Bachelor's thesis

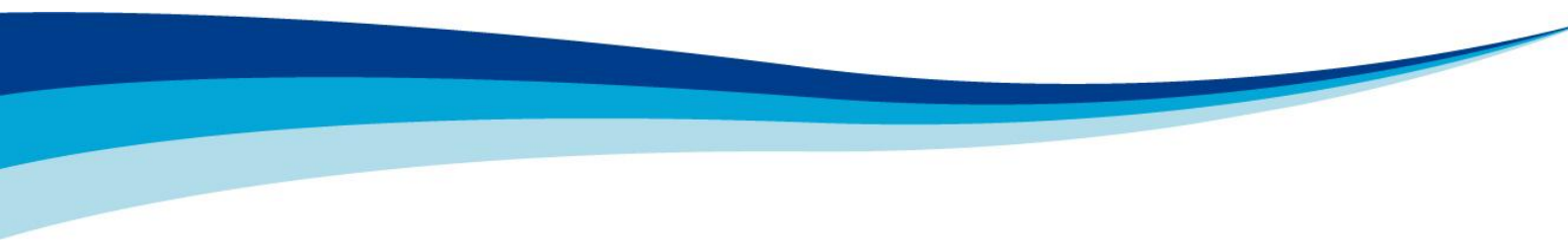
Information Technology

Embedded Software

2012

Debra Jules

VLC WIIMOTE CONTROL



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded software

Completion of the thesis| 34

Paalassalo Jari-Pekka

Debra Jules

VLC WIIMOTE CONTROL

The goal in this thesis project was to develop a new module/plugin for vlc media player capable to connect through a Bluetooth connection a wii remote or wiimote in order to control VLC media player and all the media played by it.

The vlc wiimote module was developed in C language in a Unix based operating system (Kubuntu). It can work only on Unix based OS.

It was created because it is important to help the vlc user to control VLC media player when he is far away from his keyboard and his mouse.

The main problems were to integrate and create correctly the module inside an existing software and be able to use all the functionalities of the wiimote. This is why the library Cwiid for Linux is used to handle all the functionalities of the wiimote.

The module with the wiimote is capable to play/pause a media with or without the accelerometer included in the wiimote, open the main menu and use all the functionalities of VLC media player, control and mute the volume, etc.

In conclusion, the module works properly and the user can completely control VLC media player with the wiimote.

KEYWORDS:

VLC Media Player, Wiimote, Bluetooth, Unix, Kubuntu, C language, Cwiid library, module, Software integration.

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ
TURUN AMMATTIKORKEAKOULU

Information Technology | Embedded software

Completion of the thesis| 34

Paalassalo Jari-Pekka

Debra Jules

VLC WIIMOTE CONTROL

The goal in this thesis project was to develop a new module/plugin for vlc media player capable to connect through a Bluetooth connection a wii remote or wiimote in order to control VLC media player and all the media played by it.

The vlc wiimote module was developed in C language in a Unix based operating system (Kubuntu). It can work only on Unix based OS.

It was created because it is important to help the vlc user to control VLC media player when he is far away from his keyboard and his mouse.

The main problems were to integrate and create correctly the module inside an existing software and be able to use all the functionalities of the wiimote. This is why the library Cwiid for Linux is used to handle all the functionalities of the wiimote.

The module with the wiimote is capable to play/pause a media with or without the accelerometer included in the wiimote, open the main menu and use all the functionalities of VLC media player, control and mute the volume, etc.

In conclusion, the module works properly and the user can completely control VLC media player with the wiimote.

ASIASANAT:

VLC Media Player, Wiimote, Bluetooth, Unix, Kubuntu, C language, Cwiid library, module, Software integration.

CONTENT

APPENDICES

LIST OF ABBREVIATIONS (OR) SYMBOLS

DVD :	(Digital Versatile Disc) Disc hardware format, using the UDF file system, an extension of the ISO 9660 file system format and a video format which is an extension of the MPEG-2 specification. [1]
H.264/MPEG-4 :	It is a standard for video compression, and is currently one of the most commonly used formats for the recording, compression, and distribution of high definition video. [1]
MP3 :	(MPEG Audio Layer III) is a patented encoding format for digital audio which uses a form of lossy data compression. [1]
MPEG-2 :	It is a standard for "the generic coding of moving pictures and associated audio information". It describes a combination of lossy video compression and lossy audio data compression methods which permit storage and transmission of movies using currently available storage media and transmission bandwidth. [1]
MKV :	The Matroska Multimedia Container is an open standard free container format, a file format that can hold an unlimited number of video, audio, picture or subtitle tracks in one file. [1]
WMV :	Windows Media Video is a video compression format for several proprietary codecs developed by Microsoft. [1]
Bluetooth:	Bluetooth is a proprietary open wireless technology standard for exchanging data over short distances (using short-wavelength radio transmissions in the ISM band from 2400–2480 MHz) from fixed and mobile devices. [1]
VLC:	VideoLAN Client.

[1] www.wikipedia.com

1 INTRODUCTION

The VLC media player is a software used to watch movies, listen to music or read any media. The problem is when you use VLC media player is that you must be in front of your computer to interact with it. The solution to this problem was to create a plugin to control the software with a remote control.

In this thesis I will present you what is exactly VLC media player, how it is constructed and what is happening when you run the program.

In a second part I will show you what is used to remotely control VLC media player, the definition of the remote and its functionalities.

In a last part I will explain how the VLC wiimote module is built and implemented, what is mandatory to have to use the module and what it needs to run properly the module and finally conclude.

2 VLC MEDIA PLAYER

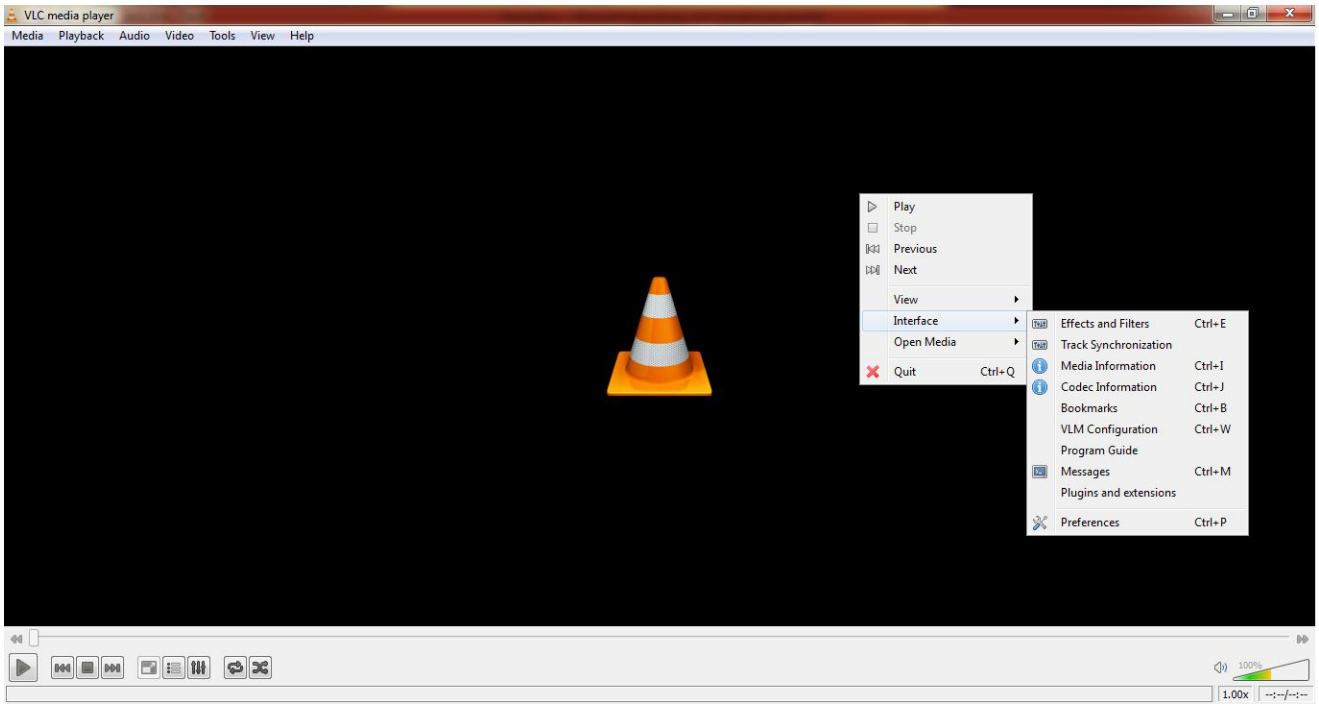
2.1 Definition.

VLC is a free and open source cross-platform multimedia player and framework that plays most multimedia files as well as DVD, Audio CD, VCD, and various streaming protocols.

It has a lot of features, that is why it can play everything: Files, Discs, Webcams, Devices and Streams.

It plays most codecs with no codec packs needed: MPEG-2, DivX, H.264, MKV, WebM, WMV, MP3 etc. and it runs on all platforms: Windows, Linux, Mac OS X, Unix etc.

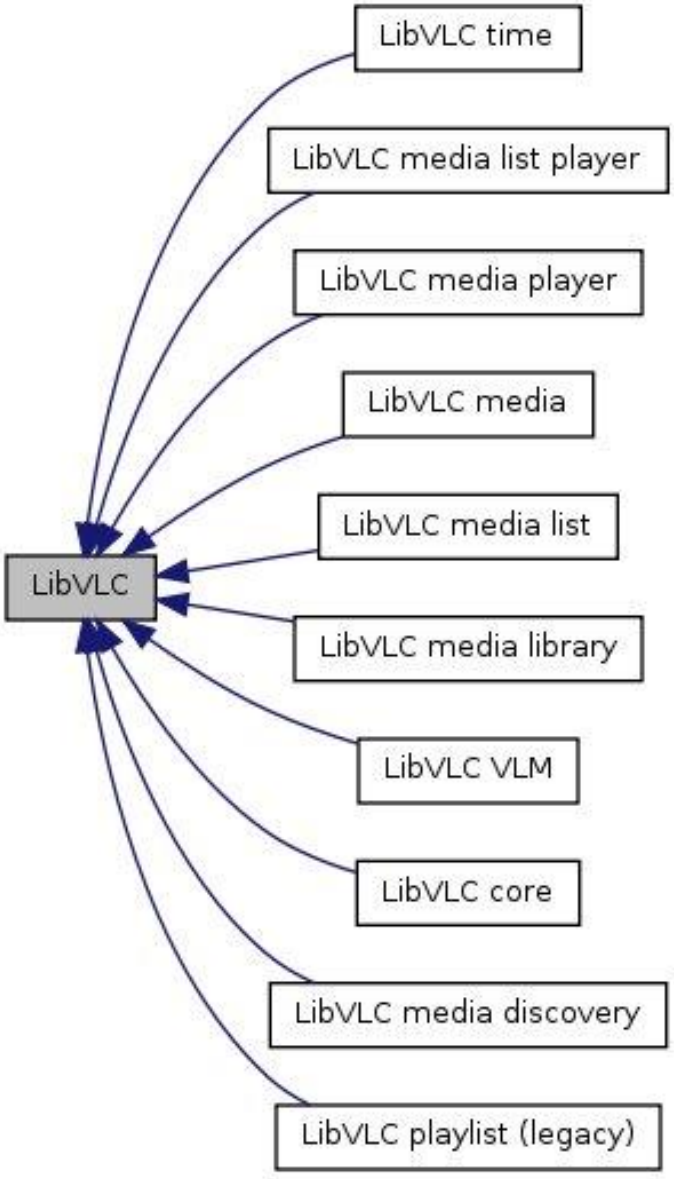
VLC is built in a modular way. This means that you can choose from a range of different modules to decide how to control VLC.



2.2 Structure.

As said previously, VLC is built in a modular way. LibVLC is the core part of VLC. This library provides an interface for programs such as VLC and a lot of other functionalities: a stream access, a video and audio output, a plugin handling, and a thread system.

Collaboration diagram for LibVLC:



All the LibVLC source files are located in the src/ directory and its subdirectories:

- *interface/:* contains code for user interaction such as key presses and device ejection.
- *playlist/:* manages playlist interaction such as stop, play, next, or random playback.
- *input/:* opens an input module, reads packets, parses them and passes reconstituted elementary streams to the decoder(s).
- *video_output/:* initializes the video display, gets all pictures and subpictures (subtitles) from the decoders.
- *audio_output/:* initializes the audio mixer.
- *misc/:* miscellaneous utilities used in other parts of libvlc, such as the thread system, the message queue, CPU detection or platform-specific code.

2.2.1 Modules

Modules are located in the *modules/* subdirectory and are loaded at runtime. Every module may offer different features. Plugin modules are loaded and unloaded dynamically by functions in *src/misc/modules.c* and *include/modules*.h.*

An interface module is made of 3 entry functions and a module description. The module description is made of macros that declare the capabilities of the modules with their priority, the module description will appear in the GUI.

Here is the module descriptor of the wiimote module:

```
vlc_module_begin ()
    set_shortname( N_("Wiimote") )
    set_description( N_("Wiimote control interface") )
    set_category( CAT_INTERFACE )
    set_subcategory( SUBCAT_INTERFACE_CONTROL )
    set_capability( "interface", 0 )
    set_callbacks( Open, Close )
vlc_module_end ()
```

Here is the 3 entry functions used in almost all module interface:

This function is called by VLC to initialize the interface module.
```
static int  Open    ( vlc_object_t * );
```

This function is called by VLC to uninitialize the module and close it.
```
static void Close   ( vlc_object_t * );
```

This function execute the job of the module. (main loop)
```
static void Run     ( intf_thread_t * );
```

Here is the list of all modules of VLC media Player:

- Playlist
- LibVLC
  - LibVLC playlist (legacy)
  - LibVLC core
    - LibVLC error handling
    - LibVLC asynchronous events
    - LibVLC logging
  - LibVLC time
  - LibVLC media
  - LibVLC media discovery
  - LibVLC media library
  - LibVLC media list
  - LibVLC media list player
  - LibVLC media player
    - LibVLC video controls
    - LibVLC audio controls
  - LibVLC VLM
- Access
- Decoder
- Encoder
- Demux
- out Es Out
- Input variables
- Interface
  - Log messages subscription
  - Interaction
- Memory
- Messages
- Objects
- OSD Menu
- Subpicture Unit
- Stream
- Strings
- Update
- Variables
  - Variable types
  - Additive flags
  - Variable actions
- Variable actions
- VLM
  - Video On Demand (VOD)
- Video Output
  - Video Subpictures

2.3 Launching VLC (version 2.0.1)

When you launch the VLC program the main thread becomes the interface thread. This thread passes throuh these following steps:

1. CPU detection.
2. Message interface initialization.
3. Command line options parsing.
4. Playlist creation.
5. Module bank initialization.
6. Interface opening.
7. Signal handler installation.
8. Audio output thread spawning.
9. Video output thread spawning.
10. Main loop and events management.

http://www.videolan.org/developers/vlc/doc/developers/html/manual.html 2001 Christophe Massiot

3 THE WIIMOTE CONTROL

3.1 Definition

The Wiimote, also known as the Wii remote, is the primary controller for Nintendo's Wii console. The main feature of the Wiimote is its motion sensing capability, which allows the user to interact via gesture recognition and pointing through the use of accelerometer and optical sensor technology.

3.2 Functionnalities

- The Wiimote has 12 buttons which are: a D-pad (directional pad or joypad), a “minus” button, a “plus” button, an “A” button and a “B” button, a “1” button and a “2” button and finally a “home” button.
- The connection between the Wiimote and the Nitendo Wii console is made by Bluetooth; the Wii Remote Bluetooth protocol can also be implemented on other devices such as a computer. The Wiimote has the ability to sense acceleration along three axes through the use of an ADXL330 accelerometer.

  > The ADXL330 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs, all on a single monolithic IC. The product measures acceleration with a minimum full-scale range of ±3 g. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.[2]

- The Wiimote provides basic audio and rumble functionality.
- The Wiimote contains a 16 KiB EEPROM.
- The Wiimote uses two AA batteries as a power source.

[2] http://www.analog.com/en/mems-sensors/mems-inertial-sensors/adxl330/products/product.html

4 THE VLC WIIMOTE MODULE

4.1 Definition

Written in C language, the VLC wiimote module was created to control the interface of VLC media player with a wiimote using bluetooth connexion. When the module is launched, it is able for example to: open a media, play/pause the media, control the volume, mute the sound, open the navigation menu, etc.

This module works only on Linux (Kubuntu 11.10) and needs a few other packages and libraries to run properly.

4.2 Packages and libraries

4.2.1 The Bluez package (version 4.100)

Bluez is the Bluetooth stack for Linux. Its goal is to make an implementation of the Bluetooth wireless standard specifications for Linux. It was initially developed by Qualcomm, and is available for Linux kernel versions 2.4.6 and up.

BlueZ provides support for the core Bluetooth layers and protocols. It is flexible, efficient and uses a modular implementation. It has many interesting features:

- Complete modular implementation
- Symmetric multi processing safe
- Multithreaded data processing
- Support for multiple Bluetooth device
- Real hardware abstraction
- Standard socket interface to all layers
- Device and service level security support

http://www.bluez.org/

4.2.2 The Cwiid library (version 0.6)

Cwiid is the name of the library which is used in the VLC wiimote module to interface the wiimote control. It needs some package to work: awk, bison, flex, gtk+-2 dev libs, python 2.4 or greater, python dev for python module etc.

4.3 The Wiimote module implementation.

The VLC wiimote module is located in *vlc\modules\control*. To be recognised, declared by VLC and well compiled it is mandatory to add these lines in *Modules.am* :

```
SOURCES_wiimote = wiimote.c

libvlc_LTLIBRARIES += \
            libdummy_plugin.la \
            libgestures_plugin.la \
            libnetsync_plugin.la \
            libhotkeys_plugin.la \
            libhello_plugin.la \
            libwiimote_plugin.la
```

Also these lines in *configure.ac* are mandatory to link the module correctly with the cwiid library and enable the wiimote module in VLC.

```
dnl
dnl  Wiimote plugin
dnl
AC_ARG_ENABLE(wiimote,
  [  --enable-wiimote           wiimote support (default disabled)])
if test "${enable_wiimote}" = "yes"
then
  AC_CHECK_HEADER(cwiid.h, AC_CHECK_LIB(cwiid, cwiid_open,
have_cwiid="true", have_cwiid="false"), have_cwiid="false")
  if test "${have_cwiid}" = "true"
  then
    VLC_ADD_PLUGIN([wiimote])
    VLC_ADD_LIBS([wiimote],[-lcwiid])
  fi
fi
```

The main problem that I encountered was to know how to use the VLC functions and how could I make them work correctly in my module. Because sometimes, it is not totally well commented or explained in the VLC libraries.

The problem in the function Wii_Callback(...) called by the Cwiid function cwiid_set_mesg_callback(...), itself called by the function Run(...) was to find a way to keep the modifications effective when the thread leaves the function Wii_Callback.

```
static void Run( intf_thread_t *p_intf )
{
.....
    /* Register a callback function */
    if ( cwiid_set_mesg_callback( p_sys->p_wiimote, &Wii_Callback ) )
    {...}
.....
}
```

Because the Wii_callback function is used to handle the wiimote signal, this function needs to modify a few things. For example when you push the button ”home”:

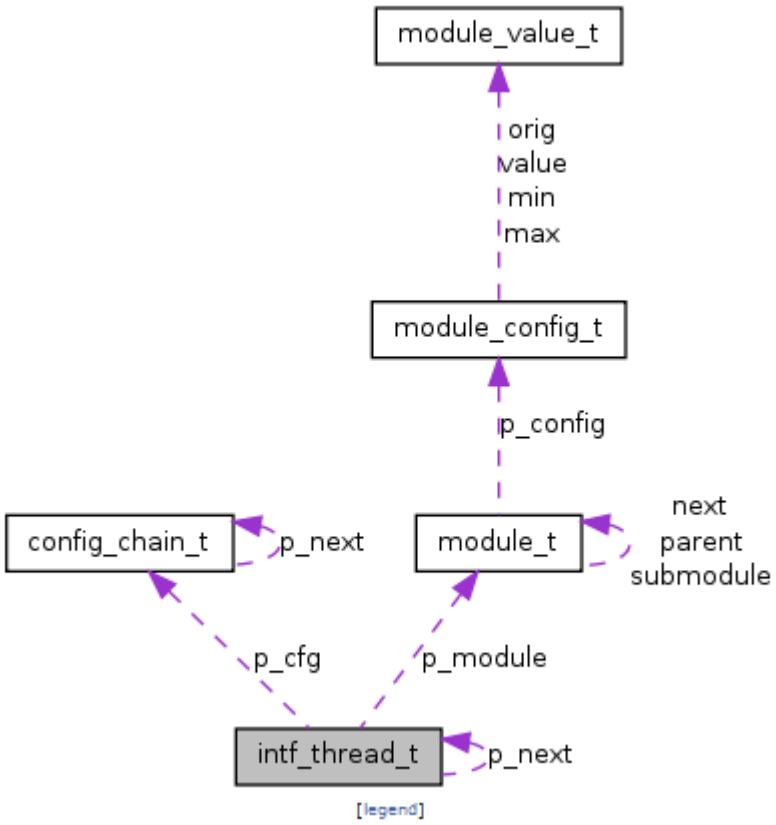
```
      case CWIID_BTN_HOME :
        if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == false
)
          var_SetBool( p_global_libvlc, "intf-popupmenu", true );
        else if ( var_GetBool( p_global_libvlc, "intf-popupmenu") ==
true )
          var_SetBool( p_global_libvlc, "intf-popupmenu", false );
        break;
```

The function will modify a value inside the structure p_global_libvlc. The only way to keep these modifications was to declare this structure in a global way.

```
          intf_thread_t *p_global_intf = NULL;
          struct libvlc_int_t *p_global_libvlc = NULL;
```

These two declarations handle almost every important parameters used to configure VLC media player and its media.

Collaboration diagram for inft_thread_t:



4.3.1 Xdotools (version 2.20110530)

This tool lets you simulate keyboard input and mouse activity.

Xdotool is used to navigate inside the VLC menu. Because this navigation is handled by the Operating system GUI itself, there is no specific function written inside VLC to navigate inside its own menus.

```
case CWIID_BTN_B :

        if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == true )
          system("xdotool key Tab");
```

4.3.2 Testing and results

When I started to implement the code I began to connect the wiimote to the computer. The wiimote was recognise by the computer's bluetooth adapter. To check everything I launched VLC media player in console mode with this line of command: ***./vlc -vvv --extrainfwiimote***

-vvv will show you all the debug message.

--extraintf <module> allows you to select extra interface modules that will be launched in addition to the main one.

I check everything with this command line and also with the function *intf_ErrMsg (char * psz_format, ...)* which print an "error" message to stderr.

Here the screen shot of the CLI which print that the wiimote module is launch and open correctly.

The message "Put the wiimote in discoverable mode (press 1+2) to connect it..." appear

```
[0x947b058] main interface debug: looking for interface module: 1 candidate
[0x947b058] wiimote interface: VLC_SUCCESS

[0x947b058] main interface debug: using interface module "wiimote"
[0x9483060] main playlist debug: playlist threads correctly activated
[0x9483060] main playlist debug: rebuilding array of current - root Playlist
[0x9483060] main playlist debug: rebuild done - 0 items, index -1
[0x94836a8] main interface debug: looking for interface module: 1 candidate
[0x947b058] wiimote interface: Put Wiimote in discoverable mode (press 1+2) to connect it...
```

After pressing these button (1+2) the Wiimote is found.

```
[0x8a72058] wiimote interface: Wiimote Found
```

If the buttons 1+2 are not pressed after a few second the CLI debug message ”No wiimotes found appear” and the module don’t stop to seek a wiimote device.

```
[0x8db3058] wiimote interface: Put Wiimote in discoverable mode (press 1+2) to connect it...

[0x8dbb6a8] main interface debug: using interface module "hotkeys"
[0x8dba198] main interface debug: looking for interface module: 0 candidates
[0x8dba198] main interface debug: no interface module matched "globalhotkeys,none"
[0x8dba198] main interface error: no suitable interface module
[0x8d17a58] main libvlc error: interface "globalhotkeys,none" initialization failed
[0x8db86f8] main interface debug: looking for interface module: 1 candidate
[0x8db86f8] dbus interface debug: listening on dbus as: org.mpris.MediaPlayer2.vlc.instance267
0
[0x8db86f8] main interface debug: using interface module "dbus"
[0x8db55d8] main interface debug: looking for interface module: 1 candidate
[0x8db55d8] main interface debug: using interface module "inhibit"
[0x8d17a58] main libvlc: Running vlc with the default interface. Use 'cvlc' to use vlc without
 interface.
[0x8dab210] main interface debug: looking for interface module: 3 candidates
[0x8fc5678] main generic debug: looking for extension module: 1 candidate
[0x8fc5678] lua generic debug: Opening Lua Extension module
[0x8fc5678] lua generic debug: Trying Lua scripts in /home/staross/.local/share/vlc/lua/extens
ions
[0x8fc5678] lua generic debug: Trying Lua scripts in /home/staross/vlc/src/.libs/vlc/lua/exten
sions
[0x8fc5678] lua generic debug: Trying Lua scripts in /home/staross/vlc/share/lua/extensions
[0x8fc5678] main generic debug: using extension module "lua"
[0x8dab210] main interface debug: using interface module "qt4"
No wiimotes found
No wiimotes found
No wiimotes found
No wiimotes found
```

And finally, here is when the module is closed when the user click on the red cross or press on the wiimote button ”2”.

```
[0x8a72058] main interface debug: removing module "wiimote"
[0x8a72058] wiimote interface: VLC_CLOSE_SUCCESS
```

5 CONCLUSION

During this thesis project I learned how to integrate and create a linux module to control VLC media player with a wiimote. The project was realised in C language, a language that I knew before. It was not hard to write the module; the hardest thing was to understand what to write and how exactly the VLC functions work and understand all the parameters handled by them.

The state of the project is now over, everything works properly, I succeeded to control VLC media player with a wiimote without problems.

In another hand, it is always possible to improve the module and create another twin module, for example which will work on Windows or any other operating system.

Finally I could say that I enjoyed working on this thesis project, which gave me more experience in embeeded software.

6 APPENDIX

Wiimote.c

```
/*****************************************************************************
* wiimote.c : wiimote module for vlc
*****************************************************************************
* Copyright (C) 2009 the VideoLAN team
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation; either version 2 of the License, or
* (at your option) any later version.
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston MA 02110-1301, USA.
*****************************************************************************/

/*****************************************************************************
* Preamble
*****************************************************************************/
/**
 * @file wiimote.c
 * @brief Wiimote interface VLC module
 */

#include <fcntl.h>

#ifdef HAVE_CONFIG_H
# include "config.h"
#endif

/* VLC core API headers */
#include <vlc_common.h>
#include <vlc_plugin.h>
#include <vlc_interface.h>

#include <vlc_input.h>
#include <vlc_vout.h>
#include <vlc_aout.h>
#include <vlc_aout_intf.h>
#include <vlc_osd.h>
#include <vlc_playlist.h>
#include <cwiid.h>
#include <vlc_keys.h>

#define BATTERY_STR_LEN 14
#define CHANNELS_NUMBER 4
#define VOLUME_TEXT_CHAN     p_global_intf->p_sys->p_channels[ 0 ]
#define VOLUME_WIDGET_CHAN   p_global_intf->p_sys->p_channels[ 1 ]

/* Forward declarations */
static int  Open    ( vlc_object_t * );
static void Close   ( vlc_object_t * );
```

```
/*****************************************************************************
* Module descriptor
*****************************************************************************/
vlc_module_begin ()
    set_shortname( N_("Wiimote") )
    set_description( N_("Wiimote control interface") )
    set_category( CAT_INTERFACE )
    set_subcategory( SUBCAT_INTERFACE_CONTROL )
    set_capability( "interface", 0 )
    set_callbacks( Open, Close )
vlc_module_end ()

/*****************************************************************************
* intf_sys_t: description and status of interface
*****************************************************************************/
struct intf_sys_t
{
    cwiid_wiimote_t* p_wiimote;        /* wiimote handle */
    struct cwiid_state state;          /* wiimote state */
    bdaddr_t bdaddr;                   /* bluetooth device address */
    int p_channels[ CHANNELS_NUMBER ]; /* contains registered channel IDs */
    uint16_t status;                   /* Remote number (1,2,3,4) */
    vout_thread_t      *p_last_vout;
    struct acc_cal wm_cal;
};

intf_thread_t *p_global_intf = NULL;
struct libvlc_int_t *p_global_libvlc = NULL;
vlc_value_t oldval;
vlc_value_t newval;
struct acc_cal wm_cal;

/*****************************************************************************
* Local prototypes
*****************************************************************************/
static void Run( intf_thread_t * );

void Wii_Callback(cwiid_wiimote_t * wiimote,  /* The Wiimote struct */
        int msg_count,                        /* The number of messages received
*/
        union cwiid_mesg msgs[],              /* The messages themselves */
        struct timespec * timestamp           /* The timestamp of the callback
*/
        );

static void DisplayRate ( vout_thread_t *, float );
static void DisplayVolume  ( intf_thread_t *, vout_thread_t *, audio_volume_t );
static void ClearChannels  ( intf_thread_t *, vout_thread_t * );

#define DisplayMessage(vout, ch, fmt, ...) \
    do { if(vout) vout_OSDMessage(vout, ch, fmt, __VA_ARGS__); } while(0)
#define DisplayIcon(vout, icon) \
    do { if(vout) vout_OSDIcon(vout, SPU_DEFAULT_CHANNEL, icon); } while(0)
```

```
/*****************************************************************************
* Open: initialize interface
*****************************************************************************/
static int Open( vlc_object_t *obj )
{
    intf_thread_t *p_intf = ( intf_thread_t * )obj;
    intf_sys_t *p_sys;

    p_intf->p_sys = p_sys = malloc( sizeof( intf_sys_t ) );
    if( p_sys == NULL )
    {
    msg_Info( p_intf, "VLC_ENOMEM\n" );
    return VLC_ENOMEM;
    }

    /*accept connections from any remotes */
    p_sys->bdaddr = *BDADDR_ANY;

    p_intf->pf_run = Run;

    msg_Info( p_intf, "VLC_SUCCESS\n" );
    return VLC_SUCCESS;
}

/*****************************************************************************
* Close: destroy interface
*****************************************************************************/
static void Close( vlc_object_t *obj )
{
    intf_thread_t *p_intf = ( intf_thread_t * )obj;
    intf_sys_t *p_sys = p_intf->p_sys;

    cwiid_close( p_sys->p_wiimote );

    msg_Info( p_intf, "VLC_CLOSE_SUCCESS\n" );
    free( p_sys );
}
```

```
/*****************************************************************************
* Run: main loop
*****************************************************************************/
static void Run( intf_thread_t *p_intf )
{
    p_global_intf = p_intf;
    p_global_libvlc = p_intf->p_libvlc;
    playlist_t *p_playlist = pl_Get( p_global_intf );

    intf_sys_t *p_sys = p_intf->p_sys;
    wm_cal = p_sys->wm_cal;
    p_sys->status = 0;
    p_sys->p_last_vout = NULL;

    /* Set the Default Volume */
    aout_VolumeSet( p_playlist, AOUT_VOLUME_DEFAULT );

    /* Connect to the wiimote */
    msg_Info( p_intf, "Put Wiimote in discoverable mode (press 1+2) to connect
it...\n" );
    while( ! ( p_sys->p_wiimote = cwiid_open(&p_sys->bdaddr, 0 ) ) );

    /* Register a callback function */
    if ( cwiid_set_mesg_callback( p_sys->p_wiimote, &Wii_Callback ) )
    {
    msg_Err( p_intf, "Unable to set message callback" );
    return;
    }

    if( cwiid_enable( p_sys->p_wiimote, CWIID_FLAG_MESG_IFC ) )
    {
    msg_Err( p_intf, "Error enabling messages\n" );
    return;
    }

    if (cwiid_get_acc_cal( p_sys->p_wiimote, CWIID_EXT_NONE, &wm_cal ) )
    {
    msg_Err(p_intf, "Error calibration");
    return;
    }

    p_sys->status++;

    /* Set the reporting mode. This says which messages we want */
    cwiid_set_rpt_mode( p_sys->p_wiimote, CWIID_RPT_BTN | CWIID_RPT_ACC );

    /* Vibrate the remote for 1 sec */
    cwiid_set_rumble( p_sys->p_wiimote, 1 );
    sleep(1);
    cwiid_set_rumble( p_sys->p_wiimote, 0 );

    msg_Info(p_intf, "Wiimote Found\n");

    /* Turn on LED */
    cwiid_set_led( p_sys->p_wiimote, p_sys->status );
    return;
}
```

```
/*****************************************************************************
* Wii_Callback: wiimote callback function
*****************************************************************************/
int k = 0;
void Wii_Callback(cwiid_wiimote_t * wiimote,    /* The Wiimote struct */
        int msg_count,                          /* The number of messages
received(2)*/
        union cwiid_mesg msgs[],                /* The messages themselves */
        struct timespec * timestamp             /* The timestamp of the callback
*/
        )
{

    int i, z, y, x;
    audio_volume_t i_newvol;

    intf_sys_t *p_sys = p_global_intf->p_sys;

    playlist_t *p_playlist = pl_Get( p_global_intf );

    /* Update the input */
    input_thread_t *p_input = playlist_CurrentInput( p_playlist );

    /* Update the vout */
    vout_thread_t *p_vout = p_input ? input_GetVout( p_input ) : NULL;

    /* Register OSD channels */
    if( p_vout && p_vout != p_sys->p_last_vout )
    for( unsigned i = 0; i < CHANNELS_NUMBER; i++ )
        p_global_intf->p_sys->p_channels[i] = vout_RegisterSubpictureChannel(
p_vout );

    p_sys->p_last_vout = p_vout;

for( i = 0; i < msg_count; i++ )
{
    if ( msgs[i].type == CWIID_MESG_BTN )
    {
      switch ( msgs[i].btn_mesg.buttons )
      {
      case CWIID_BTN_A :

        if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == true )
          system("xdotool key Return");
        else if( p_input && var_GetBool( p_global_libvlc, "intf-popupmenu") ==
false)
        {
        ClearChannels( p_global_intf, p_vout );
        int state = var_GetInteger( p_input, "state" );
        DisplayIcon( p_vout, state != PAUSE_S ? OSD_PAUSE_ICON : OSD_PLAY_ICON
);
        playlist_Pause( p_playlist );
        }
        else
        playlist_Play( p_playlist );
        break;
      case CWIID_BTN_B :
```

```
        if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == true )
          system("xdotool key Tab");
        break;
      case CWIID_BTN_MINUS :
        aout_VolumeDown( p_playlist, 1, &i_newvol );
        DisplayVolume( p_global_intf, p_vout, i_newvol );
        break;
      case CWIID_BTN_PLUS :

        aout_VolumeUp( p_playlist, 1, &i_newvol );
        DisplayVolume( p_global_intf, p_vout, i_newvol );
        break;
      case CWIID_BTN_HOME :
        if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == false )
          var_SetBool( p_global_libvlc, "intf-popupmenu", true );
        else if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == true )
          var_SetBool( p_global_libvlc, "intf-popupmenu", false );
        break;
      case CWIID_BTN_LEFT :

        if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == true )
          system("xdotool key Left");
        else if ( p_input && var_GetBool( p_global_libvlc, "intf-popupmenu") ==
false)
        {
        var_TriggerCallback( p_playlist, "rate-slower" );
        DisplayRate( p_vout, var_GetFloat( p_input, "rate" ) );
        }
        break;
      case CWIID_BTN_RIGHT :

        if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == true )
          system("xdotool key Right");

        else if ( p_input && var_GetBool( p_global_libvlc, "intf-popupmenu") ==
false)
        {
        var_TriggerCallback( p_playlist, "rate-faster" );
        DisplayRate( p_vout, var_GetFloat( p_input, "rate" ) );
        }
        break;
      case CWIID_BTN_DOWN :

        if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == true )
          system("xdotool key Down");

        else
        {
          DisplayMessage( p_vout, SPU_DEFAULT_CHANNEL, "%s",
                _("Next") );
          playlist_Next( p_playlist );
        }
        break;
      case CWIID_BTN_UP :

        if ( var_GetBool( p_global_libvlc, "intf-popupmenu") == true )
          system("xdotool key Up");

        else
```

```
        {
          DisplayMessage( p_vout, SPU_DEFAULT_CHANNEL, "%s",
                _("Previous") );
          playlist_Prev( p_playlist );
        }
        break;
      case CWIID_BTN_1 :

        i_newvol = -1;
        aout_ToggleMute( p_playlist, &i_newvol );

        if( p_vout )
        {
        if( i_newvol == 0 )
        {
            ClearChannels( p_global_intf, p_vout );
            DisplayMessage( p_vout, SPU_DEFAULT_CHANNEL, "%s",
                _("Mute") );
        }
        else
            DisplayVolume( p_global_intf, p_vout, i_newvol );
        }
        break;
      case CWIID_BTN_2 :
        cwiid_close(wiimote);
        exit(0);
        break;
      }
    }
    else if ( msgs[i].type == CWIID_MESG_ACC )
    {
  /*x = ( (double)msgs[i].acc_mesg.acc[ CWIID_X ] );
    y = ( (double)msgs[i].acc_mesg.acc[ CWIID_Y ] );
    z = ( (double)msgs[i].acc_mesg.acc[ CWIID_Z ] );*/

    x = ((double)msgs[i].acc_mesg.acc[CWIID_X] - wm_cal.zero[CWIID_X]) /
                  (wm_cal.one[CWIID_X] - wm_cal.zero[CWIID_X]);
            y = ((double)msgs[i].acc_mesg.acc[CWIID_Y] - wm_cal.zero[CWIID_Y]) /
                  (wm_cal.one[CWIID_Y] - wm_cal.zero[CWIID_Y]);
            z = ((double)msgs[i].acc_mesg.acc[CWIID_Z] - wm_cal.zero[CWIID_Z]) /
                  (wm_cal.one[CWIID_Z] - wm_cal.zero[CWIID_Z]);

    if ( z < -2 && p_input && k == 0)
    {
        ClearChannels( p_global_intf, p_vout );
        int state = var_GetInteger( p_input, "state" );
        DisplayIcon( p_vout, state != PAUSE_S ? OSD_PAUSE_ICON : OSD_PLAY_ICON
);
        playlist_Pause( p_playlist );
        k = 1;
        /*system("xdotool key Down");*/
    }
    else if ( p_input && k == 1 && z < -2 )
    {
        ClearChannels( p_global_intf, p_vout );
        int state = var_GetInteger( p_input, "state" );
        DisplayIcon( p_vout, state != PAUSE_S ? OSD_PAUSE_ICON : OSD_PLAY_ICON
);
        playlist_Play( p_playlist );
```

```
        k=0;
    }
  /*else if (49 < x && x < 50 )
      system("xdotool key Right");
    else if ( 149 < x && x < 150 )
      system("xdotool key Left" );
    printf("Acc Report: x=%d, y=%d, z=%d\n",x,y,z);*/
    }

}
}

static void DisplayVolume( intf_thread_t *p_intf, vout_thread_t *p_vout,
              audio_volume_t i_vol )
{
    if( p_vout == NULL )
    {
    return;
    }
    ClearChannels( p_intf, p_vout );

    if( var_GetBool( p_vout, "fullscreen" ) )
    {
    vout_OSDSlider( p_vout, VOLUME_WIDGET_CHAN,
        i_vol*100/AOUT_VOLUME_MAX, OSD_VERT_SLIDER );
    }
    DisplayMessage( p_vout, VOLUME_TEXT_CHAN, _( "Volume %d%%" ),
            i_vol*100/AOUT_VOLUME_DEFAULT );
}

static void DisplayRate( vout_thread_t *p_vout, float f_rate )
{
    DisplayMessage( p_vout, SPU_DEFAULT_CHANNEL, _("Speed: %.2fx"), f_rate );
}

static void ClearChannels( intf_thread_t *p_intf, vout_thread_t *p_vout )
{
    if( p_vout )
    {
    vout_FlushSubpictureChannel( p_vout, SPU_DEFAULT_CHANNEL );
    for( int i = 0; i < CHANNELS_NUMBER; i++ )
        vout_FlushSubpictureChannel( p_vout, p_intf->p_sys->p_channels[i]  );
    }
}
```

Cwiid.h

```
/* Copyright (C) 2007 L. Donnie Smith <cwiid@abstrakraft.org>
 *
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301
USA
*/

#ifndef CWIID_H
#define CWIID_H

#include <stdarg.h>
#include <stdint.h>
#include <time.h>
#include <bluetooth/bluetooth.h>    /* bdaddr_t */

/* Flags */
#define CWIID_FLAG_MESG_IFC     0x01
#define CWIID_FLAG_CONTINUOUS   0x02
#define CWIID_FLAG_REPEAT_BTN   0x04
#define CWIID_FLAG_NONBLOCK     0x08
#define CWIID_FLAG_MOTIONPLUS   0x10

/* Report Mode Flags */
#define CWIID_RPT_STATUS        0x01
#define CWIID_RPT_BTN           0x02
#define CWIID_RPT_ACC           0x04
#define CWIID_RPT_IR            0x08
#define CWIID_RPT_NUNCHUK       0x10
#define CWIID_RPT_CLASSIC       0x20
#define CWIID_RPT_BALANCE       0x40
#define CWIID_RPT_MOTIONPLUS    0x80
#define CWIID_RPT_EXT       (CWIID_RPT_NUNCHUK | CWIID_RPT_CLASSIC | \
                             CWIID_RPT_BALANCE | CWIID_RPT_MOTIONPLUS)

/* LED flags */
#define CWIID_LED1_ON   0x01
#define CWIID_LED2_ON   0x02
#define CWIID_LED3_ON   0x04
#define CWIID_LED4_ON   0x08

/* Button flags */
#define CWIID_BTN_2     0x0001
#define CWIID_BTN_1     0x0002
#define CWIID_BTN_B     0x0004
#define CWIID_BTN_A     0x0008
#define CWIID_BTN_MINUS 0x0010
```

```
#define CWIID_BTN_HOME  0x0080
#define CWIID_BTN_LEFT  0x0100
#define CWIID_BTN_RIGHT 0x0200
#define CWIID_BTN_DOWN  0x0400
#define CWIID_BTN_UP    0x0800
#define CWIID_BTN_PLUS  0x1000

#define CWIID_NUNCHUK_BTN_Z 0x01
#define CWIID_NUNCHUK_BTN_C 0x02

#define CWIID_CLASSIC_BTN_UP    0x0001
#define CWIID_CLASSIC_BTN_LEFT  0x0002
#define CWIID_CLASSIC_BTN_ZR    0x0004
#define CWIID_CLASSIC_BTN_X     0x0008
#define CWIID_CLASSIC_BTN_A     0x0010
#define CWIID_CLASSIC_BTN_Y     0x0020
#define CWIID_CLASSIC_BTN_B     0x0040
#define CWIID_CLASSIC_BTN_ZL    0x0080
#define CWIID_CLASSIC_BTN_R     0x0200
#define CWIID_CLASSIC_BTN_PLUS  0x0400
#define CWIID_CLASSIC_BTN_HOME  0x0800
#define CWIID_CLASSIC_BTN_MINUS 0x1000
#define CWIID_CLASSIC_BTN_L     0x2000
#define CWIID_CLASSIC_BTN_DOWN  0x4000
#define CWIID_CLASSIC_BTN_RIGHT 0x8000

/* Send Report flags */
#define CWIID_SEND_RPT_NO_RUMBLE    0x01

/* Data Read/Write flags */
#define CWIID_RW_EEPROM 0x00
#define CWIID_RW_REG    0x04
#define CWIID_RW_DECODE 0x00

/* Maximum Data Read Length */
#define CWIID_MAX_READ_LEN  0xFFFF

/* Array Index Defs */
#define CWIID_X     0
#define CWIID_Y     1
#define CWIID_Z     2
#define CWIID_PHI   0
#define CWIID_THETA 1
#define CWIID_PSI   2

/* Acc Defs */
#define CWIID_ACC_MAX   0xFF

/* IR Defs */
#define CWIID_IR_SRC_COUNT  4
#define CWIID_IR_X_MAX      1024
#define CWIID_IR_Y_MAX      768

/* Battery */
#define CWIID_BATTERY_MAX   0xD0

/* Classic Controller Maxes */
#define CWIID_CLASSIC_L_STICK_MAX   0x3F
#define CWIID_CLASSIC_R_STICK_MAX   0x1F
```

```
#define CWIID_CLASSIC_LR_MAX    0x1F

/* Environment Variables */
#define WIIMOTE_BDADDR  "WIIMOTE_BDADDR"

/* Callback Maximum Message Count */
#define CWIID_MAX_MESG_COUNT    5

/* Enumerations */
enum cwiid_command {
    CWIID_CMD_STATUS,
    CWIID_CMD_LED,
    CWIID_CMD_RUMBLE,
    CWIID_CMD_RPT_MODE
};

enum cwiid_mesg_type {
    CWIID_MESG_STATUS,
    CWIID_MESG_BTN,
    CWIID_MESG_ACC,
    CWIID_MESG_IR,
    CWIID_MESG_NUNCHUK,
    CWIID_MESG_CLASSIC,
    CWIID_MESG_BALANCE,
    CWIID_MESG_MOTIONPLUS,
    CWIID_MESG_ERROR,
    CWIID_MESG_UNKNOWN
};

enum cwiid_ext_type {
    CWIID_EXT_NONE,
    CWIID_EXT_NUNCHUK,
    CWIID_EXT_CLASSIC,
    CWIID_EXT_BALANCE,
    CWIID_EXT_MOTIONPLUS,
    CWIID_EXT_UNKNOWN
};

enum cwiid_error {
    CWIID_ERROR_NONE,
    CWIID_ERROR_DISCONNECT,
    CWIID_ERROR_COMM
};

struct acc_cal {
    uint8_t zero[3];
    uint8_t one[3];
};

struct balance_cal {
    uint16_t right_top[3];
    uint16_t right_bottom[3];
    uint16_t left_top[3];
    uint16_t left_bottom[3];
};

/* Message Structs */
struct cwiid_status_mesg {
    enum cwiid_mesg_type type;
```

```
    uint8_t battery;
    enum cwiid_ext_type ext_type;
};

struct cwiid_btn_mesg {
    enum cwiid_mesg_type type;
    uint16_t buttons;
};

struct cwiid_acc_mesg {
    enum cwiid_mesg_type type;
    uint8_t acc[3];
};

struct cwiid_ir_src {
    char valid;
    uint16_t pos[2];
    int8_t size;
};

struct cwiid_ir_mesg {
    enum cwiid_mesg_type type;
    struct cwiid_ir_src src[CWIID_IR_SRC_COUNT];
};

struct cwiid_nunchuk_mesg {
    enum cwiid_mesg_type type;
    uint8_t stick[2];
    uint8_t acc[3];
    uint8_t buttons;
};

struct cwiid_classic_mesg {
    enum cwiid_mesg_type type;
    uint8_t l_stick[2];
    uint8_t r_stick[2];
    uint8_t l;
    uint8_t r;
    uint16_t buttons;
};

struct cwiid_balance_mesg {
    enum cwiid_mesg_type type;
    uint16_t right_top;
    uint16_t right_bottom;
    uint16_t left_top;
    uint16_t left_bottom;
};

struct cwiid_motionplus_mesg {
    enum cwiid_mesg_type type;
    uint16_t angle_rate[3];
};

struct cwiid_error_mesg {
    enum cwiid_mesg_type type;
    enum cwiid_error error;
};
```

```
union cwiid_mesg {
    enum cwiid_mesg_type type;
    struct cwiid_status_mesg status_mesg;
    struct cwiid_btn_mesg btn_mesg;
    struct cwiid_acc_mesg acc_mesg;
    struct cwiid_ir_mesg ir_mesg;
    struct cwiid_nunchuk_mesg nunchuk_mesg;
    struct cwiid_classic_mesg classic_mesg;
    struct cwiid_balance_mesg balance_mesg;
    struct cwiid_motionplus_mesg motionplus_mesg;
    struct cwiid_error_mesg error_mesg;
};

/* State Structs */
struct nunchuk_state {
    uint8_t stick[2];
    uint8_t acc[3];
    uint8_t buttons;
};

struct classic_state {
    uint8_t l_stick[2];
    uint8_t r_stick[2];
    uint8_t l;
    uint8_t r;
    uint16_t buttons;
};

struct balance_state {
    uint16_t right_top;
    uint16_t right_bottom;
    uint16_t left_top;
    uint16_t left_bottom;
};

struct motionplus_state {
    uint16_t angle_rate[3];
};

union ext_state {
    struct nunchuk_state nunchuk;
    struct classic_state classic;
    struct balance_state balance;
    struct motionplus_state motionplus;
};

struct cwiid_state {
    uint8_t rpt_mode;
    uint8_t led;
    uint8_t rumble;
    uint8_t battery;
    uint16_t buttons;
    uint8_t acc[3];
    struct cwiid_ir_src ir_src[CWIID_IR_SRC_COUNT];
    enum cwiid_ext_type ext_type;
    union ext_state ext;
    enum cwiid_error error;
};
```

```
/* Typedefs */
typedef struct wiimote cwiid_wiimote_t;

typedef void cwiid_mesg_callback_t(cwiid_wiimote_t *, int,
                                   union cwiid_mesg [], struct timespec *);
typedef void cwiid_err_t(cwiid_wiimote_t *, const char *, va_list ap);

/* get_bdinfo */
#define BT_NO_WIIMOTE_FILTER 0x01
#define BT_NAME_LEN 32

struct cwiid_bdinfo {
    bdaddr_t bdaddr;
    uint8_t btclass[3];
    char name[BT_NAME_LEN];
};

#ifdef __cplusplus
extern "C" {
#endif

/* Error reporting (library wide) */
int cwiid_set_err(cwiid_err_t *err);
void cwiid_err_default(struct wiimote *wiimote, const char *str, va_list ap);

/* Connection */
#define cwiid_connect cwiid_open
#define cwiid_disconnect cwiid_close
cwiid_wiimote_t *cwiid_open(bdaddr_t *bdaddr, int flags);
cwiid_wiimote_t *cwiid_open_timeout(bdaddr_t *bdaddr, int flags, int timeout);
int cwiid_close(cwiid_wiimote_t *wiimote);

int cwiid_get_id(cwiid_wiimote_t *wiimote);
int cwiid_set_data(cwiid_wiimote_t *wiimote, const void *data);
const void *cwiid_get_data(cwiid_wiimote_t *wiimote);
int cwiid_enable(cwiid_wiimote_t *wiimote, int flags);
int cwiid_disable(cwiid_wiimote_t *wiimote, int flags);

/* Interfaces */
int cwiid_set_mesg_callback(cwiid_wiimote_t *wiimote,
                            cwiid_mesg_callback_t *callback);
int cwiid_get_mesg(cwiid_wiimote_t *wiimote, int *mesg_count,
                   union cwiid_mesg *mesg[], struct timespec *timestamp);
int cwiid_get_state(cwiid_wiimote_t *wiimote, struct cwiid_state *state);
int cwiid_get_acc_cal(struct wiimote *wiimote, enum cwiid_ext_type ext_type,
                      struct acc_cal *acc_cal);
int cwiid_get_balance_cal(struct wiimote *wiimote,
                          struct balance_cal *balance_cal);

/* Operations */
int cwiid_command(cwiid_wiimote_t *wiimote, enum cwiid_command command,
                  int flags);
int cwiid_send_rpt(cwiid_wiimote_t *wiimote, uint8_t flags, uint8_t report,
                   size_t len, const void *data);
int cwiid_request_status(cwiid_wiimote_t *wiimote);
int cwiid_set_led(cwiid_wiimote_t *wiimote, uint8_t led);
int cwiid_set_rumble(cwiid_wiimote_t *wiimote, uint8_t rumble);
int cwiid_set_rpt_mode(cwiid_wiimote_t *wiimote, uint8_t rpt_mode);
int cwiid_read(cwiid_wiimote_t *wiimote, uint8_t flags, uint32_t offset,
```

```
                uint16_t len, void *data);
int cwiid_write(cwiid_wiimote_t *wiimote, uint8_t flags, uint32_t offset,
                uint16_t len, const void *data);
/* int cwiid_beep(cwiid_wiimote_t *wiimote); */

/* HCI functions */
int cwiid_get_bdinfo_array(int dev_id, unsigned int timeout, int max_bdinfo,
                           struct cwiid_bdinfo **bdinfo, uint8_t flags);
int cwiid_find_wiimote(bdaddr_t *bdaddr, int timeout);

#ifdef __cplusplus
}
#endif

#endif
```