

GIANT METREWAVE RADIO TELESCOPE

PPS Generation on Serial Port using Python and Processing It

PROJECT REPORT

Submitted To

Shri Sandeep Chaudhari

By

Dhiraj Tambade

STP 2012

Index

Title	Page No.
Abstract	2
What is PPS SIGNAL?	3
Graphical overview of project	4
Introduction to NTP	5
Installation and Configuration of NTP	5
Python Installation	7
Flow Chart	8
Python Script	9
Max 485 circuitry	10
Results and Conclusion	11
References	14

Abstract

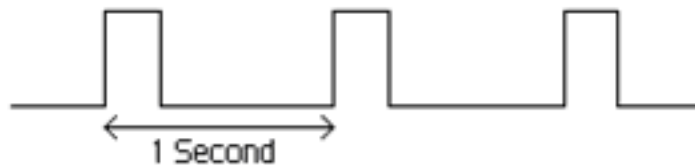
A Pulse per Second (PPS) signal is required as triggering input to packetized design. As per existing setup in GMRT, a long differential cable approx. length of 45m is required to be laid from GPS receiver to Digital Backend Lab. This is unrealistic in terms of cost and noise immunity.

To solve this problem, a PPS signal can be generated on serial port of computer which is locked to NTP server for precise timing. The PPS waveform is generated using Python programming and modules in it called Pyserial and Time. Using small MAX 485 circuitry, we will convert output of Serial Port in 0V-5V level i.e. Digital Logic.

What is PPS Signal?

A pulse per second (PPS) is an electrical signal that has a width of less than one second and a sharply rising or abruptly falling edge that accurately repeats once per second.

PPS signals are output by radio beacons, frequency standards, other types of precision oscillators and some GPS receivers.



Accuracy - 12 picoseconds to a few microseconds per second
2 nanoseconds to a few milliseconds per day

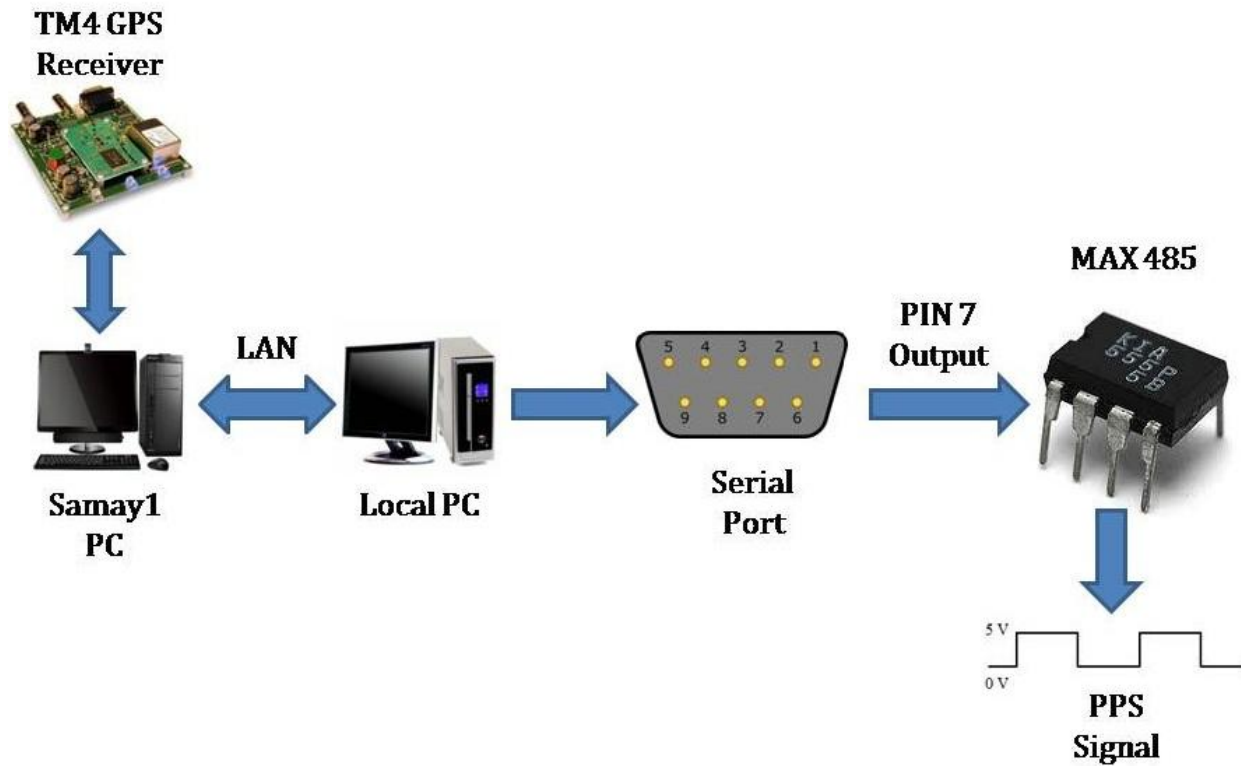
Use - Triggering input to Packetized Design
Precise timekeeping and time measurement

Common use for the PPS signal is to connect it to a PC using a low-latency, low-jitter wire connection and allow a program to synchronize to it. This makes the PC a stratum-1 time source.

Note that because the PPS signal does not specify the time, but merely the start of a second.

One must combine the PPS functionality with another time source that provides the full date and time in order to ascertain the time both accurately and precisely.

Graphical Overview of Project



Hardware Used

- ✓ PC connected to samay1 and Ubuntu OS installed on it.
- ✓ Serial Cable
- ✓ MAX 485
- ✓ Resistors - 6.8k, 4.7k
- ✓ Connecting wires
- ✓ 5V Power Supply
- ✓ Tektronics DPO2024

Network Timing Protocol (NTP)

The Network Time Protocol (NTP) is a protocol designed for accurately synchronizing local time clocks with networked time servers.

NTP is distributed, hierarchical system.

Primary Servers are machines that are synchronized to external time sources

Secondary Servers allow thousands of machines/organizations to synchronize without overloading primary servers.

- ✓ GPS is considered a stratum-0 source.
- ✓ Samay1 PC is primary server connected to TM4 GPS receiver.
- ✓ Local PC is connected to primary server and can also be used as secondary server.

How to Maintain an Accurate System Clock with ntpd

1. Install the NTP daemon:

```
sudo apt-get install ntp
```

2. Configure the daemon properly:

The configuration file for ntpd is located at /etc/ntp.conf
First section you may want to modify is the list of servers to synchronize with.

```
# You do need to talk to an NTP server or two (or three).  
server samay1.gmrt.ncra.tifr.res.in
```

3. Make sure the configuration works

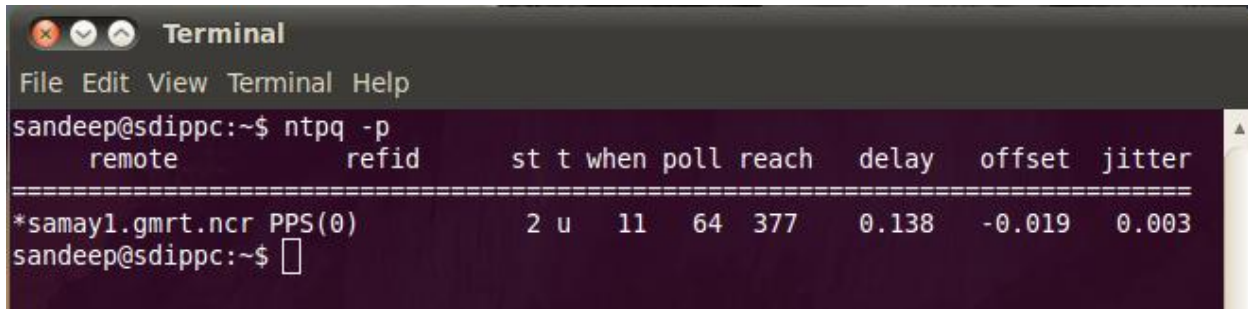
Now that you have a proper server list in your /etc/ntp.conf file, it is time to run the daemon and see if you synchronize properly! Make sure you have an active Internet connection, and then run:

```
sudo /etc/init.d/ntp restart
```

4. Check status of ntp:

ntpq -p

It will give result like this



```
Terminal
File Edit View Terminal Help
sandeep@sdippc:~$ ntpq -p
      remote          refid          st t when poll reach  delay  offset  jitter
=====
*samay1.gmrt.ncr PPS(0)          2 u  11  64  377  0.138 -0.019  0.003
sandeep@sdippc:~$
```

- ✓ Outputs a two-line header and then one line per server, with 11 fields
- ✓ 1st char: '*' for the host we sync to ; '+' for OK ; '-' and 'x' for bad ;
" for unreachable; '0' for PPS sync source
- ✓ Hostname
- ✓ Where does it get time from?
- ✓ Its stratum
- ✓ Type of peer ('u' is for 'unicast')
- ✓ Seconds ago we heard from it
- ✓ How often do we currently poll, in seconds
- ✓ Octal reach ability mask, 377 means A-OK
- ✓ Delay (round-trip) to the server, in milliseconds
- ✓ Offset in milliseconds
- ✓ Jitter in milliseconds

Python

1. Install Python:

```
sudo apt-get install ipython
```

2. Create file:


```
sudo vi <filename>.py
```

3. Write the script given on next page and save the file.

4. Run the program

```
sudo python <filename>.py
```

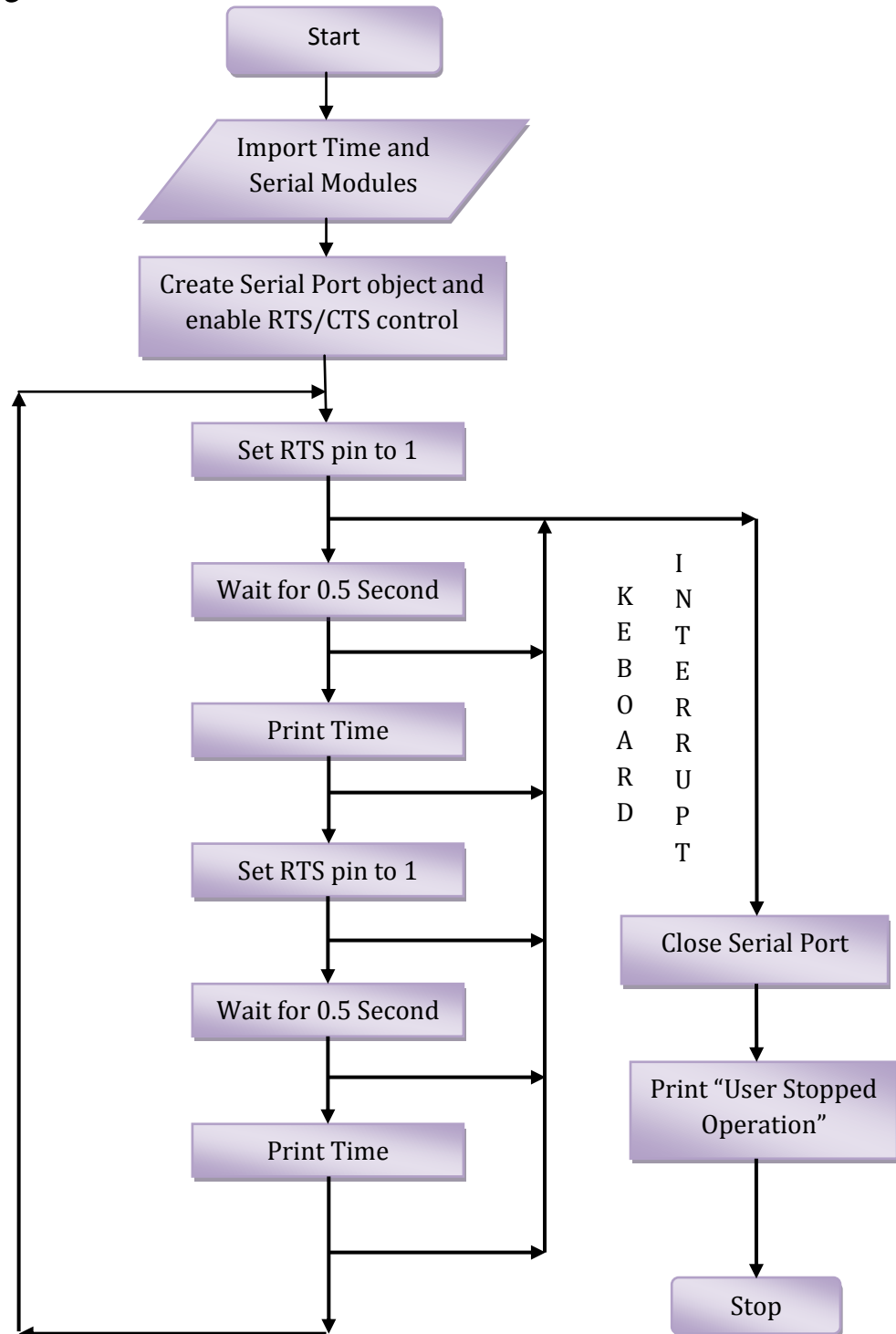
5. Output will look like this



```
Terminal
File Edit View Terminal Help
sandeep@sdippc:~$ sudo python toggle1.py
1339571913.66119
1339571914.00001
1339571914.50005
1339571915.00001
1339571915.50005
1339571916.00000
1339571916.50004
1339571917.00001
1339571917.50005
```

6. Stop operation only by pressing **ctrl+c**

Flow Chart



Python Script to generate PPS signal

```
#!/usr/bin/python
import serial
import time
ser=serial.Serial('/dev/ttyS0',rtscts=True)

try:
    while(1):

        x=time.time()-int(time.time())

        if (x+.5)>1:
            x=(x%1)-0.5

        ser.setRTS(1)

        while( (x+0.5)>(time.time()-int(time.time()))):
            pass
        print "%4.5f"%time.time()

        ser.setRTS(0)

        while ( (x+0.5)<(time.time()-int(time.time()))):
            pass
        print "%4.5f"%time.time()

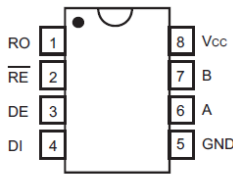
except:KeyboardInterrupt
ser.close()
print("\n User stopped operation")
```

MAX 485 Circuitry

Need -

Output of serial port varies between $\pm 15V$
 We need PPS Signal varying from 0V to 5V.

PIN CONFIGURATION



Features

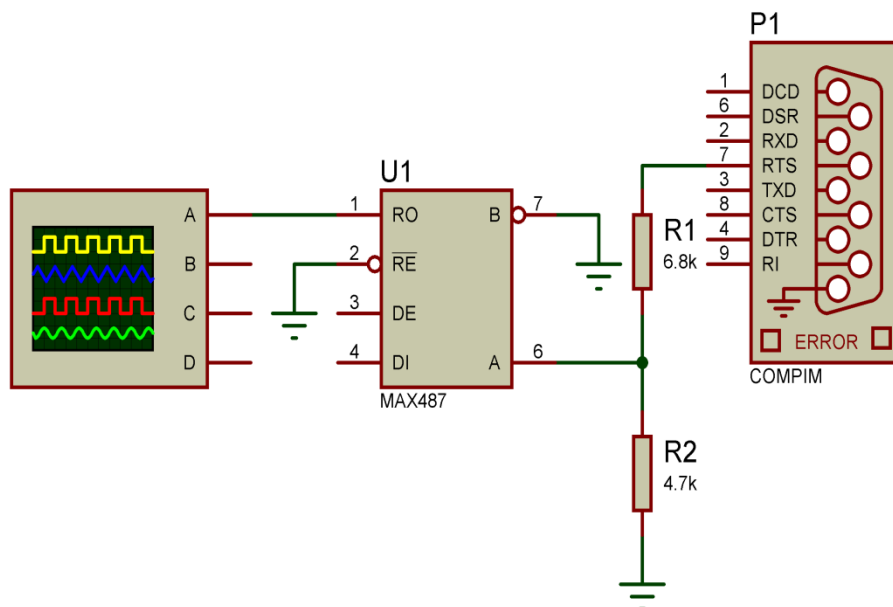
- ✓ -7V to +12V Input Voltage Range
- ✓ 30ns Propagation Delays
- ✓ Operate from a Single 5V Supply

D OR P PACKAGE

(Top View)

Circuit Description -

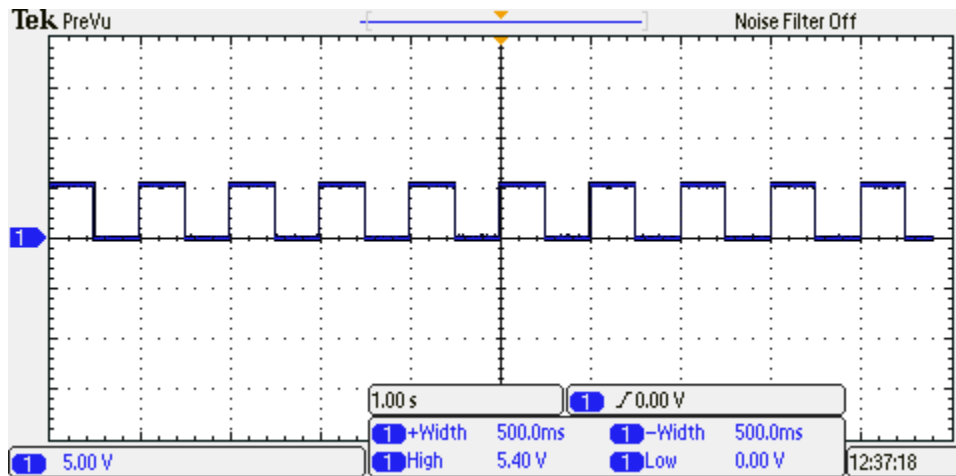
- ✓ Resistor Divider gives 40% of Serial Port output to A which becomes compatible for MAX 485.
- ✓ Receive output - If $A > B$ by 200mV, RO will be high;
 If $A < B$ by 200mV, RO will be low.
- ✓ Here we connected B to ground so that RO toggles with respect to A.



Results

Following results were observed on Tektronics DPO2024

1. PPS Signal

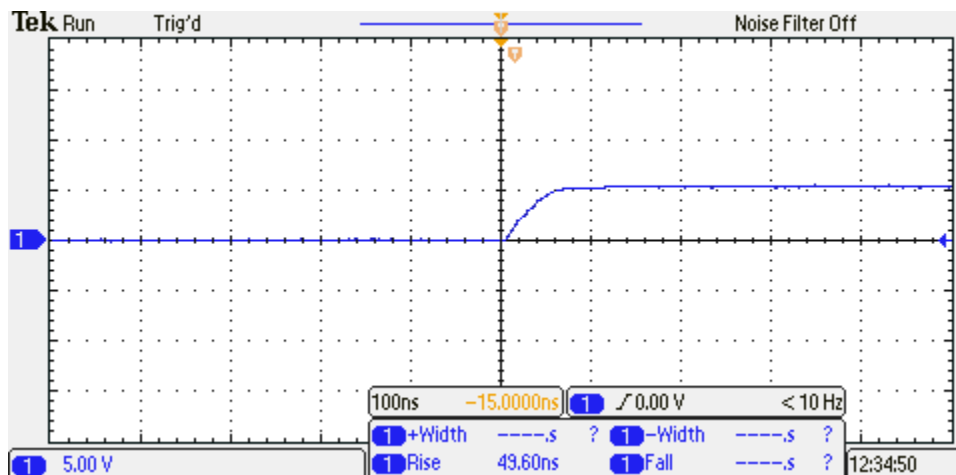


Period – 1 second

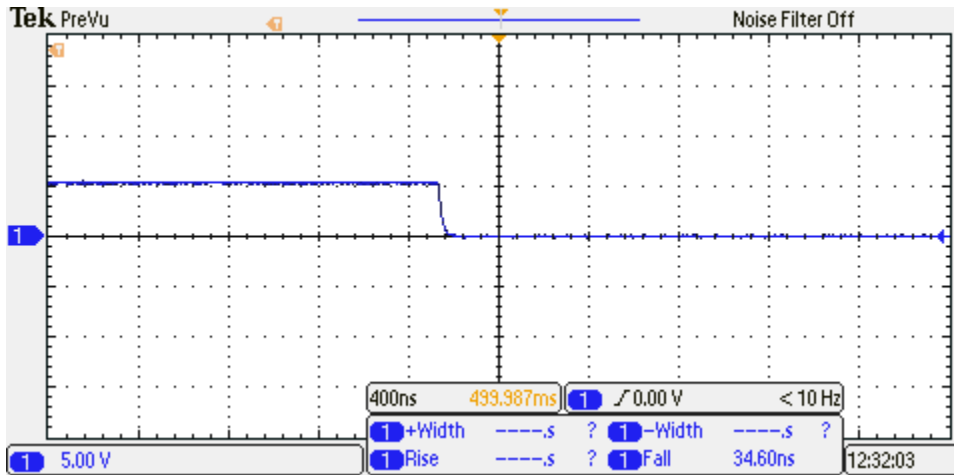
Amplitude – High: 5.4 V and Low: 0 V

Width – Positive: 500 ms and Negative: 500ms

2. Rise Time: 49.60 ns



3. Fall Time: 34.60 ns



4. Testing Result - Successful Triggering Of FPGA

```

Applications Places System Wed Jul 11, 11:04 sandeep
Terminal
File Edit View Terminal Help
1) 1341984759.500061035
0) 1341984760.00005007
1) 1341984760.500051975
0) 1341984761.00006914
1) 1341984761.500066996
0) 1341984762.000010967
1) 1341984762.500071049
0) 1341984763.00006914
1) 1341984763.500051022
0) 1341984764.00008106
1) 1341984764.500067949
0) 1341984765.00009060
1) 1341984765.500063896
0) 1341984766.00005960
1) 1341984766.500066996
0) 1341984767.00005007
1) 1341984767.500061989
0) 1341984768.00005007
1) 1341984768.500061989
0) 1341984769.006968975
1) 1341984769.507045031
0) 1341984770.00006914
1) 1341984770.500072002
0) 1341984771.000003099
1) 1341984771.500046968
0) 1341984772.00009060
1) 1341984772.500068903
0) 1341984773.000008106
1) 1341984773.500051022
0) 1341984774.00006914
1) 1341984774.500061035
0) 1341984775.00008106
1) 1341984775.500066996
0) 1341984776.00008106
1) 1341984776.500070095
^C
User stopped operation
sandeep@sdipecp:~$

gmrtr@rchpc3: ~
File Edit View Terminal Help
Initial configuration:
=====
Clearing the FPGAs... done.
Programming the Engines with r_128w_512_11_r370_mod3_16_2012_Jul_04_2019.bof a
nd the Xengines with r_if_2x_4a_r340c_2011_Feb_11_1454.bof... done.
Pausing 10GbE data exchange... Pausing Xengs... done.

For POCO functionality....
Setting POCO integration time 781184 FFT cycles => 0.999916

Syncing the F engines... Armed. Expect trigg at 11:01:16 local (05:31:16 UTC).
SPEAD packet sent.
Checking F engine clocks... ok
Setting the board indices... done
Setting the FFT shift schedule to 0x7FF... done
Configuring EQ... done
Configuring the 10GbE cores... done
Waiting 13.6 seconds for ARP to complete... done
Starting 10GbE data exchange... X engines re-enabled.
Flushing loopback muxs... done.

=====
Verifying correct data exchange...
=====
Wait 2 seconds for system to stabilise... done
Resetting error counters... done
Checking that all XAUI links are working... ok
Checking that the same timestamp F engine data is arriving at all X boards with
in a sync period... ok
Checking that FPGAs are sending 10GbE packets... ok
Checking that all X engine FPGAs are receiving 10GbE packets... ok
Waiting for loopback muxes to sync... ok
Checking that all X engines are receiving all their packets... ok
Setting the number of accumulations to 781184 (1.000 seconds) and syncing VACCs
... done
Checking vector accumulators... Waiting for an integration to finish... done. C
hecking... ok
Sending SPEAD metadata and data descriptors to 127.0.0.1:7148... done
Configuring output to 192.168.100.1:7148... skipped.
Starting transmission of data... done
Resetting error counters... done
Enabling KITT... done
gmrtr@rchpc3:~$ ^C
gmrtr@rchpc3:~$

```

References

1. www.en.wikipedia.org
2. www.ubuntuforums.org
3. www.ntp.org/
4. www.docs.python.org/py3k/tutorial/index.html
5. *MAX 485 Datasheet*
6. *Network Time Protocol (NTP): Overview and Configuration*
Stanislav Shalunov hshalunov@internet2.edu
Performance Workshop, Atlanta, 2005-04-22