Experiences in Building Python Automation Framework for Verification and Data Collection

Juki W. Tantra

PyCon APAC 2010

IC Design House

- Wireless sound ASIC.
- Mainly embedded into HTS, soundbars, and headphones.
- Two main selling points:
  - High quality audio
  - Robust wireless transmissions.

Automation Framework

- Control *device under test* (DUT) and test equipments to perform specific measurements.
- Examples:
  - Measure power consumption with various settings or over time
  - Measure RF sensitivity in various environments
  - Detect audio defects over time.
- Common features:
  - Control DUT
  - Control various equipments.

Automation Framework Benefits

- Time-saving
  - Utilize weekends and nightime to do automated tests and data collections.
- Reduce manpower needs
  - Remove manual operations.
- Open up new possibilities
  - Time-consuming data collection becomes possible.

Goals of Our Automation Framework

- Easy-to-use framework based on Python that many engineers can start using immediately.
- Reduce the required test time by automating test cases.
- Extensibility of the framework to support various use: scheduled run, control of various equipment, and remote control/execution of the script.

Why Python?

- No compilation necessary
- Python is easy to read and is very flexible
- Python is mature with much supports behind it
- I love Python

Examples of Current Capabilities

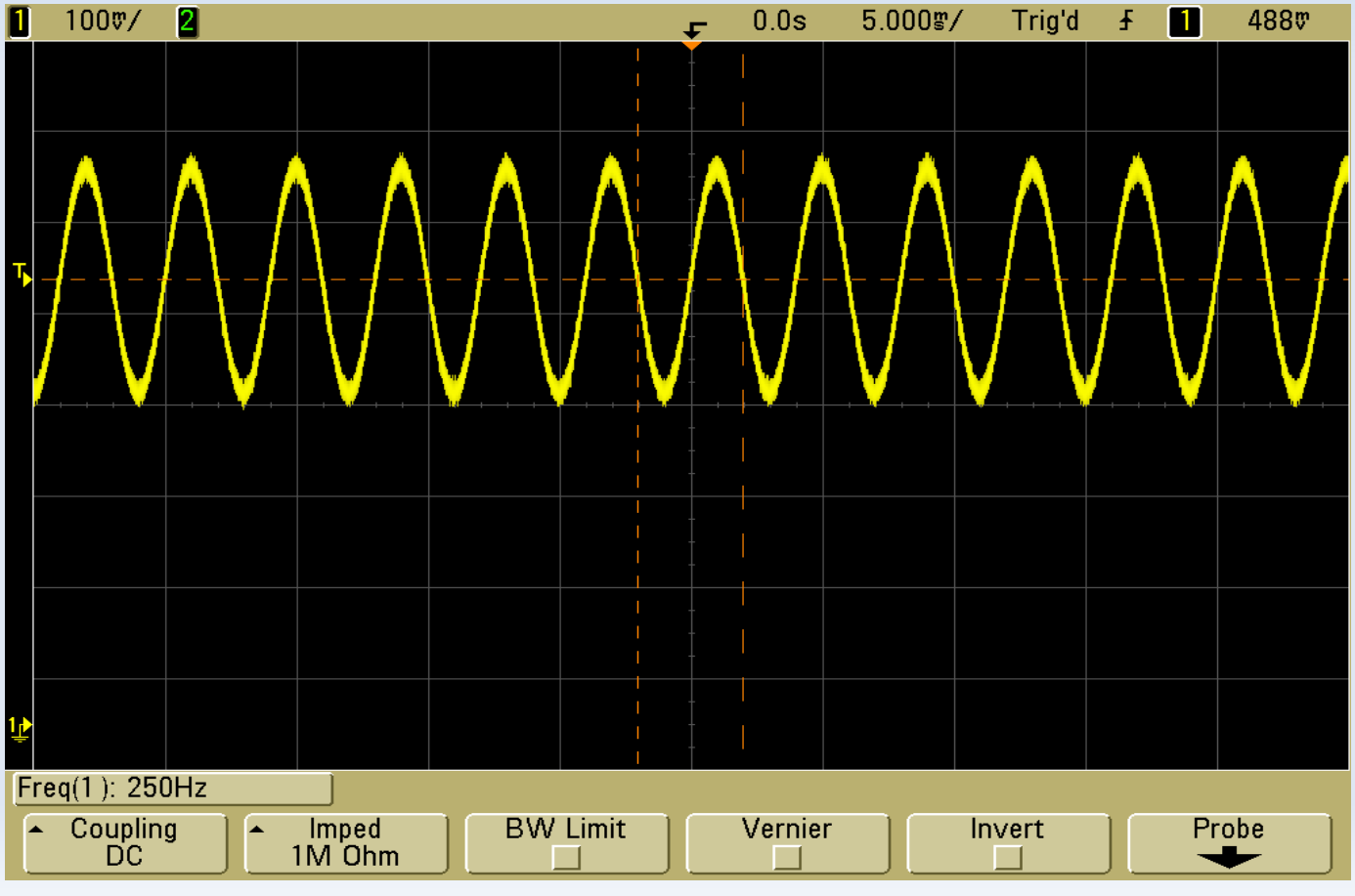
- Power measurements
- Oscilloscope control
- Signal generator control
- Spectrum analyzer control
- Power supply control
- Audio Precision control
- DUT control

PyVISA and VISA

- VXIplug&play I/O software language (VISA):
  - The interfacing protocol adopted by most of the test equipment vendors
  - VISA drivers (free) are available from NI, Agilent, and Tektronics.
- PyVISA written by Torsten Bronger
  - Provides the necessary abstraction of the VISA driver
  - Python wrapper of the VISA dll.

Example with Oscilloscope

```
dut1 = Dut(0)
dut2 = Dut(1)
scope = AgilentScope()
scope.write_setup(settings)
dut1.write_register(REGISTER_NAME, value1)
dut2.write_register(REGISTER_NAME, value1)
image1 = scope.read_png_image()
dut1.write_register(REGISTER_NAME, value2)
dut2.write_register(REGISTER_NAME, value2)
image2 = scope.read_png_image()
```



Basic Library for Oscilloscope

```
class AgilentScope(VisaDevice):
    LARGE_TIME_OUT = 50
    NORMAL_TIME_OUT = 10

    def read_setup(self):
        """
        Read the MSO setup and return it as a string.
        """
        self.set_timeout(AgilentScopeConnector.NORMAL_TIME_OUT)
        setup = self.ask_raw(":SYSTEM:SETUP?")
        setup = self.strip_term_chars(setup)
        return setup

    def write_setup(self, setup_settings):
        """
        Write the previously saved MSO setup.
        """
        self.set_timeout(AgilentScopeConnector.NORMAL_TIME_OUT)
        self.write(":SYSTEM:SETUP %s" % (setup_settings,))

    def __read_image(self, setting):
        self.set_timeout(AgilentScopeConnector.LARGE_TIME_OUT)
        img = self.ask(":DISPLAY:DATA? %s, SCREEN, COLOR" % (setting,))
        self.set_timeout(AgilentScopeConnector.NORMAL_TIME_OUT)
        return img[10:]

    def read_bmp_image(self):
        """
        PNG is faster.
        """
        return self.__read_image("BMP8bit")

    def read_png_image(self):
        return self.__read_image("PNG")

    def enable_persistence(self):
        self.write (":DISPlay:PERSistence INFinite")

    def disable_persistence(self):
        self.write (":DISPlay:PERSistence MINimum")
```

Power Measurement

```
dut = Dut(0)
for app in APP_LIST:
      dut.set_application(app)
      multimeter.config_min_max(True)
      multimeter.set_sample_count(MMETER_SAMPLE_COUNT)
      multimeter.measure_curr_dc('1')
      sleep(float(MMETER_SAMPLE_COUNT)*30/1000) # 30seconds
      curr_min, curr_max, curr_ave, count = multimeter.read_min_max()
      multimeter.config_min_max(False)
      csv_file.write(app, curr_min, curr_max, curr_ave, count)
```

Who are the users?

- Test engineers
- RF design engineers
- ASIC design engineers

- Notepad++ for Windows
  http://notepad-plus.sourceforge.net/uk/site.htm
- No IDE
- SVN for code repositories

Ingredients

- Example codes
  Over 50 examples

- Intuitive to use library
- Software consultant

app_aes_counter_sync.py
app_ap_test.py
app_capture_screen.py
app_cpld_rw_test.py
app_cu_mu_example_test.py
app_data_test.py
app_drm_block_rw_test.py
app_dual_band_control.py
app_get_num_streams.py
app_gui_settings.py
app_i2c_timeout_test.py
app_list_evk.py
app_mcu_rw.py
app_mcu_settings.py
app_mem_rw.py
app_mem_test.py
app_pcl_varying_values.py
app_radio_reg_test.py
app_read_per.py
app_register_rw_test.py
app_rf_offset.py
app_rf_offset_compensation.py
app_save_registers.py
app_send_cmd_to_cu_template.py
app_set_appl_test.py
app_set_continuous_mode.py
app_set_cu_mu.py
app_set_gui_channel.py
app_set_spdif_test.py

Benefits Quantified

Test Case	Manual	Automated
Audio defect test	5 man days	40 hours (< 2 days)
Power consumption measurement	3 man days	24 hours (1 day)
Modem sensitivity test	5 man days	40 hours (< 2 days)
Interference robustness test	5 man days	40 hours (< 2 days)

Python in This Field

- Common norm: VB, LabVIEW
- Python has all the necessary requirements to be dominant here
- Will Python be the preferred choice here?