UACM
Universidad Autónoma
de la Ciudad de México
Nada humano me es ajeno

COLEGIO DE CIENCIA Y TECNOLOGÍA
LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

"Desarrollo de un péndulo invertido didáctico e implementación de un controlador difuso"

TRABAJO RECEPCIONAL
PARA OBTENER EL TITULO DE LICENCIADO
EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

PRESENTA:
OSCAR MOLOTLA GARCÉS

Director del trabajo recepcional
M.I. Amaranto de Jesús Dávila Jáuregui

Ciudad de México, octubre 2016.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

Agradecimientos

A mi madre y padre porque gracias su apoyo y comprensión, he logrado concluir esta etapa de mi vida y finalizar este proyecto que en principio parecía una tarea titánica e interminable. También les doy las gracias por aguantarme los años que me tarde en salir de la universidad :). A mi hermana por su apoyo y por haber hecho este camino más fácil.

A mis amigos por todos esos momentos que pasamos en las aulas, en los laboratorios, en la biblioteca y en cada uno de los proyectos que realizamos, que por momentos parecían interminables.

A mi asesor Amaranto Dávila Jáuregui por su orientación, dedicación, comprensión y su apoyo para el desarrollo de este proyecto.

A mi tutor Gabriel Bengochea Villegas por darme tanto la orientación como la comprensión y ayuda a lo largo de la licenciatura.

A mis lectores el profesor Marcos Ángel Gonzáles Olvera, el profesor Fermi Vázquez Villanueva, el profesor Gabriel Bengochea Villegas,el profesor Manuel Alberto Soriano Ávila por brindarme el tiempo y comprensión para revisar mi tesis.

A al laboratorio de innovación y desarrollo tecnológico del plantel San Lorenzo Tezonco por el apoyo y las facilidades para el desarrollo de este proyecto.

A la UACM por darme la oportunidad de ser parte de esta institución y abrirme las puertas a una excelente educación académica. Por darme el apoyo para la impresión y el empastado de mi tesis.

¡Gracias!

Resumen

En la presente tesis se diseñó e implementó un péndulo invertido didáctico, el cual está controlado por medio de una técnica basa en lógica difusa y el controlador se implementó en una computadora embebida. El sistema cuenta una interfaz gráfica de usuario que permite recibir y enviar datos vía remota desde cualquier PC.

Índice general

Capítulo 1

Introducción

Durante su historia la humanidad ha tratado de controlar diversos sistemas o mecanismos para lograr llevar una vida más cómoda, los cuales se han vuelto cada vez más complejos gracias a los avances que se han logrado principalmente en el área tecnológica y científica, teniendo como resultado el desarrollo de robots, autos híbridos y aeronaves no tripuladas por mencionar algunos. Dichos sistemas necesitan ser controlados ya que deben cumplir con ciertos requerimientos, un ejemplo es el control de posicionamiento de los propulsores que tienen los transbordadores espaciales,

el cual es necesario para lograr que al despegar éste mantenga una trayectoria vertical y con un cierto grado de inclinación, logrando que el lanzamiento sea lo más eficiente posible y seguro posible.

1.1 Planteamiento del problema

El péndulo invertido es un sistema muy utilizado en el área de control, dado que permite realizar tanto prácticas de laboratorio como proyectos de investigación para probar y validar distintas técnicas de control, debido al reto que representa mantenerlo en equilibrio.

Existen diferentes variantes de péndulo invertido, entre los que destacan el péndulo simple (con un carro sobre un riel), el péndulo de Furuta, el péndulo sobre un vehículo de dos ruedas (Segway) y el que se propone en este trabajo, el cual consiste en un carro que se desplaza a lo largo de un eje con un grado de libertad y en la base de éste se coloca una barra (péndulo invertido) que tiene un desplazamiento rotacional en el mismo sentido al eje del carro, como el mostrado en la figura 1.1.

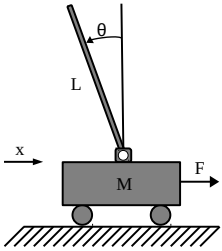

Figura 1.1: Diagrama de péndulo invertido

La gran mayoría de los sistemas didácticos de control de péndulo invertido presentan algunos inconvenientes. Por ejemplo, deben estar conectados a una tarjeta de adquisición de datos y ésta a su vez a una computadora, como consecuencia el traslado del sistema a un aula o algún otro lugar para realizar prácticas o experimentos, resulta muy complicado dado que los elemento que le componen como cables, fuente, pc y tarjeta impiden realizar un fácil traslado.

Una tarea prioritaria en los cursos de ingeniería es el uso de equipo especializado que permita a los estudiantes hacer diferentes pruebas o experimentos en software, lo cual no es eficiente cuando se desea

hacer alguna modificación de programación y observar el comportamiento del sistema por medio de gráficas en la PC, puesto que esto solo se puede realizar en una computadora impidiendo que cada estudiante realice las modificaciones que crea pertinentes. Sería importante mencionar que la mayoría de estos equipos didácticos son muy costosos, lo que ocasiona que en casi todos los laboratorios de control se tengan pocos sistemas de este tipo.

Se desea construir un sistema que cubra dichas deficiencias, debido a estos inconvenientes una posible solución sería diseñar e implementar un péndulo invertido que sea fácil de trasladar, con una conexión inalámbrica, que trasmita y reciba datos a través de una interfaz gráfica de usuario, dando la facilidad a cada estudiante trabaje de manera independiente por medio de una PC.

1.2 Objetivos

Diseñar e implementar un péndulo invertido didáctico, que ayude a fortalecer los conocimientos adquiridos en clase por el estudiante. El sistema deberá cumplir

con las siguientes características:

- Bajo consumo de energía.
- Inalámbrico.
- Portátil y compacto.
- Económico.
- Con flexibilidad para modificar la mecánica, el hardware y el software.

1.3 Justificación

El péndulo invertido es muy estudiado debido a que es un sistema no lineal, subactuado y cuenta con puntos de equilibrio inestables. Al ser un sistema mecánico-eléctrico presenta diferentes problemas como la medición de parámetros, fuerzas, retrasos, entre otros. Es por estas características que surge el interés por controlarlo y se elige la técnica de conjuntos difusos debido a que ésta no requiere el modelo matemático del sistema, ya que al ser un primer prototipo, será complicado

conocer algunos parámetros de los elementos utilizados. Se seleccionó un microcontrolador y una Raspberry pi con la finalidad de tener un péndulo invertido inalámbrico.

Como ya se mencionó anteriormente el sistema deberá ser económico, ya que la gran mayoría tienen un costo excesivo lo cual afecta a los estudiantes, debido a que no son de fácil acceso y ello complica la realizar tanto de prácticas grupales como individuales.

1.4 Alcances y limitaciones

Diseñar e implementar un prototipo de un péndulo invertido didáctico, que cuente con dos tarjetas de control donde se implementen tanto un controlador digital difuso como una interfaz gráfica de usuario.

A través de la interfaz será posible observar el comportamiento del sistema por medio de gráficas (solo se podrá observar una a la vez) y así mismo se podrán modificar los valores de los conjuntos difusos (solo los de entrada) del controlador, otra característica es que podrá ser observada vía remota desde cualquier computadora.

1.5 Metodología

Investigar las posibles formas de construir el sistema y una vez realizado, diseñar e implementar el péndulo invertido con base a los objetivos planteados. Desarrollar la etapa electrónica, comenzando por la elaboración de la tarjeta o pcb para el microcontrolador, posteriormente se realizaran diversos programas en mikroC con la finalidad de caracterizar los dispositivos electrónicos y poder desarrollar el controlador digital. Mientras que para la Raspberry Pi se llevara a cabo un estudio de su funcionamiento y de las formas en las que puede operar, teniendo la información anterior se procederá a establecer la comunicación entre el microcontrolador y la microcomputadora, se diseñará una interfaz gráfica de usuario y finalmente se efectuaran diversos experimentos para sintonizar el controlador.

Capítulo 2

Antecedentes

El primer trabajo significativo en control automático fue el regulador de velocidad centrífugo de James Watt para el control de la velocidad de una máquina de vapor, en el siglo dieciocho. Minorsky, Hazen y Nyquist, entre muchos otros, aportaron trabajos importantes en las etapas iniciales del desarrollo de la teoría de control. En 1922, Minorsky trabajó en controladores automáticos para el guiado de embarcaciones, y mostró que la estabilidad puede determinarse a partir de las ecuaciones diferenciales que describen el sistema [Ogata, 2010].

Por otra parte en la década de 1940 lo ingenieros lograron desarrollar sistemas de control lineales en lazo cerrado, los cuales tenían un mejor desempeño gracias a el desarrollo de los métodos de análisis de respuesta en frecuencia y los diagramas de Bode. Más tarde al finales de las cuarentas e inicios de los cincuentas fue implementado por completo el método de lugar de las raíces propuesto por Haves, siendo este método y el de respuesta en frecuencia el núcleo de la teoría del control clásico.

Con el paso del tiempo la teoría de control clásica, que trabaja con sistemas con una entrada y una salida, pierde fuerza contra el control moderno, debido a que las nuevas plantas no solo tienen múltiples entradas y salidas sino que se vuelven más complejas de analizar, haciendo necesario el uso de un gran número de ecuaciones para describir a estos sistemas modernos. El análisis en el dominio del tiempo y el estudio de variables de estado forman las bases de la teoría de control moderno. La primera de ellas no fue posible sino hasta 1960 debido a la disponibilidad de las computadoras digitales. La complejidad del análisis las plantas modernas y los crecientes requerimientos de éstas como la precisión, peso y costo impulso el desarrollo de la

teoría de control moderno.

Entre los años de 1960 a 1980 se realizaron diversas investigaciones de control adaptable y de control óptimo, tanto de sistemas determinísticos como estócasticos.

Con el paso del tiempo la tecnología avanzó en gran medida, dando como resultado sistemas más complejos que dieron solución a las nuevas necesidades de la revolución industrial, para lo cual entre los cuarenta y cincuenta se tenía una gran cantidad de sistemas industriales que contaban con controladores PID para temperatura, presión, velocidad, etc.

2.1 Conceptos básicos de control

Se le nombra **sistema** a un conjunto de elementos que actúan juntos y realizan una tarea determinada, con un objetivo común. El concepto de sistema no esta restringido solo para sistemas físicos, sino que se debe interpretar en un sentido muy amplio desde un sistema biológico hasta un económico.

Se le nombra **planta** a un sistema o un conjunto de elementos que funcionan juntos para efectuar un proceso. A menudo en el área de control se le dice planta a cualquier objeto físico a controlar.

Un **sistema de control** está formado por subsistemas y procesos (o plantas) unidos con el fin de controlar las salidas de un proceso. Por ejemplo, un horno produce calor como resultado del flujo de combustible. En este proceso, los subsistemas, llamados válvulas de combustible y actuadores de válvulas de combustible, se usan para regular la temperatura de una habitación al controlar la salida de calor del horno. Otros subsistemas, por ejemplo los termostatos que funcionan como sistemas detectores, miden la temperatura de la habitación. En su forma más sencilla, un sistema de control produce una salida o repuesta para una entrada o estimulo [Nise and Romo, 2002].

En la figura 2.1 se muestra un diagrama de bloques de un **sistema de control en lazo cerrado**. Como se puede observar Y representa la señal de salida, la cual realimenta al sistema y esto permite que el controlador corrija el error de dicha señal con respecto a la de referencia R, a diferencia de un sistema de control en lazo abierto que, al no contar con una retroalimentación,

tiene una respuesta menos exacta y por consiguiente es implementado en sistemas que no necesitan tanta precisión.

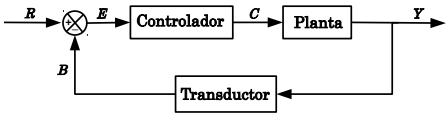

Figura 2.1: Sistema de control en lazo cerrado

2.2 Técnicas de control

Existen diferentes técnicas de control las cuales se dividen en tres grandes grupos: control clásico, moderno e inteligente, para esta última se aborda la desarrollada mediante conjuntos difusos. A continuación se en listan las características de cada una de las técnicas mencionadas.

- Control clásico:
  - Sistemas con una entrada y una salida.
  - Análisis en el dominio de la frecuencia.

- Función de transferencia.
- Para sistemas lineales.
- Ecuaciones diferenciales.
- Modelo físico del sistema.

- Control moderno:
  - Sistemas con múltiples entradas y salidas.
  - Análisis en el dominio del tiempo.
  - Espacio de estados.
  - Sistemas lineales y no lineales.
  - Ecuaciones matriciales.
  - Modelo físico del sistema.

- Control difuso:
  - Sistemas con múltiples entradas y salidas.
  - Conjuntos difusos para entradas y salidas.
  - Reglas difusas.
  - Sistemas lineales y no lineales.
  - No es necesario el modelo físico del sistema.

- Permite trabajar con información que no es exacta o precisa.

En los listados anteriores podemos observar las diferencias que existen entre estas tres técnicas de control.

2.3 Lógica difusa

La cantidad y variedad de aplicaciones tecnológicas se han incrementado de forma importante en los últimos años, lo que a llevado a tener sistemas con mayor complejidad y es en esta área donde lógica difusa toma una gran importancia, debido a que hemos deseado que los sistemas puedan realizar procesos o tomar decisiones de manera semejante a como lo realizamos los humanos.

La lógica difusa fue investigada por primera vez en el año de 1965 por Lotfi Asker Zadeh, con el principio de incompatibilidad: "conforme la complejidad de un sistema aumenta, nuestra capacidad para ser precisos y construir instrucciones sobre su comportamiento disminuye hasta el umbral más allá del cual, la precisión

y el significado son características excluyentes" [Siler and Buckley, 2005].

En 1971 Zadeh realizo la publicación de "Quantitative Fuzzy Semantics", donde se dan las bases para la lógica difusa. Más tarde 1973 algunos investigadores comenzaron a aplicar esta lógica en diferentes procesos, dando un gran aporte a esta teoría y a sus aplicaciones. Assilian y Mamdani para el año de 1974 realizaron el primer controlador difuso para una máquina de vapor, pero no es hasta 1980 cuando F.L. Smidth y Co. implementan realmente un controlador de este tipo en una plata cementera en Dinamarca.

En 1987 Hitachi aplicó un controlador difuso para controlar el tren de Sendai y para 1993 Fuji aplico esta lógica para controlar la inyección química en plantas depuradoras de agua en japón y es después de esto que la lógica difusa toma más fuerza. Mientras estas aplicaciones se realizaban Takagi y Sugeno desarrollaron las primeras aproximaciones para las reglas difusas.

Capítulo 3

Diseño y construcción del sistema

En este capítulo se describe el proceso que se realizó para el diseño y la construcción del péndulo invertido. Como primer paso se realizó el subsistema mecánico, en el cual se procedió tanto hacer la búsqueda de algunas piezas como el diseño y elaboración de otras, en función de las posibles dimensiones del sistema. En la etapa de electrónica se dará una breve explicación de cada uno de los dispositivos, proporcionando algunas de sus características electrónicas y finalmente se expli-

ca el procedimiento que se realizó para la elaboración de la tarjeta electrónica (PCB) para el microcontrolador. En la figura 3.1 se muestra el diagrama general del sistema.

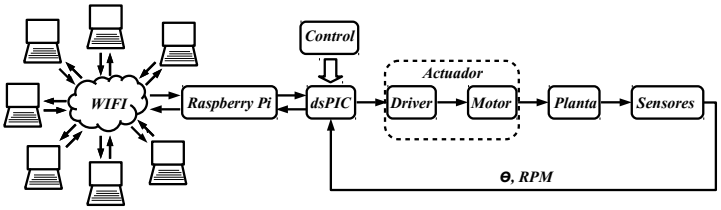

Figura 3.1: Diagrama del sistema

3.1 Diseño y construcción del sistema mecánico

Como primer paso se realizó una investigación de los posibles elementos a utilizar para la construcción del susbsistema mecánico, eligiéndose elementos se en listan a continuación. En la figura 3.2 se muestra un

diagrama de bloques del subsistema.

- Piezas mecánicas:
  - Chasis.
  - Barras para el péndulo y los ejes.
  - Chumaceras para los ejes y el péndulo.
  - Conector y cople para unir el péndulo con el potenciómetro.
  - Bandas y engranes.
  - Llantas.

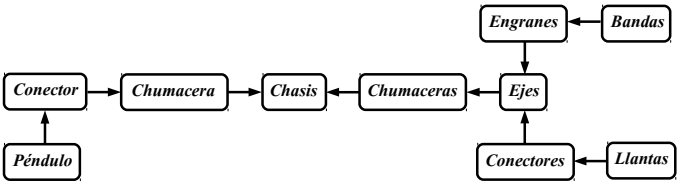

Figura 3.2: Diagrama de bloques del subsistema mecánico

Una vez realizado lo anterior se seleccionaron engranes y bandas con base a las dimensiones del sistema, dichos elementos permitirán acoplar los ejes del carro al motor, cabe resaltar que estos componentes son de plástico con la finalidad de lograr que el sistema sea lo más ligero posible.En la figura 3.3 se muestra una foto.

Figura 3.3: Engranes y Bandas

Como segundo paso se seleccionaron las llantas que tuvieran fácil acoplamiento a los ejes del carro y que las dimensiones de estas lograran obtener un sistema compacto, cumpliendo con las consideraciones de diseño. Como se observa en la figura 3.4 las llantas que se seleccionaron tienen con un cople, lo cual facilitó la unión con los ejes del carro.

Figura 3.4: Llanta

Sabiendo las dimensiones de los engranes y las llantas se diseñaron las piezas restantes en AutoCAD, el cual se utiliza para el dibujo en 2D y modelado en 3D. En las figuras 3.5, 3.6, 3.7, 3.8 y 3.9 se muestra diseño de las chumaceras, los ejes, el péndulo, un conector y un cople respectivamente.

Teniendo los diseños se maquinaron las piezas. Se elaboraron en aluminio debido a que es un material de bajo costo y ligero, comparado con otros materiales. Una vez ya hechas las piezas mecánicas se diseñó el chasis, el cual se realizó en dos partes como se muestra en la figura 3.10, con la finalidad de utilizar el espacio intermedio para colocar la etapa de electrónica y mecánica.

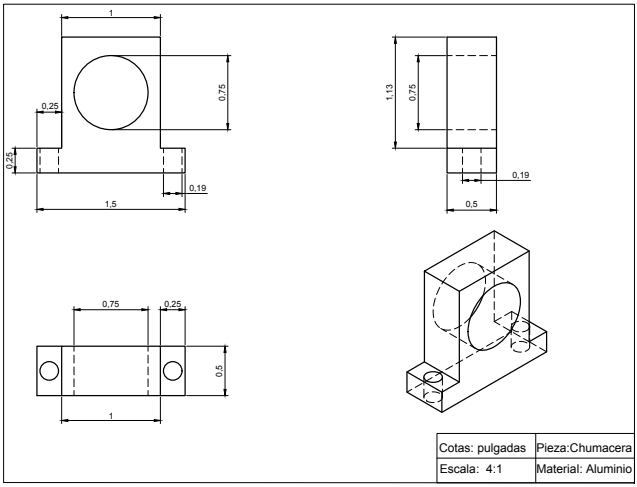

Figura 3.5: Imagen de chumacera

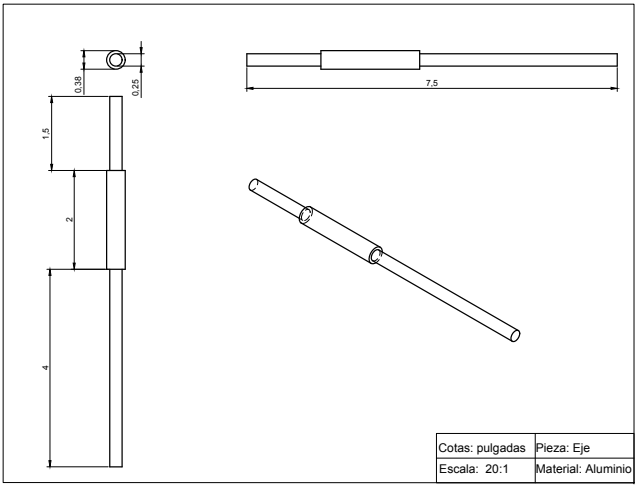

Figura 3.6: Imagen de ejes

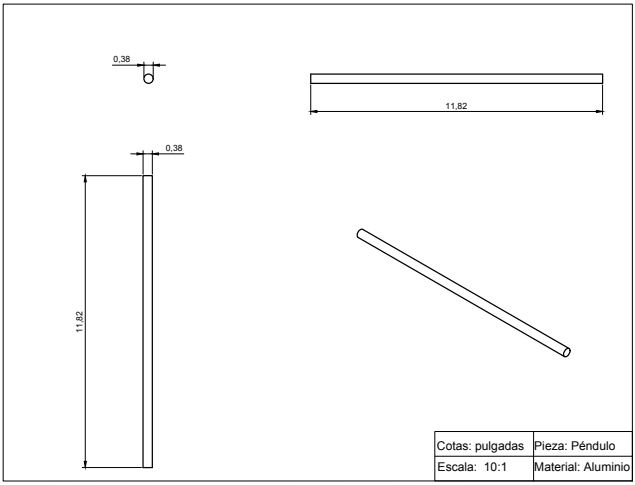

Figura 3.7: Imagen de péndulo

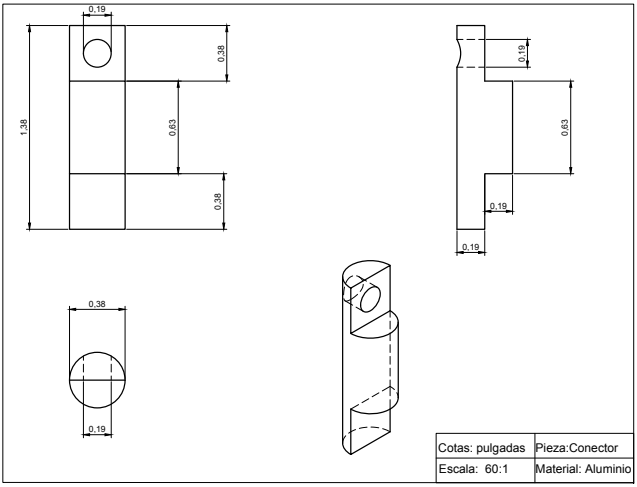

Figura 3.8: Imagen de conector

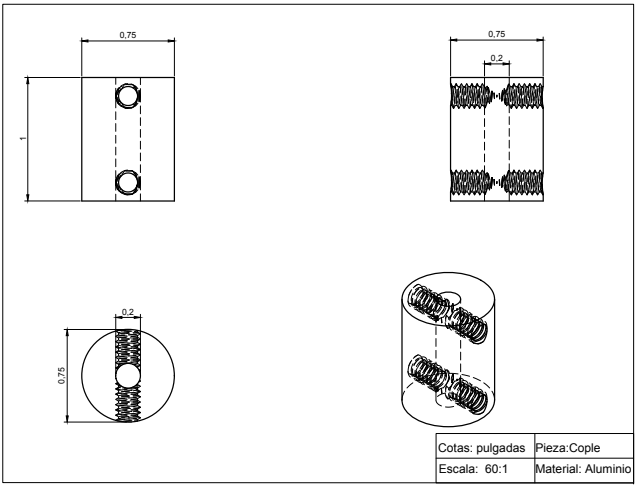

Figura 3.9: Imagen de cople

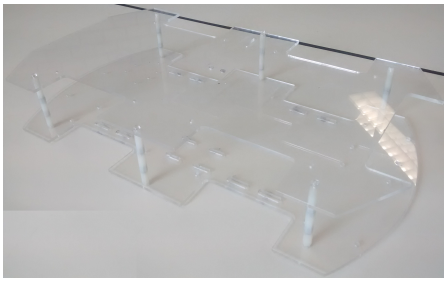

Figura 3.10: Imagen de chasis

3.2 Diseño y construcción del sistema electrónico

Al igual que en la etapa mecánica se realizó como primer paso una investigación de los elementos con los que debería contar el subsistema electrónico dado los requerimientos de éste, a continuación se en listan lo componentes necesarios para desarrollar la etapa electrónica, y en la figura 3.11 se muestra el diagrama de bloques.

- Componentes electrónicos:

- Potenciómetro para conocer posición y velocidad del péndulo.
- Motorreductor con encoder para mover el carro y conocer su velocidad.
- Driver para el motor.
- Convertidor de DC-DC para energizar la Raspberry Pi.
- Batería de iones de litio para energizar al sistema.

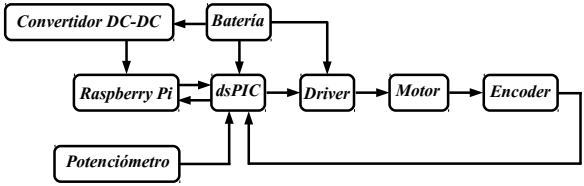

Figura 3.11: Diagrama de bloques del subsistema electrónico

El potenciómetro de precision Bi technologies model 6187R10KL1.0, con una resistencia que varía de

1kΩ a 10kΩ, una tolerancia ±10 % y una linealidad de ±1.0 %. En la figura 3.12 se muestra una fotografía.

Figura 3.12: Potenciómetro de precision [AGElectrónica, 2015]

El potenciómetro se utiliza tanto para medir la posición del péndulo como su velocidad. Para obtener la caracterización del dispositivo se realizo una relación entre el voltaje de la resistencia variable y el ángulo del péndulo. Estas pruebas consistían en colocar al péndulo en un cierto ángulo y medir el voltaje que proporcionaba el potenciómetro, es importante mencionar que para cada ángulo se realizaron varias mediciones y se calculó el promedio. Una vez hecho esto se pro-

cedió a obtener la ecuación de la recta con mínimos cuadrados, en la figura 3.12 se muestra la gráfica.

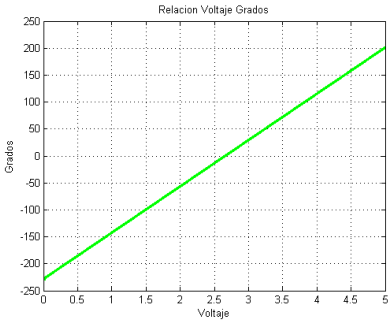

Figura 3.13: Recta

La ecuación de la recta correspondiente a la gráfica seria:

$$y[V] = 85.8851[\frac{V}{\theta}]x[\theta] - 228.8847[V]$$

Se selecciono el motorreductor con encoder de la marca Pololu, ya que es posible operarlo en un rango 6 a 12 VDC y tiene una relación de engranes de 30:1 lo que le proporciona un torque de 8 Nm y un encoder

de cuadratura de efecto hall que proporciona 16 pulsos por revolución en el eje de entrada y 480 en el eje de salida, esta acoplado al motor como se muestra en la figura 3.14.

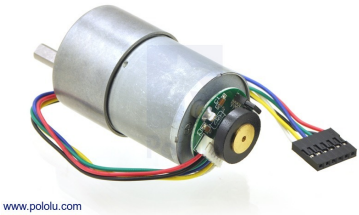

Figura 3.14: Motoreductor con encoder [Pololu, 2015]

El motorreductor puede alcanzar una velocidad máxima 350 RPM aproximadamente, el consumo de corriente sin carga es de 300 ma y de 5 A con el rotor bloqueado. El objetivo del motor es mover el carro y a través de este se calcula la velocidad a la cual se desplaza el mismo. El motor esta unido a los ejes del carro a través de engranes y bandas, como observar en la figura 3.15.

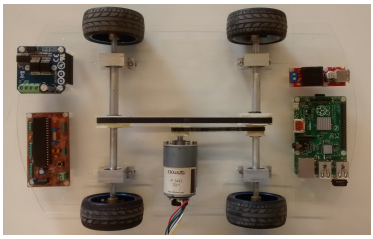

Figura 3.15: Foto de la parte superior del sistema

El motoreductor es controlado por medio de una señal PWM para lo cual utilizaremos un puente H, como interfaz de potencia entre el sistema de control y el motor. En en la figura 3.16 se muestra.

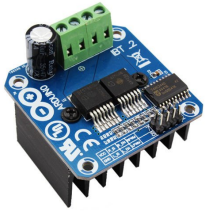

Figura 3.16: Puente H [Amazon, 2015]

El driver está basado en los integrados BTS7960, los cuales son dos medios puentes y cada uno está for-

mado por un MOSFET de canal p en la parte superior y uno de canal n en la parte inferior, sus limites de operación son 45 volts y hasta 43 amperes. Por medio del dsPIC se le proporcionan tres señales al puente H, la primera de ellas es el PWM y las otras dos señales corresponden al cambio de giro del motor, para realizar el acoplamiento de impedancias de estas señales con los medios puentes se utiliza un buffer de 8 bits con matricula 74HC244.

Uno de los aspecto que se debe tener en cuenta para que el funcionamiento de la Raspberry sea el adecuado, es la energización de ésta por lo cual se debe contar con una fuente de alimentación que proporcione 5 volts y una corriente 2 ampers o más, se optó por el convertidor DC a DC que se muestra en la figura 3.17.

Figura 3.17: Convertidor de DC a DC [Jelelectrónica, 2015]

A este convertidor se le pueden proporcionar a la entrada (bornera azul) de 7 a 24 volts y a su salida obtener(conector jack USB tipo A) 5 volts. El convertidor se basa en el rectificador monolítico MP2307, el cual está integrado por MOSFETS de potencia que puede proporcionar una corriente de hasta 4 A en pico y en operación normal de 3 ampers. Este regulador tiene un modo de control de corriente que proporciona una rápida respuesta transitoria.

Para energizar al sistema se selecciono una batería recargable de iones de litio a 12 volts y 4800 mah, la cual se muestra en la figura 3.18.

Figura 3.18: Batería de iones de litio [Topled, 2015]

A dicha batería se conecta el puente H, el microcontrolador y el convertidor DC a DC.

3.3 PCB para dsPIC

El diseño de la PCB para el dsPIC se realizó en el software Proteus 8 Professional el cual se divide en dos programas ISIS, Intelligent Schematic Input System (Sistema de Enrutado de Esquemás Inteligenté) y ARES o Advanced Routing and Editing Software (Software de Edición y Ruteo Avanzado). Como primer paso se desarrollo el esquemático del circuito electrónico para el dsPIC, en la figura 3.19 se muestra una imagen.

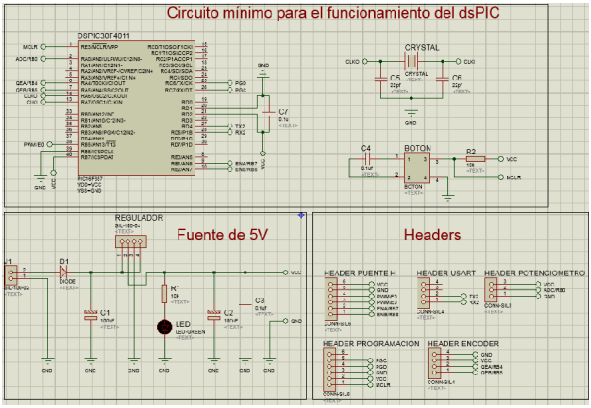

Figura 3.19: Esquemático del circuito electrónico

Una vez terminado el desarrollo del esquemático, se procedió a ubicar y enrutar el circuito, el cual nos permite terminar el diseño del PCB. Algo que es importante mencionar es una de las consideraciones de diseño, la cual fue realizar el PCB de dos caras con la objetivo de reducir al máximo el tamaño de este. En la figura 3.20 se muestra una foto de diseño finalizado del PCB en ARES.

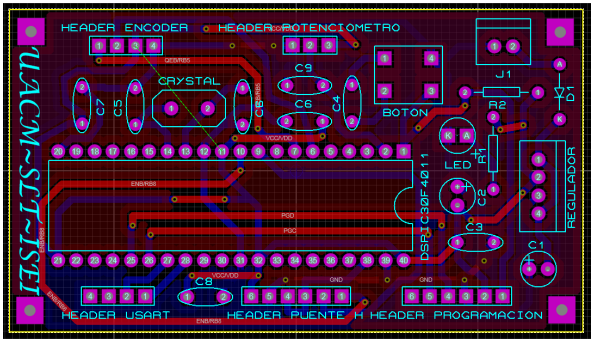

Figura 3.20: Fotografia del PCB en ARES

Finalmente se fabricó el circuito impreso, en la figura 3.21 se muestra un foto.

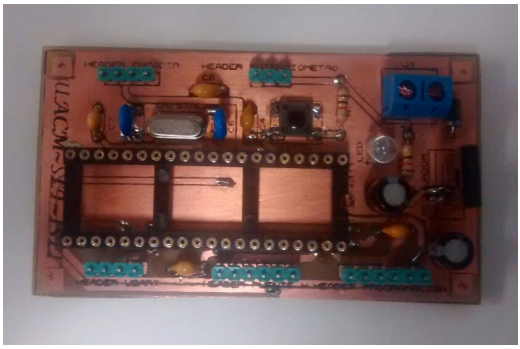

Figura 3.21: Fotografía de la tarjeta electrónica

El microcontrolador seleccionado se muestra en la figura 3.22. Una de las características por las cuales fue elegido es que forma parte de la subfamilia de los dsPIC30F dedicada a las aplicaciones de control de motores, a continuación se muestra algunas de las características de este integrado.

- Memoria Flash programable de 48 KBYTES.
- Memoria RAM para datos de 2048 KBYTES.
- Memoria EEPROM para datos de 1024 KBYTES.

- 5 temporizadores de 16 bits (TIMER).
- Modulo PWM para control de motores con 8 salidas.
- Convertidor A/D de 10 bits con una velocidad de 500 KBPS (CAD).
- Interfaz para codificador de cuadratura (QEI).
- Modulo UART con 2 canales Rx y Tx.
- Modulo I^2C.

Figura 3.22: Fotografía de dsPIC30F4011 [RS, 2015]

Como se muestra en la lista anterior este microcontrolador cuenta con diversas prestaciones, permitiendo

la implementación del control difuso. Algunas de estas son los temporizadores necesarios para las interrupciones, el CAD para la lectura del potenciómetro, el PWM para la señal de control del motor, el QEI para la lectura del encoder y el UART ó el I^2C para comunicacion con la Raspberry. En la figura 3.23 se muestra algunas fotos del ensamblado final del prototipo.

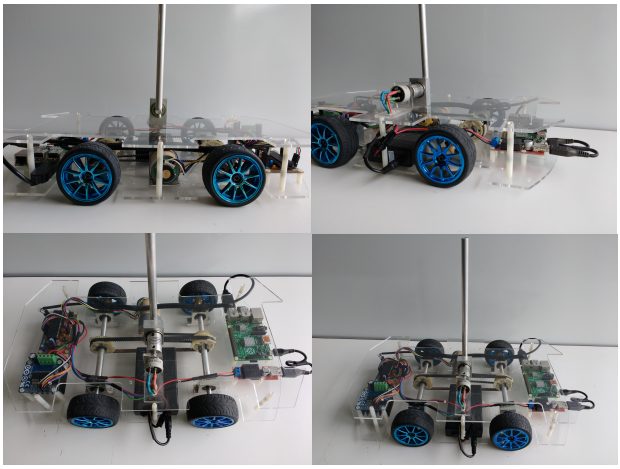

Figura 3.23: Fotografías del sistema péndulo-carro

Capítulo 4

Control

En el siguiente capítulo se darán las bases teóricas de la lógica difusa y los conceptos necesarios para desarrollar el controlador difuso. Posteriormente se describirán las consideraciones de diseño necesarias para desarrollar e implementar el controlador.

4.1 Sistemas difusos

La lógica difusa es una extension de la lógica booleana, donde la pertenecía de un elemento a un conjun-

to toma valores en el intervalo [0,1], y no a los valores discretos 1,2,3... . Permite utilizar variables lingüísticas como muy rápido, medio lleno, poca fuerza, etc. Un ejemplo de la lógica es cuando introducimos una llave para abrir una puerta, si es la primera vez no sabremos cuanta fuerza debemos aplicar y cuanto debemos girar la llave, pero si la puerta la hemos abierto más de una vez conoceremos la fuerza y los giros necesarios, a lo cual lógica clásica aplicaría siempre la misma fuerza y los mismos giros. Esta lógica emula la forma en que nosotros tomamos decisiones, de la forma "si ... entonces". Por ejemplo:

"Si la botella está cerrada entonces aplicaré mayor fuerza para girar la tapa"

"Si la botella está abierta entonces aplicaré menor fuerza para girar la tapa"

"Si el garrafon está vacío entonces aplicaré poca fuerza para cargarlo"

"Si el garrafon está lleno entonces aplicare mucha fuerza para cargarlo"

Esta forma de tomar decisiones nos permite poder describir sistemas sin conocer el modelo matemático de estos, lo que sería como cuando decimos "llueve mucho", esta descripción la hacemos en base a nuestra

experiencia puesto que no conocemos el modelo matemático que describa el comportamiento del clima, esto es de gran ayuda cuando se desea controlar sistemas muy complejos. Se puede esquematizar en tres principales bloques, en la figura 4.1 se presenta un diagrama de bloques de este proceso.

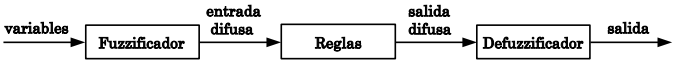

Figura 4.1: Diagrama de bloques de control difuso

Entrada: son la variables obtenidas mediante sensores u observadores, las cuales pueden ser posición, velocidad, torque, etc.

Fuzzificador: este bloque contiene las funciones de membresía, las cuales asignan un grado de pertenecía o valor difuso a las variables medidas.

Entrada difusa: este valor difuso varia en el intervalo $[0, 1]$.

Reglas: es un conjunto de reglas que tratan de describir el comportamiento o el funcionamiento del sistema.

Salida difusa: es un valor difuso que varía en el intervalo $[0, 1]$ y es el resultado de la decisión que se toma con las reglas difusas.

Defuzzificador: Es el proceso que se encarga de transformar el valor difuso a una salida que pueda ser interpretada por el actuador.

Salida: este es el valor que es enviado al actudor, el cual puede ser un PWM, un voltaje, una torque, etc.

4.1.1 Conjuntos difusos

Los conjuntos difusos están definidos por una función de pertenencia, la cual puede tomar valores dentro del intervalo de 0 y 1, siendo 1 el máximo grado de pertenecía que puede tener un elemento, está función esta definida por:

$$A : X \to [0, 1]$$

El primer ejemplo utilizado por Lotfi Asker para representar el concepto de conjuntos difusos, fue el conjunto de hombres altos. La teoría de la lógica clásica, clasifica a los hombres altos en un conjunto al que solo pertenecen aquellos que tengan una estatura superior o igual a un valor determinado, suponiendo que fuera 1.80 metros un hombre con una estatura de 1.79 metros quedaría fuera del conjunto y uno que midiera 1.81 pertenecería al conjunto de hombres altos, lo cual parece no ser lógico para la forma de razonar del humano puesto que solo difieren en dos centímetros uno del otro. La lógica difusa propone que el conjunto de hombres altos no tiene fronteras específicas para pertenecer o no a él, en la Figura 4.2 se muestra un ejemplo de la diferencia de la lógica clásica y la difusa:

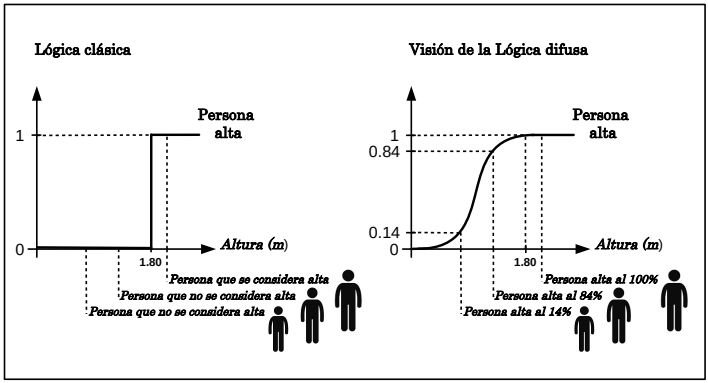

Figura 4.2: Comparación entre conjunto booleano y difuso

Como se puede observar en el caso de la lógica clásica la pertenencia de un hombre al conjunto de hombres altos es tajante, mientras que para el caso de la lógica difusa tiene un conjunto que nos permite tener grados de pertenencia y por lo tanto poder decir que un hombre es menos o más alto que otro dependiendo de su grado de pertenencia, por lo que la lógica difusa se parece más al forma de razonar de cualquier persona.

Existen diferentes funciones para representar los

conjuntos difusos las cuales son llamadas funciones de pertenencia, con el fin de lograr una representación adecuada del comportamiento de cada conjunto. Las funciones comúnmente utilizadas por su simplicidad matemática son la trapezoidal, triangular, gaussiana y sigmoidal, mostradas en las figuras 4.3, 4.4, 4.5 y 4.6.

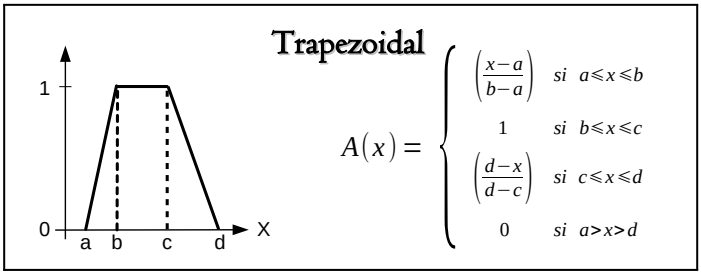

Figura 4.3: Conjunto difuso trapezoidal

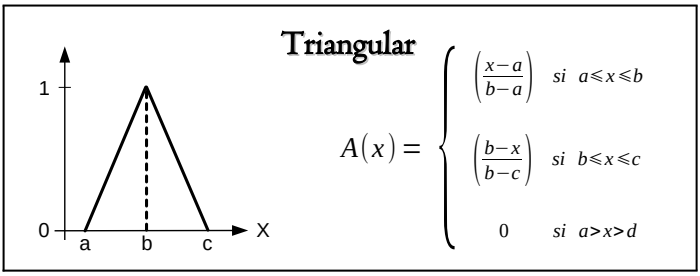

Figura 4.4: Conjunto difuso triangular

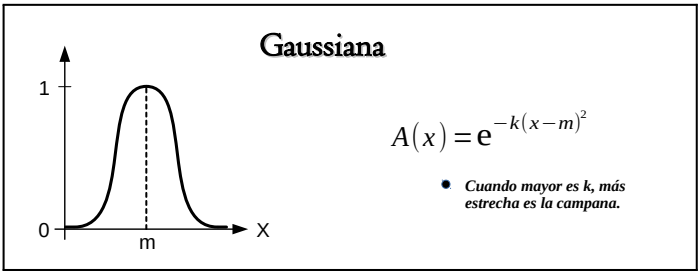

Figura 4.5: Conjunto difuso gaussiana

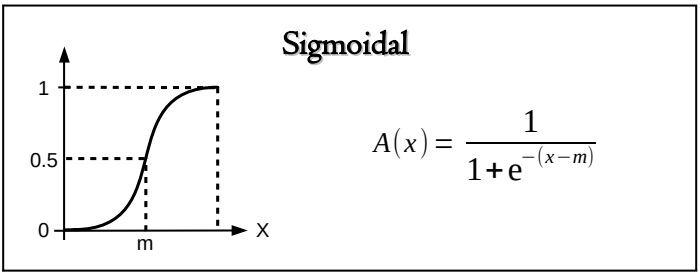

Figura 4.6: Conjunto difuso sigmoidal

4.1.2 Operaciones con conjuntos difusos

Las operaciones que se pueden realizar para conjuntos difusos son semejantes a las utilizadas en los conjuntos booleanos (unión, intersección y complemento), que son las siguientes:

- La unión (OR) de dos conjuntos difusos la podemos obtener de las siguientes dos maneras
  - $(A \cup B)(x) = A(x) \vee B(x) = A(x) + B(x) - A(x)B(x)$

- $(A \cup B)(x) = min\{A(x) + B(x), 1\}$

- La intersección (AND) de dos conjuntos difusos la podemos obtener de las siguientes dos maneras
  - $(A \cup B)(x) = A(x) \wedge B(x) = A(x)B(x)$
  - $(A \cap B)(x) = max\{A(x) + B(x) - 1, 0\}$
- El complemento (NOT) de un conjunto difuso esta dado por
  - $(X \cap A)(x) = 1 - A(x)$

Estos operadores son llamados conectores difusos y las funciones que definen la unión e intersección de conjuntos difusos pueden ser generalizadas si cumplen ciertas restricciones. Las funciones que cumplan con estas condiciones se les nombran como Conorma Triangular (Conorma-T) y Norma Triangular (Norma-T)

4.1.3 Reglas difusas

Existen diferentes tipos de reglas difusas, las cuales son de la forma "Si x es A entonces B es y", donde A y B son conjuntos difusos, x representa la entrada y y

la salida. A continuación se dará una breve explicación de tres de las reglas más utilizadas.

4.1.3.1 Modelo Mamdani

En este modelo se tienen conjuntos difusos tanto para las entradas como para la salida y las reglas tienen la siguiente forma:

$$R_i : \text{Si } x \text{ es } A_i \text{ entonces } y \text{ es } B_i,\ i = 1, 2, 3, \ldots, n,$$

donde y es la salida del sistema, x representa todas la posibles entradas al sistema y cada una de estas le pertenece al menos un conjunto A_i, y según las reglas R_i le corresponde un conjunto B_i. Para hacer el cálculo de las reglas se utiliza la norma-t.

$$R_{i,x}(y) = A_i(X) \wedge B_i(y)$$

ya que sean calculado los valores de todas las reglas se realiza la acumulación de estas, utilizando la conorma-t

$$R(y) = \bigvee_{i=1}^{n} R_i(y)$$

Una vez realizado el proceso de acumulación se hace la defuzzificación para obtener a y, dicho valor final se puede calcular mediante diferentes métodos matemáticos como el cálculo del centroide, el promedio máximo, el promedio ponderado, entre otros.

4.1.3.2 Modelo Larsen

El modelo Larsen es muy parecido al Mandani a excepción del cálculo de las reglas, este utiliza la siguiente norma-t

$$R_{x,i}(y) = A_i(x) * B_i(y)$$

como lo indica la regla se multiplica $A_i(x)$ por $B_i(y)$ y después de obtener este valor se realiza la acumulación como en el modelo anterior.

4.1.3.3 Modelo Takagi-Sugeno-Tang (TSK)

Al igual que los modelos anteriores la primera etapa se realiza de la misma manera y las reglas utilizan ecuaciones que están en función de las variables de entrada del sistema, las cuales tienen la forma:

R_i : Si x_1 es A_{i1} y x_2 es A_{i2} y ... y x_n es A_{in} entonces $y = f_i(x_1, x_2, \ldots, x_n)$

la acumulación se obtiene realizando el mismo procedimiento de los modelos anteriores y para obtener el valor de la salida se utiliza la siguiente ecuación:

$$y_f = \frac{\sum_{i=1}^{n} f_i(X) * A_i(X)}{\sum_{i=1}^{n} A_i(X)}$$

donde n es el número de reglas y X corresponde a las variables del sistema.

4.1.3.4 Sistema Difuso tipo singleton

Este modelo se deriva del anterior, pero a diferencia del Takagi-Sugeno-Tang el valor final se obtiene con el promedio ponderado debido a que la función que se elige para obtener este valor cambia solo en un eje y se mantiene constante en el otro eje, como se muestra en la figura 4.7.

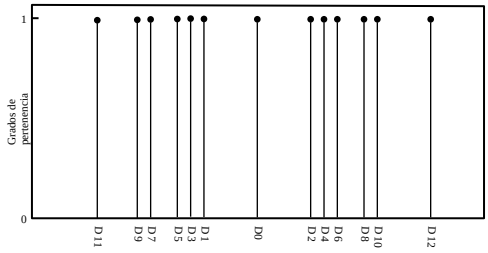

Figura 4.7: Función Singleton

Las reglas toman la forma:

R_i : si x_1 es A_{i1} y x_2 es A_{i2} y $\ldots$ y x_n es A_{in} entonces Y es D_i

La etapa de fuzziificación se realiza de la misma forma que los modelos anteriores, en la parte de activación se emplea la siguiente norma-t.

$$R_{i,x} = A_i(X) * D_i$$

Para la defuzzificación se utiliza el promedio ponderado como ya se menciono anteriormente, en la si-

guiente ecuación se muestra la forma de obtener este valor.

$$Y_f = \frac{\sum_{i=1}^{n} A_i(X) * D_i}{\sum_{i=1}^{n} A_i(X)}$$

4.2 Consideraciones de diseño del control difuso

Las tres principales consideraciones que se deben tener para el diseño del controlador difuso son el fuzzificador (conjuntos difusos de entrada), las reglas de conocimiento y la defuzzificador. En la etapa de fuzzificación , como ya se ha mencionado es la encargada de convertir mediante funciones las variables de entrada del sistema a valores difusos, y del valor asignado dependerá del grado de pertenencia que tengan las variables de entrada respecto a los conjuntos difusos. Un sistema puede tener varios conjuntos difusos de entrada, lo que implica tener control mas fino pero mas complicado de implementar.

4.2.1 Conjuntos difusos para la Fuzzificación

Para el sistema péndulo-carro se eligieron como variables de entrada tanto la posición como la velocidad del péndulo y la velocidad del carro. Para cada variable se asignaron tres conjuntos difusos y se eligen funciones triangulares debido a que representa una menor carga de procesamiento para el dsPIC y esto ayudara a que el tiempo de respuesta del microcontrolador sea más rápido. Los conjuntos difusos que se implementaron se muestra en las siguientes figuras 4.8, 4.9 y 4.10.

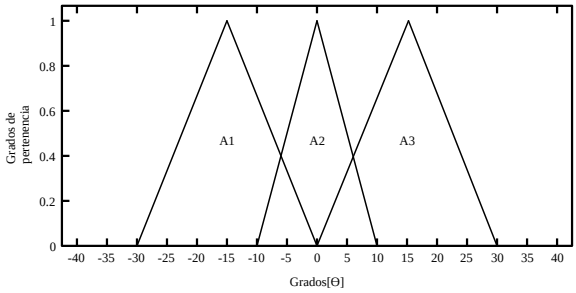

Figura 4.8: Conjuntos para el angulo del péndulo

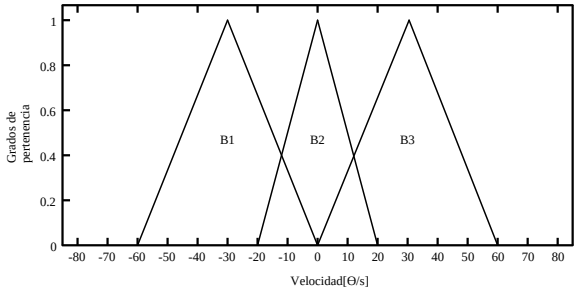

Figura 4.9: Conjuntos para la velocidad del péndulo

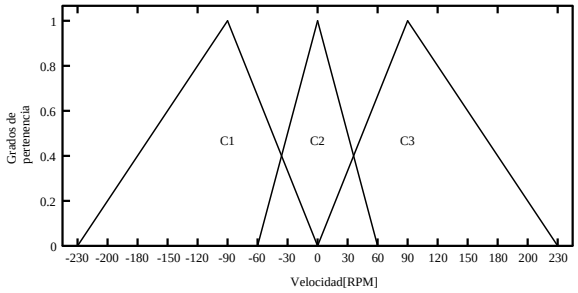

Figura 4.10: Conjuntos para la velocidad del carro

Los conjuntos difusos se delimitaron de acuerdo a las observaciones y pruebas que se realizaron con el sistema. Las funciones de membresía que representan a los conjuntos difusos son:

$$A_1 = \begin{cases} \frac{x+15}{-7,5+15} & si \quad -30 \leq x \leq 0 \\ \frac{0-x}{0+7,5} & si \quad 0 \leq x \leq -15 \\ 0 & si \quad -30 > x > -15 \end{cases}$$

$$A_2 = \begin{cases} \frac{x+5}{0+5} & si \quad -10 \leq x \leq 10 \\ \frac{5-x}{5-0} & si \quad 10 \leq x \leq 0 \\ 0 & si \quad -10 > x > 0 \end{cases}$$

$$A_3 = \begin{cases} \frac{x-0}{7,5-0} & si \quad 0 \leq x \leq 30 \\ \frac{15-x}{15-7,5} & si \quad 30 \leq x \leq 0 \\ 0 & si \quad 0 > x > 15 \end{cases}$$

$$B_1 = \begin{cases} \frac{x+60}{-30+60} & si \quad -60 \leq x \leq 0 \\ \frac{0-x}{0+30} & si \quad 0 \leq x \leq -30 \\ 0 & si \quad -60 > x > -30 \end{cases}$$

$$B_2 = \begin{cases} \frac{x+20}{0+20} & si \quad -20 \leq x \leq 20 \\ \frac{20-x}{20+0} & si \quad 20 \leq x \leq 0 \\ 0 & si \quad -20 > x > 0 \end{cases}$$

$$B_3 = \begin{cases} \frac{x-0}{30-0} & si \quad 0 \leq x \leq 60 \\ \frac{60-x}{60-30} & si \quad 60 \leq x \leq 30 \\ 0 & si \quad 0 > x > 30 \end{cases}$$

$$C_1 = \begin{cases} \frac{x+240}{-90+240} & si \quad -230 \leq x \leq 0 \\ \frac{0-x}{0+90} & si \quad 0 \leq x \leq -90 \\ 0 & si \quad -230 > x > -90 \end{cases}$$

$$C_2 = \begin{cases} \frac{x+60}{0+60} & si \quad -60 \leq x \leq 60 \\ \frac{60-x}{60-0} & si \quad 60 \leq x \leq 0 \\ 0 & si \quad -60 > x > 0 \end{cases}$$

$$C_3 = \begin{cases} \frac{x-0}{90-0} & si \quad 0 \leq x \leq 230 \\ \frac{240-x}{90-240} & si \quad 230 \leq x \leq 90 \\ 0 & si \quad 0 > x > 90 \end{cases}$$

Para los conjuntos difusos $A1 - A3$ la variable x representa la posición del péndulo, para $B1 - B3$ representa la velocidad del péndulo y para $C1 - C3$ representa los revoluciones por minuto a las que gira el motor.

4.2.2 Reglas de conocimiento

Estas reglas de conocimiento representan todas las posibles combinaciones que hay entre las tres variables del sistema. Un ejemplo de estas reglas es:

$R1$:Si el ángulo es $A1$ y la velocidad del péndulo es $B1$ y la velocidad del carro es $C1$ entonces la salida es $D27$.

En las tablas 4.1 y 4.2 se muestra las 27 posibles reglas para el sistema:

Regla	Si			Entonces
$R1$	$A1$	$B1$	$C1$	$D1$
$R2$	$A1$	$B1$	$C2$	$D2$
$R3$	$A1$	$B1$	$C3$	$D3$
$R4$	$A1$	$B2$	$C1$	$D4$
$R5$	$A1$	$B2$	$C2$	$D5$
$R6$	$A1$	$B2$	$C3$	$D6$
$R7$	$A1$	$B3$	$C1$	$D7$
$R8$	$A1$	$B3$	$C2$	$D8$
$R9$	$A1$	$B3$	$C3$	$D9$
$R10$	$A2$	$B1$	$C1$	$D10$
$R11$	$A2$	$B1$	$C2$	$D11$
$R12$	$A2$	$B1$	$C3$	$D12$
$R13$	$A2$	$B2$	$C1$	$D13$

Cuadro 4.1: Reglas para conjuntos Difusos

Regla	Si			Entonces
$R14$	$A2$	$B2$	$C2$	$D14$
$R15$	$A2$	$B2$	$C3$	$D15$
$R16$	$A2$	$B3$	$C1$	$D16$
$R17$	$A2$	$B3$	$C2$	$D17$
$R18$	$A2$	$B3$	$C3$	$D18$
$R19$	$A3$	$B1$	$C1$	$D19$
$R20$	$A3$	$B1$	$C2$	$D20$
$R21$	$A3$	$B1$	$C3$	$D21$
$R22$	$A3$	$B2$	$C1$	$D22$
$R23$	$A3$	$B2$	$C2$	$D23$
$R24$	$A3$	$B2$	$C3$	$D24$
$R25$	$A3$	$B3$	$C1$	$D25$
$R26$	$A3$	$B3$	$C2$	$D26$
$R27$	$A3$	$B3$	$C3$	$D27$

Cuadro 4.2: Reglas para conjuntos Difusos

Es importante resaltar dos aspectos de los conjunto $D1 - D27$: el primero de ellos es que son de tipo singleton y el segundo es que los valores están dados en base al conocimiento que se tiene sobre cuanta velocidad necesita el motor para desplazar el carro, esta

velocidad depende de la desviación que tenga el péndulo, entre mayor sea la desviación respecto de su punto de equilibrio, mayor será el valor de D_i.

Para obtener la activación de cada regla se utiliza la Norma-t. A continuación se muestra un ejemplo:

$$R27 = A3 * B3 * C3$$

Este mismo proceso se realiza para las 27 reglas y una vez hecha esta etapa se procede a obtener la defuzzificación

4.2.2.1 Conjuntos difusos para la defuzzificación

Los conjuntos de tipo singleton son de gran ayuda en la etapa final de control, esto se debe a que la forma de definir dichos conjuntos para la defuzzificación es un procedimiento fácil de implementar, dado que los valores que toman estos conjuntos difusos, son de altura unitaria en un eje y es en el otro donde se producen los cambios, lo que facilita tanto la acumulación como la etapa para obtener el valor final o de salida, en la figura 4.11 se muestra como se definió el conjunto difuso.

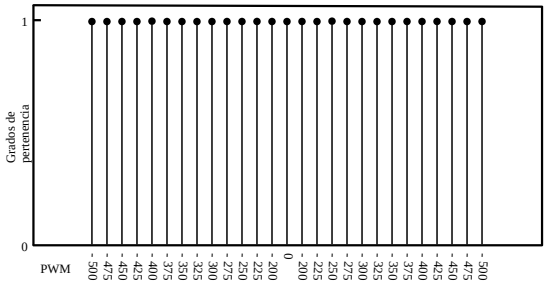

Figura 4.11: Conjuntos singleton

Para finalizar la defuzzificacion se realiza el promedio ponderado de los 27 elementos con la siguiente ecuación:

$$y_f = \frac{\sum_{i=1}^{27} Ri(\theta, \dot{\theta}, v) * (D_i)}{\sum_{i=1}^{27} Ri(\theta, \dot{\theta}, v)}$$

donde:

- y_f es valor del PWM que recibe el motor.
- $Ri(\theta, \dot{\theta}, v)$ es el peso de la regla,el cual depende del valor del angulo del péndulo, de la velocidad de este y de los RPM a los que gire el motor.

- D_i es uno de los 27 valores del conjunto singleton.

El rango en el cual varia D_i se debe a que el máximo ciclo de trabajo del PWM es 500 y por su parte 300 es el mínimo valor con el cual el motor logra que se desplace el sistema. El PWM será trasmitido del dsPIC a un puente H, el cual trasforma la señal digital a una analógica.

4.3 Implantación del controlador digital

Ya establecidos los conjuntos difusos se implementó el controlador digital, el cual fue diseñado en el software de mikroC para dsPIC (mikroC PRO for dsPIC). El código fuente del control difuso se encuentra en apéndice y en la figura 4.12 se muestra el diagrama principal, en el cual se configura tanto puertos como módulos necesarios para desarrollar el controlador.

Se estableció un periodo de muestreo de $5ms$, ya que se estimó que el péndulo se mueve dentro del intervalo de control en $100ms$ y suponiendo el peor de los casos se tienen 20 muestras en dicho intervalo. Por

lo mencionado anteriormente se hace necesario contar con una interrupción y es aquí donde se desarrolla el programa principal, ya que se realiza tanto el llamado a funciones como el cálculo de la posición del motor, en el cual se acede al registro que lleva el conteo de pulsos del encoder, posteriormente se transforma dicho valor para poder ser utilizado en el cálculo de la velocidad del motor.

Otra función obtiene el grado de pertenecía de las entrada, donde dependiendo del valor de la variable se realiza el cálculo, esto con base a las funciones que describen a un conjunto difuso triangular. Para adquirir el valor de las reglas se utiliza la Norma-t, mostrada en el apartado reglas de conocimiento del subcapitulo 4.2 y a partir de la activación mencionada se hace el cálculo de cada una de las reglas.

La última función obtiene el promedio ponderado, donde a través de un ciclo for se realiza la suma de la multiplicación de cada una de las reglas por el valor correspondiente del conjunto singleton y la suma de todas las reglas, para posteriormente dividir la primera suma con la segunda y así obtener el promedio o valor de y. El diagrama de la interrupción se muestra en las figuras 4.13 y 4.14.

Es necesario el uso del módulo UART para establecer la comunicacion con la Raspberry, para lo cual se realizó una interrupción. Dicho evento ocurre cuando se recibe un dato y posterior a esto los elementos son almacenados si al final de ellos encuentra un ok, después de esto se interrumpe la comunicación. Ya que se a realizado lo anterior se decide con base a los datos recibidos si se envía la posición del péndulo, la velocidad del motor o se asignan los elementos recibidos a las variables de los conjuntos difusos de entrada. Esté diagrama se muestra en las figuras 4.15 y 4.16

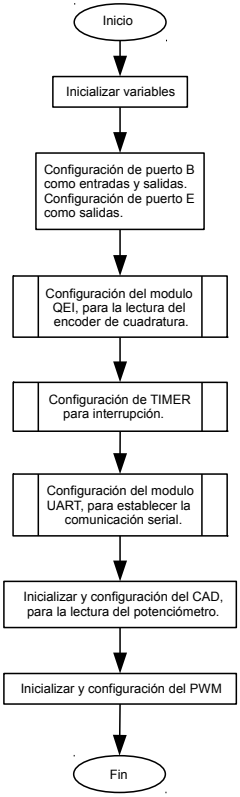

Figura 4.12: Diagrama Principal

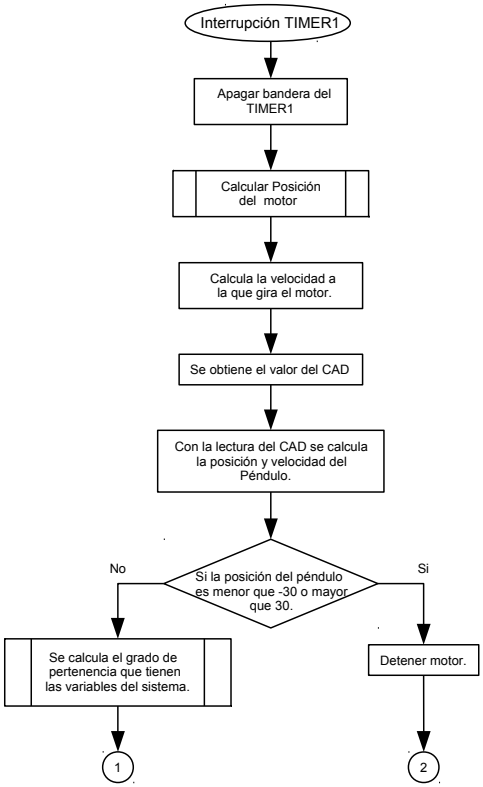

Figura 4.13: Diagrama de Interrupción TMER1 (a)

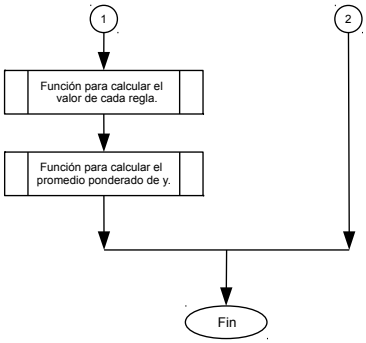

Figura 4.14: Diagrama de Interrupción TIMER1 (b)

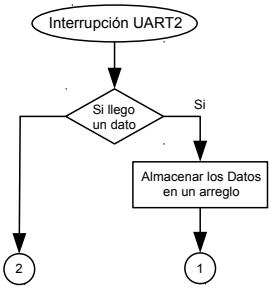

Figura 4.15: Diagrama de Interrupción UART2 (a)

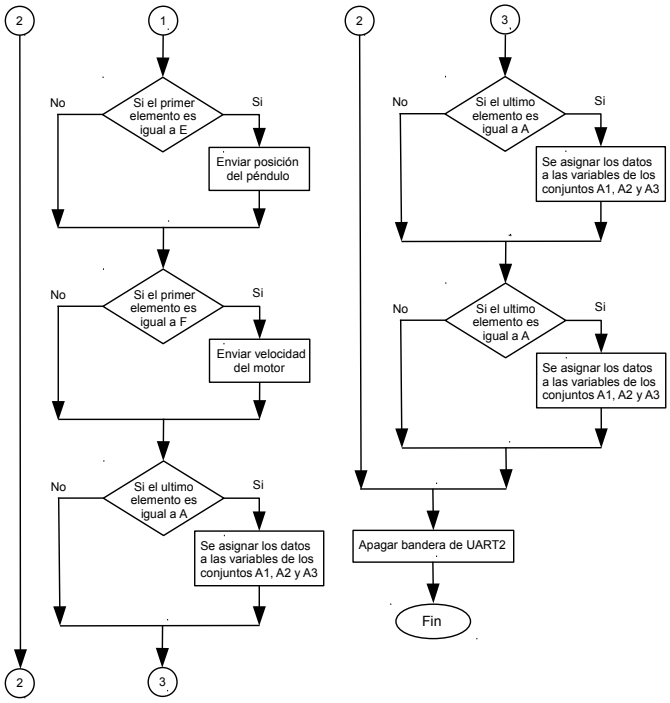

Figura 4.16: Diagrama de Interrupción UART2 (b)

Capítulo 5

Interfaz gráfica de usuario

En este capítulo se dará un breve acercamiento a la Raspberry Pi, resaltando las principales características de la microcomputadora. Posterior a esto se tendrá una pequeña introducción al lenguaje de programación Python y sus librerías, una vez ya explicado esto se pasara a la parte del diseño de la interfaz gráfica, en la cual se mostrará y se explicará el diagrama de flujo del programa desarrollado. En el apéndice B se encuentra el código fuente de la interfaz.

5.1 Rapberry Pi

La Raspberry Pi es una microcomputadora de bajo costo desarrollada en Inglaterra y en la actualidad existen varios, en la presente tesis se utilizó el modelo B + y en la figura 5.1 se muestra una imagen con algunas de sus características.

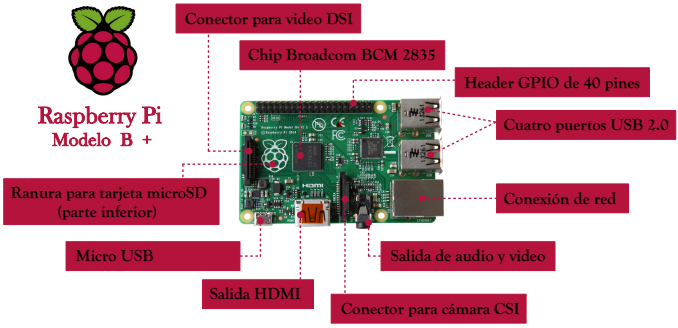

Figura 5.1: Raspberry Pi B +

El sistema está basado en un chip Broadcom BCM2835, el cual contiene un procesador central (CPU) ARM 1176JZF-S a 700 MHz (de la familia ARM11), un pro-

cesador gráfico (GPU) VideoCore IV y una memoria SDRAM de 512 MB (compartidos con el GPU). Como dispositivo de almacenamiento se utiliza una memoria microSD. El software puede ser descargado desde la página de Raspberry y en la siguiente lista se muestra las posibles distribuciones.

- RASPBIAN
- UBUNTU MATE
- SNAPPY UBUNTU CORE
- OPENELEC
- OSMC
- PIDORA
- RISC OS

Raspbian es la recomendada por la fundación, debido a que es una distribución derivada de Debian que está optimizada para la Raspberry Pi.

5.1.1 Lenguaje de Programacion Python

Python es un lenguaje de programación con sintaxis clara, multiplataforma, orientado a objetos, simple de aprender y gratuito. Por las características mencionadas y por ser un programa con el cual ya cuenta la distribución instalada en la Raspberry, se ópto por utilizar este lenguaje de programación para desarrollar la interfaz gráfica de usuario.

5.1.1.1 Librerías de Python

Una característica importante de Python es poder agregar módulos o extensiones para complementar su funcionalidad, lo cual resulta interesante cuando se desea ampliar las funciones de un programa, algunas de estas son bases de datos, arreglos, funciones matemáticas, gráficas, acceso a internet, fechas, tiempos, etc. Para desarrollar la interfaz se utilizaron algunas de estas, a continuación se explicarán brevemente.

tkinter: es una libreria que permite desarrollar una GUI (Graphic User Intefase), esta es una adaptación a el modulo gráfico Tcl/Tk de Python.

Existen otras librerías que cumplen con las mismas características como wxPython, PyQt o PySide y PyGTK, pero tkinter es una de las más utilizadas en Python.

matplotlib: esta es librería permite generar gráficas, gráficos de barras, histogramas, etc. Su forma de graficar y de programar se asemeja a lo hecho con matlab, tanto visualmente como en su exactitud. Matplotlib tiene una gran variedad de complementos, entre los que destaca la opción de graficar en 3D.

numpy : este paquete es una extensión de Python, el cual contiene una librería con funciones matemáticas de alto nivel, que tiene un arreglo n-dimensiones, sofisticadas funciones, la capacidad de entregar números aleatorios y hasta transformada de fourier entre otras cosas.

pyserial: es una extensión de Python, que permite tanto el acceso al puerto serial como a sus propiedades.

time: este módulo me permite acceder a varias fun-

ciones relacionadas con el tiempo, como retardos, tiempos de espera, fecha, hora, etc.

5.2 Diseño de la interfaz gráfica de usuario

La interfaz gráfica tiene dos principales objetivos, mostrar las gráficas de las variables del sistema y poder modificar los conjuntos difusos de entrada del controlador digital, esto último con la finalidad de evitar reprogramar el dsPIC cada vez que se desee reajustar los limites de los conjuntos. Solo se podrá graficar una señal a la vez dado que al hacer esto se consumen gran parte de los recursos de la Raspberry limitando a que se realice cualquier otra acción. En la siguiente figura 5.2 se muestra un diagrama de flujo del programa que se desarrolló para la interfaz gráfica y en el apéndice B se encuentra el código fuente.

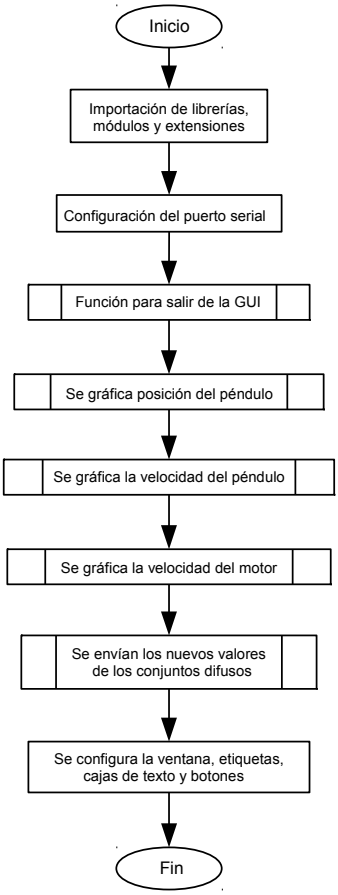

Figura 5.2: Diagrama de flujo de la interfaz gráfica

5.3 Funcionamiento de la interfaz gráfica de usuario

La interfaz cuenta con tres botones para graficar y al seleccionar uno, se activa cualquiera de las funciones que realizar dicho proceso, en las figuras 5.3, 5.4 y 5.5 se muestra el resultado de los tres casos. Para realizar el procesos mencionado se inicia una comunicación serial sincrona entre el dsPIC y la Raspberry, donde los datos que recibe esta son almacenados en un arreglo para después ser graficados. Por otra parte se tiene una caja de texto en la cual se ingresan los valores del conjunto difuso que se desee modificar, el conjunto pude ser seleccionado a través de tres botones, dos para los conjuntos del péndulo y uno para la velocidad el motor, ya realizado esto es posible modificar los nuevos valores a través del puerto serial por medio del botón enviar, en la figura 5.6 se muestra una imagen.

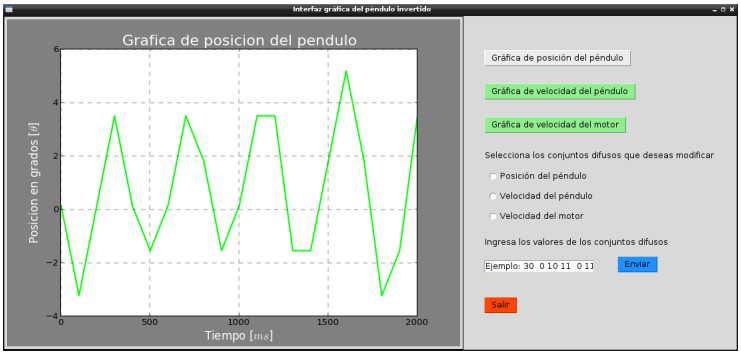

Figura 5.3: Gráfica de posición del péndulo

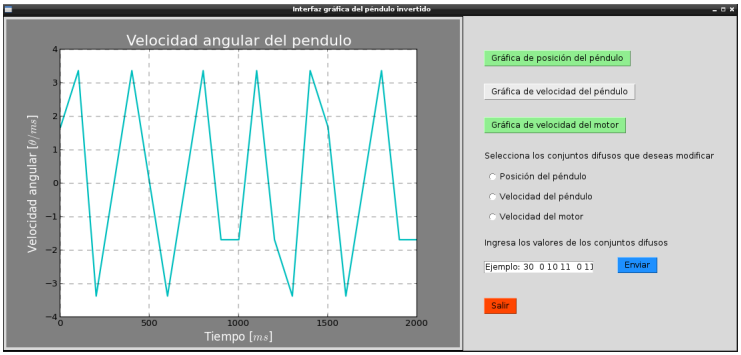

Figura 5.4: Gráfica de velocidad del péndulo

Figura 5.5: Gráfica de RPM del motor

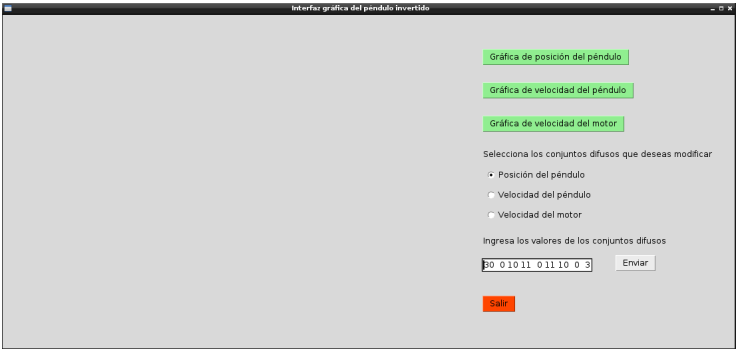

Figura 5.6: Envió de datos

Capítulo 6

Experimentos y Resultados

En este capítulo se explicará cada uno de los experimentos que se realizaron para sintonizar el controlador digital y observar las mejoras de cada una de estas pruebas se mostraran las gráficas correspondientes a cada una de las tres variable del sistema. Para cada experimento que se realizó, las reglas utilizadas son las mostradas en el capítulo 4 y los conjuntos difusos tanto de entrada como el de salida fueron modificados para cada uno. Algo que sería importante mencionar

es que para cada uno de las pruebas que se realizaron las gráficas obtenidas se hicieron independientemente una de otra dado que las interfaz solo puede graficar una señal a vez.

6.1 Experimento 1 y resultados

En las figuras 6.1, 6.2, 6.3 y 6.4 se muestran los conjuntos difusos que se utilizaron en este primer experimento. El rango de los conjuntos de velocidad del péndulo y del motor se acortaron, debido a que los valores asignados en el capítulo 4 fueron muy amplios, esto a consecuencia de que los valores que se obtuvieron anteriormente se adquirieron a través de la terminal USART de mikroC, para lo cual se fijó el sistema y se tratando de simular el posible comportamiento del péndulo. Otra consecuencia de lo mencionado fue que los valores del conjunto de tipo singleton fueron reajustados, ya que se observó que al darle un valor menor a 250 al PWM, el motor no contaba con el torque necesario para lograr desplazar al sistema.

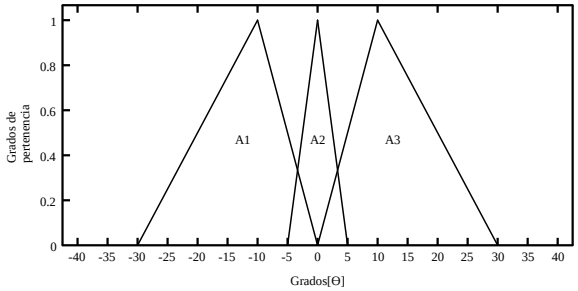

Figura 6.1: Conjuntos para el angulo del péndulo

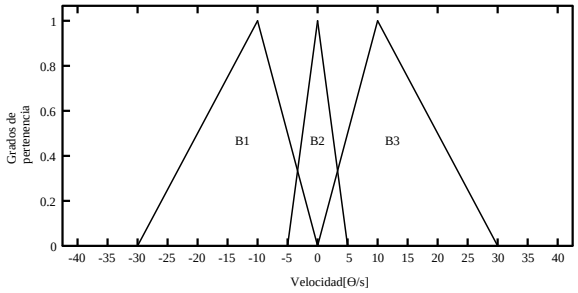

Figura 6.2: Conjuntos para la velocidad del péndulo

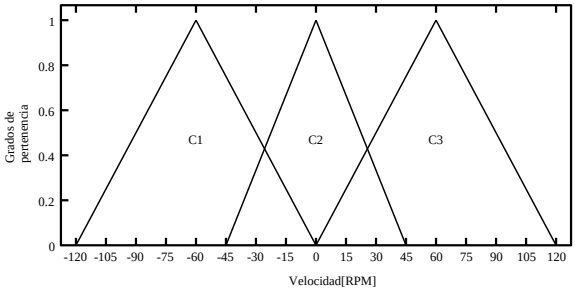

Figura 6.3: Conjuntos para la velocidad del carro

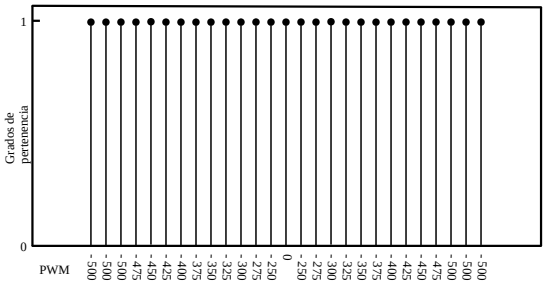

Figura 6.4: Conjuntos para el PWM

Se observó que al reducir el rango de los conjuntos se mejoró la reacción del péndulo, pero como resultado contrario a esto el sistema se vuelve más sensible, provocando que el péndulo oscile y no logre estabilizarse, como se puede ver en las gráficas de posición y velocidad que se muestran en las figuras 6.5 y 6.6 respectivamente. En la figura 6.7 se observa la gráfica del motor, en la cual la velocidad máxima es cercana a las 100 RPM, por lo que el rango de los conjuntos correspondientes a las revoluciones se deberán acortar y los conjuntos centrales correspondientes al péndulo se deberán ampliar.

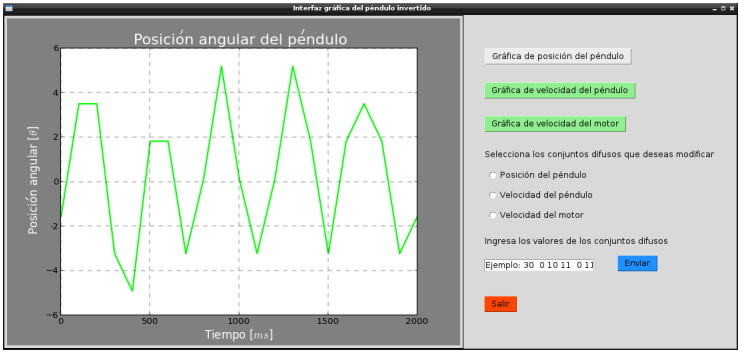

Figura 6.5: Gráfica de posición del péndulo

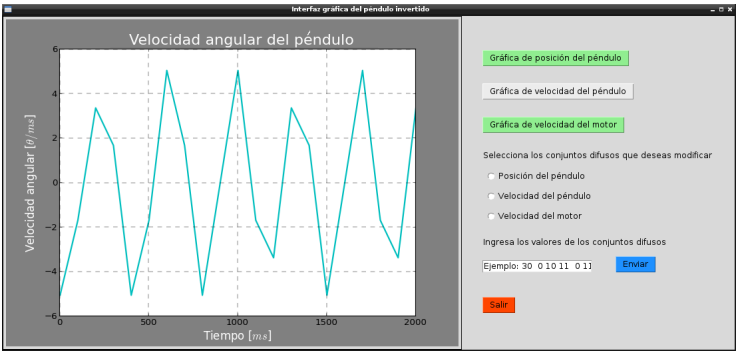

Figura 6.6: Gráfica de velocidad del péndulo

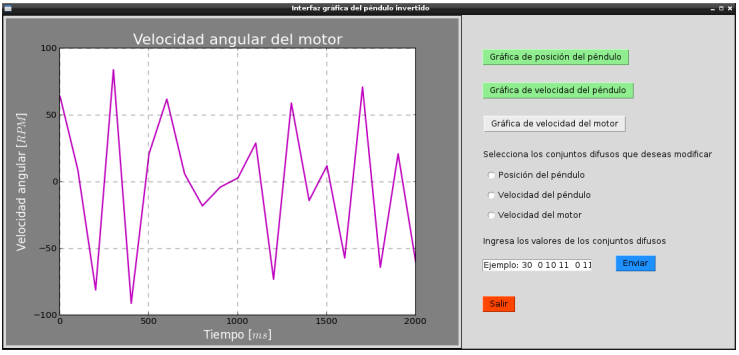

Figura 6.7: Gráfica de velocidad del motor

6.2 Experimento 2 y resultados

Al igual que en el experimento anterior se muestran los cuatro conjuntos difusos utilizados, los cuales se pueden ver en las figuras 6.8, 6.9, 6.10 y 6.11. En esta prueba se puede observar que los conjuntos centrales correspondiente a la posición y velocidad angular del péndulo se ampliaron. Como se mencionó anteriormente los conjuntos correspondientes a la velocidad del motor se disminuyeron y los valores del PWM se reacomodaron con la finalidad de mejorar la velocidad de motor.

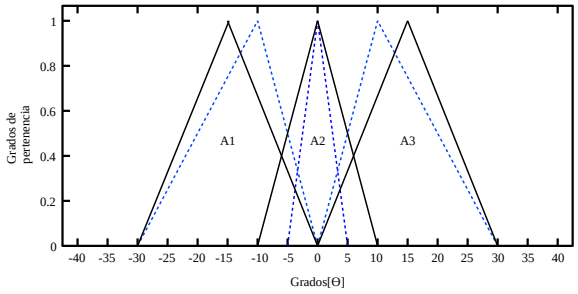

Figura 6.8: Conjuntos para el angulo del péndulo

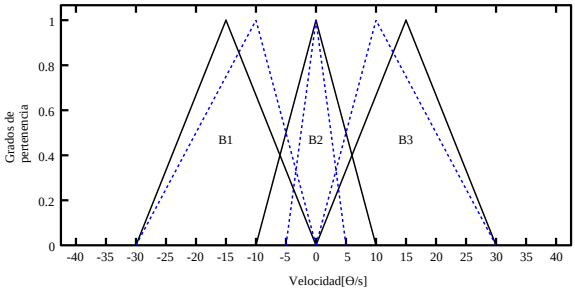

Figura 6.9: Conjuntos para la velocidad del péndulo

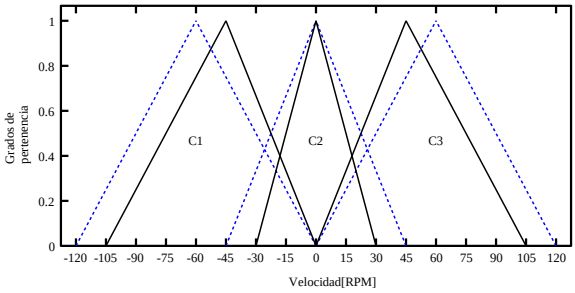

Figura 6.10: Conjuntos para la velocidad del carro

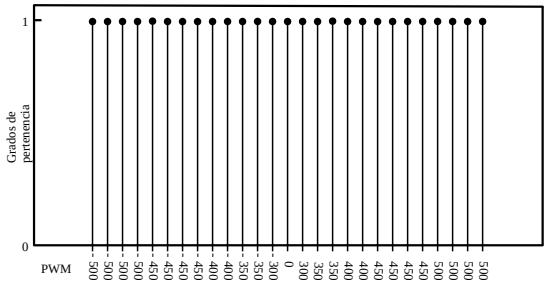

Figura 6.11: Conjuntos para la velocidad del carro

Como se puede ver en la figura 6.12 y 6.13, los límites de operación tanto de posición como de la velocidad del péndulo se disminuyeron, al igual que las revoluciones por minuto a las que gira el motor como se puede observar en la figura 6.14, lo que ayudo a que el sistema ya no tenga oscilaciones tan abruptas como en el experimento anterior, pero aun dado esto no se logra estabilizar, por lo cual se deberá seguir modificando los conjuntos difusos para tener una respuesta más suave y así el péndulo logre permanecer en equilibrio en su punto inestable.

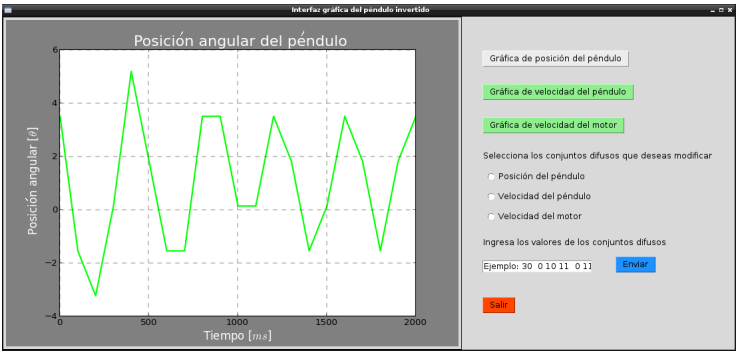

Figura 6.12: Gráfica de posición del péndulo

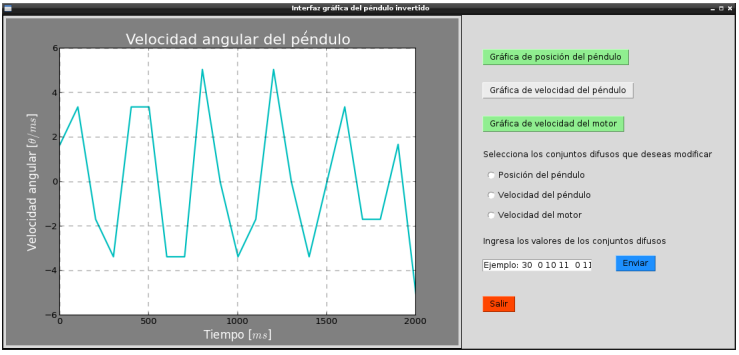

Figura 6.13: Gráfica de velocidad del péndulo

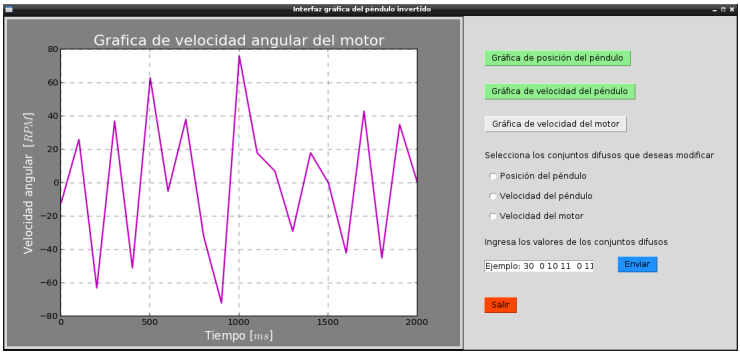

Figura 6.14: Gráfica de velocidad del motor

6.3 Experimento 3 y resultados

Después de realizar algunas pruebas se observó que al igualar el valor central de los triángulos laterales con el de los extremos del triángulo central se mejoraba la estabilización del péndulo, en la siguientes figuras 6.15, 6.16 y 6.17 se muestran los modificaciones realizadas, para el conjunto de salida que se muestra en la figura 6.18, también se realizaron algunos cambios con la finalidad de mejorar la velocidad del motor y con esto tener una respuesta más rápida del sistema.

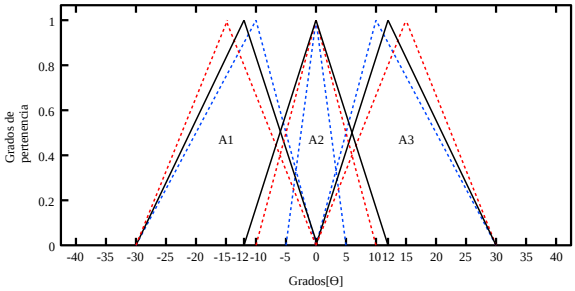

Figura 6.15: Conjuntos para el angulo del péndulo

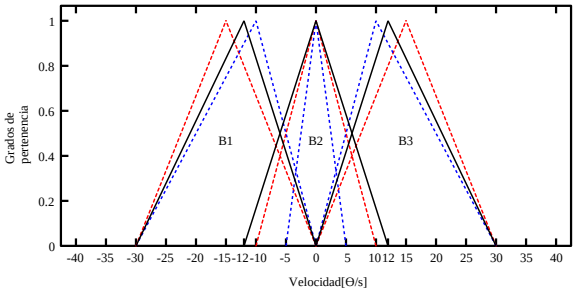

Figura 6.16: Conjuntos para la velocidad del péndulo

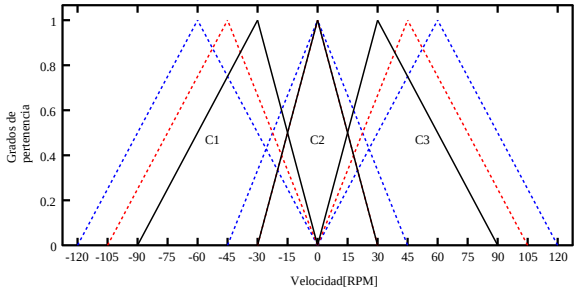

Figura 6.17: Conjuntos para la velocidad del carro

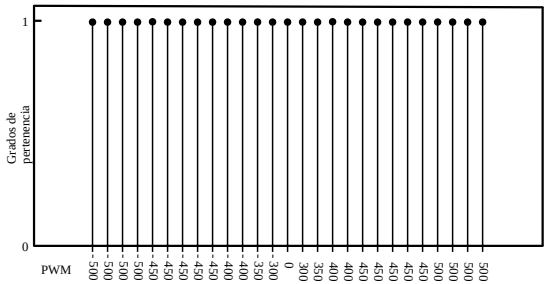

Figura 6.18: Conjuntos para la velocidad del carro

En la figura 6.19 y 6.20 se observa que las variables del péndulo se acotaron a valores cercanos a cero, lo que permitió tener un sistema más estable y por consiguiente se logra que se estabilice por alrededor de un segundo, pero después de esto comienza a oscilar y cae. Para la gráfica de la velocidad motor que se muestra en la figura 6.21 se puede ver que las RPM del motoreductor tiene un comportamiento similar a las variable anteriores, pero a diferencia de estas las oscilaciones siguen siendo muy fuertes, por lo cual se deberá buscar que el motor tenga una respuesta rápida pero con un torque menor.

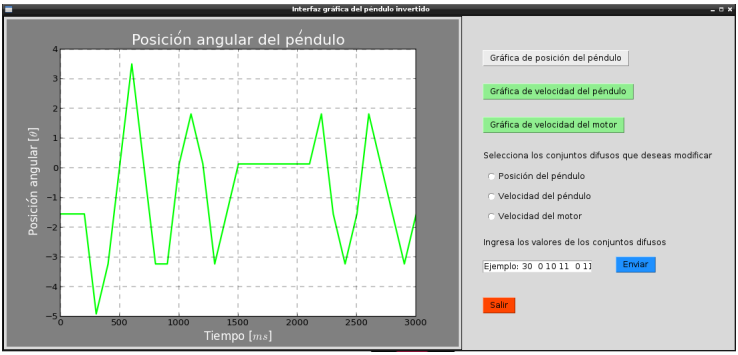

Figura 6.19: Gráfica de posición del péndulo

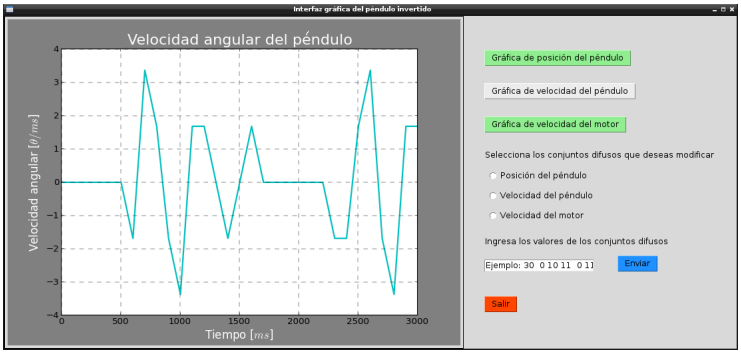

Figura 6.20: Gráfica de velocidad del péndulo

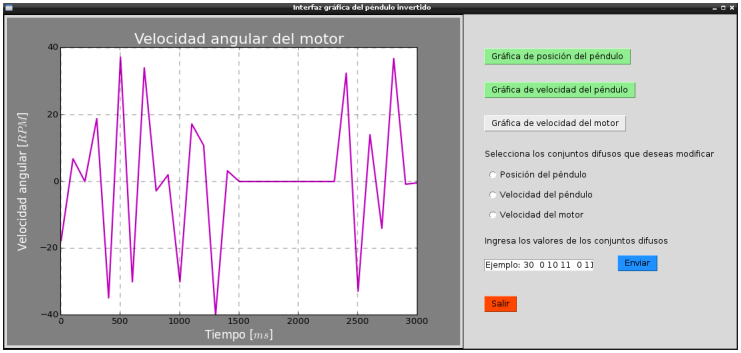

Figura 6.21: Gráfica de velocidad del motor

6.4 Experimento 4 y resultados

En este ultimó experimento se redujo el rango de los conjuntos con la finalidad de tener una respuesta más rápida en los ángulos cercanos a cero, pero tratando de mantener la misma relación de distancia que se aplicó en la prueba pasada con los conjuntos centrales y laterales, lo que mejoro la estabilidad al sistema, en las figuras 6.22, 6.23 y 6.24 se muestran los cambios realizados, para el conjunto de salida que se puede ver en la figura 6.25.

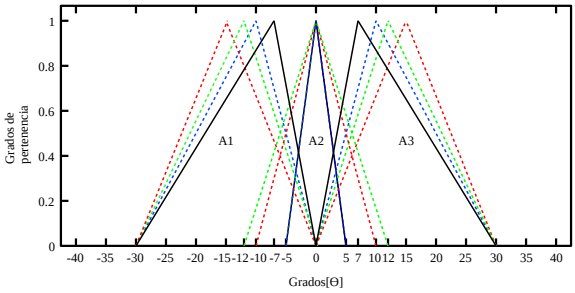

Figura 6.22: Conjuntos para el angulo del péndulo

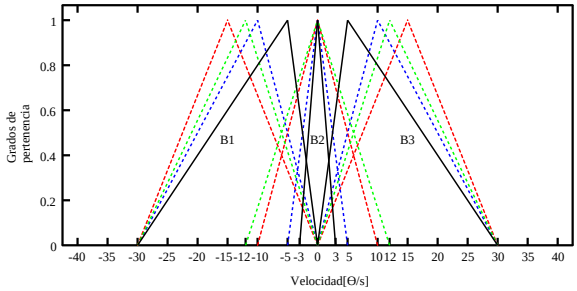

Figura 6.23: Conjuntos para la velocidad del péndulo

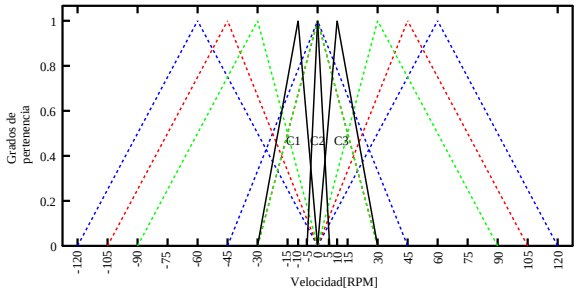

Figura 6.24: Conjuntos para la velocidad del carro

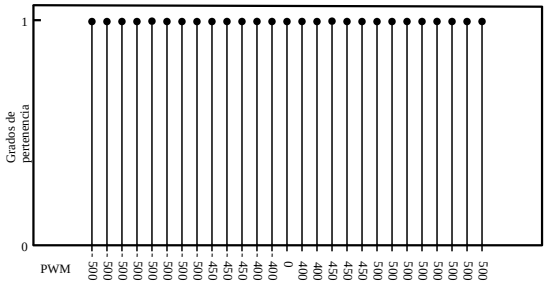

Figura 6.25: Conjuntos para la velocidad del carro

Como se observa en las figuras 6.26, 6.27 y 6.28 los cambios realizados ayudaron a tener un mejor comportamiento del sistema, logrando que el péndulo se mantenga casi estable por alrededor de cinco minuto y que por más de un minuto se mantenga en equilibrio completo. La gráfica de la velocidad del motor se mejoró por mucho en relación a los experimentos anteriores, dado que se puede apreciar una respuesta suave.

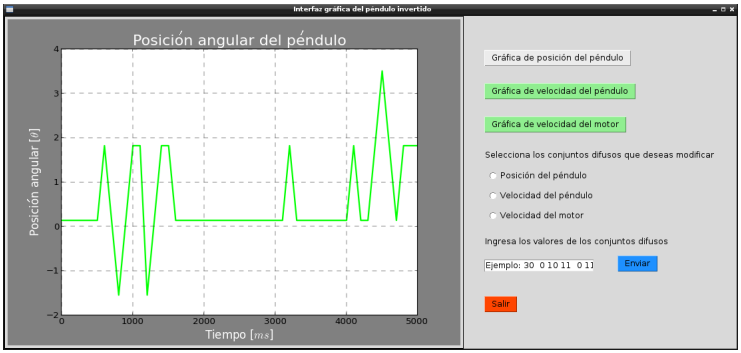

Figura 6.26: Gráfica de posición del péndulo

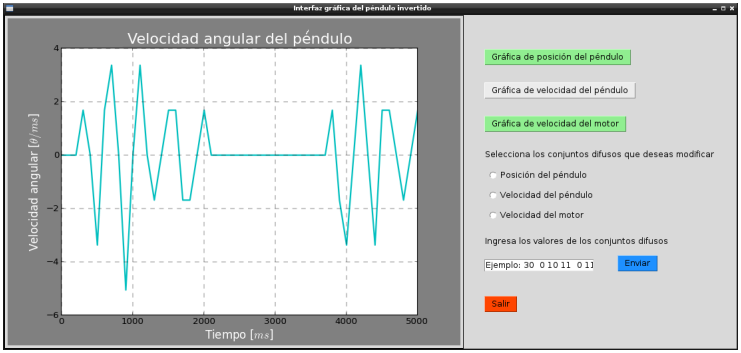

Figura 6.27: Gráfica de velocidad del péndulo

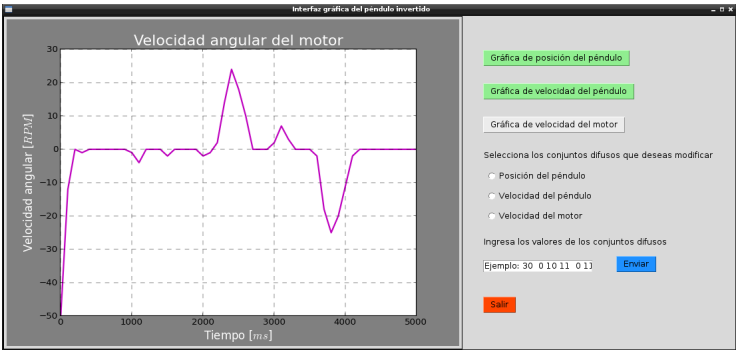

Figura 6.28: Gráfica de velocidad del motor

Capítulo 7

Conclusiones

7.1 Conclusiones

De acuerdo a las observaciones y experimentos realizados se puede concluir que el péndulo invertido tiene un desempeño satisfactorio y puede ser utilizado para la experimentación del control difuso propuesto. Debido a las características de flexibilidad con las que fue diseñado es posible utilizarlo para la implementación y prueba de otro tipo de controladores. Sin embargo es necesario realizar la programación completa de éstos

en el dsPIC y en la tarjeta Raspberry Pi desarrollar una interfaz para el ajuste de parámetros de los nuevos controladores.

Se obtuvo un sistema flexible ya que es posible modificar las características del péndulo para realizar diferentes pruebas, cambiar el péndulo por uno de mayor o menor tamaño, agregar peso al sistema, etc. También es posible modificar las características electromecánicas del sistema con cierta facilidad. Cada uno de los subsistemas cumple adecuadamente con su objetivo, sin embargo algunos de éstos presentan deficiencias en su desempeño que pueden ser mejoradas como trabajo futuro.

El controlador digital de señales dsPIC cuenta con los periféricos y la capacidad de procesamiento adecuado para realizar la adquisición y procesamiento de los datos con una frecuencia de muestreo de 5 ms. El módulo de lectura del encoder QEI permite realizar la lectura de velocidad del motor con una alta resolución de 0.1875 grados por bit y al ser un módulo que trabaja de manera independiente al procesador no consume recursos de procesamiento del microcontrolador.

El sistema de sensado de posición y velocidad angular del péndulo que está basado en el potenciómetro

de precisión que permite estimar de forma adecuada la posición de este, valor que es adquirido por el convertidor analógico digital del dsPIC con una resolución 4.887 mV por bit(0,351^o). Una desventaja que presenta este dispositivo es el tipo de acoplamiento mecánico con el eje de rotación del péndulo, el cual es muy sensible a los desplazamientos que pueda sufrir el eje del potenciómetro con respecto al eje del péndulo lo que provoca que las lecturas de posición se modifiquen y sea necesario volver a calibrarlo.

Por su parte el motoreductor que mueve el carro con los 8 Nm de torque no mostró dificultades para mover la planta y el encoder con los 480 conteos por revolución permitió tener una lectura adecuada de la velocidad a la que se desplazaba el sistema y por último el motor mostró una deficiencia mínima al tener una zona muerta (back slash) relativamente amplia que afecta la respuesta del motor en los cambios de giro, esto sabiendo que este tipo de sistemas deben contar con un tiempo de respuesta lo menor posible.

La batería no logro cumplir con los requerimientos necesarios, dado que en algunas circunstancias como un cambio de giro muy abrupto no soporta los picos de corriente del motor y esta se apaga, por lo que es

necesario sustituir la batería por otra que cumpla con los requerimientos de corriente.

La tarjeta Raspberry Pi mostró ser eficiente para implementar pequeños proyectos y tener la flexibilidad de desarrollarlos en diferentes plataformas de software libre, lo cual resulta interesante ya que da una amplia posibilidad de adecuarse a las necesidades del programador. La ventaja de tener una computadora embebida en el proyecto se reflejan en el desarrollo completo la una interfaz de usuario (python) sin la necesidad utilizar algún otro programa comercial, esta interfaz es básica pero permite observar el comportamiento de las variables del sistema y sería posible realizar interfaces más complejas lo cual queda como trabajo futuro.

La tarjeta raspberry también da la posibilidad de controlar y monitorear al sistema de forma remota a través de internet, en este caso sólo se monitorear, lo cual da muchas ventajas a un sistema didáctico ya que es posible ver el resultado en diferentes computadoras a la vez sin necesidad de tener un cable conectado a él. Una de las desventajas es el manejo de gráficos de la tarjeta raspberry ya que por algunos momentos consumen todos los recursos del sistema con lo cual se vuelve lenta y en ocasiones deja de funcionar.

La técnica de control difuso elegida logro estabilizar al péndulo por algunos momentos en su posición deseada o muy cerca de esta como se muestra en las gráficas de cada una de las pruebas, con lo cual se puede decir que este controlador es una opción adecuada cundo se desea controlar sistemas, que dadas la circunstancia de su construcción es complicado obtener un modelo matemático adecuado de la planta y se seleccionó el modelo singleton por la adecuación de sus posibles valor con el PWM. Se pude decir que este controlador difuso es relativamente fácil de implementar, pero muestra complicaciones cuando se trata sintonizar y más aún cuando alguna de las variables o elementos sufre alguna pequeña variación. Se observó que al elegir funciones triangulares el sistema mostro movimientos o cambios abruptos en algunos casos, con lo cual sería adecuado como trabajo a futuro cambiar estas funciones por otras que lograran que el sistema tenga una respuesta más suave.

7.1.1 Trabajo a futuro

- Es posible mejorar la etapa mecánica teniendo elementos más precisos o de mejor calidad.

- Seria conveniente sustituir el potenciómetro por un sensor que permitiera al péndulo un libre desplazamiento y no fuese necesario fijarlo, dado que esto complicó la sintonización del controlador.
- El motoreductor puede ser cambiado por otro que tenga una respuesta más rápida y con un back slash menor, lo cual ayudaría a tener una respuesta más precisa y rápida de la planta.
- Es preciso sustituir la batería por una que tenga mayor capacidad de corriente y que tenga un tiempo duración mayor.
- La interfaz gráfica puede ser mejorada sustituyendo la Raspberry Pi por un modelo más reciente.
- Realizar nuevos controladores digitales como neorodifuso, PID, LQR.

Capítulo 8

Apéndice

8.1 Código del controlador digital difuso

```
#define MAX_CNT_PER_REV (480 * 4 - 1)
#define MAXSPEED (unsigned int)
(((unsigned long)MAX_CNT_PER_REV*2048)/120)
#define HALFMAXSPEED (MAXSPEED>>1)

void  InitUART2(void);
```

```
void  InitTMR1(void);
void  interrupt_tmr1(void);
void  InitQEI(void);
void  CalcularPosicion(void);
float funcion_grado_de_pertenencia_izq(int a,
      int b,int c,float x);
float funcion_grado_de_pertenencia_cen(int a,
      int b,int c,float x);
float funcion_grado_de_pertenencia_der(int a,
      int b,int c,float x);
void  funcion_para_calcular_el_valor_de_cada_
      regla(void);
void  funcion_para_y(void);

float A_1[4]={0},B_1[4]={0},C_1[4]={0},R[28]
      ={0},velocidad_motor,velocidad_pendulo,
      pos_leida_v,pos_leida_teta[2]={0},
      Ang_Mot,POSCNTcopy,AngPos[2],
      Speed,pos_leida,Promedio_ponderado_y;

char output[30]={0},Var_difusa[2]={0},
     A1_a[4]={0},A1_b[4]={0},A1_c[4]={0},
     A2_a[4]={0},A2_b[4]={0},A2_c[4]={0},
     A3_a[4]={0},A3_b[4]={0},A3_c[4]={0},
```

```
     B1_a[4]={0},B1_b[4]={0},B1_c[4]={0},
     B2_a[4]={0},B2_b[4]={0},B2_c[4]={0},
     B3_a[4]={0},B3_b[4]={0},B3_c[4]={0},
     C1_a[4]={0},C1_b[4]={0},C1_c[4]={0},
     C2_a[4]={0},C2_b[4]={0},C2_c[4]={0},
     C3_a[4]={0},C3_b[4]={0},C3_c[4]={0};

int  result_0=0,result_1=0,result_2=0,
     result_3=0,result_4=0,result_5=0,
     result_6=0,result_7=0,result_8=0;

unsigned short int pos_pendulo= 0,
                   pos_motor = 0;

void main() {

 ADPCFG=0x01FE;  //El pin RB0 como
                 //analógico y RB1...RB8
                 //como digitales
 TRISB=0x0001;   //El pin RB0 como
                 //entrada y RB1...RB8
                 //como salidas
 TRISE = 0x0100; //Los pines PWM como
                 //salidas y FLTA como
```

```
                //entrada
 InitUART2();
 InitQEI();
 InitTMR1();
 ADC1_Init();

 PWM1_MC_Init(20000,1,0x01,0); //Inicializa
 //el modulo PWM1 a, 2kHZ en modo
 //independiente, habilita pin0 del PWM1 como
 //salida y sin reloj de prescala, sin reloj
 //de poscala
 PWM1_MC_Set_Duty(300,1); //ciclo de trabajo
                          //del PWM1 a 300
 PWM1_MC_Start(); //Se activa el PWM1

   do{


   }while(1);

}
/***Función para configurar el USART 2****/
void InitUART2(void)
{
```

```
UART2_Init(9600); //Se inicialoza el USART
                  //38400 bps
Delay_ms(100); //Tiempo de espera para
               //que se estabilice
               //el modulo USART2
IPC6bits.U2TXIP = 0x00; //Nivel de prioridad
                  //0 para la interrupcion
                  //de recepcion del USART 2
U2STAbits.URXISEL0 = 0; //Bits para configurar
                  //la interrupción del USART2
                  //cuando este reciba un dato
U2STAbits.URXISEL1 = 0; //Bits para configurar
                  //la interrupción del USART2
                  //cuando este reciba un dato
IFS1bits.U2RXIF = 0; //Pone a cero la bandera
                //de interrupción de UART_RX 2
IEC1bits.U2RXIE = 1; //Se habilita las
         //interrupciones  de Enable UART_RX 2
}

/**Función para configurar la interrupcion
  del TIMER 1**/
void InitTMR1(void)
{
```

```
TMR1 = 0; //Resetea el contador del TIMER 1
T1CONbits.TON = 0; //Apaga el Timer 1
T1CONbits.TSIDL = 0; //Continuar con la
                      //operacion en
                      //el modo sleep
T1CONbits.TGATE = 0; //Actiba la acomulacion
                     //de tiempo del TIMER1
T1CONbits.TCS = 0; //Usar Tcy como fuente
                   //de reloj
T1CONbits.TCKPS = 1;  //Tcy / 8 como reloj
                      //de entrada
PR1 = 3125; //Periodo de interrupcion
 //de = 0.075 segundos con una prescala de 8
IPC0bits.T1IP = 0X01; //Nivel de prioridad 1
//para la interrupción del TIMER 1
IFS0bits.T1IF = 0; //Pone a cero la bandera de
                   //interrupción del TIMER 1
IEC0bits.T1IE = 1; //Activa la interrupción
                   //para el TIMER 1
T1CONbits.TON = 1; //Activa el TIMER 1
return;
}

/****Función para configurar el modulo QEI****/
```

```
void InitQEI(void)
{
QEICONbits.QEIM = 0; //Desabilita el modulo QEI
QEICONbits.CNTERR = 0; //Elimina cualquier error
                        //de conteo
QEICONbits.QEISIDL = 0; //Continuar la operacion
                         //en modo de espera
QEICONbits.SWPAB = 0; //no intercambiar QEA y QEB
QEICONbits.UPDN_SRC = 1; //El pin de estado QEB esta
//definido como la direccion del contador de posicion
QEICONbits.PCDOUT = 0; //Operacion normal del pin
                        //como E/S
QEICONbits.POSRES = 0; //no Resetea el contador de
                        //posicion pulso de index
DFLTCONbits.CEID = 1; //Desabilita las interrupciones
                       //por error de conteo
DFLTCONbits.QEOUT = 1;//Abilita los filtros digitales
                       //de salida para los QEn pines
DFLTCONbits.QECK = 7; //divide el reloj del filtro
                       //digital de QEn con 1:64
QEICON.F8 = 1;//En modo X4 con reset por
              //igualación con el
              //contador de posición
QEICON.F9 = 1; //En modo X4 con reset por
```

```
                //igualación con el
                //contador de posición
QEICON.F10 = 1; //En modo X4 con reset por
                 //igualación con el
                 //contador de posición
MAXCNT = 1920; //Se le asigna 1920 al
                //MAXCNT porque es el máximo
                //conteo por revolución
POSCNT = 0; //Se pone a cero el contador
            //de posición
}

/**Función para calcular la posicion del motor**/
void CalcularPosicion(void)
{
POSCNTcopy = (int)POSCNT;
if (POSCNTcopy < 0)
POSCNTcopy = -POSCNTcopy;
AngPos[1] = AngPos[0];
AngPos[0] = (unsigned int)(((unsigned long)
              POSCNTcopy * 2048)/120);
            // 0 <= POSCNT <= 1920 to 0 <= AngPos
            // <= 32768
Ang_Mot =  ((AngPos[0])*(unsigned long)(360)
```

```
            /(32768));
}

/********Interrupción TIMER1********/
void interrupt_tmr1() org 0x1A
{
IFS0bits.T1IF = 0; //La bandera del TIMER1 a cero

CalcularPosicion();

Speed = AngPos[0] - AngPos[1];

if (Speed >= 0){
    if (Speed >= (HALFMAXSPEED))
    Speed = Speed - MAXSPEED;
}
else{
if (Speed < -(HALFMAXSPEED))
    Speed = Speed + MAXSPEED;
}
Speed *= 2;
velocidad_motor=((Speed)*(signed long)(400))/
                (32768);
pos_pendulo = ADC1_Read(0) >> 2;
```

```
pos_motor = (int)POSCNT;
pos_leida = ADC1_Read(0);
pos_leida_v = pos_leida*0.004887585533;
//resolucion 5v/(2^10 - 1)=
//4.88e-3, ADC de 10 bits
pos_leida_teta[1] = pos_leida_teta[0];
pos_leida_teta[0] = 85.8851*pos_leida_v
                    - 228.8847;
velocidad_pendulo = pos_leida_teta[0] -
                    pos_leida_teta[1];

if(pos_leida_teta[0]<-30 ||
   pos_leida_teta[0]>30){

      PORTB.RB8 = 0x00;
      PORTB.RB7 = 0x00;

      Promedio_ponderado_y = 0;
      PWM1_MC_Set_Duty(Promedio_
      ponderado_y, 1);

}else{
 A_1[0] = funcion_grado_de_pertenencia(A1_a,
 A1_b,A1_c,pos_leida_teta[0]);
```

```
A_1[1] =  funcion_grado_de_pertenencia(A2_a,
A2_b,A2_c,pos_leida_teta[0]);
A_1[2] =  funcion_grado_de_pertenencia(A3_a,
A3_b,A3_c,pos_leida_teta[0]);
B_1[0] =  funcion_grado_de_pertenencia(B1_a;
B1_b,B1_c,velocidad_pendulo);
B_1[1] =  funcion_grado_de_pertenencia(B2_a,
B2_b,B2_c,velocidad_pendulo);
B_1[2] =  funcion_grado_de_pertenencia(B3_a,
B3_b,B3_c,velocidad_pendulo);
C_1[0] =  funcion_grado_de_pertenencia(C1_a,
C1_b,C1_c,velocidad_motor);
C_1[1] =  funcion_grado_de_pertenencia(C2_a,
C2_b,C2_c,velocidad_motor);
C_1[2] =  funcion_grado_de_pertenencia(C3_a,
C3_b,C3_c,velocidad_motor);

funcion_para_calcular_el_valor_de_cada_regla();
funcion_para_y();

     if( Promedio_ponderado_y < 0){
         PORTB.RB8 = 0;
         PORTB.RB7 = 1;
     }
```

```
      if( Promedio_ponderado_y > 0){
          PORTB.RB8 = 1;
          PORTB.RB7 = 0;
      }
      if( Promedio_ponderado_y == 0){
          PORTB.RB8 = 0;
          PORTB.RB7 = 0;
      }

 PWM1_MC_Set_Duty(abs(Promedio_ponderado_y), 1);

}

}


/**Función para calcular el grado de pertenencia**/
float funcion_grado_de_pertenencia(int a, int b,
      int c, float x)
{
float grado_pertenencia;

   if(a <= x && x <= c)
   grado_pertenencia = (x - a)/(c - a);
```

```
   if(c <= x && x <= b)
   grado_pertenencia = (b - x) / (b - c);
   if( a > x || x > b)
   grado_pertenencia = 0;

   return grado_pertenencia;
}


/**Función para obtener el valor de las reglas**/
void funcion_para_calcular_el_valor_de_cada_
     regla(void)
{
R[0] = A_1[0]*B_1[0]*C_1[0];
R[1] = A_1[0]*B_1[0]*C_1[1];
R[2] = A_1[0]*B_1[0]*C_1[2];
R[3] = A_1[0]*B_1[1]*C_1[0];
R[4] = A_1[0]*B_1[1]*C_1[1];
R[5] = A_1[0]*B_1[1]*C_1[2];
R[6] = A_1[0]*B_1[2]*C_1[0];
R[7] = A_1[0]*B_1[2]*C_1[1];
R[8] = A_1[0]*B_1[2]*C_1[2];
R[9] = A_1[1]*B_1[0]*C_1[0];
R[10] = A_1[1]*B_1[0]*C_1[1];
```

```
R[11] = A_1[1]*B_1[0]*C_1[2];
R[12] = A_1[1]*B_1[1]*C_1[0];
R[13] = A_1[1]*B_1[1]*C_1[1];
R[14] = A_1[1]*B_1[1]*C_1[2];
R[15] = A_1[1]*B_1[2]*C_1[0];
R[16] = A_1[1]*B_1[2]*C_1[1];
R[17] = A_1[1]*B_1[2]*C_1[2];
R[18] = A_1[2]*B_1[0]*C_1[0];
R[19] = A_1[2]*B_1[0]*C_1[1];
R[20] = A_1[2]*B_1[0]*C_1[2];
R[21] = A_1[2]*B_1[1]*C_1[0];
R[22] = A_1[2]*B_1[1]*C_1[1];
R[23] = A_1[2]*B_1[1]*C_1[2];
R[24] = A_1[2]*B_1[2]*C_1[0];
R[25] = A_1[2]*B_1[2]*C_1[1];
R[26] = A_1[2]*B_1[2]*C_1[2];
}

/*Función para calcular el promedio ponderado de y*/
void funcion_para_y(void)
{
int    i;
float  D[27] = {-500,-500,-500,-500,-450,-450,
               -450,-450,-450,-450,-400,-350,
```

```
                -300,0,300,350,400,450,450,450,
                450,450,450,500,500,500,500};

float   suma[3];

for(i=0;i<=26;i++){

    suma[0] = suma[0] + R[i]*D[i];
    suma[1] = suma[1] + R[i];

}

 Promedio_ponderado_y = (suma[0])/(suma[1]);

}

/**Interrupcion USART**/
void interrupt_uart2() org 0x44
{

if (UART2_Data_Ready() == 1){ //Si
                  //Si UART2_Data_Ready()==1
                  //entonces llego un dato
```

```
UART2_Read_Text(output,"ok",30); //Si en
  //cualquiera de los 30 elementos se
  //recibe un ok entonces los elementos
  //recibidos se almacenan en la cadena
  //output
  if(output[0] == 'E'){

      UART2_Write(pos_pendulo);

  }
  if(output[0] == 'F'){

      UART2_Write(pos_motor);

  }

  if(output[27] == 'A'){ //Conjuntos difusos
                 //para posición del péndulo

                       //Conjunto izquierda
     A1_a[0] = output[0];
     A1_a[1] = output[1];
     A1_a[2] = output[2];
     A1_b[0] = output[3];
```

```
A1_b[1] = output[4];
A1_b[2] = output[5];
A1_c[0] = output[6];
A1_c[1] = output[7];
A1_c[2] = output[8];
                    //Conjunto centro
A2_a[0] = output[9];
A2_a[1] = output[10];
A2_a[2] = output[11];
A2_b[0] = output[12];
A2_b[1] = output[13];
A2_b[2] = output[14];
A2_c[0] = output[15];
A2_c[1] = output[16];
A2_c[2] = output[17];
                    //Conjunto deracha
A3_a[0] = output[18];
A3_a[1] = output[19];
A3_a[2] = output[20];
A3_b[0] = output[21];
A3_b[1] = output[22];
A3_b[2] = output[23];
A3_c[0] = output[24];
A3_c[1] = output[25];
```

```
    A3_c[2] = output[26];

    result_0 = atol(A1_a);
    result_1 = atol(A1_b);
    result_2 = atol(A1_c);
    result_3 = atol(A2_a);
    result_4 = atol(A2_b);
    result_5 = atol(A2_c);
    result_6 = atol(A3_a);
    result_7 = atol(A3_b);
    result_8 = atol(A3_c);

}

if(output[27] == 'B'){ //Conjuntos
//difusos para velocidad del péndulo

                       //Conjunto izquierda
    B1_a[0] = output[0];
    B1_a[1] = output[1];
    B1_a[2] = output[2];
    B1_b[0] = output[3];
    B1_b[1] = output[4];
    B1_b[2] = output[4];
```

```
B1_c[0] = output[6];
B1_c[1] = output[7];
B1_c[2] = output[8];
                  //Conjunto centro
B2_a[0] = output[9];
B2_a[1] = output[10];
B2_a[2] = output[11];
B2_b[0] = output[12];
B2_b[1] = output[13];
B2_b[2] = output[14];
B2_c[0] = output[15];
B2_c[1] = output[16];
B2_c[2] = output[17];
                  //Conjunto deracha
B3_a[0] = output[18];
B3_a[1] = output[19];
B3_a[2] = output[20];
B3_b[0] = output[21];
B3_b[1] = output[22];
B3_b[2] = output[23];
B3_c[0] = output[24];
B3_c[1] = output[25];
B3_c[2] = output[26];
```

```
    result_0 = atol(B1_a);
    result_1 = atol(B1_b);
    result_2 = atol(B1_c);
    result_3 = atol(B2_a);
    result_4 = atol(B2_b);
    result_5 = atol(B2_c);
    result_6 = atol(B3_a);
    result_7 = atol(B3_b);
    result_8 = atol(B3_c);

}

if(output[27] == 'C'){ //Conjuntos
//difusos para velocidad del motor

                       //Conjunto izquierda
    C1_a[0] = output[0];
    C1_a[1] = output[1];
    C1_a[2] = output[2];
    C1_b[0] = output[3];
    C1_b[1] = output[4];
    C1_b[2] = output[5];
    C1_c[0] = output[6];
    C1_c[1] = output[7];
```

```
C1_c[2] = output[8];
                    //Conjunto centro
C2_a[0] = output[9];
C2_a[1] = output[10];
C2_a[2] = output[11];
C2_b[0] = output[12];
C2_b[1] = output[13];
C2_b[2] = output[14];
C2_c[0] = output[15];
C2_c[1] = output[16];
C2_c[2] = output[17];
                    //Conjunto deracha
C3_a[0] = output[18];
C3_a[1] = output[19];
C3_a[2] = output[20];
C3_b[0] = output[21];
C3_b[1] = output[22];
C3_b[2] = output[23];
C3_c[0] = output[24];
C3_c[1] = output[25];
C3_c[2] = output[26];

result_0 = atol(C1_a);
result_1 = atol(C1_b);
```

```
      result_2 = atol(C1_c);
      result_3 = atol(C2_a);
      result_4 = atol(C2_b);
      result_5 = atol(C2_c);
      result_6 = atol(C3_a);
      result_7 = atol(C3_b);
      result_8 = atol(C3_c);

   }
}

IFS1bits.U2RXIF = 0; //pone a cero la
//bandera de la interrupción UART_RX 2

}
```

8.2 Código del la interfaz gráfica de usuario

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from numpy import arange, sin, pi, zeros
```

```
from matplotlib.backends.backend_tkagg
     import FigureCanvasTkAgg
from matplotlib.figure import Figure
import matplotlib.pyplot as plt

import serial
import io
import time
import datetime

import sys
import Tkinter as Tk

dsPIC = serial.Serial('/dev/ttyUSB0',
        baudrate=9600,timeout=0)
AngPos = [0,0]

def destroy(e):
    sys.exit()


def GrafPosPendulo():
    pos_leida_teta_0 = 0;
    pos_leida = 0;
```

```
y = zeros([51])
i = 50;
j = 0;
a = 0;
dsPIC.open()

while a == 0:
  if i>=0:
     y[i] = y[i-1]
     i = i-1
  else:
     dsPIC.write('Eok');
     time.sleep(0.1)
     txt = dsPIC.read(1)
     pos_leida = ord(txt)
     pos_leida_v =
     pos_leida*0.0196078431
     pos_leida_teta_1 =
     pos_leida_teta_0
     pos_leida_teta_0 =
     85.8851*pos_leida_v - 228.8847
     y[0] = pos_leida_teta_0
     if j==51:
        j = 0
```

```
            a = 1
        else:
            j = j+1
            i = 50

        dsPIC.flushInput()

dsPIC.close()
f = plt.figure(figsize=(8.5, 4),
dpi=80,edgecolor="#808080",
    facecolor="#808080")
plt.xlabel('Tiempo ' r'[$ms$]',
          fontsize=15, color='w')
plt.ylabel('Posicion angular ' r'
[$\theta$]', fontsize=15,color='w')
plt.title('Posicion angular del
pendulo',color='w',fontsize=20)
ax = f.add_subplot(111)
ax.grid(True, color='k',
linewidth=0.4, linestyle='--')
t = arange(0,5100,100)
line, = ax.plot(t, y,color='#00FF00',
        linewidth=2)
#ax.tick_params(axis='x',which='both',
```

```
    labelbottom='off',labelleft='off')
    canvas = FigureCanvasTkAgg(f,root)
    canvas.get_tk_widget().pack(side=
    Tk.LEFT, fill=Tk.BOTH,expand=0)
    f.canvas.show()
    for i in range(52):
          line.set_xdata(t[:i])
          line.set_ydata(y[:i])
          f.canvas.show()

def GrafVelPendulo():
    pos_leida_teta_0 = 0;
    pos_leida = 0;
    y = zeros([51])
    i = 50;
    j = 0;
    a = 0;
    dsPIC.open()

    while a == 0:
      if i>=0:
         y[i] = y[i-1]
         i = i-1
      else:
```

```
        dsPIC.write('Eok');
        time.sleep(0.1)
        txt = dsPIC.read(1)
        pos_leida = ord(txt)
        pos_leida_v =
        pos_leida*0.0196078431
        pos_leida_teta_1 =
        pos_leida_teta_0
        pos_leida_teta_0 =
        85.8851*pos_leida_v - 228.8847
        y[0] = pos_leida_teta_0 -
        pos_leida_teta_1
        if j==51:
           j = 0
           a = 1
        else:
           j = j+1
           i = 50

        dsPIC.flushInput()

dsPIC.close()
f = plt.figure(figsize=(8.5, 4),
dpi=80,edgecolor="#808080",
```

```
    facecolor="#808080")
    plt.xlabel('Tiempo ' r'[$ms$]',
    fontsize=15, color='w')
    plt.ylabel('Velocidad angular ' r'
    [$\theta / ms$]',fontsize=15, color='w')
    plt.title('Velocidad angular del pendulo',
    color='w',fontsize=20)
    ax = f.add_subplot(111)
    ax.grid(True, color='k', linewidth=0.4,
     linestyle='--')
    t = arange(0,5100,100)
    line, = ax.plot(t, y,'c',linewidth=2)
    #ax.tick_params(axis='x',which='both',
    labelbottom='off',labelleft='off')
    canvas = FigureCanvasTkAgg(f,root)
    canvas.get_tk_widget().pack(side=Tk.
    LEFT, fill=Tk.BOTH,expand=0)
    f.canvas.show()
    for i in range(52):
          line.set_xdata(t[:i])
          line.set_ydata(y[:i])
          f.canvas.show()

def GrafVelMotor():
```

```
y = zeros([51])
i = 50;
j = 0;
a = 0;
dsPIC.open()

    while(a == 0):
   if i>=0:
        y[i] = y[i-1]
        i = i-1
   else:
       dsPIC.write('Fok')
       time.sleep(0.1)
       txt = dsPIC.read(1)
       POSCNT = ord(txt);
       POSCNT = -POSCNT;
       AngPos[1] = AngPos[0];
       AngPos[0] = (POSCNT*2048)/(116);
       Speed = AngPos[0] - AngPos[1];
       if(Speed >= 0):
              if(Speed >= 16375):
                  Speed = Speed - 32751;
       else:
              if(Speed < -16375):
```

```
                    Speed = Speed + 32751;
            Speed *= 2;
            y[0] = (Speed*400)/(32768)
            if j==51:
                j = 0
                a = 1
            else:
                j = j+1
                i = 50
       dsPIC.flushInput()
    dsPIC.close()
    f = plt.figure(figsize=(8.5, 4),
    dpi=80,edgecolor="#808080",
    facecolor="#808080")
    plt.xlabel('Tiempo ' r'[$ms$]',
    fontsize=15, color='w')
    plt.ylabel('Velocidad angular ' r'
    [$RPM$]', fontsize=15, color='w')
    plt.title('Velocidad angular del motor',
     color='w',fontsize=20)
    ax = f.add_subplot(111)
    ax.grid(True, color='k', linewidth=0.4,
    linestyle='--')
    t = arange(0,5100,100)
```

```
    line, = ax.plot(t, y,'m',linewidth=2)
    #ax.tick_params(axis='x',which='both',
    labelbottom='off',labelleft='off')
    canvas = FigureCanvasTkAgg(f,root)
    canvas.get_tk_widget().pack(side=Tk.
    LEFT, fill=Tk.BOTH,expand=0)
    f.canvas.show()
    for i in range(52):
          line.set_xdata(t[:i])
          line.set_ydata(y[:i])
          f.canvas.show()

def enviar():
      if   opcion.get() == 1:
            dsPIC.open()
            n = nombre.get()
            m = n[:28] + 'Aok'
            dsPIC.write(m)
            time.sleep(0.1)
            txt = dsPIC.read(1)
            a = ord(txt)
            print a
            dsPIC.flushInput()
            dsPIC.close()
```

```
      if   opcion.get() == 2:
            dsPIC.open()
            n = nombre.get()
            m = n[:28] + 'Bok'
            dsPIC.write(m)
            time.sleep(0.1)
            txt = dsPIC.read(1)
            a = ord(txt)
            print a
            dsPIC.flushInput()
            dsPIC.close()
      if   opcion.get() == 3:
            dsPIC.open()
            n = nombre.get()
            m = n[:28] + 'Cok'
            dsPIC.write(m)
            time.sleep(0.1)
            txt = dsPIC.read(1)
            a = ord(txt)
            print a
            dsPIC.flushInput()
            dsPIC.close()

root = Tk.Tk()
```

```
root.wm_title("Interfaz gráfica del
               péndulo invertido")
root.wm_geometry("1100x500")
opcion = Tk.IntVar()
nombre = Tk.StringVar()
nombre.set("Ejemplo: 30  0 10 11  0
            11 10  0  30 ")

etiqueta1 = Tk.Label(root, text="
            Interfaz gráfica de péndulo
            invertido ",font=("Helvetica",18)).
            place(x=620,y=10)
etiqueta2 = Tk.Label(root, text="Selecciona los
            conjuntos difusos que deseas
            modificar").place(x=720,y=200)
etiqueta3 = Tk.Label(root, text="Ingresa los
            valores de los conjuntos difusos")
            .place(x=720,y=330)
nombreCaja = Tk.Entry(root, textvariable=nombre)
             .place(x=720,y=365)
BotSalir   = Tk.Button(root, text='Salir',
             bg="#FF4500",command=sys.exit)
             .place(x=720,y=420)
BotGrafica1 = Tk.Button(root, text='Gráfica de
```

```
                posición del péndulo';bg="#90EE90",
                command = GrafPosPendulo)
                .place(x=720,y=50)
BotGrafica2 = Tk.Button(root, text='Gráfica de
                velocidad del péndulo',bg="#90EE90",
                command = GrafVelPendulo).place(x=720
                ,y=100)
BotGrafica3 = Tk.Button(root, text='Gráfica de
                velocidad del motor',bg="#90EE90",
                command = GrafVelMotor).place
                (x=720,y=150)
BotEnviar = Tk.Button(root, text='Enviar',
              bg="#1E90FF",command = enviar).
              place(x=920,y=359)
PosPendulo = Tk.Radiobutton(root,
               text="Posición del péndulo",
               value=1, variable=opcion).
               place(x=720,y=230)
PosVelocidad = Tk.Radiobutton(root,
                 text="Velocidad del péndulo",
                 value=2, variable=opcion).
                 place(x=720,y=260)
VelMotor = Tk.Radiobutton(root,
             text="Velocidad del motor",
```

```
            value=3, variable=opcion).
            place(x=720,y=290)

root.mainloop()
```

Bibliografía

[AGElectrónica, 2015] AGElectrónica (2015). *http : //www.agelectronica.com/virtual*$_s$*hop/index.asp*. [Web; accedido en junio de 2015].

[Amazon, 2015] Amazon (2015). *http : //www.amazon.com/double − bts*7960*b − driver − high − power − arduino/dp/b*00*hr*2*dhj*0. [Web; accedido en junio de 2015].

[Jelelectrónica, 2015] Jelelectrónica (2015). *http : //jelelectronica.mercadoshops.com.mx/convertidor− dcdc − bajada − 724 − 5v − 3a − arduino − pic −* 21030214*xjm*. [Web; accedido en junio de 2015].

[Nise and Romo, 2002] Nise, N. S. and Romo, J. H. (2002). *Sistemas de control para ingeniería*. Cecsa.

[Ogata, 2010] Ogata, K. (2010). *Ingeniería de control moderna*. Pearson Educación.

[Pololu, 2015] Pololu (2015). *https* : *//www.pololu.com/picture/view/*0*j*4045. [Web; accedido en junio de 2015].

[RS, 2015] RS (2015). *http* : *//jp.rs* − *online.com/largeimages/f*6229954 − 01.*jpg*. [Web; accedido en junio de 2015].

[Siler and Buckley, 2005] Siler, W. and Buckley, J. J. (2005). *Fuzzy expert systems and fuzzy reasoning*. John Wiley & Sons.

[Topled, 2015] Topled (2015). *http* : *//www.topled.mx/bateria* − *portatil* − *recargable* − 12*v* − 1800 − *mah* − *para* − *tiras* − *led* − 55485147*xjm*. [Web; accedido en junio de 2015].