Beginner's Guide

Infrared Magic in GNU/Linux

Learn how to control everyone's favourite XMMS with a TV remote

INTRODUCTION

Something that I want to share with you, with 100% satisfaction, is one that fulfilled my little ambition forever. As a staunch supporter and promoter of GNU/Linux and Free Software, i always wanted to approach things in different ways. I can say that rather than making GNU/Linux chase other proprietary systems in the mist of user friendliness which dominates among desktop users, we have to prove, since the GNU/Linux has to be running everywhere, how they can enjoy the easiness, flexibility and freedom by welcoming GNU/Linux to their PCs. The best method will be convincing others what they can do with GNU/Linux which they can't do with any other OS. My little experiment can be a definite guide towards this.

WHERE IT BEGINS?

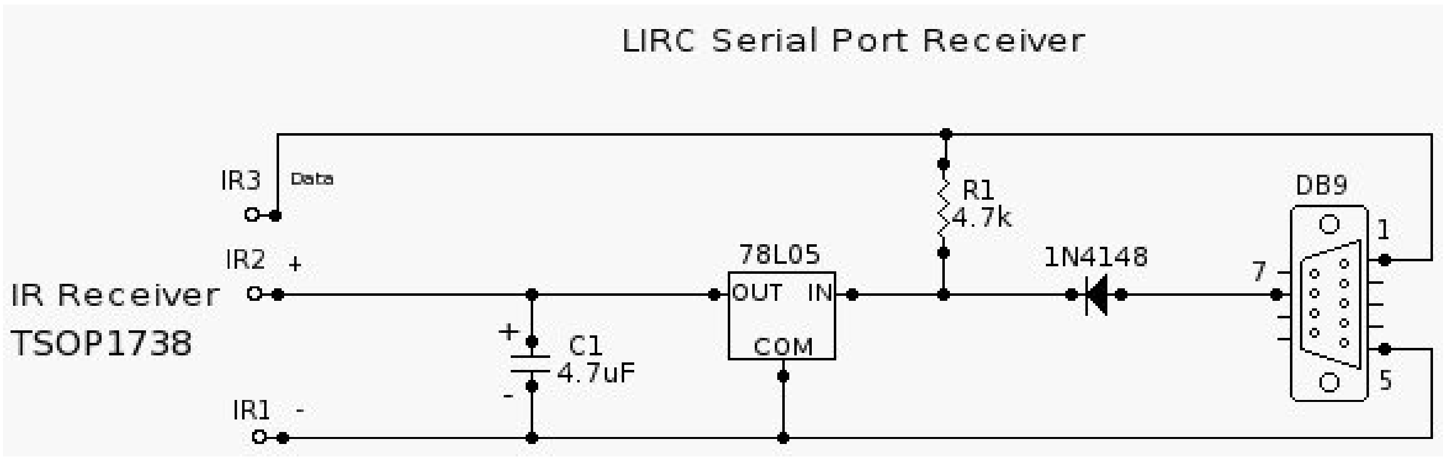
Everyone will be aware of the role played by FOSS in Hollywood Film Industries. At the same time, we can't neglect its presence in other areas too. From image manipulation to high quality graphics rendering, almost everything is possible with a variety of free tools.

But for novice users, doing some office work, playing songs and watching videos will be enough using their PCs. No one will be there in GNU/Linux world who are not familiar with XMMS (X Multimedia System), one of the oldest, powerful and most popular cross platform audio player. A feature rich application providing a good collection of plugins to boost up user's convenience.

While playing your favorite tracks, some of you may feel a little bit annoyance in reaching the PC and changing the current track and selecting your favourite randomly while you are at a considerable distance from your PC or pausing current playing track in order to attend a phone call and then continuing after that, as I did. But the doors are always opened for any one in the world of FOSS. Feel free to assure. Now onwards, no need to approach your PC to do those basic operations. You can control XMMS using a TV remote! Thanks to the excellent work done by Linux Infrared Remote Control (LIRC) project [www.lirc.org].

You can build a simple remote control system for your computer. Stay with me. I will show you how it can be done passionately.

LIRC
Linux Infrared Remote Control



>> **FIGURE 1** *Infrared sensor circuit*

LETS START

The project can be divided in to two segments. One is the Hardware Side where we will prepare an infrared sensor circuit and the second is the Software Side where we will tune the OS to work with the prepared circuit.

HARDWARE SIDE

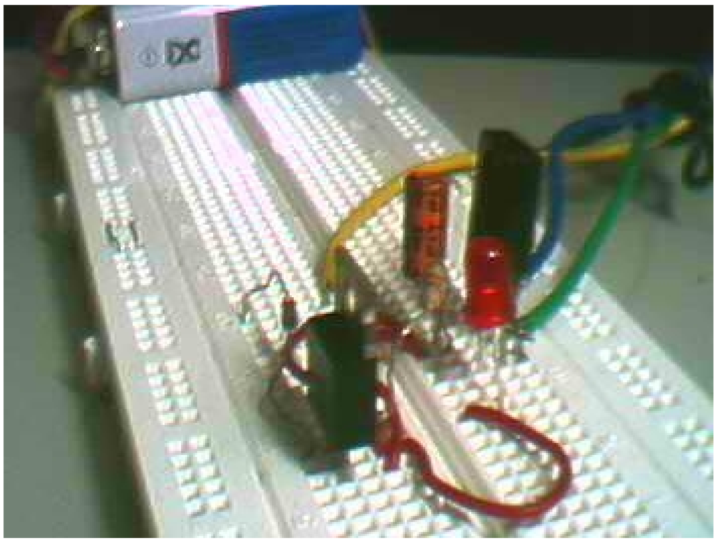
Building the sensor circuit

In order to control the music player with your remote, there must be an infrared sensor circuit which will accept the infrared signals generated by different buttons of the remote control and convert them in to pulses which will instruct the OS to perform intended tasks. To build such a circuit, components listed in TABLE 1 are required.

TABLE 1 *Circuit Components*
IC TSOP1738 - Standard infrared receiver
IC 78L05 – Voltage regulator
Diode IN4148, Resistor 4.7K, Capacitor 4.7uF – Provides power supply to the circuit
DB9 Connector - Standard serial port (female) connects circuit to the serial port of PC
Single Strand Wires – For connecting components
Red LED and 9V DC Battery – To test the circuit
DB9 Connector cover – If you want to cover it

Using the components, prepare a circuit as shown in FIGURE 1. It is highly recommended to implement the circuit on a bread broad before soldering the components forever so that we can test whether the circuit is perfectly working or not and can make any changes if required. So add a good bread board to the components list if you don't own one. When the circuit is formed following the figure, add LED to the circuit by connecting its +ve to pin 3 of TSOP1738 and its -ve to pin 2 of 78L05.

Now it is the time to test its working. Connect battery's +ve to diode's input and battery's -ve to pin 2 of 78L05. The circuit will look like as in FIGURE 2. The LED must be glowing at this time. Now start pressing buttons of your remote control. The LED will blink at each of the key press. If everything is happening as mentioned, your circuit is perfect to implement. If something seems to be wrong, check your connection and also assure that there is no fault with any of the components used.



>> **FIGURE 2** *Circuit implemented on a bread board*

NOTE *You can verify DB9 connector's working by connecting an LED to its pin 1 and 5. The LED must be responding in the same manner of the one in the bread board.*

SOFTWARE SIDE

A little bit of preparation?

Since we are connecting sensor circuit to the serial port of the system, there should be driver modules in the kernel especially to handle it. We will see how to generate driver modules for the sensor circuit later. By default the serial device support will be built in to the kernel that comes with standard GNU/Linux distributions. Every time the serial driver will dominate each of your serial port even though you are not using any devices (eg: modem, mouse etc) connected to it. In order to make sensor driver access the port, we have to remove the standard serial driver temporarily. For that, we must recompile the linux kernel to make the serial driver load as modules so that we have to load them explicitly if they want to access serial ports.

NOTE *If you can see two modules named serial_core.ko and serial_cs.ko in the /lib/modules/2.6.x/kernel/drivers/serial directory where 'x' is the version parameters of your kernel, everything is ready for you. You can safely jump in to the section **Preparing the system**.*

Kernel Compile HOW TO

Don't run away from the project! No need to consider compiling a linux kernel as a nightmare. Any one can do it and make your system boot with new kernel by directly following a few steps given below. All that you have to do is to spend a considerable amount of time which will depend on your system configuration. If you have new computer with a new generation processor and sufficient memory, it will be over in few minutes.

NOTE *I recommends you to use a standard kernel source rather than using those distributed by various GNU/Linux vendors. Mandrakelinux 10.1 shows some unresolved problems during the remote control configuration (to be explained in a coming segment). There is no hope by using a standard kernel in Mandrakelinux 10.1. I can't say the same problem exists in other versions of Mandrakelinux since i don't have experience. So it is better to do things in an alternate distribution using a standard kernel especially if you are using a non standard remote control. Visit www.lirc.org/remotes to see the list of remotes directly supported by the LIRC project. Mandrakelinux (now Mandriva) users need not worry. Once you prepared a configuration for your remote, you can stay with your favourite distribution for the rest of the procedure.*

To those people who are masters in kernel compiling. Hey.. You don't have to follow my childish steps. I can't guide you since i am not a Master of Universe in compiling a kernel like you! You need not stay here even for a sec. Take a long jump to the next segment.

Step 1
Get a standard linux kernel source. If you have rpm package, install it and the source can be found in /usr/src directory. If you have zipped archive, unzip it to a convenient directory.

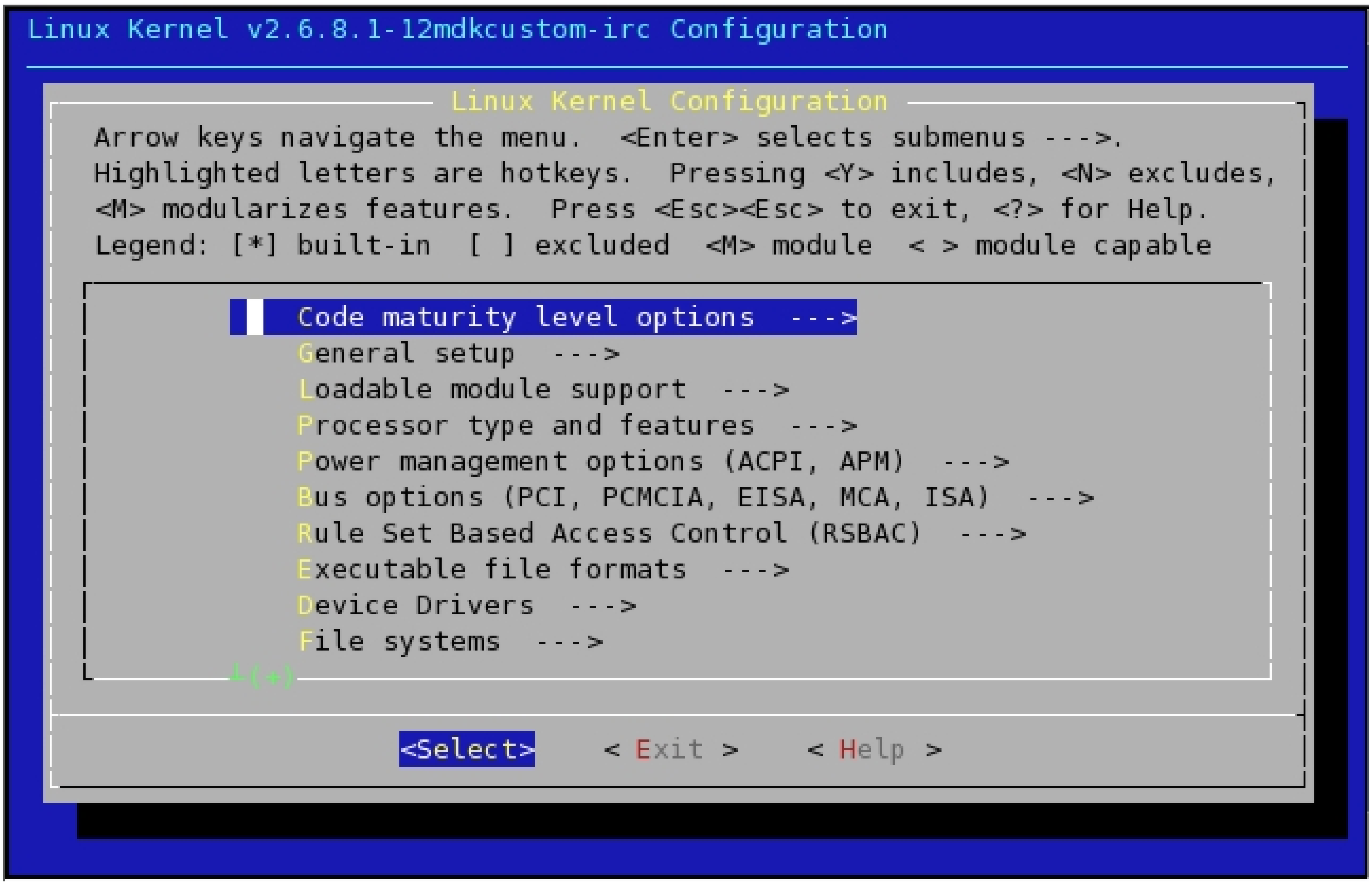
Step 2
Then get in to the directory by

```
cd /usr/src/2.6.x
```

As super user, do the following commands

```
make mrproper
make menuconfig
```

No you will see a kernel configuration window as shown in the FIGURE 3. From the listed options, select *Device Drivers-> Character devices-> Serial drivers*. From the Serial drivers window, select the first entry <*> *8250/16550 and compatible serial port* by using arrow keys and press *M* or *Space*. Assure an <*M*> is appeared instead of default <*> at the start as shown in the FIGURE 4. Here you have select the serial driver to load as module rather than built in to the kernel by default.



```
Linux Kernel v2.6.8.1-12mdkcustom-irc Configuration
                    Linux Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus --->.
 Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
 <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
 Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

            Code maturity level options  --->
            General setup  --->
            Loadable module support  --->
            Processor type and features  --->
            Power management options (ACPI, APM)  --->
            Bus options (PCI, PCMCIA, EISA, MCA, ISA)  --->
            Rule Set Based Access Control (RSBAC)  --->
            Executable file formats  --->
            Device Drivers  --->
            File systems  --->
        ↓(+)

            <Select>    < Exit >    < Help >
```

>> FIGURE 3 *Kernel configuration main window*

Then either by selecting < *Exit* > or by using *Esc* key, get back to the initial window of kernel configuration. Then select *Save Configuration to an Alternate file* at the bottom and press *Enter*. A dialog box will be asking to enter the name of alternate file. You can enter a valid name like *2.6.xcustom-irc* which must differ from the kernel version you are using. Then press *Ok* and select < *Exit* > from the window bottom and give ' Yes ' to the file saving confirmation dialog.

Step 3

The configuration is over. Now you are about to begin the compiling. Do

```
make clean
make
```

The compilation process begins and you can see hundreds of pretty formatted lines scrolling up on your monitor. For a few minute you will be eagerly watching the 'beauty of scrolling lines'. After that you will be prepared to leave your system at any cost. So before getting fooled by your system, it would not be worse to take my advice. Leave your system alone and do whatever you like and don't come back in less than an hour especially if you have an older system.

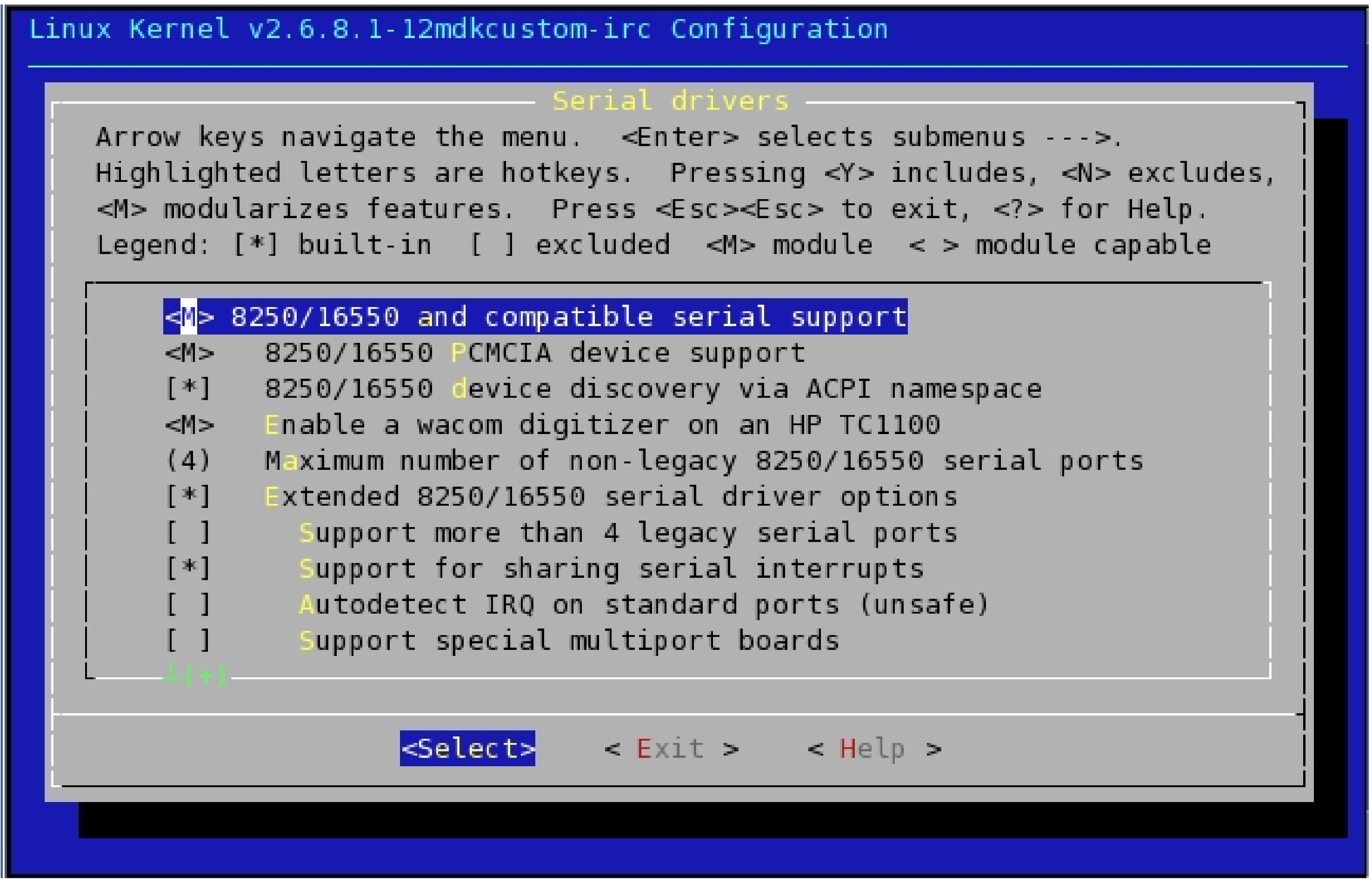
NOTE

Here I am explaining everything with an assumption that you are using a 2.6 kernel. There will be slight variation in the procedure for compiling an older 2.4 kernel. Refer README which will be residing in the source directory for a detailed procedure.

When the process gets finished, do

```
make modules
make modules_install
```

This will create a directory with a name that you gave while saving kernel configuration to an alternate file in **Step 2** under /lib/modules directory and will store all the generated modules there. You finished the kernel compiling. Check whether



```
Linux Kernel v2.6.8.1-12mdkcustom-irc Configuration

                         Serial drivers
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

      <M> 8250/16550 and compatible serial support
      <M>   8250/16550 PCMCIA device support
      [*]   8250/16550 device discovery via ACPI namespace
      <M>   Enable a wacom digitizer on an HP TC1100
      (4)   Maximum number of non-legacy 8250/16550 serial ports
      [*]   Extended 8250/16550 serial driver options
      [ ]     Support more than 4 legacy serial ports
      [*]     Support for sharing serial interrupts
      [ ]     Autodetect IRQ on standard ports (unsafe)
      [ ]     Support special multiport boards
      ↓(+)

            <Select>    < Exit >    < Help >
```

>> FIGURE 4 *Serial driver configuration as modules*

the standard serial modules are generated as mentioned earlier. If anything went wrong, refer other advanced documentations or howtos which will explain the compilation in detail.

Step 4

The next thing to do is to boot the system with the newly prepared kernel. For that, you have to add the new kernel image, which will be generated in arch/i386/boot directory of the kernel source, to system's boot menu.
Do

```
cp /usr/src/2.6.x/arch/i386/boot/bzImage
/boot/vmlinuz-2.6.xcustom-irc
```

NOTE

Be careful not to rewrite your existing kernel images under /boot directory. It is highly recommended to append the new kernel's version with the 'vmlinuz-' string as the name for newly compiled kernel image. The current kernel's version can be obtained by cat /proc/version

Then create an initial RAM disk image by

```
mkinitrd /boot/initrd-2.6.xcustom-irc.img
2.6.xcustom-irc
```

Now its time to play with boot loaders. You have to make entries for the new kernel in your boot loader. Here i am making an assumption that your OS is installed in the 2nd logical partition of your first hard disk (if you have more than one).

GRUB way

If you are using GRUB as the boot loader, open the configuration file /boot/grub/menu.lst in your favorite editor and add the following lines.

```
title Linux [IRC]
root (hd0, 0)
kernel /boot/vmlinuz-2.6.xcustom-irc root=/dev/hda6
ro
initrd /boot/initrd-2.6.xcustom-irc.img
```

LILO way

If you are using LILO instead of grub, open /etc/lilo.conf in a editor and add the following lines.

```
image=/boot/vmlinuz-2.6.xcustom-irc
label="Linux [IRC]"
root=/dev/hda6
initrd=/boot/initrd-2.6.xcustom-irc.img
read only
```

Save and exit from the configuration file and reboot the system in to the new kernel.

Preparing the system

Make the packages given in TABLE 2 readily available in your system.

TABLE 2 *Software Packages*

lirc-0.7.x.tar.bz2 (www.lirc.org) – LIRC source package
xmms-1.2.10.tar.gz or **xmms-1.2.10-1.i386.rpm** (www.xmms.org) – XMMS source or rpm
lirc-xmms-plugin-1.4.tar.bz2 (www.lirc.org) or **xmms-lirc-1.4-fr1.i386.rpm** (http://ftp.freshrpms.net/pub/freshrpms/redhat/9/xmms-lirc/) - XMMS LIRC plugin

Now the packages are ready and the hardest part of the project is about to begin. During this stage, you must keep a little bit of patience (Hmm.. one more advice!). Otherwise you will feel as you are a patient.. may have to consult a doctor also. Don't blame me...

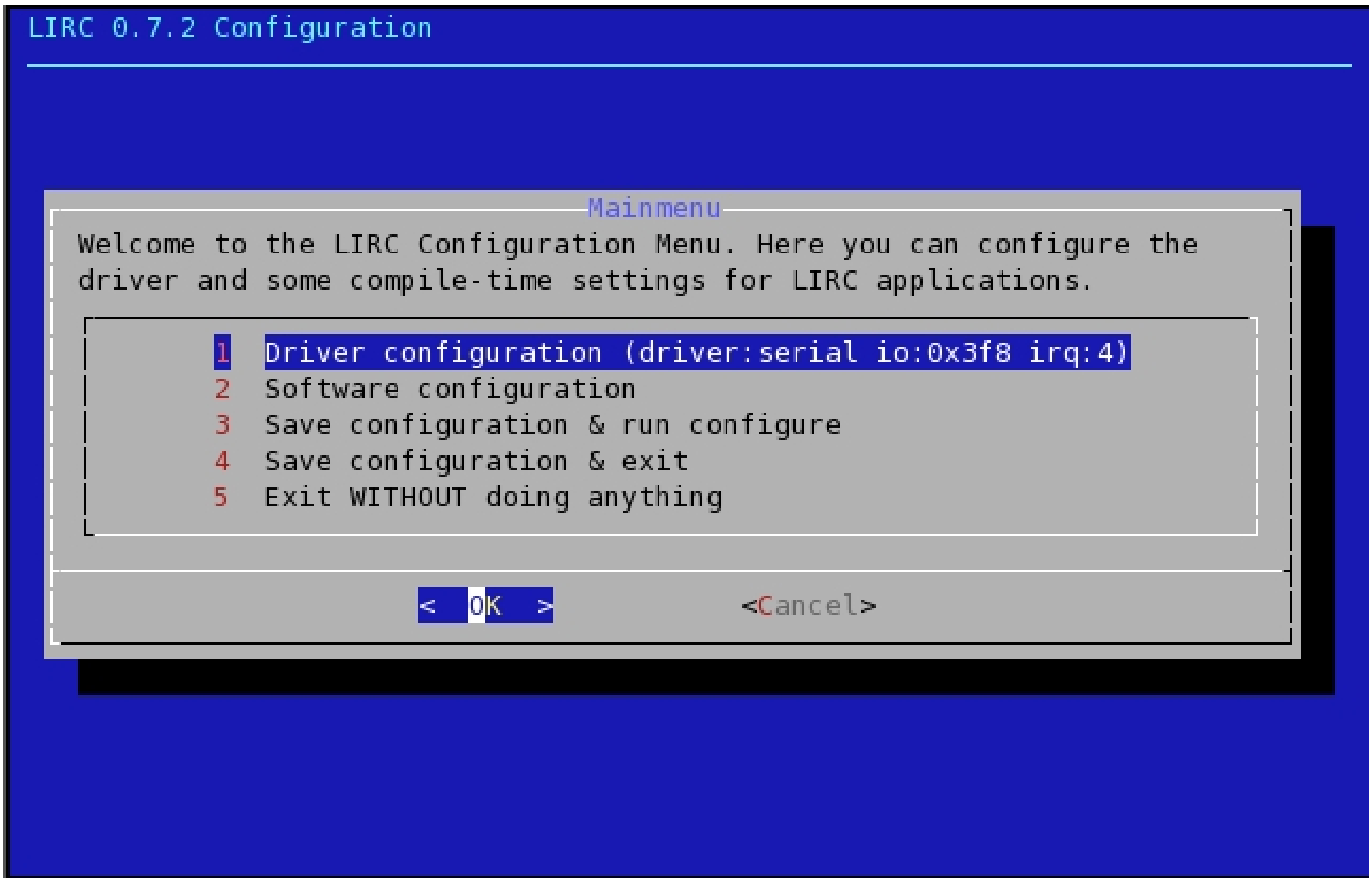
Unzip the lirc source as follows

```
bunzip2 lirc-0.7.x.tar.bz2
tar xvf lirc-0.7.x.tar
```

Then

```
cd lirc-0.7.x
./configure
```

Now you can see a window as shown in FIGURE 5. From the *Mainmenu*, select the entry *1 Driver configuration* and press *Enter*. Then a list of devices will be presented. Select *2 Home-brew (16x50 UART compatible serial port)* and press *Enter*. Then select the serial port to which you are intending to connect the sensor circuit and press *Space* and then *Enter*. Then in the *Driver specific Options*, make the second entry *2 Software generated carrier* selected and then *Enter*. Then select *2 Software Configuration* from



```
LIRC 0.7.2 Configuration

                              Mainmenu
 Welcome to the LIRC Configuration Menu. Here you can configure the
 driver and some compile-time settings for LIRC applications.

          1  Driver configuration (driver:serial io:0x3f8 irq:4)
          2  Software configuration
          3  Save configuration & run configure
          4  Save configuration & exit
          5  Exit WITHOUT doing anything

                    <  OK  >            <Cancel>
```

>> FIGURE 5 *LIRC source package configuration*

Mainmenu and *Enter*. Here only the first option *1 Compile tools for X-Windows* need to be selected. Then return to *Mainmenu* by pressing *Enter* and select the third choice *3 Save configuration and run configure* and *Enter*.

Then compile by

```
make
```

As super user, install by

```
make install
```

Now two kernel modules named lirc_serial.ko and lirc_dev.ko will be generated, which is the device driver for the circuit prepared earlier, as a result of above process under /lib/modules/2.6.xcustom-irc/misc directory. Addition to these, following device nodes will be generated under /dev.

/dev/lirc0
/dev/lirc
/dev/lircd
/dev/lircm

Now connect the sensor circuit to the serial port of the PC. Assure that you have connected to the same port which is specified during the lirc package configuration. Now load the driver modules by

```
modprobe lirc_serial
```

This will load the module lirc_serial.ko together with lirc_dev.ko. You can assure whether the modules are properly loaded by

```
lsmod
```

If the modprobe didn't work as expected, you can perform the same functions by

```
insmod /lib/modules/2.6.xcustom-irc/misc/lirc_dev.ko
insmod /lib/modules/2.6.xcustom-irc/misc/lirc_serial.ko
```

NOTE

If you get a message like 'driver resource is busy' or some thing like that, that will be due to the preloaded modules serial_core and serial_cs. Assure whether they are loaded or not. If loaded, unload them by
rmmod -f serial_cs
rmmod -f serial_core

Device is working or not?

As soon as the modules are loaded, the LED in the circuit will start glowing. To allow other users in the system to use the circuit prepared by you, you have to change the permission of lirc devices. You can do it by

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

Now we are at the stage to test circuit's interaction with OS. For this purpose, you can use an excellent utility called mode2 which shows the pulse/space length of infrared signals generated by an external device, in this case your remote control. At the terminal, give

```
mode2 -d /dev/lirc0
```

Now the command looks to be waiting to show you something. Now stop playing with your terminal. All you have to do is to take the remote control and start pressing various buttons towards the sensor circuit. At this time you must see something as shown below on your terminal.

```
pulse 594
space 517
pulse 613
space 536
pulse 623
space 508
pulse 572
space 1668
pulse 622
space 512
pulse 570
space 1666
pulse 596
space 1625
```

If you can see, yes, your device interacting with the OS properly.

Configuring Remote Control

Here we have to create a configuration file for the remote control so that the lirc system can distinguish the pulses generated by lirc circuit corresponding to the IR signals from each button of the remote. A number of templates are coming with the lirc package for certain remotes. You can find those configuration files under remotes directory of lirc source.

>> FIGURE 6 *Remote control used for the experiment*

NOTE *In the idle time, the sensor IC output will be active high generating 40Hz and at the time we press the buttons IC output be active low generating 4Hz.*

But it would be better if you create a configuration file for your remote. That is possible by using a utility named *irrecord*. Three cheers to Christoph Bartelmus who are behind this amazing utility. Start configuring remote by

```
irrecord -d /dev/lirc0 lircd.conf
```

An introductory message as shown below will appear on your terminal.

```
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.


A proper config file for lircd is maybe the most
vital part of this package, so you should invest
some time to create a working config file. Although
I put a good deal of effort in this program it is
often not possible to automatically recognize all
features of a remote control. Often short-comings of
the receiver hardware make it nearly impossible. If
you have problems to create a config file READ
DOCUMENTATION of this package, especially section
"Adding new remote controls" for how to get help.

If there already is a remote control of the same
brand available at http://www.lirc.org/remotes/ you
might also want to try using such a remote as a
template. The config files already contain all
parameters of the protocol used by remotes of a
certain brand and knowing these parameters makes the
job of this program much easier. There are also
template files for the most common protocols
available in the remotes/generic/ directory of the
source distribution of this package. You can use a
template files by providing the path of the file as
command line parameter.

Please send the finished config files to
<lirc@bartelmus.de> so that I can make them
available to others. Don't forget to put all
information that you can get about the remote
control in the header of the file.

Press RETURN to continue.
```

NOTE *Philips and certain other companies follow RC-5 encoding for their remotes. The RC-5 template is available under the generic directory of the remote templates.*

NOTE *You must follow each and every step of this process with immense care and patience which will determine the performance of the lirc system.*

Follow the instructions and when you are about to name each remote button, open a new terminal and unzip the xmms lirc source package by

```
bunzip2 lirc-xmms-plugin-1.x.tar.bz2
tar xvf lirc-xmms-plugin-1.4.tar
```

Then

```
cd lirc-xmms-plugin-1.x
```

Open the file named lircrc which is the contains the configuration for xmms lirc plugin.

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

begin xmms
  begin
    prog = xmms
    button = play
    config = PLAY
  end
  begin
    prog = xmms
    button = clear
    config = PLAYLIST_CLEAR
  end
```

>> **FIGURE 7** *Beginning portion of lircrc file*

The file shows some of the options in xmms that can be controlled using a remote. The xmms commands for corresponding functions are given after *config*= and the name of the button to control the function is given after *name*=. You need not make any changes here. Get back in to the first terminal and give the exact names found in the lircrc file here also for each of the buttons you are configuring to control different possible xmms functions. At the end of configuration, there will be a step to set the toggle bit for your remote. Do as what it says and the toggle bit will be set. You need not worry if no toggle bit is set for your remote. Save the file and exit.

If anything went wrong with remote configuration, remove the lircd.conf file and try again from beginning.

Now onwards both configuration files should reside in intended directories. The lircd.conf should reside in /etc and lircrc in /etc or in home directory as a hidden file. Do

```
cp lircd.conf /etc
cp lircrc /home/abc/.lircrc
```

assuming abc is your user name.

The XMMS side

If your system doesn't have xmms installed, install it at first. You can choose either rpm or source. It doesn't matter me. But i am not going to explain the install procedure since who told you to use a system that doesn't have even xmms? :)

Now install the xmms lirc plugin either using source or rpm.

If source, then do

```
cd lirc-xmms-plugin-1.x
./configure
make
make install
```

If anything unfortunate happens, try the rpm package by

```
rpm -ivh xmms-lirc-1.x-fr1.i386.rpm
```

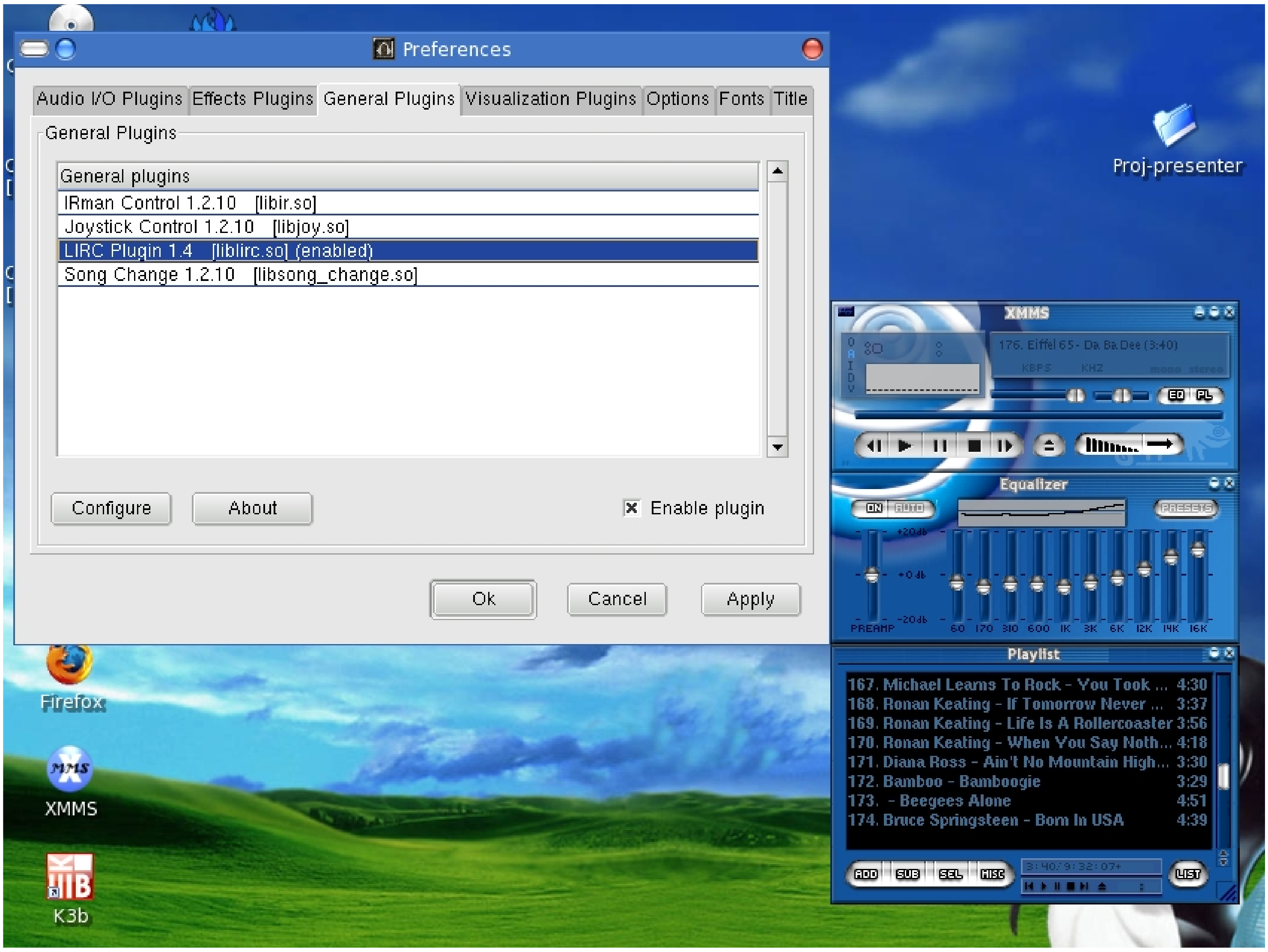
Now the most import thing is going to do. Invoke the lircd daemon, which is used to decode infrared signal and provides them on a domain socket, as super user by

```
lircd -d /dev/lirc0
```

This daemon is the one which makes the OS do actions according to the infrared signals generated by the remote control.
Now as an ordinary user, open xmms from terminal. From xmms, goto *Options*-> *Preferences*-> General Plugins and you can see *LIRC plugin 1.x* in the plugin list. Enable the plugin and *Apply* and *Ok* (FIGURE 8).

Now this is the moment what me and you are waiting for. Take your remote and press the configured buttons. Definitely you can see the expected response from your xmms. Play your favorite track .. pause it .. then play it again .. forward .. increase volume .. jump to next track .. and do whatever you like. After enjoying a little bit of music, lets think again. Is it possible to start the xmms itself with the remote?

Answer is Yes. Close xmms. Type irexec in the terminal and Enter. You can see the command is waiting for some input. Just press the button which is configured to start the xmms application in lircrc file. Xmms will be started immediately.



General Plugins

General plugins
IRman Control 1.2.10 [libir.so]
Joystick Control 1.2.10 [libjoy.so]
LIRC Plugin 1.4 [liblirc.so] (enabled)
Song Change 1.2.10 [libsong_change.so]

[✓ Enable plugin]

```
176. Eiffel 65 - Da Ba Dee (3:40)

167. Michael Learns To Rock - You Took ... 4:30
168. Ronan Keating - If Tomorrow Never ... 3:37
169. Ronan Keating - Life Is A Rollercoaster 3:56
170. Ronan Keating - When You Say Noth... 4:18
171. Diana Ross - Ain't No Mountain High... 3:30
172. Bamboo - Bamboogie                     3:29
173.  - Beegees Alone                       4:51
174. Bruce Springsteen - Born In USA        4:39
```

>> **FIGURE 8** *Enabling the lirc plugin installed for xmms*

NOTE

While starting xmms, you must not see any messages like 'could not connect to socket'. This will be due to the absence of lircd daemon. Check whether it is active by ps ax. If not there, load again. You may also see an error like could not open config files /home/abc/.lircrc and /etc/lircrc. Check whether the lircrc files are properly placed.

NOTE

If you can't play mp3 tracks in xmms, that is nothing to do with lirc. Check whether the mp3 input plugin is working properly. A few distributions like Red Hat and Ubuntu removes mp3 support from their multimedia applications due to some patent issues.

Make it permanent

Now is the right time remove the components from the bread board and prepare a well soldered circuit. You can use a circuit board for the purpose. But there is no necessity.

Rather than loading the lirc modules and invoking lirc daemons each time, you can write a shell script for the purpose as given in the FIGURE 9.

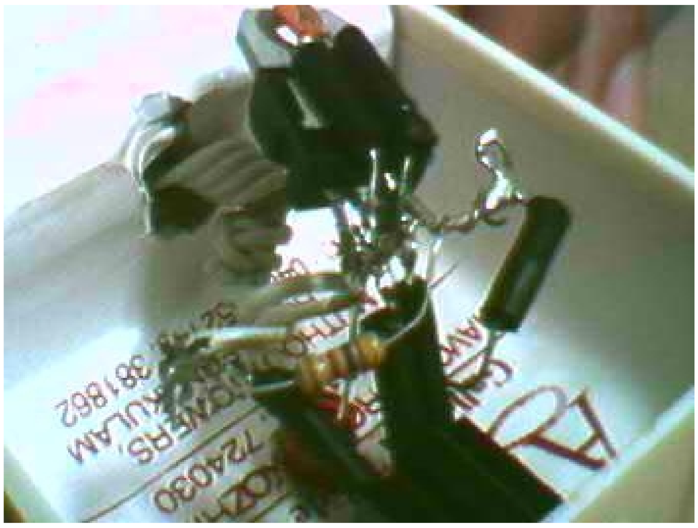
Now copy the script to /etc/init.d directory and make appropriate links to the script from /etc/rc5.d and /etc/rc3.d. Now onwards, you can use the remote control with xmms without depending those 'dirty' terminals at each and every time.

```
# description: Loads necessary modules
required for serial devices

# Load necessary modules for LIRC XMMS
system
modprobe lirc_serial

# Load the necessary modules for common
serial devices
modprobe serial_core
modprobe serial_cs

# Load lirc daemons and Programs
/usr/local/sbin/lircd -d /dev/lirc0
sudo -u abc /usr/bin/irexec --daemon
/home/abc/.lircrc
```

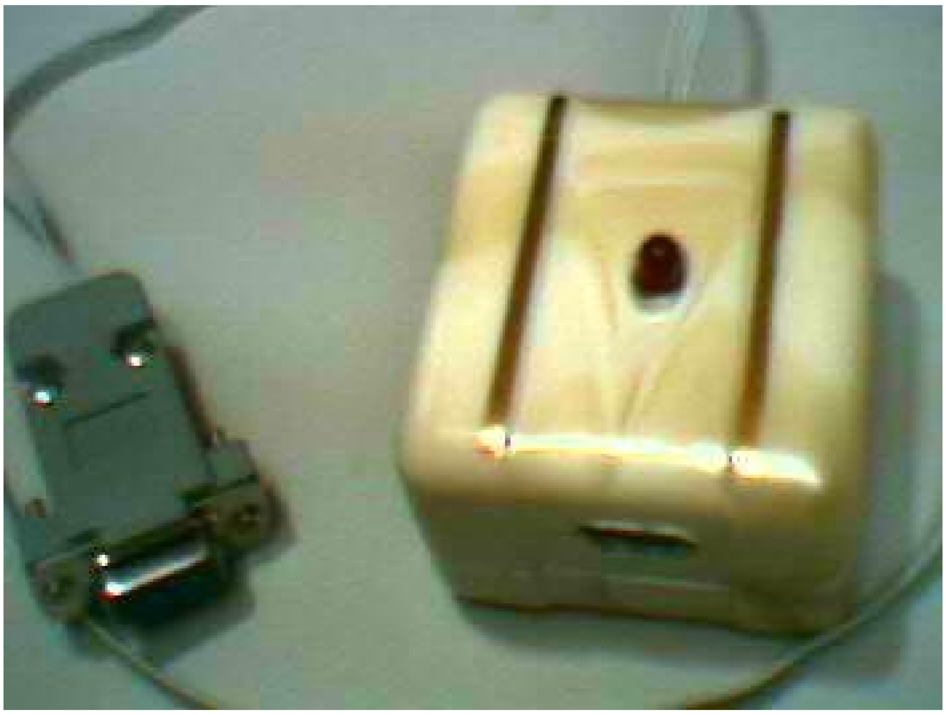
>> FIGURE 9 *Script to activate lirc xmms system*

NOTE

If you are using any serial devices (eg: modem), you must explicitly load the standard serial driver modules to make them work. You can use the same script to load the modules automatically during system boot. Two lines given at the middle of the script are used for the purpose.
Caution: You must not load the standard serial driver modules before loading lirc serial driver modules.



>> FIGURE 10 *Components soldered together*



>> FIGURE 11 *The lirc receiver*

MPLAYER can do it!

You can also control the most famous video player, mplayer, using the remote. All you have to do is to prepare a separate lircrc file for mplayer and have to run mplayer with gui support using *gmplayer -lircconf ~/.lircrc*. The parameters after *config=* in lircrc file must be specific to mplayer. Use /etc/mplayer/input.conf, which should be copied in to ~/.mplayer directory, for the options.
Google for a detailed procedure.

CONCLUSION

This simple project proves how flexible a GNU/Linux system is. Here you implemented the entire system at a low cost, spending not more than hundred rupees, using free tools to improve your day to day work without depending any third party solution. Call your friend who are addicted to a proprietary OS and show him/her what you have done. This can be the best way to convince other why they should use GNU/Linux.

COURTESY

To my lovable papa who helped in making the circuit (you know.. I may be the worst 'solder man' in this world), to my friend Arun Babu who co-operated with the work and to Pramode sir, my GNU/Linux Guru, who always inspires me with his thoughts and works. A big credit to Mr. Shuveb Hussain, Novatium Solutions, Chennai who presented this subject through one of his article.

by Tinku Sampath
Contact me at *tinkusam@gmail.com*