
  - Especifican la funcionalidad a quien lo vaya a utilizar.
  - Especifican los comandos que se pueden llamar y los eventos que deben ser implementados por quien utilice la interfaz.
- Módulo: es donde se implementa la interfaz, es decir, es donde se escribe el código de los comandos asociados a la interfaz.
- Configuración: es donde se especifican la unión de componentes, es decir, dice que interfaces se van a utilizar, por quien se va a utilizar y quien las implementa.

Por otra parte en NesC existe una separación entre la construcción y la composición, es decir, los programas están formados por componentes (pueden ser módulos y configuraciones) y cada componente está especificado por una interfaz y no sólo eso sino que los componentes se pueden conectar unos con otros (esta conexión se realiza estáticamente para incrementar la eficiencia).

Los componentes usan y proporcionan interfaces, comandos y eventos (estos últimos especificados por la interfaz del componente). Hay que tener en cuenta que en TinyOS la palabra interfaz tiene 2 sentidos, es decir, que los componentes implementan los eventos que usan y los comandos que proporcionan. Este hecho queda reflejado en la siguiente tabla:

<i>Componente</i>	<i>Comandos</i>	<i>Eventos</i>
<b>Usa</b>	Pueden llamarlos	Deben implementarlos
<b>Proporciona</b>	Deben implementarlos	Pueden señalarlos para que se ejecuten

Como se dijo anteriormente existen 2 tipos de componentes:

- Módulos: implementan el código de la aplicación, es decir, su comportamiento.
- Configuraciones: realiza las conexiones entre los componentes.

Además un componente no se preocupa si el otro componente al que se conecta es un módulo o una configuración y no sólo eso sino que un componente puede estar formado por otros componentes.

Los componentes se conectan de forma jerárquica, conectando usuarios con proveedores, como se puede observar en la siguiente figura.

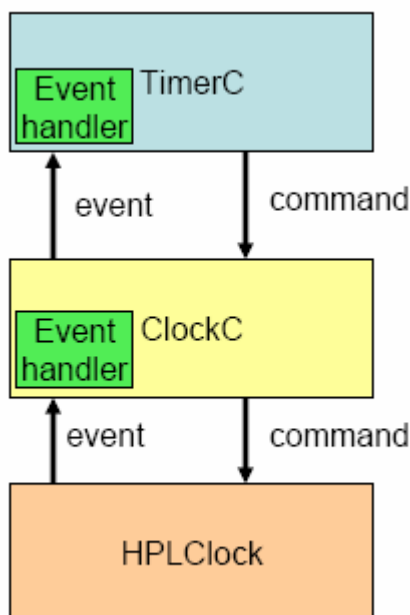


Figura 23: Estructura jerárquica de la unión de componentes

Finalmente decir que el sentido de los comandos es de la capa superior a la inferior y el control vuelve al que llama al comando, mientras que con los eventos es completamente al contrario.

### 1.5. ESTRUCTURA DEL PROGRAMA DESARROLLADO.

Para el desarrollo de la aplicación se han tenido que programar diversas interfaces para el control de los distintos dispositivos, estas interfaces son:

- **Encoder:** función que se ejecuta cuando llega una pulso del encoder (interrupción hardware por flanco de subida).
- **Puerto serie:** funciones para transmitir un byte por el puerto serie y funciones que se ejecutan tanto cuando llega un dato por el puerto serie como cuando se termina de transmitir un byte. Se han utilizado los 2 puertos serie:
  - Para la comunicación con el PC: 19200bps, 8 bits, 1 bit de stop, sin paridad ni control de flujo.
  - Para la comunicación con la controladora de los motores: 9600bps, 7 bits, 1 bit de stop, paridad par y sin control de flujo.
- **Convertidor A/D:** función para obtener el dato digitalizado a partir de una señal analógica. Este convertidor tiene como referencia una fuente interna de tensión de 4,97V por lo que las señales analógicas que se utilicen deben estar entre 0V y 4,97V. Por último, comentar que esta interfaz se ha utilizado para leer los datos del inclinómetro y las referencias de dirección dadas por el puño.
- **Temporizador:** interfaz para controlar la temporización del programa. Esta interfaz se utiliza para crear el reloj de tiempo real a 10ms que es la base del ciclo de control y donde se implementan los distintos controladores.

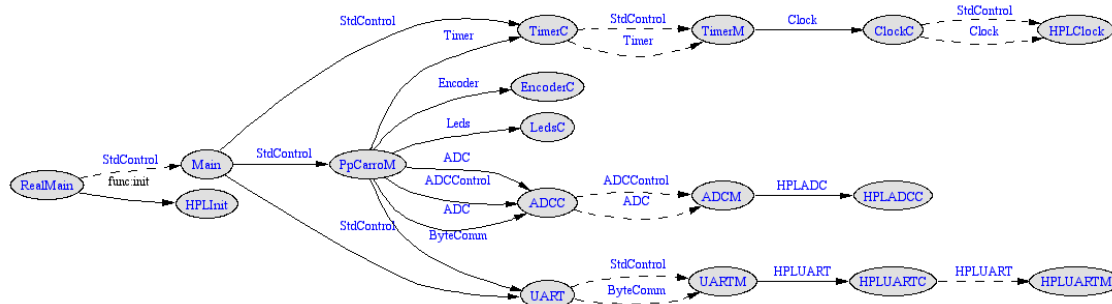


Figura 24: Módulos del programa del microcontrolador

En el programa existen 2 bloques importantes:

- **Función de lectura de datos por el puerto serie comunicado al PC (ByteComm.rxByteReady2PC):** está función es la que gestiona la comunicación con el programa monitor ubicado en el PC. Por lo tanto está función cambia el estado del microcontrolador, los cuales son: parado (para el ciclo de control y los motores), funcionando (ejecuta el controlador seleccionado) y parada de emergencia (para el controlador, los motores y activa la parada de emergencia de la controladora de los motores). Además esta función carga los valores de los distintos controladores y envía los datos al PC cuando recibe una petición. Por último esta función implementa un modo debug para utilizar con el hyperterminal muy útil para depurar. Para ello se configura el programa hyperterminal a 19200bps y se pulsa la tecla 'd', entonces a continuación aparecerá el valor de las variables que se hayan programado para esta opción.
- **Ciclo de control (RTTimer.fired):** cuando el microcontrolador está funcionando se ejecuta un ciclo de control de 10ms. Es aquí donde se implementan los controladores y funciona de la siguiente manera:
  - En primer lugar se leen los valores de los encoders y se calculan las velocidades de las ruedas.
  - En segundo lugar se actualizan las variables de control: inclinación, variación de la inclinación y velocidad lineal del vehículo.
  - Se ejecuta el control y se realizan los cambios de unidades para calcular la velocidad a aplicar a cada rueda.
  - Finalmente se aplica las nuevas velocidades a cada rueda y por último se pide que se capturen nuevos datos de los sensores de inclinación y del puño que da la referencia de inclinación para ser usados en el siguiente ciclo de control.

## 1.6. INSTALACIÓN DEL ENTORNO DE DESARROLLO.

Para instalar el entorno de desarrollo hay que ejecutar el instalable que viene en el directorio /Software/Microcontrolador/Entorno de Desarrollo del CD adjunto a la memoria (instalación completa). A continuación, reemplazar el directorio /tinyos-1.x que está (en el disco donde esté instalado el sistema operativo) en el directorio /tinyos/cygwin/opt por el código del microcontrolador que se encuentra en el directorio /Software/Microcontrolador/Codigo en el CD adjunto a esta memoria. Una vez instalado es importante reconocer:

- Ejecutable del shell del programa Cygwin (posiblemente se cree un acceso directo en el escritorio del ordenador).
- Directorio de trabajo que será (dentro del disco duro donde esté instalado el sistema operativo): /tinyos/cygwin/opt/tinyos-1.x/apps/PepeCarro.

### **1.7. ¿CÓMO SE INTRODUCE UN NUEVO CONTROLADOR?**

En primer lugar recordar que cada controlador debe tener su identificador, actualmente hay 2 utilizados el 1 y el 2, por lo tanto los nuevos se deben numerar del 3 para adelante. Además hay que decir que al introducir un nuevo controlador se debe especificar: el controlador en sí, los parámetros que se van a cargar desde el PC y los datos que se quieren mandar al PC cuando éste se lo especifique. Por lo tanto, para introducir un nuevo controlador hay que seguir los siguientes pasos:

- Ir a la función que implementa el ciclo de control (RTTimer.fired) e ir a un switch donde se elige el controlador en función de su código.
- Crear un case nuevo e implementar el controlador.
- Ir a la función LoadData y crear un case nuevo donde se especifiquen el número y el tipo de datos que se van a cargar desde el PC. Además hay que cambiar el valor de las variables num\_var\_int\_controller\_load y num\_var\_double\_controller\_load (esto se debe hacer al final de la función que recibe los datos del PC, *ByteComm.rxByteReady2PC*).
- Finalmente ir a la función SendData y crear un nuevo case donde se especifiquen el número y el tipo de datos a mandar hacia el PC. Además hay que cambiar el valor de las variables num\_var\_int\_controller\_send y num\_var\_double\_controller\_send (esto se debe hacer dentro del case SEND\_LOG en la función que recibe datos del PC, *ByteComm.rxByteReady2PC*).

### **1.8. COMPILACIÓN Y CARGA DE UN PROGRAMA.**

Para la compilación y carga del nuevo programa hay que seguir los siguientes pasos:

- Quitar la alimentación al microcontrolador y desconectar los dos cables (el que está conectado al enlace inalámbrico y el que está conectado a la controladora de motores).
- Conectar un cable serie del PC (COM1) al conector DB9 hembra del microcontrolador y después alimentar el microcontrolador de nuevo.
- Ejecutar el shell del programa Cygwin e ir al directorio de trabajo (opt/tinyos-1.x/apps/PepeCarro).
- Ejecutar el archivo por lotes ./progPPCarro y comprobar que no se produce ningún error de compilación porque sino se cargará el programa anterior y no el actual.
- Finalmente desconectar la alimentación y el cable serie que une al PC y conectar de nuevo tanto la alimentación como los conectores del enlace inalámbrico y de la controladora de motores.

### **1.9. DOCUMENTACIÓN ADJUNTA.**

En el CD adjunto en el directorio /Software/Microcontrolador/Documentación se encuentra una documentación completa del programa explicado anteriormente.

## CAPITULO II. SOFTWARE DE LA APLICACIÓN DE PC.

### 2.1. INTRODUCCIÓN.

Para la realización de experimentos útiles, es necesario tener un medio fiable para monitorizar y almacenar los datos generados por los mismos. En el caso de una plataforma móvil, hay básicamente dos opciones, con sus respectivas ventajas e inconvenientes: una primera, es el uso de una unidad de almacenamiento montada sobre el mismo vehículo, y la otra es que se ubique en un emplazamiento estático.

El primer planteamiento, a nivel de comunicaciones es más sencillo, puesto que no hay, en principio, restricciones de tiempo en el proceso de descarga de la información, puesto que se puede hacer *fuera de línea*. El inconveniente principal de dicho planteamiento es la necesidad de desarrollar y probar elementos adicionales para el almacenamiento de información. Además, de todas formas hay que poner en marcha un sistema de descarga de datos. Por otra parte, impide la monitorización en tiempo real de la plataforma, empeorando así la capacidad para detectar problemas en el funcionamiento, que podrían afectar a la seguridad del *piloto de pruebas*.

El segundo planteamiento, algo más complejo en principio, permite desarrollar una arquitectura mucho más versátil, puesto que la unidad externa de monitorización puede ser un PC o un portátil, añadiendo, de entrada, todas las características de funcionamiento que estos dispositivos son capaces de soportar. La monitorización en tiempo real, así como la configuración del *software* de control ejecutado sobre la plataforma, permiten un más fácil desarrollo de los experimentos a realizar. Por tanto, éste es el planteamiento escogido.

### 2.2. OBJETIVOS.

Los objetivos marcados para la aplicación para PC son los siguientes:

- Monitorización, en tiempo real, tanto de las medidas de los sensores, como de las señales de control, además de las variables intermedias del algoritmo de control que se esté utilizando.
- Gran capacidad de representación de la información, haciendo uso de métodos analógicos, como agujas y barras, como digitales, con *displays de siete segmentos*.
- Puesto que se necesita visualizar la evolución temporal de las señales, es necesaria una forma adicional de representación, como por ejemplo una pantalla con el formato de un osciloscopio digital.
- Capacidad para exportar a Matlab las muestras recibidas, para su análisis posterior.
- Capacidad para cambiar el estado de funcionamiento de la plataforma. Los estados necesarios son:
  1. Parada.
  2. Carga de controlador.
  3. Arranque de controlador.
  4. Parada de emergencia.
- Comunicación inalámbrica con el vehículo, ya sea WLAN o *Bluetooth*.
- Manejo sencillo de los parámetros de los controladores que se quieran utilizar, pudiendo modificarlos y cargarlos en la plataforma de forma cómoda, por medio de un sistema de asistentes basados en ventana. Toda esta información se deberá almacenar de forma coherente, ya sea en una base de datos o por medio de plantillas, por ejemplo con formato XML.

### **2.3. REQUISITOS MÍNIMOS.**

Para un correcto funcionamiento del *software* de monitorización desarrollado, se recomiendan las características mínimas siguientes:

#### **2.3.1. HARDWARE.**

- CPU 950 MHz o superior.
- 256 MB de memoria RAM.
- Puerto serie libre, o puerto USB libre (necesario adaptador a puerto serie).
- Unidad de comunicación inalámbrica por *Bluetooth* con interfaz para puerto serie, configurada a 19200 bps.

#### **2.3.2. SOFTWARE.**

- Sistema Operativo Windows 2000 o Windows XP.

### **2.4. INSTALACIÓN.**

El CD adjunto contiene el programa de instalación SETUP.EXE. Antes de ejecutarlo asegúrese de desactivar su *software* antivirus y espía, puesto que algunos de éstos pueden dar falsos positivos con el programa de instalación. Cierre las demás aplicaciones antes de comenzar.

Ejecute el programa de instalación, y siga las instrucciones que se muestran en pantalla. Una vez finalizado el proceso de instalación, tendrá una nueva entrada en su lista de programas (en el botón de Inicio, Programas), con el icono de la aplicación:



### **2.5. DESINSTALACIÓN.**

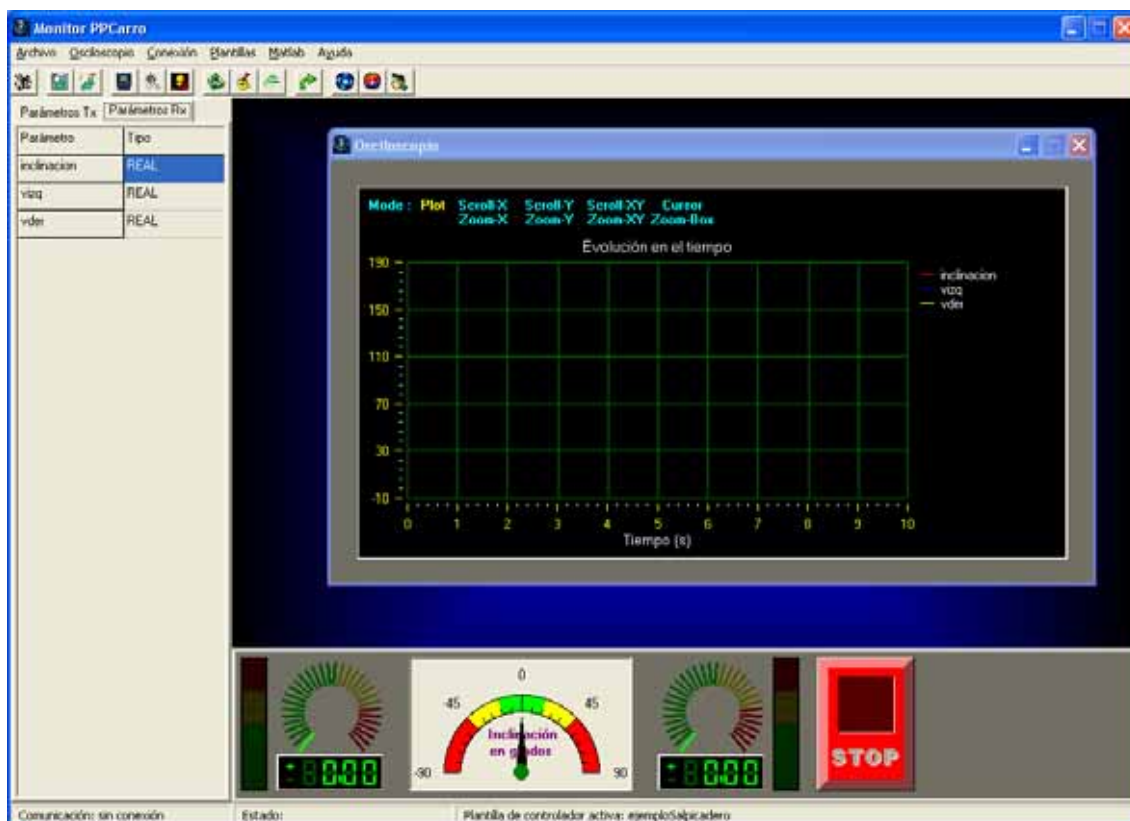
Desde el panel de control de Windows, pinche en *Agregar o quitar programas*, y seleccione la aplicación de monitorización. Pinche en quitar, y siga los pasos que se muestran en pantalla. Si desea conservar las plantillas XML de controladores generadas, así como las muestras para Matlab, cópielas antes en otra ubicación distinta a la que utiliza el monitor.

## 2.6. CARACTERÍSTICAS Y MANEJO DEL MONITOR.

### 2.6.1. USO.

El proceso normal de uso será:

1. Crear o cargar una plantilla de controlador de disco.
2. Configurar la conexión con el PPCarro, seleccionando el puerto al que se tiene conectado.
3. Poner al PPCarro en modo PARADA.
4. Cargar el controlador en el PPCarro.
5. Configurar la adquisición de datos.
6. Comenzar la captura de datos.
7. Activar el controlador en el PPCarro.
8. Finalizar la captura de datos y guardar las muestras a fichero.
9. Poner al PPCarro en modo PARADA.



## 2.6.2. PLANTILLAS.

Las plantillas se almacenan en disco en ficheros \*.pla, utilizando XML para estructurar la información. Las mismas constan de 3 partes diferenciadas:

### 1. *Cabecera*

Contiene el nombre de la misma, así como el identificador numérico del controlador que se quiere utilizar (será único para cada controlador, puesto que lo utilizará el PPCarro para identificar el algoritmo a utilizar, y podrá valer de 1 a 255).

### 2. *Parámetros de Transmisión*

Podrá contener 0 o más parámetros, cuyos campos obligatorios son:

- a. **Nombre.** Nombre del parámetro.
- b. **Tipo.** ENTERO o REAL.
- c. **Valor.** Valor del parámetro.

Además podrá tener 2 parámetros opcionales, que condicionarán el valor posible del parámetro:

- d. **Valor Mínimo.**
- e. **Valor Máximo.**
- f. **Nombre.** Nombre del parámetro.

### 3. *Parámetros de Recepción*

Podrá contener 0 o más parámetros, cuyos campos obligatorios son:

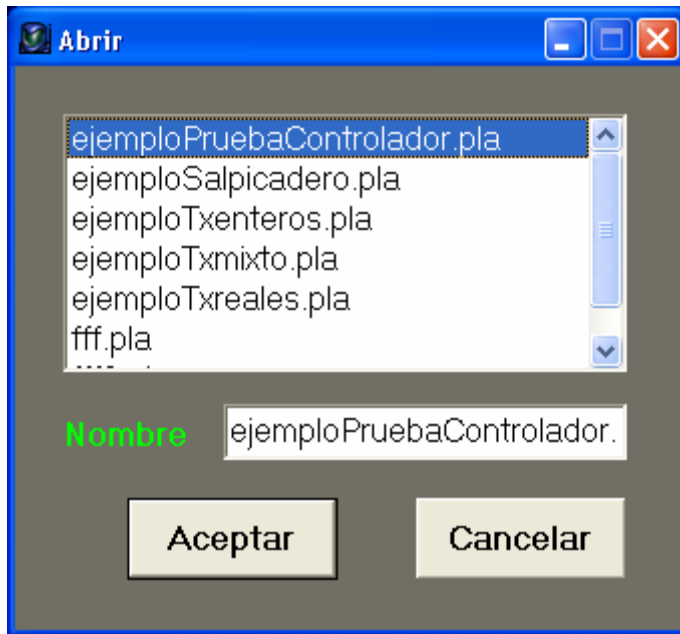
- a. **Nombre.** Nombre del parámetro.
- b. **Tipo.** ENTERO o REAL.

**NOTA:** Para que el salpicadero muestre los cambios en velocidad e inclinación del PPCarro, es necesario que los tres primeros parámetros de recepción sean definidos como reales, con los nombres siguientes:

- 1) inclinacion
- 2) vizq
- 3) vder

**2.6.2.1. CARGA DE PLANTILLA EXISTENTE.**

En el menú <Plantillas>, seleccione <Cargar Plantilla>. Sólo tiene que seleccionarla de una lista, y el sistema la cargará.



### 2.6.2.2. CREACIÓN DE PLANTILLA.

Puesto que no se presupone familiaridad del usuario con XML, se han desarrollado un conjunto de asistentes que permiten, de forma intuitiva, crear y modificar una plantilla, de forma visual.

Para crear una plantilla, en el menú <Plantillas>, seleccione <Crear Plantilla>. Se mostrará un asistente en el que habrá que introducir, de forma obligatoria, el nombre y el identificador, en los cuadros de texto correspondientes. Además, pulsando sobre las tablas de parámetros de transmisión y recepción se pueden añadir, quitar y modificar los parámetros que se quieran (todo ello por medio de sendos asistentes).

Una vez hecho esto, pulse <Guardar>, y se mostrará el diálogo correspondiente. Introduzca el nombre con el que quiere que se guarde el fichero, y seguidamente pulse <Aceptar>.

The screenshot shows a Windows-style dialog box titled "Crear Plantilla". It features a blue title bar with standard window controls. The main area is divided into several sections:

- Parámetros en Transmisión:** A table with five columns: "Nombre", "Tipo", "Valor", "Mínimo", and "Máximo". The first row is highlighted in blue.
- Parámetros en Recepción:** A smaller table with two columns: "Nombre" and "Tipo". The first row is highlighted in blue.
- Input Fields:** Two text boxes labeled "Nombre" and "Identificador" in green text.
- Buttons:** Two buttons labeled "Guardar" and "Cancelar" in black text.

### 2.6.2.3. MODIFICACIÓN DE PLANTILLA CARGADA.

Para modificar una plantilla ya cargada, en el menú <Plantillas>, seleccione <Modificar Plantilla>. Se mostrará un asistente con el mismo funcionamiento del de crear una plantilla, pero con los campos de la plantilla cargada. Sólo tendrá que pulsar con el ratón (botón derecho), y seguir el proceso descrito más arriba.

**Parámetros en Transmisión**

Nombre	Tipo	Valor	Mínimo	Máximo
k1	ENTERO	10		
k2	REAL	1,45		
k3	ENTERO	168		
k4	ENTERO	50		
k5	REAL	15,78		

**Parámetros en Recepción**

Nombre	Tipo
inclinacion	REAL
vizq	REAL
vder	REAL

**Nombre** ejemploPruebaControlador

**Identificador** 1

Guardar Cancelar

A dialog box titled "Nuevo Parámetro en Transmisión" with a blue header bar containing a minimize, maximize, and close button. The dialog has a grey background and contains the following fields:

- Nombre:** A text input field containing "k3".
- Tipo:** A dropdown menu with "ENTERO" selected.
- Valor:** A text input field containing "168".
- Mínimo:** A text input field containing "12".
- Máximo:** A text input field containing "23".

At the bottom, there are two buttons: "Aceptar" and "Cancelar".

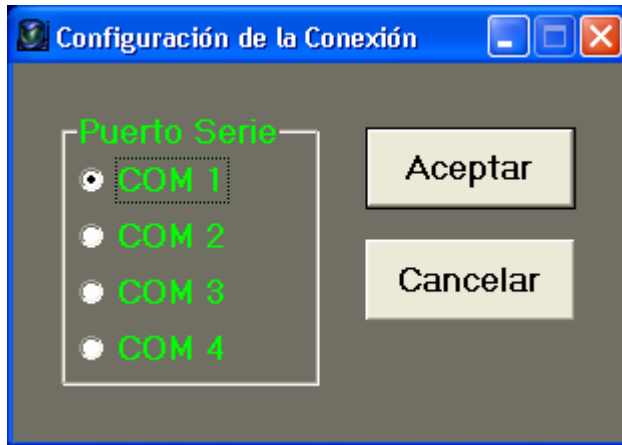
A dialog box titled "Nuevo Parámetro en Recepción" with a blue header bar containing a minimize, maximize, and close button. The dialog has a grey background and contains the following fields:

- Nombre:** A text input field containing "vzq".
- Tipo:** A dropdown menu with "REAL" selected.

At the bottom, there are two buttons: "Aceptar" and "Cancelar".

### 2.6.3. CONFIGURANDO LA COMUNICACIÓN.

Lo único que hay que hacer es, desde el menú <Conexión>, seleccionar <Configurar Conexión>, y seguidamente escoger el puerto serie al que se tiene conectado el PPCarro.

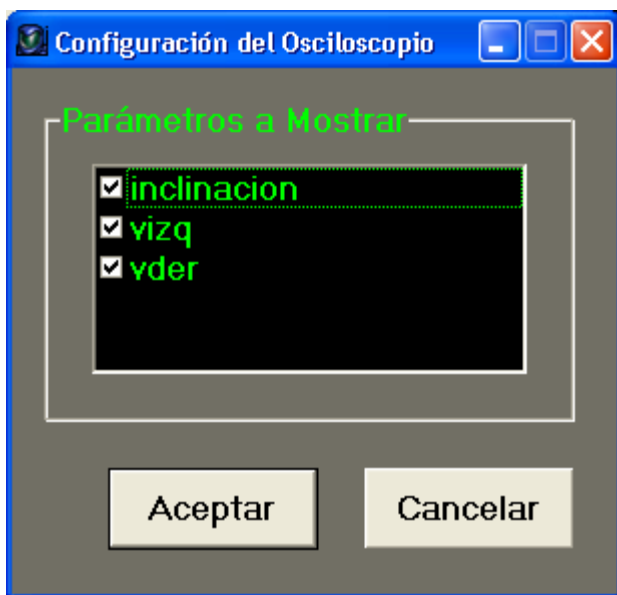


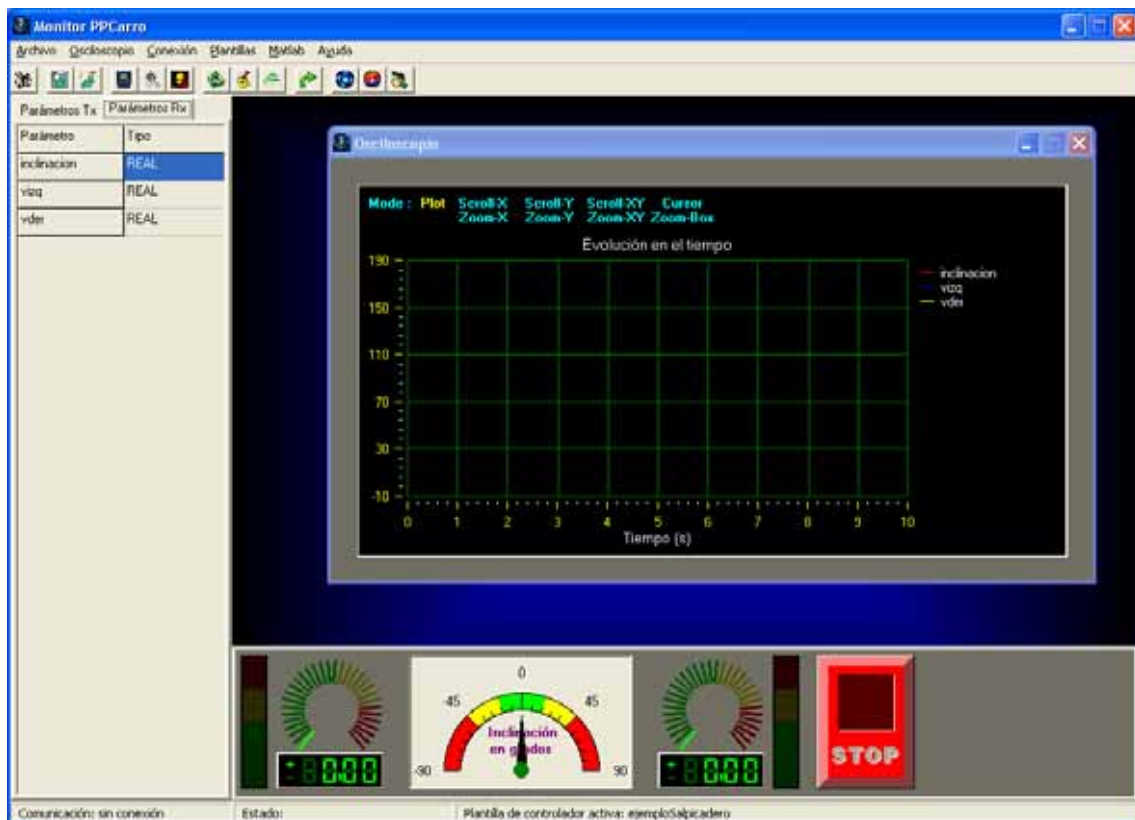
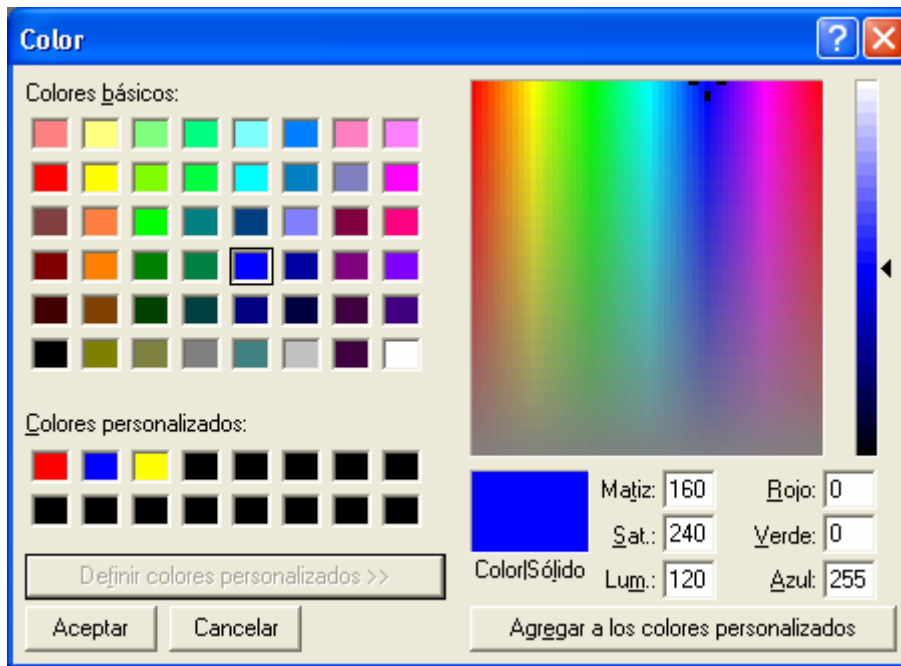
#### 2.6.4. CONFIGURANDO LA REPRESENTACIÓN GRÁFICA.

Además de los indicadores de la parte inferior de la ventana principal de la aplicación, se ha incorporado un osciloscopio interactivo para facilitar la monitorización de señales adicionales. Las señales que aquí se pueden mostrar son las definidas como parámetros de recepción en la plantilla del controlador que se quiere utilizar. Por ello, es necesario tener cargada una plantilla antes de poder utilizar esta utilidad.

Una vez que se tenga cargada una plantilla, se puede configurar cuáles de los parámetros de recepción se quieren visualizar, así como el color de su trazada. Para ello, se accede al asistente de configuración desde el menú <Osciloscopio>, <Configuración del Osciloscopio>. En la lista de parámetros que aparecen, se marcan los que se quieren mostrar. Además, para seleccionar el color de representación de cada uno de ellos, pulse, con el botón derecho, sobre el que se quiera modificar, y entre en la opción <Color...>. Desde aquí se puede seleccionar un color de la lista, o personalizarlo.

**NOTA:** Para comenzar la representación gráfica, tanto sobre el Osciloscopio como sobre el salpicadero, es necesario indicarlo por medio del menú <Matlab>, <Comenzar Captura>.





### 2.6.5. EXPORTANDO DATOS A MATLAB.

Primero habrá que comenzar la captura, como se ha indicado en el apartado anterior. Una vez se tengan los datos que se quieran, se para la adquisición de datos, por medio del menú <Matlab>, <Finalizar Captura>. Una vez hecho esto, se guarda la información por medio del menú <Matlab>, <Guardar Muestras>.

### 2.7. FUNCIONAMIENTO DE LOS ELEMENTOS PRINCIPALES.

En este apartado se va a realizar una breve descripción del funcionamiento interno de los elementos principales del *Monitor*. Para una información detallada se remite al lector a los anexos, en donde se incluye una descripción del código, tanto en formato de documentación impresa (150 páginas), como en HTML (CD adjunto). Éstos anexos contienen el *Manual del Programador*, que será necesario consultar en el caso en que se quiera realizar algún cambio o ampliación del programa.

Las partes principales del monitor son:

1. Interfaz gráfica.
2. Módulo de comunicaciones.
3. Parser XML.
4. Interfaz Matlab.

#### 2.7.1. INTERFAZ GRÁFICA.

Puesto que la implementación de esta parte depende exclusivamente del sistema operativo utilizado y la herramienta de desarrollo (Builder v5.0), se remite al lector directamente a los anexos, donde se incluye, en el manual del programador, información detallada sobre los componentes utilizados.

#### 2.7.2. MÓDULO DE COMUNICACIONES.

Se plantearon varios objetivos a conseguir con la comunicación:

1. Permitir el uso de directivas de alto nivel para indicar a la plataforma móvil su comportamiento.
2. Monitorizar el estado del sistema, incluyendo sensores, actuadores y variables internas, en tiempo real.

Para ello se ha desarrollado un protocolo a nivel de enlace, orientado a conexión, y con acuse de recibo, que a continuación se describirá.

El formato de trama genérico es el siguiente:



Donde la cabecera es un octeto, que representa, de forma única, el tipo de PDU (*Protocol Data Unit*) que se está enviando, y el campo de datos contiene los datos, que se interpretan a partir de la cabecera.

Toda comunicación se comienza desde el PC, y siempre es necesario algún tipo de asentimiento por parte del microcontrolador para que se considere exitosa la misma. Si no es así, se lanza la correspondiente excepción, y el usuario del monitor es informado del problema, para que pueda solventarlo.

Seguidamente se pasan a describir las diferentes PDUs, y su implicación en el proceso de comunicación.

#### 2.7.2.1. Parada

La comunicación comienza desde el PC, enviando la PDU:

**PARADA**

que no contiene campo de datos, sino sólo cabecera. Indica al microcontrolador que pasa a estado *parada*, que es la situación en que los motores están desbloqueados, y se puede transferir con seguridad información sobre el controlador que se quiere utilizar.

La respuesta del microcontrolador deberá ser otra PDU:

**PARADA**

indicando así que ha recibido el mensaje y ha pasado a estado *parada*.

#### 2.7.2.2. Parada de Emergencia

La comunicación comienza desde el PC, enviando la PDU:

**PARADA\_EMERGENCIA**

que no contiene campo de datos, sino sólo cabecera. Indica al microcontrolador que tiene que desbloquear inmediatamente los motores y pasar a STOP la controladora de potencia de los motores, puesto que se ha producido una situación de emergencia inesperada.

La respuesta del microcontrolador deberá ser otra PDU:

**PARADA\_EMERGENCIA**

indicando así que ha recibido el mensaje y ha pasado a estado *parada de emergencia*.

#### 2.7.2.3. CARGA DE CONTROLADOR.

La comunicación comienza desde el PC, enviando la PDU:

<b>CARGA_CTRL</b>	<b>ID_CTRL</b>	<b>PARAM_1</b>	<b>PARAM_2</b>	<b>...</b>	<b>PARAM_N</b>
-------------------	----------------	----------------	----------------	------------	----------------

Se utiliza para indicar al microcontrolador cuál es el algoritmo de control que se quiere utilizar, para ello se utiliza un identificador único **ID\_CTRL**. Por otra parte, se pasa una secuencia de parámetros que son los que se utilizan para ajustar el controlador. Todo esto estará previamente definido en la plantilla de controlador que se está utilizando, y se modifica a través de los asistentes desarrollados para tal efecto.

Una vez el microcontrolador recibe esta PDU, pasa a tener preparado el algoritmo de control indicado, e inicializa los parámetros internos del mismo con los pasados en la PDU. Si ha recibido los datos correctamente, responderá con un asentimiento positivo:

**CARGA\_CTRL**

#### 2.7.2.4. PETICIÓN DE DATOS.

La comunicación comienza desde el PC, enviando la PDU:

**PETICION\_DATOS**

Ésta indica al microcontrolador que debe responder con una PDU que contenga los datos de los parámetros de recepción definidos en la plantilla del controlador activo en ese momento. Éste es, por tanto, el mecanismo que se utiliza para monitorizar los parámetros que se quieren de la plataforma móvil.

La respuesta del microcontrolador es una PDU con formato:

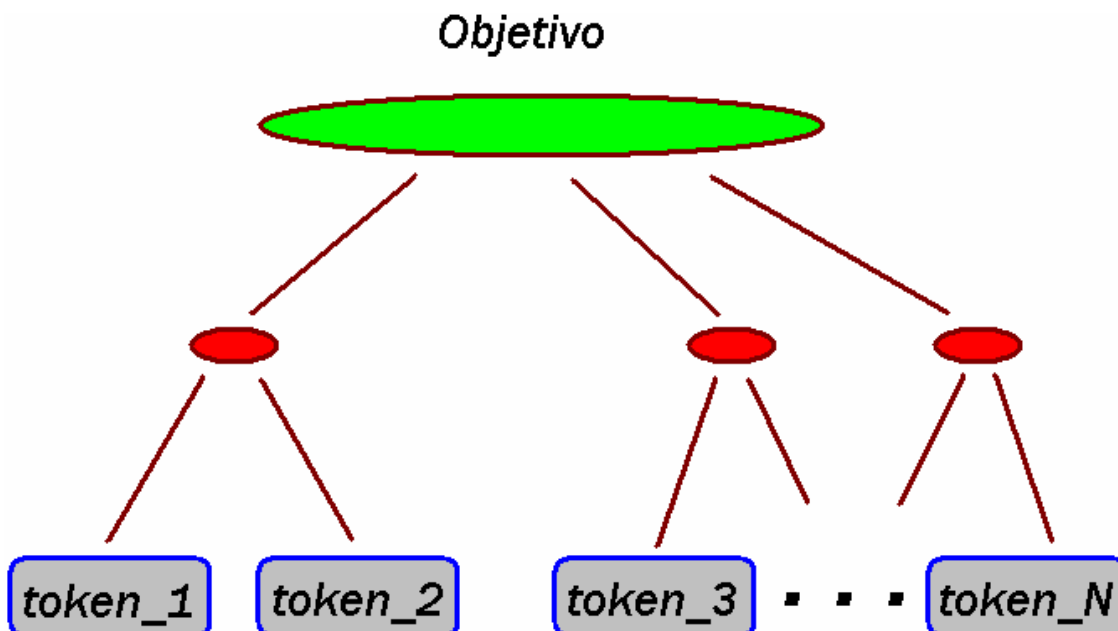
**PETICION\_DATOS    PARAM\_1    PARAM\_2    ...    PARAM\_N**

La secuencia de datos recibida es la indicada por la correspondiente plantilla, como ya se ha indicado más arriba.

#### 2.7.3. PARSER XML.

Como se ha comentado en apartados anteriores, intentando facilitar la reconfiguración y manejo de los parámetros de los controladores implementados, se ha desarrollado un módulo para mantener este tipo de información, basado en plantillas XML. Para ello se ha desarrollado un *parser* de tipo ascendente, que realiza un análisis léxico, sintáctico y semántico del mismo.

La idea subyacente en un *parser* ascendente se muestra en el siguiente diagrama:



El análisis ascendente consiste en, a partir de los *tokens* elementales detectados a nivel léxico, se intenta reconstruir el objetivo, que es, en el caso que se presenta, una plantilla de controlador.

Para ello, se van extrayendo, de izquierda a derecha, los elementos fundamentales del fichero XML, como marcas de inicio y final, contenidos, comentarios, etc. A partir de estos elementos se va realizando un proceso de *desplazamiento/reducción*, con el cual se van generando

elementos no terminales, hasta que sólo se tiene un elemento, que es el objetivo, que deberá ser un elemento de tipo plantilla.

Internamente, el analizador léxico lo único que hace es extraer los elementos terminales, y se los va pasando al analizador sintáctico. El analizador sintáctico a su vez intenta encajar, en las estructuras permitidas, los elementos terminales que va recibiendo. Además, para poder realizar el análisis semántico a la vez, una vez que se consiguen extraer elementos suficientes para que su significado conjunto sea analizable, se van generando los objetos de tipo *parámetro de recepción* o *de transmisión* pertinentes. En este proceso de inicialización, los constructores y métodos de inicialización comprueban a su vez que los parámetros son correctos, y avisan en caso contrario. Así, una vez realizada una primera pasada al fichero ya se sabe si es correcto o no. En el caso en que no sea correcto se lanza el correspondiente aviso, y si es correcto se carga en la aplicación, estando listo para ser enviado al microcontrolador.

Para una descripción detallada de la implementación se remite al lector al *Manual del Programador* incluido en los anexos.

#### **2.7.4. INTERFAZ MATLAB.**

Para poder manejar los resultados de los experimentos de forma cómoda, y con las herramientas habituales (Matlab y Simulink), se ha desarrollado un módulo que exporta a ficheros Matlab todas las muestras capturadas. Para ello, se genera el fichero a partir de los datos recibidos, y se respeta el formato definido para vectores. Además, se añade un vector adicional con marcas de tiempo, para que el proceso sea más sencillo de monitorizar. Para una descripción detallada, se remite al lector al *Manual del Programador* (apéndices).



**Escuela Superior de Ingenieros de Sevilla**

---

**DISEÑO, SINTONIZACIÓN Y SIMULACIÓN DE CONTROLADORES**

---

Alberto Prieto  
Antidio Viguria  
Benito José Vela  
Mirko Fiacchini  
Ramón Cano

Trabajo de los Cursos de Doctorado:

Control No Lineal Aplicado  
Sistema de Control No Lineal  
Robótica Industrial

**ÍNDICE.**

<b>CAPITULO I. DESCRIPCIÓN Y MODELADO DEL SISTEMA.....</b>	<b>49</b>
1.1 DESCRIPCIÓN DEL SISTEMA.....	49
1.2 MODELO DEL SISTEMA.....	49
1.2.1 <i>VEHÍCULO CON TRACCIÓN DIFERENCIAL</i> .....	49
1.2.2 <i>PÉNDULO INVERTIDO SOBRE BASE MÓVIL</i> .....	50
<b>CAPITULO II. CONTROLADORES. DISEÑO Y SIMULACIÓN.....</b>	<b>51</b>
2.1 CONTROL LQR.....	51
2.1.1 <i>UBICACIÓN DE POLOS CON LQR</i> .....	51
2.2 CONTROLADOR NO LINEAL: ASTOLFI-KALIORA.....	53
2.2.1 <i>LINEALIZACIÓN PARCIAL</i> .....	53
2.2.2 <i>ESTABILIZACIÓN DEL PÉNDULO POR MOLDEO DE LA FUNCIÓN DE ENERGÍA</i> .....	54
2.2.3 <i>ESTABILIZACIÓN DE LA VELOCIDAD</i> .....	55
2.2.4 <i>SINTONIZACIÓN DEL CONTROLADOR NO LINEAL</i> .....	55
2.3 SIMULACIONES.....	56
2.4 EFECTO DE LA SATURACIÓN, TIEMPO DE MUESTREO Y RETRASO.....	60

## CAPITULO I. DESCRIPCIÓN Y MODELADO DEL SISTEMA.

### 1.1 DESCRIPCIÓN DEL SISTEMA

El sistema, de forma simplificada, está constituido por una plataforma montada sobre dos ruedas que son accionadas por dos motores independientemente. Sobre esta plataforma se sitúa una masa que puede ser modelada como una masa puntual a una distancia  $l$  del plano de la base.

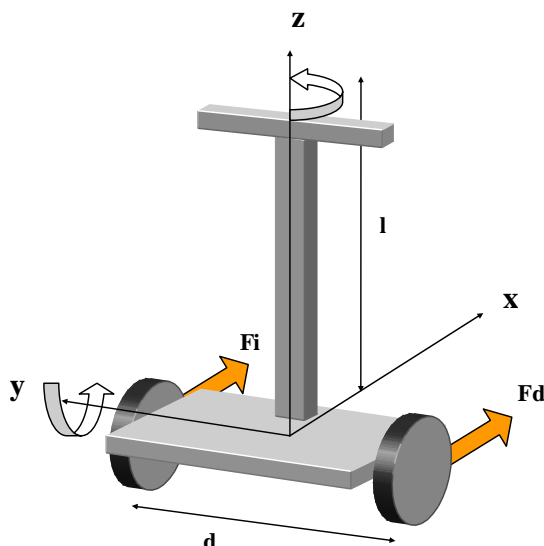


Figura 25: Esquema de la estructura del vehículo

La figura muestra de forma esquemática la estructura del vehículo, los ejes de referencias, los grados de libertad y las entradas del sistema.

### 1.2 MODELO DEL SISTEMA

Desde el punto de vista del control, el sistema puede descomponerse en dos subsistemas que prácticamente están desacoplados entre sí. Una parte está formada por un robot móvil con tracción diferencial, la otra la constituye un péndulo invertido sobre un carro móvil.

Las entradas del sistema las constituyen las fuerzas o pares que ejercen cada una de las ruedas sobre el vehículo. Las variables a controlar son el ángulo que forma el péndulo con la vertical y las velocidades de avance y de rotación del conjunto. A continuación se analizan los dos sistemas independientemente.

#### 1.2.1 VEHÍCULO CON TRACCIÓN DIFERENCIAL

En este sistema la salida la constituye la velocidad de giro respecto al eje vertical. La referencia para esta salida viene dada por una señal procedente del puño de dirección del vehículo. Considerando las fuerzas ejercidas por cada una de las ruedas,  $F_i$  y  $F_d$ , puede obtenerse las ecuaciones del vehículo

$$J\ddot{\delta} = (F_d - F_i)d$$

Donde

$F_i$  y  $F_d$ , representan las fuerzas de la rueda izquierda y derecha respectivamente

J: Momento de inercia respecto al eje vertical

$\ddot{\delta}$ : Aceleración angular alrededor del eje z

d: Distancia entre las ruedas

### 1.2.2 PÉNDULO INVERTIDO SOBRE BASE MÓVIL

En la figura se ha representado el subsistema constituido por el péndulo invertido sobre plataforma móvil.

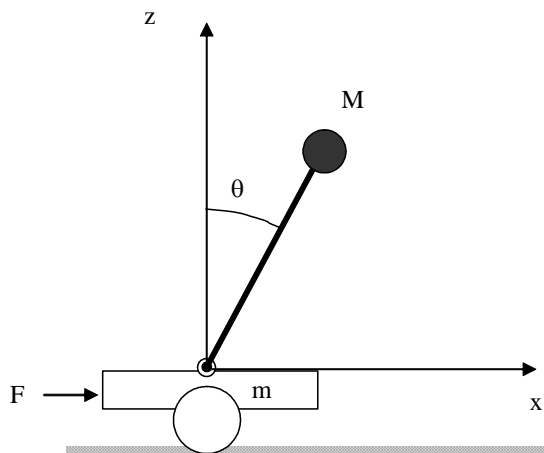


Figura 26: Péndulo invertido sobre carro móvil

El equilibrio de fuerzas en el eje x permite obtener la ecuación

$$(M + m)\ddot{x} + Ml\ddot{\theta}\cos\theta = F$$

Por otra parte el balance de momentos en torno al punto de giro conduce a

$$\ddot{x}Ml\cos\theta + Ml^2\ddot{\theta} - Mgl\sin\theta = 0$$

donde

m: Masa del carrito, 30 Kg

M: Masa del péndulo, 70 Kg

l: Altura del centro de masa, 1m

g: Aceleración de la gravedad 9,8 m/s<sup>2</sup>

A partir de las ecuaciones anteriores pueden obtenerse las ecuaciones en variables de estado del sistema

$$\ddot{\theta} = \frac{1}{l\left(\frac{m}{M} + \sin\theta\right)} \left( -\frac{F}{M}\cos\theta + \left(\frac{M+m}{M}\right)g\sin\theta - l\dot{\theta}^2\sin\theta\cos\theta \right)$$

$$\ddot{x} = \dot{v} = \frac{1}{\left(\frac{m}{M} + \sin^2\theta\right)} \left( l\dot{\theta}^2\sin\theta - g\cos\theta\sin\theta + \frac{F}{M} \right)$$

Ya que el objetivo del control es la estabilización del ángulo  $\theta$  en cero y de la velocidad lineal  $\dot{x}$ , las variables de estado de interés del sistema son solamente el ángulo  $\theta$ , la velocidad angular  $\dot{\theta}$  y la velocidad  $\dot{x} = v$ .

A continuación, se ilustran dos métodos de control. El primero es un LQR (Linear Quadratic Regulator), un control lineal para el que es necesario un modelo del sistema linealizado alrededor del punto de equilibrio. El otro es un control no lineal para el que se puede utilizar directamente las ecuaciones diferenciales no lineales obtenidas anteriormente.

## CAPITULO II. CONTROLADORES. DISEÑO Y SIMULACIÓN.

### 2.1 CONTROL LQR

El controlador LQR es un control por realimentación del vector de estados de la forma

$$u = -Kx$$

tal que el valor de K se obtiene a partir de un problema de minimización del funcional de coste

$$J = \int_0^{\infty} (x'Qx + u'Ru)dt$$

Esta función de coste es de tipo cuadrática tanto en el vector de estado como en la entrada. Las matrices Q y R penalizan respectivamente el error del estado y el esfuerzo de control.

Para poder obtener este controlador es necesario un modelo lineal. El sistema puede ser linealizado en torno al punto de equilibrio  $\theta = 0$ ,  $\dot{\theta} = 0$  y  $v=0$ . En este punto pueden realizarse las aproximaciones  $\cos\theta \approx 1$  y  $\sin\theta \approx \theta$ . Con ello las ecuaciones de estado en forma matricial serán

$$\begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & \frac{(M+m)g}{ml} & 0 \\ 0 & -\frac{M}{m}g & 0 \end{pmatrix} \begin{pmatrix} \theta \\ \dot{\theta} \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ \frac{1}{m} \end{pmatrix} F$$

donde el vector de salidas coincide con el vector de estados.

#### 2.1.1 UBICACIÓN DE POLOS CON LQR

En este apartado se analiza la situación de los polos del sistema en bucle cerrado en función del vector de ganancias K que realimenta el estado.

El sistema en bucle abierto viene dado en ecuaciones de estado por

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}$$

Teniendo en cuenta la expresión del controlador

$$u = -Kx$$

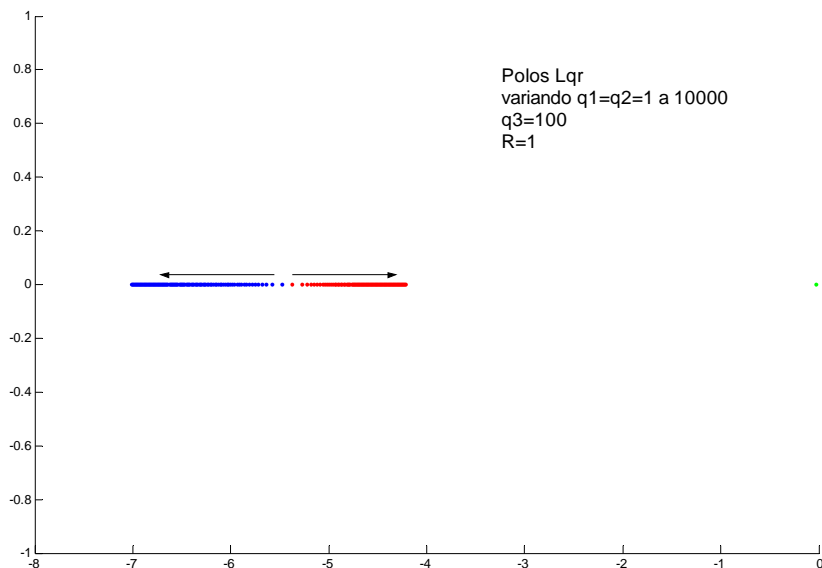
El sistema en bucle cerrado vendrá dado por las ecuaciones

$$\begin{aligned} \dot{x} &= (A - KB)x \\ y &= Cx \end{aligned}$$

Los polos de este sistema se corresponden con los autovalores de la matriz

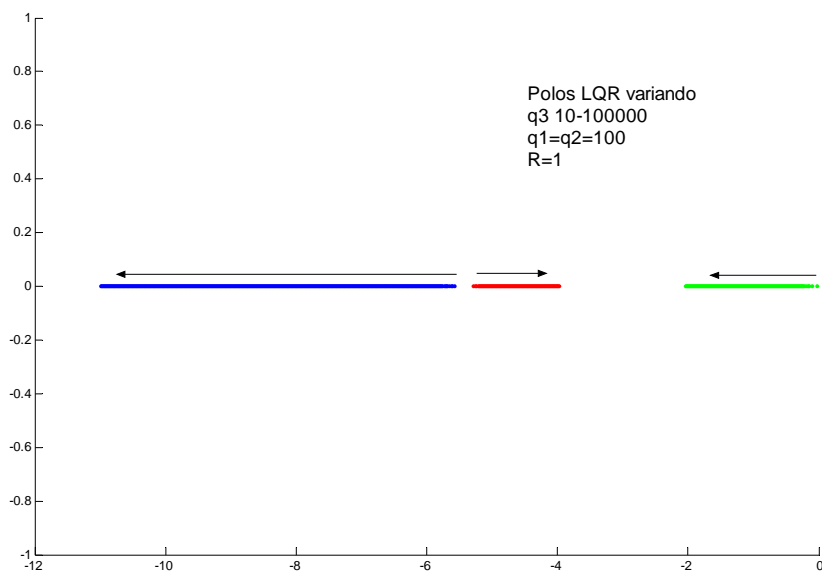
$$(A - KB)$$

y vienen determinados por los parámetros del sistema (matrices A y B) y los valores del controlador (vector K). A su vez, los valores que toma este vector dependen de la asignación que se realice a los distintos elementos de las matrices de penalización Q y R.



**Figura 27: Ubicación de los polos al variar q1 y q2**

En la figura anterior se aprecia que las variaciones de  $q_1$  y  $q_2$  afectan sólo a los polos más rápidos, mientras que el tercer polo permanece cerca del origen. En la figura siguiente se nota como al variar  $q_3$  que penaliza el error en velocidad el polo que estaba cerca del origen se hace más rápido.



**Figura 28: Ubicación de los polos al variar q3**

Finalmente se muestra la influencia del peso sobre la acción de control observándose que para valores cercanos a cero aparecen dos polos complejos conjugados que se convierten en reales cuando R se acerca a la unidad. Para valores de R mayores de uno no se aprecian cambios significativos en la ubicación de los polos.

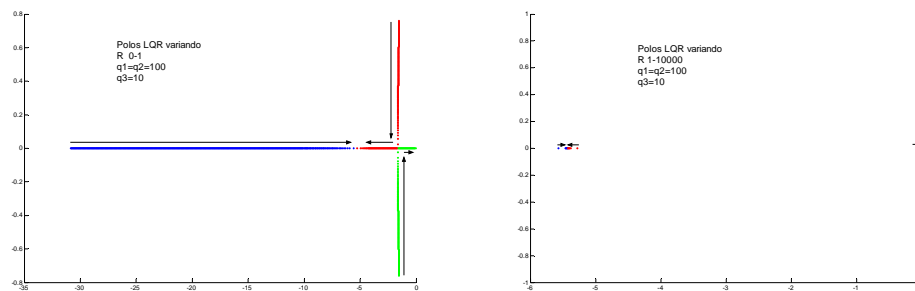


Figura 29: Ubicación de los polos al variar R

## 2.2 CONTROLADOR NO LINEAL: ASTOLFI-KALIORA

Para el diseño de controladores no lineales suele resultar conveniente realizar en primer lugar, una linealización parcial de las ecuaciones del sistema.

### 2.2.1 LINEALIZACIÓN PARCIAL

En las ecuaciones del modelo

$$\begin{aligned} (M + m)\ddot{x} + Ml\ddot{\theta}\cos\theta - Ml\dot{\theta}^2\sin\theta &= F \\ \ddot{x}\cos\theta + l\ddot{\theta} - g\sin\theta &= 0 \end{aligned}$$

si se despeja  $l\ddot{\theta}$  de la segunda y se sustituye en la primera se tiene

$$\begin{aligned} [(M + m) - M\cos^2\theta]\ddot{x} &= F + Ml\dot{\theta}^2\sin\theta - Mg\sin\theta\cos\theta \\ \ddot{x} &= \frac{F + Ml\dot{\theta}^2\sin\theta - Mg\sin\theta\cos\theta}{[(M + m) - M\cos^2\theta]} \end{aligned}$$

Definiendo una nueva variable de control u

$$u = \frac{F + Ml\dot{\theta}^2\sin\theta - Mg\sin\theta\cos\theta}{[(M + m) - M\cos^2\theta]}$$

el sistema queda reducido a

$$\begin{aligned} u\cos\theta + l\ddot{\theta} - g\sin\theta &= 0 \\ \ddot{x} &= u \end{aligned}$$

lo que permite escribir el sistema en variables de estado como

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{l}(g \sin x_1 - u \cos x_1) \\ \dot{x}_3 &= u\end{aligned}$$

donde  $x_1 = \theta$ ,  $x_2 = \dot{\theta}$ ,  $x_3 = \dot{x}$

El sistema presenta una estructura en cascada

$$\begin{aligned}\dot{X}_1 &= f_1(X_1, X_2, U) \\ \dot{X}_2 &= f_2(X_2, U)\end{aligned}$$

donde la parte dinámica relativa al subvector  $X_2$  no depende de los estados contenidos en el subvector  $X_1$ . Un método para tratar este tipo de sistemas es el conocido como *forwarding* (Salas, Gordillo, Aracil 2004) que propone generar una ley de control que establezca el subsistema superior y posteriormente añadir un término adecuado para la estabilización del subsistema inferior. Por tanto, el primer objetivo será el control del péndulo.

### 2.2.2 ESTABILIZACIÓN DEL PÉNDULO POR MOLDEO DE LA FUNCIÓN DE ENERGÍA.

Existen diversas formas de estabilizar el péndulo. Una de ellas consiste en hacer que el sistema con una acción de control adecuada se comporte como un péndulo no invertido cuyo comportamiento es estable. Para encontrar esta acción de control basta con igualar las ecuaciones de un péndulo no invertido

$$\ddot{\theta} = -\frac{g}{l} \sin \theta$$

con las de nuestro sistema

$$\ddot{\theta} = \frac{1}{l}(g \sin \theta - u \cos \theta)$$

lo que permite obtener el valor de  $u$

$$-\frac{g}{l} \sin \theta = \frac{1}{l}(g \sin \theta - u \cos \theta) \Rightarrow u = 2g \tan \theta$$

Con esta acción de control el sistema se comporta como un péndulo no invertido, es decir, estable. Si se desea un comportamiento asintóticamente estable puede añadirse al modelo del péndulo no invertido una componente de amortiguamiento. Este amortiguamiento usualmente se modela como un término proporcional a la velocidad y que se opone al movimiento.

Si se repite el proceso anterior añadiendo el término de amortiguamiento se obtiene una ley de control

$$u = 2g \tan \theta + \frac{K_a}{M} \dot{\theta}$$

que hace al sistema asintóticamente estable.

Es necesario modificar la ley de control obtenida para conseguir que la dinámica de la velocidad también sea estable.

### 2.2.3 ESTABILIZACIÓN DE LA VELOCIDAD

El artículo citado anteriormente propone dos modificaciones diferentes al moldeo de energía para conseguir la estabilidad de la dinámica de la velocidad.

Uno de ellos llega a la ley de control mediante la resolución de una ecuación en derivadas parciales que se obtienen imponiendo algunas condiciones a la función de Lyapunov. Utilizando este método la ley de control resulta compleja.

La otra ley de control es la propuesta por Astolfi-Kaliora que utiliza un término de la forma

$$u = 2gK_m \tan \theta + \frac{K_a}{M} \dot{\theta} + u_d$$

siendo los dos primeros términos los concernientes al moldeo de energía, donde  $K_m$  es una constante que añade un mayor grado de libertad al ajuste del controlador. El término  $u_d$  es el relativo a la estabilización de la velocidad, y viene dado por la expresión

$$u_d = \varepsilon \operatorname{sat} \left( \frac{K_v v}{\varepsilon} \right)$$

que permite ajustar el control mediante la elección adecuada de los valores  $K_v$  y  $\varepsilon$ .

### 2.2.4 SINTONIZACIÓN DEL CONTROLADOR NO LINEAL

El controlador no lineal descrito posee tres parámetros de ajuste ( $K_m$ ,  $K_a$  y  $K_v$ ). Esto le confiere mayor grado de libertad y mejores características, aunque esto lleva implícito una cierta dificultad para su ajuste. De hecho, para determinados valores de estos parámetros, el sistema puede llegar a ser inestable como se deduce de la situación de los polos en bucle cerrado mostrada en las figuras siguientes.

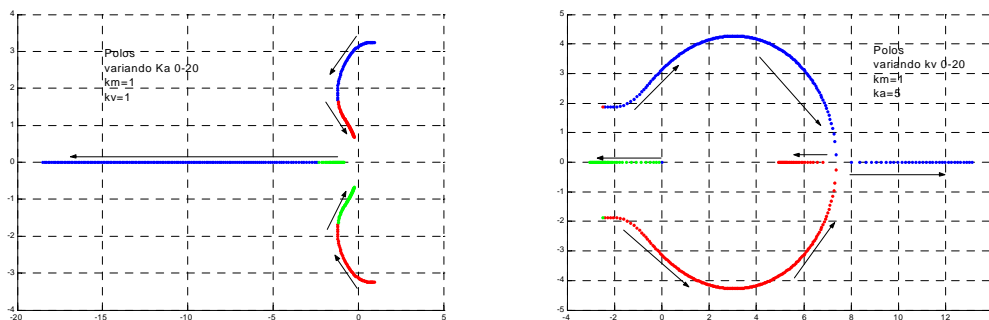


Figura 30: Ubicación de los polos al variar los parámetros del controlador no lineal

El controlador LQR presenta un comportamiento óptimo entorno al punto de equilibrio. Así que, si se determina la relación entre las constantes de este controlador y las del Astolfi-Kaliora, podrán obtenerse unos valores de  $K_m$ ,  $K_a$  y  $K_v$  que proporcionen un comportamiento similar a LQR al menos en el punto de equilibrio.

Deshaciendo la linealización parcial que se realizó como paso previo al diseño del control no lineal se tiene que

$$F = [(M + m) - M \cos^2 \theta]u + Mg \sin \theta \cos \theta - Ml\dot{\theta}^2 \sin \theta$$

donde

$$u = 2gK_m \tan \theta + \frac{K_a}{M} \dot{\theta} + \varepsilon \operatorname{sat}\left(\frac{K_v v}{\varepsilon}\right)$$

Si se linealizan estas ecuaciones en el punto de equilibrio ( $\theta = 0$ ,  $\dot{\theta} = 0$  y  $v=0$ ) se obtiene que

$$F = [Mg + 2gmK_m]\theta + mK_a\dot{\theta} + mK_v v$$

Si se compara esta ecuación con la del LQR

$$F = -K_1\theta - K_2\dot{\theta} - K_3v$$

pueden obtenerse los valores de  $K_m$ ,  $K_a$  y  $K_v$  que proporcionarán buenas características al controlador, esto es

$$K_m = \frac{-K_1 - Mg}{2gm} \quad K_a = \frac{-K_2}{m} \quad K_v = \frac{-K_3}{m}$$

### 2.3 SIMULACIONES

Utilizando la herramienta Simulink de Matlab se han realizado diferentes simulaciones usando el diagrama mostrado en la figura. En este diagrama el bloque *pendulum* es el sistema no lineal, *LQR* es el bloque de ganancia estática del controlador lineal y el bloque *Kaliora* es el relativo al controlador no lineal. El valor de la ganancia *LQR* ha sido obtenido a través del comando *lqr* de Matlab y los parámetros del controlador *Kaliora* han sido sintonizados con los del *LQR*.

Puede observarse como las señales de salida son muestreadas antes de llegar al controlador. La acción de control se aplica al sistema tras un mantenedor de orden cero y un bloque de saturación que limita su amplitud. Como puede verse existen dos lazos de realimentación. El más externo es el que estabiliza el sistema péndulo-carro, mientras que el interno controla la velocidad de giro.

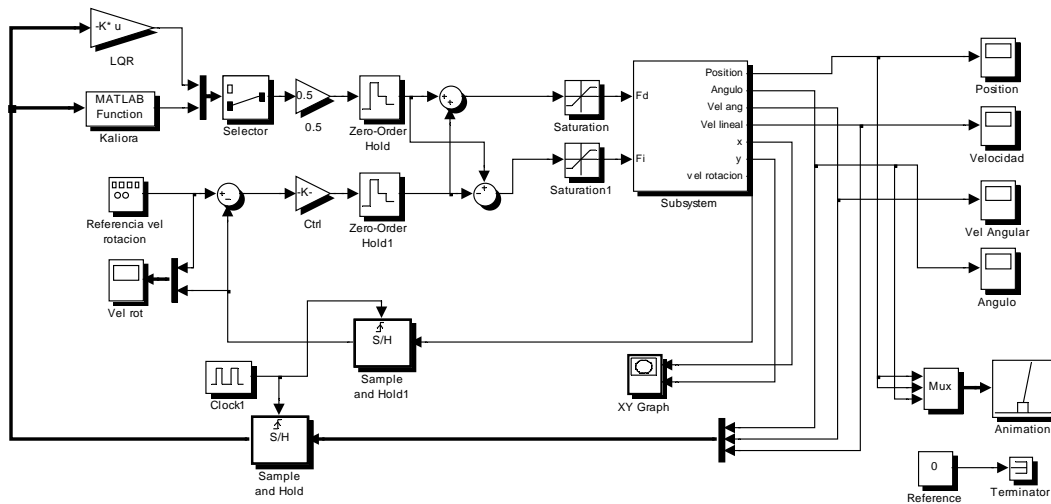


Figura 31: Esquema Simulink para simulación de controladores

En la siguiente imagen se muestra el esquema del modelo del sistema no lineal completo, donde destacan los dos subsistemas (péndulo invertido y vehículo contracción diferencial) cuyas estructuras internas son representadas en las imágenes posteriores.

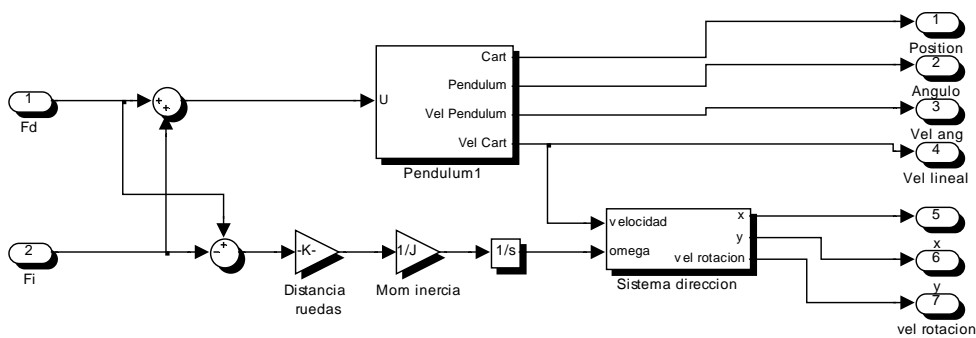


Figura 32: Esquema Simulink modelo completo del sistema

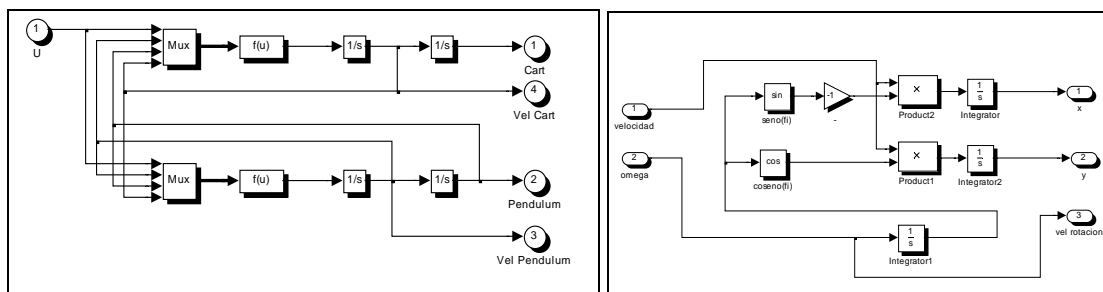


Figura 33: Esquema Simulink modelos del péndulo invertido y vehículo con tracción diferencial

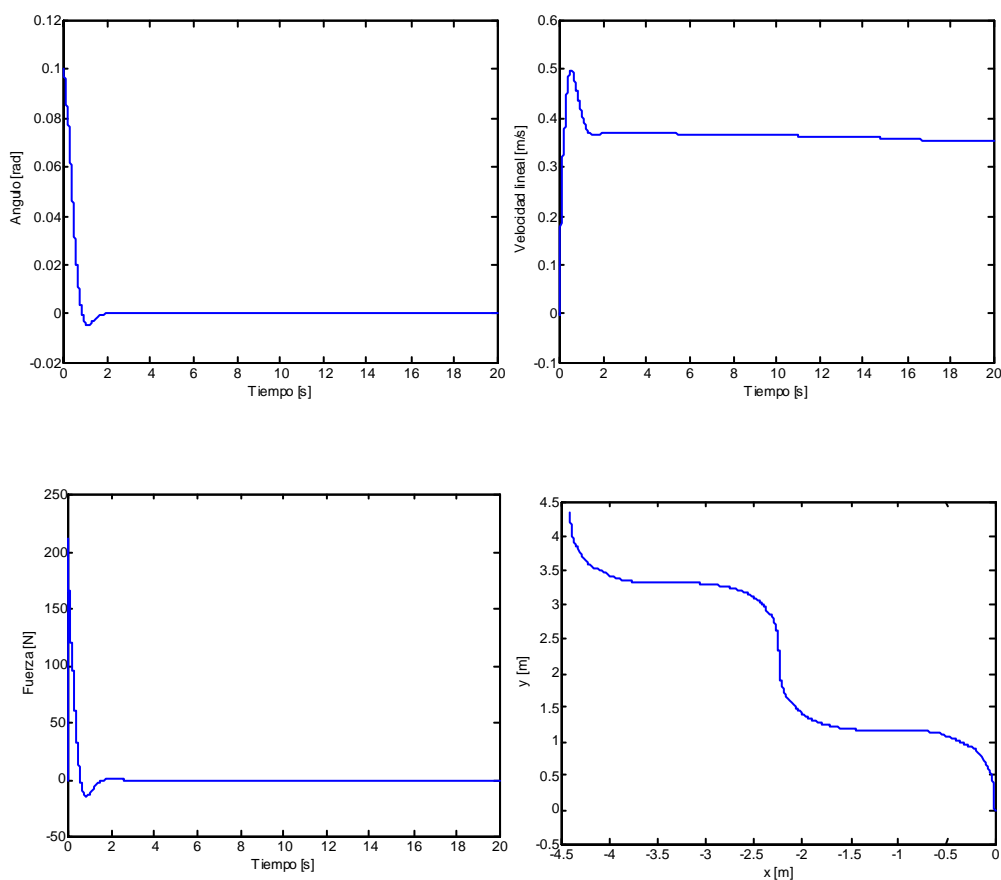
Utilizando el esquema Simulink descrito se han realizado un amplio conjunto de simulaciones. En primer lugar se muestra una de ellas para el sistema controlado con el LQR. Los valores de las matrices de ponderación Q y R son los siguientes;

$$Q = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 100 \end{bmatrix} \quad R = 1$$

y la condición inicial;

$$x(0) = \begin{bmatrix} \theta(0) \\ \dot{\theta}(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0 \\ 0 \end{bmatrix}$$

es decir que el sistema empieza parado con un ángulo inicial de  $0.1 \text{ rad} \cong 6^\circ$ . Se obtiene la siguiente evolución.

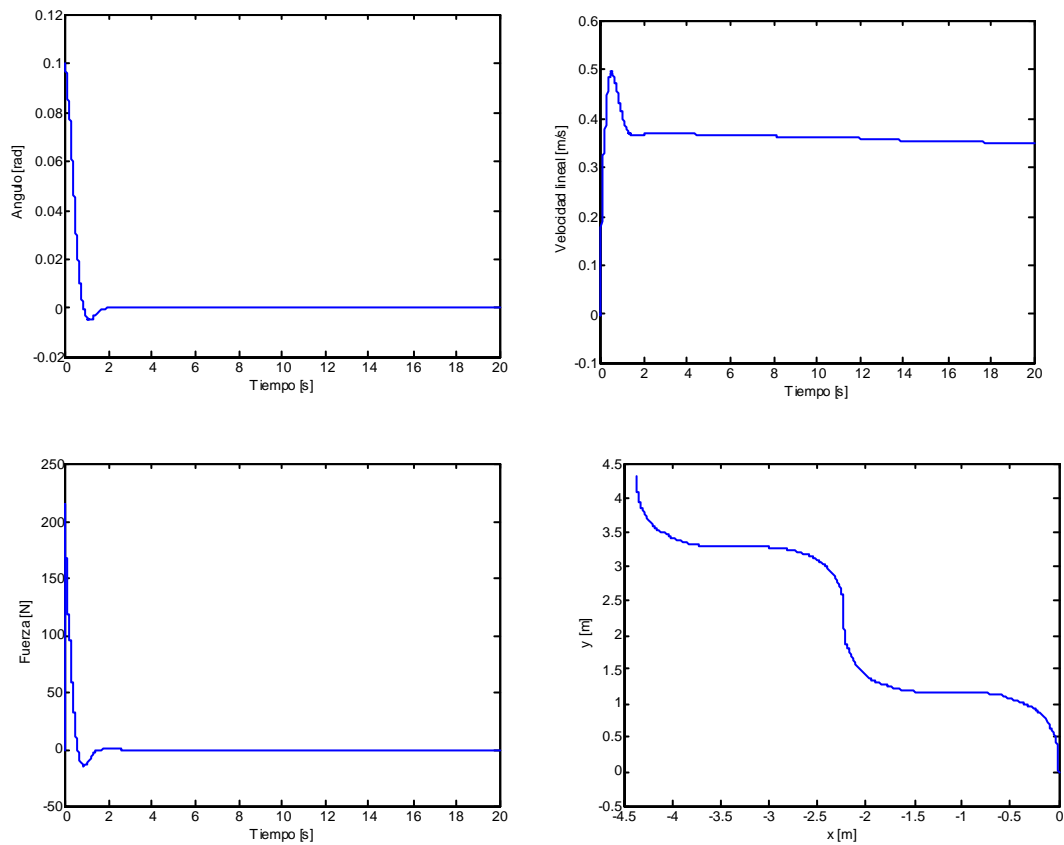


**Figura 34: Simulación del sistema controlado por LQR**

En las figuras puede observarse como la entrada y el ángulo se estabilizan rápidamente, mientras que la velocidad converge a cero más lentamente. Esto es debido a que el parámetro de ponderación de la velocidad  $q_3$  es mucho más pequeño que los otros.

Además, se ha representado la trayectoria seguida por el sistema en el plano, cuando la referencia proporcionada es una senoide.

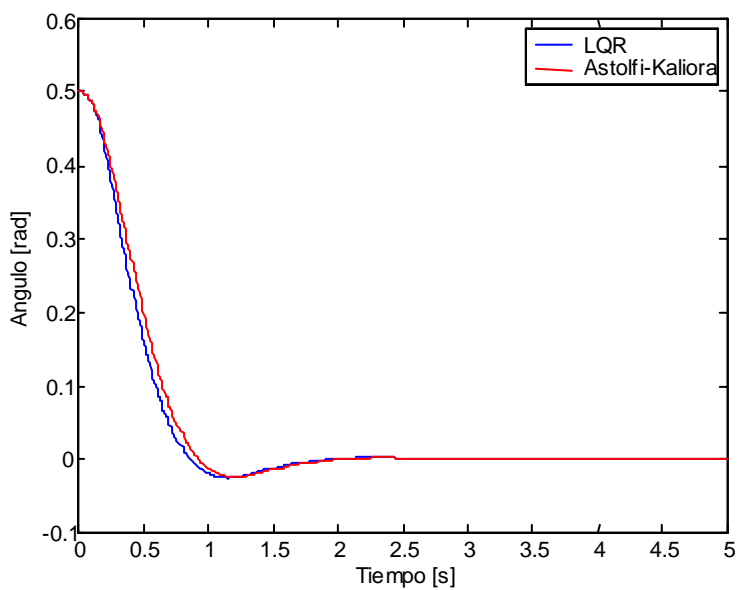
Repetiendo la simulación con el controlador de Astolfi-Kaliora, sintonizado a partir de los parámetros del LQR, las evoluciones temporales son las siguientes



**Figura 35: Simulación del sistema con controlador No lineal**

Las figuras muestran una evolución del sistema muy parecida a la obtenida con el controlador *LQR*.

Para mostrar las diferencias entre los dos controladores es necesario llevar al sistema más lejos de la posición de equilibrio. Así, realizando nuevas simulaciones con un ángulo inicial mayor se obtienen los siguientes resultados



**Figura 36: Comparación de controladores**

## **2.4 EFECTO DE LA SATURACIÓN, TIEMPO DE MUESTREO Y RETRASO**

La presencia de una saturación en la entrada del sistema a controlar conlleva que para ángulos iniciales grandes el control no consigue estabilizarlo. Esto es debido a que, para ángulos mayores de un valor crítico, la entrada que permitiría recobrar la estabilidad es mayor de la saturación por lo que no puede ser aplicada. Por tanto, el efecto de la saturación es una disminución de la cuenca de atracción del sistema. En otras palabras que los estados iniciales para los que el sistema en bucle cerrado es estable, forman una región menor de la que se tendría sin ningún límite en la amplitud de la entrada.

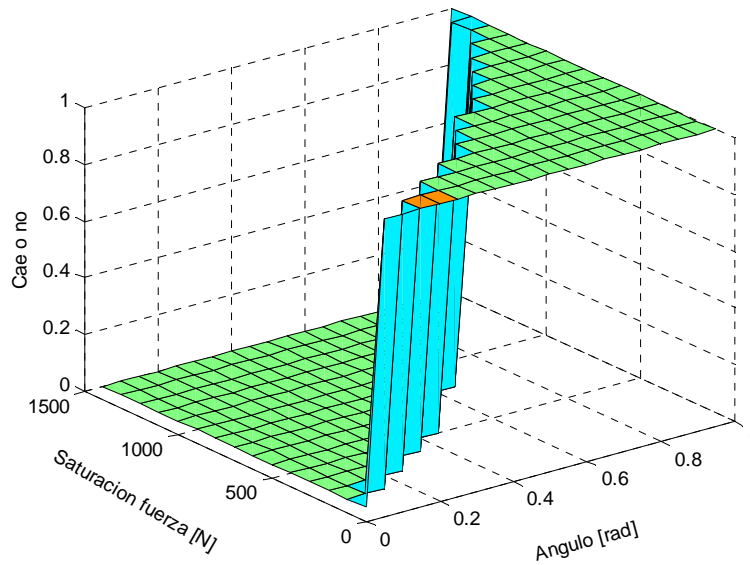
Otras causas de disminución de la cuenca de atracción son el tiempo de muestreo y la presencia de retraso en bucle cerrado. Los controladores han sido diseñados en tiempo continuo, es decir, como si en la entrada al controlador estuvieran disponibles, en cada instante, los valores actuales del vector de estado. De esta forma en cada instante se aplica al sistema la entrada de control oportuna. En realidad esto es un comportamiento ideal, diferente del real. El control se realiza digitalmente así que los datos en entrada y salida del controlador son valores disponibles en instantes discretos. Por esto es necesario un convertidor analógico-digital para muestrear la salida del sistema y uno digital-analógico en la entrada, el mantenedor de orden cero en el modelo Simulink. Además, la presencia de estos convertidores se traducen en un retraso en bucle cerrado. En el sistema real el controlador está constituido por un algoritmo que, en cada instante de muestreo, solicita la adquisición de medidas a los sensores. Los sensores pueden necesitar un pequeño intervalo de tiempo para proporcionar estas medidas, causando el retraso.

El efecto del tiempo de muestreo también conlleva una pérdida de rapidez en la respuesta del controlador. Así, si la entrada de control es proporcionada cada  $T$  segundos, durante ese intervalo se mantiene constante y por tanto con un valor diferente de los que tendría si fuera tiempo continuo y sin retraso. Esta pérdida de rapidez, de precisión, afecta al comportamiento del sistema en bucle cerrado disminuyendo la región de atracción. Claramente cuanto más grande es el tiempo de muestreo y el retraso, mayor es el efecto de éstos.

A continuación se ilustran los resultados de un análisis del efecto de saturación, tiempo de muestreo y retrasos al variar éstos. Los resultados obtenidos son los mismos para LQR y para el controlador de Astolfi-Kaliora.

La primera prueba ha consistido en realizar una sucesión de simulaciones considerando el controlador tiempo continuo, es decir sin muestrear las señales, y variando el ángulo inicial y el valor de saturación. Concretamente, se ha fijado un ángulo inicial y un valor de saturación, se ha simulado para evaluar si el controlador es capaz de llevar a cero el ángulo que forma la barra con la vertical. Posteriormente, se ha repetido la simulación incrementando el ángulo inicial. Cuando todos los ángulos de interés han sido analizados, se ha incrementado el valor de saturación y repetido las simulaciones para los diferentes ángulos. Se ilustra en la figura el resultado obtenido con este procedimiento.

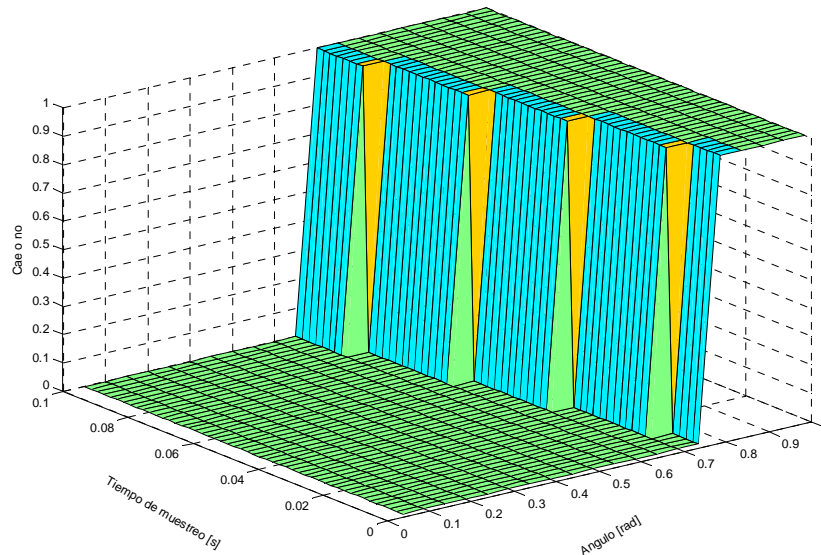
En los ejes del plano se hallan los valores del ángulo inicial, comprendido entre 0.05 rad y 1 rad, y de la saturación de la fuerza, entre 100 N y 1500 N. La superficie representa los resultados de las simulaciones considerando cero para los valores que han conseguido la estabilidad y uno si el bucle no es estable.



**Figura 37: Influencia de la saturación sobre la estabilidad**

Se puede notar como, al aumentar el valor de la fuerza admitida en la entrada, crece el ángulo inicial estabilizable. Por ejemplo, con saturación de 1000 N, que es la que pueden proporcionar los motores utilizados, el sistema consigue estabilizarse para ángulo de hasta 0.75 rad, es decir, de cerca 43°.

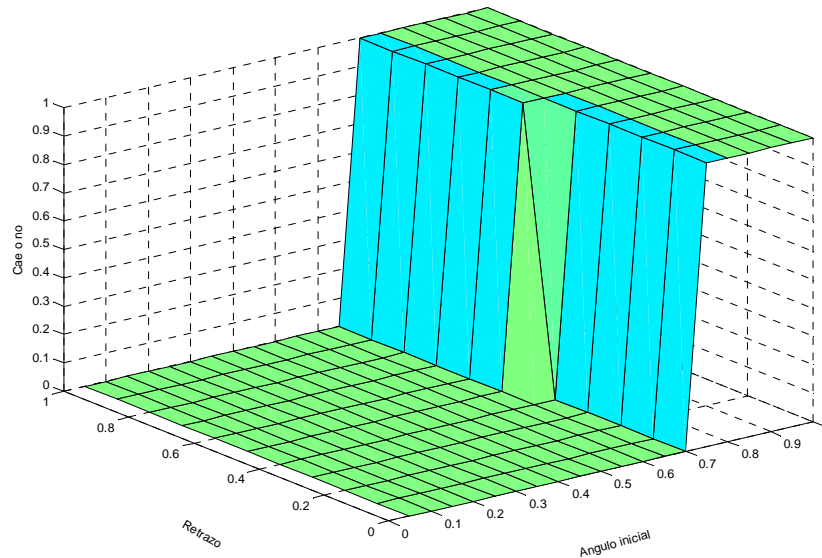
Posteriormente se ha introducido un bloque de muestreo, con saturación fija en 1000 N y se ha simulado variando el ángulo inicial y el valor del tiempo de muestreo entre 0.002 s y 0.1 s con incremento de 0.002 s.



**Figura 38: Influencia del tiempo de muestreo sobre la estabilidad**

Se observa como utilizando tiempos de muestreo pequeños, hasta 0.01 s, el máximo ángulo inicial estabilizable es 0.75 rad, es decir, el mismo del sistema tiempo continuo. El valor crítico

del ángulo inicial disminuye de forma lineal al crecer el tiempo de muestreo, hasta llegar al valor de 0.55 rad para  $T=0.1$  s.



**Figura 39: Influencia del retraso sobre la estabilidad**

Finalmente en la figura de arriba se ilustran los efectos del retraso en el bucle cerrado. Se ha fijado el valor de saturación a 1000 N y el tiempo de muestreo a  $T=0.02$  y se han repetido las simulaciones variando el valor del retraso. Nótese que los valores de retraso que aparecen están normalizados con el tiempo de muestreo, es decir, se escalan según el cociente retraso partido por el tiempo de muestreo. Se han utilizados valores que van de 0 a 1, que se corresponden a un retraso entre 0 y 0.02 s. Se puede observar que la influencia del retraso sobre la estabilidad es pequeña; la diferencia del ángulo crítico entre retraso nulo y retraso igual al tiempo de muestreo es de 0.05 rad, menos de  $3^\circ$ .

### Referencias

F. Salas, F. Gordillo, J. Aracil. A forwarding controller for the pendulum on a cart. April 2004