Home Theater PC
Media Control System

Media Control System Overview

What are the advantages of installing a Home Theater PC based media control system?

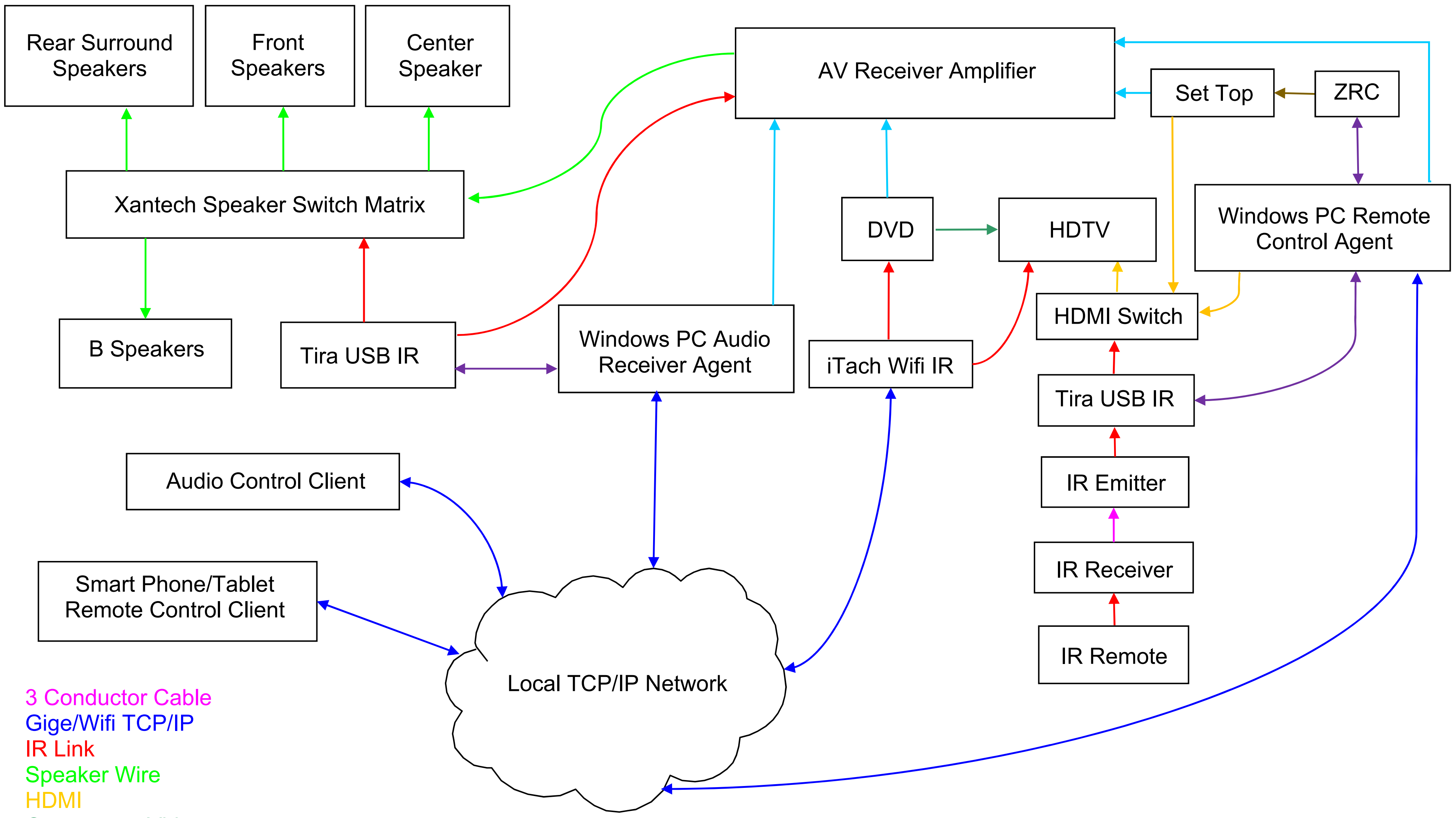
- Play media from any home computer using a single home theater
- Home theater control from a PC or smartphone
- Consolidation to a single universal remote
- A simplified, more intuitive user interface for playing music, television and movies

Media content is now available from a multitude of online sources: downloadable movies, Internet radio, mp3 music files, digital cameras, ripped DVDs, and a vast array of web based video content. Given the trend toward online entertainment content, integrating a home theater with the computers in your home will simultaneously broaden choice and enhance your home entertainment experience.

A speaker switch matrix is used for turning speakers on or off in a particular room. Surround sound is great for movies but not necessarily desirable for everyday television. There are occasions when you would like to enjoy music in the den but not in the computer room. For a party or working around the house, you may want to turn speakers on throughout your home. A myriad of combinations all managed from any computer anywhere in the house thanks to wifi networking.

The implementation example for this media control system was written to manage a Yamaha RX-V800 AV Receiver but the software was written to readily adapt to any home theater implementation with infrared remote control. Adapting the software to a particular receiver does require C and Java language programming knowledge. However, the software is generic enough to readily adapt to any system by only modifying the header files that contain the infrared codes. Any variety or arrangement of system components can be managed by simply defining the relevant remote control commands. The software for this system and all the development tools are freely available.

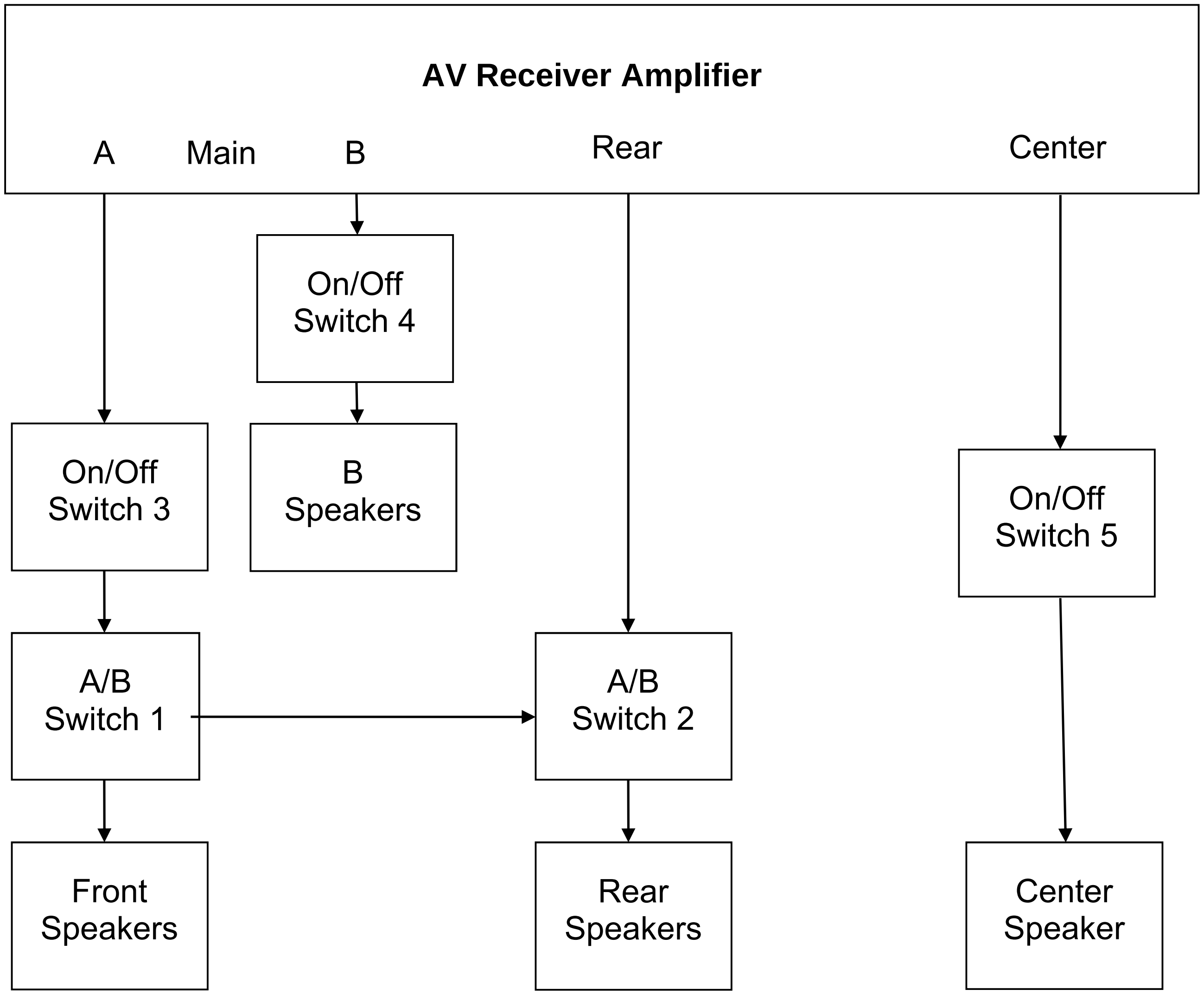
Media Control System Diagram



3 Conductor Cable
Gige/Wifi TCP/IP
IR Link
Speaker Wire
HDMI
Component Video
Toslink/Coax S/PDIF
USB
Zigbee Remote Control

IR Receiver and IR Emitter is the Xantech model 48095DBKIT
iTach Wifi IR is the Global Cache model WF2IR
Tira USB IR is the Home Electronics Tira-2.1 Remote Control
ZRC is the Texas Instruments CC2531EMK Zigbee Remote Control USB Dongle

Xantech Speaker Switch Matrix



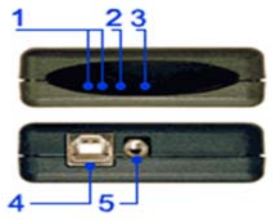
On/Off Switch is the Xantech model CC12

A/B Switch is the Xantech model SR21

Tira IR Receive/Transmit Device

The Home Electronics Tira 2.1 IR receiver/transmitter is used by both the audio receiver and remote control agents. The audio receiver agent uses the Tira device to send IR commands to control the audio receiver. The remote control agent uses the Tira device to receive IR commands from a universal remote and transmit IR commands to control an HDMI switch.

1. Two IR LEDs for transmitting IR codes
2. Short range sensor for capturing IR codes
3. Long range, 38KHz IR receiver
4. USB "B" Connector
5. 3.5mm jack for connecting external emitters (Center pin "+", sleeve "-")



IR commands are transmitted by a library call to the Tira device with an array of chars argument. Generating the array of chars for desired IR commands is done using the `mcs/util/tira/demo.exe` utility. Simply connect a Tira device to a PC USB port, run demo.exe, activate capture mode, point the an IR remote at the Tira receiver and press the button on the remote you wish to capture. The demo utility will output the IR codes in an array of chars format. Captured IR commands can be copied and pasted as data into agent software and sent to the Tira device on demand to automate control. The following example shows the output generated after receiving an IR command:

```
mcs\util\tira\demo.exe 5
Tira library loaded
Tira activated on com port 5
Tira set callback handler succeeded

a               activates capture mode
b               activate the capture callback method
c               cancels capture mode
!               transmit the dynamic IR code
i               audio power on
o               audio power off
f               fm
g               georgia
j               jimson
t               tv
v               dvd
y               video aux
u               master volume up
d               master volume down
m               mute toggle
)               channel a to front speakers
(               channel a to rear speakers
]               rear speakers from channel a
[               rear speakers from rear channel
+               channel a on
-               channel a off
>               channel b on
<               channel b off
}               subwoofer on
```

```
{               subwoofer off
;               dsp effect toggle
:               dsp dolby normal
9               hdmi tv source
0               hdmi dvd source
_               hdmi dvr source
1               fm1
2               fm2
3               fm3
4               fm4
5               fm5
6               fm6
7               fm7
8               fm8
l               effect speaker level menu
s               settings menu
↑               menu up
↓               menu down
→               menu right
←               menu left
q               quit
>a
Playback capture activated
IR Code captured!
DataSize: 163
0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x3E, 0x04,
0x00, 0x00, 0x3E, 0x02, 0x00, 0x00, 0x46, 0x00, 0x00, 0x00,
0xD0, 0x00, 0x00, 0x00, 0x2B, 0x13, 0x00, 0x00, 0x10, 0x01,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xDC, 0x82, 0xAC, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00,
    0x00, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x90, 0x65, 0x00, 0x00,
    0x03, 0x00, 0x00, 0x00, 0x47, 0x00, 0x00, 0x00, 0xFF, 0xFF,
    0xFF, 0xFF, 0x00, 0x01, 0x02, 0x02, 0x02, 0x03, 0x02, 0x02,
    0x02, 0x03, 0x02, 0x03, 0x02, 0x03, 0x02, 0x03, 0x02, 0x02,
    0x02, 0x03, 0x02, 0x02, 0x02, 0x03, 0x02, 0x02, 0x02, 0x02,
    0x02, 0x02, 0x02, 0x02, 0x02, 0x03, 0x02, 0x03, 0x02, 0x02,
    0x02, 0x03, 0x02, 0x03, 0x02, 0x03, 0x02, 0x02, 0x02, 0x02,
    0x02, 0x02, 0x02, 0x02, 0x02, 0x03, 0x02, 0x02, 0x02, 0x02,
    0x02, 0x02, 0x02, 0x03, 0x02, 0x03, 0x02, 0x03, 0x02, 0x04,
    0x00, 0x05, 0x02
    Capture deactivated
```

Transmit IR codes are defined in `mcs/common/tira_tx.h` and are used by both the receiver and remote control agents.

Besides capturing IR codes for transmit, the tira demo utility also captures received IR codes that trigger a callback. Capture receive IR codes by activating callback capture, pointing the IR remote at the Tira device and pressing the button of interest to print out a callback IR code:

```
>b
Callback capture activated
231700000000
```

Receive IR codes are defined in `mcs/agent/remote/tira_rx.h` and are used by the remote control agent.

Itach Wifi to IR Device

The iTach WF2IR device is used by the remote control agent to automate sending IR commands.

1.1.1.1	**Power Input**	-5 to 16V DC@300mA (wall power adapter included) –International plugs available -USB power cable draws power from USB port (included)
1.1.1.2	**Setup**	Integrated web server for easy configuration Configure through TCP/IP set up commands
1.1.1.3	**iHelp Setup Utility**	Downloadable **iHelp** software simplifies network discovery and setup
1.1.1.4	**IR Learning Utility**	Downloadable **iLearn** software allows for the capture and playback of IR commands for control uses
1.1.1.5	**Network Connection**	Industry standard 802.11b wireless interface, 2.5″ antenna Supports wireless LAN, adhoc (for setup) and Infrastructure modes Compatible to 802.11b
1.1.1.6	**LED Indicators**	Power and all connectors to indicate activity
1.1.1.7	**IR and Sensor Interface**	Connector: 3.5mm stereo jack 3 Independent selectable IR outputs or sensor inputs 3 IR emitters included 1 IR blaster included (supported on 3rd IR port)
1.1.1.8	**Device Packaging**	Aluminum extrusion case, rubber end caps and plastic face plates Dimensions (LxWxH): 3.25″ x 2.25″ x 1.25″ Weight: 3.25 oz. (6 oz. with power supply)

Use the iLearn utility to capture IR remote commands and paste them into file `itach_codes.cmd`. Running the perl utility `itach_header.pl >..\..\agent\remote\itach.h` will automatically convert the captured commands into a c language header file. The `itach.h` file is used by the `mcs\util\itach\demo.exe` and the remote control agent. See the readme file located in `mcs/util/itach/itach_capture_tips.txt` for more details about capturing itach IR commands.

Zigbee Remote Control Device

Zigbee remote control functionality has been integrated into the remote control agent using a Texas Instruments CC2531EMK USB dongle paired to a set top box. The remote control agent accepts IR messages or TCP/IP text messages and converts them into radio frequency Consumer Electronics Remote Control (CERC) commands.

Details for programming the USB dongle are located in the `mcs\util\zrc\cc2531\readme.txt` file.

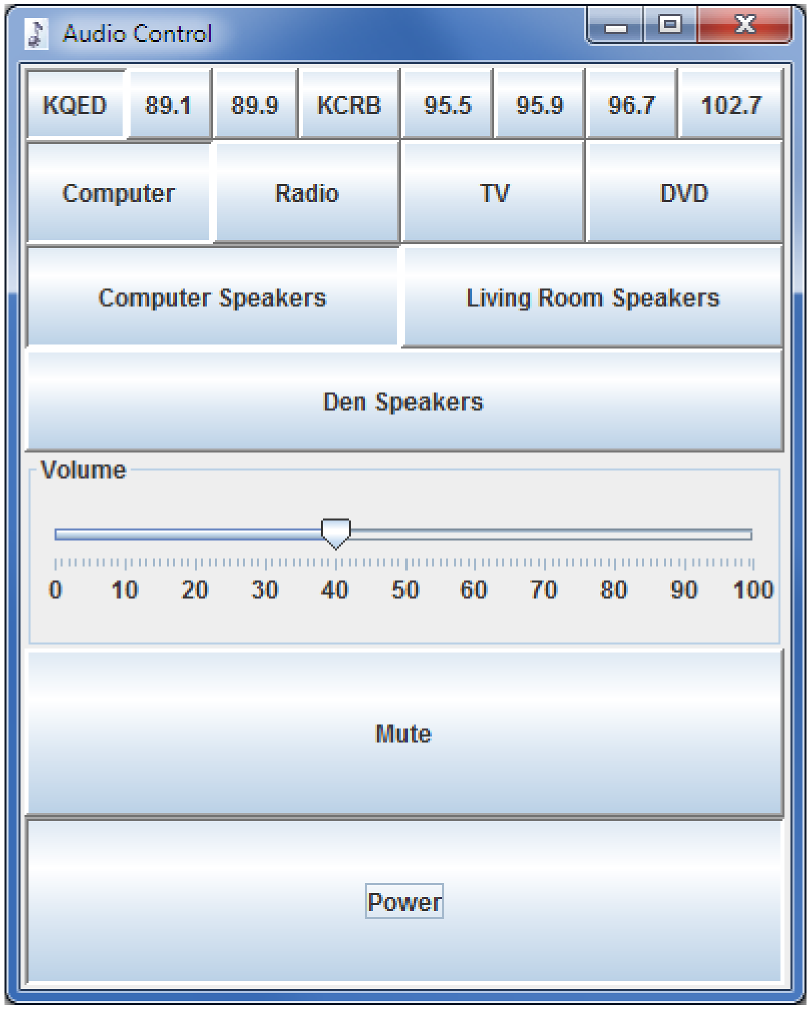
Zigbee USB Dongle (Texas Instruments Part # CC2531EMK)

The “demo.exe” test program is used to pair the TI Zigbee USB dongle to a Comcast set top box:

```
Windows demo.exe app <-> TI Zigbee USB Dongle <-> Comcast Set Top Box

> mcs\util\zrc\demo.exe

Configuring node........done
Calling RTI_InitReq...done
Reading pairing entry table...........done
Pairing entry found

CERC commands:
0    0
1    1
2    2
3    3
4    4
5    5
6    6
7    7
8    8
9    9
g    guide
↑    up arrow
↓    down arrow
→    right arrow
←    left arrow
cr   ok
x    exit

p    pair
u    unpair
q    quit
```

Audio Control Client

The following screen capture shows the Java based audio control client utility used for the installation example. The buttons shown can be easily reconfigured for a different installation. Pressing any button sends an IP message request to the audio receiver agent. The audio receiver agent translates the IP message and then transmits the corresponding IR command to the receiver.



The java based client utility requires the JRE to execute and the SE JDK to build. Run the utility with:

```
        javaw –jar acClient.jar
```

or, to see debug messages:

```
        java –jar acClient.jar
```

Java software is freely available from http://www.oracle.com/technetwork/java/javase/downloads

Audio Receiver Agent

Audio control clients send IP messages to a central audio receiver agent process called arAgent.exe. An IP text message such as “audio power on” is translated using the `tiraCmd` array found in `mcs/common/tira_tx.c`. The `tiraCmd` array contains structures that associate IP message strings to IR commands. For example, the `AUDIO_POWER_ON` command is defined by:

```
    {AUDIO_POWER_ON,     // unique integer
     SZ_AUDIO_POWER_ON,  // unique ip command string sent from a client
     &audioPowerState,   // valid values AUDIO_POWER_ON or AUDIO_POWER_OFF
     sizeof(irPowerOn),
     irPowerOn},         // char array passed to the Tira device for transmitting an IR command
```

The `irPowerOn` char array is generated as described in the section entitled *Tira IR Receive/Transmit Device*.

Remote Control Agent

A home theater PC like a Mac Mini plays media content on a TV and runs a remote control agent called rcAgent.exe. The rcAgent process accepts IR commands coming from an IR remote or TCP/IP text messages from a remote control client and then performs one or more actions. For example, when a user presses power on the universal remote, the television and the audio receiver turn on simultaneously.

The rcAgent revolves around an array of remote control command translations, `rcCmd`. Upon receiving an IR or IP input message, the rcAgent looks up the corresponding translation entry and then outputs the appropriate keyboard shortcut, IR command, or Zigbee remote control command based on context:

```
static struct rcCmd {
   int id;                       // cmdID enum
   const char * sz;              // ip command string input
   struct irRx * irRx;           // tira ir input
   struct shortcut * shortcut;   // synthesized keyboard shortcut output
   struct irTx * irTx;           // itach wifi to ir output array (TV, DVD, VCR)
   struct cercCmd * zrc;         // zigbee remote control output
}
```

The `irRx` structure defines IR codes received from a Tira IR receiver:

```
struct irRx {
   int size;             // number of ir code strings
   const char ** code;   // possible ir codes for a particular command
   int matchLen;         // number of chars to qualify as a match
   const char * sz;      // command message and debug description
   DWORD timestamp;      // GetTickCount() when ir cmd last received
};
```

Video Power Command Example

As an example, consider the `VIDEO_POWER` translation. Upon receiving the IP text message `SZ_VIDEO_POWER` or the `TIRA_VCR_POWER` IR code, the rcAgent looks up the `rcCmd`:

```
{
   VIDEO_POWER,                    // cmdID enum
   SZ_VIDEO_POWER,                 // ip command string input
   &irRxCmd[TIRA_VCR_POWER],       // tira ir input
   NULL,
   irTxCmd[ITACH_POWER].remote,    // itach wifi to ir output
   NULL
},
```

Tira device IR codes received are matched to a character string defined in `mcs/agent/remote/tira_rx.h`:

```
static const char * vcrPower[] = {
"231700000000",  // 12 byte encoding of an IR cmd from a universal remote
"ABBCFDFD555D"   // other possible encoding for the same command
};
```

IR codes are implementation specific and must be manually defined. Assuming the Tira device is configured on com port 5, run the remote control agent console program with the "–d" debug option. Running in debug mode prints out IR codes received from a universal remote to validate IR codes defined in `tira_rx.h`:

```
> rcAgentConsole.exe -d
irRx match found 16.20.00: 231700000000 -> ir vcr power
```

After defining and recompiling, IR code definitions become members of an array of `irRxCmd` structs:

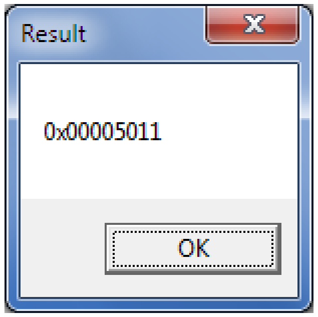
```
TIRA_VCR_POWER,
{sizeof(vcrPower)/sizeof(char *),vcrPower,CMD_CODE_LENGTH_EXT,SZ_IR_VCR_POWER,0},
```

Note in this implementation universal remote VCR codes are intentionally used because no physical VCR exists so those codes are free for rcAgent translations.

Upon receiving the IR command `TIRA_VCR_POWER` or the IP text message `SZ_VIDEO_POWER`, the rcAgent sends the IP message `irTxCmd[ITACH_POWER].remote[TV_MODE]` to the iTach device with the IR codes to turn on the TV display. The rcAgent also sends the `SZ_REGISTER` IP text message to the audio control agent to turn on the receiver. See the "Media Control System Diagram" at the beginning of this document to visualize message flow for this example.

Pause Command Example

The remote control agent also accepts commands to synthesize keyboard shortcuts to remotely control a media player. IR or IP commands for play, pause, stop, fast forward, rewind etc., translate via an `rcCmd` lookup to a corresponding keyboard shortcut for the running application. The included vkcode.exe (virtual key code) utility will generate the hex code for a particular keyboard shortcut. For example, running vkcode.exe and pressing <CTRL>-P will pop up message box:



```
0x00005011
```

0x5011 is added to the `shortcuts.h` header as a `#define` in an array of `virtualKey` structures:

```
    #define KBD_CTRL_P 0x5011   // remote pause -> windows media center Ctrl-P
    {
      SC_PLAY_PAUSE,
      {
         {
            KBD_SPACE,     // Netflix play/pause shortcut
            KBD_SPACE,     // VideoLAN play/pause shortcut
            KBD_CTRL_P,    // Windows Media Center play/pause shortcut
            KBD_SPACE      // YouTube play/pause shortcut
         },
         {
            SZ_KBD_SPACE,
            SZ_KBD_SPACE,
            SZ_KBD_CTRL_P,
            SZ_KBD_SPACE
         }
      }
   },
```

In this example, the `SC_PLAY_PAUSE` shortcut identifies a set of keyboard shortcuts for different applications; i.e. Netflix, VideoLAN and YouTube use <space>, Windows Media Center uses <ctrl><p>. Shortcuts are associated in `rcCmd` translations. In this example, the `SC_PLAY_PAUSE` id is used as an index into the array of `virtualKey` structures in an `rcCmd` translation:

```
{
   PAUSE,                               // unique enum ID
   SZ_PAUSE,                            // IP message input
   &irRxCmd[TIRA_VCR_PAUSE].key,        // tira IR message input
   &virtualKey[SC_PLAY_PAUSE].shortcut, // keyboard shortcut output
   irTxCmd[ITACH_PAUSE].remote,         // itach IR command output
   &cercCmd[ZRC_PAUSE]                  // zigbee remote control output
},
```

Receiving an IP or IR `PAUSE` request when a media player application (e.g. Netflix, VideoLAN, etc.) is playing causes the rcAgent to synthesize the appropriate keyboard `&virtualKey[SC_PLAY_PAUSE].shortcut`. When a DVD is playing the rcAgent sends `irTxCmd[ITACH_PAUSE].remote[DVD_MODE]` IR codes as an IP message to the iTach device to transmit the IR pause command. If cable tv is playing, the rcAgent sends the zigbee `&cercCmd[ZRC_PAUSE]` command to the set top box to pause live TV.

Third Party Products

The following are several of the links to different companies that provided the products that made this project a reality:

http://www.xantech.com – Xantech IR controlled relays and extenders

http://www.globalcache.com/products/itach/models2 - Wifi to IR control device

http://cygwin.com – free POSIX compatible GCC compiler

https://www.visualstudio.com/vs/compare – free Microsoft Visual Studio Community C++ compiler

http://mingw.org – free windows compatible GCC compiler

http://www.oracle.com/technetwork/java/javase/downloads – free Java development and runtime software

http://home-electro.com – the Tira PC USB interface IR receiver/transmitter

http://www.apple.com/macmini - Mac Mini

http://www.hdhomerun.com – Prime TV Tuner used to record shows using Windows Media Center

http://www.howtogeek.com/258695/how-to-install-windows-media-center-on-windows-10 – WMC install instructions

http://shark007.net/ – codecs needed to run WMC on windows 8 and later (use Shark007 SUGGESTED settings)

http://realvnc.com – free Virtual Network Computing (VNC) software

http://www.videolan.org – free media player software

http://www.monoprice.com – Blackbird 4k Pro 3x1 HDMI Switch with HDCP 2.2 Support

https://store.ti.com – CC2531EMK Zigbee Remote Control USB Dongle (search ebay for a less expensive option)

Compiler Setup

C++ freeware compilers can be used from Cygwin, MinGW, or Microsoft.

Cygwin:
Download and run their 32 bit setup.exe utility. When you get to the “Select Packages” page, Search for g++ and add the appropriate packages. Repeat by doing a Search for gcc and adding relevant packages. The Cygwin compiler requires Cygwin DLLs for compatibility with their unix based tools. When using a Cygwin shell, the Cygwin compiler is needed for console debugging.

MinGW:
Download and run the mingw-get-setup.exe. Next, open a command-line shell, cd to the mingw bin directory and run:

```
mingw-get.exe install mingw32-base
mingw-get.exe install mingw32-gcc-g++
```

Mingw only uses native Microsoft DLLs so it is a good choice for a production build.

Microsoft:
Download and run their Visual Studio Community installer. When you get to the page where you select “Workloads”, you only need to check “Desktop development with C++”. Installing additional tools is optional.

Compiling is accomplished by running their batch file from a command shell to initialize the environment:

```
<install dir>\VC\Auxiliary\Build\vcvarsall.bat x86
```

Manually defining the environment variable “`set MSVC=1`” directs build.bat files to use the Microsoft compiler instead of Cygwin or MinGW. Running a build.bat creates the executable corresponding to a subdirectory folder.

Software Licensing

Source Code Download

https://github.com/jhmpub/mcs.git