Escola Politècnica Superior de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# TRABAJO FINAL DE CARRERA

**TÍTULO DEL TFC: Estudio de las posibilidades de un Wiimote como interfaz ce control**

**TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad en Sistemas de Telecomunicación.**

**AUTOR: Miguel Ángel Granado Navarro**

**DIRECTOR: Marcos Quílez Figuerola**

**FECHA: 28 de julio de 2009**

**Título:** Estudio de las posibilidades de un Wiimote como interfaz de control

**Autor:** Miguel Ángel Granado Navarro

**Director:** Marcos Quílez Figuerola

**Fecha:** 28 de julio de 2009

**Resumen**

El objetivo de este proyecto es explorar las posibilidades del Wiimote como interfaz de control, considerando su uso como herramienta docente. Tanto el Wiimote, como Arduino, pueden ser, una solución atractiva a muchos de los problemas de aprendizaje e interacción del alumno con la tecnología. Se realiza un estudio del diseño del Wiimote, así como el análisis de su integración en diseños electrónicos.

El trabajo está estructurado en 3 capítulos. En el primero de ellos se presentan las características y especificaciones técnicas del mando de la Wii (Wiimote), detallando cada uno de los componentes que lo conforman. Además, también se realiza una breve descripción del principio de funcionamiento del mando, basado básicamente en el acelerómetro y en la conexión inalámbrica.

A continuación, en el segundo capítulo, se muestran una serie de aplicaciones en las que el Wiimote interviene como HID (Dispositivo de Interfaz Humana). Una de las aplicaciones consiste en el control de una estructura de NXT Lego. Esta estructura tiene definidos tres movimientos relativos a dos ejes, y uno de los botones del Wiimote. La aplicación se desarrolla con el entorno de programación LabVIEW.

Por último, en el tercer capítulo, se da una visión del Wiimote (Nunchuck), como interfaz de control de prototipos hardware. Se diseñan dos aplicaciones desarrolladas sobre la plataforma Arduino. En ellas, se desea controlar el movimiento de unos motores, mediante cableado o de forma inalámbrica. En este último caso, para la comunicación inalámbrica se utilizan dos módulos XBee, basados en la tecnología ZigBee, la cual permite comunicaciones de hasta 100 metros, con unos requerimientos de potencia mínimos.

Finalmente se incluye un conjunto de referencias seleccionadas, y contrastadas, principalmente páginas Web, que contienen información relevante para el desarrollo de este proyecto y de otros trabajos que puedan abordarse en el futuro.

**Title:** Study of the possibilities of Wiimote like control interface

**Author:** Miguel Ángel Granado Navarro

**Director:** Marcos Quílez Figuerola

**Date:** July, 28th 2009

## Overview

The objective of this project is to explore the possibilities of the Wiimote like control interface, considering its use like educational tool. As much the Wiimote, as Arduino, can be an attractive solution to many of the problems of learning and interaction of the student with the technology. A study of the design of the Wiimote, as well as the analysis of its integration in electronic designs is made.

The work is divided in three chapters. In the first chapter the characteristics and engineering specifications of the Wii control are introduced (Wiimote), detailing each one of the components that make the system up. In addition, a brief description of the principle of the control operation is set forth, basically based on the accelerometer and the wireless connection.

The second chapter, corresponds to series of applications in which the Wiimote takes part like HID (Human Interface Device). One of the applications consists of the control of a NXT Lego structure. This structure has three defined movements regarding two axes, and one regarding the Wiimote bellboys. The application is developed with the environment of LabVIEW programming.

Finally, in the third chapter, a vision of the Wiimote occurs (Nunchuck), like interface of control of prototypes hardware. Two applications developed with the Arduino platform are designed. In them, it is desired to control the movement of motors, by means of wiring or of wireless form. In this last case, for the wireless communication two XBee modules are used, based on the ZigBee technology, which allows communications of up to 100 meters, with minimum power requirements.

Finally a set of selected and contrasted references is included, mainly Web pages, which contain excellent information in the development of this project and others works than can be approached in the future.

# ÍNDICE


**INTRODUCCIÓN Y OBJETIVOS ..... 1**

**CAPÍTULO 1. ANÁLISIS DE LAS CARACTERÍSTICAS FÍSICAS Y TÉCNICAS DEL WIIMOTE ..... 2**

**1.1. Introducción ..... 2**

**1.2. Características técnicas. Componentes ..... 2**
1.2.1. La memoria ..... 2
1.2.2. Bluetooth ..... 4
1.2.3. Cámara de infrarrojos ..... 4
1.2.4. El vibrador (rumble) ..... 5
1.2.5. Los botones ..... 5

**1.3. Funcionamiento ..... 6**

**1.4. Comunicación ..... 7**

**1.5. Nunchuck. Expansión del Wiimote ..... 8**
1.5.1. Características técnicas. Componentes. ..... 8
1.5.2. Comunicación con el Wiimote ..... 9
1.5.3. Cableado y conexionado ..... 9

**CAPÍTULO 2. WIIMOTE COMO DISPOSITIVO DE INTERFAZ HUMANA (HID) PARA PC ..... 11**
2.1. Introducción ..... 11
2.2. Wiimote como mouse del PC ..... 11
2.3. Aplicación GlovePIE ..... 12
2.4. Pizarra o pantalla interactiva ..... 14
2.5. Control remoto de un NXT Lego ..... 16
2.5.1. Entorno de programación LabVIEW ..... 16
2.5.2. Librerías NXT Toolkit ..... 17
2.5.3. Librería WiimoteLib ..... 19
2.5.4. Detalle de la Aplicación ..... 21
2.5.5. Descripción del proceso ..... 24

**CAPÍTULO 3. WIIMOTE COMO INTERFAZ DE CONTROL DE PROTOTIPOS HARDWARE ..... 27**

**3.1. Aplicaciones desarrolladas con el Nunchuck ..... 27**
3.1.1. Control de una cortina motorizada ..... 27
3.1.1.1. Entorno de programación Arduino ..... 27
3.1.1.2. Librerías de Arduino ..... 32
3.1.1.3. Descripción del proceso ..... 34
3.1.2. Control remoto de una cortina motorizada ..... 40
3.1.2.1. Módulos XBee y la tecnología Zigbee ..... 40
3.1.2.2. Descripción del proceso ..... 42

**CONCLUSIONES** ..... **49**

**ESTUDIO DE AMBIENTALIZACIÓN** ..... **50**

**BIBLIOGRAFÍA** ..... **51**

**ANEXOS** ..... **52**

**ANEXO 1:** Acelerómetro del Wiimote, modelo ADXL330 ..... 53

**ANEXO 2:** Acelerómetro del Nunchuck, modelo LIS3L02AL ..... 54

**ANEXO 3:** Especificaciones técnicas del motor de la cortina suministrada por Bandalux ..... 56

**ANEXO 4:** Driver SN754410 ..... 57

**ANEXO 5:** Sistema completo del control de la estructura NXT LEGO, mediante el Wiimote ..... 58

**ANEXO 6:** Referencia de los proyectos realizados por Johny Cheng Lee ..... 59

**ANEXO 7:** Referencia de la aplicación GlovePIE ..... 61

**ANEXO 8:** Referencia de la adquisición de datos entre Nunchuck y Arduino ..... 67

# ÍNDICE DE FIGURAS

**Figura 1.1.** Esquema del circuito electrónico del Wiimote .....6
**Figura 1.2.** Esquema del acelerómetro ADXL330 .....6
**Figura 1.3.** Esquema de funcionamiento de la pieza de silicio .....7
**Figura 1.4.** Vista superior del joystick (a la izquierda) y acelerómetro ST Microelectronics LIS3L02AL (a la derecha) .....8
**Figura 1.5.** Detalle del bus I2C del Nunchuck.....10
**Figura 2.1.** Pantalla de control del programa WiinRemote .....12
**Figura 2.2.** Fotografía del montaje de la pantalla interactiva .....14
**Figura 2.3.** Esquema del montaje de un lápiz de infrarrojos.....15
**Figura 2.4.** Pantalla principal del programa Wiimote WhiteBoard.....15
**Figura 2.5.** Detalle de las diferentes librerías instaladas en LabVIEW .....18
**Figura 2.6.** Contenido de bloques de la Librería NXT Toolkit: NXT Toolkit (1) y NXT Direct Commands (2). Bloques de salida dentro de la librería Direct Commands (3).....19
**Figura 2.7.** Esquema del estado del Wiimote en LabVIEW .....20
**Figura 2.8.** Diagrama de bloques del establecimiento y desconexión de la comunicación con el NXT. .....21
**Figura 2.9.** Array de elementos que contiene el estado del acelerómetro y los botones.....22
**Figura 2.10.** Diagrama de bloques a realizar cuando la condición de entrada proveniente del array es True .....22
**Figura 2.11.** Esquema del umbral de sensibilidad de cada eje del acelerómetro.....23
**Figura 2.12.** Fotografía de la estructura del NXT Lego, con sus posibles movimientos.....24
**Figura 2.13.** Panel frontal de la Aplicación en LabVIEW .....26
**Figura 3.1.** Distribución de componentes de la placa Arduino.....28
**Figura 3.2.** Duemilanove .....28
**Figura 3.3.** Diecimila.....28
**Figura 3.4.** Bluetooth .....29
**Figura 3.5.** Mega .....29
**Figura 3.6.** Nano.....29
**Figura 3.7.** Pantalla principal de la aplicación Arduino – 0013 .....30
**Figura 3.8.** Imagen del chip FTDI contenido en la placa Arduino .....31
**Figura 3.9.** Esquema de conexionado de pins del chip ATmega8.....32

**Figura 3.10.** Fotografía del sistema completo de control de una cortina con la plataforma Arduino XBee. ..... 34

**Figura 3.11.** Diagrama de conexionado del chip SN754410 (a la izquierda), relación de entradas y salidas del chip (a la derecha). ..... 40

**Figura 3.12.** Módulo XBee ..... 42

**Figura 3.13.** Pantalla principal del programa X-CTU de configuración de las comunicaciones entre módulos XBee ..... 43

**Figura 3.14.** Sistema completo del control remoto de la cortina, mediante el Nunchuck ..... 48

**Figura A.3.1.** Esquema que representa la ubicación de los motores LW 25-B44 de Somfy ..... 56

**Figura A.5.1.** Sistema completo de la aplicación ..... 58

# ÍNDICE DE TABLAS

**Tabla 1.1.** Intervalos de memoria utilizados para cada periférico .....3
**Tabla 1.2.** Relación de los registros de memoria de cada botón .....3
**Tabla 1.3.** Tipo de interruptor de cada botón .....5
**Tabla 1.4.** Descripción de la función de cada elemento del cableado .....10
**Tabla 2.1.** Script simple con GlovePIE.....13
**Tabla 3.1.** Características de las principales tipos de placa Arduino .....28
**Tabla 3.2.** Estructuras, variables y funciones de Arduino. .....33
**Tabla 3.3.** Estructura del programa de adquisición de datos del Wiimote, mediante la plataforma Arduino. .....35
**Tabla 3.4.** Umbral de valores de las diferentes variables de estado del Wiimote.....37
**Tabla 3.5.** Umbral de valores para las diferentes acciones del sistema. .....38
**Tabla 3.6.** Modificaciones del programa original (Tabla 2.6.), con el uso de los umbrales anteriores. .....38
**Tabla 3.7.** Configuración de los módulos XBee emisor, y receptor.....44
**Tabla 3.8.** Programa de control remoto, referente al módulo emisor. .....45
**Tabla 3.9.** Programa de control remoto, referente al módulo receptor.....46
**Tabla A.1.1.** Especificaciones técnicas del acelerómetro ADXL330.....53
**Tabla A.2.1.** Características mecánicas del acelerómetro LIS3L02AL .....54
**Tabla A.2.2.** Características eléctricas del acelerómetro LIS3L02AL .....55
**Tabla A.3.1.** Especificaciones técnicas de los motores Somfy LW 25-B44 .....56
**Tabla A.4.1.** Especificaciones técnicas del driver .....57
**Tabla A.4.2.** Especificaciones técnicas del driver detalladas.....57

# INTRODUCCIÓN Y OBJETIVOS

El mando de la video consola Wii, conocido como Wiimote, ha revolucionado el mundo de los video juegos. Se trata de un objeto de actualidad con un nivel de complejidad técnica considerable, pero con unas características muy especiales y novedosas que convierten su uso en algo intuitivo y llamativo. Teniendo en cuenta el atractivo de este mando entre la mayoría de los jóvenes, se ha considerado que incluirlo como un elemento más en el desarrollo de prototipos puede suponer una motivación extra a las que ya ofrece de por sí el aprendizaje basado en proyectos (PBL, Project Based Learning).

El objetivo de este trabajo es explorar las posibilidades del Wiimote como interfaz de control, tomándolo como base para el aprendizaje de contenidos relacionados con la electrónica a partir de la experimentación. Así, se estudia el diseño del Wiimote detallando todos los elementos que lo componen: sensores y actuadores, electrónica analógica, electrónica digital y los sistemas de comunicaciones por cable e inalámbricos. Además, se realiza un análisis de las capacidades de integración del Wiimote en diseños electrónicos, ya sean de arquitectura software, o bien, de arquitectura hardware. Lo descrito anteriormente proporciona un entorno y una serie de herramientas que permiten resolver de una forma divertida diferentes problemas de ingeniería. Todo ello, constituye un contexto atractivo, que puede ayudar a potenciar el aprendizaje basado en proyectos.

El trabajo está estructurado en 3 capítulos. En el primer capítulo se introducen las características y especificaciones técnicas del Wiimote, detallando cada uno de los componentes que lo conforman. Además, también se realiza una breve descripción del funcionamiento del mando, basado principalmente en un acelerómetro y en la posibilidad de transmitir la información de forma inalámbrica.

En el segundo capítulo, se muestran una serie de aplicaciones, en las que el Wiimote interviene como HID (Dispositivo de Interfaz Humana). Una de las aplicaciones consiste en el control de una estructura de NXT Lego, mediante el Wiimote y el desarrollo de una aplicación en el entorno de programación LabVIEW. Por último, en el tercer capítulo, se da una visión del Wiimote (Nunchuck), como interfaz de control de prototipos hardware. Como ejemplo de aplicación se ha escogido el control del movimiento de una cortina motorizada. Se han diseñado dos prototipos basados en la plataforma Arduino. Uno de ellos acciona la cortina a distancia, de forma inalámbrica. Para ellos, se utilizan dos módulos XBee, basados en la tecnología ZigBee, que permite comunicaciones de hasta 100 metros, con unos requisitos de potencia muy reducidos.

Finalmente se presentan las conclusiones extraídas del desarrollo de este trabajo. Además de los anexos referentes a componentes electrónicos, esquemas de los diseños, se adjunta una recopilación de referencias extraídas de diferentes páginas Web. Aunque actualmente esta información está disponible en la red, se ha preferido incluirla en los anexos a modo de archivo, ante la posibilidad que esas páginas dejen de ser accesibles.

# CAPÍTULO 1. ANÁLISIS DE LAS CARACTERÍSTICAS FÍSICAS Y TÉCNICAS DEL WIIMOTE

## 1.1. Introducción

El mando de la Wii, más conocido como Wiimote (de Control remoto Wii), es el dispositivo de control de una de las consolas más vendidas de todos los tiempos, la consola Nintendo Wii. La características principales son las de interpretar los movimientos reales en el espacio, y la capacidad de apuntar hacia diferentes objetos en la pantalla, todo ello de forma inalámbrica. Esta característica lo diferencia del resto de consolas fabricadas hasta la actualidad, en las cuales el movimiento debía ser transferido mediante un joystick, o bien, mediante botones.

La forma del mando es muy similar a la de cualquier mando de televisión, además de ser muy ergonómico para su sujeción. Las dimensiones son: 14,6 x 3,5 x 3,1 cm, y su peso no supera los 200 gramos. En la superficie superior del mando se encuentran los diferentes botones, un total de 9, con la siguiente simbología: POWER, A, B, -, HOME, +, 1, 2 y la cruz digital (UP, DOWN, LEFT y RIGHT). También se dispone de un altavoz en la parte central, y de 4 leds en la parte inferior que indican diferentes funciones.

Como fuente de energía, el Wiimote utiliza dos pilas AA que proporcionan una autonomía de hasta 60 horas si sólo se utiliza el acelerómetro, y de 30 horas aproximadamente si además también se hace uso del puntero ( IR cámara, situada en la parte frontal del mando). No obstante es difícil de determinar la duración de las pilas o baterías, ya que depende del tipo de uso que se realice.

## 1.2. Características técnicas. Componentes

Entre los dispositivos electrónicos que forman el Wiimote, destacan los siguientes: el chip de 16 KB de memoria EEPROM, el chip bluetooth (modelo: Broadcom BCM2042), el acelerómetro ADLX330, la cámara de infrarrojos (IR camera), el vibrador (rumble), el altavoz, los leds y finalmente los diferentes botones.

### 1.2.1. La memoria

El chip de 16 KB EEPROM dispone de una sección de 6 KB, en la cual el usuario puede leerla y escribirla de forma libre. En esta parte de la memoria es posible registrar diferentes perfiles de usuario. Este rango de datos se encuentra localizado entre los siguientes registros de memoria: 0x0070 al 0x176F. La memoria se encuentra dividida en diferentes partes: memoria accesible para el usuario, registros de control y los registros de entrada.

Los registros de control hacen referencia a los diferentes periféricos incluidos en el mando, tales como el altavoz, los llamados extension controllers, como por ejemplo el Nunchuck, y la cámara de infrarrojos. La dirección de memoria consta de 16 bits. En la Tabla 1.1. se muestran los rangos de registros de memoria que se utilizan para cada periférico:

**Tabla 1.1.** Intervalos de memoria utilizados para cada periférico

| Inicio | Final | Uso |
|---|---|---|
| 0xA20000 | 0xA20009 | Altavoz |
| 0xA40000 | 0xA400FF | Extensión Controller (Nunchuck) |
| 0xB00000 | 0xB00033 | Cámara infrarrojos |

En el desarrollo de este proyecto se realizará un análisis con mayor detenimiento de los registros de memoria referentes al Nunchuck, el cual se utilizará en el desarrollo de una aplicación, concretamente, en el control remoto del cierre y la apertura de una cortina.

En último lugar, aparecen los registros de entrada, controlados de forma directa por el chip Broadcom. Se diferencian dos tipos de entradas, por un lado los registros que hacen referencia a los tres ejes del acelerómetro, y por otro lado, los referentes a los 11 botones que conforman el mando. También se debe considerar como entrada, el puerto de expansión donde se puede obtener una señal externa, como por ejemplo la proveniente del Nunchuck. Los botones envían en el reporte de entrada un 1 cuando está presionado y un 0, cuando no lo está. Por defecto, el estado de los botones son enviados en lo que se denomina “Data Reporting Mode” ( 0x30 ). Normalmente, la configuración preestablecida en el Wiimote, establece que se envíe de forma continuada esta información.

La información referente al estado de los botones queda registrada como bits en 2 bytes. En la Tabla 1.2. se muestra la correspondencia de los registros con cada uno de los botones:

**Tabla 1.2.** Relación de los registros de memoria de cada botón

| Bit | Máscara | Primer Byte | Segundo Byte |
|---|---|---|---|
| 0 | 0x01 | Izquierda (Cruz digital) | 2 |
| 1 | 0x02 | Derecha (Cruz digital) | 1 |
| 2 | 0x04 | Abajo (Cruz digital) | B |
| 3 | 0x08 | Arriba (Cruz digital) | A |
| 4 | 0x10 | Más | Menos |
| 5 | 0x20 | Otros usos | Otros usos |
| 6 | 0x40 | Otros usos | Otros usos |
| 7 | 0x80 | Desconocido | HOME |

### 1.2.2. Bluetooth

El mando de la Wii incorpora un chip Bluetooth de la marca Broadcom, modelo BCM2042. La función del chip es la comunicación con otro dispositivo Bluetooth, en cualquiera de los dos sentidos, es decir, del dispositivo Bluetooth del Wiimote hacia cualquier otro dispositivo, y a la inversa. Es un dispositivo sin cables, que utiliza el estándar bluetooth para establecer una comunicación con algunos dispositivos receptores que utilicen este estándar.

En el caso del Wiimote utiliza el estándar Bluetooth HID, mediante el cual el dispositivo proporciona el servicio de entrada y salida de datos de procedencia humana desde y hacia el anfitrión. Se define anfitrión, como aquel dispositivo que utiliza o solicita los servicios de un dispositivo de interfaz humana (HID).

La utilización de este perfil como definición de dispositivo HID del bus serie universal (USB), incorpora bastantes ventajas, como la de aprovechar los controladores de clase predeterminados para dispositivos USB. Este estándar permite la utilización de los servicios de HID, mediante la capa L2CAP. La capa L2CAP (Logical Link Control and Adaption Protocol, Protocolo de control y adaptación del enlace lógico), se utiliza dentro de la capa de protocolos del estándar Bluetooth, para pasar paquetes con y sin orientación a la conexión a sus capas superiores incluyendo tanto al Host Controller Interface (HCI) como directamente al gestor del enlace.

Además también permite la inicialización y el control de los dispositivos que se describen a sí mismos, y proporciona un enlace con muy bajo retardo y requisitos de potencia mínimos.

### 1.2.3. Cámara de infrarrojos

El Wiimote incluye en su parte superior una cámara monocromo de 128 x 96 píxeles, similar a una webcam, pero sensible a luz infrarroja. El procesador incorporado en la misma es capaz de seguir hasta 4 objetos móviles. El análisis del subpíxel (8x) permite proporcionar una resolución de 1024 x 768, para los puntos seguidos. La barra sensora que se proporciona con la consola Wii contiene dos leds infrarrojos en ambos extremos, los cuales permiten seguir mediante el Wiimote información sobre los puntos. Con el filtro de infrarrojos intacto, las fuentes de 940 nm son detectadas con aproximadamente dos veces la intensidad de fuentes con 850nm, pero no se resuelven, tan bien, en distancias demasiado próximas. Si por casualidad el filtro no estuviese presente, este podría seguir cualquier elemento brillante.

La cámara se puede encontrar en tres estados diferentes: cámara en estado de no recogida de datos, cámara recogiendo datos y con una sensibilidad media, y finalmente, cámara recogiendo datos y con alta sensibilidad. Cuando esta sensibilidad se reduce, también se reduce a su vez, la resolución del subpíxel, acercándose a la verdadera resolución del sensor (128 x 96 píxeles).

### 1.2.4. El vibrador (rumble)

Otro elemento del Wiimote es el vibrador, más conocido como rumble, implementado con un pequeño motor conectado a un peso excéntrico. Al girar el motor, esta excentricidad provoca la vibración. El modelo del motor es el SEM 8728DA, con un voltaje de 3,3 V de continua y una intensidad máxima de 35 mA. El vibrador se activa cuando el bit correspondiente al vibrador se encuentra a 1. En caso contrario, estará desactivado.

### 1.2.5. Los botones

El mando de la Wii incorpora 11 botones distribuidos por toda su superficie (ver Figura 1.1.), además del botón de sincronismo situado en un lateral de las pilas, y del botón B, situado en la parte posterior del mando. Existen diferencias físicas en cuanto al tipo de botones, algunos basan su funcionamiento en un interruptor de tipo membrana (Cruz digital, A, B, 1 y 2), mientras que otros se basan en micro interruptores (POWER, -, HOME, +). En la Tabla 1.3. se realiza esta diferenciación, además de añadir en la posición de la placa electrónica en la que esta localizada (ver Figura 1.1.).

**Tabla 1.3.** Tipo de interruptor de cada botón

| Botón | Tipo de interruptor | Localización en la placa ( Nomenclatura) |
|---|---|---|
| A | Membrana | Superior (SW9) |
| B | Membrana | Posterior (SW8) |
| - | Micro interruptor | Superior (SW10) |
| HOME | Micro interruptor | Superior (SW11) |
| + | Micro interruptor | Superior (SW5) |
| 1 | Membrana | Superior (SW7) |
| 2 | Membrana | Superior (SW6) |
| Up | Membrana | Superior (SW4) |
| Down | Membrana | Superior (SW3) |
| Left | Membrana | Superior (SW1) |
| Right | Membrana | Superior (SW2) |
| Sync | Micro interruptor | Posterior (SW12) |
| Power | Micro interruptor | Superior (SW13) |

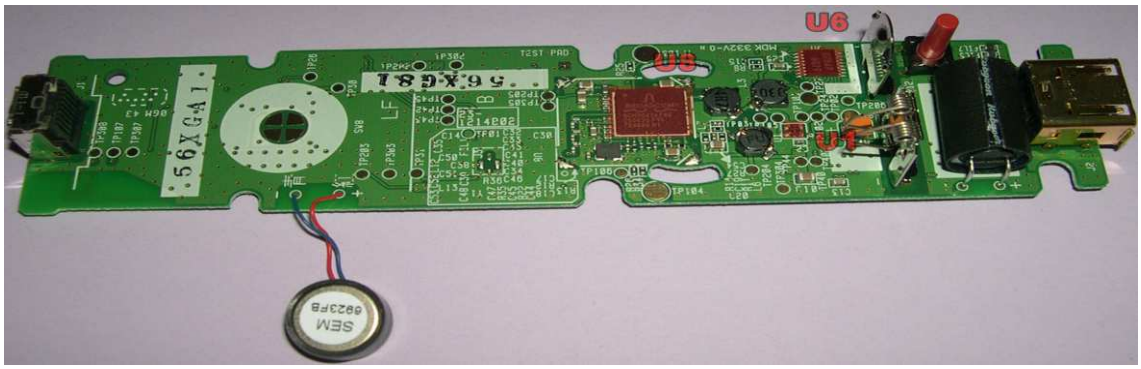



**Figura 1.1.** Esquema del circuito electrónico del Wiimote [5]

## 1.3. Funcionamiento

El Wiimote basa su funcionamiento en la capacidad de detectar la aceleración y el sentido de los movimientos en los tres ejes (x, y, z) mediante el acelerómetro que lleva incorporado. Un acelerómetro es un dispositivo electrónico que permite detectar y medir vibraciones, mediante la medida de la aceleración. Este dispositivo se conoce como un MEMS (Sistema Micro Electro-Mecánico), ya que convierte un movimiento mecánico en un valor electrónico. Es decir, la inclinación en el movimiento del mando en cualquiera de los tres ejes, es convertido en una diferencia de potencial, que una escala caracterizada, permite conocer el tipo de movimiento que se ha realizado.

El chip MEMS utilizado en el desarrollo del Wiimote es el ADXL330, desarrollado por la multinacional Analog Devices. Las dimensiones del dispositivo electrónico son pequeñas ( 4 x 4 x 1,5 mm). El rango de detección del acelerómetro abarca ± 3 g. Además dispone de una sensibilidad típica de 300 mV / g.

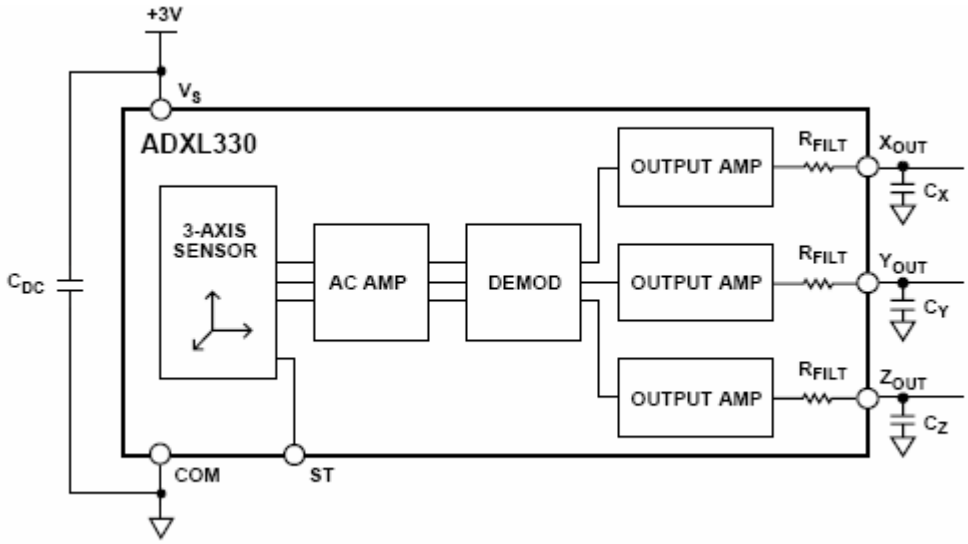



**Figura 1.2.** Esquema del acelerómetro ADXL330

En el esquema de la Figura 1.2. se aprecia el funcionamiento del acelerómetro. La parte del chip que detecta los movimientos es el sensor, formado por una pieza de silicio sujetada en uno de sus extremos (ver Figura 1.3.), y colocado entre el campo eléctrico que generan dos capacitores. Se encarga de la transmisión del movimiento. Cuando sometemos el mando a un desplazamiento, la pieza de silicio se desplaza hacia uno de los dos lados (capacitares), con mayor o menor deformación, provocando un cambio en el campo eléctrico. Esta variación del campo eléctrico provoca un determinado movimiento, y en consecuencia una determinada acción en la pantalla.

En la parte interna se diferencian tres etapas, la salida amplificada, a continuación un demodulador y en la parte final, el amplificador de AC.

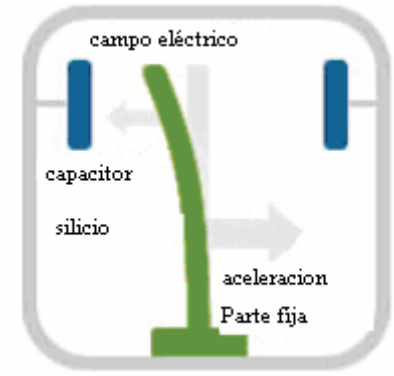



**Figura 1.3.** Esquema de funcionamiento de la pieza de silicio [3]

Toda la información capturada por el acelerómetro queda registrada en un de los registros de la memoria interna que dispone el Wiimote.

Se ha detallado de forma particular como base de funcionamiento el acelerómetro, ya que es lo que lo diferencia de otros mandos de diferentes consolas. El funcionamiento de los botones es bastante sencillo, cuando se presiona cualquiera de ellos, se activa un 1 en el registro correspondiente, si está desactivado estará con el valor 0. De forma posterior, los datos seguirán el mismo camino detallado en el párrafo anterior.

## 1.4. Comunicación

El Wiimote se comunica con algunos dispositivos mediante Bluetooth, ya que dispone de un chip Broadcom BCM2042. Los datos del acelerómetro, de los botones, así como los datos de la cámara de infrarrojos, se envían al chip integrado en la placa del Wiimote. Este chip transmite a su vez vía radio los datos registrados hacia la consola, PC o con algún otro dispositivo compatible, que disponga también de un chip Bluetooth.

Se puede establecer la comunicación en ambos sentidos a la frecuencia de 2,4 Ghz, llegando a alcanzar un ancho de banda total de 2,1 Mbps. Una de las características más importantes que dispone el estándar Bluetooth es su bajo retardo en las comunicaciones entre ambos dispositivos, sin llegar a obtener los valores que se obtendrían si la comunicación se estableciese mediante un cable. No obstante, el valor de este retardo se ha reducido de forma considerable. Por último, otra característica importante en la referencia al estándar, es el bajo consumo que ofrece, aportando una mayor duración a las baterías, en este caso pilas, del Wiimote. Aunque como veremos en capítulos posteriores, el ZigBee, ofrece un menor consumo.

## 1.5. Nunchuck. Expansión del Wiimote

El Nunchuck forma parte del mando de la Wii. Su nombre deriva de la palabra Nunchaku, una antigua arma de artes marciales. Este elemento no puede funcionar de forma independiente al mando de la Wii, debe estar siempre conectado mediante el conector situado en la parte inferior del mando. No obstante, el fabricante Nyko desarrolló un Nunchuck inalámbrico que se comunica con el Wiimote mediante una comunicación por radiofrecuencia a 2,4 Ghz.

### 1.5.1. Características técnicas. Componentes.

El Nunchuck está formado por un menor número de controles que el Wiimote. Básicamente esta formado por dos componentes esenciales: un acelerómetro, para controlar los tres ejes de movimiento, y dos potenciómetros, para controlar el eje x y el eje y del joystick. Además también dispone de dos botones en la parte superior, nombrados C y Z.

El chip MEMS utilizado en el desarrollo del Nunchuck es el LIS3L02AL (ver Figura 1.4.), desarrollado por la multinacional ST. Las dimensiones del dispositivo electrónico son pequeñas, y su peso no supera los 10 miligramos. El rango de detección del acelerómetro cubre $\pm$ 2 g. Además dispone de una sensibilidad típica de 660 mV / g.

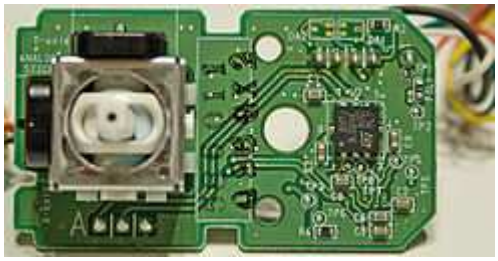


**Figura 1.4.** Vista superior del joystick (a la izquierda) y acelerómetro ST Microelectronics LIS3L02AL (a la derecha) [12]

### 1.5.2. Comunicación con el Wiimote

El bus I2C es un bus de comunicaciones en serie. En nuestro caso, el Nunchuck utiliza este bus, tal y como podemos ver en la Figura 1.5. Es uno de los buses más utilizados en muchos equipos electrónicos, ya sean memorias, procesadores de señal, codificadores de vídeo y muchos otros. El bus I2C precisa únicamente de dos líneas, por un lado la línea de datos (SDA), y por otro, la línea de reloj (SCL), siendo ambas bidireccionales. También es necesaria una tercera línea, la correspondiente a la masa.

Cada dispositivo que se conecta al bus se le reconoce por una única dirección, y puede trabajar tanto de emisor o receptor, dependiendo de la función de cada dispositivo. Un dispositivo master, es aquel que inicializa la transferencia de datos en el bus y genera la señal de reloj para permitir esa transferencia. Cabe destacar, que el bus incluye un detector de colisiones, que posibilita que existan varios master conectados. Generalmente, se suele considerar un master a un microcontrolador.

La transmisión bidireccional serie se realiza con transmisiones de palabras de 8 bits a diferentes velocidades. Hay cuatro posibles alternativas de velocidad, comprendidas entre los 100 Kbps y los 3,4 Mbps. A continuación se detalla las 4 escalas de velocidad permitidas en el bus I2C:

- **Modo Estándar**: 100 Kbps ( velocidad más extendida).
- **Modo Rápido**: 400 Kbps.
- **Modo Rápido Plus**: 1 Mbps.
- **Modo Alta Velocidad:** Superior a 3,4 Mbps.

Existe una única limitación en el número máximo de dispositivos a conectar, determinada por la capacidad del conjunto de los dispositivos conectados, que no debe superar los 400 pF.

Cuando se realizan las comunicaciones entre el master y los esclavos, existen dos condiciones, una de llamada START y otra STOP. El master debe informar al resto de dispositivos cuando se cumpla alguna de estas dos situaciones, es decir, cuando comience la transmisión de datos, y cuando esta finalice. En cada transmisión de datos, el receptor devuelve un ACK (acknowledge), es decir un reconocimiento, conforme se han recibido los datos correctamente.

### 1.5.3. Cableado y conexionado

El Nunchuck se conecta al Wiimote mediante un cable de 90 cm de longitud. Este cable esta formado por cuatro hilos, diferenciados por su color. En la Tabla 1.4. se detalla el uso de cada uno de los hilos dependiendo de su color:

**Tabla 1.4.** Descripción de la función de cada elemento del cableado

| Color del hilo | Función |
|---|---|
| Rojo | Alimentación de 3,3 V (No obstante también es posible la alimentación a 5 V) |
| Blanco | Tierra (Ground) |
| Verde | Flujo de datos |
| Amarillo | Clock (sincronización temporal) |

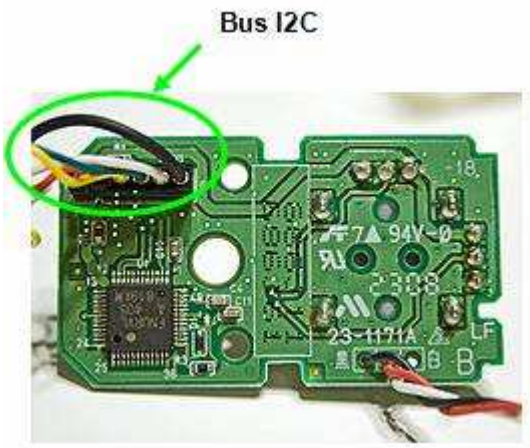



**Figura 1.5.** Detalle del bus I2C del Nunchuck [12]

# CAPÍTULO 2. WIIMOTE COMO DISPOSITIVO DE INTERFAZ HUMANA (HID) PARA PC

## 2.1. Introducción

Desde la aparición en el mercado de la consola Wii, y en especial de uno de sus componentes, el Wiimote, se han realizado diferentes aplicaciones prácticas con el mando. Muchas personas con conocimientos de diferentes lenguajes de programación y con la ayuda de la librería del Wiimote, creada por Brian Peek [21], han realizado algunas aplicaciones de gran utilidad en la vida cotidiana.

Un ejemplo, es Johnny Chung Lee, estudiante de la universidad de Carnegie Mellon, quien ha realizado diferentes aplicaciones en las que utiliza el Wiimote como interfaz de control. Dos de estas aplicaciones creadas por Johnny Chung Lee (ver Anexo 6) han sido la sustitución del ratón del PC por el Wiimote (aplicación llamada WiinRemote), y la pantalla interactiva (aplicación llamada Wiimote Whiteboard). Además de estas aplicaciones, Johnny ha realizado muchas otras, como el seguimiento de los dedos con el Wiimote, o el seguimiento del movimiento de la cabeza de una persona y localizarla en el área de una pantalla. En ambas aplicaciones, interviene la cámara de infrarrojos y el acelerómetro.

En todos estas aplicaciones se utiliza el lenguaje de programación C#, orientado a objetos. Con esta herramienta existe la ventaja de disponer la librería llamada WiimoteLib.dll, que permite la comunicación con el Wiimote, adquiriendo de forma detallada el estado del mismo en cada momento.

No obstante, existen otros entornos de programación, que permiten la interacción entre el mando y otro dispositivo Bluetooth. Este es el caso del entorno de programación LabVIEW, que en este tema veremos que, mediante la programación de bloques, se consigue la comunicación entre ambos dispositivos, Wiimote y la estructura NXT Lego.

## 2.2. Wiimote como mouse del PC

Una posible alternativa al ratón del PC, podría ser el Wiimote. No obstante, cabe destacar que los movimientos del Wiimote se realizan de forma rápida y es difícil seleccionar aquello que se desea.

Es posible realizar esta acción mediante la aplicación WiinRemote que podemos descargarla del siguiente enlace: http://onakasuita.org/wii/index-e.html [10], de forma gratuita. Para conseguir que el Wiimote funcione como un ratón normal, seguimos los siguientes pasos:

*Proceso de comunicación entre el Wiimote y el PC*

1. Pulsamos los botones 1 y 2 del Wiimote de forma simultánea, y no los soltamos hasta el final del proceso.
2. Abrimos el controlador Bluetooth de nuestro dispositivo y añadimos el Wiimote. El nombre con el cual es detectado es Nintendo RVL-CNT-01.
3. No se dispone de ninguna clave de paso.
4. Finalmente, el Wiimote es reconocido como un dispositivo de interfaz humana (HID).

A continuación ejecutamos directamente la aplicación WiinRemote.exe contenida en el archivo descargado. Al ejecutarlo aparece la pantalla de la Figura 2.1. Antes de todo, debemos calibrar el dispositivo, pulsando en el botón "Calibrate". Después de la calibración, ya es posible mover el Wiimote, y comprobamos como el cursor del ordenador se mueve conforme al movimiento ejercido sobre el mando. Desde la aplicación es posible variar sensiblemente la velocidad del cursor, así como comprobar cual es el botón pulsado o el valor correspondiente a cada eje, a través del estado del mismo (ON / OFF).

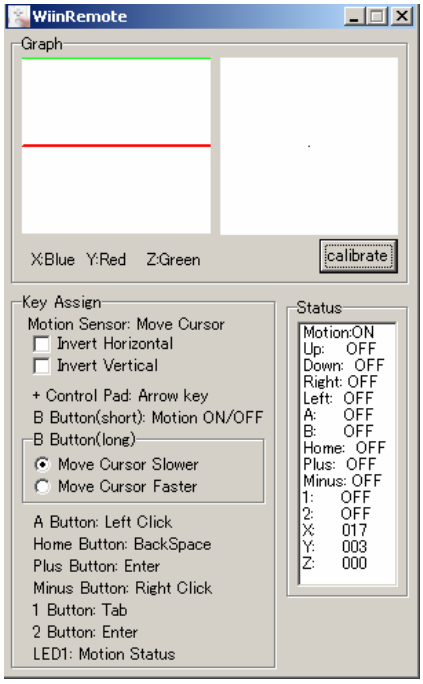



**Figura 2.1.** Pantalla de control del programa WiinRemote

## 2.3. Aplicación GlovePIE

La aplicación GlovePIE recibe el nombre de Glove Programmable Input Emulator). La principal característica de esta aplicación es que sirve para emular todo tipo de sistemas, ya sea un joystick, un teclado o cualquier otro periférico. Cuando se ejecuta el programa, aparece una pantalla en blanco

donde podemos escribir lo que se denomina script. La aplicación GlovePIE soporta multitud de hardware, entre los cuales se encuentra el Wiimote.

Con GlovePIE se pueden realizar scripts en las que intervengan las variables del Wiimote, y las teclas del teclado, relacionando la acción de ambas. Con estas relaciones, se consigue que la función de la tecla correspondiente del teclado sea igual que el botón o acelerómetro del Wiimote, o a la inversa.

Esta función es muy utilizada para los juegos, ya que existen multitud de ellos que tienen predeterminadas las teclas para unas acciones determinadas. Con ello se pueden establecer las relaciones que se consideren oportunas entre el teclado y los botones del Wiimote. En el siguiente ejemplo (ver Tabla 2.1.) podemos ver un script muy sencillo en el que se establecen relaciones simples entre ambos elementos:

**Tabla 2.1.** Script simple con GlovePIE [25] (ver Anexo 7)

```
// Controles de dirección

Left = Wiimote.Up

Right = Wiimote.Down


//Controles de aceleración


Up = Wiimote.One

Down = Wiimote.Two
```

Tal y como se puede apreciar en este script (Tabla 2.1.) se relaciona las 4 flechas direccionales del teclado (), con 4 botones disponibles en el Wiimote (UP, DOWN, 1 y 2).

Por ejemplo, cuando presionamos la tecla 1 del Wiimote, mediante el script se interpreta que la tecla presionada del teclado es la flecha hacia arriba, que en las opciones de este juego en particular, tiene la acción de acelerar. En cambio, si apretamos el botón 2, se realiza la acción inversa, es decir el vehículo reduce su velocidad hasta frenarse del todo.

Para conocer la nomenclatura correspondiente a los diferentes botones y acelerómetros del Wiimote podemos acceder al siguiente enlace: http://www.elotrolado.net/wiki/Glovepie [25] (ver Anexo 7, donde existe una tabla en al que se detalla esta información).

## 2.4. Pizarra o pantalla interactiva

La pantalla interactiva es una aplicación que nos permite disponer de un display interactivo multipunto. El material necesario para realizar la siguiente aplicación es el siguiente:

- El Wiimote.
- Un lápiz con red infrarrojo ( se explica su fabricación a lo largo de este apartado).
- Un ordenador con dispositivo Bluetooth.
- Opcionalmente un proyector. No es del todo necesario, ya que con el monitor del ordenador también es posible realizarlo.
- Aplicación Wiimote WhiteBoard, que es posible descargarla del siguiente enlace: http://johnnylee.net/projects/wii/ [23].

El funcionamiento del siguiente sistema consiste en situar el Wiimote de forma fija apuntando hacia la pantalla que se desea manipular (ver esquema Figura 2.2.), teniendo en cuenta que el ángulo de apertura del mando es de 45º. A continuación fabricamos un lápiz con un led de infrarrojos. Una vez disponemos del material situado, emparejamos el PC con el Wiimote mediante el controlador Bluetooth del sistema operativo. Por otro lado, ejecutamos la aplicación Wiimote Whiteboard, y nos aparecerá el cuadro de la Figura 2.4. Posteriormente calibramos el dispositivo, para ello pulsamos en el botón “Calibrate Location”, el cual nos abre una pantalla en blanco en toda la extensión de la pantalla, mostrando un círculo con una cruz en una de las esquinas de la misma. Para proceder con la calibración, acercamos el lápiz infrarrojos hasta ese punto hasta que desaparezca, continuando con los restantes tres puntos de calibración. Finalizado este proceso, podemos manipular la imagen mediante el uso del lápiz de infrarrojos.

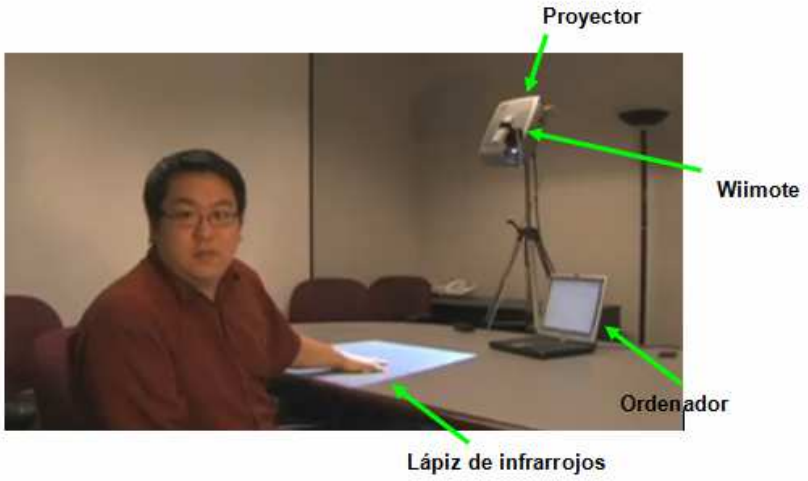



**Figura 2.2.** Fotografía del montaje de la pizarra interactiva [23]

La fabricación del lápiz de infrarrojos es sencilla. En nuestro caso lo realizamos nosotros mismos, una vez disponemos del material necesario. El material necesario para su fabricación es el siguiente: un soporte (funda de un bolígrafo), un led infrarrojos, una resistencia para limitar la intensidad de paso, un pulsador y dos pilas botón. En la Figura 2.3., se detalla como debe quedar el montaje final. Para comprobar si el lápiz de infrarrojos funciona correctamente, se debe recordar que los infrarrojos no es una luz visible, y por lo tanto si se aprieta el pulsador no veremos ningún tipo de luz. Podemos comprobar si funciona correctamente mediante el uso de una cámara.

No obstante nos encontramos con un serio problema, debido a la directividad del led de infrarrojos, según el apuntamiento o inclinación realizada, la luz infrarroja no era detectada por la cámara del Wiimote. Determinamos que la mejor opción era buscar un led de infrarrojos que fuese menos directivo, para que de esta forma no debamos buscar la inclinación o movimiento para el cual la cámara detecte la luz.

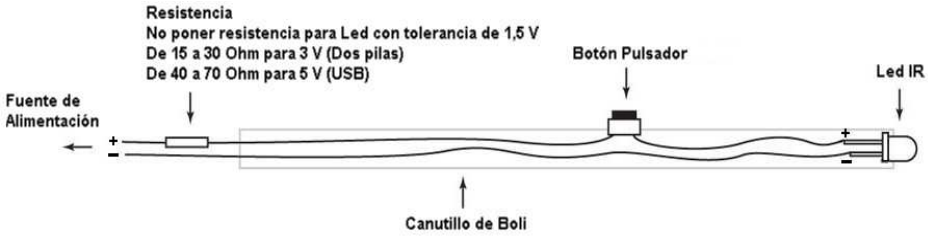



**Figura 2.3.** Esquema del montaje de un lápiz de infrarrojos [23]

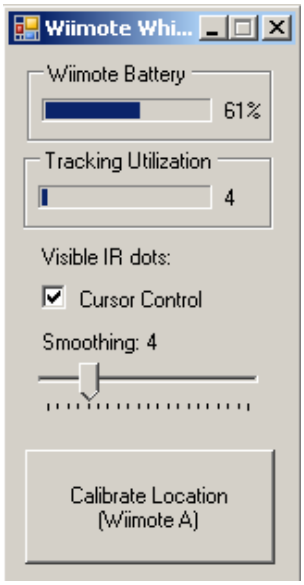



**Figura 2.4.** Pantalla principal del programa Wiimote WhiteBoard

## 2.5. Control remoto de un NXT Lego

El objetivo de la siguiente aplicación es controlar de forma remota un NXT Lego mediante el uso del mando de la Wii. Para realizar esta aplicación se realiza un estudio de las posibilidades de interfaz disponibles para el desarrollo de la aplicación. Dada la imposibilidad de conectar de forma directa el Wiimote al NXT Lego, se decide que se debe implementar un programa, con ayuda del PC, que realice la función de interconexión entre ambos dispositivos. Para ello existen multitud de entornos de programación para realizar esta función, tales como Microsoft Visual Studio, en los diferentes lenguajes de programación disponible, así como la aplicación LabVIEW.

Después de realizar una búsqueda determinando las ventajas e inconvenientes del uso de cada una de las aplicaciones enfocando a la aplicación a realizar, se determina que la más apropiada es la aplicación LabVIEW. Se decide utilizar está aplicación por los siguientes motivos:

- Programación mediante bloques y de forma muy visual, que facilita enormemente el desarrollo de la aplicación. Además es un entorno muy utilizado en la docencia.

- Nacional Instruments, el fabricante de LabVIEW ofrece una Librería para NXT Lego, con una serie de bloques determinados, que permiten ejercer diferentes acciones sobre cada uno de los componentes conectados al Brick del NXT Lego.

- En la comunidad de Nacional Instruments existe también una Librería para LabVIEW, que permite adquirir los datos provenientes del Wiimote mediante el estándar Bluetooth.

Al disponer de todas las utilidades especificadas anteriormente, nos ofrece la posibilidad, que mediante la unión de todas ellas, se pueda generar la aplicación que se desea. La aplicación que deseamos desarrollar adquiere los datos del Wiimote, diferenciándolos en diferentes variables, y a continuación, según el valor obtenido en cada una de estas variables, ejerce una acción sobre alguno de los elementos que conforman la estructura del NXT Lego.

### *2.5.1. Entorno de programación LabVIEW*

LabVIEW es una herramienta de programación gráfica, que permite el desarrollo, control y diseño de aplicaciones mediante el uso de bloques que tienen unas funciones determinadas. La primera versión del programa fue puesta a la venta en el año 1986, desarrollada por la empresa Nacional Instruments. En un principio, únicamente se desarrollo para plataformas MAC, pero en versiones posteriores se implemento para su uso en otras plataformas, como Windows, UNIX y Linux.

Desde la primera versión de programa, hasta la última versión del mismo se han realizado multitud de mejoras, adaptándolo especialmente a las nuevas tecnologías y estándares surgidos desde entonces. La primera versión fue llamada LabVIEW 1.0, mientras que la última versión que ha aparecido, es la LabVIEW 8.6, que se puede ejecutar perfectamente en el sistema operativo Windows Vista.

La principal ventaja de LabVIEW es su gran capacidad para integrarse con multitud de dispositivos hardware, independientemente del fabricante del mismo. Además, otra característica muy importante, es que gracias a que es un software basado en herramientas gráficas, hace que su utilización se realice de una forma más versátil, y que su aprendizaje sea de forma gradual, adquiriendo nuevos conocimientos del programa, para poder realizar aplicaciones muy sencillas.

Otras funciones que permite la aplicación son las detalladas a continuación:

- Adquisición de datos y análisis matemático
- Automatización industrial y programación de PAC's (Controlador de Automatización Programable).
- Diseño de controladores
- Diseño embebido de micros y chips
- Control y supervisión de procesos
- Visión artificial y control de movimiento
- Robótica
- Domótica

### *2.5.2. Librerías NXT Toolkit*

La librería NXT Toolkit ha sido desarrollada por Nacional Instruments para la plataforma de programación LabVIEW. Es posible la descarga de la librería de forma gratuita a través del siguiente enlace: http://zone.ni.com/devzone/cda/tut/p/id/4435 [18].

Se debe descargar el archivo del enlace anteriormente especificado e instalarlo en el PC que dispone de la aplicación LabVIEW. Cuando se haya finalizado la instalación, ejecutamos la aplicación LabVIEW, y seleccionamos View > Functions Palette, aparecerá el recuadro de la Figura 2.5. En él, se muestran todas las librerías instaladas en la aplicación. La librería NXT Toolkit se localiza dentro del conjunto Addons, con dos componentes (ver Figura 2.6.): NXT Toolkit (1) y la NXT Direct Commands (2).

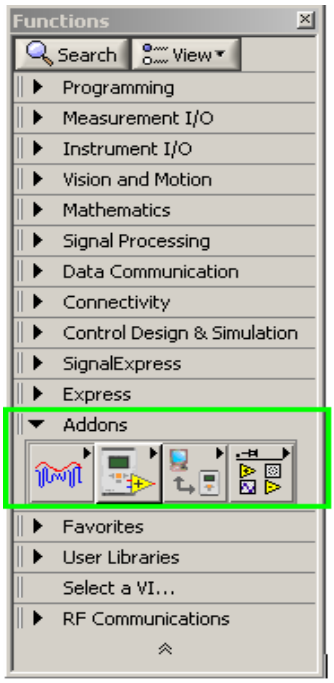



**Figura 2.5.** Detalle de las diferentes librerías instaladas en LabVIEW, en particular las Addons

Hay dos formas posibles de ejecución de las aplicaciones desarrolladas. Una de ellas consiste en descargar la aplicación diseñada en el NXT Lego, para posteriormente ejecutarlo, desde el mismo NXT Lego. En cambio, la otra forma, consiste en el envío de comandos directos ejecutados directamente desde LabVIEW.

El número de bloques contenido en la librería NXT Toolkit es mayor que el de NXT Direct Commands. En el caso de la aplicación que se realiza en este proyecto se hace uso de los comandos de la NXT Direct Commands, ya que se envían los comandos de forma directa desde LabVIEW.

En esta librería encontramos diferentes elementos clasificados en: conexiones, entradas, salidas, sonido, archivos de entrada y salida, mensajes Bluetooth, ejecución del programa y utilidades. Como ejemplo, podemos detallar uno de ellos, clasificado dentro de los elementos de salida, que tiene como función accionar un motor (Motor Unlimited ( )) conectado a uno de los tres puertos disponibles (Port A, Port B o Port C), y en el sentido que se marque. El funcionamiento de cada uno de los bloques y las entradas y salidas del mismo, se pueden conocer mediante la ayuda disponible en la aplicación LabVIEW. En ella, seleccionando el bloque, se muestra con todo detalle información del mismo, con el nombre de entradas y salidas, y sus posibles conexiones.

Figura 2.6. Contenido de bloques de la Librería NXT Toolkit: NXT Toolkit (1) y NXT Direct Commands (2). Bloques de salida dentro de la librería Direct Commands (3).

### 2.5.3. Librería WiimoteLib

Como se ha comentado en el apartado de objetivos, para el desarrollo de la aplicación, se precisa conocer con exactitud el estado de cada uno de los botones y del acelerómetro que conforman el Wiimote. La adquisición de datos se puede visualizar mediante el uso de las siguientes aplicaciones generadas con LabVIEW que se pueden descargar del siguiente enlace: http://decibel.ni.com/content/docs/DOC-1353 [20].

En el enlace mencionado anteriormente se puede descargar el archivo WiimoteManaged(wml152).zip , el cual contiene diferentes archivos VI's, y se corresponde con la versión más estable de las aplicaciones. El requisito para poder trabajar con estas aplicaciones es disponer de la versión 8.6 de LabVIEW, ya que con versiones anteriores el comportamiento y las funcionalidades descritas no son las adecuadas.

De todos los archivos contenidos en la carpeta, seleccionamos el llamado Simpled Polled Adquisition.vi. Al abrir esta aplicación se observa un diagrama de bloques que tiene como objetivo adquirir los datos provenientes del Wiimote y representarlos en el Panel Frontal, en un bloque que se asemeja en su forma y representación al Wiimote.
El diagrama de bloques se divide en 5 pequeñas fases fácilmente diferenciadas que se detallan a continuación:

Abrir Librería WiimoteLib.

Junto a la carpeta, se adjunta un archivo .dll llamado WiimoteLib. En concreto, se corresponde a la versión 1.5.2 de esta librería. Esta librería creada por Brian's Peek ofrece la posibilidad de interactuar con el Wiimote, a través de diferentes lenguajes de programación.

(La librería se puede descargar desde el siguiente enlace: http://www.codeplex.com/WiimoteLib [21], versión 1.5.2)

- *Establecer conexión con el Wiimote.*

  En esta fase, LabVIEW establece una conexión con el Wiimote a través de Bluetooth. A continuación, una vez establecida la conexión, se pueden adquirir los datos que se deseen.

- *Mostrar el último estado del Wiimote.*

  Según el botón o el movimiento que se este ejerciendo sobre el mando, el programa identificará el valor al que corresponda y lo mostrará en el siguiente Panel ( ver Figura 2.7.). En la parte superior de la Figura 2.7., se representan los valores correspondientes a los 3 ejes del acelerómetro, y en la inferior la referente a los 11 botones.

**Figura 2.7.** Esquema del estado del Wiimote en LabVIEW

- *Cerrar conexión con el Wiimote.*

Para finalizar, cuando no se desee recoger ningún dato más, la conexión se cierra, y en consecuencia se desconecta la sesión entre ambos dispositivos.

- *Cerrar Librería WiimoteLib.*

Como ya no se precisa de la interacción con el Wiimote, la librería no es necesaria, y por lo tanto se debe cerrar.

### *2.5.4. Detalle de la Aplicación*

En este apartado, se realiza de forma detallada una explicación de cada una de las partes que dispone la aplicación para el control de la estructura de NXT Lego, mediante el Wiimote.

La aplicación que engloba todo el sistema se llama "Aplicación_interactiva_Wiimote_NXT.vi". Esta aplicación se puede dividir en 5 grandes etapas, destacando dos de ellas: la de adquisición de los datos provenientes del Wiimote, y la de acción, que ejecuta una serie de acciones en el NXT, según el valor que se recibe.

Con anterioridad a estas dos fases, debemos establecer la conexión entre el NXT y el PC, mediante el estándar Bluetooth. A la izquierda de la Figura 2.8. , se observa como establecemos esta conexión. En primer lugar, el primer bloque busca el NXT etiquetado como NXTMQ dentro de todos los dispositivos que se encuentren en el área de alcance del Bluetooth. A continuación, cuando ya se ha encontrado el NXT correspondiente, se crea un NXT Object, que será el objeto utilizado en el resto de la aplicación. Finalmente, a la derecha de la Figura 2.8., se encuentra el bloque de desconexión del objeto, que cierra la conexión entre ambos dispositivos. Entre estos dos diagramas de bloques, se adhiere el programa correspondiente.

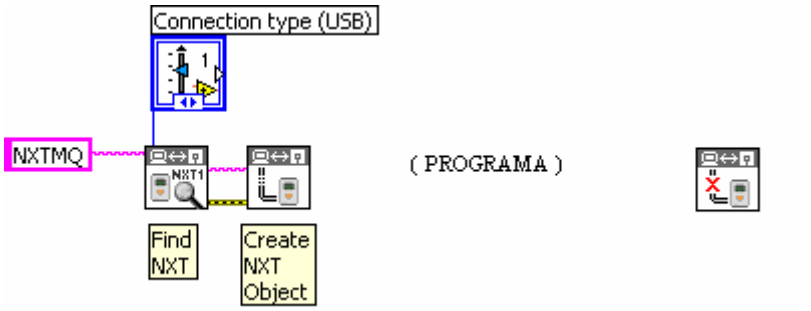



**Figura 2.8.** Diagrama de bloques del establecimiento y desconexión de la comunicación con el NXT.

La primera etapa es la que se denomina adquisición de datos. En esta etapa se adquiere el estado en el que se encuentran cada una de las variables, tanto de los 11 botones, como de los tres ejes correspondientes al acelerómetro. Esta etapa la resolvemos con una de las aplicaciones extraídas del enlace: http://decibel.ni.com/content/docs/DOC-1353 [20].

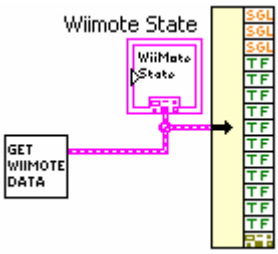



**Figura 2.9.** Array de elementos que contiene el estado del acelerómetro y los botones

El primer bloque de la Figura 2.9. obtiene los datos, el segundo, muestra esos datos de forma representativa, y finalmente el último bloque, llamado "unbundle" en LabVIEW, separa cada cluster de llegada, clasificándolos en diferentes elementos correspondientes a las diferentes variables ( 3 ejes de acelerómetro y 11 botones). Los tres primeros elementos (SGL) corresponden a los valores de cada uno de los ejes del acelerómetro ( X, Y , Z), mientras que los 11 restantes (TF) corresponden a los botones (UP, DOWN, LEFT, RIGHT, A, B, -, HOME, +, 1 y 2). El último elemento pertenece al error out, en el cual se muestra un identificador de un error, en el caso que exista.

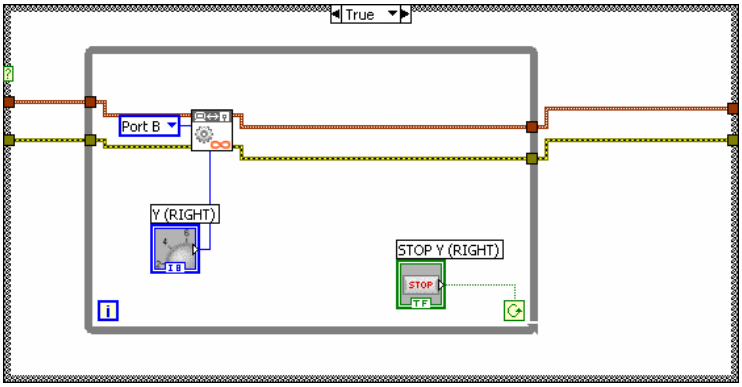



**Figura 2.10.** Diagrama de bloques a realizar cuando la condición de entrada proveniente del array es True

En la segunda etapa, denominada etapa de acción, se ejecuta una acción determinada según el valor que se obtenga de cada elemento. Como ejemplo, en la Figura 2.10. , se corresponde a la acción a realizar cuando se presiona el botón UP de la cruz digital. En ese caso, el valor que llega al unbundle, y concretamente al elemento UP, es de True. En consecuencia, en la secuencia de case, se ejecuta la acción correspondiente a una entrada de True.

En el caso que la entrada sea de TRUE se acciona el motor conectado al Puerto B en el sentido normal a una velocidad determinada por el indicador. La velocidad puede recibir valores comprendidos entre 0 y 100. Esta acción se realiza mientras el botón se encuentra presionado, ya que el sistema que acciona el motor se encuentra dentro de una estructura llamada While Loop, con la condición de salida “Continue if True”, es decir, continuar si a la entrada se recibe True. En el caso que el botón no esté presionado, a la entrada del Case se recibirá un False (F), que accionará aquel sistema que se haya diseñado dentro de esa estructura.

Si deseamos cambiar el sentido de movimiento del motor, debemos seleccionar la entrada de Direction del Motor Unlimited ( ), seleccionar Botón derecho > Bolean Pallete > .

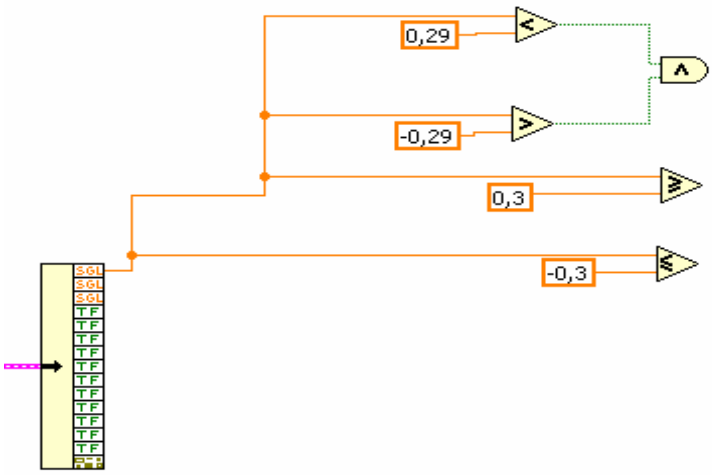



**Figura 2.11.** Esquema del umbral de sensibilidad de cada eje del acelerómetro

El movimiento del mando es muy difícil que lo efectuemos en un único eje. Este hecho nos ocasiona algunos problemas, ya que puede suceder que deseemos trasladar un movimiento únicamente a uno de los puertos y no a más de uno. Para solventar este problema, o más bien, esta imprecisión, hemos decidido habilitar unos intervalos de tolerancia en cada uno de los ejes, para de esta forma evitar en mayor medida accionar aquello que no deseamos.

La solución que planteamos en esta aplicación tiene una consecuencia, el hecho que no se accione el movimiento hasta que no se alcance un valor un poco elevado ( 0,3 o -0.3). No obstante, en general, es más aceptable que se deba realizar un movimiento más brusco para accionar lo que deseamos, que en cambio, se accione de forma muy sensible algún elemento que no sea deseable.

El intervalo implementado lo podemos visualizar en la Figura 2.11. , correspondientes a los valores obtenidos en el eje x del acelerómetro. En la parte superior, observamos que, cuando el valor del elemento x sea mayor que

-0.29 y (A) menor que 0.29, se acciona la parada del motor conectado a un puerto determinado.

En la parte inferior, observamos dos intervalos diferentes, cuando el valor del elemento x es superior a 0.3 se activa el motor correspondiente en el sentido normal. En cambio, cuando el elemento x es inferior a -0.3, se activa el mismo motor conectado a ese puerto, con la diferencia, que el sentido se invierte.

La aplicación la hemos implementado para los siguientes elementos: UP, DOWN, RIGHT, LEFT, A, y el acelerómetro. En la Figura 2.12., que se corresponde con la estructura de Lego utilizada se muestran los posibles movimientos junto al botón o movimiento de ejes del acelerómetro que lo activan.

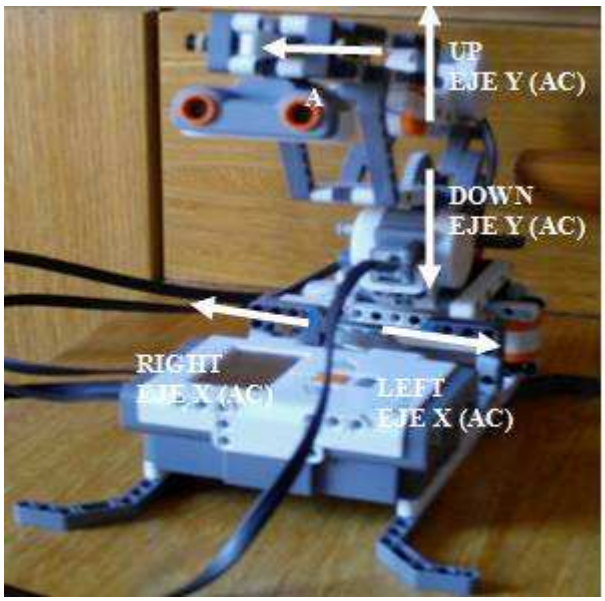



**Figura 2.12.** Fotografía de la estructura del NXT Lego, con sus posibles movimientos.

### *2.5.5. Descripción del proceso*

A continuación, se procede a realizar una descripción detallada de todo el proceso de desarrollo de la aplicación de control de la estructura NXT de Lego, mediante el Wiimote.

El material, los dispositivos y herramientas utilizados en la implementación de la aplicación son los siguientes:

- Un ordenador con sistema operativo Windows Vista.
- Dispositivo Bluetooth incorporado en el PC.

- Programa LabVIEW versión 8.6.
- Librería NXT Toolkit 2.0.
- Librería WiimoteLib 1.5.2.
- Aplicación Simple Polled Adquisition contenida en el archivo *WiimoteManaged(wml152).zip*
- Estructura NXT de Lego.
- Batería o en su defecto 6 pilas para la alimentación del Brick del NXT de Lego.
- Y finalmente la aplicación, llamada Aplicación_interactiva_Wiimote_NXT.vi.

En primer lugar, establecemos la conexión entre el PC y los dos dispositivos Bluetooth (Wiimote y NXT Lego). Para ambos casos se utiliza el controlador disponible en el sistema operativo. A continuación se detalla de forma breve como realizar las conexiones de ambos dispositivos:

➢ <u>Conexión Wiimote</u>

1. Abrimos el controlador Bluetooth de Windows Vista, y seleccionamos la opción para que pueda detectar dispositivos Bluetooth.
2. Presionamos los botones 1 y 2 del Wiimote hasta el final del proceso.
3. Agregamos dispositivo llamado Nintendo RVL-CNT-01.
4. No usar ninguna clave de paso.
5. Finalmente, nos aparece que el dispositivo ha sido instalado como HID (Dispositivo de Interfaz Humana).

➢ <u>Conexión NXT Lego</u>

1. Abrimos el controlador Bluetooth de Windows Vista, y seleccionar la opción para que pueda detectar dispositivos Bluetooth.
2. Comprobamos en el Brick, Bluetooth > Visibility > ON.
3. Agregamos dispositivo llamado NXTMQ
4. Usamos la clave de paso que está en la documentación: 1234
5. Cuando aparece en la pantalla del Brick Passkey, se introduce 1234 y presionamos el botón central (naranja).
6. Finalmente, el dispositivo será reconocido y se le asignará un puerto COM de entrada y otro de salida, normalmente correlativos.

En segundo lugar, procedemos a abrir el archivo Aplicación_interactiva_Wiimote_NXT.vi donde se encuentra el sistema completo. Existen dos formas de ejecutar la aplicación, como Highlight Execution ( ) y como Do not Hightligth Execution ( ). La diferencia entre ambas reside en que con la ejecución Highlight, debido a que se ejecuta de

forma más lenta, permite ver mejor los errores que se producen y donde se producen concretamente. En nuestro caso, ejecutamos el programa en modo normal.

Para ejecutar el programa, tanto si se encuentra en el diagrama de bloques o en el panel frontal, presionamos la flecha blanca marcada hacia la derecha ( ).

Desde el Panel Frontal (ver Figura 2.13.) que hemos creado, podemos controlar diferentes variables, como la velocidad de los motores, su parada, o la parada de todo el sistema. Además también podemos visualizar el estado del Wiimote. Es muy importante no realizar la parada del sistema completo con el botón , ya que no es recomendable, porque las desconexiones de ambos dispositivos Bluetooth (Wiimote y NXT Lego), no se realizan correctamente, y estamos obligados a volver a iniciar los pasos de conexión del dispositivo con el controlador de Windows. En consecuencia la parada del sistema debemos realizarla con el botón STOP más grande situado en la parte superior del Panel Frontal. En la Figura 2.13. hemos intentado detallar todos los elementos que la componen con la adición de varios comentarios, que nos ayudan a entender mejor los diferentes elementos que hemos insertado.

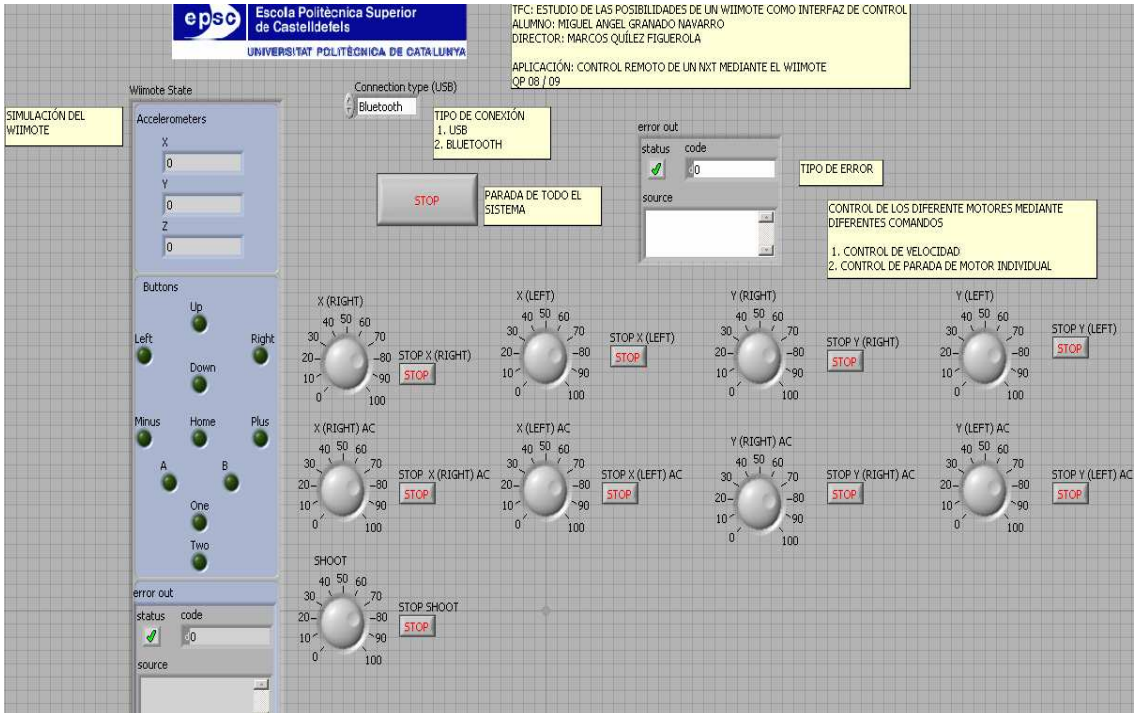



**Figura 2.13.** Panel frontal de la Aplicación en LabVIEW

Se puede ver el esquema completo de la aplicación Aplicación_interactiva_Wiimote_NXT.vi (ver Anexo 5 , Figura A.5.1.).

# CAPÍTULO 3. WIIMOTE COMO INTERFAZ DE CONTROL DE PROTOTIPOS HARDWARE

## 3.1. Aplicaciones desarrolladas con el Nunchuck

En el siguiente apartado se detallan una serie de aplicaciones desarrolladas con diferentes entornos de programación, en las cuales interviene el Nunchuck como interfaz de control de prototipos hardware. La primera se desarrolla conectando directamente el Nunchuck contra la placa Arduino que ejerce las acciones, mientras que en la segunda se establece una comunicación inalámbrica entre dos placas Arduino, que incluyen un módulo XBee cada una.

### 3.1.1. Control de una cortina motorizada

El objetivo de la presente aplicación es controlar una cortina motorizada mediante una placa Arduino Duemilanove y el dispositivo Nunchuck conectado a la entrada. Previamente, mediante el entorno de programación Arduino, basado en lenguaje C, con funciones relativamente básicas, se debe programar el chip ATmega de la placa con aquellas instrucciones que se deseen.

#### *3.1.1.1. Entorno de programación Arduino*

El entorno de programación Arduino [6] es una herramienta muy popular, y representa una solución a muchos de los problemas de aprendizaje e interacción del alumno con la tecnología. Está basada en un hardware abierto, formado por una placa reducida, que contiene una serie de entradas y salidas (ver Figura 3.1.), ya sean analógicas o digitales, además de salidas PWM (Modulación por ancho de pulsos), muy utilizadas en el control de la velocidad de motores. Su entorno de programación está implementado mediante el lenguaje Processing/Wiring. Este lenguaje de programación recoge muchas funciones contenidas en el lenguaje C, además de algunas otras que forman parte de librerías propias de Arduino.

Al ser una plataforma en formato abierto, no ha de adquirirse ningún tipo de licencia para el desarrollo de cualquier aplicación. Se pueden realizar cualquier tipo de proyecto en el que intervengan diferentes entradas, y mediante la programación en Arduino, establecer unas salidas concretas.

El elemento fundamental que constituye la placa es el chip ATmega, ya sea el ATmega8, ATmega168 o el ATmega328. La diferencia entre esta serie de chips reside en la capacidad de la memoria interna (8K , 16K y 32K respectivamente), que ha ido aumentando progresivamente. Las principales características del chip son su sencillez y bajo coste.

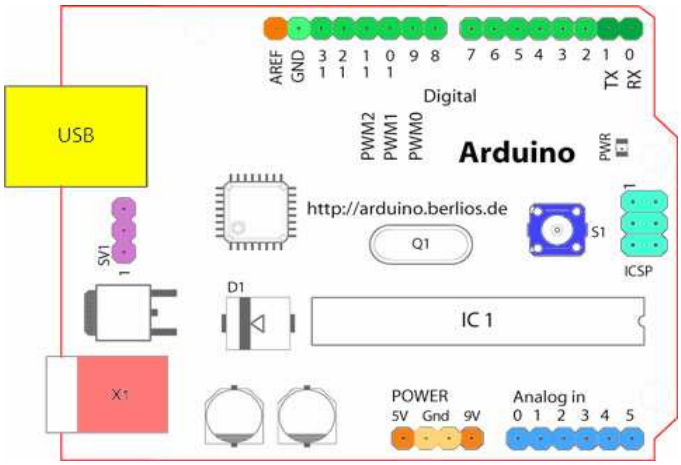
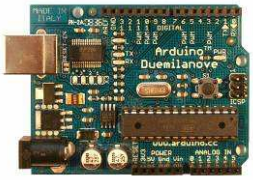
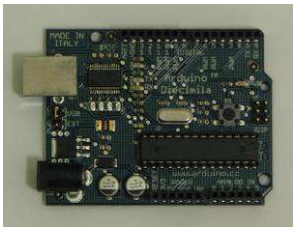


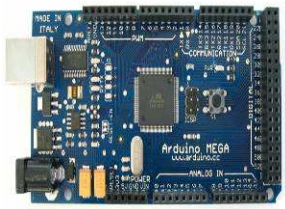

**Figura 3.1.** Distribución de los componentes que forman la Placa Arduino [7]

Para descargar los programas desde la aplicación Arduino hacia la placa existen diferentes interfaces dependiendo de la placa utilizada. Las diferentes interfaces son las siguientes: RS-232, USB y Bluetooth. En la actualidad la más utilizada es la conexión USB.

Arduino dispone de diferentes tipos de placas, con unas características muy similares, a nivel de entradas y salidas, chip, y otras funciones, pero difieren en la interfaz externa y en el método de comunicación entre ellas, si se da el caso. A continuación se detallan los diferentes tipos de placas existentes con algunas de sus características principales (Tabla 3.1.):

**Tabla 3.1.** Características de los principales tipos de placas Arduino

| Tipo de placa | Principales características |
|---|---|
|  **Figura 3.2.** Duemilanove [6] | • Microcontrolador Atmega 168<br>• Alimentación a 5 V<br>• 14 E/S ( 6 salidas PWM)<br>• 6 entradas analógicas<br>• 16 KB de Memoria Flash<br>• Frecuencia (clock): 16 Mhz.<br>• Comunicación serie a 19200 bauds<br>• Interfaz de comunicación con el PC: USB |
|  **Figura 3.3.** Diecimila [6] | • Microcontrolador ATmega 168<br>• Alimentación a 5 V<br>• 14 E/S ( 6 salidas PWM)<br>• 6 entradas analógicas<br>• 16 KB de Memoria Flash (2 KB utilizados para el bootloader.<br>• Frecuencia (clock): 16 Mhz.<br>• Comunicación serie a 19200 bauds<br>• Interfaz de comunicación con el PC: USB |

| | |
|---|---|
| <br>**Figura 3.4.** Bluetooth [6] | • Microcontrolador ATmega 168<br>• Alimentación entre 1,2 V y 5 V<br>• 14 E/S ( 6 salidas PWM)<br>• 6 entradas analógicas<br>• 16 KB de Memoria Flash<br>• Comunicación serie a 115200 bauds<br>• Interfaz de comunicación con el PC: Bluetooth<br>• Módulo Bluetooth Bluegiga WT11 |
|  <br>**Figura 3.5.** Mega [6] | • Microcontrolador ATmega 1280<br>• Alimentación a 5 V<br>• 54 E/S ( 14 salidas PWM)<br>• 16 entradas analógicas<br>• 128 KB de Memoria Flash (4 KB bootloader)<br>• Frecuencia (clock): 16 Mhz.<br>• Interfaz de comunicación con el PC: USB<br>• Reset automático en la descarga del programa |
| ***Nano***<br> <br>**Figura 3.6.** Nano [6] | • Microcontrolador ATmega 168<br>• Alimentación a 5 V<br>• 14 E/S ( 6 salidas PWM)<br>• 8 entradas analógicas<br>• 16 KB de Memoria Flash (2 KB bootloader)<br>• Interfaz de comunicación con el PC: Mini USB<br>• Dimensiones: 7,3 x 1,70 cm<br>• Reset automático en la descarga del programa |

Existen otros modelos de placas Arduino, no obstante, las detalladas en la tabla, son las más utilizadas por sus prestaciones.

Es posible realizar los diseños que se deseen mediante la aplicación software Arduino, que se puede descargar de forma gratuita y sin licencia del siguiente enlace: http://arduino.cc/en/Main/Software [26]. La aplicación es soportada por las plataformas Windows, MAC y Linux, y se debe descargar la versión que se desee particularizando el sistema operativo.

La última versión del software Arduino corresponde a la 0016. Las diferencias entre cada una de las aplicaciones radica principalmente en varias mejoras de la aplicación, y sobretodo, en la inclusión de nuevos modelos de placas, o bien de chips. En consecuencia si se dispone de un modelo de placa bastante actual, posiblemente no será posible utilizar una versión del programa inferior. Para comprobar, cual es la versión que se puede utilizar para los diferentes modelos de placa y chip, se debe acceder al enlace especificado en el párrafo anterior.

La aplicación Arduino dispone de una estructura muy simple de utilizar, con pocos menús y herramientas. En la Figura 3.7, se muestra la pantalla principal de la aplicación, en la cual se puede comprobar la división en tres partes: menú y barra estándar (compilación, nuevo, descarga del programa,…), desarrollo del programa, y finalmente, el monitor serie. El monitor serie resulta muy útil para realizar comprobaciones sobre el diseño, ya que se visualiza en tiempo real los datos que se desea mostrar. En consecuencia, se pueden determinar errores en la aplicación e ir realizando todos los cambios pertinentes, hasta observar que los valores obtenidos son los deseados.

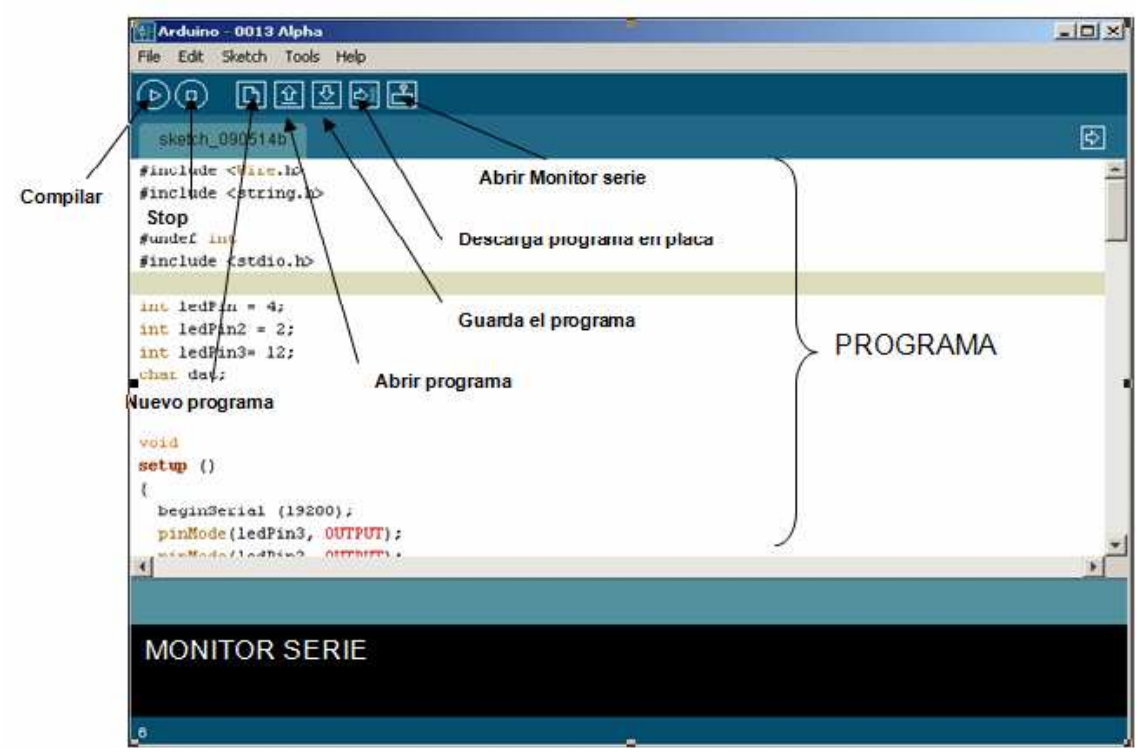



**Figura 3.7.** Pantalla principal de la aplicación Arduino – 0013

En la segunda parte, la referente al desarrollo del programa sigue la siguiente estructura:

- Declaración de librerías.
- Declaración de variables
- Void Setup (): Esta función se ejecuta cuando se inicializa el programa. Su función es la de inicializar variables, definir los modos de entrada o salida de los pines, indicar librerías, etc.
- Void Loop (): Como su nombre indica, esta función realiza un bucle, su función es controlar de forma activa la placa Arduino.

Por último, antes de descargar cualquier programa en la placa se deben realizar una serie de comprobaciones en nuestra placa y chip. Es importante conocer el modelo de placa que se está utilizando, así como del modelo de

chip Atmega, de lo contrario, en la descarga, la aplicación ocasionará una serie de errores, y el programa no será descargado en la placa de forma correcta. También es importante, comprobar el puerto serie asignado por el PC a la placa, que se debe corresponder con el seleccionado en la aplicación.

Cuando se conocen los datos descritos anteriormente, debemos configurarlos en la aplicación siguiendo los siguientes pasos:

- *Configuración de la placa*

Debemos acceder al menú Tools > Board > Modelo de placa. En el caso, que el modelo de placa que se dispone no aparezca en la lista, se comprobará si la versión del programa que se dispone es la correcta, en caso contrario, procederemos a la descarga des de la página oficial de Arduino, de la versión que se precise.

- *Configuración del puerto serie*

Accedemos al menú Tools > Serial Port > Número de puerto serie asignado por el PC. En el caso que el número de puerto serie no aparezca en la lista, comprobaremos si se ha instalado correctamente el puerto.

En el caso que la comunicación entre la placa Arduino y el PC se realice mediante la interfaz USB precisaremos previamente la instalación de los drivers correspondientes al chip FTDI (ver Figura 3.8.), sino el PC no será capaz de reconocer el dispositivo. En el siguiente enlace: http://www.ftdichip.com/FTDrivers.htm [22], es posible descargar los drivers del chip de forma gratuita, únicamente escogemos el sistema operativo en el cual tengamos instalada la aplicación Arduino.

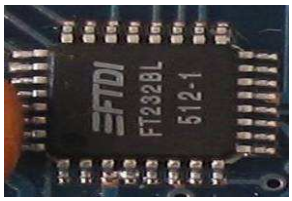



**Figura 3.8.** Imagen del chip FTDI contenido en la placa Arduino [22]

Cuando la configuración, así como el programa se ha realizado con éxito es posible descargar el programa en la placa. En primer lugar, y antes de oprimir el botón de Upload, reseteamos el bootloader de la placa, en el caso de que este no se resetee de forma automática, como es el caso de algún tipo de placas (ejemplo: Mega o nano). A continuación, procedemos a descargar el programa desde el botón Upload, y observaremos como los leds de la placa correspondiente a Tx y Rx, empezarán a parpadear simultáneamente. Si la descarga se ha realizado correctamente, en el monitor aparecerá la siguiente

afirmación “Done uploading”. Si este mensaje no es visible, es probable que la descarga no se haya realizado con éxito por diferentes motivos, en este caso, en el monitor aparecerá en rojo los diferentes errores que se hayan producido. En la página Web: www.arduino.cc [6] se detallan soluciones a diferentes errores.

En la Figura 3.9. se muestra de forma detallada la correspondencia de los pins del chip, en este caso del ATmega8 (igual que el caso del ATmega168 y del ATmega 328), y su relación con la nomenclatura utilizada en la librería correspondiente de la aplicación Arduino. Cuando deseemos hacer referencia a los diferentes pins en la aplicación, se deberá tener en cuenta la nomenclatura referenciada en rojo en la figura.

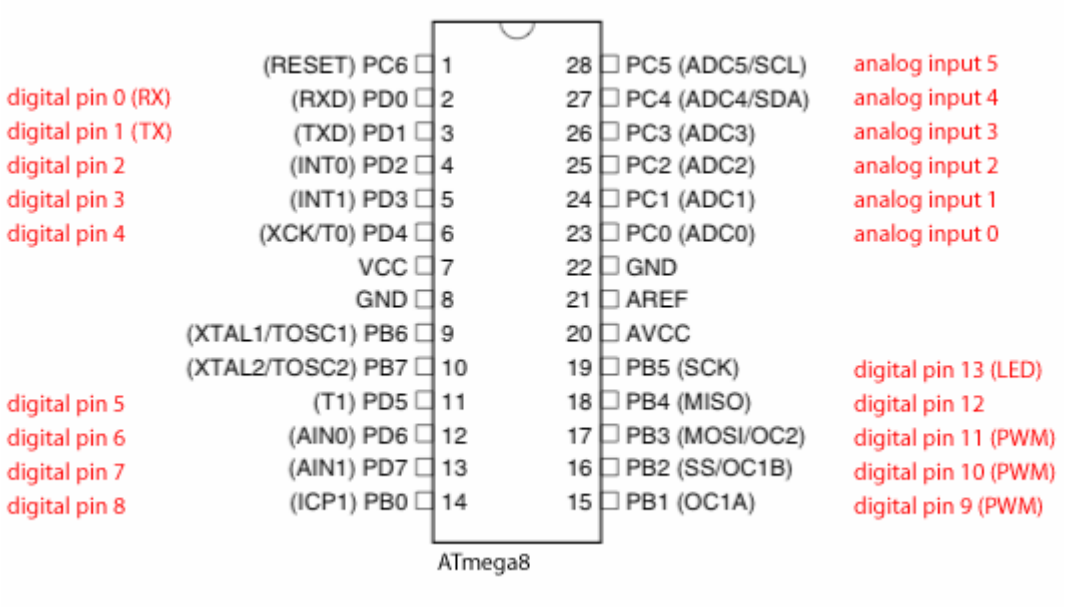



**Figura 3.9.** Esquema del conexionado de pins del chip ATmega8 [7].

### *3.1.1.2. Librerías de Arduino*

Mediante la aplicación Arduino es posible realizar multitud de programas mediante el lenguaje de programación de Arduino, basado prácticamente en su totalidad en el lenguaje C.

Los programas basados en Arduino se diferencian en tres partes: estructuras, variables y constantes, y funciones. En la Tabla 3.2. se muestran algunos ejemplos de cada una de estas partes:

**Tabla 3.2.** Estructuras, variables y funciones de **Arduino.**

| **Estructuras** | |
|---|---|
| Estructuras de control | If, else if, while, break, continue, return, etc. |
| Sintaxis adicional | Punto y coma, claudators, barras, etc. |
| Operadores aritméticos | Más, menos, multiplicación, división, etc. |
| Operadores de comparación | Igual a, menor que, mayor que, etc. |
| Operadores boléanos | Y, o, no. |
| Operadores compuestos | Incremento, decremento, etc. |
| **Variables** | |
| Constantes | HIGH/LOW, True/False, etc. |
| Tipo de datos | Int, char, bolean, flota, double, etc. |
| Conversión | Char (), byte(), int(), long(), flota(),etc. |
| **Funciones** | |
| Entradas y salidas digitales | pinMode, digitalWrite, etc. |
| Entradas y salidas analógicas | Int analogRead, analogWrite, etc. |
| Entradas y salidas avanzadas | ShiftOut, unsigned long PulseIn, etc. |
| Tiempo | Unsigned long millis (), delay(), etc. |
| Matemáticas | Min (x, y), max (x, y), sqrt(), etc. |
| Números aleatorios | randomSeed, long random , etc, |
| Comunicación | Serial |

Para ver más información acerca de la utilización de cada una de las estructuras, variables y funciones se puede acceder al siguiente enlace: http://arduino.cc/en/Reference/HomePage [8], donde se muestra de forma detallada como utilizar cada una de ellas, mediante el uso de algunos ejemplos.

También se dispone de diferentes librerías con diferentes funciones. En el caso de este proyecto se utiliza una de ellas de forma especial, la librería llamada “Wire Library”. Esta librería permite la comunicación con dispositivos I2C, mediante el pin 4 y 5 de la placa Arduino. En el pin 4 se coloca la señal de datos, mientras que en el pin 5 (entrada analógica), se conecta la entrada de reloj. Algunas de las funciones que forman parte de esta librería son las siguientes:

- Begin()
- Begin (adress)
- RequestFrom (adress, count)
- beginTransmission()
- endTransmission()
- send()
- byte available()
- byte receive()
- onReceive(handler)
- onRequest(handler)

### *3.1.1.3. Descripción del proceso*

Después de realizar una descripción de todas las herramientas y utilidades necesarias para la realización de la aplicación correspondiente (ver Figura 3.10), procedemos a realizar una descripción detallada de todo el proceso de desarrollo de la aplicación.

El material, los dispositivos y herramientas utilizados en la implementación de la aplicación son los siguientes:

- Un ordenador con sistema operativo Windows Vista
- Aplicación Arduino 0013
- Placa Arduino Duemilanove
- Nunchuck
- Cortina Bandalux motorizada
    - Modelo de los motores: Somfy LW 25-B44
- Quadruple Half-H Driver SN754410
- Un puerto USB
- Un cable USB, para la comunicación entre el Arduino y el PC.

En primer lugar, cortamos el cable que sale del Nunchuck y lo conectamos con el Wiimote, para posteriormente conectar cada uno de los hilos a diferentes entradas de la placa Arduino. Si no deseamos conectar el cable, existe una pequeña placa que se conecta al conector del final del cable, y que posteriormente se puede ensamblar perfectamente a la placa Arduino.

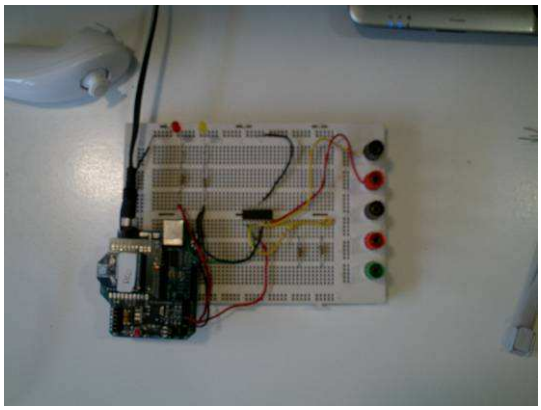


**Figura 3.10.** Fotografía del sistema completo de control remoto de una cortina con la plataforma Arduino XBee.

A continuación procedemos a la implementación de un programa que adquiera los datos provenientes del Nunchuck. Para ello, utilizamos el programa creado por Windmeadow Labs (ver Tabla 3.3.), que se puede encontrar en el siguiente enlace: http://www.windmeadow.com/node/42 [24], y que se muestra a continuación.

**Tabla 3.3.** Estructura del programa de adquisición de datos del Wiimote, mediante la plataforma Arduino. [24] (ver Anexo 8)

```
#include <Wire.h>
#include <string.h>

#undef int
#include <stdio.h>

uint8_t outbuf[6];            // array to store arduino output
int cnt = 0;
int ledPin = 13;

void
setup ()
{
  beginSerial (19200);
  Serial.print ("Finished setup\n");
  Wire.begin ();              // join i2c bus with address 0x52
  nunchuck_init (); // send the initilization handshake
}

void
nunchuck_init ()
{
  Wire.beginTransmission (0x52);      // transmit to device 0x52
  Wire.send (0x40);           // sends memory address
  Wire.send (0x00);           // sends sent a zero.
  Wire.endTransmission ();    // stop transmitting
}

void
send_zero ()
{
  Wire.beginTransmission (0x52);      // transmit to device 0x52
  Wire.send (0x00);           // sends one byte
  Wire.endTransmission ();    // stop transmitting
}

void
loop ()
{
  Wire.requestFrom (0x52, 6);  // request data from nunchuck
  while (Wire.available ())
    {
      outbuf[cnt] = nunchuk_decode_byte (Wire.receive ());   //
receive byte as an integer
      digitalWrite (ledPin, HIGH);    // sets the LED on
      cnt++;
    }

  // If we recieved the 6 bytes, then go print them
  if (cnt >= 5)
    {
      print ();
    }

  cnt = 0;
```

```
  send_zero (); // send the request for next bytes
  delay (100);
}

// Print the input data we have recieved
// accel data is 10 bits long
// so we read 8 bits, then we have to add
// on the last 2 bits.  That is why I
// multiply them by 2 * 2
void
print ()
{
  int joy_x_axis = outbuf[0];
  int joy_y_axis = outbuf[1];
  int accel_x_axis = outbuf[2] * 2 * 2;
  int accel_y_axis = outbuf[3] * 2 * 2;
  int accel_z_axis = outbuf[4] * 2 * 2;

  int z_button = 0;
  int c_button = 0;

 // byte outbuf[5] contains bits for z and c buttons
 // it also contains the least significant bits for the accelerometer
data
 // so we have to check each bit of byte outbuf[5]
  if ((outbuf[5] >> 0) & 1)
    {
      z_button = 1;
    }
  if ((outbuf[5] >> 1) & 1)
    {
      c_button = 1;
    }

  if ((outbuf[5] >> 2) & 1)
    {
      accel_x_axis += 2;
    }
  if ((outbuf[5] >> 3) & 1)
    {
      accel_x_axis += 1;
    }

  if ((outbuf[5] >> 4) & 1)
    {
      accel_y_axis += 2;
    }
  if ((outbuf[5] >> 5) & 1)
    {
      accel_y_axis += 1;
    }

  if ((outbuf[5] >> 6) & 1)
    {
      accel_z_axis += 2;
    }
  if ((outbuf[5] >> 7) & 1)
    {
      accel_z_axis += 1;
    }
```

```
  Serial.print (joy_x_axis, DEC);
  Serial.print ("\t");

  Serial.print (joy_y_axis, DEC);
  Serial.print ("\t");

  Serial.print (accel_x_axis, DEC);
  Serial.print ("\t");

  Serial.print (accel_y_axis, DEC);
  Serial.print ("\t");

  Serial.print (accel_z_axis, DEC);
  Serial.print ("\t");

  Serial.print (z_button, DEC);
  Serial.print ("\t");

  Serial.print (c_button, DEC);
  Serial.print ("\t");

  Serial.print ("\r\n");
}

// Encode data to format that most wiimote drivers except
// only needed if you use one of the regular wiimote drivers
char
nunchuk_decode_byte (char x)
{
  x = (x ^ 0x17) + 0x17;
  return x;
}
```

Con este programa de adquisición de datos del Nunchuck, obtenemos los valores de las 7 variables disponibles que se corresponden con los dos ejes del joystick (joy_x_axis y joy_y_axis), con los tres ejes del acelerómetro (accel_x_axis, accel_y_axis y accel_z_axis), y los dos botones (z_button y c_button). Los valores entre los cuales oscilan estas variables son los reflejados en la Tabla 3.4. :

**Tabla 3.4.** Umbral de valores de las diferentes variables de estado del Wiimote

| Variable | Intervalo de valores |
|---|---|
| Joy_x_axis | 0 – 255 |
| Joy_y_axis | 0 – 255 |
| Accel_x_axis | 0 – 1023 |
| Accel_y_axis | 0 – 1023 |
| Accel_z_axis | 0 – 1023 |
| C_button | 0 ó 1 |
| Z_button | 0 ó 1 |

Como podemos comprobar en el programa, incluimos la librería Wire comentada en apartados anteriores, que tiene la función de establecer comunicaciones series. Además observamos que la velocidad (baud rate) de comunicación serie de la placa Arduino utilizada (Duemilanove, ver Figura 3.2.), es de 19200 bps. En el caso de otras placas, esta velocidad puede variar, este es el caso de la placa Arduino Bluetooth, que dispone de una baud rate de 115200 bps.

Una vez hemos adquirido los datos correspondientes a las diferentes variables, establecemos diferentes acciones de salida, mediante los pins definidos como salidas. Previamente, hemos establecido una serie de intervalos de valores para definir a partir de que valor de cada variable, los motores de la cortina giran en un sentido, o en el inverso. Finalmente y después de realizar una serie de pruebas prácticas, con el uso del Nunchuck y el monitor serie de Arduino, hemos decidido establecer el siguiente rango de valores, detallados en la Tabla 3.5.:

**Tabla 3.5.** Umbral de valores para las diferentes acciones del sistema.

| Valor | Subida de la cortina | Bajada de la cortina |
|---|---|---|
| Joy_x_axis | >150 | <50 |
| Joy_y_axis | >150 | <50 |
| Accel_x_axis | >550 | <500 |
| Accel_y_axis | >550 | <500 |

A continuación hemos realizado la parte de código referente a las acciones que deseamos que se realicen cuando se cumple una de las condiciones expresadas en la tabla anterior. Para ello, hemos definido dos pins como salida (2 y 3), que se corresponden con el ledPin y el ledPin2. Cuando una de las condiciones anteriormente detalladas se cumple se escribe sobre ese pin de salida (ver Tabla 3.6.).

**Tabla 3.6.** Modificaciones del programa original (Tabla 3.3.), con el uso de los umbrales anteriores.

```
    if((joy_x_axis)<50)
  {
    digitalWrite(ledPin, HIGH);
    delay(20);
    digitalWrite(ledPin, LOW);
  }

    if((joy_x_axis)>150)
  {
    digitalWrite(ledPin2, HIGH);
    delay(20);
    digitalWrite(ledPin2, LOW);
```

```
  }

    if((joy_y_axis)>150)
  {
    digitalWrite(ledPin2, HIGH);
    delay(20);
    digitalWrite(ledPin2, LOW);
  }

    if((joy_y_axis)<50)
  {
    digitalWrite(ledPin, HIGH);
    delay(20);
    digitalWrite(ledPin, LOW);
  }


    if((accel_y_axis)>550)
  {
    digitalWrite(ledPin, HIGH);
    delay(20);
    digitalWrite(ledPin, LOW);
  }

    if((accel_y_axis)<500)
  {
    digitalWrite(ledPin2, HIGH);
    delay(20);
    digitalWrite(ledPin2, LOW);
  }
```

Como la salida del pin ofrece un voltaje que no es suficiente para los motores de la cortina, debemos establecer una solución a este inconveniente. Para ello estudiamos la posibilidad de insertar un driver de motores que pueda suministrar 24 V de continua a su salida. Después de comparar diferentes drivers existentes en el mercado, decidimos escoger el modelo SN754410 del fabricante Texas Instruments, que nos ofrece la salida que deseamos.

Utilizamos una fuente de alimentación para alimentar con el voltaje $V_{cc2}$ de 24 V de continua. Por otro lado para la alimentación del chip establecemos un voltaje de 5 V, ya que se permite un voltaje dentro del intervalo entre 4,5 V y 5,5 V. Para alimentar el chip aprovechamos uno de los pins de la placa Arduino Duemilanove que ofrece 5 V a su salida.

Tal y como podemos observar en la siguiente Figura 3.11. ( Imagen de la derecha), el driver dispone de 4 entradas y 4 salidas, de las cuales utilizamos únicamente la mitad.

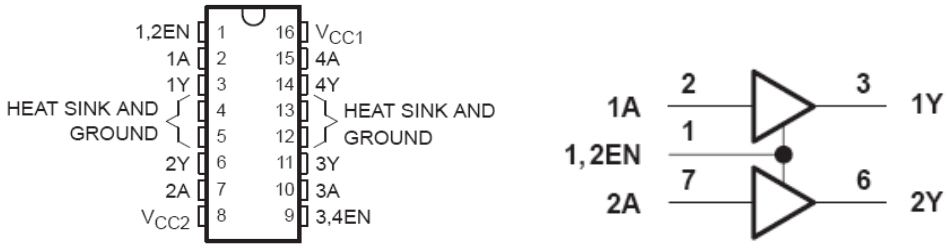



**Figura 3.11.** Diagrama de conexionado del chip SN754410 (a la izquierda), relación de entradas y salidas del chip (a la derecha).

Los pins 2 y 3 de salida de la placa Arduino los colocamos a las dos entradas del driver SN754410, concretamente a la 1A y a la 2A. Posteriormente, el cableado de la cortina lo conectamos a las dos salidas, la 1Y y 2Y.

Por último, comprobamos el funcionamiento de la totalidad del sistema, y verificamos que cuando se cumple una de las condiciones detalladas en la Tabla 3.5. , se realiza aquella acción que deseamos. También, realizamos una comprobación previa, antes de conectar la cortina, añadimos un led en cada salida, y verificamos como correctamente se encienden al realizar determinada acción.

### 3.1.2. Control remoto de una cortina motorizada

En esta aplicación tratamos de controlar remotamente una cortina motorizada mediante el uso de dos placas Arduino con dos módulos XBee, uno como envío de datos (procedentes del Nunchuck), y el otro como recepción de datos (procedentes del XBee de envío).

#### *3.1.2.1. Módulos XBee y la tecnología ZigBee*

Los módulos XBee (ver Figura 3.12.) permiten una comunicación serie entre ellos de señales TTL en distancias de 100 metros en exteriores sin ningún gran obstáculo por el medio y de 30 metros en interiores. No obstante, estas distancias varían según la potencia de transmisión del módulo y del tipo de antena del que dispone. Normalmente, estos sistemas disponen de una potencia que se encuentra entorno a 1 mW, y una antena dipolo.

El protocolo de comunicación usado por este tipo de dispositivos es el ZigBee, o también conocido como el estándar IEEE 802.15.4. Este estándar define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal (PAN) con tasas bajas de transmisión de datos.

Las comunicaciones ZigBee trabajan en la banda de 2,4 Ghz, y realiza las comunicaciones en un único canal, y por lo tanto con una única frecuencia.

Las principales aplicaciones de la redes ZigBee se centran en redes de sensores en entornos industriales, médicos y principalmente domóticas. Las redes pueden estar formadas por gran cantidad de dispositivos, entorno a los 65535 dispositivos, debido al número máximo de direcciones de red que es posible asignar.

Las redes ZigBee admiten gran diversidad de configuraciones inalámbricas, tales como, enlaces punto a punto, multipunto, peer to peer y otros tipos de configuraciones. En el caso que se muestra en este proyecto se utiliza la configuración más simple posible, un enlace punto a punto. Un módulo XBee actúa como emisor, y el otro como receptor, formando parte de la misma red de área personal (PAN).

Las principales ventajas que ofrecen estos módulos se resumen en 5 características:

- **Bajo Coste**. Al ser dispositivos de baja potencia y de antenas con un dipolo simple, su coste no es elevado.

- **Bajo consumo de potencia**. Tal y como se ha descrito anteriormente, el consumo esta entorno a 1 mW.

- **Redes flexibles y extensibles**. Sus amplias posibilidades de configuración y la gran cantidad de equipos que pueden llegar a formar parte de la misma, hacen que sean redes muy flexibles.

- **Instalación simple y barata**.

- **Bandas libres**. Los módulos trabajan en una banda que no es necesaria ningún tipo de licencia y además trabaja en un canal de los 16 posibles.

Otra característica importante, es que también presenta una dirección MAC como los dispositivos Ethernet, que los diferencia entre ellos, ya que no es posible que exista dos dispositivos con la misma dirección MAC.

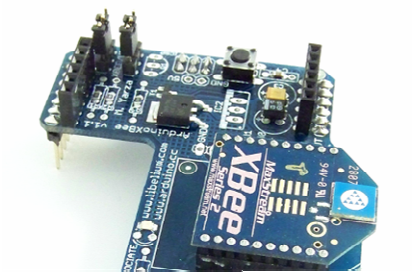



**Figura 3.12.** Módulo XBee [17]

### *3.1.2.2. Descripción del proceso*

Después de realizar una descripción de los elementos necesarios para la realización de la aplicación correspondiente, procedemos a realizar una descripción detallada de todo el proceso de desarrollo de la aplicación.

El material, los dispositivos y herramientas utilizados en la implementación de la aplicación son los siguientes:

- Un ordenador con sistema operativo Windows Vista
- Aplicación Arduino 0013
- Placa Arduino Duemilanove
- Nunchuck
- Cortina Bandalux motorizada
    - Modelo de los motores: Somfy LW 25-B44
- Quadruple Half-H Driver SN754410
- Un puerto USB
- Un cable USB, para la comunicación entre el Arduino y el PC.

En primer lugar, y a diferencia de la anterior aplicación configuramos los dos módulos XBee por separado, asignando a uno el rol de envío, y otro con el rol de recepción. Para ello, es posible utilizar cualquier programa de comunicación serie, como por ejemplo el Hyperterminal, que está incluido en los sistemas operativos Windows. Sin embargo, por mayor simplicidad en la configuración, escogemos un software de la empresa DIGI, llamado X-CTU, y que es posible descargarlo de forma gratuita mediante el siguiente enlace: http://www.digi.com/support/kbase/kbaseresultdetl.jsp?kb=125 [19].

Accedemos al programa X-CTU y realizamos los siguientes pasos:

1. Accedemos al programa X-CTU.
2. En la pestaña PC Settings, escogemos el puerto COM que ha sido asignado al Arduino XBee.
3. Seleccionamos el modelo de Modem: XB-24.

4. Oprimimos el botón llamado Read.
5. Existen dos opciones para introducir los parámetros mediante comunicación Terminal (similar a Hyperterminal), es decir, mediante la pestaña Terminal. No obstante, existe otra pestaña llamada Modem Configuration que permite la configuración de una forma más intuitiva. Únicamente configuramos los parámetros que aparecen en la Tabla 3.7.
6. Finalmente, para que los cambios en los diferentes parámetros modificados se transmitan al módulo, presionamos el botón Write.

En la Figura 3.13., podemos observar la pantalla principal del programa de configuración de dispositivos con comunicación serie.

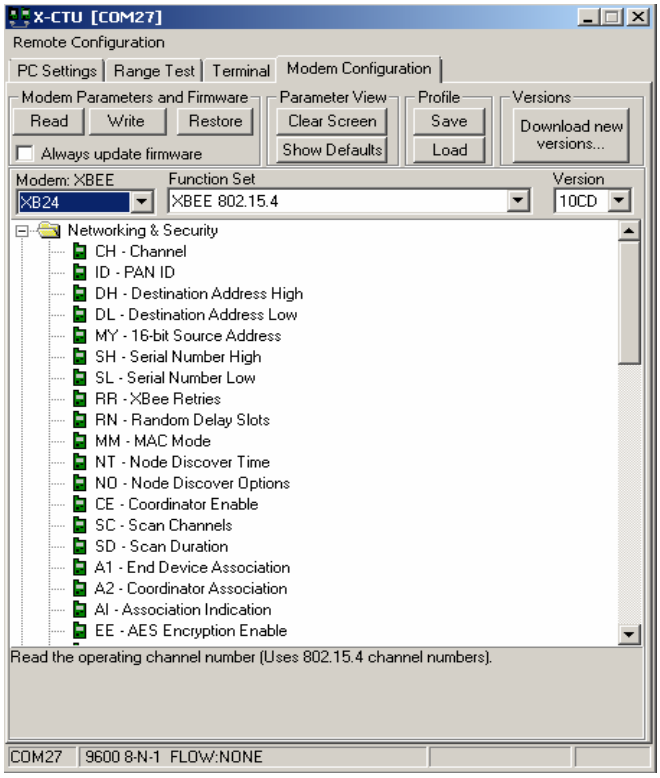
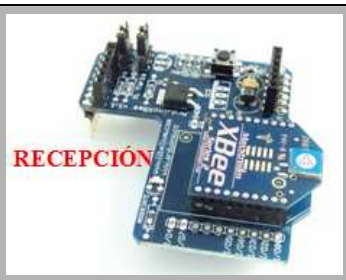



**Figura 3.13.** Pantalla principal del programa X-CTU de configuración de las comunicaciones entre módulos XBee [19]

**Tabla 3.7.** Configuración de los módulos XBee emisor, y receptor [14].

| Placa (Dirección Comunicación) / Valor de configuración | **Ref: S9706** |   **Ref: S9379** |
|---|---|---|
| ATID | ATID3332 | ATID3332 |
| DH | DH0 | DH0 |
| DL | DL1 | DL0 |
| MY | MY0 | MY1 |
| BD | BD4 | BD4 |
| WR | WR | WR |
| CN | CN | CN |

El parámetro ATID define el identificador de la red personal (PAN (Personal Area Network).

Los parámetros configurados anteriormente (ver Tabla 3.7.) para la comunicación de los dos módulos Xbee tienen el siguiente significado:

- **DH**: Ajusta los 32 bits más significativos para direccionamiento. Defecto=0.

- **DL**: Ajusta los 32 bits menos significativos para direccionamiento. Defecto=0.

- **MY**: Configura la dirección de 16 bits para el módulo. Si MY=0xFFFF o 0XFFFE, se habilita el modo de direccionamiento de 64 bit. Defecto =0.
    - **0** : 8 bit sin paridad o 7 bit con cualquier paridad.
    - **1** : 8 bit even.
    - **2** : 8 bit odd.

- **BD**: Modo API habilitado con carácter de escape. Ajusta la tasa de transmisión del módulo y su cliente conectado a través de la interfaz serial. Defecto=3.
    - **0**: 1200
    - **1**: 2400
    - **2**: 4800
    - **3**: 9600
    - **4**: 19200
    - **5**: 38400
    - **6**: 57600

- **WR**: Guarda en la memoria no-volátil del módulo, todos los valores de los parámetros.

- **CN**: Sale del modo de Comando.

A continuación procedemos la conexión del Nunchuck a la placa Arduino que tiene la función de emisor (Ref: S9706) según el procedimiento establecido en el apartado .

En este caso hemos realizado dos aplicaciones diferentes, una para una placa y otro para la otra. En la placa emisora ponemos la aplicación de adquisición de datos provenientes del Nunchuck, y que sean enviados por comunicación serie. En cambio, en la aplicación que colocamos en la placa receptora, adquirimos los datos serie provenientes del modulo XBee de la placa emisora, se procesan, y se emiten unas salidas, en este caso hacia el driver de los motores.

En la Tabla 3.8. se muestra, la parte del programa de la placa Arduino emisora, que no se diferencia de la anterior aplicación, ya que la adquisición de los datos del Nunchuck se realiza de la misma forma. Como se puede observar, en este caso, en la acción sobre el resultado de las condiciones de los intervalos, realizamos una transmisión serie de una letra (**Serial.print**).

**Tabla 3.8.** Programa de control remoto, referente al módulo emisor.

```
if((joy_x_axis)>=130)
{
  Serial.print('A');
}

else if((joy_x_axis)<=125)
{
  Serial.print('a');
}

else if((joy_y_axis)>=129)
{
  Serial.print('B');
}

else if((joy_y_axis)<=125)
{
  Serial.print('b');
}

else
if((((accel_x_axis)>685)&&((accel_x_axis)<715)&&((accel_y_axis)>485)&&((acce
l_y_axis)<515)))
{
  Serial.print('C');
}

else
if((((accel_x_axis)>285)&&((accel_x_axis)<315)&&((accel_y_axis)>485)&&((acce
l_y_axis)<515)))
```

```
{
  Serial.print('c');
}

else if((accel_y_axis)<450)
{
  Serial.print('D');
}

else if((accel_y_axis)>=550)
{
  Serial.print('d');
}

else if (c_button=1)
{
  Serial.print('H');
}

else if(c_button=0)
{
  Serial.print('L');
}
```

En lo que hace referencia a la programación en la placa Arduino receptora (ver Tabla 3.9.), interpretamos los valores enviados por la comunicación serie. Para ello, mediante la función **Serial.read** leemos los datos que llegan al módulo, y a partir de ese dato se determina la acción a realizar.

Existe la posibilidad de variar la velocidad referente a la subida y la bajada de la cortina mediante el uso de alguna de las salidas PWM incluidas en la placa Arduino Duemilanove. Para ello, declaramos la salida de los pins referentes a estas salidas PWM. A continuación, sustituimos la función **digitalWrite** por la función **analogWrite(pin, value)**. La variable value se corresponde al ciclo de trabajo (duty cicle), y comprende un intervalo de valores entre 0 (0% de DC) y 255 (100% de DC). Para un valor de 0 de la variable, corresponderán a 0 V, en cambio, para un valor de 255, se establecerán aproximadamente un voltaje de 5 V.

**Tabla 3.9.** Programa de control remoto, referente al módulo receptor.

```
dat=Serial.read();

if(dat=='A')
{
  digitalWrite(ledPin, HIGH);
  delay(100);
  digitalWrite(ledPin, LOW);
}
else if(dat=='a')
{
  digitalWrite(ledPin2, HIGH);
  delay(100);
  digitalWrite(ledPin2, LOW);
}
if(dat=='B')
```

```
{
  digitalWrite(ledPin, HIGH);
  delay(100);
  digitalWrite(ledPin, LOW);
}
else if(dat=='b')
{
  digitalWrite(ledPin2, HIGH);
  delay(100);
  digitalWrite(ledPin2, LOW);
}
if(dat=='C')
{
  digitalWrite(ledPin, HIGH);
  delay(100);
  digitalWrite(ledPin, LOW);
}
else if(dat=='c')
{
  digitalWrite(ledPin2, HIGH);
  delay(100);
  digitalWrite(ledPin2, LOW);
}
if(dat=='D')
{
  digitalWrite(ledPin, HIGH);
  delay(100);
  digitalWrite(ledPin, LOW);
}
else if(dat=='d')
{
  digitalWrite(ledPin2, HIGH);
  delay(100);
  digitalWrite(ledPin2, LOW);
}

else if(dat=='H')
{
  digitalWrite(ledPin3, HIGH);
  delay(100);
  digitalWrite(ledPin3, LOW);
}
```

Como la salida del pin ofrece un voltaje que no es suficiente para los motores de la cortina, establecemos una solución a este inconveniente. Para ello estudiamos la posibilidad de insertar un driver de motores que pueda suministrar 24 V de continua a su salida. Después de comparar diferentes drivers existentes en el mercado, escogemos el modelo SN754410 del fabricante Texas Instruments, que nos ofrece la salida que deseamos (24 V).

Utilizamos una fuente de alimentación para alimentar con el voltaje $V_{cc2}$ de 24 V de continua. Por otro lado para la alimentación del chip establecemos un voltaje de 5 V, ya que se permite un voltaje dentro del intervalo entre 4,5 V y 5,5 V. Para alimentar el chip aprovechamos uno de los pins de la placa Arduino Duemilanove que ofrece 5 V a su salida.

Tal y como observamos en la Figura 3.11. (Imagen de la derecha), el driver dispone de 4 entradas y 4 salidas, de las cuales utilizamos únicamente la mitad.

Los pins 2 y 3 de salida de la placa Arduino los colocamos a las dos entradas del driver SN754410 (ver detalle en la Tabla A.4.1 y Tabla A.4.2.), concretamente a la 1A y a la 2A. Posteriormente, el cableado de la cortina lo conectamos a las dos salidas, la 1Y y 2Y.

Por último, verificamos el resultado final, con todo el sistema montado (ver Figura 3.14.), y efectivamente, comprobamos que se cumplen los criterios establecidos en la aplicación, además verificamos que se lleva a cabo la acción correcta.

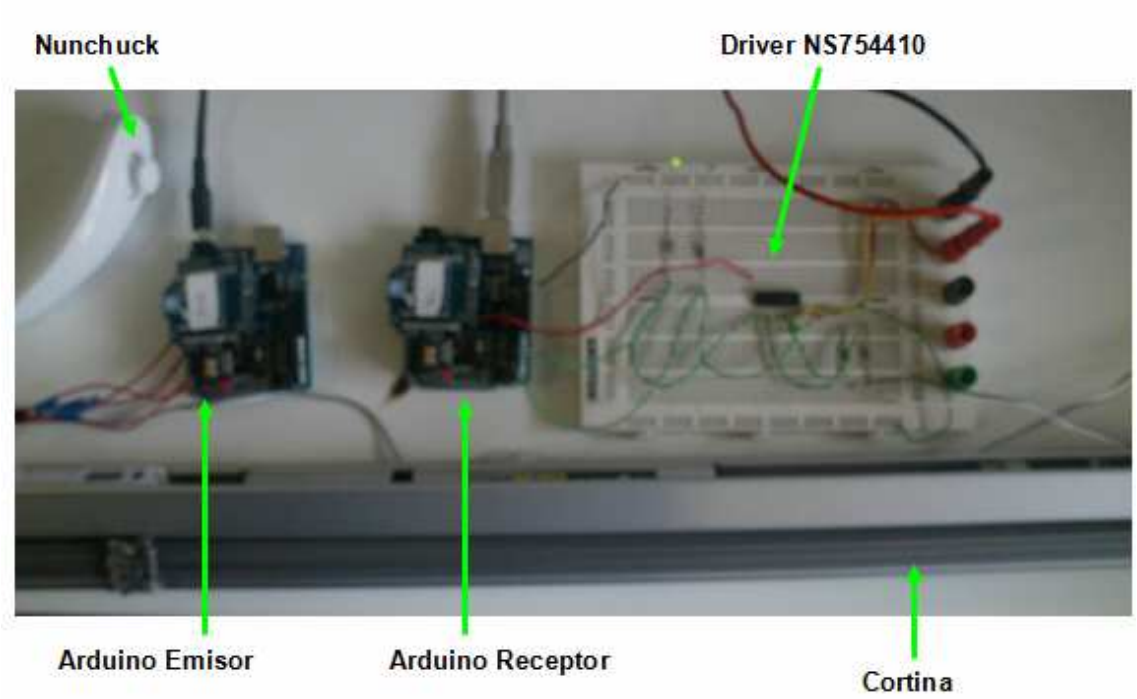



**Figura 3.14.** Sistema completo del control remoto de la cortina, mediante el Nunchuck

# CONCLUSIONES

El Wiimote puede ser usado como interfaz de control. La comprensión del funcionamiento interno del Wiimote, su estudio, y su uso en proyectos requiere un aprendizaje multidisciplinario, que permite aprender y experimentar con:

- Entornos de programación de alto nivel, como por ejemplo LabVIEW.
- Entornos de programación de bajo nivel, como por ejemplo Arduino.
- Diseño de pequeñas estructuras electrónicas.
- Configuración de enlaces inalámbricos, como el caso de los módulos XBee.
- Estudio de diferentes protocolos de comunicación (Bluetooth, ZigBee e I2C).

Este trabajo, dada las herramientas utilizadas, como el Wiimote y el Arduino, representan una solución a muchos de los problemas de aprendizaje e interacción del alumno con la tecnología, y permite trabajar una serie de competencias y habilidades, como por ejemplo:

- El trabajo en grupo.
- La planificación de tareas.
- Resolución de problemas.
- Desarrollo de iniciativas creativas e innovadoras.
- Coordinación y cooperación entre los diferentes componentes del grupo.
- Comunicación eficaz.
- Responsabilidad.

Hemos comprobado la imposibilidad de control Bluetooth directo del Wiimote. En todos los casos, una aplicación realizada en un entorno de programación y un PC, realizan la función de interfaz, entre el Wiimote y el dispositivo a controlar.

Gracias al acelerómetro incluido en el Wiimote, el manejo de cualquier estructura se realiza de forma más intuitiva, que en el caso de un joystick o mediante botones. No obstante, en el movimiento del Wiimote es difícil moverse en un único eje del acelerómetro. Por ello, y ante las necesidades de las aplicaciones desarrolladas, es necesario incluir unos umbrales de movimiento en cada uno de los ejes.

Es muy interesante la utilización del estándar Bluetooth HID, ya que la mayoría de sistemas operativos que podemos encontrar en el mercado reconocen perfectamente dispositivos USB HID, tales como teclados o ratones, sin la instalación de ningún driver especial.

En este trabajo hemos utilizado dos herramientas de programación, como son LabVIEW y el entorno de desarrollo de Arduino, dada su gran implantación en la docencia. Ambas nos han permitido aprender de una forma gradual cada una de sus peculiaridades. En particular, el uso de LabVIEW en la docencia

persiste des de hace varios años, como consecuencia de la gran cantidad de librerías que contiene, que permiten la interacción con multitud de instrumentos y dispositivos electrónicos.

Este trabajo ofrece las bases para un posterior desarrollo de nuevas aplicaciones y proyectos.

# ESTUDIO DE AMBIENTALIZACIÓN

Tanto el Wiimote, como la placa Arduino, cumplen los requisitos esenciales establecidos en la Directiva Europea 1999/5/CE, en la que se establecen unas directrices en relación a los componentes utilizados en la fabricación del aparato y el proceso de ensayos específicos a los que se ha sometido.

Al final de la vida útil de cada uno de estos aparatos, se debe depositar en contenedores de residuos de aparatos eléctricos y electrónicos, para su posterior reciclaje.

El uso de pilas como fuente de alimentación, tanto del NXT Lego como del Wiimote es perjudicial para la salud como para el medioambiente, si posteriormente no se realiza un buen reciclaje de las mismas. Las pilas contienen materiales metálicos pesados que resultan perjudiciales para la salud. Con el paso del tiempo la capa que las protege se descompone fácilmente, y si estas se encuentran depositadas en un lugar no adecuado, como por ejemplo en el suelo, los residuos se filtran y pueden llegar a contaminar el agua, con lo que ello comporta. En consecuencia es recomendable el uso de baterías recargables, o bien conexión directa con fuente de corriente, que permiten una duración más larga, y evita la sucesiva sustitución de gran cantidad de pilas.

# BIBLIOGRAFÍA

## *Libros*

[1] Gasperi M. *LabVIEW for LEGO MINDSTORMS NXT*. National Technology and Science Press. 2008.

## *Paginas Web*

[2] http://es.wikipedia.org/wiki/Control_Remoto_Wii

[3] http://www.cristalab.com/blog/dentro-del-wiimote-del-nintendo-wii-c37627l/

[4] http://wiki.wiimoteproject.com/Reports

[5] http://www.wiire.org/Wii/wiimote

[6] http://www.arduino.cc

[7] http://www.arduino.cc/es/

[8] http://arduino.cc/en/Reference/HomePage

[9] http://www.ladyada.net/learn/arduino/lesson5.html

[10] http://onakasuita.org/wii/index-e.html

[11] http://www.comunidadelectronicos.com/articulos/i2c.htm

[12] http://es.wikipedia.org/wiki/Nunchuk

[13] http://todbot.com/blog/2008/02/18/wiichuck-wii-nunchuck-adapter-available/

[14] http://www.scribd.com/doc/13069890/Primeros-Pasos-Con-Arduino-y-XBee

[15] http://forums.ni.com/ni/board/message?board.id=170&thread.id=249428

[16] http://www.somfy.es

[17] http://www.caveo.com.ar/modulos_xbee.htm

## *Software y herramientas*

[18] http://zone.ni.com/devzone/cda/tut/p/id/4435

[19]http://www.digi.com/support/kbase/kbaseresultdetl.jsp?kb=125

[20] http://decibel.ni.com/content/docs/DOC-1353.

[21] http://www.codeplex.com/WiimoteLib

[22] http://www.ftdichip.com/FTDrivers.htm

[23] http://johnnylee.net/projects/wii/

[24] http://www.windmeadow.com/node/42

[25] http://www.elotrolado.net/wiki/Glovepie

[26] http://arduino.cc/en/Main/Software

# ANEXOS

# ANEXO I: Acelerómetro del Wiimote, modelo ADXL330

**Tabla A.1.1.** Especificaciones técnicas del acelerómetro ADXL330

| Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| SENSOR INPUT | Each axis | | | | |
| Measurement Range | | ±3 | ±3.6 | | g |
| Nonlinearity | % of full scale | | ±0.3 | | % |
| Package Alignment Error | | | ±1 | | Degrees |
| Inter-Axis Alignment Error | | | ±0.1 | | Degrees |
| Cross Axis Sensitivity[1] | | | ±1 | | % |
| SENSITIVITY (RATIOMETRIC)[2] | Each axis | | | | |
| Sensitivity at $X_{OUT}$, $Y_{OUT}$, $Z_{OUT}$ | $V_S$ = 3 V | 270 | 300 | 330 | mV/g |
| Sensitivity Change Due to Temperature[3] | $V_S$ = 3 V | | ±0.015 | | %/°C |
| ZERO g BIAS LEVEL (RATIOMETRIC) | Each axis | | | | |
| 0 g Voltage at $X_{OUT}$, $Y_{OUT}$, $Z_{OUT}$ | $V_S$ = 3 V | 1.2 | 1.5 | 1.8 | V |
| 0 g Offset vs. Temperature | | | ±1 | | mg/°C |
| NOISE PERFORMANCE | | | | | |
| Noise Density $X_{OUT}$, $Y_{OUT}$ | | | 280 | | μg/√Hz rms |
| Noise Density $Z_{OUT}$ | | | 350 | | μg/√Hz rms |
| FREQUENCY RESPONSE[4] | | | | | |
| Bandwidth $X_{OUT}$, $Y_{OUT}$[5] | No external filter | | 1600 | | Hz |
| Bandwidth $Z_{OUT}$[5] | No external filter | | 550 | | Hz |
| $R_{FB}$ Tolerance | | | 32 ± 15% | | kΩ |
| Sensor Resonant Frequency | | | 5.5 | | kHz |
| SELF-TEST[6] | | | | | |
| Logic Input Low | | | +0.6 | | V |
| Logic Input High | | | +2.4 | | V |
| ST Actuation Current | | | +60 | | μA |
| Output Change at $X_{OUT}$ | Self-test 0 to 1 | | −150 | | mV |
| Output Change at $Y_{OUT}$ | Self-test 0 to 1 | | +150 | | mV |
| Output Change at $Z_{OUT}$ | Self-test 0 to 1 | | −60 | | mV |
| OUTPUT AMPLIFIER | | | | | |
| Output Swing Low | No load | | 0.1 | | V |
| Output Swing High | No load | | 2.8 | | V |
| POWER SUPPLY | | | | | |
| Operating Voltage Range | | 2.0 | | 3.6 | V |
| Supply Current | $V_S$ = 3 V | | 320 | | μA |
| Turn-On Time[7] | No external filter | | 1 | | ms |
| TEMPERATURE | | | | | |
| Operating Temperature Range | | −25 | | +70 | °C |

# ANEXO 2: Acelerómetro del Nunchuck, modelo LIS3L02AL

**Tabla A.2.1.** Características mecánicas del acelerómetro LIS3L02AL

(Temperature range -40°C to +85°C) All the parameters are specified @ Vdd =3.3V, T = 25°C unless otherwise noted

| Symbol | Parameter | Test Condition | Min. | Typ.(2) | Max. | Unit |
|---|---|---|---|---|---|---|
| Ar | Acceleration Range(3) | | ±1.8 | ±2.0 | | g |
| So | Sensitivity(4) | Full-scale = 2g | Vdd/5–10% | Vdd/5 | Vdd/5+10% | V/g |
| SoDr | Sensitivity Change Vs Temperature | Delta from +25°C | | ±0.01 | | %/°C |
| Voff | Zero-g Level(4) | T = 25°C | Vdd/2-6% | Vdd/2 | Vdd/2+6% | V |
| OffDr | Zero-g level Change Vs Temperature | Delta from +25°C | | ±0.5 | | mg/°C |
| NL | Non Linearity(5) | Best fit straight line<br>Full-scale = 2g<br>X, Y axis | | ±0.3 | ±1.5 | % |
| | | Best fit straight line<br>Full-scale = 2g<br>Z axis | | ±0.5 | ±1.5 | % |
| CrossAx | Cross-Axis(6) | | | ±2 | ±4 | % |
| An | Acceleration Noise Density | Vdd=3.3V;<br>Full-scale = 2g | | 50 | | $\mu g/\sqrt{Hz}$ |
| Vt | Self test Output Voltage Change(7),(8) | T = 25°C<br>Vdd=3.3V<br>Full-scale = 2g<br>X axis | -20 | -50 | -100 | mV |
| | | T = 25°C<br>Vdd=3.3V<br>Full-scale = 2g<br>Y axis | 20 | 50 | 100 | mV |
| | | T = 25°C<br>Vdd=3.3V<br>Full-scale = 2g<br>Z axis | 20 | 50 | 100 | mV |
| Fres | Sensing Element Resonance Frequency(9) | all axes | 1.5 | | | kHz |
| Top | Operating Temperature Range | | -40 | | +85 | °C |
| Wh | Product Weight | | | 0.08 | | gram |

Obsolete Product(s) - Obsolete Product(s)

**Tabla A.2.2.** Características eléctricas del acelerómetro LIS3L02AL

(Temperature range -40°C to +85°C) All the parameters are specified @ Vdd =3.3V, T=25°C unless otherwise noted

| Symbol | Parameter | Test Condition | Min. | Typ.(2) | Max. | Unit |
|---|---|---|---|---|---|---|
| Vdd | Supply Voltage | | 2.4 | 3.3 | 3.6 | V |
| Idd | Supply Current | mean value | | 0.85 | 1.5 | mA |
| Vst | Self Test Input | Logic 0 level | 0 | | 0.3*Vdd | V |
| | | Logic 1 level | 0.7*Vdd | | Vdd | V |
| Rout | Output Impedance | | 80 | 110 | 140 | kΩ |
| Cload | Capacitive Load Drive(3) | | 1 | | | nF |
| Top | Operating Temperature Range | | -40 | | +85 | °C |

Obsolete Product(s)

# ANEXO 3: Especificaciones técnicas del motor de la cortina suministrada por Bandalux.

En el siguiente anexo se muestra una tabla con las características técnicas referentes a los motores de la cortina veneciana utilizada en el desarrollo de algunas aplicaciones detalladas en este trabajo.

**Tabla A.3.1.** Especificaciones técnicas de los motores Somfy LW 25-B44

| Característica | Valor |
|---|---|
| Nominal torque (Nm) | 0.4 |
| Stall torque at 25 ºC (nm) | 0.8 |
| Nominal Speed (r.p.m) | 40 |
| No load speed (r.p.m) | 50 |
| Maximum number of turns in one direction | - |
| Current consumption at nominal torque (mA) | 300 |
| Current consumption at stall (mA) | 450 |
| Nominal supply voltage (Vdc) | 24 |
| Minimum supply voltage (Vdc) | 20 |
| Maximum off-center possibility of output shaft (mm) | +/-0.7 |
| Output shaft play | 65º |
| Maximum Difference in the 2 output shaft positioning | 30º |
| Maximum running time in blind (mn) | 7 |
| Working temperature | -10 to 60º |
| Protection Index | Ip40 |
| Dielectric strength with a Safety Extra Low Voltage | III |
| Standards | CE |
| Supply cable type ($mm^2$) | 2 x 0.25 |
| Supply cable length (m) | 2.5 |
| Net Weight (g) | 195 |

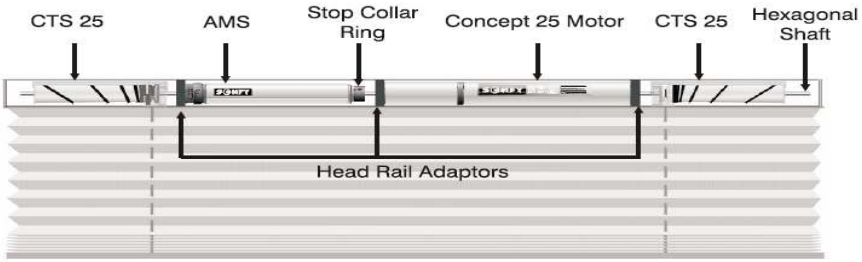



**Figura A.3.1.** Esquema que representa la ubicación de los motores LW 25-B44 de Somfy

# ANEXO 4: Driver SN754410

**Tabla A.4.1.** Especificaciones técnicas del driver

| | MIN | MAX | UNIT |
|---|---|---|---|
| Output supply voltage, $V_{CC1}$ | 4.5 | 5.5 | V |
| Output supply voltage, $V_{CC2}$ | 4.5 | 36 | V |
| High-level input voltage, $V_{IH}$ | 2 | 5.5 | V |
| Low-level input voltage, $V_{IL}$ | −0.3‡ | 0.8 | V |
| Operating virtual junction temperature, $T_J$ | −40 | 125 | °C |
| Operating free-air temperature, $T_A$ | −40 | 85 | °C |

**Tabla A.4.2.** Especificaciones técnicas del driver detalladas

| PARAMETER | | TEST CONDITIONS | | MIN | TYP† | MAX | UNIT |
|---|---|---|---|---|---|---|---|
| $V_{IK}$ | Input clamp voltage | $I_I = -12$ mA | | | −0.9 | −1.5 | V |
| $V_{OH}$ | High-level output voltage | $I_{OH} = -0.5$ A | | $V_{CC2}-1.5$ | $V_{CC2}-1.1$ | | V |
| | | $I_{OH} = -1$ A | | $V_{CC2}-2$ | | | |
| | | $I_{OH} = -1$ A, $T_J = 25°C$ | | $V_{CC2}-1.8$ | $V_{CC2}-1.4$ | | |
| $V_{OL}$ | Low-level output voltage | $I_{OL} = 0.5$ A | | | 1 | 1.4 | V |
| | | $I_{OL} = 1$ A | | | | 2 | |
| | | $I_{OL} = 1$ A, $T_J = 25°C$ | | | 1.2 | 1.8 | |
| $V_{OKH}$ | High-level output clamp voltage | $I_{OK} = -0.5$ A | | | $V_{CC2}+1.4$ | $V_{CC2}+2$ | V |
| | | $I_{OK} = 1$ A | | | $V_{CC2}+1.9$ | $V_{CC2}+2.5$ | |
| $V_{OKL}$ | Low-level output clamp voltage | $I_{OK} = 0.5$ A | | | −1.1 | −2 | V |
| | | $I_{OK} = -1$ A | | | −1.3 | −2.5 | |
| $I_{OZ(off)}$ | Off-state high-impedance-state output current | $V_O = V_{CC2}$ | | | | 500 | µA |
| | | $V_O = 0$ | | | | −500 | |
| $I_{IH}$ | High-level input current | $V_I = 5.5$ V | | | | 10 | µA |
| $I_{IL}$ | Low-level input current | $V_I = 0$ | | | | −10 | µA |
| $I_{CC1}$ | Output supply current | $I_O = 0$ | All outputs at high level | | | 38 | mA |
| | | | All outputs at low level | | | 70 | |
| | | | All outputs at high impedance | | | 25 | |
| $I_{CC2}$ | Output supply current | $I_O = 0$ | All outputs at high level | | | 33 | mA |
| | | | All outputs at low level | | | 20 | |
| | | | All outputs at high impedance | | | 5 | |

# ANEXO 5: Sistema completo del control de la estructura NXT LEGO, mediante el Wiimote

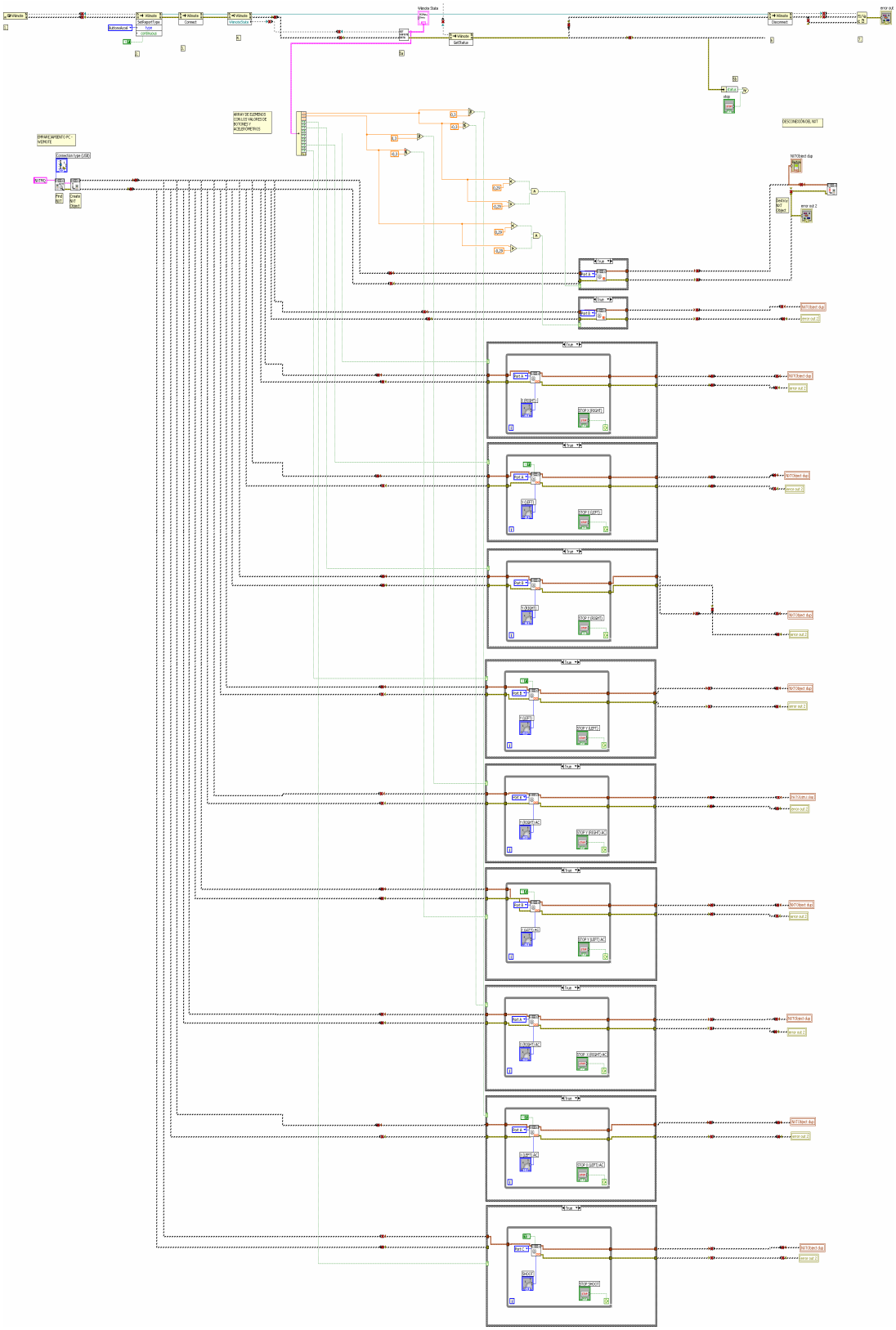


**Figura A.5.1.** Sistema completo de la aplicación

# ANEXO 6: Referencia de los proyectos realizados por Johny Cheng Lee



## Johnny Chung Lee > Projects > Wii

As of June 2008, Nintendo has sold nearly 30 million Wii game consoles. This significantly exceeds the number of Tablet PCs in use today according to even the most generous estimates of Tablet PC sales. This makes the Wii Remote one of the most common computer input devices in the world. It also happens to be one of the most sophisticated. It contains a 1024x768 infrared camera with built-in hardware blob tracking of up to 4 points at 100Hz. This significantly out performs any PC "webcam" available today. It also contains a +/-3g 8-bit 3-axis accelerometer also operating at 100Hz and an expandsion port for even more capability. These projects are an effort to explore and demonstrate applications that the millions of Wii Remotes in world readily support.

Any software on this page is primarily meant for developers and may not run without proper the development tools installed. Downloading and installing this software is **at your own risk**, and no support or guarantee is provided with this software. The official discussion forums for my wiimote projects can be found here: **WiimoteProject.com**

**NOTE:** For most of these projects, you don't need the Nintendo Wii console. You only need the Wii controller and a bluetooth connection.

### Tracking Your Fingers with the Wiimote

Using an LED array and some reflective tape, you can use the infrared camera in the Wii remote to track objects, like your fingers, in 2D space. This lets you interact with your computer simply by waving your hands in the air similar to the interaction seen in the movie "Minority Report". The Wiimote can track upto 4 points simultaneously. The multipoint grid software is a custom C# DirectX program.

**Software**
To run the grid program you see in the video:
**1.** First, follow this walkthrough on using the wiimote with C#. You may need to download a copy of Visual C# Express to compile/run this sample if you don't have it yet.
**2.** Download a copy of the DirectX SDK. You may not need this to simply run the sample grid program, but you will need it if you want to make any changes to it.
**3.** Download the Wiimote Multipoint Grid sample program. Make sure your wiimote is connected via bluetooth, and then run the ".exe" shortcut in the main folder.

A visit to this project's **FAQ and Advanced Discussion** post may be very englightening. You may also find the official discussion forums for my wiimote projects helpful: **WiimoteProject.com**

### Low-Cost Multi-point Interactive Whiteboards Using the Wiimote

Since the Wiimote can track sources of infrared (IR) light, you can track pens that have an IR led in the tip. By pointing a wiimote at a projection screen or LCD display, you can create very low-cost interactive whiteboards or tablet displays. Since the Wiimote can track upto 4 points, up to 4 pens can be used. It also works great with rear-projected displays.

**Software**
The calibration and mouse cursor emulation software is available for you to download and try yourself. Note: My mouse emulation code isn't perfect. If any of you are programmers and can get it working with Alias Sketchbook, drop me a line.
**1.** Connect your wiimote to your PC via Bluetooth. There are a number of tutorials online on how to do this, possibly even for you specific software/hardware configuration. The Wiimote works with many (but not all) Bluetooth drivers. You can report/read about compatibility issues at **WiimoteProject.com**
**2.** Download the Wiimote Whiteboard software to the right. Please read the "READ ME.txt" file first! Make sure your wiimote is connected via Bluetooth, and then run the ".exe" in the main folder. NOTE: Good placement of the wiimote is key to good tracking. View the README for more info.

**Multitouch:** The multitouch demos are custom C# DirectX programs. You may download the sample program to the right, but this is provided for developers without support or documentation. The code is built on top of this Wiimote library. Unfortunately, multi-touch capable applications are currently extremely rare. Hopefully, that will change as more developers explore its potential.

**Building pens:** Here is a simple schematic of the light pen. The LEDs that I use are Vishay TSAL6400s running at 100mA, but lots of other LEDs will work too. You also might be able to jump start your experimentation by retro-fitting a mini keychain light with an IR LED. I'm currently looking into manufacturing and selling IR pens, but this may take several months.

**Mac/Linux Versions:** Due to personal time contraints, I probably won't be able to make a port myself. But fairly mature versions are available online. However, I haven't tried them myself. I've also created a Source Forge Project - Wiimote Whiteboard, but it does not seem to be getting much love.

A visit to this project's **FAQ and Advanced Discussion** post may be very englightening. You may also find the official discussion forums for my wiimote projects helpful: **WiimoteProject.com**

**Downloads:**
*Windows (32-bit)*
Wiimote Whiteboard* (original 12/19/07) - includes source
Wiimote Whiteboard v0.2* (updated 3/27/08) - includes source
Wiimote Whiteboard v0.3* (updated 8/20/08) - includes source

* there appear to be some issues with 64-bit machines, and BlueSoleil. Alternative Bluetooth software may work. Check the forum for more info.

*Mac (Java)*
You can do a quick webserach or try the version created by *uweschmidt* which is a fairly mature Java-based version for Mac.

**Multitouch:**source code

**Fun software (mostly free) to try with this:**
Mapping software: Microsoft Virtual Earth, Google Earth
Physics simulation software: Phun
Virtual Tourism: Photosynth
Media Browsing: CoolIris
Painting progams: Alias Sketchbook Pro
Note taking:One Note
Handwritting recognition: Windows Tablet PC Edition - works great if you have it, not a stand alone program to install.

## Head Tracking for Desktop VR Displays using the Wii Remote

Using the infrared camera in the Wii remote and a head mounted sensor bar (two IR LEDs), you can accurately track the location of your head and render view dependent images on the screen. This effectively transforms your display into a portal to a virtual environment. The display properly reacts to head and body movement as if it were a real window creating a realistic illusion of depth and space.

The program only needs to know your display size and the size of your sensor bar. The software is a custom C# DirectX program and is primarily provided as sample code for developers without support or additional documentation. You may need the most recent version of DirectX installed for this to work.

**Software**
To run the DesktopVR program you see in the video:
**1.** Connect your wiimote to your PC via Bluetooth. If you don't know how to do this, you can follow this tutorial. I've been told it works with other Bluetooth drivers, but I have not tested them myself.
**2.** Download the WiiDesktopVR (v02) sample program. Read the README file on program usage and configuration. Launch the "WiiDesktopVR.exe" in the main folder. A potentially more stable/Vista/64-bit compatible version has been created by Andrea Leganza. There also may be more variants on the web.

NOTE: If you are having trouble with running the program, you can check my project blog post about it or check the forum for assistance. I am unable to replicate these problems, so it hard for me to debug them. But, other people have figured it out. Things that have been identified to help: delete the "config.dat" file and re-run the program, install a new version of Direct X, or istall .NET 2.0.

**Developers Notes:** The code is built on top of this Wiimote library. To compile the program, you will need a C# IDE and the DirectX SDK. More notes are in the README.

A visit to this project's **FAQ and Advanced Discussion** post may be very englightening. You may also find the official discussion forums for my wiimote projects helpful: **WiimoteProject.com**

## Coming Later: 3D tracking, and more....

Unfortunately, time constraints in the next couple of months have significantly reduced my ability to work on more projects. But, you can subscribe to updates via my project blog [procrastineering.com]

# ANEXO 7: Referencia de la aplicación GlovePIE



Derecha = Wiimote.Right

**Botones**

A = Wiimote.A
B = Wiimote.B
- = Wiimote.Minus
+ = Wiimote.Plus
Home = Wiimote.Home
1 = Wiimote.One
2 = Wiimote.Two

Sabiendo esto podemos empezar a crear nuestros scripts básicos facilmente, para ello solamente habrá que seguir la siguiente nomenclatura

Tecla del teclado = Botón del Wiimote

Vamos a hacer un script sencillo para cualquier juego de carreras, habría que hacerlo así:

---

```
// Script de ejemplo de juego de carreras hecho por Scatsy


//Controles de dirección

Left = Wiimote.Up
Right = Wiimote.Down

//Controles de aceleración

Up = Wiimote.One
Down = Wiimote.Two
```

---

Con este script conseguiremos que, girando el wiimote dejando la cruceta a la izquierda, al apretar el boton 1 aceleraremos y con el 2 frenaremos, y pulsando arriba o abajo en el Wiimote (que girado sería izquierda y derecha) giraremos el coche.

Los comentarios se ponen con // delante, así el GlovePie no lo interpretará como una acción.

Con esta programación básica podremos tener controles para juegos sencillos, emuladores o controles multimedia, a continuación pongo un ejemplo algo más elaborado, un script para controlar el Media Player Classic

---

```
// Media Player Control by Scatsy V 0.1

// Controles de archivo

Ctrl+o = Wiimote.One // Abrir Archivo
Ctrl+l = Wiimote.Two // Cargar Subtitulos

// Controles de reproducción

Space = Wiimote.A // Play-Pause
Period = Wiimote.B // Stop
Ctrl+right = Wiimote.Right // Avanzar 5 segundos
Ctrl+Left = Wiimote.Left // Retroceder 5 segundos
Add = Wiimote.up // Retraso de audio +10 ms
Subtract = Wiimote.Down // Retraso de audio -10 ms

// Controles del reproductor

Ctrl+Enter = Wiimote.Home // Pantalla Completa
Alt+X = Wiimote.B + Wiimote.A // Salir del Media Player
Up = Wiimote.Plus // Subir Volumen
Down = Wiimote.Minus // Bajar Volumen
```

---

Este script hace uso de todos los botones del Wiimote e incluso de una funcion usando 2 botones a la vez, como se puede ver es muy sencillo.

## Programación media del Wiimote

En esta sección extenderé un poco más acciones con el Wiimote, más concretamente las acciones relacionadas con la inclinación del mando, lo primero que tenemos que ver son los valores de inclinación.

Wiimote.RawForceX
Wiimote.RawForceZ
Wiimote.RawForceY

Estos valores controlan la inclinación de los distintos ángulos del Wiimote y varian según se posicione el mismo, estos valores se programan de manera similar a los botones, solo que tenemos que tener en cuenta los valores. Para ver los valores de cada eje podemos usar el siguiente script

```
debug = "X=" + Wiimote.RawForceX + " Y=" + Wiimote.RawForceY + " Z=" + Wiimote.RawForceZ
```

Esto nos mostrará los valores de cada eje en la ventana de Debug del GlovePIE, y con esto podremos empezar a crear acciones para estos valores, por ejemplo, vamos a ver de nuevo un script para juegos de coches

---

```
// Script de ejemplo de juego de carreras hecho por Scatsy

Up = Wiimote.RawForceX <= -10 // Acelera

Down = Wiimote.RawForceX >= 20 // Frena

Left = Wiimote.RawforceZ <= -10 // Gira Izquierda

Right = Wiimote.RawforceZ >= 10 // Gira Derecha
```

---

Con este script tendremos el control del coche inclinando el mando hacia los lados y hacia delante y atras, por supuesto tendremos que coger el mando como en el script de carreras anterior.

En **Up = Wiimote.RawforceX <= -10** lo que he echo es programar el mando para que al inclinarlo en el eje X (es decir, al inclinarlo hacia adelante) igual o menos que -10, se presione la tecla **Up** del teclado, es decir la flecha de arriba, he elegido -10 por gustos personales pero se puede poner el valor que mejor te vaya, esto es lo mismo para el resto de valores.

También podemos programar los leds, es lo más sencillo, los valores para los leds son los siguientes:

Wiimote.Led1
Wiimote.Led2
Wiimote.Led3
Wiimote.Led4

Si queremos encender o apagar algúno de los leds, simplemente le añadimos **"= True"** o **"= False"**.

Por ejemplo, para encender los leds de los lados:

```
Wiimote.Led1 = True

Wiimote.Led2 = False

Wiimote.Led3 = False

Wiimote.Led4 = True
```

## Programación avanzada del Wiimote

Aqui vamos a profundizar en las variables y en las condicionales, es un tema muy complejo del que se podría llenar libros enteros, pero empezaré por lo mas básico, que son los **if**, **then**, **else** y **endif** luego explicaré un poco lo de las variables.

Estas 4 palabritas que he puesto definen casi todo en la programación avanzada del Wiimote, sirven para hacer condicionales, el GlovePIE usa un lenguaje de muy bajo nivel por lo tanto es bastante sencillo de comprender, pero por supuesto, con estas condicionales se pueden hacer cosas basante complejas, como siempre veamos un ejemplo.

---

```
//Ejemplo de vibración por Scatsy

if wiimote.b then // El Wiimote tiembla al usar el disparo primario

wiimote.rumble = true

wait 400 ms
```

```
wiimote.rumble = false
Endif

if pressed (wiimote.a) then // El Wiimote tiembla al usar el disparo secundario
wiimote.rumble = true
wait 300 ms
wiimote.rumble = false
Endif

If Pressed(e) Then // El Wiimote tiembla si se pulsa e
Wiimote.Rumble = True
wait 200 ms
Wiimote.Rumble = False
EndIf
```

---

Como veis, todas las condicionales empiezan por **if** y acaban en **endif**, en la primera condicional if wiimote.b then le estamos diciendo básicamente al wiimote que se prepare para algo cuando esté pulsado el botón **b** del wiimote, siempre tendremos que empezar así **if** acción **then** reacción, si lo traducís al español tiene lógica "**si** Wiimote.b **entonces**", a continuación tendremos que poner la condición, en el caso del ejemplo, le hemos dicho que el wiimote vibre "wiimote.rumble = true" cuando esté pulsado el boton B "if wiimote.b", luego le hemos dicho que espere 300 milisegundos y se apague "wait 300 ms | Wiimote.rumble = false", por último hemos puesto el **endif** que es lo que hace que la condición se acabe.

En la segunda condición, hay algo cambiado, el **pressed**, el sistema es el mismo, pero esto indica que en vez de vibrar mientras se pulse, vibrará durante 300 ms cuando sea pulsado, es decir, en la primera condición el Wiimote vibrará mientras tengamos pulsado el botón B del wiimote, mientras que en la segunda condición, el Wiimote vibrará durante 300 ms cuando pulsemos el boton A del Wiimote.

Por último, la tercera condición es exactamente la misma que la segunda, salvo que en vez de indicar a un botón del Wiimote, indicamos a un boton del teclado.

Hay otras formas de definir todo esto, usando la función **Else**, como siempre, un ejemplo

---

```
//Ejemplo de leds por Scatsy

if wiimote.b = true then
wiimote.led3 = true
wiimote.led2 = true
else
wiimote.led2 = false
wiimote.led3 = false
endif
```

---

Este ejemplo es muy sencillo y similar a los demas, en la primera condicional del ejemplo anterior, le habiamos dicho al GlovePIE que mientras estuviera pulsado el boton B del Wiimote, este vibrara, y que al soltarlo esperara 300 ms y dejara de vibrar, con este script hacemos lo mismo, mientras pulsemos el boton B del Wiimote, los leds 2 y 3 se encenderan y cuando lo soltemos se apagarán, si intentamos hacer esto de la misma forma que el ejemplo anteriór, lo único que conseguiremos es que los leds empiezen a parpadear, sin embargo podemos hacer lo mismo del ejemplo anterior con este modo, en resumen, la funcion **Else** hace que mientras esté pulsado ese boton o ejecutandose esa acción, se haga algo hasta que se suelte o acabe la acción.

## Variables

Las variables mucho mas sencillas de lo que parece, en realidad se programa igual que cualquier función, solo que tendremos que enunciarlas antes, veamos un ejemplo

---

```
//Ejemplo de variables por Scatsy

Var.A = Wiimote.A and Wiimote.B
Var.B = Wiimote.A and not Wiimote.B

if Var.A = true then

Wiimote.Led1 = True
```

```
Wiimote.Led4 = True
else
Wiimote.Led1 = False
Wiimote.Led4 = False
end

if Var.B = True Then

Wiimote.Led2 = True
Wiimote.Led3 = True
else
Wiimote.led2 = False
Eiimote.Led3 = False
endif
```

---

En el ejemplo, hemos enunciado primero las variables y lo que son, Var.A es que esten pulsados los botones A y B del Wiimote, mientras que Var.B es que esté pulsado el botón A del Wiimote pero no el B, entonces cuando las condiciones de estas variables se cumplan "Var.A = True", la acción que hemos hecho se ejecutará, esto viene bien por si queremos asignar 2 o mas botones a una determinada acción o que cuando ocurran 2 cosas a la vez se ejecute una determinada acción (por ejemplo cuando el Wiimote esté en horizontal y pulsemos el botón A del Wiimote).

También podemos encadenar todo tipo de variables y acciones, por ejemplo, tomando el mismo ejemplo de antes pero añadiendole una cosa mas

---

```
//Ejemplo de variables por Scatsy

Var.A = Wiimote.B
Var.B = Wiimote.A

if Var.A = true then

Wiimote.Led1 = True
Wiimote.Led4 = True
else
Wiimote.Led1 = False
Wiimote.Led4 = False
end

if Var.B = True Then

Wiimote.Led2 = True
Wiimote.Led3 = True
else
Wiimote.led2 = False
Eiimote.Led3 = False
endif

if Var.B and Var.A = true then
Wiimote.Rumble = true
Else
Wiimote.Rumble = False

endif
```

---

En este ejemplo que es muy similar al anterior, si pulsamos el botón B del Wiimote, se encenderán, los leds de los laterales, si pulsamos el botón A del Wiimote, se encenderán los leds del centro y ahora si pulsamos los dos botones a la vez, se encenderan todos los leds (por las variables A y B) y ademas el Wiimote Vibrará por que se han cumplido las variables A y B. Como podéis ver, las variables son muy comodas.

## Programación del Nunchuk

GlovePIE, aparte de soportar el Wiimote, también soporta el Nunchuk, es decir, el addon principal del Wiimote, programarlo es también muy sencillo, es casi igual que el Wiimote pero cambiando los nombres, vamos a ver sus botones y valores

**Botones**

Nunchuk.ZButton = Botón Z del Nunchuk
Nunchuk.CButton = Boton C del Nunchuk

**Joystick**

Wiimote.Nunchuk.JoyY = Eje Y del Joystick
Wiimote.Nunchuk.JoyX = Eje X del Joystick

**Inclinación**

Nunchuk.RawforceX = Inclinación del Eje X
Nunchuk.RawforceY = Inclinación del Eje Y
Nunchuk.RawforceZ = Inclinación del Eje Z

Salvo los botones, el resto de valores han de tratarse de forma analógica, al igual que haciamos con la inclinación del Wiimote, vamos a ver un ejemplo para tenerlo mas claro

---

```
// Ejemplo de movimiento en FPS creado por Scatsy

//Movimiento del personaje

W = Wiimote.Nunchuk.JoyY < -0.4 // Adelante

A = Wiimote.Nunchuk.JoyX < -0.4 // Paso Lateral izquierdo

S = Wiimote.Nunchuk.JoyY > 0.4 // Atras

D = Wiimote.Nunchuk.JoyX > 0.4 // Paso Lateral derecho

Ctrl = Wiimote.Nunchuk.ZButton // Agacharse

Space = Wiimote.Nunchuk.CButton // Saltar
```

---

Como veis es bastante sencillo, la única complicación es el movimiento en si, como ya he dicho, salvo los botones, el resto de valores tienen que ser analógicos, es decir, según empujemos el joystick del Nunchuk los valores cambiarán, estos van de **-0,93** a **1,07** aproximadamente, en el caso del eje Y **-0,93** sería empujando el joystick hacia alante del todo y **1,07** emujandolo hacia atras del todo, en el ejemlpo, cuando he puesto W = Wiimote.Nunchuk.JoyY < -0.4 // Adelante lo que he "dicho" al GlovePIE es que cuando el valor del eje Y sea menor de -0,4, pulse la tecla **W**, consiguiendo así que el personaje en nuestro FPS avance, esto, por supuesto, es personalizable y puedes poner el valor que quieras.

Programar los valores de inclinación es igual de sencillo que hacerlo con el Wiimote y se hace de la misma forma, solo cambia el nombre, veamos un ejemplo

---

```
// Ejemplo de recarga y uso en un FPS echo por Scatsy

R = Nunchuk.Rawforce >= 30 // Recarga E = Nunchuk.Rawforce <= -30 // Usa
```

---

Con esto, inclinando el Nunchuk hacia un lado o a otro recargaremos el arma en nuestro FPS o usaremos un objeto (siempre y cuando las teclas que tengais sean esas que son las que vienen normalmente en los FPS), inclinando el Nunchuk a la izquierda veremos los valores positivos, mientras que inclinandolo a la derecha veremos los negativos, esto, como todo, es personalizable para el gusto de cada uno.

Podeis ver los valores del Nunchuk con este Script

```
debug = "X=" + nunchuk.joyX + " Y=" + Nunchuk.joyY + " RZ=" + Nunchuk.RawForceZ + " RY=" + Nunchuk.RawForceY + " RX=" + Nunchuk.RawForceX
```

## Listado de funciones comunes

| Wiimote | | | | | Nunchuk | | |
|---|---|---|---|---|---|---|---|
| **Botones** | **Cruceta** | **Vibración** | **Leds** | **Sensores** | **Botones** | **Cruceta** | **Sensor** |
| Wiimote.A | Wiimote.Left | Wiimote.Rumble | Wiimote.Led1 | Wiimote.RawForceX | Nunchuk.C | Nuncuk.JoyY | Nunchuk.RawforceX |
| Wiimote.B | Wiimote.Right | | Wiimote.Led2 | Wiimote.RawForceY | Nunchuk.Z | Nuncuk.JoyX | Nunchuk.RawforceY |
| Wiimote.One | Wiimote.Up | | Wiimote.Led3 | Wiimote.RawForceZ | | | Nunchuk.RawforceZ |
| | | | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Wiimote.Two | Wiimote.Down | Wiimote.Led4 | Wiimote.RawAccX | | Nunchuk.RawAccX |
| Wiimote.Home | | | Wiimote.RawAccY | | Nunchuk.RawAccY |
| Wiimote.Plus | | | Wiimote.RawAccZ | | Nunchuk.RawAccZ |
| Wiimote.Minus | | | | | |

## Construcción de una WiiBar Casera

En realidad no hay una forma standard de crear una WiiBar casera, basta con tener un led en el centro superior del monitor o tv o 2 leds separados unos 20 CM, dependiendo de la distancia a la que vayas a usar el Wiimote de la barra.

Puedes ver diferentes barras caseras en este enlace de Elotrolado

## Scripts para usar el Wiimote en el PC

A continuación os dejo una lista de scripts para usar el Wiimote en el PC, estos están en formato .PIE que es el usado por GlovePIE

**F**
[1]Freets of Fire by Iron Tiger [Con Nunchuk]
**G**
[2]Google Hearth by J.Coulston
**H**
[3]Half Life 2 by Arianol and Scatsy [Con Nunchuk]
**M**
[4]Media Player Classic by Scatsy
[5]Mouse by Vkapadia
**T**
[6]Trackmania by Scatsy
**V**
[7]Virtua Tennis by Ethan [Con Nunchuk]
[8]VLC by Aepex
**W**
[9]WiiDrum by Lorenzo
[10]Winamp by Wylzon
[11]Wii Battery by J.Coulston
**Z**
[12]Zelda Ocarina Of Time by Carl Kenner and Bomboy02 [Con Nunchuk]

## Artículos relacionados

Joytokey

## Enlaces interesantes

Página del GlovePIE
Página del WiinRemote
Obtenido de "http://www.elotrolado.net/wiki/Glovepie"
Categorías: ElOtroLado:Artículos en desarrollo | Wii | PC
Esta página fue modificada por última vez el 21:51, 24 abr 2007.
Publicidad • Términos y condiciones • Staff
©1999-2009 Algunos derechos reservados

# ANEXO 8: Referencia de la adquisición de datos entre Nunchuck y Arduino



**Windmeadow Labs**
*Hacking, rants and cool projects*

Wii hacks | Atmel AVR microcontrollers | Catapults | Electronics | Hacking | Lasers | PhotoMosaic | Blog | About


**high dynamic piezos**
high / low voltage actuators ring type ,wide range of supplies
www.piezomechanik.com

**USB 3-Axis Accelerometer**
Low cost acceleration data logging USB enabled, Software API included.
www.MotionNode.com

**Serial Port Data Logger**
Standalone x86 computer, code in C Up to 10 serial ports, OEM $99+
www.tern.com

**Wii**
¡Vaya Oferta! La colección Erase ahora incluye una Wii de Nintendo.
www.planetadirecto.com/oferta_w

**I2C-PC Interface**
USB, RS232 to three I2C/SPI Busses BL233 Host IC's $3.95
www.i2cchip.com


**Navigation**

Contact Chad

**User login**

Username: *

Password: *

Log in

- Create new account
- Request new password

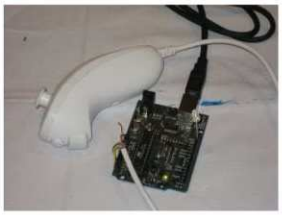
**Syndicate**

Home

## Read wii nunchuck data into arduino

Submitted by chad on Tue, 2007-06-05 04:59
Atmel AVR microcontrollers | Electronics | Hacking | Wii Hacks

With a little hacking, data can be read from a wii nunchuck directly into an Arduino, using TWI (aka I2C). The nunchuck contains a 3 axis accelerometer, joystick and buttons for only $19.95. The same accelerometer in kit form, cost $34.95 at **Sparkfun.** Plus the nunchuck is already wired up in a nice clean case! So the wii nunchuck should fit nicely into anyones robotic project.



The nunchuck uses a proprietary connector. I just cut the end off of my nunchuck cable. The cable has 4 wires.
white - ground
red - 3.3+v
green - data
yellow - clock

Attach white to the Arduino's ground, red to 5 volt+, green to analog pin 4, yellow to analog pin 5. The nunchuck is only supposed to get 3.3+ volts. So far it has worked fine at 5 volts, but be warned. I am guessing that using the higher voltage will shorten the nunchucks lifespan.

My Arduino has an Atmel atmega168 chip. The atmega168 can read the TWI protocol. The **"Wire" library** is bundled with the Arduino IDE and contains a number of libraries that will be used for reading the TWI protocol. A few small changes have to be made to the default wire library. The Arduino's twi.h header needs two changes. Look in lib/targets/libraries/Wire/utility. Then delete twi.o. Open up twi.h. Uncomment line:
// #define ATMEGA8.

Since the nunchuck uses "Fast" I2C, we will need to change the default speed:
#define TWI_FREQ 400000L.

For the Arduino to communicate with the nunchuck, it must send a handshake. So first send 2 bytes "0x40,0x00". Then send one byte "0x00" each time you request data from the nunchuck. The data from the nunchuck will come back in 6 byte chunks.

| Byte | Description | Values of sample Nunchuk |
|---|---|---|
| 1 | X-axis value of the analog stick | Min(Full Left):0x1E / Medium(Center):0x7E / Max(Full Right):0xE1 |
| 2 | Y-axis value of the analog stick | Min(Full Down):0x1D / Medium(Center):0x7B / Max(Full Right):0xDF |
| 3 | X-axis acceleration value | Min(at 1G):0x48 / Medium(at 1G):0x7D / Max(at 1G):0xB0 |
| 4 | Y-axis acceleration value | Min(at 1G):0x46 / Medium(at 1G):0x7A / Max(at 1G):0xAF |
| 5 | Z-axis acceleration value | Min(at 1G):0x4A / Medium(at 1G):0x7E / Max(at 1G):0xB1 |
| 6 | Button state (Bits 0/1) / acceleration LSB | Bit 0: "Z"-Button (0 = pressed, 1 = released) / Bit 1: "C" button (0 = pressed, 1 = released) / Bits 2-3: X acceleration LSB / Bits 4-5: Y acceleration LSB / Bits 6-7: Z acceleration LSB |

***see wiili.org***

Here is a simple program to read data from the nunchuck. It will read in data from the nunchuck then send it out over the Arduino's serial connection to your PC. On the PC, I just read it in with minicom, but you can use your favorite serial terminal program. The Arduino can't keep up with reading both serial and TWI at the same time. I have the Arduino read in about 50 bytes from the nunchuck then print that back to the PC over the serial connection. This means there will be a slight delay. Also you may see some strange characters every now and then. I think that is because of the serial connection interfering with the TWI connection.

**Update:** I updated the code some to slow down the serial data transmit and TWI requests. That seems to have fixed the bad data I was seeing.
**If you are able to reproduce this hack, please drop me an email at chad AT chadphillips . org**

```
#include <Wire.h>
#include <string.h>

#undef int
#include <stdio.h>

uint8_t outbuf[6];              // array to store arduino output
int cnt = 0;
int ledPin = 13;

void
setup ()
{
  beginSerial (19200);
```

**Search**
Search this site:
Search

**Older Stories**
- Arduino Semaphor
- Wii nunchuck data logger
- Using Microchip 25LC080 serial eeprom with an Arduino
- Review: Napoleon bar end mirror
- Motorcycle Airbag jackets/vests - Are they worth it?
- Review - Genuine Scooter Company Blur 150
- Using mod backha for a better test environment
- Convert lawnmowe to run off of propar camp stove
- Simple wood gasifi
- Wiimote controlled firefighting robot

More...

**Recent comments**
- Thanks a lot Bib hapi Biber 16 hours 45 min a
- Resistor 1 day 7 hours ago
- 5V vs 3.3V Calibration Change? 1 day 16 hours ag
- mm.. very good 1 day 18 hours ag
- TrueCrypt is free open-source 2 days 19 hours a
- amazing 2 days 20 hours a
- You mean the motion plus ? 3 days 7 hours ag
- Hey guys, thank for sharing 3 days 11 hours a
- Nunchuck and motion plus data interface to adruino 4 days 5 hours ag
- film indir film izl 4 days 12 hours a

**Notes**
notes
Missouri School MAP score

```
  Serial.print ("Finished setup\n");
  Wire.begin ();                  // join i2c bus with address 0x52
  nunchuck_init (); // send the initilization handshake
}

void
nunchuck_init ()
{
  Wire.beginTransmission (0x52);        // transmit to device 0x52
  Wire.send (0x40);             // sends memory address
  Wire.send (0x00);             // sends sent a zero.
  Wire.endTransmission ();      // stop transmitting
}

void
send_zero ()
{
  Wire.beginTransmission (0x52);        // transmit to device 0x52
  Wire.send (0x00);             // sends one byte
  Wire.endTransmission ();      // stop transmitting
}

void
loop ()
{
  Wire.requestFrom (0x52, 6);   // request data from nunchuck
  while (Wire.available ())
    {
      outbuf[cnt] = nunchuk_decode_byte (Wire.receive ());      // receive byte as an integer
      digitalWrite (ledPin, HIGH);      // sets the LED on
      cnt++;
    }

  // If we recieved the 6 bytes, then go print them
  if (cnt >= 5)
    {
      print ();
    }

  cnt = 0;
  send_zero (); // send the request for next bytes
  delay (100);
}

// Print the input data we have recieved
// accel data is 10 bits long
// so we read 8 bits, then we have to add
// on the last 2 bits.  That is why I
// multiply them by 2 * 2
void
print ()
{
  int joy_x_axis = outbuf[0];
  int joy_y_axis = outbuf[1];
  int accel_x_axis = outbuf[2] * 2 * 2;
  int accel_y_axis = outbuf[3] * 2 * 2;
  int accel_z_axis = outbuf[4] * 2 * 2;

  int z_button = 0;
  int c_button = 0;

 // byte outbuf[5] contains bits for z and c buttons
 // it also contains the least significant bits for the accelerometer data
 // so we have to check each bit of byte outbuf[5]
  if ((outbuf[5] >> 0) & 1)
    {
      z_button = 1;
    }
  if ((outbuf[5] >> 1) & 1)
    {
      c_button = 1;
    }

  if ((outbuf[5] >> 2) & 1)
    {
      accel_x_axis += 2;
    }
  if ((outbuf[5] >> 3) & 1)
    {
      accel_x_axis += 1;
    }

  if ((outbuf[5] >> 4) & 1)
    {
      accel_y_axis += 2;
    }
  if ((outbuf[5] >> 5) & 1)
    {
      accel_y_axis += 1;
    }

  if ((outbuf[5] >> 6) & 1)
    {
      accel_z_axis += 2;
    }
  if ((outbuf[5] >> 7) & 1)
    {
```

..

USB 3-Axis Accelerometer
Low cost acceleration data logging USB enabled, Software API included.
www.MotionNode.com

high dynamic piezos
high / low voltage actuators ring type ,wic range of supplies
www.piezomechanik.com

```
      accel_z_axis += 1;
    }

  Serial.print (joy_x_axis, DEC);
  Serial.print ("\t");

  Serial.print (joy_y_axis, DEC);
  Serial.print ("\t");

  Serial.print (accel_x_axis, DEC);
  Serial.print ("\t");

  Serial.print (accel_y_axis, DEC);
  Serial.print ("\t");

  Serial.print (accel_z_axis, DEC);
  Serial.print ("\t");

  Serial.print (z_button, DEC);
  Serial.print ("\t");

  Serial.print (c_button, DEC);
  Serial.print ("\t");

  Serial.print ("\r\n");
}

// Encode data to format that most wiimote drivers except
// only needed if you use one of the regular wiimote drivers
char
nunchuk_decode_byte (char x)
{
  x = (x ^ 0x17) + 0x17;
  return x;
}
```

**Update** I just tried running my code under Arduino 10. I had to make a different change to twi.h. Here are the first few lines of twi.h

```
#define ATMEGA8

#ifndef CPU_FREQ^M
#define CPU_FREQ 16000000L
#endif

#ifndef TWI_FREQ^M
#define TWI_FREQ 100000L
#endif
```

**Micro Controller Supplier**
Find & Contact Low-Cost Suppliers See why so many businesses trust us
V V Ads by Google

### 5V vs 3.3V Calibration Change?

Submitted by Leaning Tower (not verified) on Fri, 2009-07-17 15:39.

Hi, thanks for this great page!
I've got it working and I ran into something that might be useful to others. The x, y, z values from the accelerometer are not centered at 512, as you might assume. Mine are centered around 600. That is, if I hold the nunchuk level, the z reading indicating "0 G" is about 600, and the min/max readings for a steady position are around 340/850. If you assume that 512 is the "middle" of the 1024 range, it will seem like your readings are tilted a bit forward and to the right all of the time.

Consulting the chip's data sheet (ST LIS3L02AL), it looks like this could be caused by running it at 5V instead of 3.3V. It doesn't seem to do any harm, but you might get different readings at different supply voltages.

Hope that's helpful to someone!

» **reply**

### mm.. very good

Submitted by **ukraine** (not verified) on Fri, 2009-07-17 13:58.

mm.. very good

» **reply**

### qwerty

Submitted by **Anonymous** (not verified) on Thu, 2009-07-09 12:58.

dasfsdfsdf **qwerty** sdsdfsdfsdf

» **reply**

### Calibration

Submitted by BrandonToner (not verified) on Thu, 2009-06-25 05:29.

Does anyone know how to get the calibration data from the nunchuck?

» **reply**

### Nunchuck and motion plus data interface to adruino

Submitted by uphiearl (not verified) on Wed, 2009-07-15 02:15.

I got the program to work under 016 no problem. Now to modify it to read the Wii motion plus data. It is a 3 axis gyro. I have done it on a Parallax Propeller micro no problem.