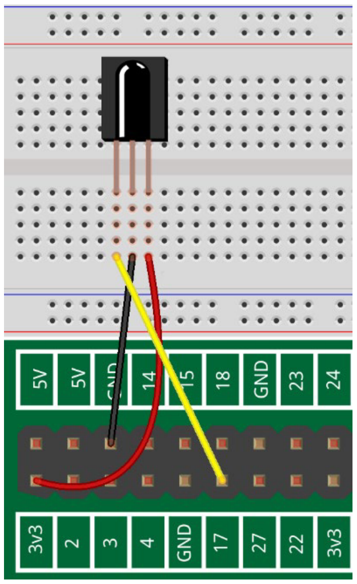
Infrared Receiver

These instructions are adapted from: https://www.instructables.com/id/Setup-IR-Remote-Control-Using-LIRC-for-the-Raspber/

This handout assumes the Raspberry Pi is running Raspbian Buster. Also a working knowledge of the terminal, basic commands from the command prompt and editing text files is assumed.

Wire the Infrared Receiver



Wire the infrared received as shown in the diagram. Note that with the raised lens area facing you, the pin on the left is the output and goes to pin 17 of the Raspberry Pi. The pin on the right is power and goes to 3v3. The center pin goes to ground.

Beware: This diagram is right for TSOP328xx and many other IR receivers. However, the TSOP329xx receivers exchange power and ground. Be sure you have the right receiver. Check the data sheet for what you have and revise the connections if necessary.

Install lirc

1) Open terminal window and install LIRC. Be forewarned that this will likely raise an error "Failed to start Flexible IR remote input/output application support" as the installed files have .dist appended and the suffix must be removed as shown below.

```
sudo apt-get update
sudo apt-get install lirc
```

Don't worry if you got the error message. Just rename the file as shown.

```
sudo mv /etc/lirc/lirc_options.conf.dist /etc/lirc/lirc_options.conf
```

2) Reinstall lirc now that the lirc_options.conf file has been renamed

```
sudo apt-get install lirc
```

This time there will be no error message.

Edit lirc_options.conf

Edit /etc/lirc/lirc_options.conf as follows:

```
sudo nano /etc/lirc/lirc_options.conf
```

You will need to change two lines.
Locate the line that says “driver” and change “devinput” to “default”
Locate the line that says “device” and change “auto” to “/dev/lirc0”

Press **ctrl-o** and **ctrl-x** to save and exit.

Remove .dist suffix from lircd.conf.dist

Remove suffix .dist from /etc/lirc/lircd.conf.dist

```
sudo mv /etc/lirc/lircd.conf.dist /etc/lirc/lircd.conf
```

Edit /boot/config.txt

Edit /boot/config.txt by adding one line in the lirc-rpi module section as follows. This example assumes the Raspberry Pi is 'listening' on pin 17 for the IR receiver but any gpio pin can be used.

```
sudo nano /boot/config.txt
```

At the end of the file, add the following line:

```
dtoverlay=gpio-ir,gpio_pin=17
```

Press **ctrl-o** and **ctrl-x** to save and exit.

Check status and reboot

Stop, start and check status of *lircd* to ensure there are no errors!

```
sudo systemctl stop lircd.service
sudo systemctl start lircd.service
sudo systemctl status lircd.service
```

Press the **q** key after the status message is displayed.

```
sudo reboot
```

Test remote

This step assumes you have an IR receiver hooked up to your Raspberry Pi on the pin specified in /boot/config.txt. You did this as the first step in this document.

1) Stop the *lircd* service and test remote using the *mode2* command

```
sudo systemctl stop lircd.service
sudo mode2 -d /dev/lirc0
```

An error of *Cannot initiate device /dev/lirc0* means you omitted the reboot.

3) Point the remote at the receiver and press some buttons. You should see something like this:

```
pulse 616
space 1626
pulse 619
space 1625
pulse 622
```

The numbers will be different from the example depending on which keys you press.

4) Press **ctrl-c** to exit

Set Up for Access with Python

Rename the configuration file.

```
cd /etc/lirc/lircd.conf.d
```

The following is one long line:

```
sudo mv /etc/lirc/lircd.conf.d/devinput.lircd.conf
/etc/lirc/lircd.conf.d/devinput.lircd.conf.copy
```

Download .conf File for Your Remote

Locate the configuration fine for your remote at http://lirc-remotes.sourceforge.net/remotes-table.html There are also configuration files here: http://lirc.sourceforge.net/remotes/

If you cannot find the exact file for your remote, try a similar one from the same manufacturer.

Still in the /etc/lirc/lircd/conf.d directory, create a new file with the nano editor and copy/paste the contents of the configuration file you selected. The example uses `myremote`, but you may name the file anything you like. It must end in `.conf`

```
sudo nano myremote.conf
```

Paste your file into the editor window, then **ctrl-x** to exit. Reply **y** to the "Save buffer?" question.

You must stop and restart the *lircd* service any time you change a .conf file:

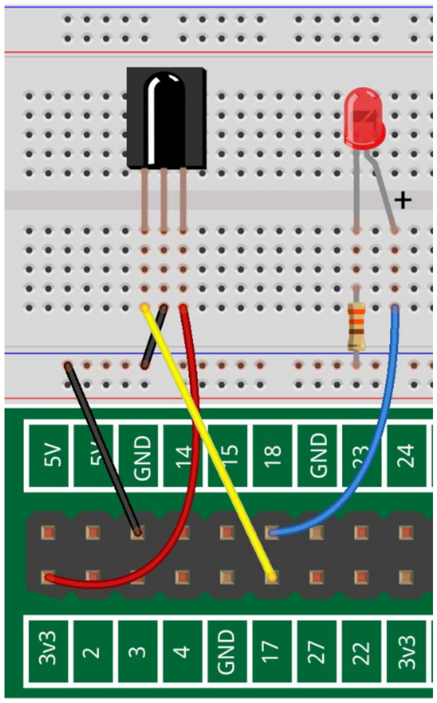
```
sudo systemctl stop lircd.service
sudo systemctl start lircd.service
```

Python Test Code

Type the following program into the Thonny program window. Save and run it.

```
# https://www.instructables.com/id/Easy-Setup-IR-Remote-
Control-Using-LIRC-for-the-Ra/

from lirc import RawConnection
conn = RawConnection()

def ProcessIRRemote():

#get IR command
#keypress format=(hexcode, repeat_num, command_key, remote_id)
    try:
        keypress = conn.readline(.0001)
    except:
        keypress=""
    if (keypress != "" and keypress != None):
        data = keypress.split()
        sequence = data[1]
        command = data[2]
        #ignore command repeats
        if (sequence != "00"):
           return
        print(command)

print("Starting Up...")
while True:
    ProcessIRRemote()
```

Controlling Devices with Your Remote and Python



In the program listing above, note the line that says `print(command)`. At that point in the program, the command has been decoded and can be used in an IF statement. When you run the code below, pressing a button on the remote causes the program to print the key name, like KEY_1.

You can insert code into the program to take action when particular leys are pressed. Here is an example.

In the diagram at the right, an LED with 330Ω resistor has been added, and the blue rail is connected to GND on the Raspberry Pi. The center pin of the IR receiver is connected to the blue rail with an M/M jumper wire.

The program on the next page uses the *Power* button on the remote to control the LED. Additions to the program are shown in red. The `redLED.off()` line assures that the LED is off when the program starts.

After the command is printed, an IF statement checks whether the KEY_POWER key was pressed. If so, `redLED.toggle()` is called, which turns the LED on if it was off, and off if it was on. The other commands are printed, but to not initiate any action.

You can "hook" any command that your remote can generate and cause it to take any action that can be controlled by the Python program. When your program is working reliably, you can consider commenting out or removing the `print` statement.

KENNESAW STATE UNIVERSITY

```
# https://www.instructables.com/id/Easy-Setup-IR-Remote-
Control-Using-LIRC-for-the-Ra/

from lirc import RawConnection
from gpiozero import LED

redLED = LED(18)
redLED.off()
conn = RawConnection()

def ProcessIRRemote():

#get IR command
#keypress format=(hexcode, repeat_num, command_key, remote_id)
    try:
        keypress = conn.readline(.0001)
    except:
        keypress=""
    if (keypress != "" and keypress != None):
        data = keypress.split()
        sequence = data[1]
        command = data[2]
        #ignore command repeats
        if (sequence != "00"):
           return
        print(command)
        if command == "KEY_POWER":
            redLED.toggle()

print("Starting Up...")
while True:
      ProcessIRRemote()
```