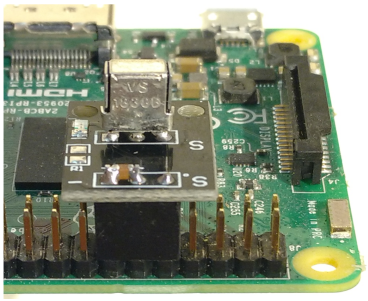
Seattle Raspberry Jam

A Raspberry Pi Meetup for Beginners to Experts

RASPBERRY JAM

Hardware Project #5: IR Remote

Assembly



Plug the IR receiver module into the Raspberry Pi's pin header on pins 10, 12 and 14 as shown in the picture to the right. **WARNING! Incorrectly installing the module could damage the Raspberry Pi and the IR receiver. Please have one of the organizers check that it is plugged in correctly before applying power.**

Preliminary Software Setup

First of all, we need to make some changes to */boot/config.txt* to power the module and setup the driver. Open this file in a text editor with the following command:

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

And add the following lines to the bottom of the file:

```
gpio=18=op,dh
dtoverlay=gpio-ir,gpio_pin=15,gpio_pull=off
```

The first line instructs the Raspberry Pi to set GPIO 18 (the VCC pin on the IR module) as an output and to drive it high. The second line sets up the gpio-ir driver to use GPIO 15 (the S pin on the module) as an input with no pull. Save and close the file with CTRL-X, then type Y and finally press Enter. Now run the following command to reboot the Raspberry Pi:

```
sudo reboot
```

Once the Raspberry Pi is back up and running, continue on to the next section.

How to Create a Keycode Table?

If you are not using the same remote as we are, you will need to create your own keymap. First, we need to decifer the protocol your remote is using. To do this, type `ir-keytable` and press Enter. Then note down all of the protocols after "Supported Protocols:." Type in `ir-keytable -p PROTOCOL -t`, substituting in one of the protocols from earlier. If text appears on the screen when you press the keys on the remote, then you have found your protocol! Otherwise, try the next one. After that, you need to figure out which codes your remote is sending for each key. Run the same `ir-keytable -p PROTOCOL -t`, and note down the text after "scancode" for each key. Next, use the reference at http://bit.ly/2XNEYGL to figure out the names of each key (for example, the up arrow would map to KEY_UP). Open up the *custom_remote* keymap file as described in the *Adding a Keycode Table* section and enter the same text as supplied, but replace the codes and KEYs with the ones you found. Finally, change the text after "type:" to the protocol you found earlier. (Note that for our example code to run, one of the keys will need to be linked to KEY_POWER)

Adding a Keycode Table

Now we need to create a keycode table file for the remote. This keycode table will be used to map hexadecimal codes from the receiver into easier-to-understand key names. For the specific remote we are using, we have already created a keymap. If you are using a different remote, see the "How to Create a Keycode Table?" box. Create a new file and open it with the following command:

```
sudo nano /etc/rc_keymaps/custom_remote
```

And add the following text (you can also download this file from www.github.com/polarpiberry/Seattle-Raspberry-Jam/tree/master/Hardware%20Project%20%235):

```
# table custom_remote, type: NEC
0x19 KEY_0
0x45 KEY_1
0x46 KEY_2
0x47 KEY_3
0x44 KEY_4
0x40 KEY_5
0x43 KEY_6
0x07 KEY_7
0x15 KEY_8
0x09 KEY_9
0x08 KEY_LEFT
0x5a KEY_RIGHT
0x18 KEY_UP
0x52 KEY_DOWN
0x1c KEY_OK
0x16 KEY_NUMERIC_ASTERISK
0x0d KEY_NUMERIC_POUND
```

The first line sets the table name and the protocol we are using (for our remote, this is NEC protocl, which is the most common protocol for this type of remote).

Using IR-Keytable

We now need to install ir-keytable, which is a handy Linux tool that can read and set various parameters for IR receivers, such as keycode tables.

```
sudo apt-get install ir-keytable
```

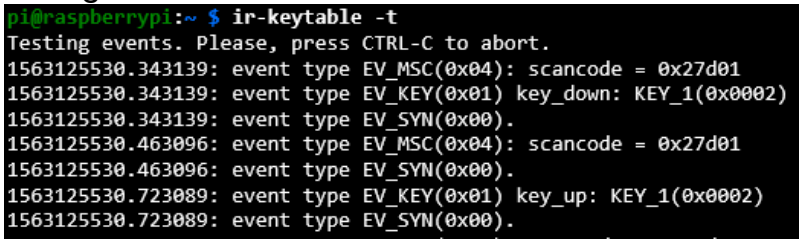
We can now run the following command to load our custom keycode table.

```
sudo ir-keytable -p NEC -c -w /etc/rc_keymaps/custom_remote
```

Essentially, this sets the current keymap to our *custom_remote* file, and changes the protocol to NEC. To test if our setup is correct, run the following command:

```
ir-keytable -t
```

Press some of the buttons on the remote; text similar to what is in the picture to the right should appear on the screen with each press. If it does not, make sure that all of the other steps ran successfully.



Loading on Boot

We can make the ir-keytable run on boot to save us from having to run this command every time the Raspberry Pi turns on. One way to do this is to add code to */etc/rc.local,* which is a shell script that runs on startup. Open this file with the following command:

```
sudo nano /etc/rc.local
```

Add the following line of code to the bottom of the file, just above the exit 0 statement. (the same as before):

```
sudo ir-keytable -p NEC -c -w /etc/rc_keymaps/custom_remote
```

Some Code

Finally, we can use some code to watch for key presses and shutdown the Raspberry Pi if the power key is pressed. Download the code from www.github.com/polarpiberry/Seattle-Raspberry-Jam/tree/master/Hardware%20Project%20%235. Before we can run this code, we need to compile it. Enter the following command:

```
g++ ir_recv.cpp -o ir_recv
```

This will compile the code into an executable called *ir_recv* using g++, a C++ compiler. Now run the code with the following command:

```
./ir_recv
```

If all is well, text should appear on the screen when you press the keys on the remote. Do not be confused by the numbers and letters that are printed after "KEY DOWN:," as they are hexadecimal key codes and not the actual characters that the keys represent. However, if text does not show up, it could mean that the IR receiver is connected to a different input device (the default in our code is 0). If this is the case, continue on to the next section.

If the Code Failed to Run...

First, check which input device your IR receiver is on:

```
ls -l /dev/input/by-path
```

Look for any reference to "ir" in the resulting text and compare to the picture below. If the part of the "ir" line after the "→" is different to the picture below, then the *ir_recv.cpp* code needs to be

```
pi@raspberrypi:~ $ ls -l /dev/input/by-path
total 0
lrwxrwxrwx 1 root root 9 Jul 14 18:13 platform-3f980000.usb-usb-0:1:1.2-event-mouse -> ../event2
lrwxrwxrwx 1 root root 9 Jul 14 18:13 platform-3f980000.usb-usb-0:1:1.2-mouse -> ../mouse1
lrwxrwxrwx 1 root root 9 Jul 14 18:13 platform-ir-receiver@1b-event -> ../event0
```

changed. First, note down the number in the "event" part. (e.g. our number would be 0). Next, open up the code file with the nano text editor:

```
nano ir_recv.cpp
```

Then find the line of code pictured below (near the top of the file) and change the "event0" to "eventN" (where N is the number you noted earlier).

```
int fd = open("/dev/input/event0", O_RDONLY);
```

Now compile and run the code as before, and hopefully this time it will work. Congratulations! You have successfully attached an IR remote to your Raspberry Pi. If you feel up to it, go ahead and try the challenge below.

Challenge: Change the shutting down message and try to add functions for the other keys. If you are feeling very brave, you can try to implement a function for the key-released and key-still-pressed events. Also, you could try using a different remote.