UNIVERSIDAD TECNOLÓGICA METROPOLITANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA Y COMPUTACIÓN

Proyecto Arquitectura de Computadores

Informe Final - Javabot

Mónica Riquelme Vásquez 17664206-8
Gonzalo Zúñiga Palacios 17268591-9
8/21/2013

Informe del proyecto de Arquitectura de Computadores, profesora Karina Harispe Aguilar.
Primer semestre del año 2013.

Índice

Contenidos

Descripción del Proyecto

Nombre del proyecto: Javabot.

Tema: Uso de mando Nunchuk mediante interfaz de Arduino.

Nombre del Grupo: RZ.

En la actualidad, la variedad de usos que tiene la informática en el que hacer de las personas es algo cada vez más común en el diario vivir de las personas. Es por esto, que las utilidades del conocimiento aplicado están ligados en la innovación y creatividad de los desarrolladores.

Como objetivo del ramo Arquitectura de Computadores cursado para este proyecto, es utilizar consolas de videojuegos y utilizarlas de distinta forma para crear aplicaciones creativas reforzando el uso de hardware fuera del común utilizado por un computador de escritorio o laptop. Es con esto, que surgió la idea de utilizar un mando de la consola Nintendo Wii, específicamente el mando Nunchuk, para que haga la función de control para un robot móvil el cual se moverá al manipular éste control. Por lo que se desarrollará en base de un Arduino, donde se conectarán diversos dispositivos electrónicos para coordinar el funcionamiento del robot.

Objetivos

El objetivo general de éste informe es especificar los datos técnicos y de investigación que se utilizaron para poder desarrollar en su totalidad el presente proyecto Javabot. Parte de esta investigación incluye el manejo de Arduino como microprocesador, el ejecutar códigos que realicen las funciones necesarias, la adaptación de motores para poder ser sincronizados en forma electrónica y por último la utilización del Nunchuk de la consola Nintendo Wii para el mando o control del robot. A continuación se detallan los objetivos específicos del proyecto:

- Explicar en qué consiste un motor de corriente continua, sus características de funcionamiento y estructura.
- Como aplicar el control de velocidad a un motor.
- Definir las características del driver L293B y su funcionalidad.
- Especificar la conexión del motor al driver L293B.
- Explicar en qué consiste un Arduino, sus características de funcionamiento y estructura.
- Como implementar el control de velocidad de un motor en el Arduino.
- Definir las características de la programación en el Arduino.
- Especificar la programación hecha en Arduino para el Javabot.
- Conocer las características internas del mando Nunchuk a nivel de conexión.
- Relacionar el mando Nunchuk al uso de esté en el desarrollo del proyecto.
- Realización de pruebas de control.
- Unir todos los componentes para dar vida a Javabot.

Motores corriente continua

El motor de corriente continua es una máquina que convierte la energía eléctrica en energía mecánica, provocando un movimiento rotatorio y liberando calor por la fricción o campo magnético.

Una máquina de corriente continua (generador o motor) se compone principalmente de dos partes, un estátor que da soporte mecánico al aparato y tiene un hueco en el centro generalmente de forma cilíndrica. En el estátor además se encuentran los polos, que pueden ser de imanes permanentes o devanados con hilo de cobre sobre núcleo de hierro para generar campo magnético. El rotor es generalmente de forma cilíndrica, también devanado y con núcleo, al que llega la corriente mediante dos escobillas. También se construyen motores de CC con el rotor de imanes permanentes para aplicaciones especiales.

Sentido de giro

El sentido de giro de un motor de corriente continua depende del sentido relativo de las corrientes circulantes por los devanados inductor e inducido. La inversión del sentido de giro del motor de corriente continua se consigue invirtiendo el sentido del campo magnético o de la corriente del inducido.

Los cambios de polaridad de los bobinados, tanto en el inductor como en el inducido se realizarán en la caja de bornes de la máquina, y además el ciclo combinado producido por el rotor produce la fuerza electromotriz.

El sentido de giro lo podemos determinar con la regla de la mano derecha, la cual nos va a mostrar el sentido de la fuerza. La regla de la mano derecha es de la siguiente manera: el pulgar nos muestra hacia dónde va la corriente, el dedo índice apunta en la dirección en la cual se dirige el flujo del campo magnético, y el dedo medio hacia dónde va dirigida la fuerza resultante y por lo tanto el sentido de giro.

Control de Motores corriente continua

En la actualidad la mayoría de los motores utilizados en la industria son manejados de forma directa desde las líneas de distribución eléctrica, ya sea corriente alterna (CA) o corriente continua (CD). Esto puede ser entendido como que las terminales de los devanados del motor se conectan directamente con las líneas de suministro eléctrico. En estos casos el comportamiento del motor está definido por la naturaleza de la carga que se acople al eje del motor. Para el caso de una carga liviana el motor desarrollara una velocidad relativamente alta y un par de giro bajo pues es el requerimiento de la carga, por el contrario, si se dispone de una carga pesada o difícil de mover, el motor se moverá a una velocidad menor y entregara más par pues una mayor carga lo exige. Como se puede observar al conectar directamente el motor a la red eléctrica AC o CD se define su comportamiento y este se mantendrá inalterable para determinado voltaje fijo de línea de suministro.

Parte del sistema de control de velocidad, que será necesario para el desempeño del proyecto, será mediante la regulación de la tensión del motor (mayor tensión involucra mayor velocidad, o analógicamente, menor tensión involucra menor velocidad o nula) o interfiriendo en el período de los pulsos del motor (control PWM mediante el Arduino).

Driver L293B

Para poder compatibilizar los motores con la placa de Arduino, es necesario poseer un driver que nos proporcione la conexión entre éstos elementos.

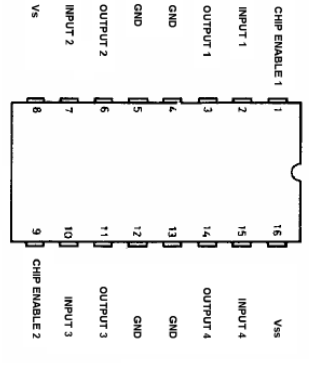
Las propiedades del driver L293B son los siguientes:

- Cuatro canales de salida (habilitados de dos en dos).
- Corriente de salida de hasta 1[A] por canal.
- Señales de control compatibles TTL (max. 7[V]). (conexión directa con el PIC)
- Posibilidad de controlar hasta cuatro motores sin inversión de giro o dos motores con control de giro.
- Posibilidad de alimentación externa de motores de hasta 36[V].
- El modelo L293D incluye diodos de protección internos.

Lo característico de este driver, es que posee dos canales en paralelo, los cuales cada uno sirve para controlar un motor a la vez (ver Tabla conexión de Pines de driver L293B y Diagrama conexión general), pero existe la limitante que cada uno soporta una corriente de 1[A]. Al realizar mediciones, el amperaje promedio por motor era de 1.4[A] por lo que con el uso continuo del dispositivo, era posible de que se quemara la placa, es por eso que utilizarán 2 placas de driver, una por motor.

Para la conexión individual de las placas, se unirán los canales puentiandolos de la siguiente manera:

- 1 con 9
- 2 con 10
- 3 con 11
- 6 con 14
- 7 con 15



Pin	Función
1	Habilitación del puente 1
2	Entrada del amplificador 1
3	Salida del amplificador 1
6	Salida del amplificador 2
7	Entrada del amplificador 2
8	Fuente de alimentación (motores)
9	Habilitación del puente 2
10	Entrada del amplificador 3
11	Salida del amplificador 3
14	Salida del amplificador 4
15	Entrada del amplificador 4
16	Fuente de alimentación (lógica)

Pin	Función
1	PWM
2	DIR 2,3,4,5
3	Motor A
6	Motor B
7	DIR 2,3,4,5
8	Fuente de alimentación (motores)
9	PWM
10	DIR 2,3,4,5
11	Motor A
14	Motor A
15	DIR 2,3,4,5
16	Fuente de alimentación (lógica)

Tabla conexión de Pines de driver L293B.

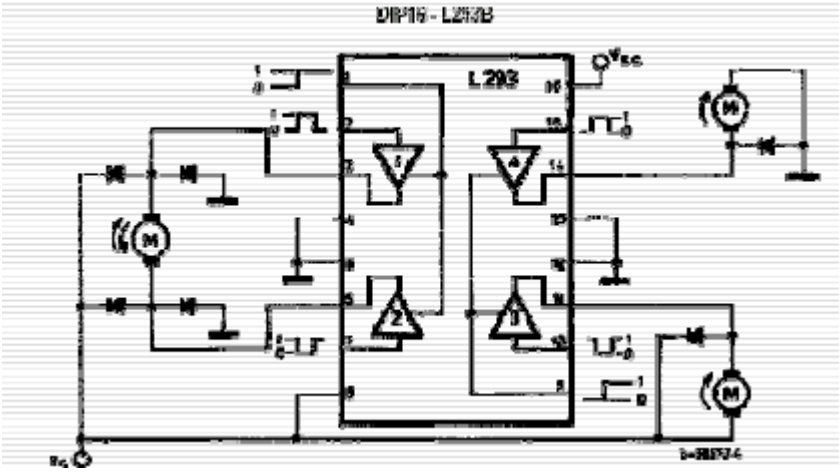


Diagrama conexión general.

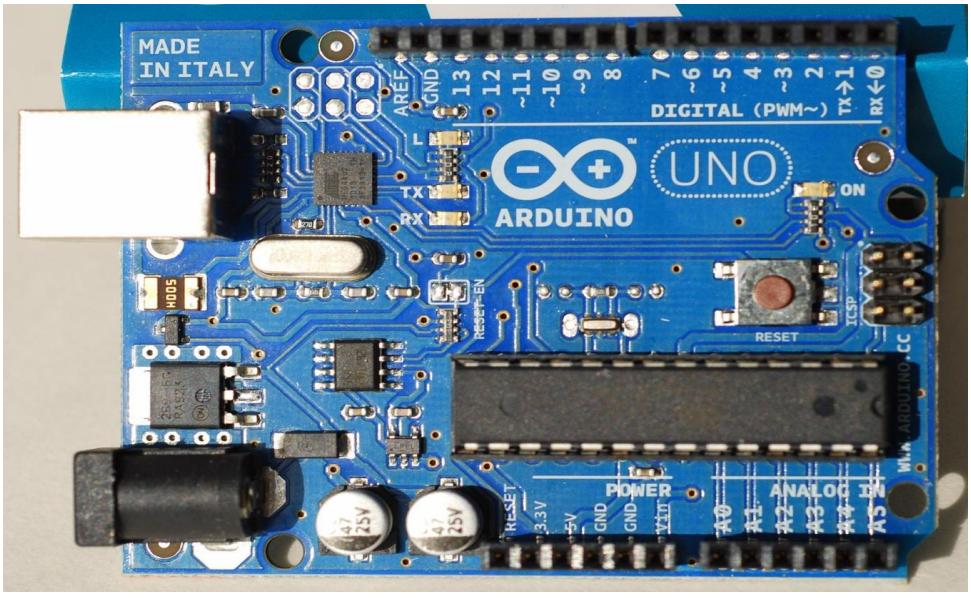
Arduino

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.
El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida.4 Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (*boot loader*) que corre en la placa.
Desde octubre de 2012, Arduino se usa también con microcontroladoras CortexM3 de ARM de 32 bits , que coexistirán con las más limitadas, pero también económicas AVR de 8 bits. ARM y AVR no son plataformas compatibles a nivel binario, pero se pueden programar con el mismo IDE de Arduino y hacerse programas que compilen sin cambios en las dos plataformas. Eso sí, las microcontroladores CortexM3 usan 3.3V, a diferencia de la mayoría de las placas con AVR que usan mayoritariamente 5V. Sin embargo ya anteriormente se lanzaron placas Arduino con Atmel AVR a 3.3V como la Arduino Fio y existen clónicos de Arduino Nano y Pro como Meduino en que se puede conmutar el voltaje.
Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.
Al ser open-hardware, tanto su diseño como su distribución es libre. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.
El proyecto Arduino recibió una mención honorífica en la categoría de Comunidades Digital en el Prix Ars Electrónica de 2006.
En nuestro caso utilizaremos el Arduino UNO ya que es que el tenemos a disposición y es el más básico de los hardwars.

Cuadro informativo **ITEM**	Arduino Uno (retired)
uC	ATmega328
INPUT VOLTAGE	5-12V
SYSTEM VOLTAGE	5V
CLOCK SPEED	16MHz
DIGITAL I/O	14
ANALOG INPUTS	6
PWM	6
UART	1
FLASH SPACE	32Kb
BOOTLOADER	Optiboot
PROGRAMMING INTERFACE	USB via ATMega8U2

Esquema de pines: entradas y salidas

Arduino UNO, consta de 14 entradas digitales configurables entrada y/o salidas que operan a 5 voltios. Cada pin puede proporcionar o recibir como máximo 40 mA. Los pines 3, 5, 6, 9, 10 y 11 pueden proporcionar una salida PWM (Pulse Width Modulation). Si se conecta cualquier cosa a los pines 0 y 1, eso interferirá con la comunicación USB. UNO también tiene 6 entradas analógicas que proporcionan una resolución de 10 bits. Por defecto miden de 0 voltios (masa) hasta 5 voltios, aunque es posible cambiar el nivel más alto, utilizando el pin Aref y algún código de bajo nivel.



Fue a partir del informe anterior que decidimos utilizar las salidas PWM ya que ofrecen un mejor control del ciclo de trabajo del motor con menos potencia; teniendo en cuenta que la otra opción habría sido variar los voltajes entre 0-12V.
La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de *pulse-width modulation*) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.
El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período.

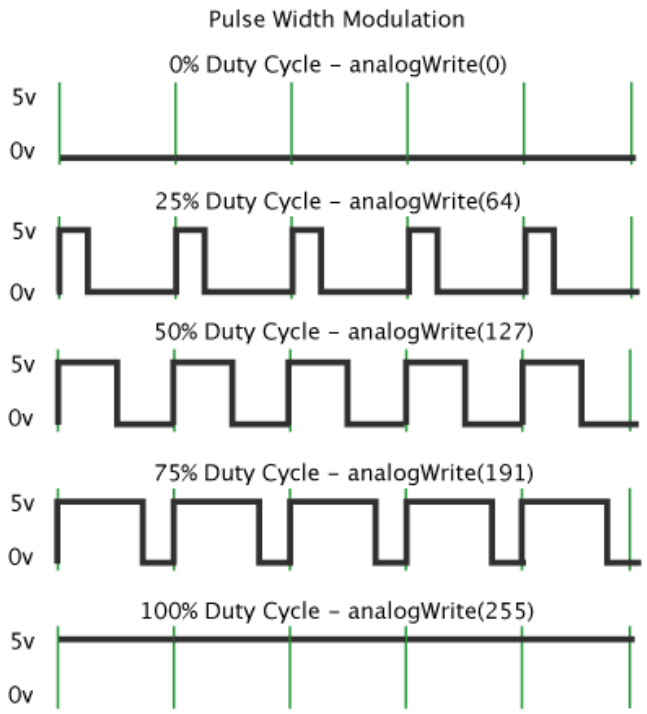
Expresado matemáticamente:

$$D = \frac{\tau}{T}$$

D : Es el ciclo de trabajo.
τ : Es el tiempo en que la función es positiva (ancho del pulso).
T*:* Es el período de la función.

La construcción típica de un circuito PWM se lleva a cabo mediante un comparador con dos entradas y una salida. Una de las entradas se conecta a un oscilador de onda dientes de sierra, mientras que la otra queda disponible para la señal moduladora. En la salida la frecuencia es generalmente igual a la de la señal dientes de sierra y el ciclo de trabajo está en función de la portadora.
La modulación por ancho de pulsos es una técnica utilizada para regular la velocidad de giro de los motores eléctricos de inducción o asíncronos. Mantiene el par motor constante y no supone un desaprovechamiento de la energía eléctrica. Se utiliza tanto en corriente continua como en alterna, como su nombre lo indica, al controlar: un momento alto (encendido o alimentado) y un momento bajo (apagado o desconectado), controlado normalmente por relevadores (baja frecuencia) o MOSFET o tiristores (alta frecuencia).
Otros sistemas para regular la velocidad modifican la tensión eléctrica, con lo que disminuye el par motor; o interponen una resistencia eléctrica, con lo que se pierde energía en forma de calor en esta resistencia. Otra forma de regular el giro del motor es variando el tiempo entre pulsos de duración constante, lo que se llama modulación por frecuencia de pulsos.
La Modulación simula una salida analógica con una salida digital. El control digital se usa para crear una onda cuadrada, una señal que conmuta constantemente entre encendido y apagado. Este patrón de encendido-apagado puede simular voltajes entre 0 (siempre apagado) y 5 voltios (siempre encendido) simplemente variando la proporción de tiempo entre encendido y apagado. A la duración del tiempo de encendido (ON) se le llama Ancho de Pulso (pulse width). Para variar el valor analógico cambiamos, o modulamos, ese ancho de pulso. Si repetimos este patrón de encendido-apagado lo suficientemente rápido por ejemplo con un LED el resultado es como si la señal variara entre 0 y 5 voltios controlando el brillo del LED.

En el gráfico de abajo las líneas verdes representan un periodo regular. Esta duración o periodo es la inversa de la frecuencia del PWM. En otras palabras, con la Arduino la frecuencia PWM es bastante próxima a 500Hz lo que equivale a periodos de 2 milisegundos cada uno. La llamada a la función analogWrite() debe ser en la escala desde 0 a 255, siendo 255 el 100% de ciclo (siempre encendido), el valor 127 será el 50% del ciclo (la mitad del tiempo encendido), etc.



Una vez cargado y ejecutado el ejemplo mueve la Arduino de un lado a otro, lo que ves es esencialmente un mapeado del tiempo a lo largo del espacio. A nuestros ojos el movimiento difumina cada parpadeo del LED en una línea. A medida que la luminosidad del LED se incrementa o atenúa esas pequeñas líneas crecen o se reducen. Ahora estás viendo el ancho de pulso (pulse width).

analogWrite()

Descripción: Escribe un valor analógico (PWM) en un pin. Puede ser usado para controlar la luminosidad de un LED o la velocidad de un motor. Después de llamar a la función analogWrite(), el pin generará una onda cuadrada estable con el ciclo de trabajo especificado hasta que se vuelva a llamar a la función analogWrite() (o una llamada a las funciones digitalRead() o digitalWrite() en el mismo pin). La frecuencia de la señal PWM sera de aproximadamente 490 Hz.
En la mayoría de las placas Arduino (aquellas con el ATmega168 o ATmega328), se podrá generar señales PWM en los pines 3, 5, 6, 9, 10, y 11. No hace faltar configurar el pin como salida para poder usar la función analogWrite(). La función *analogWrite* no tiene ninguna relación con los pines de entrada analógicos ni con la función *analogRead*.

Sintaxis: analogWrite(pin, valor)

Parámetros:

- Pin: Es el pin en el cual se quiere generar la señal PWM.
- Valor: El ciclo de trabajo deseado comprendido entre 0 (siempre apagado) y 255 (siempre encendido).

Devuelve: Nada

Notas y problemas conocidos: Las señales PWM generadas en los pines 5 y 6 poseerán ciclos de trabajo superiores a lo esperado. Esto es así porque para esos dos pines se utiliza el mismo temporizador que se utiliza en las funciones millis() y delay(). Este efecto se notará mucho más en ciclos de trabajo bajos (por ejemplo de 0 a 10) y puede ser que aunque configuremos esos pines con una señal de ciclo de trabajo cero no llegue a ser verdaderamente 0.

Ejemplo: Produce una señal donde conectamos el LED, cuyo ciclo de trabajo es proporcional a la tensión leída en el potenciómetro.

```
int ledPin = 9; // LED conectado al pin digital 9
int analogPin = 3; // potenciómetro conectado al pin 3
int val = 0; // variable en el que se almacena el dato leído
void setup()
{
pinMode(ledPin, OUTPUT); // sets the pin as output
}
void loop()
{
val = analogRead(analogPin); // lee la tensión en el pin
analogWrite(ledPin, val / 4); // los valores de analogRead van desde 0 a 1023 y los
                                valores de analogWrite values van
```

desde 0 a 255, por eso ajustamos el ciclo de trabajo a el valor leído dividido por 4.

}

Programación

A continuación se muestra la programación hecha al Arduino para controlar el motor del Javabot:

```
#include <math.h>
Void setup()
Int MotorI = 9; // motor Izquierdo conectado al pin digital 9
Int MotorD = 10; // motor derecho conectado al pin digital 10
Int dir1_I = 2;
Int dir2_I = 3;
Int dir1_D = 4;
Int dir2_D = 5;
Void loop()
analogWrite(MotorI, 255);
analogWrite(MotorD, 255); // valor 0-255 del cual depende de la velocidad a la que
                          girará el motor.
```

Mando Nunchuk

El mando Nunchuk es un complemento del mando inalámbrico Wii Remote de la consola Nintendo Wii desarrollado para aumentar la experiencia de jugabilidad para esta consola. Éste dispositivo posee dos botones C y Z, un stick analógico y un acelerómetro integrado (figura).

Fig. Mando Nunchuk

Tiene un puerto de conexión que consta de 6 contactos, por defecto se usan sólo 4 ya que con estos, están todas las señales para poder realizar una conexión con un puerto serie y también para utilizarlos con microcontroladores y/o Arduino. En la figura, se muestra la forma y distribución detallada de estos pines de contactos.

A continuación, se detalla las características de cada cable conectado:

1. Cable Verde, datos.
2. No conectado.
3. Cable Rojo, 3.3 [V].
4. Amarillo, clock o reloj.
5. No conectado
6. Blanco, GND.

Fig. Puerto de conexión

Relación con el Proyecto

Como se ha mencionado en las características del mando Nunchuk, su puerto de conexión permite la posibilidad de ser conectado a microcontroladores, específicamente para el proyecto será conectado con un Arduino.

Es así, específicamente se utilizará el stick análogo para que el robot avance, retroceda, doble hacia la izquierda y también hacia la derecha, permitiendo un control total del robot. Es por esto, que fue necesario realizar tres mediciones sobre los ejes coordenados que permite controlar el stick (posee dos potenciómetros de 30K Ω para el eje X e Y), que constan de un estado neutro o en reposo, valor máximo y valor mínimo, los resultados son los siguientes:

Eje Coordenado	Valor Máximo	Valor Central	Valor Mínimo
X	223	127	28
Y	61	-31	-127
Z	80	0	-80

Con estos valores, ya es posible sincronizar las velocidades de los motores del robot con los valores que entreguen los ejes coordenados a través del stick del Nunchuk, pues se relacionará el valor máximo de Y para acelerar y el valor mínimo para retroceder, así también en X para realizar los giros, y posiblemente Z también para realizar giros.

Manual de Usuario

El siguiente manual comprende el uso correcto del robot prototipo Javabot, así como las especificaciones técnicas que posee el dispositivo.

Especificaciones Técnicas

Piezas:

a. Caja metálica
b. Motores DC (x2)
c. Arduino UNO
d. Batería (12[V], 1.3 [AH]).
e. Placa (de control más drivers).
f. Nunchuk Wii.

Características:

a. Caja metálica: caja que protege el interior de Javabot, en su interior se encuentra el Arduino Uno, Placa, Batería. En su parte inferior se encuentran los motores que están conectados a una rueda cada uno y también posee una rueda delantera como eje. Para poder abrir la caja, ésta tiene unos seguros a los costados que al ser presionados permite poder abrirla sin mayor dificultad.
b. Motores: Javabot posee dos motores DC que están conectados a la placa de control. Estos motores están conectados a ruedas que permiten el impulso del robot.
c. Arduino Uno: microcontrolador ubicado al interior de la caja metálica, en el cual están cargados las operaciones que realiza el robot, en él se conecta el mando Nunchuk Wii.
d. Batería: fuente de energía de 12V DC, con capacidad de 1.3 AH. Se encuentra ubicada al interior de la caja metálica, la cual está conectada a la placa de control.
e. Placa de control: en ella es donde se conectan la mayoría de los dispositivos de Javabot, el motor, driver y Arduino. Está ubicada al interior de la caja metálica.
f. Nunchuk Wii: mando que permite el control de Javabot mediente el uso de su stick analógico. Es un dispositivo periférico, pero está conectado a la caja metálica.

Uso correcto de Javabot

Para prolongar la durabilidad del dispositivo, es recomendable seguir las siguientes indicaciones:

a. Utilizar en un espacio amplio y plano para permitir el libre desplazamiento de Javabot, así evitando colisiones que puedan dañar tanto la estructura como el interior de éste.
b. Evitar golpear el mando Nunchuk Wii.
c. No mantenerlo acelerado mientras el cable del Nunchuk Wii está a su máxima capacidad, de lo contrario, puede que se corte o dañe los dispositivos internos de Javabot.
d. No tirar del robot con el cable del mando Nunchuk Wii.
e. Guardarlo en un lugar fresco y seco.

Conclusión

Como conclusión, se destaca el conocimiento adquirido en el aspecto electrónico necesario para desarrollar el proyecto, teniendo que entender y comprender el funcionamiento, conexión e implementación de ciertas componentes para permitir que los dispositivos realizasen las funciones deseadas, todo esto a través de la programación con el Arduino, cosa que también se tuvo que aprender durante el desarrollo del proyecto.

Una de las cosas más complicadas, fue la sincronización de los motores hacia al Arduino, teniendo que fabricar drivers para que pudiesen ser conectados, también la forma de hacer que doble el robot, como lo mencionado en el informe se investigó sobre variación de voltaje y en variar la frecuencia de giro, que fue la alternativa más viable para esto.

En general, como proyecto de asignatura relacionada al uso de hardware, cumple su objetivo ya que se tuvo que utilizar conceptos nuevos de realmente tener que fabricar cosas de forma manual, cosas que claramente salían de la rutina de crear líneas de código en los que se desenvuelven usualmente estudiantes de informática.bb

Hitos del Proyecto

Hitos	Avance Porcentual	Estado Actual
Descripción del proyecto y sus hitos.	100%	Completado
Estudio del Motor cc y driver l293b	100%	Completado
Uso del Arduino y la programación	100%	Completado
Conexión Arduino-Nunchuk	100%	Completado
Implementación y corrección	100%	Completado
Informe Final	100%	Completado

Bibliografía

Motor:

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

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

http://www.slideshare.net/fabricio_salgado_diaz/control-de-velocidad-de-mquinas-de-corriente-continua

http://robots-argentina.com.ar/MotorCC_PuenteH.htm

http://www.iesluisdelucena.es/dpp/docs/presentaciones/Control_de_motores_CC_rev140111.pdf

Driver l293b:

http://www.programarpicenc.com/libro/cap12-l293d-l293b-motores-cc-dc.html

Arduino:

http://www.arduino.cc/es/
http://es.wikipedia.org/wiki/Arduino

Nunchuk:

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

http://electronicavm.wordpress.com/2012/03/29/wii-nunchuk-arduino/