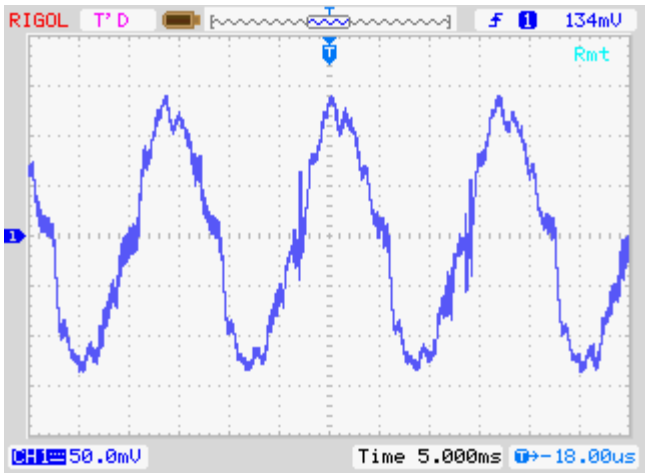
We Saw a Chicken …

COMPUTERS SUCK

MY RASPBERRY PI TALKS TO MY OSCILLOSCOPE

02013-12-15 | SCRUSS | 3 COMMENTS

… it complains that the oscilloscope is always making waves.



Ahem. Anyway. I have a Rigol DS1102E 100 MHz Digital Oscilloscope. For such a cheap device, it's remarkable that you can control it using USB Test & Measurement Class commands. I'd been wanting to use a Raspberry Pi as a headless data acquisition box with the oscilloscope for a while, but Raspbian doesn't ship with the usbtmc kernel module. I thought I was stuck.

Alex Forencich turned up in the forum with an all-Python solution: Python USBTMC (source: alex-forencich / python-usbtmc). I got this working quite nicely today on both the Raspberry Pi and my Ubuntu laptop. Here's how I installed it:

1. Check your device's USB code with lsusb:
   $ lsusb
   Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
   ….
   Bus 001 Device 004: ID 1ab1:0588 Rigol Technologies DS1000 SERIES
2. Ensure that libusb-1.0 is installed:
   sudo apt-get install libusb-1.0-0
3. Create a new group, usbtmc:

sudo groupadd usbtmc

4. Add yourself to this group:
   sudo usermod -a -G usbtmc pi
5. As root, create a file /etc/udev/rules.d/usbtmc.rules. You'll need to put in your device's ID values:
   # USBTMC instruments
   # Rigol DS1100 – ID 1ab1:0588 Rigol Technologies DS1000 SERIES
   SUBSYSTEMS==”usb”, ACTION==”add”, ATTRS{idVendor}==”1ab1″, ATTRS{idProduct}==”0588″, GROUP=”usbtmc”, MODE=”0660″
   (all of the *SUBSYSTEMS* to *MODE=* should be one one line)
6. Download and install the latest pyusb (Raspbian version is rather old):
   git clone https://github.com/walac/pyusb.git
   cd pyusb
   python setup.py build
   sudo python setup.py install
7. Now get python-usbtmc:
   git clone https://github.com/alexforencich/python-usbtmc.git
   cd python-usbtmc
   python setup.py build
   sudo python setup.py install
8. For this simple demo, you'll need to convert the USB vendor IDs to decimal:
   0x1ab1 = 6833
   0×0588 = 1416
9. Now, start python as root (sudo python) then type:
   import usbtmc
   instr = usbtmc.Instrument(6833, 1416)
   print(instr.ask(“*IDN?”))
10. This *should* return something like:
    Rigol Technologies,DS1102E,DS1EB13490xxxx,00.02.06.00.01

If you get the status line, congratulations! You now have a fully working usbtmc link. I haven't had much time to play with this, but I know I can make really nice screenshots to an attached USB drive using the command: instr.write(“:HARDcopy”). Many more commands can be found in the **DS1000D/E Programming Guide**, available on Rigol's site.

I had a couple of problems, though:

1. ~~The library seems to need root privileges, despite the udev rule thing~~. After creating the udev rule, you will need to reboot. This is the simplest way of getting it to work without being root.
2. ~~Reading from the 'scope's memory chokes on non-UTF8 characters. If I do:~~
   ~~rawdata = instr.ask(“:WAV:DATA? CHAN1″)[10:]~~
   ~~I get a lengthy Python error which ends:~~

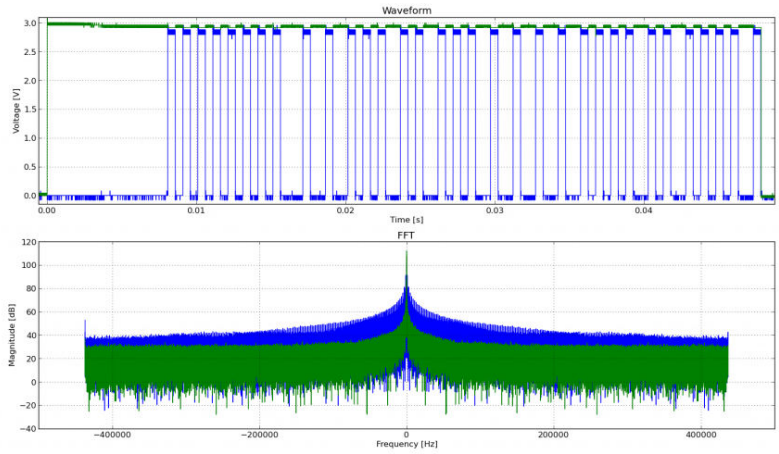
…
~~File "/usr/lib/python2.7/encodings/utf_8.py", line 16, in decode~~
~~return codecs.utf_8_decode(input, errors, True)~~
~~UnicodeDecodeError: 'utf8' codec can't decode byte 0×99 in position 10: invalid start byte~~
~~I have no idea what that means, or how to fix it~~. Alex suggested using ask_raw instead of ask, and the data comes through with no complaints.

I've still got to work my way through the Rigol's data format, but other people have done that before:

1. Controlling a Rigol oscilloscope using Linux and Python | C i b o M a h t o . c o m
2. Ken Shirriff's blog: Four Rigol oscilloscope hacks with Python

I'll post any updates here, along with the Raspberry Pi forum topic: USB Test & Measurement class (usbtmc) driver?

Incidentally, if you're working with WFM data dumps from the Rigol 'scopes (and you should, because they make storing data to USB drives quick), mabl/pyRigolWFM is basically magic. Not merely can it describe and decode those binary files, it can do pretty graphics with no thought required:



Hat tip for the mention: MP3 Options & Oscilloscope Interfacing For Raspberry Pi @Raspberry_Pi #piday #raspberrypi « adafruit industries blog

Update, 2013-12-20: I've successfully managed to run most of Ken's examples with Alex's code. The major modification you have to do is use ask_raw instead of ask. Example code shown below:

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3
4 """
5 Download data from a Rigol DS1102E oscilloscope and graph with matplotl
6         using  Alex Forencich's python-usbtmc pure python driver
```

```
 7
 8  https://github.com/alexforencich/python-usbtmc
 9
10  scruss - 2013-12-20
11
12  based on
13  Download data from a Rigol DS1052E oscilloscope and graph with matplotl
14  By Ken Shirriff, http://righto.com/rigol
15
16  which in turn was
17  Based on http://www.cibomahto.com/2010/04/controlling-a-rigol-oscillosc
18  by Cibo Mahto.
19  """
20
21  import usbtmc
22  import time
23  import numpy
24  import matplotlib.pyplot as plot
25
26  # initialise device
27  instr =  usbtmc.Instrument(0x1ab1, 0x0588) # Rigol DS1102E
28
29  # read data
30  instr.write(":STOP")
31  instr.write(":WAV:POIN:MODE RAW")
32  # first ten bytes are header, so skip
33  rawdata = instr.ask_raw(":WAV:DATA? CHAN1")[10:]
34  data_size = len(rawdata)
35
36  # get metadata
37  sample_rate = float(instr.ask_raw(':ACQ:SAMP?'))
38  timescale = float(instr.ask_raw(":TIM:SCAL?"))
39  timeoffset = float(instr.ask_raw(":TIM:OFFS?"))
40  voltscale = float(instr.ask_raw(':CHAN1:SCAL?'))
41  voltoffset = float(instr.ask_raw(":CHAN1:OFFS?"))
42
43  # show metadata
44  print "Data size:      ", data_size
45  print "Sample rate:    ", sample_rate
46  print "Time scale:     ", timescale
47  print "Time offset:    ", timeoffset
48  print "Voltage offset: ", voltoffset
49  print "Voltage scale:  ", voltscale
50
51  # convert data from (inverted) bytes to an array of scaled floats
52  # this magic from Matthew Mets
53  data = numpy.frombuffer(rawdata, 'B')
54  data = data * -1 + 255
55  data = (data - 130.0 - voltoffset/voltscale*25) / 25 * voltscale
56
57  # creat array of matching timestamps
58  time = numpy.linspace(timeoffset - 6 * timescale, timeoffset + 6 * time
59                        num=len(data))
60
61  # scale time series and label accordingly
62  if (time[-1] < 1e-3):
63      time = time * 1e6
64      tUnit = "µS"
```

```
65  elif (time[-1] < 1):
66      time = time * 1e3
67      tUnit = "mS"
68  else:
69      tUnit = "S"
70
71  # Plot the data
72  plot.plot(time, data)
73  plot.title("Oscilloscope Channel 1")
74  plot.ylabel("Voltage (V)")
75  plot.xlabel("Time (" + tUnit + ")")
76  plot.xlim(time[0], time[-1])
77  plot.show()
```

Send the author to the moon!

No related posts.

DEBIAN OSCILLOSCOPE PYTHON RASPBERRYPI RASPBIAN RIGOL USB USBTMC

3 THOUGHTS ON “MY RASPBERRY PI TALKS TO MY OSCILLOSCOPE”

Mike Alport

02013-12-16 AT 13:58

Thanks very much. More than enough to work on.
If anyone is interested getting USBTMC to work on a Mac, to find the vendor USB code, since OSX doesn’t have a lsusb command, you can use Apple info:
Click the apple in the top left corner — > Choose “About This Mac” — > Click “More Info” button –> Click on the “System Report” button –> Choose Hardware and USB option under it to list all USB devices connected.

T-Rex

02014-05-29 AT 15:37

For the Mac, there is no need for all the mouse interaction. Use command-line

alias lsusb='system_profiler SPUSBDataType'

★ scruss

02014-05-30 AT 06:41

Yes, but the output of system_profiler is hardly compatible with lsusb, and most Mac users are quite good with the mouse.