**Use Raspberry Pi to Measure Broadband Speeds**

**to Hold Your ISP Accountable**

**Using three different methods –**

**Text file, CSV file and IFTTT/Google Doc**

First things first -

Connect the monitor, mouse, and keyboard to the Pi

Connect power to the Pi

• Wait until the Pi is completes its bootup

* The first time boot of the Pi may take up to 5 minutes to complete (the newest version of the Pi will boot quicker than older versions)
* Subsequent boots should complete in about a minute

Update software on the Pi

sudo apt-get update

sudo apt-get dist-upgrade

This makes sure that your Pi is up-to-date and less likely to cause you problems.

We’ll be using [speedtest-cli](https://github.com/sivel/speedtest-cli), a command line interface to the [speedtest.net](http://www.speedtest.net/) servers written in Python. It’s really easy to install. Just open up a terminal window on your Raspberry Pi and type the following at the command line -

**$** sudo apt-get install python-pip

**$** sudo pip install speedtest-cli

This will install pip — a package management system for Python — if you don’t already have it installed, and then the speedtest-cli package from the pip repositories.

Once installed it’s rather easy to grab and measure your broadband speed.

**$** speedtest-cli

Retrieving speedtest.net configuration...

Retrieving speedtest.net server list...

Testing from Windstream (xxx.xxx.xxx.xxx)...

Selecting best server based on latency...

Hosted by MI CONNECTION (Mooresville, NC) [38.91 km]: 28.271 ms

Testing download speed........................................

Download: 12.99 Mbit/s

Testing upload speed..................................................

Upload: 0.83 Mbit/s

**$**

Now that we have our command line tool installed, we can run it automatically using cron, which allows you to schedule commands to run at specified times (every hour), and log the output to a file.

The easiest way to do this is to create a quick script, let’s call it speedtest-cron.sh (use whatever text editor you are comfortable using), which will log the date and the output of the test to a file -

#!/bin/bash

date >> /home/pi/speedtest.log

/usr/local/bin/speedtest --simple >> /home/pi/speedtest.log

Then go ahead and edit your crontab file, adding an entry to run the script test once an hour. You can do that by typing -

**$** crontab -e

at the command line, and then adding the line at the end of the file -

0 \* \* \* \* /home/pi/speedtest-cron.sh

before saving it. This will run the test once an hour, at the top of the hour, appending the date-stamped output of the speedtest command to a log file.

**$** cat speedtest.log

Sun Jan 31 19:49:01 GMT 2016

Ping: 50.085 ms

Download: 12.75 Mbit/s

Upload: 0.83 Mbit/s

**$**

However the output of the speedtest-cli package, even in its “simple” mode, is pretty messy. We could go into the package and fix things so that the output is somewhat more useful, CSV format perhaps, or we could [rewrite the whole thing in Perl](https://github.com/Tux/speedtest). But since this is a quick hack, it’s probably easiest just to fix things with a bash script.

Which is exactly what the [speedtest-cli-extras](https://github.com/HenrikBengtsson/speedtest-cli-extras) script does. It captures the output of the script, reformats it, and outputs it on a single line with time stamps and values separated by semicolons -

**$** git clone https://github.com/HenrikBengtsson/speedtest-cli-extras.git

**$** cd speedtest-cli-extras/bin

**$** ./speedtest-csv

2017-04-26 00:34:20;2017-04-26 00:35:03;Windstream;xxx.xxx.xxx.xxx;MI CONNECTION (Mooresville, NC);38.91 km;28.632 ms;13.00 Mbit/s;0.83 Mbit/s;http://www.speedtest.net/result/6248527133.png

**$**

As you can see the output is much more useful, especially if we want to create graphs, than we had previously. Running this every hour from cron is going to start to pile up evidence in a nicely formatted data file.

These two files in different formats are OK but you either have to be at home with the Pi to view the info from the cron jobs or you have to SSH into the Raspberry Pi connected to modem/network. Better yet, you can use IFTTT along with a Google Sheets on a Google Drive and view it from anywhere! IFTTT allows you to do many different things automatically. To do this you need to have an IFTTT account and a Google Drive account (can be the same as you account for Gmail).

If you don’t have a Google account then go to <https://drive.google.com> and create one. If you do have an account then log into it. Next go to IFTTT.com (<https://ifttt.com/>) and create your account there. Once you have an account click on you account name in the top right-hand corner and select Services. From there click on the Google Drive square. From there click on Settings in the top right-hand side. From there you can enter the info to tie the Google Drive to your IFTT account so that you can send info to it from IFTT.

Now this is where [IFTTT](https://ifttt.com/)‘s Maker Channel, [introduced towards the middle of last year](http://makezine.com/2015/06/26/ifttt-adds-new-channel-makers/), comes in handy. Alasdair Allen has [created a recipe on IFTTT](https://ifttt.com/recipes/379108-log-speedtest-results-to-spreadsheet) (<https://ifttt.com/applets/379108p-log-speedtest-results-to-spreadsheet>) to take the data passed to a Maker Channel event called “speedtest” and automatically fill a Google Sheet with the output of speedtest-cli script.

The easiest way to get the data from our Raspberry Pi to IFTTT at this point is to modify the speedtest-cli-extras script. Instead of printing out the output to a log file, we’ll make a POST web request with the event name and our secret key — the key is assigned when you connect the channel — of the form -

https://maker.ifttt.com/trigger/speedtest/with/key/**{secret\_key}**

with a JSON body consisting of  three values — the latency, download, and upload speeds — to be passed on to the action in the recipe.

After you [download](https://gist.githubusercontent.com/aallan/bafc70a347f3b9526d30/raw/b9760af4b29fad0335c20a8a24607a52bba0cb6f/speedtest-ifttt.sh) (<https://gist.githubusercontent.com/aallan/bafc70a347f3b9526d30/raw/b9760af4b29fad0335c20a8a24607a52bba0cb6f/speedtest-ifttt.sh>) it you should go ahead and substitute your own **SECRET\_KEY** ( Get this by Clicking on your account name, then Services, then the Maker Webhooks tile and then Settings ) before testing it out by running the modified script manually at the command line. If all goes well you should see something like this -

**$** ./speedtest-ifttt.sh

Congratulations! You've fired the speedtest event

**$**

Go ahead and check your Google Drive, there should be a new Sheet called “Speedtest.”

This Sheet should have a single row with four columns populated. The first is the date stamp on the IFTTT Maker Channel event, the second is the ping time in ms (latency of the connection), the third the download speed in Mbit/s, with the final column being the upload speed in Mbit/s.

|  |  |  |  |
| --- | --- | --- | --- |
| January 26, 2016 at 02:49AM | 34.039 | 12.56 | 1.32 |

If that’s worked, all we have to do is modify our speedtest-cron.sh script to run our new script when it’s called at the top of the hour by cron,

#!/bin/bash

date >> /home/pi/speedtest.log

/home/pi/speedtest-ifttt.sh >> /home/pi/speedtest.log

echo "" >> /home/pi/speedtest.log

and each hour we’ll now add another row to the Google Sheet.

At this point you now have a Raspberry Pi-based monitoring system in place to measure your broadband speed once an hour, and automatically log it to the cloud.

I am currently doing three different versions of this –

On the hour the first version

At twenty after the hour the IFTTT version

At forty after the hour the CSV version (just started this week again)

Using these in the crontab

0 \* \* \* \* /home/pi/David/speedtest-cron.sh

20 \* \* \* \* /home/pi/speedtest-ifttt-cron.sh

40 \* \* \* \* /home/pi/David/speedtest-csv-cron.sh

While getting this ready I have looked at the perl version of this using Loggly and the syslog on the Pi. Looks neat and you can do graphs and such with it fairly easily.