ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Balancín de dos ruedas con controlador Pololu"

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRICIDAD ESPECIALIZACIÓN ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

INGENIERO EN ELECTRÓNICA Y TELECOMUCACIONES

Presentado por:

José Antonio Intriago Torres

Fernando Luis Rodríguez Gallegos

GUAYAQUIL – ECUADOR

AÑO 2011

AGRADECIMIENTO

A Dios.

A la familia.

A todas las personas que apoyaron el desarrollo de este trabajo.

TRIBUNAL DE SUSTENTACIÓN

_____

Ing. Carlos Valdivieso

Profesor de Seminario de Graduación

_____

Ing. Hugo Villavicencio V.

Profesor Delegado del Decano

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta tesina, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Fernando Rodríguez

_____

José Intriago

RESUMEN

En este proyecto implementamos las técnicas aprendidas con respecto al uso de los microcontroladores. Para con estos lograr la creación de un balancín de dos ruedas con controlador Pololu. Aplicamos también los conocimientos adquiridos del control automático, gracias a los cuales nos es posible mantener el sistema equilibrado.

ÍNDICE GENERAL

Contenido

ÍNDICE DE FIGURAS

CONCLUSIONES

1. El modelo que trabajamos representa una planta de control relativamente complicada. Puesto que la mayoría de sistemas no presentan una sensibilidad tan alta como el nuestro. Es decir son sistemas capaces de mantenerse operativos aun con errores de consideración. Mientras que en el presente caso, literalmente todo el sistema se desplomaría.
2. El uso de librerías de Pololu facilitó bastante nuestro trabajo al momento de programar, pues estas incluyen funciones que miden la duración de pulsos y controlan la velocidad de los motores. No obstante para su correcta implementación se requiere analizar minuciosamente la estructura de las mismas.
3. La tarjeta Orangutan SV-328 fue un gran apoyo para la culminación de este proyecto. Pues como se menciona anteriormente no solo que nos permitió trabajar con comandos que se ajustaban a las necesidades específicas de este proyecto, sino que también incluye un microcontrolador y un driver para los motores. Los cuales son elementos de vital importancia para nuestro trabajo.

RECOMENDACIONES

1. Para un mejor rendimiento del sistema de control se deberían considerar más pulsos pasados. En este caso solo se consideró el error inmediato anterior.

2. Los motores utilizados pueden llegar a demandar hasta 5A mientras que el Orangutan SV-328 provee hasta 3A. Por lo cual, bajo ninguna circunstancia estos dos elemento deben interconectarse directamente.

3. Al momento de determinar experimentalmente las constantes kp y ki. Es recomendable empezar por la proporcional manteniendo ki en 0. Una vez se encuentre una constante que mantenga el sistema con un cierto equilibrio, se debe proceder a variar el valor de ki para refinar la estabilidad del sistema.

Anexos

Anexo A: Procesamiento de la señal del MX2125

4. TEORIA DE FUNCIONAMIENTO

El sensor MX2125 que incorpora el módulo 28017 de Parallax consta de un receptáculo cuadrado con un elemento calorífico que calienta una burbuja de gas, y cuatro sensores de temperatura o termopilas a cada lado del receptáculo como se muestra en la figura 2.

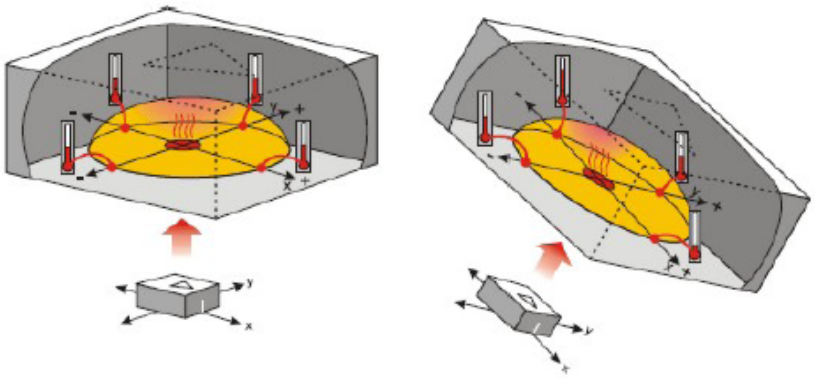


Figura 2. *Interior del sensor MX2125*

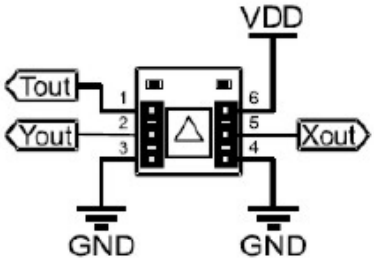
Cuando el sensor se encuentra totalmente nivelado, la burbuja de gas caliente se desplaza centrada hacia arriba en el interior del receptáculo. Los cuatro sensores de temperatura o termopilas registran un mismo valor térmico.

Cuando el sensor sufre algún tipo de giro en cualquiera de sus ejes, la burbuja de gas se desplaza en el interior de la carcasa provocando un aumento de temperatura en algunas de las termopilas y reducción de temperatura en otras. Comparando esas temperaturas se detecta tanto la aceleración dinámica como estática (gravedad y giro). La electrónica integrada en el sensor MX2125 convierte las medidas de temperatura en señales PWM fácilmente manejables por cualquier controlador (PIC, Atmel, Basica Stamp, Arduino, etc..)

5.- PATILLAJE

El encapsulado y distribución de las patillas del módulo acelerómetro 28017 se muestra en la figura 3 y la descripción de las mismas en la tabla adjunta.

Pin N°	Nombre	Descripción
1	Tout	Salida analógica de temperatura
2	Yout	Salida PWM del eje Y
3	GND	Tierra de alimentación
4	GND	Tierra de alimentación
5	Xout	Salida PWM del eje X
6	Vdd	Alimentación de +3.3V hasta +5V



6. PROTOCOLO DE COMUNICACIÓN

Cada eje proporciona una señal PWM, como la mostrada en la figura 4, con una frecuencia de 100Hz (periodo de 10000 µS) cuya anchura es proporcional a la aceleración. Cualquier tipo de controlador puede medir la duración de la anchura del pulso y obtener así los valores de aceleración.

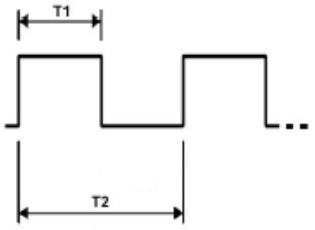


Figura 4. *Señal de salida para cada eje*

Con una alimentación de +5V, una aceleración de 0g se corresponde con el 50% del ciclo útil aunque puede fluctuar entre 48.7% y 51.3%. Esto significa que si la señal PWM de salida tiene un periodo de 10000 µS (100 Hz), una anchura del ciclo útil de 5000 µS (50%) se corresponde con una aceleración de 0 g. La aceleración se calcula con la siguiente fórmula proporcionada por el fabricante:

$$A(g) = ((T1 / T2) - 0.5) / 12.5\ \%$$

De cara a emplear un microcontrolador, la siguiente ecuación es equivalente y facilita los cálculos que hay que realizar:

$$A(g) = (((T1 / 10) - 500) * 8) / 1000$$

Conocida la aceleración de cualquiera de los ejes, también se puede calcular el ángulo de giro del mismo mediante la función arco seno (asin) de g:

$$\text{ángulo} = \text{asin}\ (g)$$

Algunos compiladores como el compilador C de la firma CCS que hemos empleado en los ejemplos que proponemos, el resultado de la función asin() se ofrece en radianes. Para pasarlos a grados sexagesimales se realiza la siguiente operación:

$$1\ \text{Radián} = 360/2\pi = 57.295779\ ^\circ$$

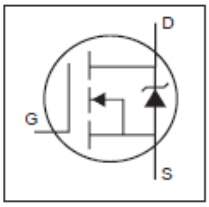
Anexo B: Características del MOSFET IRFZ44N

IOR Rectifier

IRFZ44N

HEXFET® Power MOSFET

- Advanced Process Technology
- Ultra Low On-Resistance
- Dynamic dv/dt Rating
- 175°C Operating Temperature
- Fast Switching
- Fully Avalanche Rated



V_{DSS} = 55V

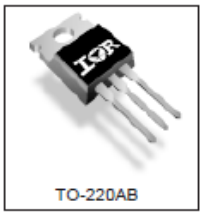
$R_{DS(on)}$ = 17.5mΩ

I_D = 49A

Description

Advanced HEXFET® Power MOSFETs from International Rectifier utilize advanced processing techniques to achieve extremely low on-resistance per silicon area. This benefit, combined with the fast switching speed and ruggedized device design that HEXFET power MOSFETs are well known for, provides the designer with an extremely efficient and reliable device for use in a wide variety of applications.

The TO-220 package is universally preferred for all commercial-industrial applications at power dissipation levels to approximately 50 watts. The low thermal resistance and low package cost of the TO-220 contribute to its wide acceptance throughout the industry.



TO-220AB

Absolute Maximum Ratings

	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10V	49	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V	35	
I_{DM}	Pulsed Drain Current ①	160	
P_D @T_C = 25°C	Power Dissipation	94	W
	Linear Derating Factor	0.63	W/°C
V_{GS}	Gate-to-Source Voltage	± 20	V
I_{AR}	Avalanche Current①	25	A
E_{AR}	Repetitive Avalanche Energy①	9.4	mJ
dv/dt	Peak Diode Recovery dv/dt ③	5.0	V/ns
T_J T_{STG}	Operating Junction and Storage Temperature Range	-55 to + 175	°C
	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	
	Mounting torque, 6-32 or M3 srew	10 lbf•in (1.1N•m)	

Thermal Resistance

	Parameter	Typ.	Max.	Units
$R_{\theta JC}$	Junction-to-Case	—	1.5	°C/W
$R_{\theta CS}$	Case-to-Sink, Flat, Greased Surface	0.50	—	
$R_{\theta JA}$	Junction-to-Ambient	—	62	

Anexo C: Características del Orangutan SV-328

General specifications

Processor:	ATmega328 @ 20 MHz
RAM size:	2048 bytes
Program memory size:	32 Kbytes
Motor driver:	TB6612FNG
Motor channels:	2
User I/O lines:	8[1]
Max current on a single I/O:	40 mA
Minimum operating voltage:	6 V
Maximum operating voltage:	13.5 V
Continuous output current per channel:	1 A
Peak output current per channel:	3 A
Maximum PWM frequency:	80 kHz
Reverse voltage protection?:	Y
External programmer required?:	Y
LCD included?:	Y

AVR Pin Assignment Table Sorted by Function

Function	megaxx8 Pin
digital I/Os (x8)	PD0, PD1, PC0 – PC5
analog inputs (x8)	PC0 – PC5, ADC6, ADC7
motor 1 control (A and B)	PD5 and PD6
motor 2 control (A and B)	PD3 and PB3
red user LED	PD1
green user LED	PD7
user pushbuttons (x3)	PB1, PB4, and PB5
buzzer	PB2
LCD control (RS, R/W, E)	PD2, PB0, and PD4
LCD data (4-bit: DB4 – DB7)	PB1, PB4, PB5, and PD7
user trimmer potentiometer	ADC7 (through shorting block)
temperature sensor (LV-168 only) battery voltage monitor (SV-xx8 only)	ADC6 (through SMT jumper)
ICSP programming lines (x3)	PB3, PB4, PB5
reset pushbutton	PC6
UART (RX and TX)	PD0 and PD1
I2C/TWI (SDA and SCL)	PC4 and PC5
SPI	inaccessable to user

Bibliografía

(1) Segway Inc., Características del Segway

http://www.segway.com/about-segway/segway-technology.php

Fecha de Consulta: 08/04/2011

(2) Wikimedia Foundation, Datos históricos y características de los robots

http://es.wikipedia.org/wiki/Robot

Fecha de Consulta: 09/04/2011

(3) Cornell University, Características del AVR Studio

http://courses.cit.cornell.edu/ee476/AtmelStuff/doc1019.pdf

Fecha de Consulta: 22/04/2011

(4) Labcenter Electronics, Información referente al programa Proteus

http://www.labcenter.com/products/vsm_overview.cfm

Fecha de Consulta: 24/04/2011

(5) Pololu Corporation, Datos generales referentes al hardware del proyecto

http://www.pololu.com

Fecha de Consulta: 24/04/2011

(6) Parallax Inc., Características del acelerómetro MX2125

http://www.parallax.com/dl/docs/prod/compshop/SICMemsicTut.pdf

Fecha de Consulta: 26/04/2011

(7) Ingeniería de Microsistemas Programados S.L., Análisis de la salida del MX2125

http://www.msebilbao.com/notas/downloads/Acelerometro%20de%202%20ejes%2028017.pdf

Fecha de Consulta: 11/05/2011

CAPITULO 1

1 Descripción del Problema

1.1 Antecedentes

El presente proyecto es un ejemplo clásico de la aplicación de los fundamentos del Control Automático a sistemas que utilizan microcontroladores. Combinación que da como resultado sistemas sumamente eficientes en consumo de energía, espacio utilizado y precisión; gracias al alto rendimiento de estos integrados.

Los principios por los cuales opera el balancín de dos ruedas son la base de los sistemas de locomoción de los robots bípedos. Esta no solo que es una aplicación sumamente interesante de este sistema, sino también una con un amplio campo de desarrollo a futuro. Si partimos del principio obvio de que el entorno en el que nos desenvolvemos está perfectamente adaptado a nuestro sistema de locomoción y agregamos a esto la interacción en crecimiento con sistemas autónomos en prácticamente cualquier campo. Podremos llegar a la

conclusión de que eventualmente este tipo de robots se volverán parte integral de la vida cotidiana.

1.2 Situación actual

Uno de las aplicaciones más notables de este proyecto es el vehículo de transporte ligero giroscópico, Segway. Aunque por razones de costo su uso no se encuentre muy difundido, su bajo consumo de energía y versatilidad son características que encajan perfectamente en los requerimientos de la sociedad actual. Mismos que son consecuencia directa de la implementación de microcontroladores. Podemos agregar a estas características su elegancia, la cual claramente se aprecia en la Figura 1. (1)

Figura 1 Segway

Cabe mencionar los inicios de este producto, debido a lo que nos es posible inferir de los mismos. Este vehículo fue inventado por Dean Kamen y presentado en diciembre de 2001. Este sistema de transporte autobalanceado posee su ordenador y sus motores en la base, por lo cual se mantiene horizontal todo el tiempo. Dependiendo de hacia dónde se incline el usuario, el Segway tomará esa dirección. Su motor es eléctrico y es capaz de alcanzar los 20km/h. (1)

Tenemos por tanto un vehículo unipersonal fácil de manejar, eficiente y en especial no dependiente del petróleo. Aun considerando el factor del costo, estas características deberían bastar para que el producto tenga una mayor acogida que la que posee en la actualidad. De hecho así fue, los medios en su momento llegaron a llamar a este el vehículo del futuro, habían quienes hasta especulaban que las ciudades serían construidas alrededor de este medio de transporte.

¿Qué pasó entonces? Lamentablemente el diseño posee un gran error, debido a la naturaleza mecánicamente inestable de los balancines. Cuando sus baterías están cerca de descargarse, es posible que no se genere suficiente potencia, ocasionando que literalmente el conductor caiga. A primera impresión parecería poca cosa, bastaría con que el propietario mantuviese

cargadas sus baterías con regularidad. Pero esto simplemente no se ajusta a la forma de ser del individuo común. Para el cual es bastante normal andar con las "últimas de gasolina". Es el producto el que debe adaptarse al usuario y no al revés, de esto y su costo deriva la baja demanda del mismo.

Otra aplicación fundamentada en este proyecto, es la elaboración de robots bípedos. Aunque su uso se encuentre aun muy poco difundido, así mismo por motivos de costo, la evolución de estos autómatas se está dando a pasos agigantados. Esto por supuesto es debido al altísimo potencial de aplicación que tienen estos dispositivos. En las ilustraciones siguientes, vemos la clase de trabajos (Figura 2) que dan origen a sistemas tales como el que se aprecia en la Figura 3.

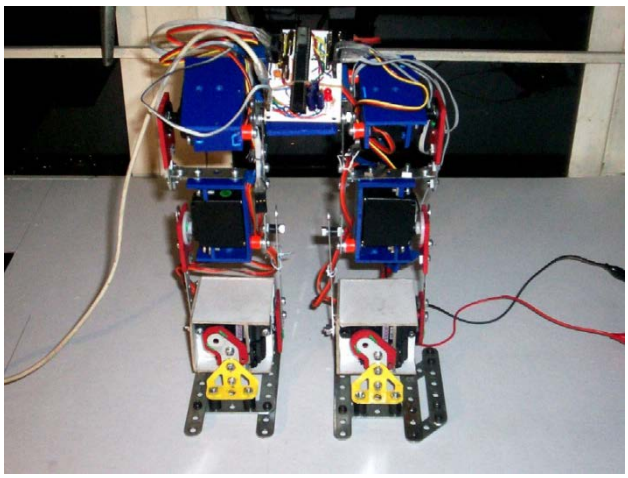


Figura 2 Robot bípedo, UPIICSA

Figura 3 Robot bípedo, Expo 2005 Aichi

En la actualidad, los robots comerciales e industriales son ampliamente utilizados, pues realizan tareas de forma más exacta y barata que los humanos. También se les utiliza en trabajos demasiado sucios, peligrosos o tediosos. Sin embargo los robots bípedos no solo estarán en capacidad de realizar este tipo de tareas, sino también de aquellas que requieran una interacción más cotidiana y menos rutinaria con las personas. Debido por supuesto a la capacidad que poseen de transportarse libremente en el mismo entorno de los humanos. (2)

La palabra robot, es de origen checo y significa siervo o esclavo; fue inventada por el escritor checo Karel Capek en su obra teatral R.U.R., estrenada en Europa en 1920. Pero el concepto de máquinas automatizadas se remonta a la antigüedad, con mitos de seres mecánicos vivientes. Las máquinas semejantes a personas, ya aparecían en los relojes de las iglesias medievales.

El control por realimentación, el desarrollo de herramientas especializadas y la división del trabajo en tareas más específicas, fueron los ingredientes esenciales para la automatización de las fábricas en el siglo XVIII. A medida que mejoraba la tecnología, se desarrollaron máquinas especializadas para tareas como poner tapones a las botellas o verter caucho líquido en moldes para neumáticos. (2)

De todo esto podemos darnos cuenta que la utilización de autómatas ha sido una añoranza muy antigua de nuestra sociedad. Claro que esto no es de extrañar; reducción de costos, fidelidad, velocidad y alta precisión. Cualidades que el promedio de los obreros está lejos de cumplir, pero que "coincidencialmente" son las características que describen a un robot. Lo cual explica el arduo y largo camino que se ha recorrido para llegar a los modelos actuales, camino que vemos representado a continuación en la Figura 4.

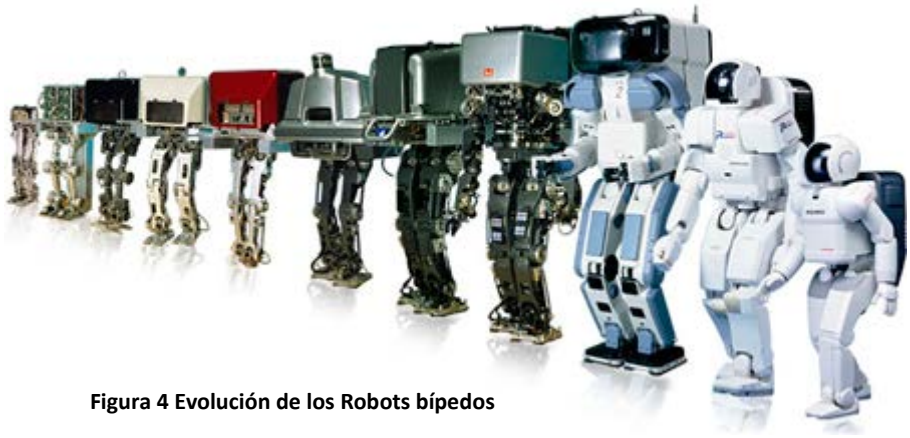


Figura 4 Evolución de los Robots bípedos

1.3 Historia

El balancín de dos ruedas es un proyecto que se ha vuelto muy popular en el campo de la mecatrónica. Este proyecto se basa en los mismos principios teóricos del igualmente popular experimento del péndulo invertido. La idea de un robot móvil tipo péndulo invertido ha surgido en años recientes y ha atraído la atención de los investigadores de sistemas de control de alrededor del

mundo. Los robots o los medios de transporte de este tipo son mecánicamente inestables, por lo cual es necesario explorar las diversas posibilidades de implementación de sistemas de control para mantener el equilibrio. De esto deriva la necesidad del desarrollo de proyectos de la naturaleza del presente.

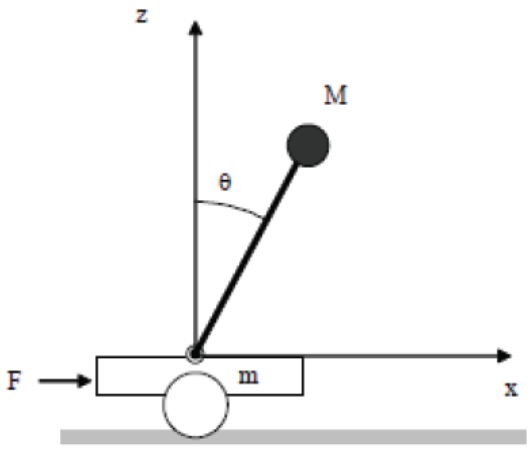


Figura 5 Diagrama de fuerzas del péndulo invertido

Figura 6 nBot

Las imágenes anteriores muestran las fuerzas que rigen el movimiento del balancín (Figura 5), sistema que es prácticamente idéntico al del péndulo invertido; versus un robot balancín perfectamente operativo (Figura 6). De hecho las primeras máquinas que se balanceaban activamente eran controladas

por péndulos invertidos. La importancia del balance activo en extremidades para la locomoción no es un problema nuevo. Mas el progreso en la construcción de sistemas con extremidades físicas que ocuparan estos principios fue retardado por la evidente dificultad del trabajo. Es más, no fue sino hasta finales de los años setenta que los experimentos en balance reanudaron su desarrollo. (2)

Otra de las aplicaciones principales de los principios del balancín, además de los robots, es el posicionamiento de un satélite. Para esto debemos visualizar al satélite en movimiento, mientras las antenas que se encuentran en la Tierra le impiden desplazarse demasiado, para que no salga de rango. Básicamente estos dos cuerpos (satélite y antena) están unidos por un vector virtual, como se aprecia en la Figura 7.

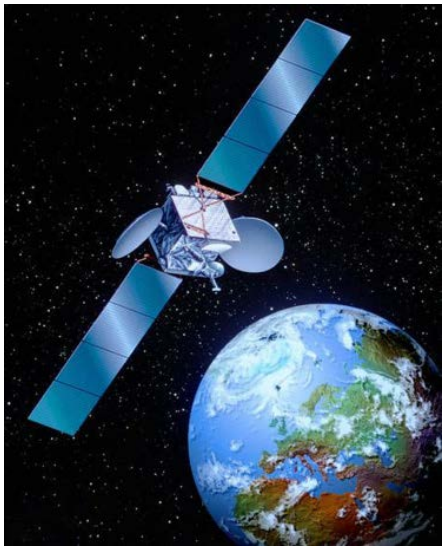


Figura 7 Sistema Tierra – satélite

CAPITULO 2

2 Herramientas

Este capítulo ha sido dividido en dos partes; Herramientas del Software y Herramientas de Hardware. Pues estos marcadamente distintos grupos constituyen la totalidad de nuestro proyecto. El primero trata de los programas que utilizaremos para la programación y simulación del mismo, mientras el segundo de los componentes que conforman su parte física.

2.1 Herramientas del Software

Estas herramientas cubren dos funciones; la compilación de nuestro código y la simulación del sistema. Para la compilación haremos uso del programa AVR Studio. Se utilizará Proteus para visualizar las conexiones del puente H que se implementará.

2.1.2 AVR Studio

Es el software que utilizaremos para programar al Orangutan SV-328. Para esto nos valdremos de uno de los dos compiladores que este software nos

ofrece, el AVR GCC, el cual es un compilador de lenguaje C. Este genera un archivo (.c), mismo que contiene el código a cargarse en el microcontrolador.

La capacidad de programar en C facilitará bastante la elaboración de nuestro código. Pues se trata de un leguaje de alto nivel, creado precisamente para simplificar el trabajo del programador. Ya que es un lenguaje muy bien estructurado que nos permite acceder a múltiples funciones asociadas a distintas librerías, mismas que son básicamente encapsulados de rutinas utilizadas frecuentemente. A continuación observamos la ventana inicial del AVR Studio (Figura 9), en la cual se nos permite seleccionar entre los dos compiladores mencionados. (3)

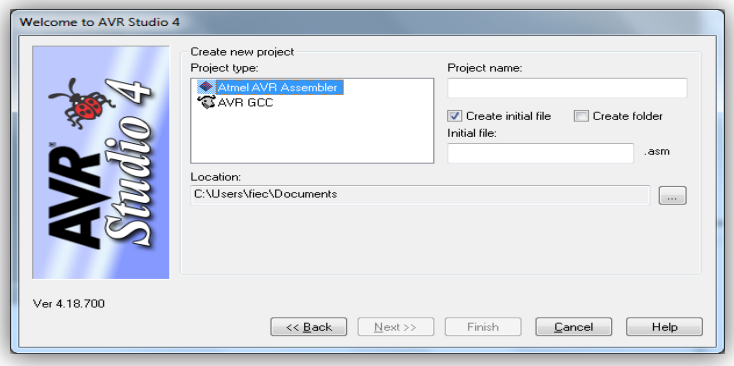


Figura 8 AVR Studio, selección de compilador

2.1.3 Proteus

Finalmente nos valdremos de este software para visualizar el comportamiento del hardware que utilizaremos, una vez se lo cargue con nuestro código. Este nos permite simular los componentes y sus conexiones.

Presenta una interfaz muy amistosa con el usuario (Figura 10), donde cada elemento posee una representación gráfica, en la cual se encuentran resaltadas sus entradas y salidas. Esta característica a más de otras, lo han convertido en el simulador más popular para microcontroladores. (4)

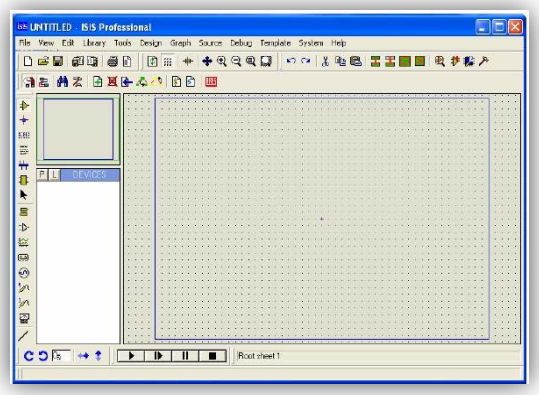


Figura 9 Proteus, ambiente de trabajo

2.2 Herramientas de Hardware

Como fue mencionado anteriormente, los siguientes componentes constituyen la parte física de nuestro proyecto. Partimos de su controlador, el Orangutan SV-328, que se encarga de enviar una señal adecuada al motor, en función de las lecturas que recepta el sensor. Luego se describe al

giroscopioLPY550AL, al acelerómetro MX2125; finalmente se describen los motores y las llantas.

2.2.1 **Orangutan SV-328** (5)

Es un controlador para robots muy compacto, que usaremos como procesador central de nuestro sistema. Este generará respuestas adecuadas en función de los datos que receptará de los sensores.

Cabe recalcar que esta tarjeta tiene incorporado el microcontrolador ATmega328P, un TB6612FNG, capaz de controlar 2 motores DC y 8 entradas analógicas. En la Figura 11 se destacan los componentes que integran a esta tarjeta.

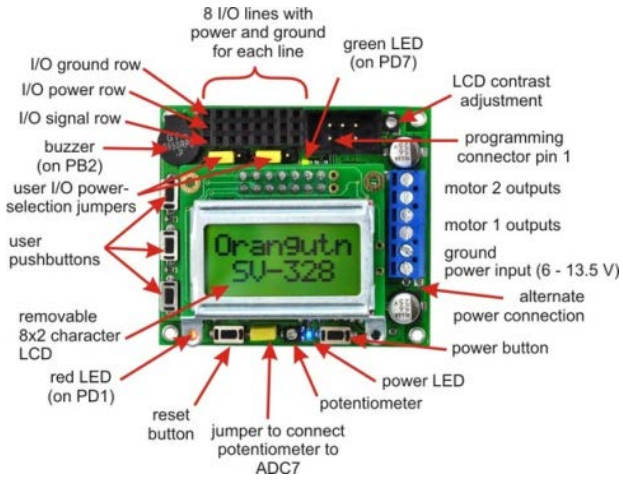


Figura 10 Orangutan SV-328

2.2.3 Acelerómetro MX2125(6)

A pesar de que este sensor es un acelerómetro, es capaz de generar lecturas de posición angular (característica que se usará en este trabajo). Esto es posible gracias a su particular diseño. En el cual se dispone de una cámara que contiene gas y un elemento calorífico, de tal manera que el aire caliente tenderá a subir al contrario del más frio. Gracias a este principio los sensores de temperatura que este componente contiene permiten estimar la inclinación a la que se encuentra.

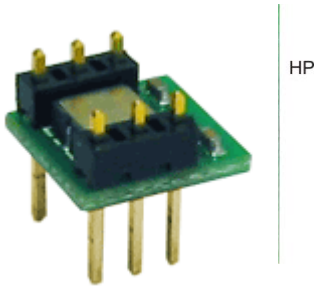


Figura 11 Acelerómetro MX2125

2.2.4 **Motor 19:1 de 37Dx52L mm con encoder 64 CPR** (5)

Poderoso motor de 12 voltios con un encoder de cuadratura integrado. Gracias al cual nos será posible determinar la velocidad de nuestro sistema. Posee una prescicion de 1216 conteos por revolución y es capaz de trabajara a 6v mas esta supuesto a trabajar con 12v. En la siguiente vista lateral del motor

(Figura 14), se observan los cables que corresponden a la alimentación (rojo y negro) y los del encoder.

Figura 12 Motor con encoder

2.2.5 **Llantas 90x10mm** (5)

En este trabajo se utilizarán llantas plásticas con neumáticos de silicón, cuyas dimensiones son de de 90x10mm. En la Figura 15 se aprecia en detalle las dimensiones de las mismas.

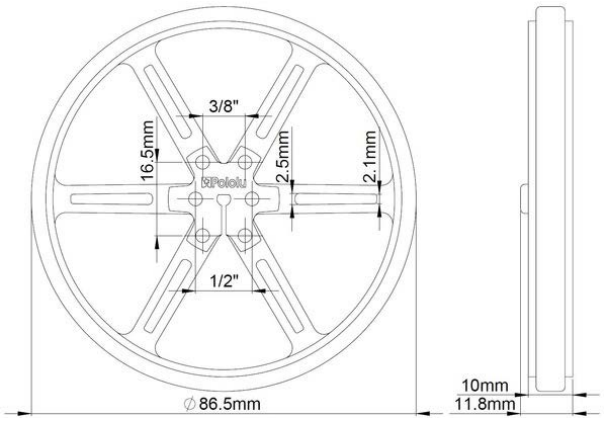


Figura 13 Llantas

2.2.5 MOSFET IRFZ44N

Se utilizan cuatro de estos MOSFETS para armar el puente H que intermediará entre el Orangutan SV-328 y los motores. Puesto que los motores pueden llegar a demandar más corriente que la que los drivers toleraran. Soportan corrientes de hasta 49 A, lo cual excede con creces el tope de 5 A que cada motor puede llegar a requerir.

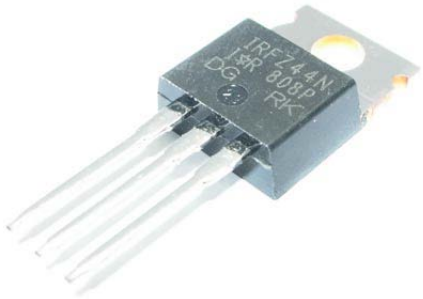


Figura 14 MOSFET IRFZ44N

CAPITULO 3

3 Diseño de la Solución

A continuación se presenta el procedimiento que se empleó para encarar este problema. Básicamente nos valemos del acelerómetro MX2125, sensor que envía lecturas respecto a una variable específica a un controlador. Este aplica a las mismas un lazo PID, en función del cual se hará llegar a los motores pulsos (PWM) con el fin de recuperar la posición de equilibrio. El controlador PID a utilizarse es de la forma:

$$\mathbf{v= } \boldsymbol{kp} * \boldsymbol{error} + \boldsymbol{ki}(\mathbf{error} + \mathbf{error_{anterior}}) + \mathbf{kd}(\mathbf{error} - \mathbf{error_{anterior}})$$

***v**= velocidad del motor*
***kp**= constante proporcional*
***ki**= constante integral* ***Kd**= constante diferencial*

El error (0° - ángulo actual) será calculado a partir de los pulsos emitidos por el MX2125. Dichos pulsos representan en su duración el ángulo experimentado por este sensor. Cuando este se encuentra en la posición de equilibrio genera pulsos altos de 5ms; por lo cual este valor será una importante referencia. La siguiente fórmula es recomendada por el fabricante para estimar

el ángulo de inclinación (en radianes) del MX2125 y por supuesto es implementada en nuestro código: (7)

aceleración= (((duración del pulso en alto / 10.0) - 500) * 8) / 100
10 representa el periodo de la señal (10 ms constantes)

La tarjeta controladora, Orangutan SV-328, dispone del microcontrolador ATmega328P. El cual receptará la lectura del sensor y generará una respuesta acorde. Esta señal generada (PWM) controlará el estado de los MOSFETS IRFZ44N (mismos que integran el puente H) por medio del driver TB6612FNG. Este también viene incorporado al Orangutan SV-328. De esta manera se controlará la velocidad y sentido de los motores para recuperar la posición de equilibrio.

Debido a que los motores pueden llegar a demandar hasta 5A y el Orangutan SV-328 provee solo hasta 3A, nos valdremos de un arreglo de MOSFETS (puente H) para precautelar la integridad de la tarjeta controladora. De tal forma que los pulsos (PWM) emitidos desde el Orangutan SV-328, controlarán el estado del puente y este a su vez será el que alimente de manera directa a los motores.

Finalmente, nuestro sistema será energizado con una fuente de 12V, a excepción del acelerómetro MX2125. Pues este requiere una alimentación de

3.3V a 5V y consume menos de 4mA. Por lo cual utilizaremos uno de los pines Vcc del Orangutan SV-328, pues estos generan 5V y son capaces de proveer hasta 100mA.

3.1 Diagrama de Bloques

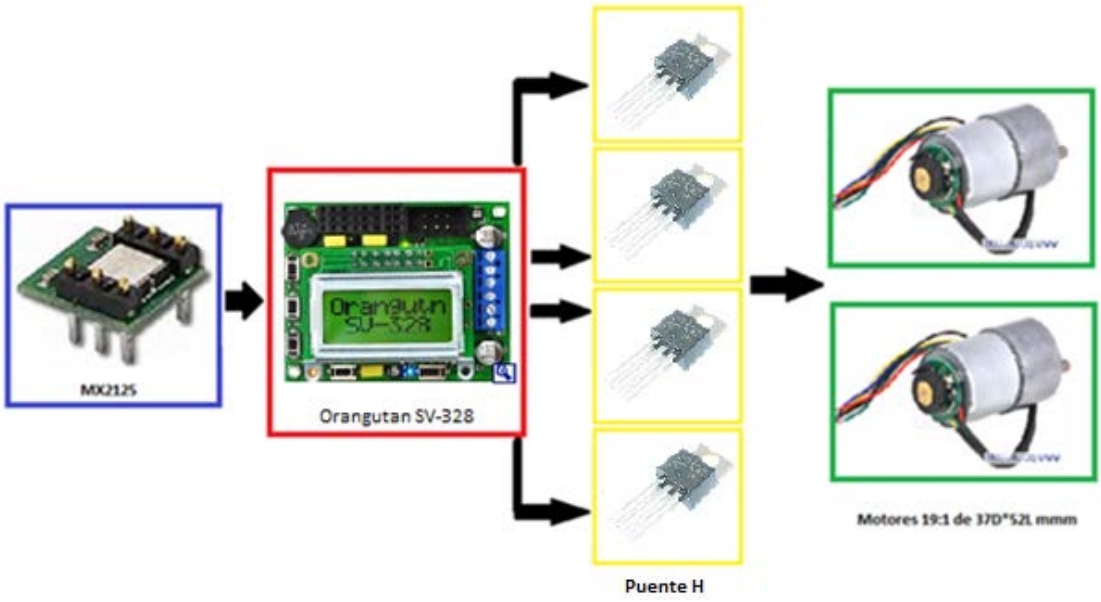


Figura 15 Diagrama de bloques del proyecto

3.2 Diagrama de Flujo del Controlador

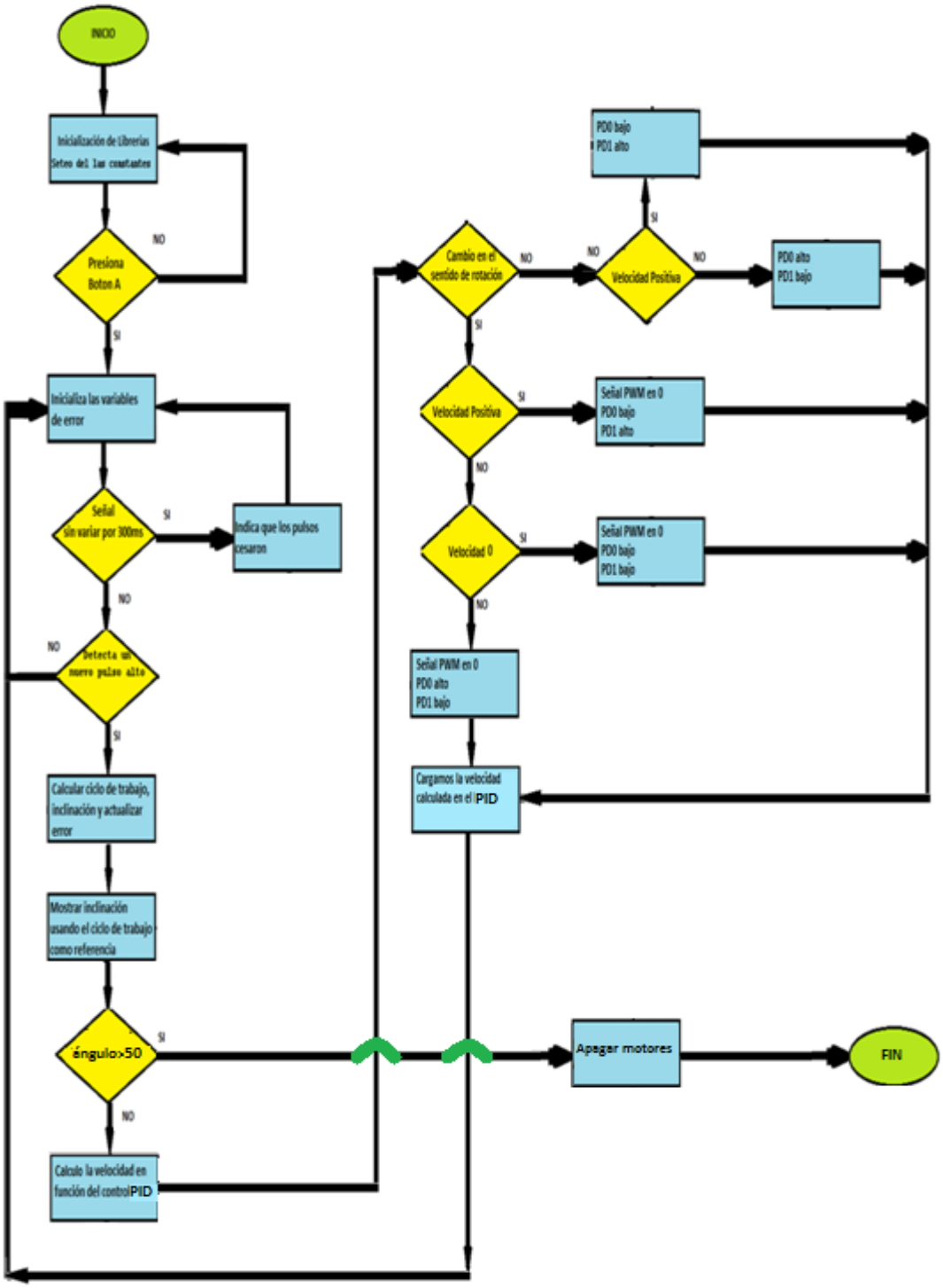


Figura 16 Diagrama de flujo del proyecto

3.3 Código

```
/* Código a implementarse en el proyecto Balancín de dos ruedas con controlador Pololu
 * Este programa combina las funciones del  OrangutanPulseIn y OrangutanServos.
 * Las primeras para medir los pulsos generados por el MX2125 conectados al pin PC5
 * del Orangutan SV-328. Las funsiones del OrangutanServos se utilizan para controlar la
 * velocidad de los motores, determinada previamente por un controlador PI. */

//*************************************LIBRERÍAS************************
#include <pololu/orangutan.h>  //recoge todas las librerías compatibles con el Orangutan
#include <math.h>              //librería de funciones matemáticas
#define pi 3.141592
#define kp 6.5                 //constantes para controlador PID
#define ki 0.5
#define kd 1

//***********************************************************************
const unsigned char pulseInPins[] = { IO_C5 };     //habilita PC5 como entrada de pulsos

//*****************************Función de habilitación*********************
void arranque()                          //paso previo para iniciar el programa principal
{
        while(!button_is_pressed(BUTTON_A)) //ciclado mientras no se presione el botón A
  {
    clear();                             //limpia el LCD
    print("Balancin");                   //escribe en el LCD
    lcd_goto_xy(0,1);                    //coloca el cursor en la coordenada especificada
    print("Pulse A");
    delay_ms(100);                       //retardo
  }
  wait_for_button_release(BUTTON_A);         //espera a que se suelte el botón A
  clear();
}

//*******************************Programa principal*********************
int main()
{
        arranque();

        int recuperable= 1;              //bandera que indica si el ángulo es recuperable
        long motor_speed_old= 1;         //velocidad anterior
        int error= 0;                    //error actual
        int old_error= 0;                //error anterior
```

```
pulse_in_start(pulseInPins,1);          //inicia lectura de pulsos en PD0

while(recuperable==1)                   //lazo principal
{
        unsigned long acPulse;          //duración del pulso actual (0.4 us)
        unsigned char estado;           //estado actual de la entrada (alta 1, baja 0)

        get_current_pulse_state(0,&acPulse,&estado);  //pasa los argumentos como
                                                        punteros

        //Si más de 300 ms transcurrido desde el último cambio en PD0
        //indicamos que los pulsos han parado

        if (pulse_to_microseconds(acPulse) >= 300000UL)
        {

                clear();                //limpia el LCD
                print("Sin pulsos ");   //escribe en el LCD
        }

//*******************************Interpreta la señal*****************************
        else if (new_high_pulse(0) && new_low_pulse(0)) //si se tiene un nuevo
                                                          pulso
        {
                float ax, angulo;
                unsigned long pulso_alto_conteos= get_last_high_pulse(0);
                unsigned long periodo_en_conteos= pulso_alto_conteos +
                get_last_low_pulse(0);
                unsigned long pulso_alto_us= 0.4 * pulso_alto_conteos;
                //duty cycle = pulso alto / (pulso alto + pulso bajo)
                //lo multiplicamos por 100 para convertirlo a porcentaje y le suma_
                //mos la mitad del denominador al numerador para redondearlo
                unsigned long duty_cycle_percent = (100 * pulso_alto_conteos +
                periodo_en_conteos / 2) / periodo_en_conteos;

                if (duty_cycle_percent == 50)           //posición deseada
                {
                        clear();
                        lcd_goto_xy(0,0);
                        print("Angulo:");
                        lcd_goto_xy(0,1);
                        print("Angulo:");
                        print_unsigned_long(0);         //muestro ángulo

                        old_error= error;
```

```
                error= 0;                       //actualiza los errores
        }

        else if (duty_cycle_percent > 50)
        {
                ax= (((pulso_alto_us / 10.0) - 500) * 8) / 1000;
                angulo= ((asin(ax)) * 360.0) / (2 * pi); //cálculo del ángulo

                clear();
                lcd_goto_xy(0,0);
                print("Angulo:");
                lcd_goto_xy(0,1);
                print_unsigned_long(angulo);    //muestro ángulo

                old_error= error;
                error= 0 - angulo;              //actualiza los errores
        }

        else
        {
                ax= (((pulso_alto_us / 10.0) - 500) * 8) / 1000;
                angulo= ((asin(ax) * -1) * 360.0) / (2 * pi); //cálculo ángulo

                clear();
                lcd_goto_xy(0,0);
                print("Angulo: ");
                lcd_goto_xy(0,1);
                print("-");
                print_unsigned_long(angulo);    //muestro ángulo

                old_error= error;
                error= 0 + angulo;              //actualiza los errores
        }

//*****************************Ángulo irrecuperable**************************
        if (angulo>50)
        {
                recuperable=0;
                set_motors(0,0);  //apaga los motores y termina el programa
        }

//*******************************Fija Velocidad*******************************
        else
        {
```

```
                long motor_speed= kp * error + ki * (error + old_error) + kd*
        (error - old_error); //ajusta la velocidad de control generada por el lazo PID

                 if(motor_speed > 0)
                {
                        motor_speed=motor_speed+30; //ajusta velocidad

                        if(motor_speed > 255)
                        {
                                motor_speed=255;        //límite de velocidad
                        }
                 }

                else if (motor_speed < 0)
                 {
                        motor_speed=motor_speed-30; //ajusta velocidad

                        if(motor_speed < -255)
                        {
                                motor_speed=-255;       //límite de velocidad
                        }
                }
//*********************************Habilita MOSFETS*************************
//PD0 y PD1 se utilizan para habilitar los MOSFETS de la parte "baja" del puente H
//los MOSFETS de la parte alta son habilitados directamente por la señal PWM del
Orangutan SV-328
                if((motor_speed * motor_speed_old) > 0)//no hay cambio en el
                                                        sentido de  rotación
                {
                        if(motor_speed > 0)             //continúa "avance"
                        {
                                set_digital_output (IO_D0,LOW);
                                set_digital_output (IO_D1,HIGH);
                        }

                        else                            //continúa "retroceso"
                        {
                                set_digital_output (IO_D1,LOW);
                                set_digital_output (IO_D0,HIGH);
                        }

                }

                else                    //cambio en el sentido de rotación o parado
                {
```

```
                    if(motor_speed > 0)      //cambiar a "avance"
                    {
                        set_motors(0,0);
                        set_digital_output (IO_D0,LOW);
                        set_digital_output (IO_D1,HIGH);
                    }

                    else if (motor_speed == 0)//frenado
                    {
                        set_motors(0,0);
                        set_digital_output (IO_D0,LOW);
                        set_digital_output (IO_D1,LOW);
                    }

                    else                      //cambiar a "retroceso
                    {
                        set_motors(0,0);
                        set_digital_output (IO_D1,LOW);
                        set_digital_output (IO_D0,HIGH);
                    }
                }

                set_motors(motor_speed, motor_speed);  //fija la velocidad de los .
                                                        motores
                motor_speed_old= motor_speed;
                delay_ms(10);

            }
        }
    }
}
```

CAPITULO 4

4 Simulaciones y pruebas

4.1 Simulación del puente H

Con el fin de facilitar la visualización del puente H empleado se presenta a continuación la representación del mismo en Proteus:

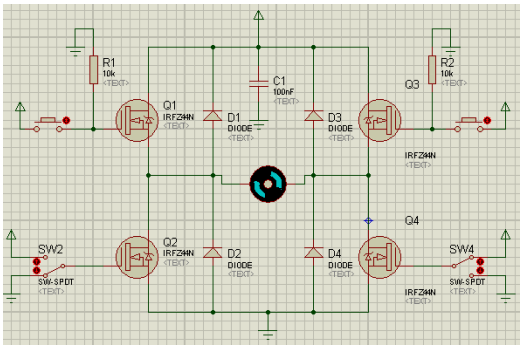


Figura 17 Puente H en "avance"

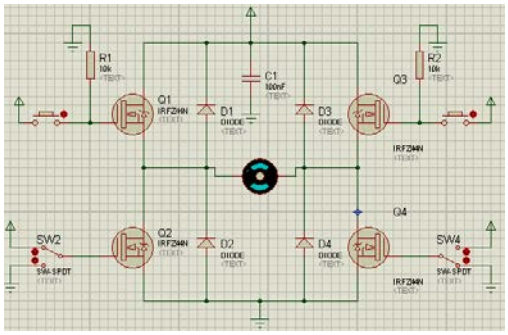


Figura 18 Puente H en "retroceso"

Nótese que los MOSFETS de la parte “baja” son alimentados por switches, mientras que los de la parte “alta” por botoneras. Lo cual es muy semejante a la implementación real de nuestro sistema, pues la parte baja recibe una señal constante (alto o baja) de los pines PD0 y PD1 del Orangutan SV-328. La parte baja en cambio es excitada por pulsos (PWM) generados en la tarjeta. Esto se hizo por supuesto con el fin de evitar problemas de desincronización.

4.2 Implementación

Nótese que los MOSFETS de la parte “baja” son alimentados por switches, mientras que los de la parte “alta” por botoneras. Lo cual es muy semejante a la implementación real de nuestro sistema, pues la parte baja recibe una señal constante (alto o baja) de los pines PD0 y PD1 del Orangutan SV-328. La parte baja en cambio es excitada por pulsos (PWM) generados en la tarjeta. Esto se hizo por supuesto con el fin de evitar problemas de desincronización.

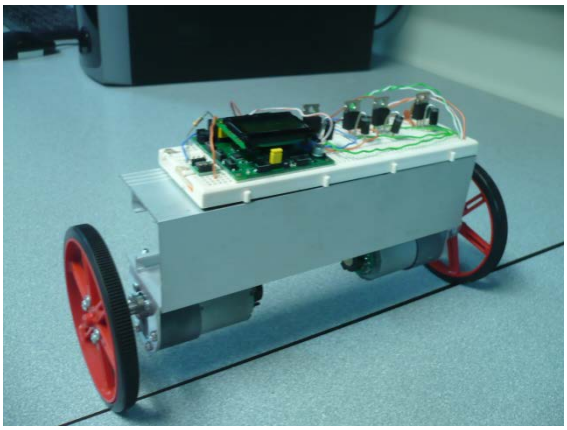


Figura 19 Balancín de dos ruedas con controlador Pololu

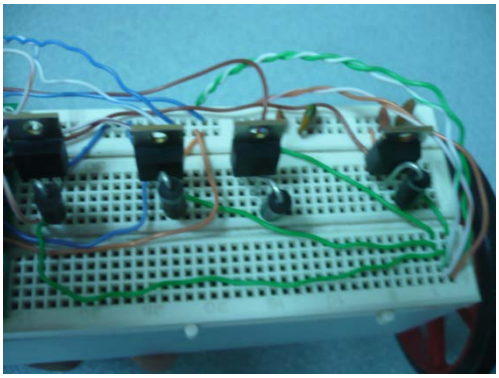


Figura 20 Puente H

Figura 21 Mensaje inicial

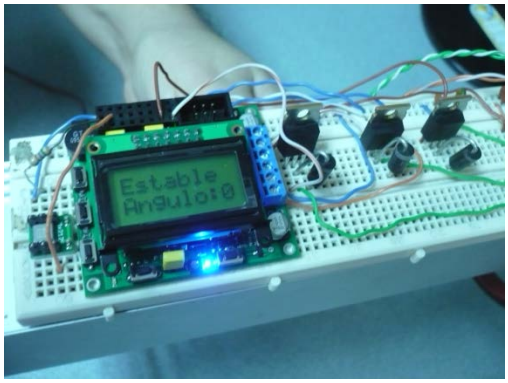


Figura 22 Posición de equilibrio

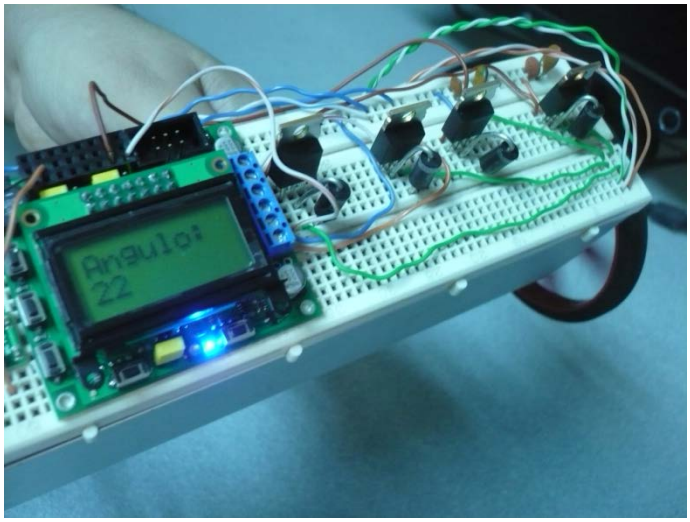


Figura 23 Ángulo positivo

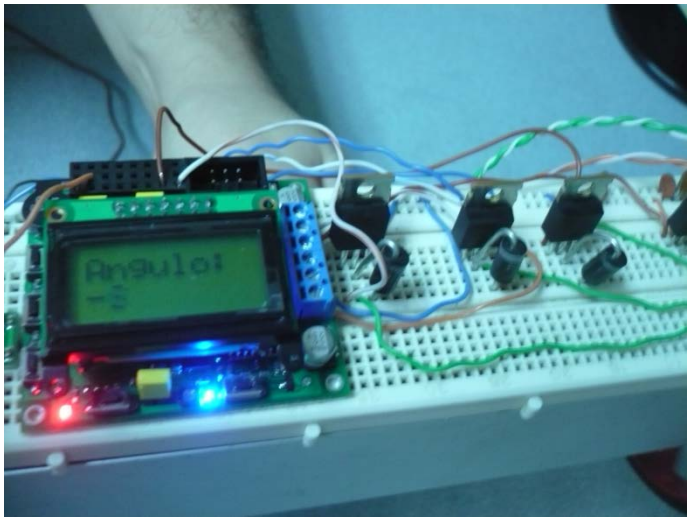


Figura 24 Ángulo negativo