Capítulo 2

Descripción de Componentes Utilizados

2.1 Arduino

Arduino es un prototipo electrónico de plataforma abierta de fácil uso que consta básicamente de una placa de circuito impreso que contiene un microcontrolador de la marca “ATMEL” que cuenta con suficientes entradas y salidas análogas y digitales para cubrir las necesidades del proyecto. Arduino posee una gran cantidad de ventajas a comparación de otros microcontroladores:

- Viene en una placa donde todo está listo para empezar a ser utilizado.
- Es muy fácil de programar
- El costo del microcontrolador no es tan elevado a comparación de otros microcontroladores.

La plataforma de Arduino al ser un sistema de “Open Source” ha permitido que muchas personas se interesen en proyectos novedosos y que por lo mismo compartan sus proyectos en internet, como por ejemplo el uso de un control Nunchuk. Mediante Arduino es más fácil controlar el acelerómetro del Nunchuk juntamente con los dos botones extras y el joystick para aplicaciones futuras.

De esta manera al finalizar esta tesis con el conocimiento obtenido en este proyecto se pueden seguir desarrollando aplicaciones sobre estos sensores tanto como para este vehículo como para otras aplicaciones.

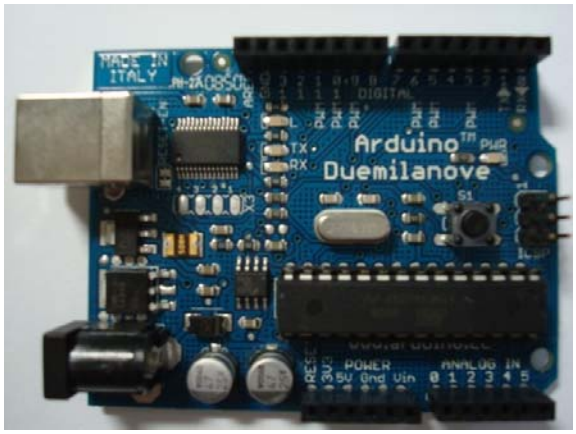

Figura 2.1: Arduino Duemilanove

El sistema Arduino utilizado en esta tesis, se muestra en la figura 2.2, este fue programado mediante el Software: "Arduino 0013", ver figura 2.3, gracias a este programa y al hardware con el que cuenta Arduino fue posible hacer la lectura de los sensores y generar la acción de los actuadores. Arduino está basado en 3 partes principales mediante las cuales este dispositivo se comunica con el exterior: Una parte de entradas y salidas digitales ubicada en la parte superior, una parte de entradas análogas y una parte de pines de voltajes ubicadas en la parte inferior de Arduino, estas se muestran claramente en la figuras 2.4 y 2.5.

Así mismo, la descripción de las terminales con que cuenta el sistema Arduino se describen en las tablas 2.1, 2.2 y 2.3

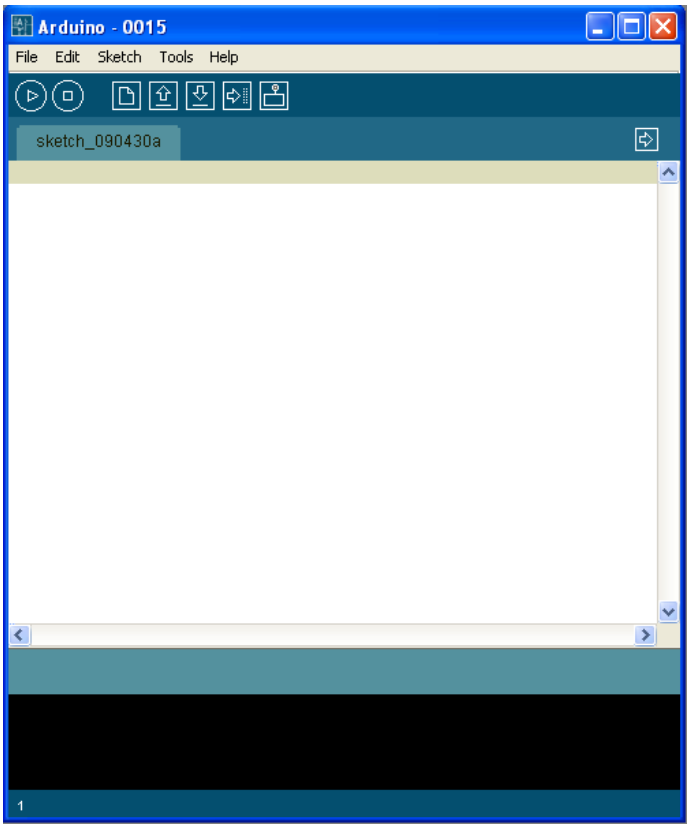

Figura 2. 2: Interfaz del programa Arduino.

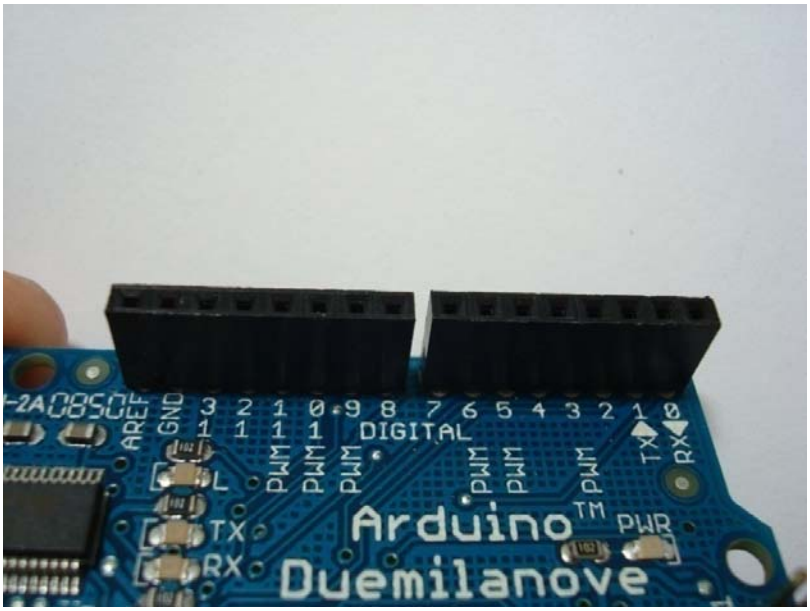

Figura 2. 3: Pines digitales Arduino

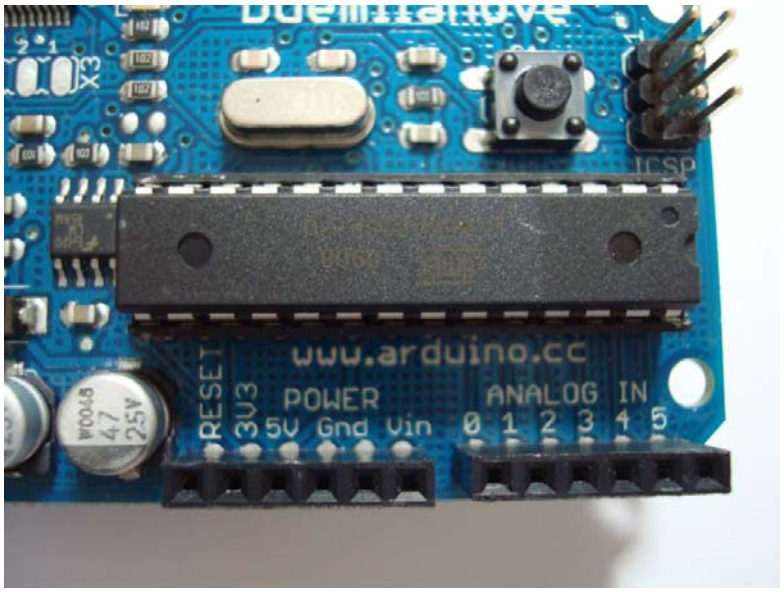

Figura 2. 4: Pines análogos y de voltaje.

Tabla 1: Pines Digitales Usados

Pines Análogos	Aplicación
3	PWM Motor 1
4	Activación motores
5	PWM Motor 2

6	PWM Motor 3
7	Sensor Ultrasónico
9	PWM Motor 5

Tabla 2: Pines análogos Arduino

Pines Análogos	Aplicación
4	Datos Nunchuk
5	Señal reloj Nunchuk

Tabla 3: Pines de Voltaje Arduino

Pines de Voltaje	Aplicación
Gnd.	Tierra
5V	5 Volts Ultrasónico
3V3	3.3 Volts Nunchuk

El sistema Arduino posee dos opciones diferentes de alimentación, una es por medio del cable USB y la otra es mediante una entrada de voltaje la cual puede ser suministrada de entre 5V y 12V para un rendimiento optimo, la primera opción nos permite además de alimentarlo conectarlo a la computadora para así ser programado así como también enviar datos hacia la computadora por medio del puerto serial.

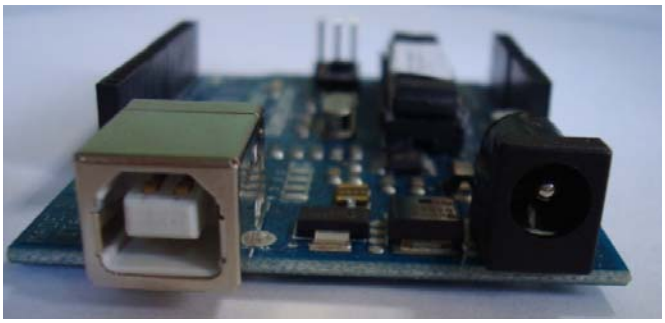

Figura 2. 5: Fuentes de alimentación Arduino. Izquierda: USB, Derecha: Alimentación externa

Con respecto al vehículo propuesto para esta tesis al ser necesario que este en el aire requiere forzosamente que no esté sujetado a nada por lo tanto la opción de conexión por medio de USB para transmisión de datos no estará disponible y no se podrán visualizar las variables utilizadas en la computadora por el momento ya que en esta tesis no se considera el envío de datos de manera inalámbrica sin embargo esto sería posible para futuras aplicaciones.

2.2 Sensores Ultrasónicos

Después de analizar varias opciones para determinar la distancia a la que el vehículo se encuentra con respecto al suelo se decidió utilizar un sensor ultrasónico con la posibilidad de medir hasta 4 metros de distancia con una gran precisión, su funcionamiento básico es el envío de pulsos ultrasónicos los cuales al rebotar contra un objeto producen un eco que es recibido por el sensor para de esta manera determinar la distancia al objeto.

Figura 2. 6: Sensor ultrasónico SRF05.

Este sensor ultrasónico, mostrado en la figura 2.7, tiene 2 maneras para funcionar [7]:

- Modo 1 – Señal de activación y de eco en diferentes pines

  En este modo se utilizan terminales independientes para la señal de activación y para la señal de retorno del eco. Para activar este modo se debe dejar la terminal de modo sin conexión[1].

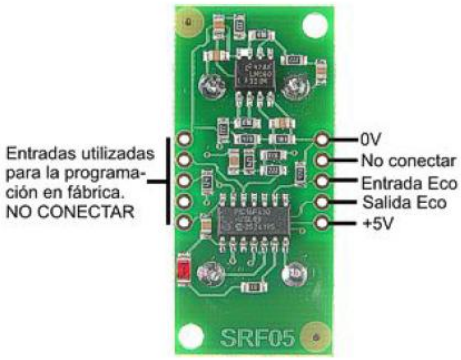

Figura 2. 7: Modo de conexión 1[2].

[1] Información obtenida en la hoja de datos del SRF05.
[2] Figura obtenido gracias a la hoja de datos del sensor en español.

El primer modo es el más fácil de utilizar viéndolo desde la óptica de la programación ya que se cuenta con terminales independientes para la entrada y para la salida, sin embargo si se ve desde el punto de vista de conexiones, implica que salgan dos cables y no uno solo desde el Arduino hacia el sensor SRF05. La figura 2.9 muestra el diagrama de tiempos que debe considerarse cuando el sensor está siendo utilizado en el modo 1.

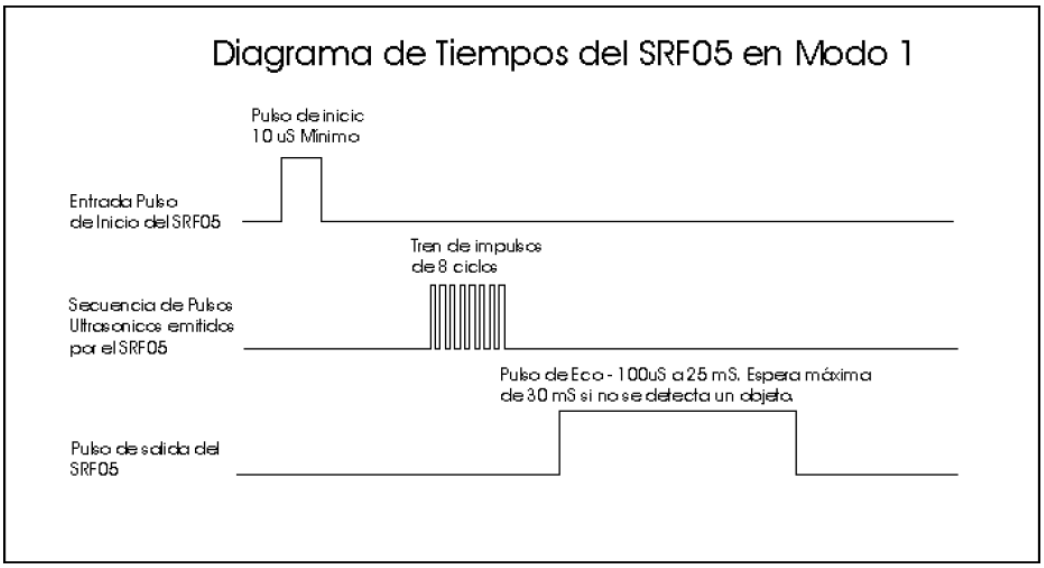

Figura 2. 8: Diagrama de tiempos en modo 1[3].

- Modo 2 – Señal de activación y eco diferente

En este modo se utiliza un solo pin para las señales de activación y eco lo que nos permite reducir el número de terminales que deben utilizarse en el microcontrolador.

[3] Diagrama obtenido gracias a la hoja de datos del sensor en español.

Para utilizar este modo es necesario conectar la terminal de modo a la tierra del Arduino.[4]

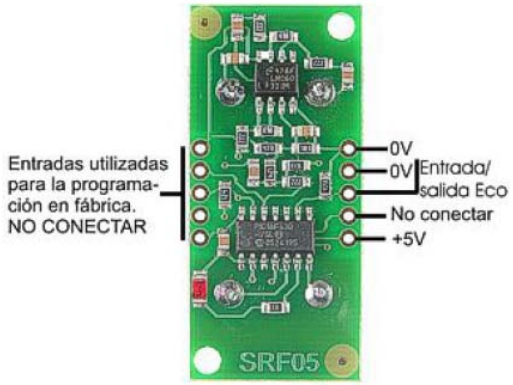

Figura 2. 9: Modo de conexión 2[5]

En este modo la señal de operación y la señal del eco se envían por la misma terminal, por lo tanto es necesario configurar el pin como emisor y receptor teniendo en cuenta el tiempo que se toma para enviar y recibir el pulso. Este pin en un principio debe ser declarado como emisor y solo puede ser cambiado a receptor hasta 700µS después de haber finalizado la señal de activación. El diagrama de tiempos para que el sensor funcione en el modo 2 se muestra en la figura 2.11

[4] Información obtenida en la hoja de datos.
[5] Figura obtenida en la hoja de datos en español

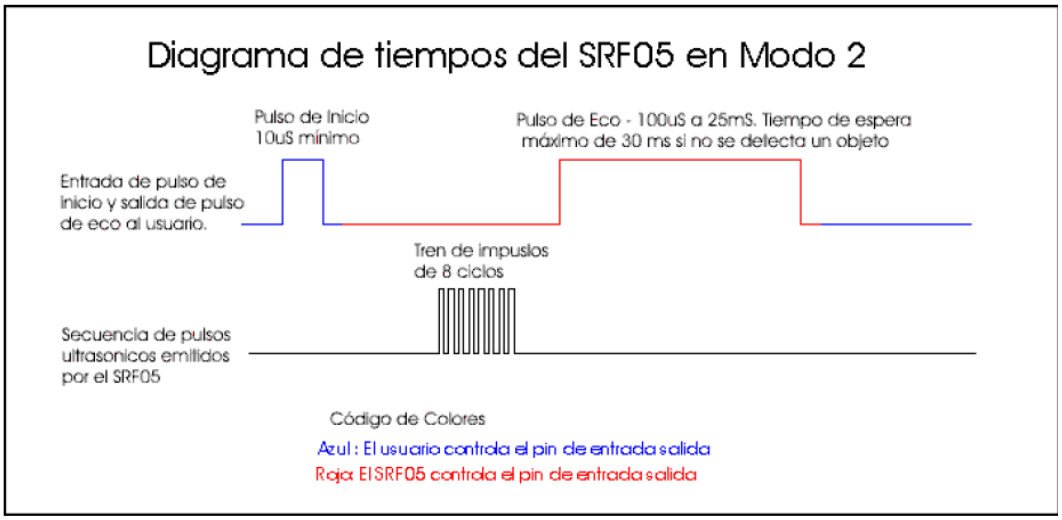

Figura 2. 10: Diagrama de tiempos en Modo 2[6]

Para poder medir la distancia en el Arduino se utiliza una función llamada “Pulse In” la cual lee un pulso en estado HIGH o LOW sobre la terminal apropiada, y según la programación, espera a que el pin se ponga en estado lógico ‘1’ o ‘0’ es decir detecta un cambio de flanco positivo o negativo y cuando éste llega se inicia una cuenta que se parará en cuanto se detecte otro cambio, entonces la función entregará como resultado un valor en microsegundos. Esta instrucción funciona correctamente entre valores de 10µs a 3 minutos [8].

El funcionamiento básico de la instrucción “Pulse in” es el siguiente:

```
int pin = 7;
unsigned long duration;

void setup()
{
  pinMode(pin, INPUT);
}
```

[6] Diagrama obtenido en la hoja de datos en español

```
void loop()
{
  duration = pulseIn(pin, HIGH);
}
```

A partir de esta función se puede calcular la distancia, para esto es necesario tener en cuenta que el eco que se recibe es un pulso por lo tanto su ancho es proporcional a la distancia con respecto al objeto, este sensor si no detecta nada en 30µs su nivel cambia a estado bajo. Para calcular la distancia en cm se debe tener en cuenta que la velocidad del sonido es 340 m/s o 29 microsegundos por centímetro.

El sonido llega hasta el objeto, rebota y regresa como eco por lo que se toma el tiempo en milisegundos y se divide entre 29 para obtener la distancia y después dividimos entre dos porque solo queremos la distancia que a el pulso le tomo regresar o en llegar, no las 2 [9].

Para calcular la distancia en pulgadas es necesario seguir un proceso similar, pero es necesario revisar las hojas de datos, estas nos proporcionan dos datos importantes: el sonido viaja a 1130 pies por segundo y se recorren 73.746 microsegundos por pulgada, por lo tanto para obtener una distancia en pulgadas es necesario dividir la señal que nos da entre 74 para obtener la distancia total y finalmente dividir entre 2 para obtener la distancia del sensor al objeto[7].

En el código del programa solo es necesario aplicar las siguientes fórmulas para obtener las distancias en centímetros y en pulgadas.

[7] Información obtenida con el ejemplo de PING que viene en la página de Arduino.

```
long msPulgadas(long microsegundos)
{
  return microsegundos / 74 / 2;
}

long msCentimetros(long microsegundos)
{

  return microsegundos / 29 / 2;
}
```

El resultado de estas mediciones se nos da por medio del puerto serial por lo que solo es necesario activar la función de monitoreo serial en la interfaz del programa del Arduino. El código completo de cómo medir la distancia con estos sensores se puede ver en el Apéndice A[8].

2.3 Nunchuk

El vehículo que se propuso en esta tesis tiene 4 brazos ubicados a 90° cada uno formando una X, el sistema de control propuesto necesita detectar la inclinación de los motores por lo tanto es necesario un dispositivo capaz de medir al menos dos ejes para tomar dos brazos como el eje 'X' y los otros dos como el eje 'Y'.

[8] Este código se encuentra en el Apéndice A en la pagina 78.

La mayoría de los acelerómetros disponibles en el mercado son para aplicaciones de montaje superficial, aunque el precio de estos dispositivos aumenta según la cantidad de ejes que pueden medir, debido a esto se decidió utilizar el acelerómetro del Nunchuck ya que con este dispositivo puede medir 3 ejes: 'X', 'Y' y 'Z' y tiene la ventaja de estar en un circuito impreso y cuenta con dos botones y un Joystick todo esto en un precio de 400 pesos aproximadamente.

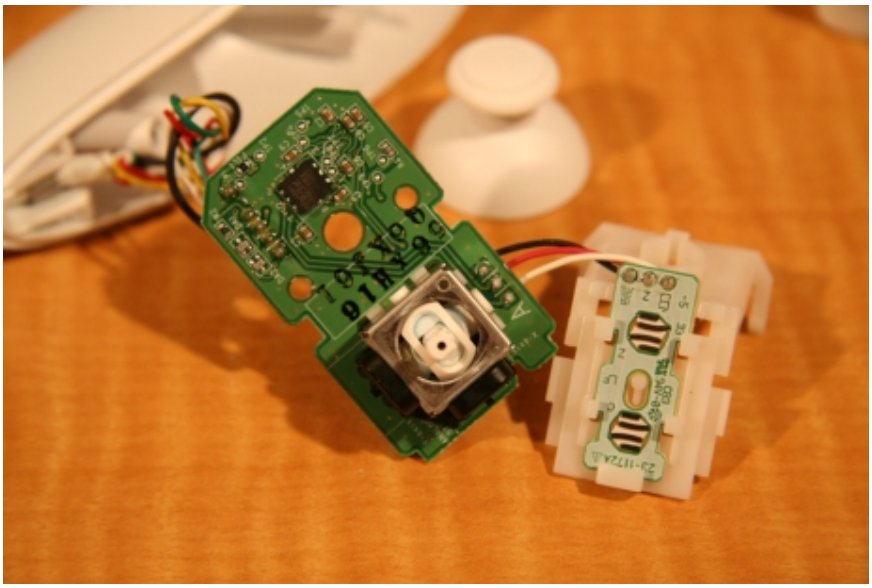

Figura 2. 11: El acelerómetro del Nunchuk, los botones y el joystick.[9]

Figura 2. 12: un acelerómetro comercial montado en placa superficial.

[9] Figura gracias a la página: http://gear.ign.com/articles/748/748146p1.html

En las imágenes 2.12 y 2.13 se puede ver que los acelerómetros son similares y de tamaños reducidos, pero el que viene montado en el circuito impreso del Nunchuk pose muchas ventajas a comparación de un simple acelerómetro.

Un acelerómetro es un dispositivo que mide fuerzas de aceleración estáticas como la constante de gravedad que se ejerce sobre un cuerpo o dinámicas como la inclinación que se aplica sobre un acelerómetro.

Estos dispositivos funcionan porque debido a la medición de aceleración estática debido a la gravedad se puede encontrar un ángulo al que el dispositivo está siendo movido con respecto a la tierra, y debido a la aceleración dinámica se puede saber la dirección con la que se está moviendo [10].

Existen muchos tipos de acelerómetros pero dos de los más importantes y comunes son los de efecto piezoeléctrico y los de efecto capacitivo. Los acelerómetros piezoeléctricos se valen de estructuras microscópicas de cristales los cuales al ser excitados por fuerzas de aceleración producen un voltaje determinado que se usa para saber la inclinación, los acelerómetros de efecto capacitivo contienen micro estructuras separadas una de otra por una distancia determinada por lo que poseen una capacitancia fija, pero estas al ser alejadas o acercadas por efecto de la inclinación producirán un cambio en la capacitancia el cual es usado para saber la inclinación[10].

[10] Información obtenida en Dimension Engineering.

}

El Nunchuk es un control que trabaja en conjunto con otro control de la consola Wii llamado Wiimote. El Nunchuk cuenta con un acelerómetro de 3 ejes, un control análogo o Joystick y dos botones [11].

Figura 2. 13: Control Nunchuk.[11]

Para poder conectarlo al Arduino es necesario saber cuáles son los pines que se tienen que utilizar, este control en su conector cuenta con 6 pines de los cuales solo 4 llegan hasta el circuito impreso donde se encuentra el acelerómetro.

Figura 2. 14: Conector del control Nunchuk.

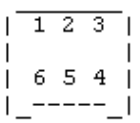

Figura 2. 15: diagrama de conexión del Nunchuk.[12]

[11] Figura gracias a la página: http://www.wiili.org.

Los pines de conexión son los siguientes:

Tabla 4: Pines Nunchuk y su conexión

# de Pin	Descripción	Arduino
1	Datos	Pin 4 análogo
2	No conectado	No conectado
3	3.3 V	3.3 V
4	señal de Reloj	Pin 5 análogo
5	No conectado	No conectado
6	Tierra	Tierra

En internet existen muchas páginas de información que exponen públicamente los programas que usaron para poder utilizar el acelerómetro y los controles del Nunchuk, incluso existen librerías que pueden ser utilizadas para medir la aceleración en el acelerómetro pero no todas estas fuentes son confiables y la mayoría de esas no dicen la manera en la que se conecta el Arduino con el Nunchuk por lo que existe la posibilidad de dañar alguno de ellos.

Debido a este factor existía la posibilidad de no conectar correctamente el Nunchuk y dañarlo, pero investigando [11] se pudo conectar correctamente además de que proporciono un código mediante el cual se había logrado obtener datos provenientes del Nunchuck. La tarea ahora consistía en utilizar correctamente este programa, adaptarlo a las necesidades del proyecto, depurarlo quitando las partes que no se necesitaban, entender cómo funcionaba y en cierta manera mejorar su

[12] Diagrama obtenido en la página: www.wiili.org en la sección "Wiimote/Extension Controllers/Nunchuk"

funcionamiento al cambiar la estructura original dándole un enfoque de programación orientado a objetos.

La estructura de programación renovada se muestra a continuación donde solamente se asignan las tareas a realizar y el código que realiza todo se presenta en la parte final del código para tener todo más ordenado y estructurado.

```
void setup()
{
  beginSerial (19200);
  Wire.begin ();
  Nunchuk_init ();        // iniciación para el Nunchuk
  pulseWidth = minPulse;   // ancho de pulso se reduce al mínimo

}


void loop()
{
  // Intervalo de tiempo deseado para que se active la función
  //de inclinación del Nunchuk

  t++;
  if( t == 25 )
  {

    inclinacion();

  }

  delay(20);

}
```

El código completo de cómo controlar el Nunchuk se encuentre en el apéndice B[13].

Este código nos da los resultados de la inclinación de los 2 ejes que necesitamos, el eje 'X' y el eje 'Y', y debido a las modificaciones realizadas sobre el programa los valores

[13] Código del Nunchuck en Apéndice B en la pagina 80.

entregados ahora están dados en forma de porcentaje en donde el valor de 50 es el centro de los dos ejes.

Anteriormente los valores dados por el Nunchuk eran los siguientes:

Tabla 5: Rango de valores del Nunchuk.

Ejes	Mínimo	Máximo	Rango de valores
X	10	163	153
Y	10	163	153

Estos valores por medio de una regla de 3 fueron adaptados para realizar un porcentaje de inclinación y un porcentaje de PWM para los motores.

$$\%\ inclinacion = \frac{(valorX)(100)}{163} \qquad \%\ PWM = \frac{(ValorY)(255)}{163}$$

Con estas pequeñas funciones agregadas al programa es más fácil administrar las variables que se reciben para poder así controlar los motores y saber la inclinación del dispositivo. La inclinación con respecto a los ángulos y al valor de porcentaje de inclinación entregado por medio del Arduino es el siguiente:

Tabla 6: Inclinación del eje X

x<50°		x>50°	
Inclinación	PWM	Inclinación	PWM
0°	50	0°	50
2.1°	49	1°	51
2.4°	48	1.7°	52

3.3°	47	2.8°	53
4.5°	46	4.7°	54
5.5°	45	5.9°	55
6.5°	44	6.9°	56
7.5°	43	7.9°	57
8.5°	42	8.9°	58
9.5°	41	9.9°	59
10.5°	40	10.9°	60
12.6°	39	12.9°	61
13.9°	38	13.9°	62
14.8°	37	15.3	63
15.9°	36	16.3	64

En la tabla 7 se puede observar cómo cambian los datos de porcentaje de inclinación del acelerómetro en donde los valores menores a 50 significan que el acelerómetro está siendo girado a la izquierda y los valores mayores a 50 significan que este está inclinándose hacia la derecha.

Figura 2. 16: Inclinación del eje X del acelerómetro.

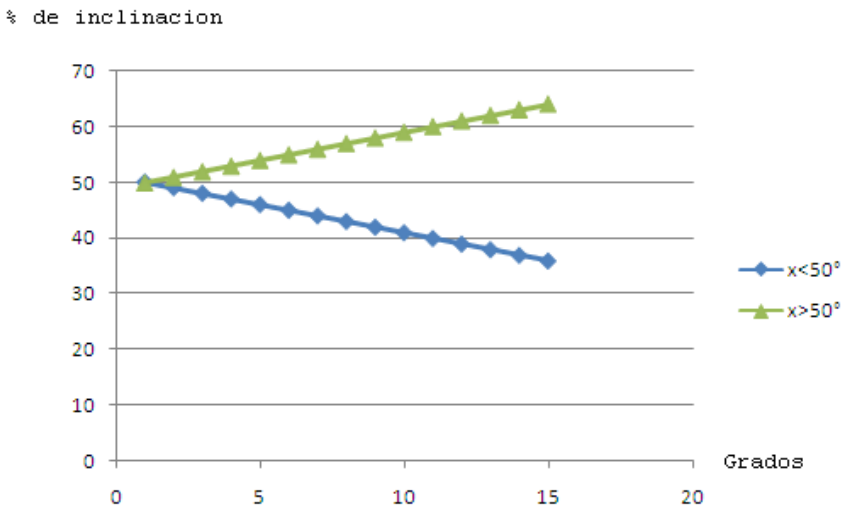

Figura 2. 17: Grafica del comportamiento del eje X.

En la grafica mostrada en la Figura 2.18 se puede ver comportamiento con relación al ángulo de inclinación del acelerómetro y del % que se maneja en el programa para la inclinación desde 0 a 100.

Tabla 7: Inclinación del eje Y

Y<50	Columna1	Y>50°	Columna2
0°	50	0°	50
2.6°	49	0.3°	51
3.8°	48	1.8°	52
4.7°	47	2.7°	53
5.8°	46	3.9°	54
6.8°	45	4.8°	55
7.9°	44	5.9°	56
8.9°	43	6.9°	57
9.9°	42	7.9°	58
11.1°	41	8.9°	59
12.2°	40	9.9°	60
14.3°	39	12.3°	61
15.5	38	13.3°	62

16.5	37	14.3°	63
17.6	36	15.7°	64

Los datos en la tabla 8 muestran los valores obtenidos para la inclinación del acelerómetro respecto al eje Y además del porcentaje de inclinación, donde los valores menores a 50 significan que el acelerómetro está siendo girado hacia atrás y los valores menores a 50 significan que este está inclinándose hacia adelante.

Figura 2. 18: inclinación del acelerómetro en eje Y.

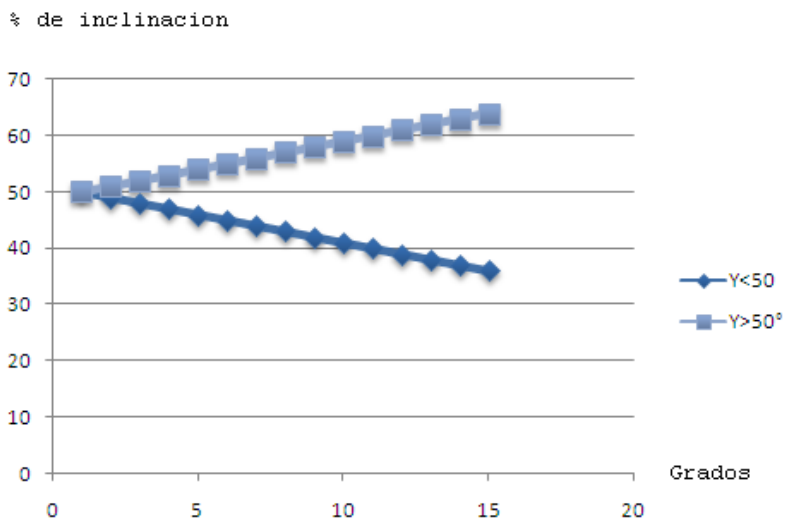

Figura 2. 19: Grafica del comportamiento del eje X.

2.4 Motores

En esta tesis se utilizaron dos diferentes tipos de motores, los primeros fueron de DC especiales para aviones de radio control, elegidos así para que pudieran proveer la fuerza necesaria para levantar el prototipo, los segundos motores que se utilizaron fueron sin cepillado de uso exclusivo para helicópteros de 4 hélices a los lados como en nuestro prototipo, estos motores poseen una mayor capacidad por lo que su uso es el indicado para esta tesis.

Figura 2. 20: Motores de DC para aviones RC.

Figura 2. 21: Motores Brushless.

2.4.1 Motores DC

El primer tipo de motores con cepillado tienen un peso de 10 gr con un diámetro de 28 mm y un largo de 5 cm, estos se alimentan con un voltaje variable de entre 6 y 12 volts idealmente, aunque fueron probados a 18 volts para ver si de esta manera los 4 juntos podían levantar la estructura sin el uso de piñones y engranes para conectar las hélices.

Este tipo de motores son dispositivos electromotrices que convierten la energía eléctrica en movimiento mecánico, estos basan su funcionamiento en la acción de campos magnéticos opuestos que hacen girar el rotor (eje interno) en dirección opuesta al estator (imán externo o bobina), con lo cual cuando éste es sujetado por su superficie el eje del rotor es lo único que gira y de esta manera es como generamos el movimiento hacia las hélices [12].

Para cambiar la dirección del motor solo es necesario cambiar la polaridad de alimentación, pero para hacer que este varíe su velocidad sin perder fuerza es necesario controlarlo por medio de PWM y un puente H.

Tabla 8: Características motor con cepillado

Corriente:	2 A
Peso:	70 gr
Diámetro:	28 mm
Largo:	50 mm

Estos motores a 18 volts tenían un pico de corriente de 2.6 A y se estabilizaban a 2 A por lo cual el control de estos motores se podía realizar por medio de un puente H de la familia L293.

2.4.2 Motores Brushless

Los motores sin cepillado están hechos para una gran eficiencia y un bajo peso debido a que son para uso específico de aeromodelismo, los 4 motores que se utilizaron fueron de la marca "HIMAX" modelo "HC2812-0650"[14]. Estos motores tienen las siguientes características:

Tabla 9: Características Motor Brushless

Potencia:	150 W
I del ESC:	10 A
Peso:	64 gr
Diámetro:	28 mm
Largo:	40 mm

Los motores sin cepillado al tener una gran potencia y consumir una gran cantidad de corriente se tienen que controlar por un ESC (Electronic Speed Controller) y no por no por un puente H.

Este tipo de motores, a diferencia de los otros no utilizan escobillas para realizar el cambio de polaridad en ellos si no que lo hacen mediante un proceso de conmutación electrónica y no mecánica. Los motores al tener escobillas disminuyen el rendimiento, producen calor y ruido.

Los motores sin cepillado se alimentan con una señal trifásica que realmente es una combinación de pulsos que hacen que la señal de DC tenga una componente de AC, pero se les clasifica como motores de DC porque poseen imanes permanentes los

[14] Datos de los motores obtenidos en la página de Dragan Fly.

cuales reciben pulsos en una determinada manera haciendo que el campo secuencial gire más rápido o más lento al aumentar o disminuir la frecuencia de los pulsos.

En los motores convencionales al aumentar el voltaje se aumenta la velocidad pero en estos no, aquí los ESC subministran la corriente necesaria para la velocidad que se elija sea fija [13].

Algunas ventajas principales de estos dispositivos son las siguientes:

- Mayor eficiencia
- Mayor rendimiento
- Conmutación electrónica
- Mayor potencia para el mismo tamaño
- Rango de velocidad elevado al no tener limitación mecánica.

2.5 Controles Electrónicos de Velocidad (Electronic Speed Controller)

Un ESC es usado principalmente para controlar motores por medio de radio control, pero en este caso lo utilizaremos solamente para poder variar la velocidad del motor

por medio de una señal PWM proveniente del Arduino. Los ESC trabajan principalmente por medio de FET's (Field Effect Transistors) debido a que son más rápidos para hacer transiciones además de que pueden soportar una corriente de entre 15 a 25 A, lo cual no se puede lograr fácilmente con puentes H. El ESC que se utilizo en esta tesis fue el modelo: "Phoenix 25" de la marca "Castle Creations" el cual tiene las siguientes características [14]:

Tabla 10: Características ESC

Voltaje	12.6 a 19.2 V
Corriente:	25 A
Conmutación:	11 000 Hz
Peso:	17 gr
dimensiones:	3.5 x 2 x .4 cm

Estos ESC tienen la característica de ser controlados por microprocesadores lo que permite un mejor desempeño de los motores, cuentan también con sistemas de seguridad y de limitación de corriente para que en dado caso de que a algún motor le llegue a faltar la señal de PWM automáticamente se apague para no dañarse.

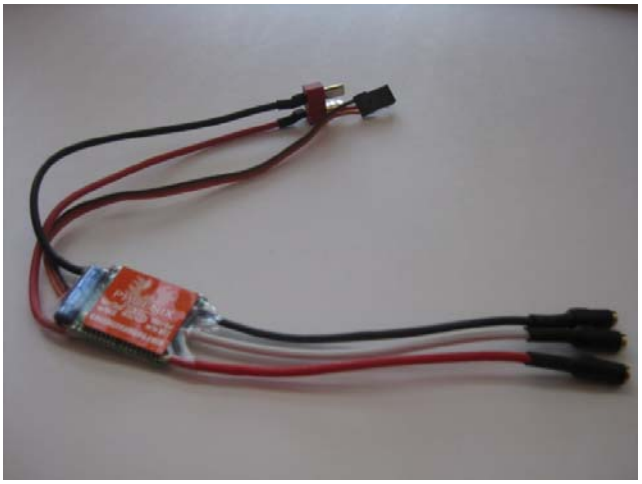

Figura 2. 22: ESC.

Los reguladores de velocidad (ESC) para su funcionamiento básicamente miden el tiempo de 1ms a 2ms como un servo estándar y lo transforman en un máximo y mínimo de Rpm para el motor[15].

2.6 Puente H

Un puente H es un circuito electrónico que nos da la posibilidad de girar en ambos sentidos un motor eléctrico de corriente directa. El nombre de puente H de debe a la colocación de 4 interruptores y dependiendo de la combinación de estos se puede realizar que el motor gire en una dirección o en otra.

[15] Datos obtenidos de la página E – Radio control.

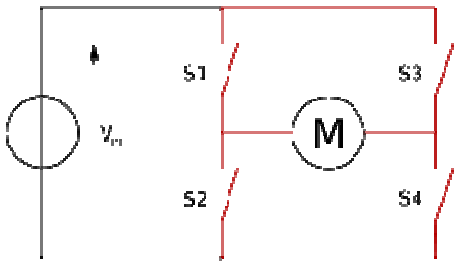

Figura 2. 23: Puente H.

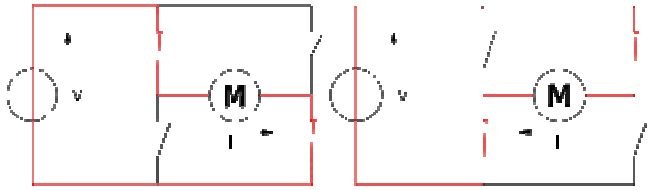

Figura 2. 24: Acciones de los interruptores en un puente H [15].

En la Figura 2.25 el primer circuito hace que el motor gire en una dirección debido al voltaje recibido, en cambio en la otra figura el voltaje entra al motor inversamente y hace que este gire en la otra dirección. Una pequeña tabla muestra las acciones correspondientes en los interruptores:

Tabla 11: Acciones de los interruptores en puente H

S1	S2	S3	S4	Acción
1	0	0	1	El motor gira en avance
0	1	1	0	El motor gira en retroceso
0	0	0	0	El motor se detiene por inercia
0	1	0	1	Frenado rápido de motor

Para poder controlar un motor es recomendable usar un puente H para cada motor ya que estos alimentados a 18 V tienen una corriente de pico de 2.6 A y una corriente estable de 2 A. Al principio se utilizaron puentes H del modelo “L239D” que incluye una protección de diodos y la posibilidad de controlar 4 motores, sin embargo se utilizo un puente H para cada motor y no limitar la corriente en el integrado pero de todas maneras ese modelo de puente H no daba los requerimientos de corriente así que se prosiguió a utilizar el modelo L293B el cual al no tener una protección de diodos puede soportar las necesidades de corriente de los motores [16].

A continuación se presenta un diagrama que muestra cómo conectar fácilmente los motores mediante los puentes H y el control PWM proveniente del Arduino.

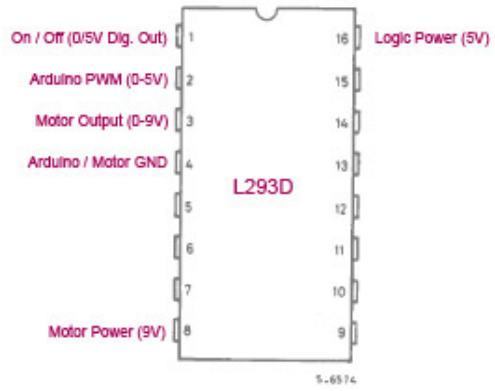

Figura 2. 25: Esquemático de un motor, un Arduino y un puente H L239D [17].

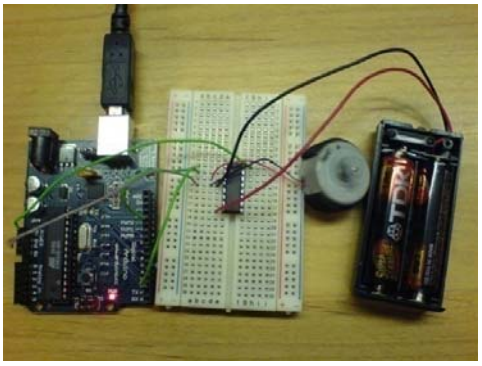

Figura 2. 26: Motor, Arduino y puente H L239D[16].

2.7 PWM

PWM son las iníciales en inglés de Pulse Width Modulation cuya traducción es Modulación por Ancho de Pulso, esta principalmente sirve para controlar el ciclo de trabajo de una señal periódica, con fines de transmitir información a través de un canal de comunicaciones o para controlar de la cantidad de energía que se envía

a una carga y un ejemplo de esto es que los motores que se utilizaron es esta tesis fueron controlados por PWM.

[16] Figura de la conexión sugerida por la página : http://gestaltung.fh-wuerzburg.de

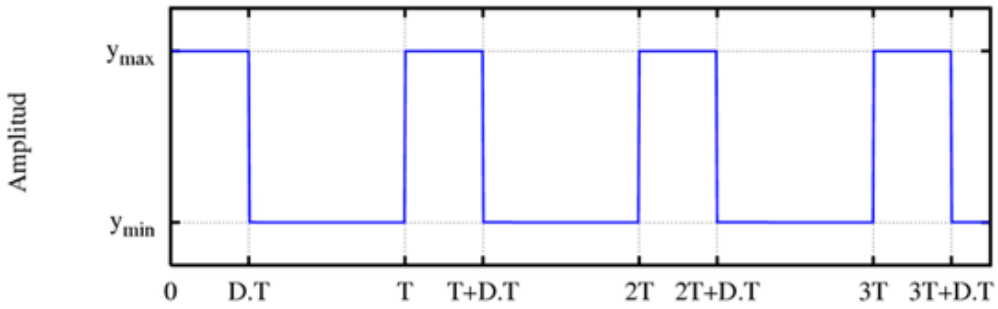

Figura 2. 27: Control PWM de una señal [18].

La frecuencia de una señal se puede definir como la cantidad de pulsos en estado encendido o apagado que hay por segundo, este también es el inverso del periodo y viceversa.

$$f = \frac{1}{T} \qquad T = \frac{1}{f}$$

El periodo se mide en segundos y la unidad de medida son los Hertz los cuales son el inverso de la unidad de tiempo, los segundos.

Un parámetro importante dentro del PWM es el ciclo de trabajo que en inglés es “Duty Cycle” el cual determina el porcentaje de tiempo que el pulso está en estado activo durante un ciclo. La señal PWM se utiliza para controlar circuitos analógicos. El periodo

y la frecuencia del tren de pulsos pueden determinar la potencia entregada a dicho circuito.

En un Arduino la señal de salida PWM nos permite cambiar el ciclo de trabajo o el tiempo que el pulso está activo utilizando la función analogWrite(). El siguiente programa es un muestra sencilla de cómo se pueden modificar intervalos de tiempo a través de la función digitalWrite() [19].

```
int digPin = 10; // pin digital 10
void setup()
{
    pinMode(digPin, OUTPUT);    // pin en modo salida
}
void loop()
{
    digitalWrite(digPin, HIGH); // asigna el valor HIGH al pin
    delay(500);                 // espera medio segundo
    digitalWrite(digPin, LOW);  // asigna el valor LOW al pin
    delay(500);                 // espera medio segundo
}
```

El programa anterior es la forma más básica de aplicar un control PWM a cualquier pin del Arduino, pero existe una función específica que solo se puede aplicar a 6 pines digitales del Arduino (3, 5, 6, 9, 10 y 11), esta función se llama “analogWrite()” [20].

Con esta función lo que se logra es generar una señal cuadrada estable de un determinado ciclo de trabajo hasta que se vuelva a aplicar la misma función en el pin, los valores del ciclo de trabajo van desde:

0 = Siempre apagado 255 = Siempre encendido

0 = 0 Volts 255 = 5 volts

El funcionamiento básico del control de motores es el siguiente:

```
int Motor = 9;        // Motor conectado en pin 9
int val = 0;          // valor entre 0 y 255

void setup()
{
  pinMode(Motor, OUTPUT);   // Pin de motor como salida
}

void loop()
{
  analogWrite(Motor, val);  //Se asigna el ciclo de trabajo al motor
}
```

El código completo con el cual se controlaron los cuatro motores viene en el apéndice[17].

[17] Código utilizado para controlar los 4 motores en el apéndice C en la Página 84.