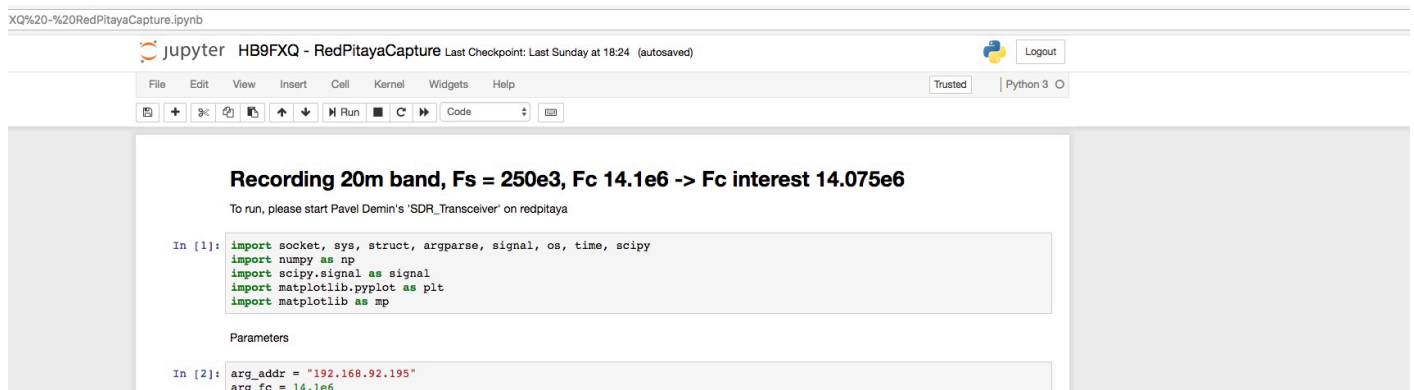

notes.superlogical.ch

[Home](#). [Pages](#). [Posts](#).

Using the RedPitaya in a Jupyter Notebook

03.03.2018 10:20



The screenshot shows a Jupyter Notebook titled "XQ%20-%20RedPitayaCapture.ipynb" with the kernel name "HB9FXQ - RedPitayaCapture". The notebook content includes a title "Recording 20m band, Fs = 250e3, Fc 14.1e6 -> Fc interest 14.075e6" and a note: "To run, please start Pavel Demin's 'SDR_Transceiver' on redpitaya". The code is as follows:

```
In [1]: import socket, sys, struct, argparse, signal, os, time, scipy
import numpy as np
import scipy.signal as signal
import matplotlib.pyplot as plt
import matplotlib as mp

Parameters

In [2]: arg_addr = "192.168.92.195"
arg_fc = 14.1e6
```

Using Jupyter notebooks for rapid prototyping, short proof of concepts and documentation is fun. Now I wanted to introduce some signal acquisition using the RedPitaya (+ <http://pavel-demin.github.io/red-pitaya-notes/sdr-transceiver/>).

Pretty simple task. Two years ago I've contributed a [short python script](#) to fetch samples directly from the trx project. Pretty simple to include it directly into a notebook:

```
fs_to_indexfs = {20000:0, 50000:1, 100000:2, 250000:3, 500000:4, 1250000:5}
```

```
class Transceiver(object):
def __init__(self, ipaddr):
self.p_ip_address = ipaddr
self.control_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
self.data_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
def shutdown(self):
print("RX Shutdown")
self.data_socket.close()
self.control_socket.close()
```

```
def start_rx(self, freq, fs_index, corr, n_bytes, iq_filename):
# Setup control socket
self.control_socket.connect((self.p_ip_address, 1001))
self.control_socket.send(struct.pack('>> rx force shutdown\n'))
self.shutdown()
f_iq_raw.close()
```

The acquisition looks like:

```
trx = Transceiver(arg_addr)
start = time.time()
activeReceiver = trx;
```

```
trx.start_rx(arg_fc, fs_to_indexfs[arg_fs], 0, arg_bytes_to_fetch, arg_raw_iq_file)
```

```
end = time.time() - start  
print(time.strftime("%H:%M:%S", time.gmtime(end)))
```

```
samples = np.fromfile(arg_raw_iq_file, dtype=np.complex64)  
x1 = np.array(samples).astype("complex")  
print(len(x1))
```

That's it. Now there is some numpy / matplotlib fun ahead. Let's quickly render e spectrogram:

