

## Control XMMS with your TV Remote

For many of us, the computer is much more than a machine that just crunches numbers. Some of us use it as a gaming station, while others use it as a glorified typewriter. It is however, commonly used as a multimedia station. When you watch movies on your computer, you can't do anything else except say, munch some pop-corn. It literally demands complete attention. But when it comes to listening to songs, we can just play them and go about doing our daily chores. But unless you have a perfect play list that enlists your favorite songs and also lists them in the order you prefer, you are likely to keep coming to your computer to instruct your favorite music player according to your desire(and mood).

Wouldn't it be nice if there were a remote control for the computer as well? Why not? Even infrequently controlled appliances like the A/C have remote controls now-a-days. Well, in those lines, now what we are going to do is build a simple infra-red receiver and control XMMS, the common Linux media player through it. You can then move through songs in the play list, increase or decrease volume, skip forwards or backwards, pause or stop the playback, load or clear play lists, quit the player, etc all with your TV remote control!

This fun stuff is made possible by the guys who created LIRC, or the Linux Infrared Remote Control Project. The LIRC project makes it possible to use home brew or commercially available hardware and provide a remote control facility to your Linux system. In this article, we shall also see how to create a Infrared receiver using commonly available components. This receiver will cost you around Rs. 50 and will not take more than an hour to construct.

### Installing LIRC

The LIRC package is available from [www.lirc.org](http://www.lirc.org). We will consider the source form of the package for our discussion. Once you download the package from the web site, follow the usual 5 step installation common to most source package installs:

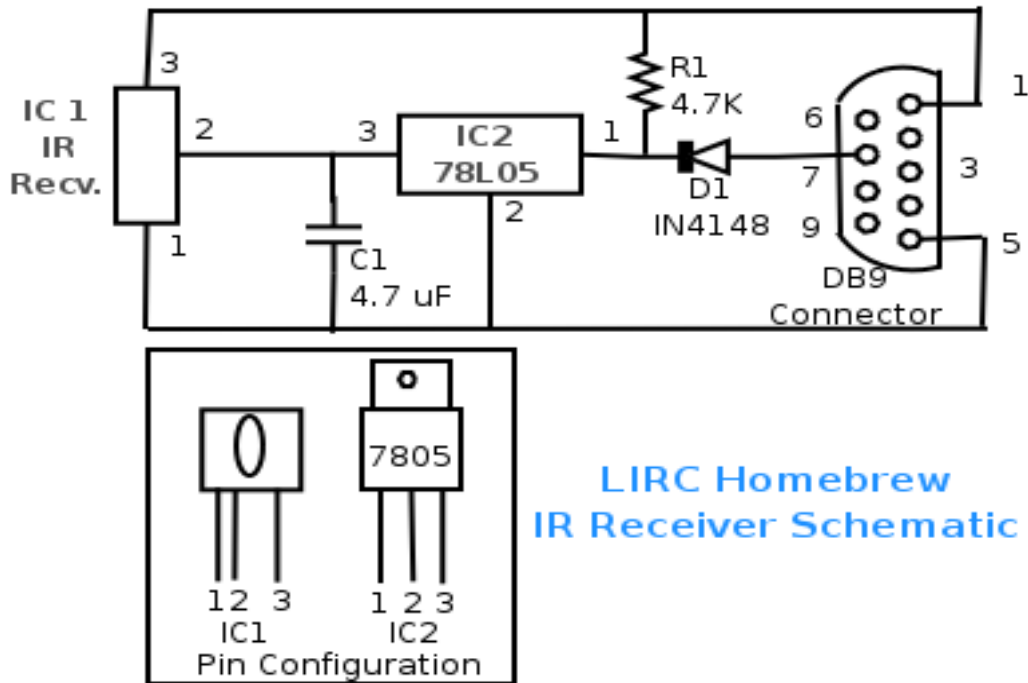
```
$ tar xvjf lirc-x.x.tar.bz2
$ cd lirc-x.x
$ ./configure
$ make
$ sudo make install
```

the LIRC package consists of 3 sets of items. There are kernel modules built for the kernel that you are running, there are some

general utility programs and finally there is a daemon that is the most important of all programs. It is this daemon that is responsible for decoding of the IR signals received from the serial port. The decoded signals are then made available over a UNIX domain socket so that applications can read and find out if any button that they are concerned about gets pressed. Actually, to do the fun thing presented in this article, you need not be a programmer at all.

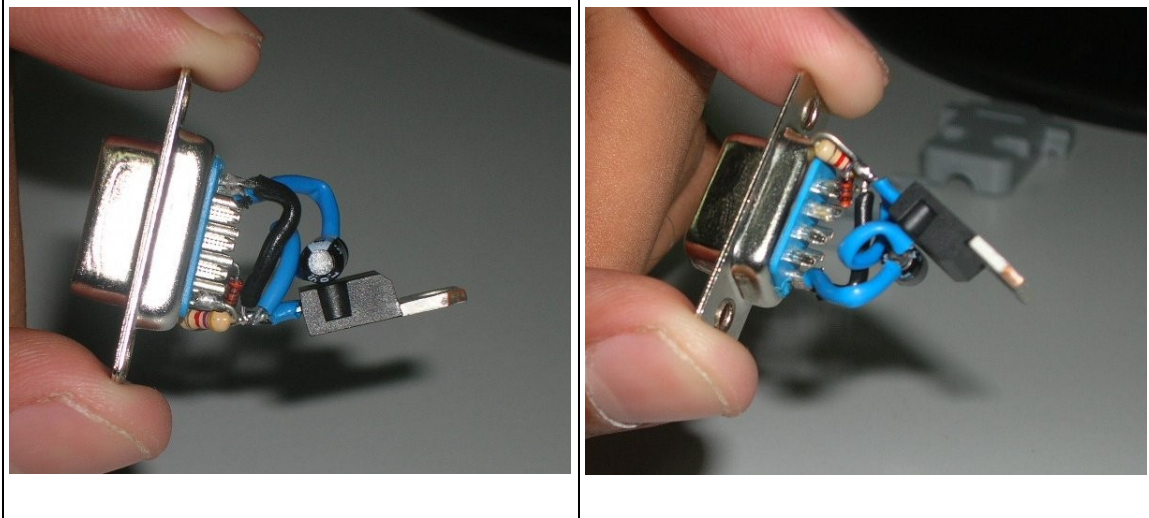
### The Device and how to assemble it

The circuit you need to build is a very simple one. All you need to know is how to solder. The heart of the circuit is an IR receiver.



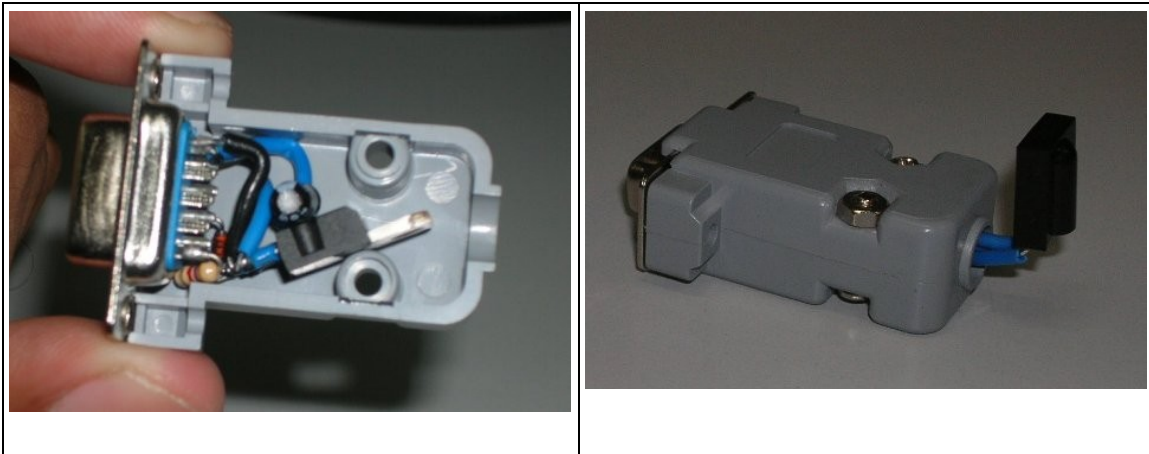
*The Device Schematic*

The part number mentioned in the LIRC web site is TSOP 1738 (and a whole list of working IR receivers is given at <http://www.lirc.org/receivers.html>), but when I asked around for that part in Chennai, what I found out was very strange indeed. Whatever the part number you wanted, you always got the same IR receiver! I tried many shops in Chennai's Ritchie street, but this was the default response from all shops I tried. They all seemed to stock just one or a few makes of the IR receiver. Sad that I did not get exactly what I wanted, I tested the device, once assembled with great anticipation. I was thrilled to see it work. So if you find yourself in the same situation, it would be just better to get whichever IR receiver you lay your hands on. Chances are high that it may just work!



*The components assembled on a DB-9 connector*

The rest of the components are just ones that supply power to the IR receiver. Since there are only a few components, you can consider packing them into the DB9 connector itself. This makes the IR device very compact and handy. But as I found out, doing that took a lot more time than I had imagined. But the final result was indeed neat. To just test the circuit, you can assemble it on a bread board or a general purpose PCB as well.



DB-9 Connector, Final Assembly

There is one very important requirement with regards to the kernel that you need to take care of. The serial port driver in the kernel must be compiled as a module and not compiled into the kernel itself. The reason for this is that, the LIRC driver accesses the serial port in a very low level fashion. If the serial port driver is already loaded, there will be problems accessing the hardware port. The serial port is a good example of a hardware peripheral that cannot be shared. If you have a distribution that comes with a kernel that has the serial driver

compiled in, you need to build a new kernel. A simple way to determine if the serial driver has been compiled as a module would be to look into the directory(substitute 2.x.x for your kernel version):

```
/lib/modules/2.x.x/kernel/drivers/serial/
```

If there is a file named `serial_core.ko` or `serial_core.o`, you are lucky. If not, it has been compiled into the kernel and you need a new kernel. I cannot discuss the steps of compiling your own kernel here. You need to look that information up in the right place. Just make sure that the serial port drivers are compiled as modules. Reboot the kernel if you compiled one fresh.

### **Loading the necessary modules**

We are going to use the serial device driver developed by the LIRC team. You first need to load it. Run the following command to do that:

```
# modprobe lirc_serial
```

This command will first load the module `lirc_dev` and then load `lirc_serial`. At this point, the Linux kernel's serial driver must not be loaded, if it is, use the `rmmmod` command to remove it first. If you want to load both these modules manually, you can use the `insmod` command to do so.

```
# insmod lirc_dev.o  
# insmod lirc_serial.o
```

Supply the full file path to `insmod` if the module files are not in the current directory.

### **The device files**

After you have installed LIRC, there are three new entries created in the `/dev` directory. The first one is the device file, the second one a socket and the third one, a pipe.

```
$ ls -l /dev/lirc*  
crw-rw-rw- 1 root root 61, 0 Jul 27 21:32 /dev/lirc  
srw-rw-rw- 1 root root 0 Jul 27 22:07 /dev/lircd  
prw-rw-rw- 1 root root 0 Jul 27 21:32 /dev/lircm
```

### **Setting access permissions**

To gain non-root access to the device, you need to set proper device permissions. Logging in as root for regular usage of the system is a bad idea. So, to use LIRC as a normal user, execute the following command:

```
# chmod a+rw /dev/lirc*
```

### **Testing if it works**

Once you have the hardware ready, its time to test it. Attach it to the serial port and grab a terminal. Run the “mode2” utility now. There is another common utility named “mode3”, make sure you are not running that by mistake.

```
$ mode2 -d /dev/lirc
space 2646
pulse 4581
space 4397
pulse 573
space 1625
pulse 624
space 1574
pulse 574
```

Get your TV or DVD player remote control and press buttons at random. The utility mode2 can be used to test hardware devices. It dumps information regarding the infrared pulses generated by remote controls, as shown in the listing. If you see some output like in the listing, it worked! The most difficult part of the project is over, now you can move on to the relatively simple configuration part of it.

### **Creating the configuration file**

We now have raw pulses spewing out from the serial port in response to the IR signals. Each button on the remote produces a distinct signal. It would be a good idea to name signals in relation to the buttons that produce them. For example, if you are using your DVD player remote, naming the signal that is produced while you press the play button as “play” would really be a good idea. LIRC uses a complex configuration file, but the task is made easy with the “irrecord” command that can create a useable config file for you. There are templates for various remote controls available on the LIRC web site, but not for the ones that you commonly find here in India. Anyways, creating a new config file does not take considerable time, so its worth the exercise.

```
$ irrecord -d /dev/lirc lircd.conf
```

```
irrecord - application for recording IR-codes for usage
with lirc
```

Copyright (C) 1998,1999 Christoph  
Bartelmus(lirc@bartelmus.de)

This program will record the signals from your remote control and create a config file for lircd.

Press RETURN to continue.

Copyright information is first printed

Now start pressing buttons on your remote control.

It is very important that you press many different buttons and hold them down for approximately one second. Each button should generate at least one dot but in no case more than ten dots of output. Don't stop pressing buttons until two lines of dots (2x80) have been generated.

Press RETURN now to start recording.

What you need to do now is press random buttons on your remote controls until each of the buttons produce a few dots on the screen.

```
.....  
.....  
Found const length: 128160  
Please keep on pressing buttons like described above.  
.....  
.....  
Space/pulse encoded remote control found.  
Signal length is 23.  
Found possible header: 220 7788  
Found trail pulse: 209  
No repeat code found.  
Signals are space encoded.  
Removed header.  
Signal length is 11
```

The parameters of your remote control have been gathered such that LIRC is now able to distinctly make out buttons you are pressing. Next, you need to enter the name for a button, hit enter and then press that button on the remote. LIRC will now associate the name you entered with the signal it received from the remote control. Repeat this for all the buttons you think you will use.

Now enter the names for the buttons.

Please enter name for the next button(<ENTER> to finish)

power

Now hold down button "power".

Please enter name for the next button(<ENTER> to finish)

vol-up

Now hold down button "vol-up".

Please enter name for the next button(<ENTER> to finish)

vol-down

Now hold down button "vol-down".

Please enter name for the next button(<ENTER> to finish)

prog-up

Now hold down button "prog-up".

[A lot more configuration here....]

Please enter name for the next button(<ENTER> to finish)

I hit the Enter key here. Next irrecord asks me to press and release one button on the remote repeatedly, as fast as possible, so that it can determine the toggle bit.

Checking for toggle bit.

Please press an arbitrary button repeatedly as fast as possible (don't hold it down!).

.....

Toggle bit is 2.

Successfully written config file.

The lircd.conf file should now have been created in your current directory. You must now copy it to the /etc directory.

```
$ sudo cp lircd.conf /etc
```

During this exercise, it is better to maintain line-of-sight in between the remote control and the IR receiver on the device. What I found out later is that even without proper line-of-sight, the device picks up the remote's IR beams easily. In my case, the dongle is actually attached to the rear of my computer, where you would almost always find the serial port. I'm comfortably able to use the remote even while working at the system since the beams are bouncing off walls to reach the IR receiver.

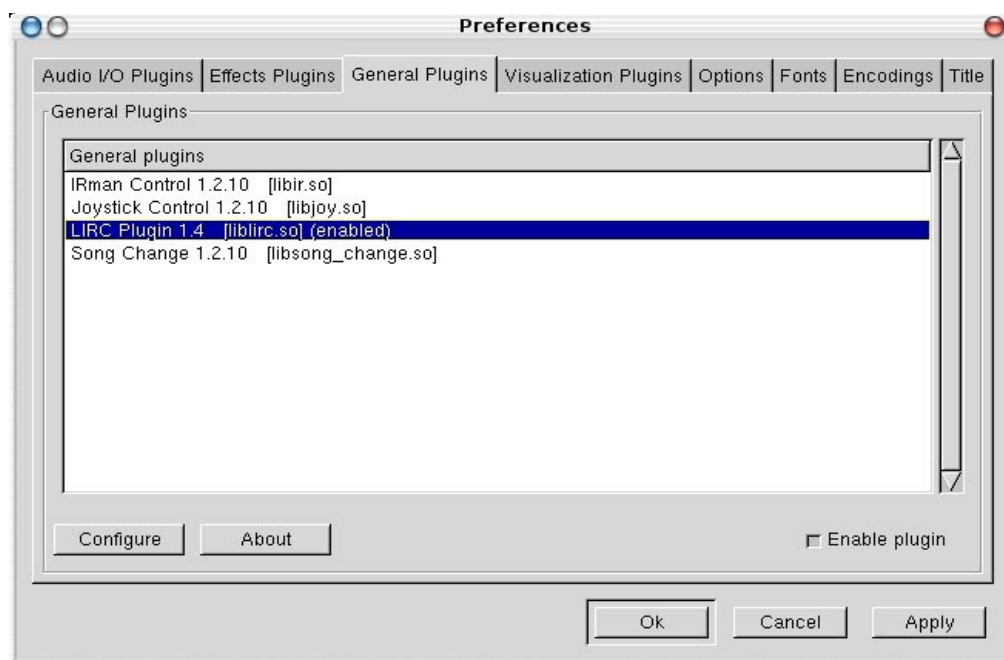
## Installing lirc-xmms-plugin

Now get the LIRC XMMS plugin from <http://www.lirc.org/software.html>. Once downloaded, you need to follow the same 5 step procedure to install it.

```
$ tar xvzf lirc-xmms-plugin-1.x.tar.gz
$ cd lirc-xmms-plugin-1.x
$ ./configure
$ make
$ sudo make install
```

Please note that to compile this package you need the development library/header packages for GTK-1.2 and XMMS. These will be named gtk-1.2-dev or gtk-1.2-devel, or similar, depending on your distro. If you, by chance installed LIRC from a binary package, you will need its development package as well.

Now we have the XMMS plugin installed. Do remember to enable this plugin once you execute XMMS by selecting the “LIRC Plugin 1.4” option and clicking on the “Enable Plugin” check box on the lower right corner. You can bring up the “Preferences” dialog by pressing “Ctrl+P” after running XMMS.



*XMMS Configuration*

Before you do anything else, you need to relate the buttons you have defined with the actions to be performed by XMMS. This is done through the “lircrc” file. Thoughtfully, there is a “lircrc” file provided with the lirc-xmms-plugin package. If you fire up your favorite editor

to open that file, you will see that its format is quite straight forward. In the lircrc file, you will find an "xmms" section, all you need to do is, change the name of the buttons(in the "button=" lines) to the names that you have given in the config file you created using the irrecord command. For easy reference you can open it in a separate window.

```
begin xmms
  begin
    prog = xmms
    button = play
    config = PLAY
  end
  begin
    prog = xmms
    button = clear
    config = PLAYLIST_CLEAR
  end
  begin
    prog = xmms
    button = pause
    config = PAUSE
  end
  begin
    prog = xmms
    button = stop
    config = STOP
  end
  .....[more items here]
end xmms
```

## **Starting other programs with your remote control**

You might have noticed that before the XMMS section in the lircrc file, there is a bunch of lines that read something like this:

```
begin
  prog = irexec
  button = power
  config = xmms&
  flags = once
end
```

As part of the LIRC package, a program "irexec" gets installed. The purpose of this program is to execute any command you like when particular buttons are pressed on the remote control. So you can add more lines like the ones used to start "XMMS" when the "power" button is pressed. Provide the executable name you wish to run in the

“config=” line. Please note that you need to be running the “irexec” program to execute other programs with a touch of your remote control button.

With necessary modifications to the lircrc file found in the lirc-xmms-plugin source directory, you should copy it to /etc for system wide usage.

```
$ sudo cp lircrc /etc
```

### **Starting LIRC**

First make sure that the necessary modules are loaded by doing “lsmod”. You must be able to see two modules “lirc\_dev” and “lirc\_serial”. If you don't, do “modprobe lirc\_serial”.

Now, start the two needed programs “lircd” and “irexec” from the command line. If you are not interested in starting other programs with your remote control, you need not start the “irexec” program. Please note that the “lircd” command will automatically become a daemon(background) process returning you to the shell prompt immediately. You can see if its running with the “ps -e” command if you need to. This is not the case with the “irexec” program, however. You need the pass to it the “-d” option if you need it running in daemon mode, which is recommended.

```
$ lircd
$ irexec -d
```

Thats it! You must now be able to control your XMMS with your remote control and also start other programs with your remote control as defined in the lircrc file!

### **If it did not work**

Since there are many things that you need to do, here is a checklist to try out if things did not go your way.

- Make sure the LIRC kernel modules are loaded and listed.
- Make sure all the /dev/lirc\* device entries have appropriate permissions. See the section “Setting access permissions” for details
- See if the two processes “lircd” and “irexec” are running.
- Check the logs. Open a terminal and do “tail -f /var/log/lircd”. Then start XMMS and see if something is out of place.
- Start XMMS from a terminal and see if there are any error messages being printed out from the LIRC XMMS plugin.

Please note that here I am assuming, on your system, the “mode2” command printed some output, which means that your hardware is working. If that is not the case you need to check you hardware.

**About the Author:**

*Shuveb Hussain enjoys programming Linux and loves the freedom offered by Free Software. The author blogs at <http://binarykarma.org>. You can reach him at shuveb at binarykarma dot org. This article first appeared in Linux For You (<http://lfymag.com>).*