# Table of Contents

**Overview** 3
- Parts Required

**Account Setup** 5
- Ustream
- Google+

**Hardware Setup** 9
- Connecting the Pieces

**Software Setup** 12
If you run a recurring video broadcast on Ustream or Google+ Hangouts on Air, or if you have favorites you like to watch fervently, you might enjoy or have practical use for this lighted “On Air” sign. It’s inspired by the studio warning lights used in radio broadcast and TV production, with an internet-connected twist.

Not just for visual flair, these signs also have an important use: they tell others when it’s time to keep quiet, and not to interrupt, barge in the room or hog all the bandwidth watching cat videos.

This “connected” sign is foolproof in a way that a manually-switched sign is not. You won’t leave it on (or off) accidentally. By polling the streaming server(s) directly, the status is always correct with respect to delays, early starts, vacation breaks, etc. The broadcaster and sign can be separated by a wall or by half the planet.
Parts Required

- Raspberry Pi computer (http://adafru.it/998) (any model) and associated hardware:
  - Power supply (http://adafru.it/501) and USB MicroB cable (http://adafru.it/592)
  - SD card (http://adafru.it/102) (4GB or larger)
  - USB WiFi adapter (http://adafru.it/814) or Ethernet network connection
  - Case (http://adafru.it/1140)
  - Keyboard and monitor for setup (can be run "headless" once complete)

- PowerSwitch Tail II (http://adafru.it/268) appliance controller
- Mains-powered lighted sign
- Jumper extension wires (http://adafru.it/824)

There are some classy professional studio warning lights available out there, but these do not come cheap — some get into the hundreds of dollars! Really most any lamp or lighted sign will do, or can be adapted if you're craft-inclined. Our example sign is an inexpensive lighted house number box, a 12V DC adapter and some 2" adhesive vinyl letters from an art store. The more technically-minded might forego the PowerSwitch Tail and assemble something from LPD8806 LED strips (http://adafru.it/306), controlled by your own code on the Raspberry Pi.

The PowerSwitch Tail II uses a relay for switching and is therefore compatible with all lighting types — incandescent, LED or fluorescent/CFL.
Account Setup

**THIS GUIDE IS INACTIVE. Changes to streaming services will require updated instructions which we don’t have yet.**

One does not simply walk into Ustream or Google+ and start making demands. In order to monitor and regulate the load on their servers, both systems require developer credentials before one can perform actions like polling online statuses. Even if you’re not broadcasting your own content, just monitoring others’, you still need to set up accounts. This is the least fun part, so let’s get it out of the way first…

**Ustream**

Begin at [developer.ustream.tv](http://developer.ustream.tv) and sign up for an account:

![Ustream Developer Network](image)

Once you have a developer account, [apply for an API key](http://developer.ustream.tv)

The key is a sequence of letters and numbers that accompanies each request to the server, uniquely identifying its origin. Later, we’ll plug this key into our software.

You can refer to your Ustream Data API Key on the “My Account” page:
The key has been blocked out here for privacy. Don’t share your key around. If you keep code on Github or other public code repositories, always remove the key before synchronizing code with the server, otherwise you could be blamed for others’ abuse of the service.

Finally, you need the name of the account you want to monitor. Normally this can be found at the end of the channel URL, for example:

http://ustream.tv/channel/adafruit-industries

(Since becoming an affiliate channel, Adafruit’s URL is shortened to just ustream.tv/adafruit, but the channel name remains the original adafruit-industries.) Whatever channel you’ll be monitoring, we’ll plug the name into our software later.

Google+

If you don’t already have one, register for an account with Google at accounts.google.com.
Then visit developers.google.com and sign in. You may need to agree to additional terms of use before proceeding.

Visit the APIs Console at code.google.com/apis/console, then select “Create project...” and enter a name for the project.

After the project is named and created, go to the Services pane (second item in the left column, below the project name):

Scroll down the page a bit to find the Google+ API switch, and turn it on (not the Google+ Hangouts API, unless you’re also using that for other projects):
Now select the API Access pane from the left column. This page should show your unique developer API key. Later, we'll plug this key into our software:

As with the Ustream key before, the Google developer key has been blocked out here for privacy. Don’t share your key around. If you keep code on Github or other public code repositories, always remove the key before synchronizing code with the server, otherwise you could be blamed for others’ abuse of the service.

Finally, you need the name of the account you want to monitor. This is a long number that can be found in the URL bar when visiting the Google+ profile page for the account you want to watch:

For example, the Adafruit account ID is 112526208786662512291. Later, we'll plug this into our software.
Hardware Setup

If starting from scratch, you'll begin by loading up the Linux operating system on a 4GB or larger SD card. We'll not go into lengthy detail on this, as it's already covered well in other tutorials:

- [Preparing an SD card for your Raspberry Pi](#) (note: Mac users now have an easier option in [RPi-sd card builder](#))
- [First time configuration](#)
- [Network setup](#)

For this project, we recommend using the latest Raspbian “Wheezy” distribution. Also, when configuring the system it’s very important to select the proper time zone — this project is time-sensitive.

Before continuing, get your Raspberry Pi to a state where it’s booting, connected to the local network (either wired or wireless) and reliably communicating with the outside internet.

Connecting the Pieces

Plug your sign/light into a wall socket and confirm that it works.

The Raspberry Pi at this point is assumed powered and networked; we'll leave those elements out of the remaining explanation.
Plug “On Air” sign (or other indicator) into PowerSwitch tail. If your light has an on/off switch, leave it in the “on” position. Depending on the type of lamp used, there might be an AC adapter brick involved; this isn’t shown in the diagram, but would plug into the PowerSwitch Tail the same.
Connect Pin 1 on the PowerSwitch Tail (labeled “+in”) to GPIO 24 on the Raspberry Pi GPIO header. Connect Pin 2 (“-in”) to the adjacent ground pin on the GPIO header. Pin 3 (Ground) isn’t used. You don’t necessarily need a Pi Plate (http://adafru.it/801) or Cobbler (http://adafru.it/914), though those may make wiring easier depending what you have on-hand. A pair of female-to-male extension jumper wires (http://adafru.it/824) may be all you need.

The two pins on the header are outlined in red here:

Any of the GPIO pins labeled in green can be used instead of #24, just be sure to update the code to match the GPIO selection. Likewise, any other GND pin may be used.

Finally, plug the PowerSwitch tail into a wall outlet or power strip. Nothing should happen yet; that will require software. But if you’d like to test it, try moving the GPIO 24 wire to either a 3.3V or 5V pin. The light should come on.

At first thought it might seem “cleaner” to hide the Raspberry Pi inside your lighting fixture, but this may actually create an awkward jumble of wires. The Pi needs to remain powered on, it can’t be switched with the light. Merging the two would likely involve either a lot of cabling inside the box, or wires leading in and out for the Pi, PowerSwitch Tail, etc.

We felt it was easier just to keep the Pi outside the box, down by the outlet along with the PowerSwitch Tail, then run a single slim power cord to the light:
Software Setup

THIS GUIDE IS INACTIVE. Changes to streaming services will require updated instructions which we don’t have yet.
The “On Air” software can be downloaded from Github, or just copy and paste this line in a terminal window to download the single file that’s needed:

```
```

No additional packages should need to be installed — the current release of Raspbian Wheezy has all prerequisites installed.

Edit the file “onair.py” to set up your developer account keys and a polling schedule:

```
nano onair.py
```

Look for this section near the beginning of the code:

```
# Ustream settings -----------------------------------------------
# 'uKey' is your Developer API key (request one at developer.ustream.tv).
# 'uChannel' is Ustream channel to monitor.
uKey     =  'PUT_USTREAM_DEVELOPER_API_KEY_HERE'
uChannel =  'adafruit-industries'

# Google+ settings -----------------------------------------------
# 'gKey' is your API key from the Google APIs API Access page (need to switch
# on G+ on API Services page first). 'gId' is the account ID to monitor (can
# find this in the URL of your profile page, mild nuisance but ID is used
# because user name is not guaranteed unique.
gKey =  'PUT_GOOGLE_API_KEY_HERE'
gId  =  '112526208786662512291' # Adafruit account ID
```

uKey and gKey are your developer API keys that we previously set up on Ustream and Google+, respectively. uChannel and gld are the Ustream channel name and the Google+ account ID to monitor, respectively. These steps were explained on the Account Setup page.

A little further down, you’ll see a table like this:

```
times = [
  ("06:00",  60),  # 6am, office hours starting, poll once per minute
  ("21:25",  10),  # 9:25pm, Show & Tell starting soon, poll 6X/minute
  ("21:35",  30),  # S&T underway, reduce polling to 2X per minute
  ("21:55",  10),  # 9:55pm, AAE starting soon, poll 6X/minute again
  ("22:05",  30),  # AAE underway, slow polling to 2X per minute
  ("23:10",  60),  # AAE over (plus extra), return to once per minute
  ("00:00", 900) ] # After midnight, gone home, poll every 15 minutes
```

The developer APIs permit a limited number of queries per day. In order to provide good response times without blowing past those limits, the software allows you to set up different time periods and polling frequencies within those periods. For example, you might want to poll only once per minute most of the time, but then go into a very fast polling mode (every 5 or 10 seconds) for short periods when you know streams
are likely to begin or end, giving the appearance of a much more responsive, interactive system.

Each item in the table consists of two values, surrounded by parenthesis. The first, a string (in quotes), is the start of a time period, expressed in 24-hour local time (e.g. "20:15" is 8:15 pm in your time zone, which may be different than the broadcaster's time zone). The second is a number (no quotes) indicating the polling frequency, in seconds (e.g. 60 = once per minute).

Once everything's configured to your liking, try running the software manually:

```
sudo python onair.py
```

The light still probably won’t do anything, unless the monitored channel(s) are streaming at the time. This is mostly an opportunity to check for any error messages. Syntactical errors (missing quotes, etc.) will usually prevent the script from running at all. Or if you see an error message like this:

```
Error: : ('items',)
```

This probably means your developer key or channel ID are missing or improperly formatted.

Once the software is running smoothly without errors, we’ll configure the system to run the script automatically at startup. To abort the program in the terminal window, you’ll need to tap Control+C a few times…it’s a little funny that way.

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

Before the final "exit 0" line, insert these two lines:

```
cd /home/pi
python onair.py &
```

If you placed the onair.py script in a different location, the first line should be changed accordingly. "sudo" isn’t necessary here because the rc.local script is already run as root.

Reboot the system to test the startup function:

```
sudo reboot
```

After rebooting, log in and type:
You should see some numbers and a mention of python. That's good, the code is running. Keyboard and monitor are no longer needed.

Left to its own devices, the light should now come on when either the Ustream or Google+ accounts are broadcasting.