# PYTHON RFTEST.PY

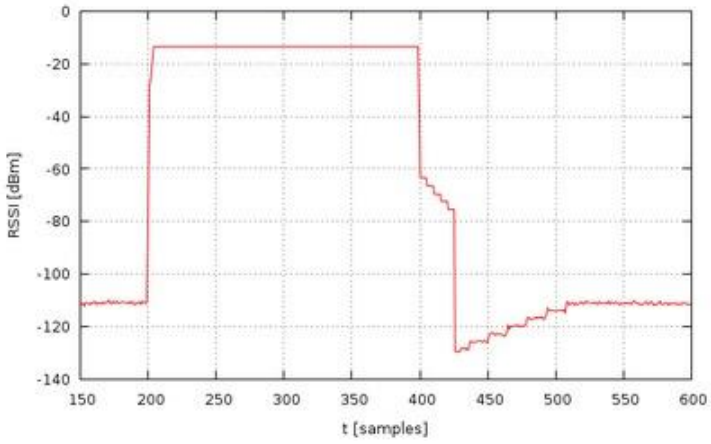
Recently I've been working on an automatic test harness for the spectrum sensing hardware I've designed at the Jožef Stefan Institute. It takes the form of a Python script that talks to a vector signal generator on one end and a VESNA on the other. It tests the receiver under different conditions and after a few minutes or so (depending on which tests have been selected) writes a nice report that includes characteristics like receiver noise figure, power level detector offset and linearity errors, local oscillator accuracy and so on, plus all the raw data in case it's needed to double-check the calculations.

Looking back, this should be the first thing to make after I got the prototype circuit boards on my desk. It would definitely spare quite a few hours of dull button pressing and note taking. Unfortunately even this academic environment is not immune to pressing dead-lines and sometimes it just makes more sense to go with the slow-and-certain way of manually doing things than opt for including software development in a task that will in all certainty save hours of work but might also explode into a week-long debugging session.

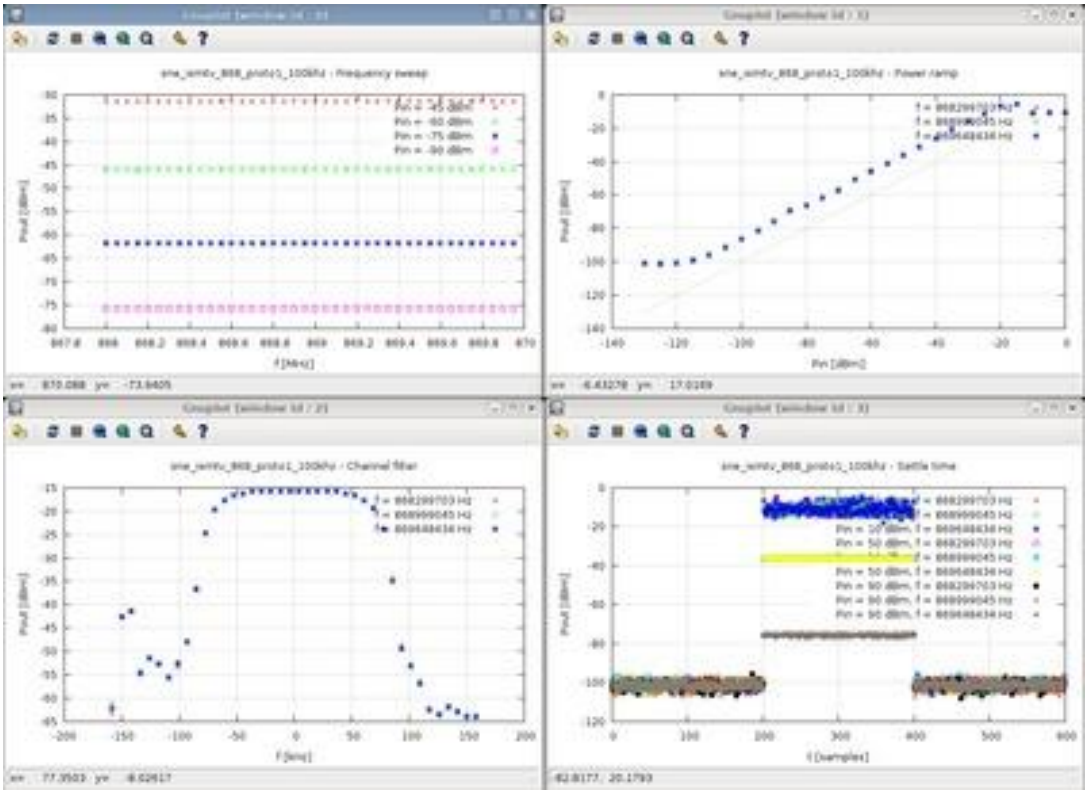
In the end however, the development went quite smoothly. Controlling our stack of Rohde&Schwarz instruments turned out to be surprisingly easy. Using the *USB Test & Measurement Class* under Linux is as complicated as reading and writing strings to a*/dev/usbtmc3* character device (my Debian Squeeze system already included all the necessary kernel modules). The high-level interface is the same as the one used on the venerable GPIB and consists of plain human-readable ASCII statements that control various aspects of the instrument or return measurement results. On our equipment the same interface is also exported as a telnet-like service on an Ethernet port, but I opted for USB since my laptop only has one wired network interface.

With this setup it's now finally feasible to exhaustively test a pile of receivers. This will also give me some statistical data to see how various characteristics differ from one device to the other without the need to depend solely on data provided by component manufacturers.

Definitely a good thing to have on hand when other researchers come knocking on the door asking why their theoretical calculations don't fit the measurements.




Above is one interesting result of these measurements: since these receivers are used for spectrum sensing, it's important to know how the power level detectors behave in different conditions. This is the result of a step-response test of the TV-band UHF receiver based on the NXP TDA18219HN tuner. This tuner has several stages of automatic gain control and the power level detector takes their individual gains into account when calculating the power at the antenna interface. While the turn-on response is nice and fast, it takes quite some time for the tuner to settle back and provide an accurate measurement when the signal source is turned off.



This a collection of test results for a narrow-band sub-1 GHz tuner based on the Texas Instruments CC1101 chip. VESNA spectrum-sensor application supports quite a few different configurations using this hardware. While the measurements above show a properly working receiver, so far the automated tests have already found several software bugs regarding the receiver configuration and one out-of-spec radio board.

Source: https://www.tablix.org/~avian/blog/archives/2012/10/python_rftest_py/