Tartu University

Faculty of Science and Technology

Institute of Technology

Miroslav Rolko

Development and performance analysis of the Network Time Protocol Server

Master's thesis (30 EAP)
Robotics and Computer Engineering

Supervisor(s):

MSc Viljo Allik
MSc Indrek Sünter

Tartu 2019

Resümee/Abstract

Võrguaja protokolli serveri arenduse ja toimivuse analüüs

Käesolev magistritöö kirjeldab lihtsa võrguaja protokolli (NTP) serveri ehitamist monoplaatarvutisse Tartu Observatooriumi aatomikella tarbeks. See selgitab NTP-protokolli terminoloogiat ja on loodud NTP-serveri ja kliendi jaoks. Mõõtmise eesmärgil kirjeldatakse ka seda, kuidas ehitatatakse NTP-klienti, mis genereerib impulssi sekundis, mis põhineb NTP-kellaajale. Selle kasutamise tulemusi võrreldakse NTP plotteri testitulemustega. Seejärel hinnatakse NTP-serveri eksperimentaalset konfiguratsiooni, millele järgneb arutelu võimalike paranduste ja tulevaste projektide üle.

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia, T125 Automatiseerimine, robootika, control engineering, T170 Elektroonika, T190 Elektrotehnika

Marksõnad: mikrokontroller, aatomkell, monoplaatarvuti, aeg, protokoll, tarkvara, satelliit, pulss

Development and performance analysis of the Network Time Protocol Server

This Master thesis describes the process of building a simple network time protocol (NTP) server on a single-board computer for the Atomic clock at Tartu Observatory. It explains the terminology of NTP protocol and set up for the NTP server. It also describes how to build a NTP client which generates pulse per second by its own script for the measurement purposes. This new pulse is synchronized with the NTP timestamp. Results of its use are compared against the test results from NTP plotter. The experimental configuration of the NTP server is then evaluated which is followed by a discussion regarding possible improvements and future projects.

CERCS: T120 Systems engineering, computer technology, T125 Automation, robotics, control engineering, T170 Electronics, T190 Electrical engineering

Keywords: microcontroller, atomic clock, single-board computer, time, protocol, software, satellite, pulse

Contents

Terms and abbreviations

AC	Alternating current
ADC	Analog-to-digital converter
AMP	Amplifier
ARM	Advanced RISC machine
ASCII	American Standard Code for Information Interchange
BDS	The BeiDou Navigation Satellite System
BNC	Bayonet Neill–Concelman
CCIR	International Radio Consultative Committee
CMOS	Complementary metal–oxide–semiconductor
COM-port (communication port)	Serial port interface
CPU	A central processing unit
CS	Current sense
CTS	Clear to send
DC	Direct current
DCD	Data Carrier Detect
DDR	Double Data Rate
DTSS	Digital Time Synchronization Service
ESA	European Space Agency
GGA	Global Positioning System Fix Data
GLONASS	Global Navigation Satellite System
GPIO	General purpose input / output
GPS	Global Positioning System
GPSD	Global Positioning System daemon
GNSS	Global Navigation Satellite System
HDMI	High-Definition Multimedia Interface

ID	Identifier
IP	Internet protocol
I^2C	Inter-integrated circuit
IAU	International Astronomical Union
ITU	International Telecommunication Union
IRS	Interrupt service routine
LAN	Local Area Network
LED	Light emitting diode
LEO	Low-Earth orbit
MCU	Microcontroller unit
NASA	The National Aeronautics and Space Administration
NTP	Network time protocol
NMEA	National Marine Electronics Association
OS	Operation system
OSC	Crystal oscillator
PC	Personal computer
PCB	Printed circuit board
PPM	Parts per million
PPS	Pulse per second
PPSAPI	Pulse per second application programming interface
PuTTY	Open-source terminal emulator, serial console and network file transfer application
RAM	Random-access memory
RS-232	Serial standard
RTC	Real time clock
RTS	Request to send
SCIP	software for mixed-integer programs

SD	Solid drive
SDA	Serial data line
SDC	Serial data clock
SPI	The Serial Peripheral Interface
RTC	Real Time Clock
TAI	International Atomic Time
TCP	Transmission Control Protocol
TSIP	Trimble Standard Interface Protocol
TTL	Transistor-transistor logic
UART	Universal asynchronous receiver/transmitter
UNIX	Multitasking, multiuser computer operating systems
UDP	User Datagram Protocol
USB	Universal serial bus
UTC	Coordinated Universal Time
VCO	Voltage control oscillator
VMS	Virtual machines server
VFO	Variable-frequency oscillator
WAN	Wide Area Network

1. Introduction

The Network Time Protocol (NTP) is one of the oldest internet protocols still in use. It has been operational since before 1985 and was invented by David L. Mills. It is a widely accepted method for synchronizing time in computer networks. [1].

The benefits of building NTP servers have never been as important. Internet technologies continue to change the shape of everyday life while network demands and data flow are constantly increasing. The interest in, and use of space technologies is also rapidly growing. Space agencies around the world continue to launch satellites into orbit for various reasons. Efficient and effective inter-satellite communication and ranging systems are essential to keep these satellites and their services functioning.

Small, dedicated NTP servers can be set up at almost no cost. These servers can then be used to monitor private and business networks as well as to calibrate network products after assembly. NTP servers may also be included as an additional system for atomic clocks which possess a highly precise output of one pulse per second.

This thesis project has been undertaken at the Space Technology Department of Tartu Observatory. The observatory has its own atomic clock installed but, at the time of writing, a 10 MHz output was only being used to synchronize the frequency of the ground station and the devices in the laboratory. The atomic clock GPS-89 was also transmitting at a frequency of one pulse per second (PPS) with 60ns precision through a BNC cable. The clock was not connected to any external system and its highly accurate PPS was not being used.

The idea of this project was to build a simple NTP server for the existing ground station. Tartu Observatory is likely to use this new NTP server for spacecraft ranging in future projects like ESTCube-2 [2]. Such a server can provide an accurate timestamp and frequency range for packets transferred between a spacecraft and a Ground Station and can also provide accurate time in the observatory's local network. The new NTP server can also be also used to create accurate timestamps of astronomical observations.

There are many NTP servers available today which provide a time precision in microseconds but whether this is precise enough for all applications remains questionable. An imprecise NTP server can be a potential hazard for clients on that NTP server. For example, if brokers are trading shares on financial market and their timestamp polled from an NTP server is less precise than that of the market then this can result in financial losses. One of the objectives of this thesis is to introduce a novel approach to determining the accuracy of a timestamp provided by an NTP server. .

This thesis is divided into the following sections; firstly, the requirements for building an NTP server are investigated. Secondly, discussions on what technologies were available for its construction as well as descriptions of existing solutions. Thirdly, the NTP protocol will be described followed by the fourth part of the thesis which describes the network protocol daemon. The fifth part of the thesis describes a new method for analyzing the accuracy of an NTP server and how it differs from other published solutions. In the same section, the results of measurements will be analyzed and discussed. Finally, the experimental NTP server will be evaluated followed by a discussion of possible improvements and future projects.

The objectives of this Master thesis for the construction of a new NTP server for atomic clock GPS-89 are as follows:

- Gather information about requirements for the new NTP server;
- Select components, design and draw a block diagram and integrate available hardware into a working NTP server and client system based on a single-board Linux computer with ARM architecture;
- Introduce network time protocol and daemon software,
- Install and connect the newly built NTP server to an existing GPS-89 rubidium clock;
- Improve system performance by optimizing the ARM multicore architecture;
- Undertake NTP server and client load testing. Analyze the system performance and time accuracy by comparison of NMEA and TSIP protocols and measurements with other time servers, and;
- Compare performance of LAN speed at 100 megabits per second (Mbps) and one gigabit per second (Gbps).

A more detailed list of these procedures is provided in section 2.1.

2. NTP server and client requirement, design and installation

The development of NTP server has been divided into following sections:

- Build requirements;
- Design, and;
- Assembly.

2.1 Requirements for NTP Server and Client

2.1.1 Microcontroller

The microcontroller should be from the Advanced RISC Machine (ARM) family and include the optimum number of pins for this project. Also, it should have enough processing power so as to obtain reasonable results for the experiment. The MCU is preferable because it has low power consumption, a small foot print, and good reliability and uses open source software.

2.1.2 Measurements

The PPS output from the atomic clock will be the main reference for the NTP server and will be connected to an oscilloscope for the purposes of this study. The NTP server should be able to synchronize time in a local area network (LAN) within 50μs accuracy in relation to a UTC timestamp.

2.1.3 Communication

Communication between the NTP server, client and computer has to be established via an Ethernet socket using the TCP or UDP protocol and will be controlled using Putty or WinSCP command lines. Ethernet sockets should support speeds of one Gbps or more.

2.1.4 Price

Both the NTP server and client should be of low cost while providing maximum efficiency.

2.1.5 Memory

NTP servers and clients produce NTP statistics as per their internal protocols. There should be enough memory on the server to store this data for a period of more than one year.

2.1.6 Protocols

The NTP server needs to be able to support the National Marine Electronics Association (NMEA) and Trimble Standard Interface Protocol (TSIP).

2.1.7 Power

The NTP server should have a low demand on electricity consumption.

2.1.8 Software

For this purpose, a network time software daemon (ntpd) should be used and for simplicity, the system should also be able to run on LINUX architecture.

2.1.9 NTP client

Network time protocol client should be built on individual single-board computer and generate pulse per second by script written in any program language. Generate pulse has to be synchronized with UTC timestamp provided by network time protocol daemon.

2.2 Server Design and Atomic Clock Description

The optimal design for this project is to build the NTP server on a single-board computer with a GPS module attached that supports all the required protocols.

2.2.1 Components selection

Most of the components and single board-computers were ordered from various online suppliers and some older GPS modules with TSIP protocol were provided by Tartu Observatory. A Raspberry Pi Expansion Board was ordered from the electronics supplier Uputronics. [3]

2.2.2 NTP server simple diagram

The atomic clock GPS-89 was the main reference for the NTP server. The server was connected to a GPS module which transmitted the GPS timestamp for the PPS that was produced by the atomic clock. The entire system was connected to one local switch. The pulse that is transmitted by the clock is sent to the bridge and then on to the oscilloscope for pulse analysis. This can be seen in Figure 1.

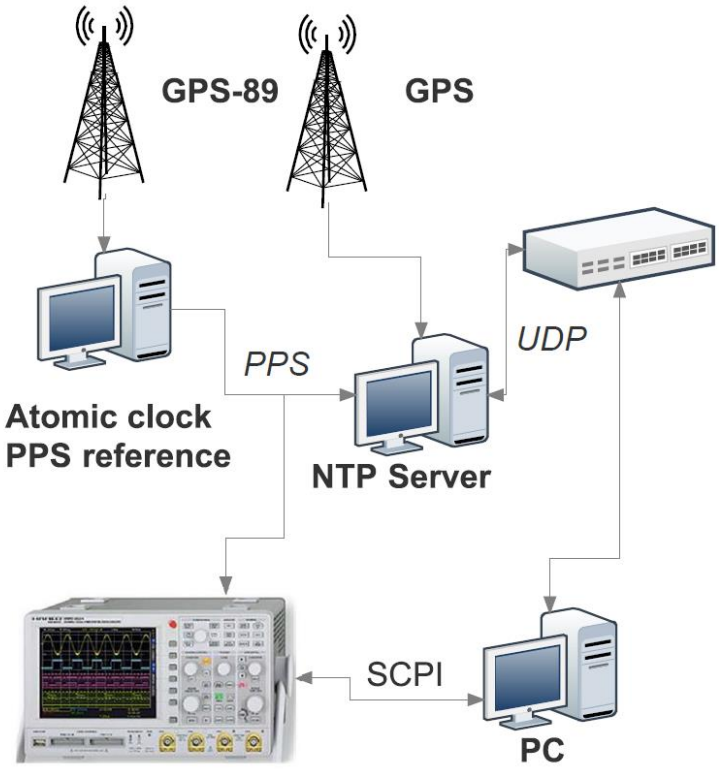


Figure 1: ***Simple diagram of the proposed NTP server for an atomic clock.***

2.2.3 Schematic design software

DesignSpark PCB 6.1 was the PCB layout drawing program chosen for this project as an additional external board for an older GPS module was used. This design software was recommended by an engineer at Tartu Observatory [4].

2.2.4 Single board computer selection for the server

After conducting research on the internet about existing NTP server setups [5]it seemed that one of the best choices for building an NTP server and client was to buy single-board computers such as NanoPi NEO2 or Raspberry Pi 3B. They both seemed to possess the suitable architecture and efficiency for such a system.

NanoPi NEO2 [6] has a one Gbps ethernet socket for LAN, making it the preferable device as the Raspberry Pi 3B [7] can support a maximum of only 100 Mbps. It is also possible to order Raspberry Pi 3B+ which is already equipped with a one Gbps ethernet socket, but it can support a maximum of only 300 Mbps. The NanoPi NEO2 with a one Gbps ethernet socket seemed to provide the ideal environment for stress testing procedures as well as providing enough bandwidth for potentially larger demands. The software for both NanoPi and Raspberry Pi computers is listed in Table 1.

2.2.4.1 NTP servers for LINUX operating systems

NanopiNEO2	**Armbian_5.65_Nanopineo2_Ubuntu_bionic_next_4.14.78**
Raspberry Pi 3B	Linux raspberrypi 4.19.23-v7

Table 1. *NTP servers for LINUX operating systems*

2.2.5 GPS module with TSIP and NMEA protocols selection

For this project, Tartu Observatory provided the Trimble
GPS module SMT360™. This module operates with both NMEA and TSIP protocols. The RES SMT360™ Timing Module is used on the carrier board. This makes available the concurrent reception of GNSS (GPS, Galileo, GLONASS, BeiDou) with a PPS pulse that has a 15ns accuracy. The main power supply requires 3V for the Trimble chip and 5V for the antenna. [8]

2.2.5.1 External PCB board design

The GPS module has been attached to a small PCB board which was previously designed by a Tartu Observatory engineer. The external PCB board contains a separate power supply for the antenna and the Trimble chip and also has an additional RS-232 driver - so it is possible to connect the GPS module directly through a serial USB converter to a PC. Once the GPS module is connected to a PC, the Trimble GPS can be set up using GUI Trimble Visual Timing Studio software. [9]The external PCB board design can be seen in Figure 2.

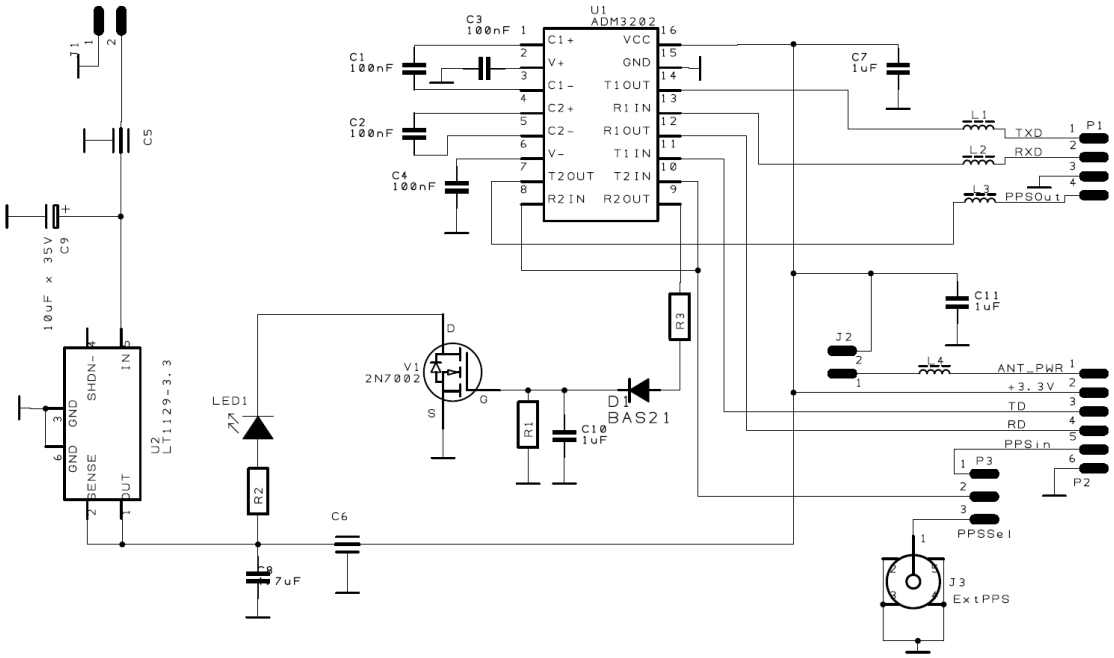


Figure 2: ***External PCB board interface for GPS SMT360 [Author: Viljo Allik Tartu Observatory] Appendix A***

2.2.6 Raspberry Pi+ GPS Expansion Board

Following research online, it was decided to order the GPS Expansion Board (Product code: HAB-GPSPI-NAN) from the electronics supplier Uputronics. This module generates a PPS pulse with a length of 100ms and also supports the NMEA protocol. [10]

It is a solderless design and is easy to install on top of the Raspberry Pi 40- pin header. Its design will also reduce signal noise as cabling is not required. This expansion board includes the advanced u-blox MAX-M8Q [11] [12]chip revision 04 (released on January 2019), which possesses concurrent reception of up to four GNSS (GPS, Galileo, GLONASS, BeiDou) and provides a 60ns time pulse accuracy similar to that of the atomic clock.

2.2.7 NTP NanoPi NEO2 server connection diagram

As the atomic clock produces a 5V, PPS output, it is necessary to design a small PCB board with a voltage divider in order to protect the GPIOs on the single-board computer. The divider formula is shown in Figure 3. It is assumed that each pin can theoretically handle a maximum of 4V. After the PPS is voltage level adjusted to the suitable 3.3V level, then the signal will be connected to the NanoPi NEO2 on pin PA6 on the twenty four pin header of the NanoPi NEO2. Serial interface pins are connected according to Table 2 and shown in Figure 4. For the analyses of the PPS signal from the atomic clock, it is also bridged to the oscilloscope. An external power supply of 3.3V and 5V is connected to the external GPS board.

2.2.7.1 Divider for both single-boards and NanoPi NEO2 pin settings

$$V_{out} = (\frac{Z_2}{Z_1+Z_2})* V_{in} \quad 3.3\ \text{V} = (\frac{1\ \text{k}\Omega}{1\ \text{k}\Omega+1\ \text{k}\Omega})* 5\ \text{V}$$

Figure 3: ***Divider formula***

NanoPiNEO pins	Encoders interface pins
PG6	TXD
PG7	RXD
PA6	1 PPS

Table 2.

NanoPi NEO2 right pinheader			
3.3V OUT	1	2	5V IN
SDA	3	4	5V OUT
SCL	5	6	GND
PG11	7	8	PG6
GND	9	10	PG7
PA0	11	12	PA6
PA2	13	14	GND
PA3	15	16	PG8
3.3V OUT	17	18	PG9
PC0	19	20	GND
PC1	21	22	PA1
PC2	23	24	PC3

Table 3.

2.2.7.2 NanopiNEO2 based interface block diagram

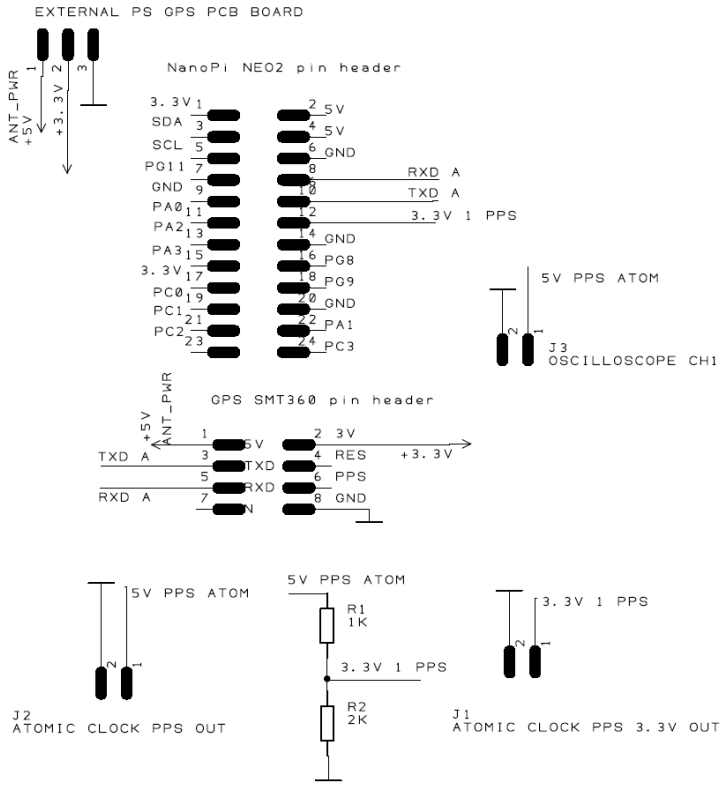


Figure 4: ***Block diagram of NTP server for Atomic clock***

2.2.8 NTP Raspberry PI B server and client simple schematics

The Raspberry Pi 3B computer was also used to construct an NTP server so as to compare its performance with the NanoPi NEO2 computer. Both of these single-board computers have a very similar ARM architecture. As with the NanoPi NEO2 server, the Raspberry Pi 3B server adjusted signal from the atomic clock will be sent to pin 18 on the GPIO (Table 4). The Raspberry Pi 3B was also used for the NTP client. On the left in Figure 5 is shown the forty-pin header of the GPS expansion board that includes a real time clock (RTC) and serial interface. This is connected to the forty-pin header of the NTP server interface which can be seen in the middle of Figure 5. On the right-hand side of Figure 5 is a diagram of the forty-pin header of the client with PPS Client interface which is connected to oscilloscope channel one.

On the bottom of Figure 5 is a block diagram of the PPS interface from the atomic clock with a small voltage divider connected to channel two of the oscilloscope.

Raspberry pins	PPS, RS232 interface pins
GPIO14	TXD
GPIO15	RXD
GPIO18	1 PPS

Table 4.

Raspberry 3B Pinout			
3.3V	1	2	5V
SDA	3	4	5V
SCL	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Table 5.

2.2.8.1 Raspberry Pi 3B based interface NTP Server and Client block diagram

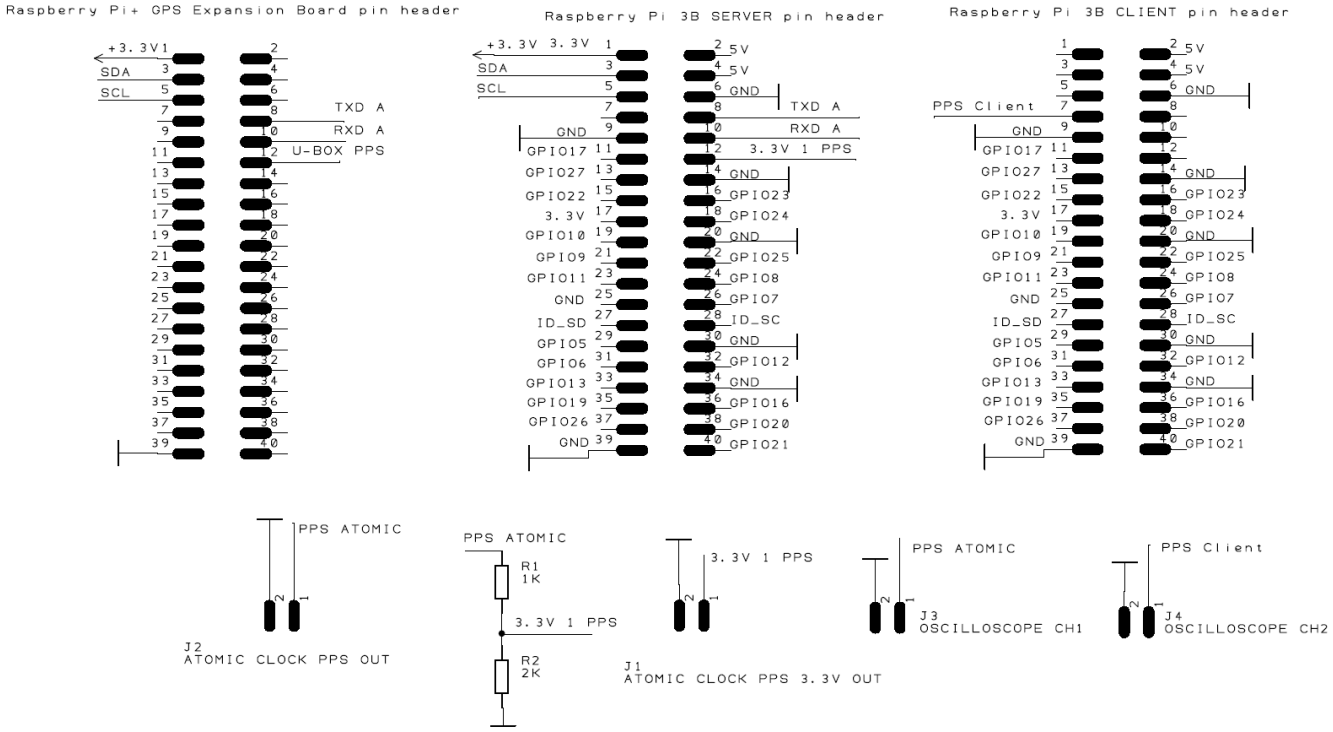


Figure 5: ***Block diagram of NTP server for Atomic clock***

2.2.9. Comparing hardware specification of single-board computers

A comparison of NanoPi NEO2 and Rasberry Pi 3B single-board computers is presented in Table 6.

NANOPI NEO2	Raspberry Pi 3 B
4x CPU: Allwinner H5, Quad-core 1.2 GHz 64-bit high-performance Cortex A53	4x CPU: Broadcom BCM2837 chipset running at 1.2 GHz 64-bit quad-core ARM Cortex-A53
DDR3 RAM 1GB RAM	1 GB LPDDR2 memory
Connectivity: 10/100/1000M Ethernet, RTL8211E-VB-CG chip	1 x 10/100 M Ethernet port, 802.11 b/g/n Wireless LAN, Bluetooth 4.1 (Classic & Low Energy)
USB Host: USB Type A x 1 and USB pin header x 2	4 x USB 2.0 port
MicroSD Slot: MicroSD x 1 for system boot and storage	MicroSD Slot: MicroSD x 1 for system boot and storage
LED: Power LED x 1, System LED x 1	LED: Power LED x 1, System LED x 1
GPIO1: 2.54mm pitch 24 pin-header, compatible with Raspberry Pi's GPIO pin1 - pin 24. It includes UART, SPI, I2C, IO etc.	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip providing ,27 GPIO pins as well as +3.3 V, +5 V and GND supply lines, UART, I2C, SPI
GPIO2: 2.54mm pitch 12 pin-header. It includes USB, IR receiver, I2S, IO etc.	1 x HDMI video/audio connector
Serial Debug Port: 2.54mm pitch 4pin-header	Serial Debug Port: 2.54mm pitch 4pin-header
Audio In/Out: 2.54mm pitch 4 pin-header	1 x RCA video/audio connector
MicroUSB: Power input(5V/2A) and OTG	MicroUSB: Power input(5V/2.5A) and OTG
PCB Dimension: 40 x 40mm	PCB Dimension: 85 x 56mm
Working Temperature: -30℃ to 70℃	Working Temperature: -25℃ to 80℃
Weight: 13g(WITHOUT Pin-headers)	Weight: 39.45g
OS/Software: Ubuntu bionic next 4.14.78	OS/Software: Linux raspberrypi 4.19.23-v7

Table 6.

2.2.10 Atomic Clock

Most atomic clocks utilize atomic isotopes Caesium $_{-133}$ or Rubidium $_{-87}$. Count pulses of time are measured from microwaves emitted by the electrons around those atoms. These electrons move from a lower to a higher orbit under the influence of lasers and the electrons become excited as they absorb energy from these lasers. While returning to a lower energy level, the electrons radiate the excessive energy as microwaves.

High and low energy atoms are separated by magnets. The separated low energy atoms are bombarded by an appropriate wavelength of radiation which converts them to high energy atoms. They are then moved by a second magnet to a detector which in turn, produces a current. This current is tied to an oscillator, which creates a feedback loop for the process. If low energy atoms predominate then the oscillator slows down, and the current stops. This loss of current is registered and extra voltage is applied to the system to compensate for this change in frequency. [13]

Clocks today are based around a frequency source generally called oscillators and come in a variety of types:

Quartz Oscillators
- TCXO – Temperature Compensated Crystal Oscillator
- OCXO – Oven Controlled Crystal Oscillator

Atomic Oscillators
- Rb – Rubidium Oscillator
- Cesium – Cesium Atomic Oscillator
- H-Maser – Hydrogen Atomic Oscillator

2.2.10.1 Atomic Clock Pendulum-GPS-89

The NTP server was built for the local atomic clock Pendulum-GPS-89
The GPS-controlled frequency standards, Pendulum GPS-88 and GPS-89, deliver a precision frequency and time reference with a PPS signal output of 5V DC, pulse frequency Sine wave. Atomic clock possess frequency stability locked to GPS frequency offset (24 h mean):$1x10^{-12}$ (at temperature 20°C to 26°C) [14].

2.3 Assembly

The final, working design has not been decided yet, but according to Tartu Observatory they would use a 90 inch rack-mount module to house all necessary equipment once the NTP server has been built and proven to work.

3. NTP protocol

The Network Time Protocol (NTP) is a protocol used to synchronize computer clock times in a network, such as for radio or satellite receivers and mobile phone network. It provides a time accuracy of less than a millisecond on a LAN and up to a few milliseconds on a WAN. The NTP protocol synchronizes and calibrates local and external NTP servers on diverse network paths so as to provide local time with the highest accuracy. NTP is based on the Internet Protocol (IP) and the User Datagram Protocol (UDP) on port 123. [1]

NTP has the following four synchronization modes, a client mode, a server mode, a peer mode and a broadcast mode. There are differences in how they measure offsets and network delays. For this thesis, we will focus on the first two. Both client and server modes are built in a hierarchical structure divided into stratum levels.

NTP aligns the system clocks in participating computers to Coordinated Universal Time (UTC). UTC is based on the solar day, which depends on the rotation of Earth about its axis [15]and which was also defined as part of the Gregorian calendar introduced by Pope Gregory XIII in October 1582.

3.1 NTP levels

At the top level of NTP is Stratum 0. In this stratum, there are not many machines. Typically they are ground-based cesium clocks or the cesium clocks on satellites of geographical navigation systems such as the Global Positioning System (GPS) which was launched by the U.S. Department of Defense in 1978.

There is also the Galileo Global Navigation Satellite System (GNSS) launched by the European nations, the BeiDou Navigation Satellite System (BDS) launched by China and GLONASS which is owned by the Russian Federation. Devices connected with Stratum 0 cesium clocks and satellites are on the next level at Stratum For example, according to ntp.org there were approximately 2000 NTP servers in Stratum 1 across Europe at the time of writing this thesis. Following this are Stratum 2 servers which are clients of Stratum 1 and then according to the servers on stratum N+1 are clients to servers on Stratum N. The maximum stratum level is 16. This system of strata is presented in Figure 6.

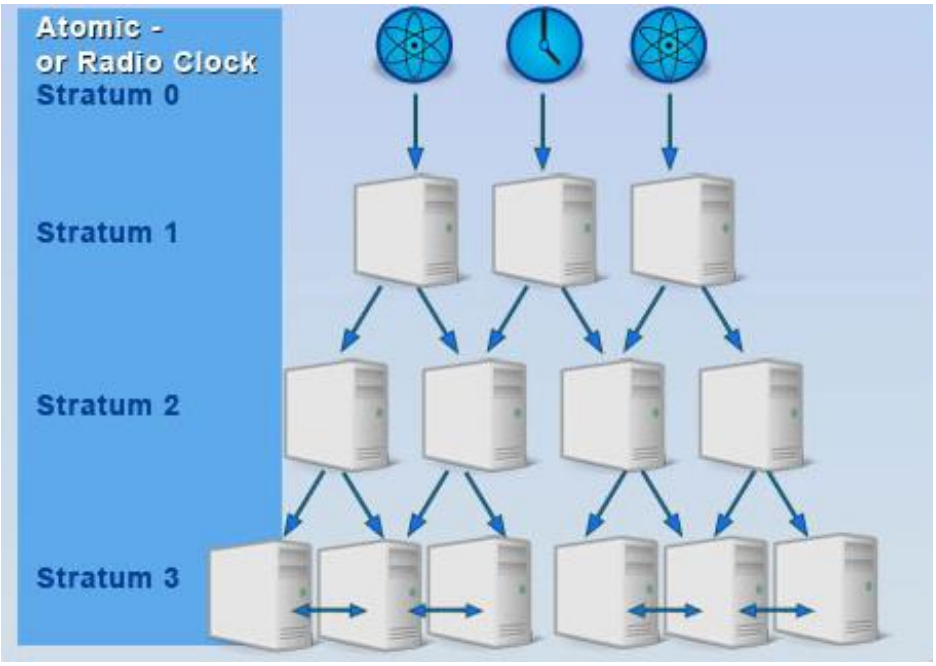


Figure 6: ***NTP server Stratums hierarchy***

3.2 Global Positioning System

The Global Positioning System, commonly known as GPS, provides satellite navigation for vehicles and much more. Each navigation system connects with a group of satellites that have a fixed constellation. These systems use trilateration [16] or other techniques to determine the position of the GPS receiver. They also determine absolute time using the UTC format and broadcast the UTC timestamp for synchronization. [17]

Each GPS satellite carries:

- Stabilized stratum 0 atomic clock hardware
- Advanced location tracking circuitry
- Transmitters that constantly broadcast their position and clock time

3.2.1 Leap second

The speed of the Earth's rotation is slowing down over time due to many complex factors. This means that the solar day increases in length by approximately 0.001 seconds per solar day. [18]

The rate at which the earth slows down however is not uniform. There are complex factors behind this such as massive sea currents causing tidal acceleration, the movement of the fluid metallic core inside the Earth or changing atmospheric currents the result of these dynamic factors is that the need for leap seconds isn't predictable.

Once the difference between UTC and UTC1 reaches 0.9 seconds, then 1s is added to UTC to compensate for this difference.
GPS time is eighteen seconds ahead of UTC, because GPS time was set to match UTC in 1980, but has since diverged. Since the UTC format was implemented earlier than the GPS format there was already added 9 leaps- second to UTC before GPS was equal, so if nine seconds is subtracted from twenty seven we get an offset of eighteen seconds.

3.3 Protocols

3.3.1 NMEA protocol

The NMEA standard has been in use since 1983 and most GPS receivers use the standard call 0183 version which can support 4800 or 9600 b/s on the serial interface RS232. USB converters also can be used but they can possibly introduce unnecessary latencies in communication between a PC and a GPS receiver. NMEA consist of sentences in American Standard Code for Information Interchange (ASCII) format like in figure. ASCII format. [19]

```
$--GGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x, M,x.x,M,x.x, xxxx*hh<CR><LF>
```

	GGA - Global Positioning System Fix Data
1	$--: Talker identifier
2	GGA: Sentence formatter*
3	hhmmss.ss: UTC of position*
4	llll.ll, a: Latitude North/South (N/S)*
5	yyyyy.yy, a: Longitude East/West (E/W)*
6	x: GPS quality indicator
7	xx: Number of satellites in use (00-12)
8	x.x: Horizontal dilution of precision
9	x.x, M: Antenna altitude above/below mean sea level (geoid), meters*
10	x.x, M: Geoidal separation, meters
11	x.x: Age of differential GPS data
12	xxxx: Differential reference station ID
13	*hh: Cheksum*

Table7: *NMEA protocol in ASCII format*

3.3.2 TSIP protocol

The TSIP is a binary packet protocol option for the user to maximize control over the configuration of a GPS receiver which can also optimize performance. TSIP supports over 20 commands associated with response packets. GPS receivers with the TSIP protocol can also correct an offset of PPS if it reaches a critical threshold - it switches to recovery mode and automatically synchronizes PPS with the GPS. TSIP packet structure is the same for both commands and reports. [20] The packet format is:

<DLE> <id> <data string bytes> <DLE> <ETX>

- <DLE> is the byte 0x10
- <ETX> is the byte 0x03
- <id> is a packet identifier byte, which can have any value excepting <ETX> and <DLE>.

An example packet:
0x10,
0x8F, 0xAB,
0x00, 0x03, 0xE5, 0x1F, 0x06, 0x88, 0x00, 0x00,
0x0C, 0x17, 0x36, 0x16, 0x18, 0x01, 0x07, 0xDC,
0x10, 0x03

*Figure 7**: TSIP protocol in binary format***

3.3.3 NTP Packet header structure

The NTP server is stateless and responds to each received client NTP packet in a simple transactional manner by adding fields to the received packet and passing the packet back to the original sender, without reference to preceding NTP transactions. Upon receipt of a client NTP packet, the receiver timestamps the receipt of the packet as soon as possible within the packet assembly logic of the server. The packet is then passed to the NTP server for processing.

The NTP packets sent by the client to the server and the responses from the server to the client use a common format, as shown in table 8. [21]

0-1	3-4	5-7	8-15	16-23	24-31	bit
LI	VN	MODE	STRATUM	POLL	PRECISIOM	
ROOT DELAY (32-bit)						
ROOT DISPERSION (32-bit)						
REFERENCE IDENTIFIER (64-bit)						
REFERENCE TIMESTAMP (64-bit)						
ORIGIN TIMESTAMP (64-bit)						
RECEIVE TIMESTAMP (64-bit)						
TRANSMIT TIMESTAMP (64-bit)						

***Table 8**: NTP Packet structure*

Packet header overview	
LI	Leap Indicator (2 bits).
VN	NTP Version Number (3 bits).
MODE	NTP packet mode
STRATUM	Stratum level of the time source (8 bits).
POLL	The log2 value of the maximum interval between successive NTP messages in seconds.
PRECISION	Clock precision (8-bit signed integer) The precision of the system clock, in log2 seconds.
ROOT DELAY	The total round-trip delay from the server to the primary reference.
ROOT DISPERION	The maximum error due to clock, only in server messages.
REFERENCE ID	For stratum 1 servers this value is a four-character ASCII code that describes the external reference source.
REFERENCE TIMESTAMP	System clock was last set or corrected, in 64-bit time-stamp format.
ORIGIN TIMESTAMP	Time at which the request departed the client for the server.
RECEIVE TIMESTAMP	The time at which the client request arrived at the server.
TRANSMIT TIMESTAMP	The time at which the server reply departed the server.

Table 9: *NTP Packet structure description* [21]

3.3.4 NTP server packet diagram

Basic operating principles of network time protocol are described below. These four parameters are passed into the client timekeeping function to drive the clock synchronization function. [22]

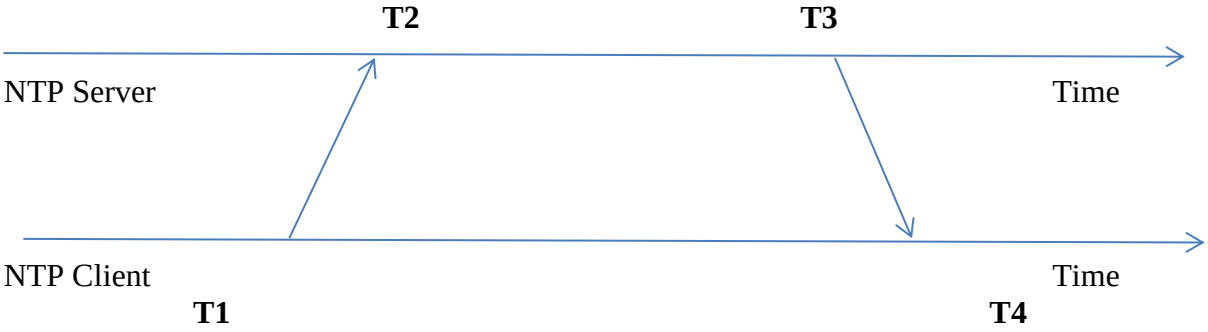


Figure 8: ***NTP server packet diagram***

NTP protocol principles	
T2-T1	The Client requests NTP server to provide NTP service (including network transmission time).
T3-T2	NTP server processing time
T4-T3	the NTP server replies to Client with NTP service information (including network transmission time
T4-T1	total spending time
T4-T3 to T4-T2	The time delay from Client to NTP server is between

Table 10*.*

3.3.5 NTP packet data traffic.

From the NTP server packet diagram we can estimate an approximate delay between the client and the server. NTP client polls data using a polling frequency algorithm from the NTP server. The NTP clock algorithm selects the best server with best sample data with smallest delay and offset. These values are calculated from last four timestamps. [1]:

Round-trip delay during NTP exchange of packets between client and server can be represented as [1]:

$$\theta = ½ [(T2 – T1) + (T3 – T4)] \text{ and } \delta = (T4 – T1) – (T3 – T2)$$

*Figure 9: **NTP clock offset (**θ**) round trip delay (**δ**) formula***

3.3.6 NTP Reference timestamp

The reference timestamp is stored as a 64-bit unsigned integer. The first part of N bits contains the number of seconds and the next M bits contain the number of milliseconds. So, the timestamp is composed of 32 bits for representation of seconds passed since 0 hours on 1 January 1900 and 32 bits to represent fractions of seconds, as is shown in Figure 10. The multipliers to the right of the point are 1/2, 1/4, 1/8, 1/16, etc. [23].

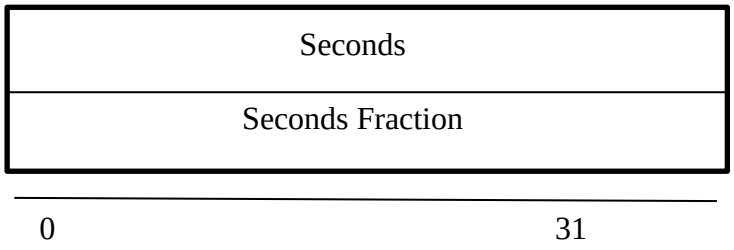


*Figure 10: **Reference timestamp structure***

Many OSs are operating with this time format; therefore this may cause problems on January 1, 2036. At that moment the 64-bit unsigned integer will overflow and a new epoch will start. The user will face two problems. They will receive the date 1900-01-01 00:00:00 UTC, not 2036-02-07 06:28:15 as the new time. As a result of this, UNIX time will fail to start. [24]

3.3.7 UNIX timestamp

UNIX timestamp is a single a signed 32-bit number which increments every second in most of our Unix OS and started from since 00:00:00 Thursday, 1 January 1970. UNIX time stamp is calculated and calibrated from NTP reference time stamp UTC by LINUX kernel [25].

The UNIX timestamp has a 70 year offset from the NTP timestamp as it started later in 1970. The UNIX timestamp calculation from NTP timestamp:

NTP offset = (year offset*365 days + 17 leaps seconds)*86400s = 2208988800s
UNIX timestamp seconds = NTP timestamp – NTP offset
UNIX timestamp seconds fraction = (NTP timestamp fraction * 1000000)/ 2^{32}

*Figure 11: **UNIX timestamp calculation formula.***

4. NTP Daemon

NTP daemon software is operating in Unix, VMS and Windows systems and others under NTP service using selected Reference clock drivers for synchronizing system clock. It is simple application running in the background and using UDP protocol and port 123 [1].

At the time of writing, the most recent NTP version was NTP release 4.2.8 p12. The command ntpq transfer to simple command, insert –help all option will display.

4.1 Reference Clock Driver

A reference clock driver requires reference a GPS signal which is transferred to the clock hardware and I/O interface processed by serial ASCII timecode [26] to NMEA or binary TSIP output.
Then driver performs certain filtering and grooming functions. The entire suite of NTP algorithms is available to filter the received data, select the best clocks or servers, and combine their offsets to synchronize the system clock [1].

The system clock timestamp is captured at a designated on-time character in the timecode. The difference between the driver timestamp and system timestamp represents the clock offset. [1] page 133.

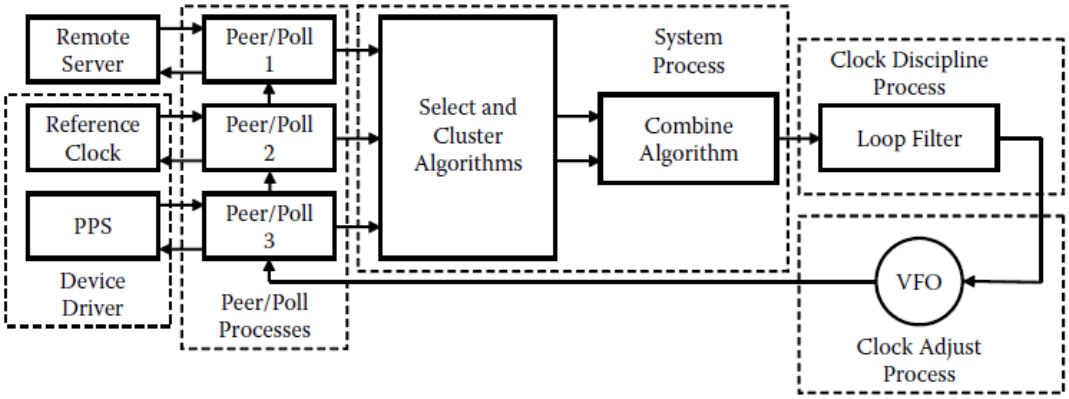


Figure 12: ***Device driver.*** [1] page 133

Each specific driver to each reference clock must be compiled in the kernel modules, but most common clocks like NMEA, TSIP drivers are included already by default. Drivers have address in **"/etc/ntp.conf "**in the form "127.127.t.u". (-t driver type, -u unite number 0-3) List of most common drivers is provided on web [27].

In this project, with selected standard serial RS-232 protocol for communication with each specific device. A symbolic link must be created for each driver and device combination that is used.

The "server" command in the first line of figure 13 configures a "reference clock" and "fudge" command in second line provides additional information like address argument specifies the clock address, the refid and stratum options control can be used to override the defaults for the device. The NMEA sample of the setting is analyzed below and each parameter is described in table 14.

```
server 127.127.20.0 mode 16 minpoll 4 maxpoll 4 iburst prefer
fudge  127.127.20.0 refid NMEA time1 + 0.105 flag2 0 refid NMEA stratum 1
```

Figure 13: ***Reference clock driver configuration example***

Mode	Specific mode number which in figure presents NMEA clock driver serial port communication speed of 9600 baud(9600 bit per seconds.
minpoll and maxpoll	Specify the min. and max. polling interval for reference clock messages in $\log_2$
Prefer	Marks the reference clock as preferred.
iburst	Iburst mode sends up ten queries within the first minute to the NTP server otherwise by default 1 per second.
time1	Driver specific constant added to the time offset produced by driver, a fixed-point decimal number in second.
flag2	Flags customizing the clock driver specifies the PPS signal on-time edge: 0 – rising edge
Refid	Specifies and ASCII string from one four characters.
Stratum	Specifies stratum number assigned to the driver in the range 0 to 15
driftfile	Record frequency of local clock oscillator, updated one per hour.

Table 11*: Reference clock driver configuration example*

4.2. PPS and Interface and driver

A pulse per second (PPS) is produced usually by 10 MHz signal like Cesium, Rubidium clock or quartz oscillators or any other laboratory equipment. This 10MHz signal is processed by divider and then by the end of divider is obtained 1 pulse per second. The PPS signal is aligned to the roll-over of seconds and synchronized with UTC as shown in figure 14. The PPS output is transferred into data carried control(DCD) line RS232 interface to provide a hardware interrupt input for synchronization and DCD RS232 pin is utilized as a PPS input from the GPS receiver. The combination of GPS timing data and PPS output we can achieve highly accurate timing reference. [28]

The NMEA or any other reference clock driver associated with the PPS signal calls the PPSAPI application programming interface to measure difference between the system clock and PPS signal and process it in median filter. In our case PPSAPI calls kernel case to process the PPS signal directly, PPS is consider being valid only if the offset produced by clock filter, selected cluster and combine algorithm less than 400ms. Please see figure 14 [1].

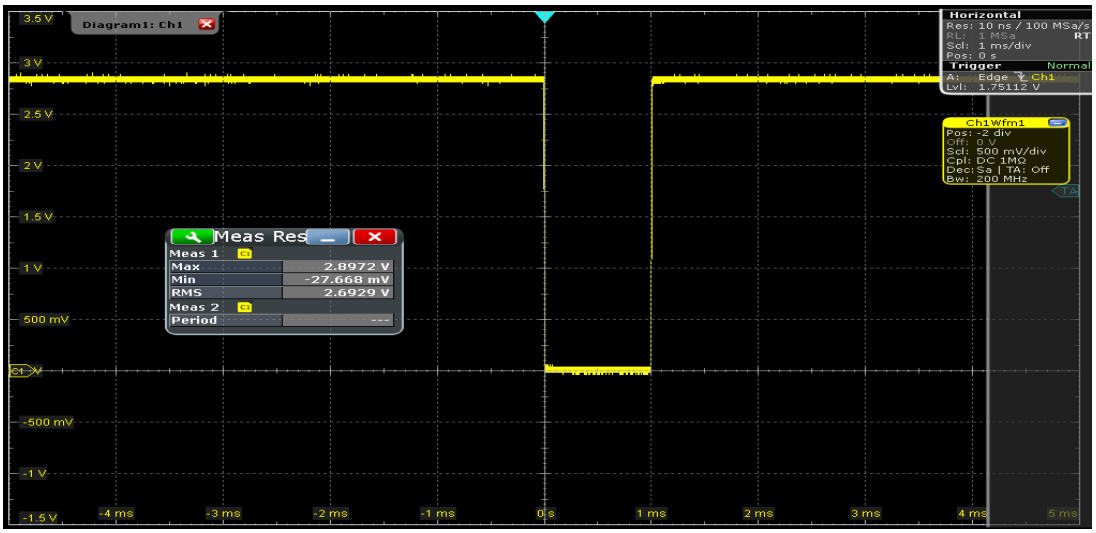


Figure 14: ***Pulse per second (PPS)***

4.3. GPSD and SHM driver

Global Position Daemon driver (GPSD) is a daemon that receives data from a GPS receiver and then transmitting these data to share memory (SHM) driver of ntpd. SHM driver receives its reference clock info from a shared memory-segment function in ntpd per every second interval [27].

4.4 Network time protocol configuration “ntp.conf”

This section describes set up of reference clock drivers used in our experiment. We have tested PPS API NMEA, SHM NMEA GPSD, and SHM TSIP GPSD reference drivers.

4.4.1 PPS API NMEA

The NMEA messages are connected to the system via a serial port interface named gps0 and PPS signal is wired through the DCD pin and get process in serial port interface gpspps0. GPSD service must be stopped, otherwise driver will not work correctly.

ln -sf /dev/ttyAMA0 /dev/gps0
ln -sf /dev/pps0 /dev/gpspps0
server 127.127.20.0 mode 18 minpoll 4 maxpoll 4 prefer
fudge 127.127.20.0 flag1 1
fudge 127.127.20.0 flag2 0
fudge 127.127.20.0 flag3 1
fudge 127.127.20.0 time2 0.470

```
pi@raspberrypi:~ $ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
oGPS_NMEA(0)     .GPS.            0 l   16   16  377    0.000    0.003   0.001
+clock2.infonet. .PPS0.           1 u   33   64  177    4.213    0.030   0.042
+ntp-2.arkena.ne 193.190.230.65   2 u   25   64  377   41.128    0.123   0.386
+dnscache-paris2 145.238.203.14   2 u   41   64  377   46.375    1.874   1.180
```

Figure 15: ***PPS API NMEA driver***

4.4.2 SHM NMEA GPSD reference clock

The NMEA messages connected to the system via a serial port gps0 using GPSD service and time stamp obtained from the share memory driver.

#ln -sf /dev/ttyAMA0 /dev/gps0
Kernel-mode PPS reference-clock for the precise seconds
server 127.127.22.0 minpoll 4 maxpoll 4
fudge 127.127.22.0 refid kPPS
Coarse time reference-clock - nearest second
server 127.127.28.0 minpoll 4 maxpoll 4 iburst prefer
fudge 127.127.28.0 time1 +0.105 flag1 1 refid GPSD stratum 1

```
pi@raspberrypi:~ $ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
oPPS(0)          .kPPS.           0 l   16   16  377    0.000    0.001   0.001
*SHM(0)          .GPSD.           1 l   15   16  377    0.000   -4.345   0.685
```

*Figure 16: **SHM NMEA reference clock***

4.4.3. SHM TSIP GPSD reference clock

The TSIP messages connected to the system via a serial port palisade0 using GPSD service and time stamp obtained from the shared memory driver.

ln -sf /dev/ttyAMA0 /dev/palisade0

Kernel-mode PPS reference-clock for the precise seconds
server 127.127.22.0 minpoll 4 maxpoll 4
fudge 127.127.22.0 refid kPPS

Coarse time reference-clock - nearest second
server 127.127.28.0 minpoll 4 maxpoll 4 iburst prefer

```
pi@raspberrypi:~ $ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
oPPS(0)          .PPS.            0 l    1   16  377    0.000   -0.008   0.002
*SHM(0)          .GPSD.           1 l   16   16  377    0.000  -13.995  10.189
```

*Figure 17: **SHM TSIP reference clock***

4.4 NTP Stats

The network time protocol daemon stores statistics in local file system /var/log/ntpstats. The file contains following parameters: loopstats, peerstat,clockstats. [29]

To collect statistics, add the directory path to the configuration file “ntp.cong”

statsdir /var/log/ntpstats/
statistics loopstats peerstats clockstats

Peerstats file contains clock offset and jitter data for every valid clock filter update for the server, peer and clock driver.

Row data in peerstat file

48773 10847.650 127.127.4.1 9714 -0.001605376 0.000000000 0.001424877 0.000958674

Item	Units	Description
48773	MJD	Date
10847.650	S	Time past midnight
127.127.4.1	IP	Reference clock address
9714	Hex	status word
-0.001605376	S	clock offset
0.000000000	S	roundtrip delay
0.001424877	S	Dispersion
0.000958674	S	RMS jitter

Table 12.

The "loopstats" log lists offset and jitter values for every valid discipline algorithm [1] update. The delay and clock offset counted by formula provided in 2.3.3.

Row data in loopstats

50935 75440.031 0.000006019 13.778190 0.000351733 0.0133806 6

Item	Units	Description
50935	MJD	Date
75440.031	S	Time past midnight
0.000006019	S	Clock offset
13.778	PPM	Frequency offset
0.000351733	S	RMS jitter
0.013380	PPM	RMS frequency jitter
6	$Log_{2\ s}$	Clock discipline loop time constant

Table 13.

4.4.1 Modifies version of the Julian date (MJD)

A modified version of the Julian date, denoted as MJD, can be obtained by subtracting 2,400,000.5 days from the Julian date JD. The MJD therefore gives the number of days since midnight on November 17, 1858. [30]

$$MJD \equiv JD - 20400000.5$$

Figure 18: ***Modified version of the Julian date***

The **Julian Day** number (**JD**) is the count of the number days that have elapsed since Greenwich Mean Noon on 1 January -4712 (4713 BC) in the Julian Proleptic Calendar.

The Julian start date is calculated from three following cycles Solar, Lunar and Roman Indication from the last time these three events have been coincident. This time format provides astronomers measure secular time differences over long time.

The number 20400000.5 where 0.5 days is subtracted so as to have MJD start a midnight and the 2400000 is used to reduce the 7-digit day number of JD to a more tractable 5 digits format for the logs and statistics.

The both time format MJD and JD are reference to UTC which is time format used as primary time standard in the words and it is recognized by the IAU, ITU and CCIR [31].

4.4.2 NTP plotter beta V1.0.32.075

NTP Plotter is a freeware program for the plotting of graphs for the NMEA/PPS ref-clock driver. It produces graphs of offset, frequency error and jitter from loopstat log files which ntpd produces. [32]

5. Experiments and a new method of measuring the accuracy of an NTP server.

For these experiments, it was assumed that an atomic clock was precise enough and so it was used as an absolute reference for time synchronization. For this purpose, the atomic clock GPS-89 Rubidium, located at Tartu Observatory, was used. This atomic clock provides for this purpose 10 MHz: Sine wave, > 0.6 Vrms in 50 Ω. It generates PPS approximately 5V in open output in 50 Ω duty cycle 20% with 60 nanosecond precision relative to UTC.

In ideal conditions, an atomic clock can provide a theoretical precision of 10^{-11} per second. By ideal conditions, it is meant that all oscillation frequencies of the oscillators on the build platform are synchronized and are kept in a constant environment. In such conditions, this system can possibly measure at even higher levels of PPS precision if required.

If it is considered that the atomic clock with 60 nanosecond accuracy is reasonable for a PPS signal output. This precise output is sufficient input for the NTP protocol as NTP software ntpd can support higher accuracy than 1μs. For networks, a timestamp of 1μs accuracy is appropriate as signals travel long distances through cables and routers and a higher accuracy cannot be expected.

NTP daemon programs are often used to provide higher accuracy timestamps for local networks. NTP statistics software is often used for generating statistics for variables such as clock offset and jitter.

How can these be used to verify the accuracy of an NTP system? Using statistics software we can use loopstat or peerstat files to assess variables such as offset and jitter between the system timestamp and device driver stamp that have been produced from reference clocks like GPS and PPS pulses. The question still remains though how accurate are these data, and are there any alternatives for assessing the time accuracy of an NTP server?

One possible method of verifying the accuracy of PPS pulses is published on the internet. The suggestion is to create the NTP Client and then generate PPS pulses on the parallel port of a GPIO pin. In this scenario, one single-board computer is used which is sharing all the resources for both the NTP server and client while also running two different NTP protocols at the same time. This is not the best approach as each program is consuming a lot of

CPU/MCU resources on both the server and client. Such a system would almost certainly be influencing any given results.

The method proposed in this thesis is to build a separate client and server on separate single-board computers and assign them each a single core for the ntpd service in order to achieve a higher level of accuracy of the NTP timestamp. The NTP client will generate a new PPS by the script introduce in appendix C and capture the system timestamp. Then new generated PPS will be compared to the original physical PPS produced by the atomic clock on the oscilloscope. These signals will be connected to separate channels and will give us the possibility to analyze differences between them. The original PPS produced by the GPS module will not be taken into account as it is assumed that a PPS provided from an atomic clock is the most precise reference.

An addition to this, physical PPS signals from an atomic clock and an additional NTP server was also examined in our LAN.

Using this novel method, the author will attempt to verify how well an NTP server is operating and how accurate a timestamp it can provide.

5.1 Experiment phases

Firstly, an NTP server was built. Connected to the GPIO input of this server was a highly accurate PPS output from a GPS-98 atomic clock with a PPS accuracy of 60ns. The GPS receiver then delivered data in ASCII or BINARY format for the reference clock driver through the serial port on the single-board computer. The combination of these devices was expected to result in a highly accurate timestamp generated by this device driver.

The second stage was to build an NTP client connected to a 1Gbps ethernet LAN switch.

Thirdly, PPS driver software was developed that could be added to the LINUX kernel as a custom module. This driver software executes a function of "**gettimeofday()"** via a system call to the kernel. The function obtains the current time, expressed as seconds and microseconds synchronized with UTC, and stores it in the "**timeval structure"** pointed to by "**tp"** variable.

Following this, simple C++ script was created that initiates the driver module. The rising edge of the new second occurs while the reading and writing on the device "**callback function**" is triggered. A pulse of 500us length is generated and then delivered to the GPIO output of the NTP client.

Once this has been established, then both PPS outputs will be connected to the Rohde and Schwarz oscilloscope on Channel 1 and Channel 2. The mathematical channel will be set up for collecting statistics and drawing graphs. A simple overview of this experimental setup can be seen in Figure 19.

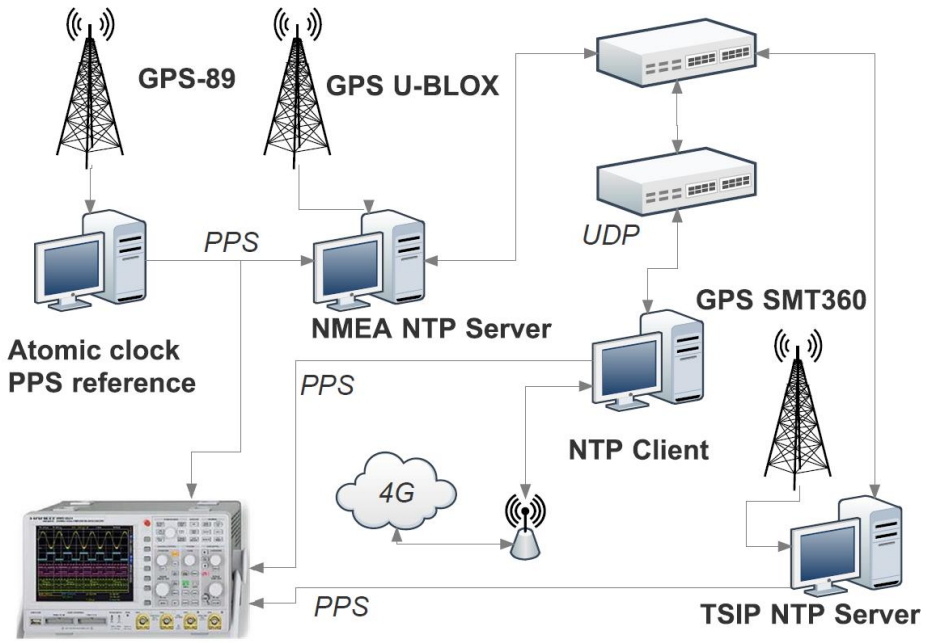


Figure 19: ***Simple overview of experiment***

Finally, three different configurations will be tested and analyzed for performance:

1. In first set up a NanoPi NEO2 acts as the NTP server as per requirement was to set up speed of one Giga bit per second over the LAN. A PPS pulse is generated from the atomic clock and GPS receiver SMT360™ was connected to transmit the NMEA data through a serial interface order to lock PPS. Raspberry Pi 3B acts as the NTP client which synchronizes with the NTP server. It will generate its own PPS by the script and it's synchronized with UTC timestamp. The first setup has been running over the one common switch connected in LAN. Both PPS signals generated from the following configurations and analyzed on the oscilloscope on the separated channels:
    a. Raspberry and NanoPi NEO has default setup, no enhanced parameters are applied.
    b. Raspberry and NanoPi NEO has optimized setup. One separate core was utilized only for ntpd running task and CPU was set up to higher frequency.

2) In second setup a Raspberry Pi 3B acts as the NTP server with 1 PPS generated from Atomic clock and new module GPS receiver U-blox MAX-M8Q was installed to transmit NMEA data through serial interface to lock PPS. Raspberry Pi acts as NTP client synchronize with NTP server and generating its own PPS by script synchronized with UTC timestamp. Now onwards only enhanced setup will be used as in test 1.b. Both PPS signal from NTP server and client are compared on the oscilloscope on the separate channels for further analysis:
    a. Both Raspberries Pi's setup up over the single switch in LAN.
    b. Both Raspberries Pi's setup up over the multiply switches in LAN
    c. Raspberries Pi NTP server set up under stress of data traffic.
    d. Test over the WLAN 4G

3) Additional NTP server NanoPi NEO2 with TSIP protocol will be added to the existing NTP server running on the Raspberries Pi with NMEA protocol for comparing of TSIP and NMEA protocols. Both PPS are connected to separate channels on the oscilloscope and offset is observer on mathematical channel.

5.2 Experiment result

The Rohde and Schwarz RTO 1014 oscilloscope from Tartu Observatory was used to produce histograms and statistics. The PPS signal from an atomic clock was installed on channel 1 (CH1) of the oscilloscope and the signal from the NTP client was installed on channel 2 (CH2). A mathematical channel was also set up and the formula shown in Figure 19 was applied. The time recorded from the positive edge of the PPS on CH1 was subtracted from the time of the PPS's positive edge recorded from CH2. This configuration makes it possible to observe the time difference between the client and server PPSs. This reveals how well the NTP client is synchronized with the NTP server. This method provides the opportunity to verify data produced by ntpd and determine the accuracy of the NTP server on the oscilloscope. Most of the measurements were conducted in ten hours window.

Oscilloscope settings:
Mode Run continuously
Horizontal resolution 200ps 5GSa/s
Trigger set up to 2.4 V
Horizontal scale 20us/div
Vertical scale CH1 4V/div
Vertical scale CH2 1.62V/div

Table 14.

Measure Result = |CH1 –CH2|
On the positive edge of PPS

Figure 20: ***Oscilloscope formula***

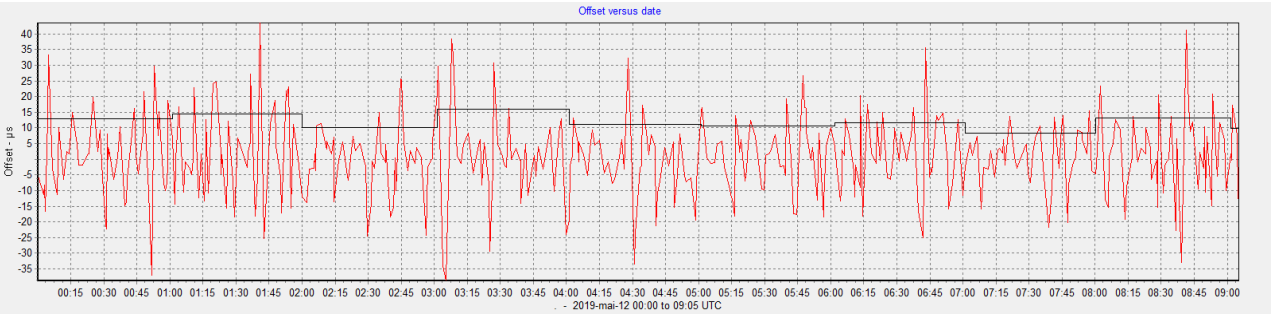
5.2.1. NanoPi NEO2 NTP server and Raspberry Pi 3B NTP client

The following result was obtained from the NTP client. The PPS pulse was generated by a pulse generator driver and a 100µs offset was deducted from the positive time edge of the PPS pulse. This adjustment is optional but it was used to keep the oscilloscope on the 20µs scale.

5.2.1.1

1 a) Measurement by NTP plotter NTP server Nanopi NEO2

From the loopstats produced by NTP plotter we can conclude that the clock offset between the system timestamp and the driver timestamp on the NTP client was approximately 1µs (Figure 20). In this case, the NTP client has provided an average of 1µs accuracy on the NTP timestamp.



Loopstats | Peerstats | Setup

Table | Offset | Delay | Dispersion | Jitter

Peer	Number of entries	Mean Offset (ms)	RMS Offset (ms)	Minimum Offset (ms)	Maximum Offset (ms)	Mean Delay (ms)	Minimum Delay (ms)	Maximum Delay (ms)	Mean Dispersion (ms)	Mean RMS jitter (ms)
172.17.73.8	435	0,001	0,012	-0,039	0,044	0,424	0,357	0,450	0,639	0,013

Figure 21: ***NTP plotter NTP client peer offset***

5.2.1.2

1 a) Measurement by the new technique NTP server Nanopi NEO2

The experimental configuration however requires a different approach to measurement. The PPS generated by the NTP client from the timestamp of the experimental system is compared to a PPS produced directly from an atomic clock. The time offset recorded by this approach was on average 58 µs. This included the 100µs which was mentioned in section 5.2.1.

Using the oscilloscope, it was seen that the difference between the system timestamp and the moment a new PPS was generated was 58 µs. If an estimated small pulse delay of 4 µs is taken into account then this is an unexpected result. Small delay comes from time when the script is executed to generate pulse till the time when pin is set up high on the GPIO.

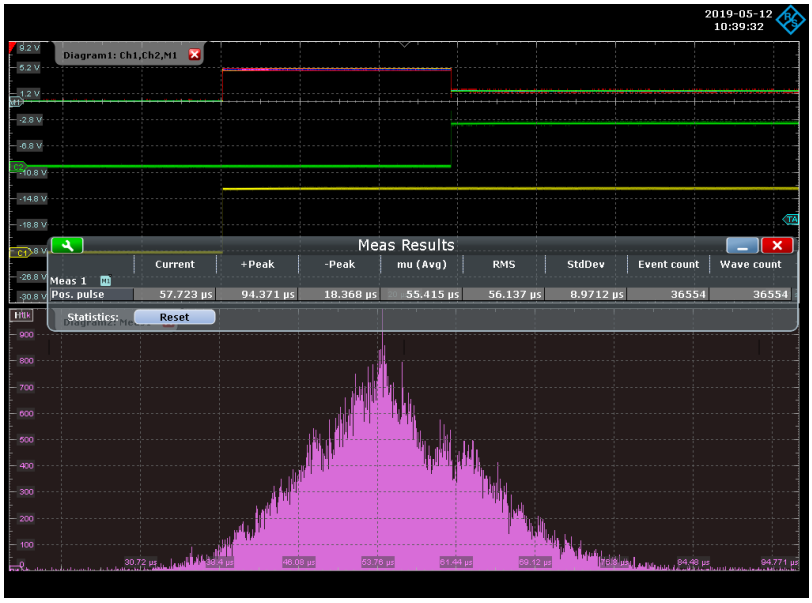


Figure 22: ***Nanopi NEO2 and SMT360 default settings with 100us adjustment.***

5.2.1.3

1 b) Measurement by the NTP plotter NTP server Nanopi NEO2

The same scenario was tested with modified settings where ntpd had its own dedicated core and the frequency of the CPU was set to 816MHz. Results presented on the NTP plotter (Figure 22) show that in this case, the NTP client has an average offset of 2µs compared with the NTP timestamp.

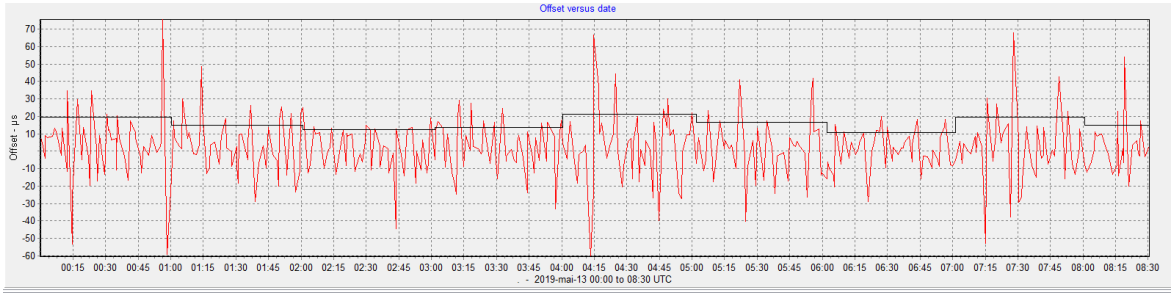


Table | Offset | Delay | Dispersion | Jitter

Peer	Number of entries	Mean Offset (ms)	RMS Offset (ms)	Minimum Offset (ms)	Maximum Offset (ms)	Mean Delay (ms)	Minimum Delay (ms)	Maximum Delay (ms)	Mean Dispersion (ms)	Mean RMS jitter (ms)
172.17.73.8	401	0,002	0,017	-0,060	0,076	0,428	0,292	0,477	0,652	0,033

Figure 23: ***NTP plotter NTP client peer offset***

5.2.1.4

<u>1 b) Measurement by the new technique NTP server Nanopi NEO2</u>

In the modified configuration, we can see there is a 7µs improvement. The normal distribution curve improved slightly too. However, from the results it can be concluded that NTP data are not as accurate as was expected.

This is because NTP data reflect different types of data to those we are generating with the experimental technique. The NTP plotter software provides average offset and jitter data of the NTP client based on the system timestamp and driver timestamp. The experimental method compares instead the system timestamp of the NTP client with the PPS generated by an atomic clock. However, this should not have as significant an impact on the results as has been observed.

One of the main reasons why results differ is that NTP plotter is getting data from log files generated by ntpd which contains samples only per minute, therefore it is not best tool to measure data for the short period of the time. By proposed method we analyse samples produced per second. This makes measurement by this new technique much more accurate.

Figure 24: ***Nanopi NEO2 and SMT360 enhanced settings with 100us adjustment.***

5.2.2 Raspberry Pi 3B NTP server and client enhanced setting

5.2.2.1

<u>2 a) Measurement by the NTP plotter</u>

Results from the NTP plotter show that the time precision of the NTP client has improved significantly. But it is not immediately clear that they have improved by more than two times. Statistics reveal again that the NTP client has an average accuracy of 1µs compared with the NTP timestamp and from the graphic (Figure 24) we can observe that the scale offset on the clock has decreased.

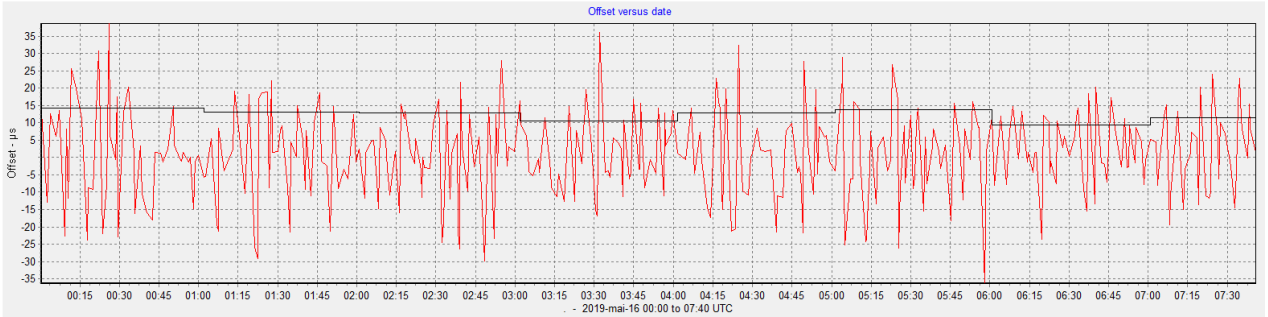


Table | Offset | Delay | Dispersion | Jitter

Peer	Number of entries	Mean Offset (ms)	RMS Offset (ms)	Minimum Offset (ms)	Maximum Offset (ms)	Mean Delay (ms)	Minimum Delay (ms)	Maximum Delay (ms)	Mean Dispersion (ms)	Mean RMS jitter (ms)
172.17.72.229	369	0,001	0,012	-0,036	0,039	0,621	0,570	0,673	0,660	0,030

Figure 25: ***NTP plotter NTP client peer offset***

5.2.2.2

2 a) Measurement by the new technique

Using this configuration, large improvements in accuracy were recorded. In fact, the NTP client's accuracy improved so much that the 100µs offset from the script had to be removed. This is likely to be because the clock offset and jitters are minimized meaning that more accurate results are able to be captured. If the pulse delay of 4µs is subtracted from the average result of 18µs then an accuracy of 14 µs was achieved when compared to the atomic clock reference (Figure 25).

Figure 26: ***Raspberry Pi 3B and GPS Expansion Board with 4us adjustment and enhanced performance.***

5.2.2.3

2 b) Measurement by the NTP plotter & by the new technique

The results recorded when using multiple LAN switches were not very different to those obtained in previous tests and did not bring any new information.

5.2.2.4

2 c) Measurement by the NTP plotter under stress test

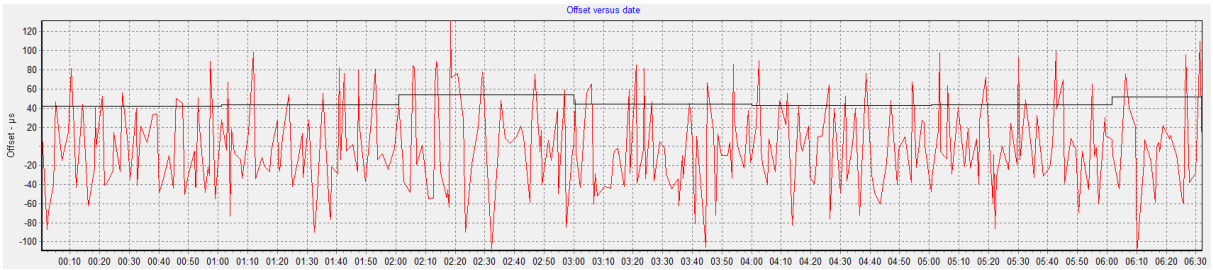
The stress test was set up as follows. Random amounts of packets, in the range between 2000-3000 and with a set up polling rate of 6, were transmitted from the NTP client to the NTP server. The formula in figure 26 estimates approximately 15 000 clients served on the NTP server. The variable can be changed inside of the program, detail stated in Appendix E. The NTP server's performance did not drop and its accuracy did not decrease as this would require stronger load (see Figure 28), However, NTP client accuracy was impacted quite a lot as you can observe in Figure 27. Measurements of the NTP client were not affected by the generation

of packets to the NTP server as the task of packet generation was assigned to an additional NTP client running on a separate unit in LAN.

(Number of Packets per Second) x (Polling Rate) = Number of Clients.

Test stress formula

Figure 27: ***Number of the clients formula***



Loopstats | Peerstats | Setup

Table | Offset | Delay | Dispersion | Jitter

Peer	Number of entries	Mean Offset (ms)	RMS Offset (ms)	Minimum Offset (ms)	Maximum Offset (ms)	Mean Delay (ms)	Minimum Delay (ms)	Maximum Delay (ms)	Mean Dispersion (ms)	Mean RMS jitter (ms)
172.17.72.229	325	0,000	0,046	-0,109	0,132	0,448	0,352	0,534	0,666	0,050

Figure 28: ***NTP plotter NTP client peer offset***

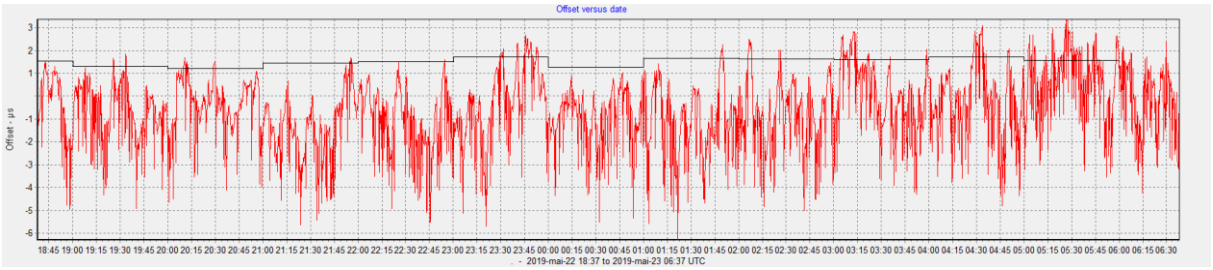
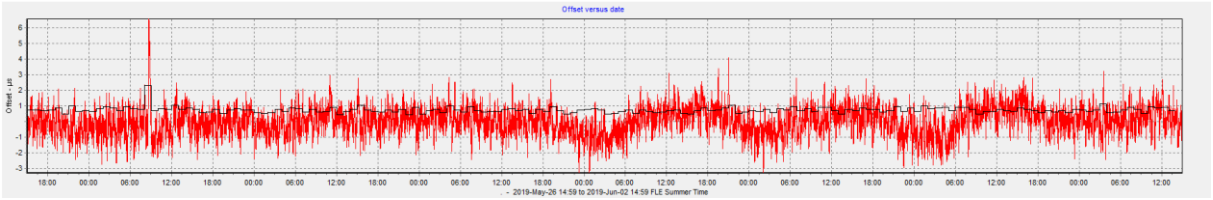


Figure 29: ***NTP plotter NTP server peer offset after 10 hours***



Loopstats | Peerstats | Setup

Table | Offset | Delay | Dispersion | Jitter

Peer	Number of entries	Mean Offset (ms)	RMS Offset (ms)	Minimum Offset (ms)	Maximum Offset (ms)	Mean Delay (ms)	Minimum Delay (ms)	Maximum Delay (ms)	Mean Dispersion (ms)	Mean RMS jitter (ms)
127.127.20.0	45900	0.000	0.001	-0.003	0.007	0.000	0.000	0.000	0.233	0.001

Figure 30: ***NTP plotter NTP server peer offset after 100 hours***

5.2.2.5

2 c) Measurement by the new technique under stress test

Looking at the measurement obtained by the new technique under stress test it is obvious that accuracy has suffered. Normal distribution curve has on both sides from the average some unexplained peak spikes. It could be explained by UDP packets got loss on the NTP client side once client tried to pull data from NTP server on the port 123. Accuracy of the individual client has reduced from 18 µs to 52 µs. Results show that if the NTP server is under a large stress, accuracy on the NTP timestamp is reduced. This can however be solved by setup in "**ntp.conf**" file by setting up different parameters which control traffic data.

Figure 31: ***Raspberry Pi 3B and GPS Expansion Board with 4us adjustment and enhanced performance under stress.***

5.2.2.6

2 d) Measurement by the NTP plotter & by the new technique

This measurement was not possible to conduct using a 4G WLAN connected through a mobile **H**uawei p20. The delay between the client and server fluctuated between 20ms and 150ms. As a result of this, ntpd could not lock onto the signal and the NTP client did not synchronize with the NTP server. Unfortunately, measurement data were therefore corrupted and did not provide any usable results. It seems that a 4G network has difficulty synchronizing with ntpd. The ntpd reported around 40ms offset over a short period of time but after an hour of testing, the time offset began changing in a volatile manner, so it was decided to end this test. It is possible that devices connected via a 4G mobile network have a generally poor UTC timestamp. Further testing outside the scope of this thesis would be required to assess this, however.

5.2.3 Two Raspberry Pi 3B NTP servers

The purpose of this test was to reveal how two PPS signals from separate NTP servers will perform in the same LAN. The setting on the oscilloscope was the same as in previous tests. The Raspberry Pi 3B computer received the PPS signal from the atomic clock and the GPS receiver U-blox MAX-M8Q transmitted the NMEA data. Previous testing revealed that of the two computers tested, the Raspberry Pi 3B has a better NTP timestamp accuracy. For the following test, it was decided to connect a second Raspberry Pi 3B to the GPS receiver SMT360 with TSIP protocol and so use this as the second NTP server. The second PPS was produced by this older GPS module. The NanoPi NEO2 was used as the NTP client, but didn't generate any PPS output. Display output of this test from the PuTTY interface is included in Appendix G.

5.2.3.1

Measurement by the NTP plotter

This result shown in Figure 29 display client synchronization with two different NTP servers in a local LAN. The pink line (172.17.72.229) represents results from the NTP server using an NMEA input and connected with the atomic clock. The blue line (172.17.72.186) represents the results of an NTP server using a TSIP protocol timestamp. From the results, we can conclude that the NMEA module had a slightly better accuracy. The results also show however that when the TSIP time offset reaches a critical threshold, the PPS corrects itself by GPS receiver as it has been designed to do. Outside of these periods of long delays however, the TSIP module generally performed better than the NMEA protocol as it has binary format which is better for data processing and also it can be observed that some section of the graph has better curve than NMEA. But the as it can be observe from figure 29 new NMEA module has very good performance too. One reason why these new GPS module with u-blox performing so good is that they have also implemented quantization error on the PPS.

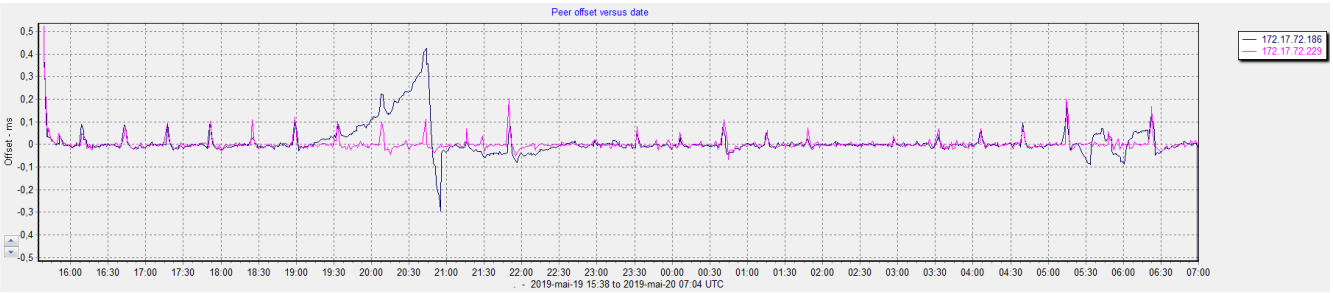


Table | Offset | Delay | Dispersion | Jitter

Peer	Number of entries	Mean Offset (ms)	RMS Offset (ms)	Minimum Offset (ms)	Maximum Offset (ms)	Mean Delay (ms)	Minimum Delay (ms)	Maximum Delay (ms)	Mean Dispersion (ms)	Mean RMS jitter (ms)
172.17.72.186	763	0,004	0,241	-6,378	0,521	0,387	0,343	0,769	20,873	0,028
172.17.72.229	707	0,004	0,039	-0,066	0,523	0,416	0,365	0,791	22,121	0,021

Figure 32: ***Peer offset of two physical NTP servers***

5.2.3.2

3) Measurement of two physicial PPS by oscilloscope

Results of these tests revealed that the two NTP servers installed on the same LAN had very similar levels of time accuracy (Figure 30). The resulting offset on average was 17ns while the average jitter was 13ns when compared with the atomic clock PPS as the reference. That is actually quite a good result for an old GPS module which has a guaranteed precision of 16ns for a PPS signal.

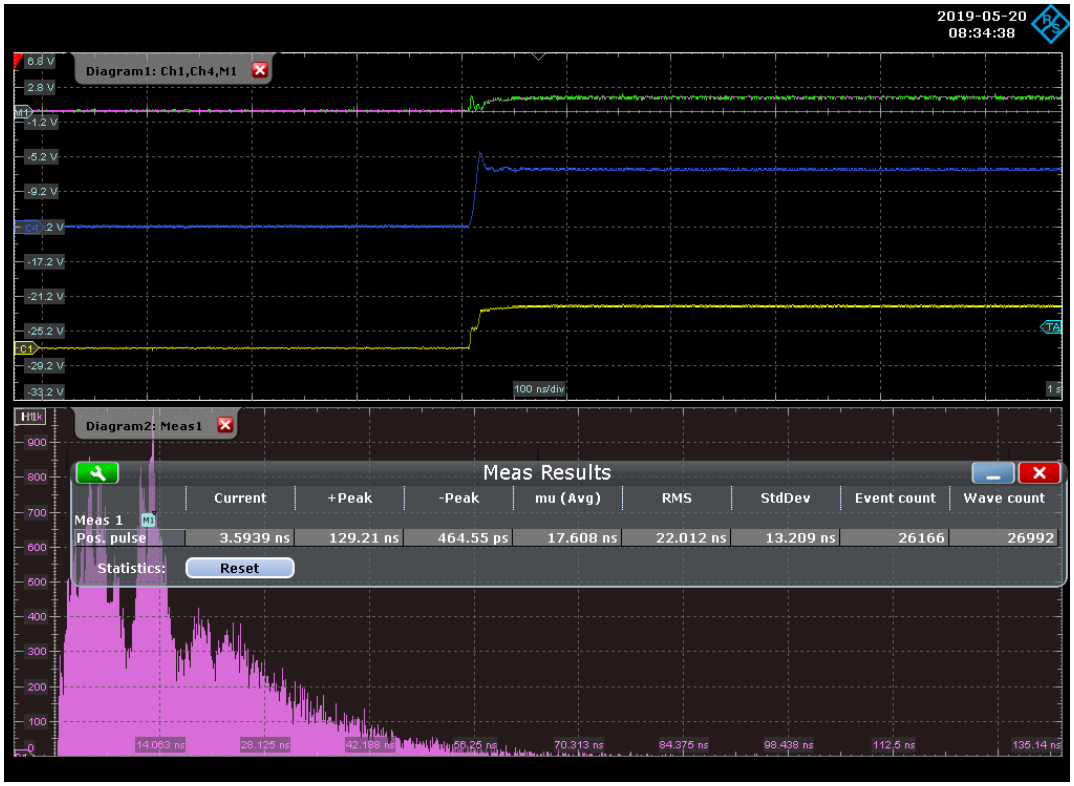


Figure 33: ***TWO Raspberry Pi 3B NTP servers with two separates physical PPS .***

6. Evaluation Potential and Improvements

Once testing was complete it could be clearly seen that Raspberry Pi 3B is a better choice for building an NTP server. NanoPi NEO2 has difficulties in processing the signal from a GPS module. One possible reason is that it is not built for such complex tasks as being an NTP server. It seems NanoPi NEO2 also has some limitations due to power saving. This is because it has no heating or cooling mechanism and it has been manufactured for the completion of medium level tasks. This therefore creates a possibly limitation on the GPIO frequency and this in turn could explain why the offset was fluctuating and causing jitter to be unstable on the PPS and clock driver. With a few enhancements and the right setup, NanoPi NEO2 is likely able to perform in the role of an NTP server as well as Raspberry Pi3.

During this project, it was discovered that the PPS which is coming from the atomic clock is not synchronized with the rubidium clock. Only 10MHz output is synchronized to the rubidium standard. The atomic clock has its own GPS module similar to what is used in this

thesis and from GPS module comes the PPS signal straight to PPS out. This could possibly be solved by using 10MHz as the MCU clock PLL frequency on the STM32F4 discovery board. Then it would be possible to use the timer EXT TRIG input to start counting from the PPS timing edge and generate a pulse which would synchronize by the rubidium clock and then it provide even better precision on the PPS than achieve now.

Another possibility how to improve slightly the accuracy of NTP server would be to run software on the atomic clock and set up the length of the cable of GPS antenna.

On the measurement obtain from the experiment it is obvious that data from NTP stats and plotter are not reflecting accuracy of the NTP server in the best manner. There could be added some additional component to NTP software which would compensate observed offset between the atomic clock and NTP client and therefore get more an accurate improvement on the timestamp on the client side.

Also, one more missing measurement could be completed, if the client is connected through the general WLAN. From the unofficial result it was observed by the proposed measurement following offset of 450µs when delay between client and server was approximately 7-8ms.

7. Applications

This stable NTP server was built for use with the atomic clock at Tartu Observatory and is already in use there. The NTP server developed through this project distributes a UTC timestamp with a maximum accuracy of 1-2µs on the NTP server side and for local device synchronization within 15-50µs accuracy as was assign per requirements.

8. Conclusion

The NTP server built for Atomic Clock at Tartu Observatory has almost fulfilled all the objectives of this master thesis. The performance was studied by conducting experimental measurements. During the process, it was discovered that Raspberry Pi3 B is a more suitable option for an NTP server than NanoPi NEO2. The new NTP server is able to synchronize all NTP clients in a LAN within 50µs accuracy of the UTC timestamp as per the stated requirements. Following these experiments, it is better understood how these NTP servers can be improved for better performance. Stress tests could be performed in a variety of ways to ensure that the system can serve many clients at the same time. The question also remains why the accuracy on the timestamp for the NanoPi NEO2 was less efficient than Raspberry Pi3 B. Further analyses need to be undertaken in order to discover the cause. The current NTP server computer at Tartu Observatory will be replaced with a more powerful Raspberry Pi3 B+ as it provides a better bandwidth, supporting 300 Mbps, and will be able to serve more clients. Finally, once proper housing will be received, it is necessary to install NTP server unit according to standard safety rules for electronic devices and also network security should carefully be set up.

9. Acknowledgements

I would like to thank my supervisors Viljo Allik and Indrek Sünter for all their support and knowledge they shared with me. They truly demonstrated their professionality in their field and they were guiding me through the whole project at all the time. Thank you to whole Tartu Observatory for gaining new knowledge about space and technology. Many thanks to Ilmar Ansko for helping me with Linux commands and giving me useful tips, information and support. Big thanks to Geoffrey Longhurst for checking the English grammar and his interest in these analyses. Huge thanks to my family and friends for the support during this master thesis.

10. Kokkuvõte

Tartu observatooriumis aatomi kella jaoks ehitatud NTP-server on osaliselt täitnud selle projekti eesmärgid. Töö käigus avastati, et Raspberry Pi3 B on sobivam valik NTP serverile kui NanoPi NEO2. Uus NTP server suutis sünkroniseerida LAN-i seadmete UTC ajastandardiga täpsusega 50 μs, nagu nõutud. Pärast tehtud mõõtmisi on parem mõista, kuidas antud NTP-servereid parema jõudluse saavutamiseks parandada. Stressteste saab läbi viia mitmel viisil, et tagada süsteemi samaaegne teenindamine paljudele klientidele. Töö käigus ilmnes, et NanoPi NEO2 aja täpsus oli kehvem võrreldes Raspberry Pi3 B-ga ja seetõttu ei olnud võimalik tagada 1 Gbit / s Etherneti võrgu võimekust. Põhjuse avastamiseks tuleb teha täiendavaid analüüse. Tartu observatooriumi praegune NTP-serverarvuti asendatakse võimsama Raspberry Pi3 B + -ga, kuna see tagab parema kohaliku Etherneti võrgu võimekuse, toetades kuni 300 Mbit / s, millega saab teenindada rohkem kliente. Lõpetuseks, kui nõuetekohane seadme korpus on teostatud, on vajalik paigaldada NTP-server vastavalt standardtingimustele ja võrgu turvalisus tuleb hoolikalt seadistada.

Bibliography

[1] D. L. Mills, "The Network Time protocol on the earth and in space," in *COMPUTER NETWORK TIME SYNCHRONIZATION*, 6000 Broken Sound Parkway NW, Suite 300, CRC Press is an imprint of Taylor & Francis Group, an Informa business, 2011, p. 438.

[2] H. E. ,. S. ,. K. S. I. ,. O. ,. K. J. T. E. Iaroslav Iakubivskyi, "ESTCube-2 plasma brake payload for effective deorbiting," in *7th European Conference on Space Debris*, 2017.

[3] N. C. Limited., "UPUTRONICS," Nevis Computers Limited, 2012. [Online]. Available: https://store.uputronics.com/.

[4] "DesignSpark PCB is a free-of-charge schematic capture and PCB layout tool for electronics design automation," RS Components is a trading brand of Electrocomponents PLC, 2010," A. E. &. R. components, 2010. [Online]. Available: https://www.rs-online.com/designspark/home.

[5] E. David Taylor, "Welcome to David & Cecilia Taylor's Web pages," 2019. [Online]. Available: https://www.satsignal.eu/ntp/Raspberry-Pi-NTP.html.

[6] "NanoPi NEO2 Armbian," 2019. [Online]. Available: wiki.friendlyarm.com/wiki/index.php/NanoPi_NEO2.

[7] J. L. P. L. A. M. R. M. E. U. David Braben, "Raspberry Pi 3 Model B," Raspberry Pi Foundation, 2016. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b.

[8] Trimble, "https://www.trimble.com," Trimble, 1978. [Online]. Available:

http://trl.trimble.com/docushare/dsweb/Get/Document-733639/.

[9] Trimble, "www.trimble.com," Trimble, [Online]. Available: http://trl.trimble.com/docushare/dsweb/Get/Document-550718/GPSStudioUG_1B_August2011.pdf.

[10] UPUTRONICS, "UPUTRONICS," Limited., Nevis Computers, 2012. [Online]. Available: https://store.uputronics.com/files/HAB-GPSPI+-ASSY.pdf.

[11] u.-b. i. a. S. company, "https://www.u-blox.com/en," u-blox is a Swiss company, 1997. [Online]. Available: https://www.u-blox.com/sites/default/files/MAX-M8-FW3_DataSheet_%28UBX-15031506%29.pdf.

[12] u.-b. i. a. S. company, "u-blox.com," u-blox is a Swiss company, 1997. [Online]. Available: https://www.u-blox.com/sites/default/files/products/documents/u-blox8-M8_ReceiverDescrProtSpec_(UBX-13003221)_Public.pdf.

[13] C. Lisdat, "The most precise atomic clock ever made is a cube of quantum gas," *New Scientist Mazine,* no. DOI: 10.1126/science.aam5538, 5 10 2017.

[14] Spectracom, "manuals.spectracom.com," Spectracom, 1972. [Online]. Available: https://spectracom.com/sites/default/files/document-files/pendulum-gps-88_89_gps-controlled-frequency-standards.pdf.

[15] David W. Allan Neil Ashby Clifford C. Hodge, "The Science of Timekeeping," in *Application Note 1289*, USA, Hewlett-Packard Company, 2019, p. 88.

[16] Lifewire, "What Is Trilateration," Lifewire, 10 2016. [Online]. Available: https://www.lifewire.com/trilateration-in-gps-1683341.

[17] Masterclock, "www.masterclock.com," Masterclock, 2019. [Online]. Available: https://www.masterclock.com/support/library/gps-network-time-synchronization.

[18] R. A. Nelson, D. D. McCarthy, S. Malys J. Levine, B. Guinot, H. F. Fliegel, R. L. Beard and T. R. Bartholomew, "The leap second: its history and possible future," *metrologia,* no. Metrologia, 2001, 38, 509-529, p. 21, 1 5 2019.

[19] Marport, "www.marport.com," Marport, 2019. [Online]. Available: http://www.marport.com/doc_web/PSY/topics/r-WinchNMEA.html.

[20] T. N. Limited, "www2.etown.edu/," Trimble Navigation Limited, 1999. [Online]. Available: http://www2.etown.edu/wunderbot/DOWNLOAD/AgGPS114/TSIP%20Reference%20Manual%20-%20Rev%20C.pdf.

[21] CISCO, "www.cisco.com," CISCO, 6 2012. [Online]. Available: https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-58/154-ntp.html.

[22] N. T. University, "ccnet.ntu.edu.tw," National Taiwan University, 2003. [Online]. Available: https://ccnet.ntu.edu.tw/english/ntp.html.

[23] D. Payette, "www.ntp.org," Network Time Protocol, [Online]. Available: http://www.ntp.org/ntpfaq/NTP-s-algo.htm.

[24] t. f. e. From Wikipedia, "en.wikipedia.org," 4 2019. [Online]. Available: https://en.wikipedia.org/wiki/Year_2038_problem.

[25] L. Colorado, "stackoverflow.com," Luis Colorado, 2015. [Online]. Available: https://stackoverflow.com/questions/29112071/how-to-convert-ntp-time-to-unix-epoch-time-in-c-language-linux/29138806.

[26] T. C. C. f. S. D. Systems, "public.ccsds.org," The Consultative Committee for Space Data Systems, 10 2010. [Online]. Available: https://public.ccsds.org/Pubs/301x0b4e1.pdf.

[27] D. L. Mills, "www.ntp.org," Network Time Protocol, [Online]. Available: http://doc.ntp.org/4.1.2/refclock.htm.

[28] D. Evans, "www.aplu.ch," [Online]. Available: http://www.aplu.ch/home/gpsclock.html.

[29] N. T. Protocol, "www.nwtime.org," Network Time Protocol, 7 9 2015. [Online]. Available: https://www.nwtime.org/wp-content/uploads/2016/04/NTP-Handbook.pdf.

[30] t. f. e. From Wikipedia, "en.wikipedia.org," Wikipedia, 11 5 2019. [Online]. Available: https://en.wikipedia.org/wiki/Julian_day.

[31] S. Allen, "www.astronomy.ohio-state.edu," 2011. [Online]. Available: http://www.astronomy.ohio-state.edu/~pogge/Ast350/timesys.html.

[32] D. Taylor, "www.satsignal.eu," 1 1 2014. [Online]. Available: https://www.satsignal.eu/software/net.htm.

[33] R. S. Connell, "github.com," 2016. [Online]. Available: https://github.com/rascol/Raspberry-Pi-PPS-Client.

[34] H. Bruun, "www.ntp.567.dk," 24 1 2018. [Online]. Available: http://www.ntp.567.dk/index.php?link=guides&link2=NTPtesttools.

Appendices

Appendix A GPS External Board PCB Layout

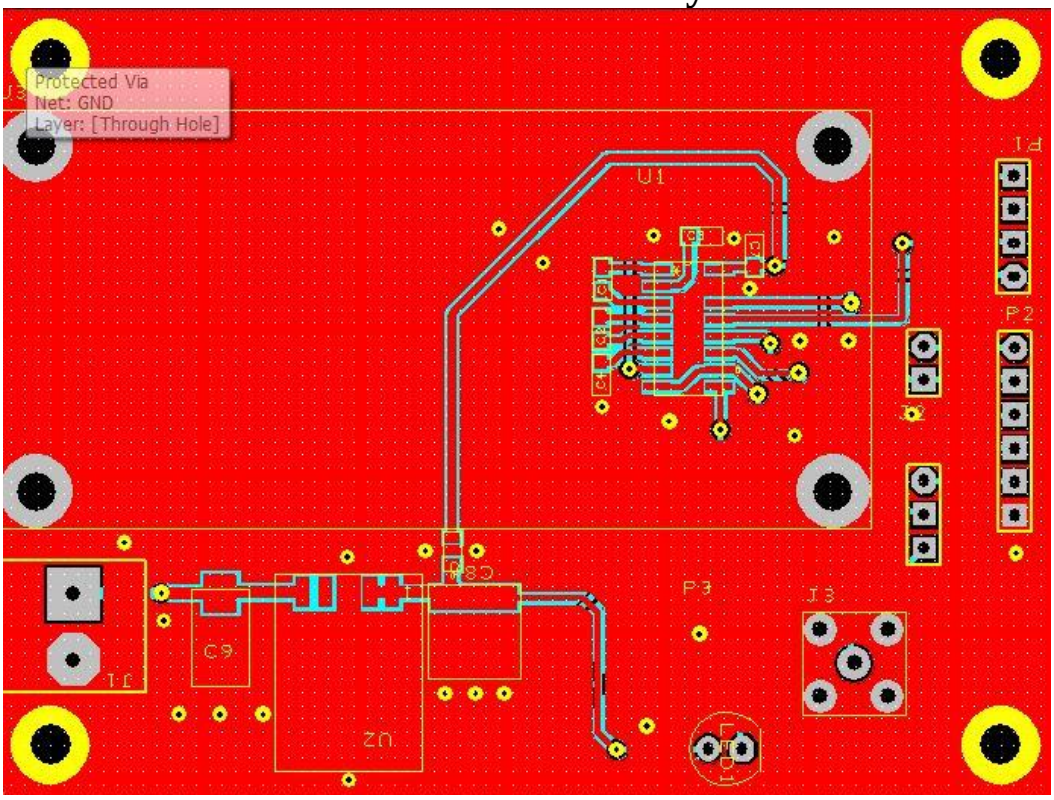


Figure 34: ***External GPS PCB board***

Appendix B NanoPI NEO2 Server Installation

Please follow instructions below for installation:

1) SD card Formatter https://www.sdcard.org/downloads/formatter_4/
2) Win32 Disk Imager https://www.askvg.com/win32-disk-imager-write-any-bootable-image-to-usb-drive-in-windows/
3) Download Armbian_5.65_Nanopineo2_Ubuntu_bionic_next_4.14.78 from https://dl.armbian.com/nanopineo2/archive/
4) Format and install this image to Sandisk SD card min 16GB
5) Install Putty https://www.putty.org/
6) Connect to NEO2 as user: **root**, password **1234** , new username: miroNEO2
7) Install "**WINscp**" https://winscp.net/eng/download.php
8) # nano /boot/armbianEnv.txt
9) Add additional lines in line 4,5 into armbianEnv.txt bold:

    i. verbosity=1
    ii. console=both
    iii. overlay_prefix=sun50i-h5
    iv. overlays=uart1 pps-gpio
    v. param_pps_pin=PA6
    vi. rootdev=UUID=ee092379-cddf-4aff-beb3-7cc62d0fe9bd
    vii. rootfstype=ext4
    viii. usbstoragequirks=0x2537:0x1066:u,0x2537:0x1068:u

10) # sudo reboot

11) Check if PPS is registered in Kernel:

```
root@nanopineo2:~# dmesg | grep pps
[    0.349310] pps_core: LinuxPPS API ver. 1 registered
[    0.349314] pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo Giometti
<giometti@linux.it>
[    4.185158] pps pps0: new PPS source pps@0.-1
[    4.185226] pps pps0: Registered IRQ 63 as PPS source

root@nanopineo2:~# ls -l /dev | grep pps
crw------- 1 root root    252,   0 Jan  9 17:19 pps0
```

12) Install pps-tools for PPS verification

```
root@nanopineo2:~# apt install pps-tools
root@nanopineo2:~# ppstest /dev/pps0
```

13) Install Libpcap provides a portable framework for low-level network monitoring. Libpcap can provide network statistics collection, security monitoring and network debugging:

```
root@nanopineo2:~# apt-get install libcap-dev
```

14) Now standard NTP package requires to be removed from the system.

```
root@nanopineo2:~# /etc/init.d/ntp stop
[ ok ] Stopping ntp (via systemctl): ntp.service.
root@nanopineo2:~# apt remove ntp
```

15) Download and unpacked latest NTP daemon in our case ntp-4.2.8p12

```
root@nanopineo2:~# cd ~
root@nanopineo2:~# wget get http://archive.ntp.org/ntp4/ntp-4.2/ntp-4.2.8p12.tar.gz
root@nanopineo2:~# tar zxvf ntp-4.2.8p12.tar.gz
root@nanopineo2:~# cd ntp-4.2.8p12
```

16) New NTP daemon compiling procedure, now it is important that all necessary flags are selected during compiling. In our scenario we need NMEA and TSIP(PALISADE) flags for selected protocols and we compile with setting –j4 as nanopi has 4 cores.

```
root@nanopineo2:~/ntp-4.2.8p12# ./configure --enable-linuxcaps --enable-ATOM --enable-NMEA --enable-ipv6 --enable-PALISADE --enable-all-clocks
root@nanopineo2:~/ntp-4.2.8p12# make –j4
root@nanopineo2:~/ntp-4.2.8p12# make install
```

17) Finally custom build of NTPD is was installed into /usr/local/sbin/ntpd, but default location of NTPD is usr/sbin/ntpd, so order to run ntp service correctly we need to create following links:

```
root@nanopineo2:~/ntp-4.2.8p12# ln -s /usr/local/bin/ntpd /usr/sbin/ntpd
root@nanopineo2:~/ntp-4.2.8p12# ln -s /usr/local/bin/ntpdc /usr/bin/ntpdc
root@nanopineo2:~/ntp-4.2.8p12# ln -s /usr/local/bin/ntpq /usr/bin/ntpq
root@nanopineo2:~/ntp-4.2.8p12# ln -s /usr/local/bin/ntpsweep /usr/bin/ntpsweep
root@nanopineo2:~/ntp-4.2.8p12# ln -s /usr/local/bin/ntptrace /usr/bin/ntptrace
```

18) Linux UBUNTU this specific build requires to activate the service as it is still inactive after installation of this custom NTPD build, maybe not needed in Debian version.

```
root@nanopineo2:~# service ntp status
● ntp.service
 Loaded: masked (/dev/null; bad)
Active: inactive (dead) ntpq -p

root@nanopineo2:~# systemctl unmask ntp.service
root@nanopineo2:~# service ntp enable
root@nanopineo2:~# service ntp start
root@nanopineo2:~# /etc/init.d/ntp restart
root@nanopineo2:~# ntpq -crv –pn
```

19) Custom installation NTP daemon has created ntp.conf in following directory /run/ntp.conf.dhcp, this file has to be deleted as it is causing problem. Router is assigning NTP information in DHCP which pi using over /etc/ntp.conf. The file in /etc/init.d/ntp file will also require some small adjustments.

```
root@nanopineo2:~# rm /run/ntp.conf.dhcp
root@nanopineo2:~# rm /etc/dhcp/dhclient-exit-hooks.d/ntp
```

20) Firstly we modify etc/init.d/ntp so that after every reboot ntp service will initialize from /etc/ntp.conf.

Secondly we create symlink for NMEA or TSIP protocol so that after every restart of system symlink will be created automatically.
Time can updated time after every reboot by command ntpdate -u and hwclock –w.

root@nanopineo2:~# nano /etc/init.d/ntp
Modify lines 25 and 26 as follows:

if [-e /run/ntp.conf.dhcp]; then →
if **[** -e /etc/ntp.conf]; then

NTPD_OPTS="$NTPD_OPTS -c /run/ntp.conf.dhcp" →
NTPD_OPTS="$NTPD_OPTS -c /etc/ntp.conf"

Add these lines into structure case$1 in ntp and close the file

```
log_daemon_msg "Initializing SMLINK" "ntpd"
ln -s /dev/ttyS1 /dev/gps0 #NMEA
#ln –s /dev/ttyS1 /dev/palisade0 #TSIP
#stty -F /dev/ttyS1 9600 cs8 -cstopb parodd parenb raw #TSIP
#stty -F /dev/ttyS1 9600 cs8 -cstopb parodd parenb  #NMEA
log_daemon_msg "Pre-synchronizing time" "ntpd"
ntpdate -u ee.pool.ntp.org
hwclock -w
log_daemon_msg "Starting NTP server" "ntpd"
```

root@nanopineo2:~# reboot

21) The next is setting of ntp.conf file for NMEA, TSIP and SHM these lines have to be added into etc/ntp.conf.

```
#Pulse Per Second ATOM clock
server 127.127.22.0 minpoll 4 maxpoll 4
fudge 127.127.22.0 refid PPS

# TSIP (PALISADE) The Primary reference direct setting without GPSD service
server 127.127.29.0 # Trimble Palisade GPS (Stratum 1).
# Set packet delay
fudge 127.127.29.0 time1 0.020
# and set flag2 to turn off event polling.
fudge 127.127.29.0 flag2 1

#NMEA direct setting without GPSD service
server 127.127.20.0 mode 16 minpoll 4 maxpoll 4 prefer
fudge  127.127.20.0 refid NMEA

# SHM(share memory driver) time reference-clock - nearest second supported by GPSD service
server 127.127.28.0 minpoll 4 maxpoll 4 iburst prefer
fudge 127.127.28.0 time1 +0.535 flag1 1 refid GPSD stratum 1
```

```
#Server for the reference
server 2.ee.pool.ntp.org
server 2.europe.pool.ntp.org
server 0.europe.pool.ntp.org

#Additional logging for debugging
logfile /var/log/ntp
logconfig =all
```

22) GPSD service installation

```
root@nanopineo2:~# apt-get update
root@nanopineo2:~# apt-get install gpsd
root@nanopineo2:~# apt-get install gpsd-clients

root@nanopineo2:~# nano /etc/default/gpsd

        START_DAEMON="true"
        USBAUTO="false"
        DEVICES="/dev/ttyS1"
        GPSD_OPTIONS="-n"

root@nanopineo2:~# reboot
```

23) Optionally, the aforementioned symlinks can be created using udev rules. For example, the following creates a symlink /dev/gps0 that points to /dev/ttyS1 with read-write access to all users:

```
root@nanopineo2:sudo nano /etc/udev/rules.d/10-gps.rules
KERNEL=="ttyS1",SYMLINK+="gps0",MODE="0666"
KERNEL=="pps0",MODE="0666"
```

24) Useful command, GPSD service helps for diagnosis:

```
root@nanopineo2:~# service gpsd stop
root@nanopineo2:~# gpsd -N -n -D 3 /dev/palisade0
root@nanopineo2:~# gpsd -n -N -D4 /dev/palisade0
```

25) Other Useful command for debugging using GPSD service

```
root@nanopineo2:~# ls –lah
root@nanopineo2:~# ls –lah /dev/gps0
root@nanopineo2:~# sudo stty -a -F /dev/ttyS1
root@nanopineo2:~# chown ntp:ntp /var/lib/ntp/
root@nanopineo2:~# ntptime
```

Appendix C NTP Client PPS driver generator installation

When NTP server is ready next task would be establish PPS client by adding kernel module to the kernel. This requires solving some few points.

1. Order to use system function of kernel it is necessary to install custom kernel image.
2. Once done client PPS driver generator will be programmed in C and added as module to custom kernel.
3. When driver added to kernel small script in C++ requires to be programmed to execute driver in command line.
4.

Recompiling kernel

1) First before we recompile kernel, let's check if the system is up to date. Custom kernel build was tested with Linux raspberrypi 4.14.98-v7+ [33]

~ $ sudo apt-get update
~ $ sudo reboot

2) Let's setup builder folder in home file
~ $ mkdir rpi
~ $ cd rpi

3) Get missing decencies
~ /rpi $ sudo apt-get install bc

4) Check if you have installed git:
~ $ sudo apt-get install git

5) For retrieving the Linux source get the rpi-source script:
~/rpi $ sudo wget https://raw.githubusercontent.com/notro/rpi-source/master/rpi-source -O /usr/bin/rpi-source && sudo chmod +x /usr/bin/rpi-source && /usr/bin/rpi-source -q --tag-update

6) Run the script to download the Linux source that matches the installed version of Linux on your RPi:
~/rpi $ rpi-source -d ./ --nomake –delete

7) On the RPi,
~/rpi $ cd linux
~/rpi/linux $ KERNEL=kernel7
~/rpi/linux $ make bcm2709_defconfig

8) Recompile kernel
~/rpi/linux $ make -j4 zImage

9) Download PPS driver with scrip from https://github.com/ryzic/IMPULSE.git and follow instruction for installation.

Appendix D Single-computer board performance optimization

NanoPi NEO2 CPU isolation and taskset

1) For the NanoPi NEO2 requires **“isolcpus”** command to be added at the very end of /boot/boot.cmd , isolating of fourth core can be done with argument **isolcpus=3.**

   setenv bootargs "root=${rootdev} rootwait rootfstype=${rootfstype} ${consoleargs} hdmi.audio=EDID:0 disp.screen0_output_mode=${disp_mode} panic=10 consoleblank=0 loglevel=${verbosity} ubootpart=${partuuid} ubootsource=${devtype} usb-storage.quirks=${usbstoragequirks} ${extraargs} ${extraboardargs} **isolcpus=3**"

2) Then the boot script has to be recompiled by following command:
   root@nanopineo2:~# mkimage -C none -A arm -T script -d /boot/boot.cmd /boot/boot.scr
   root@nanopineo2:~# reboot

3) Checking if isolation command of core was successful:
   root@nanopineo2:~# sudo cat /sys/devices/system/cpu/isolated

4) Taskset function to assing isolated to core to NTP PID displayed in top command.
   root@nanopineo2:~# sudo taskset -pc 3 714

5) For displaying CPU ID launch `htop` from the command line. Press <F2> key, go to "Columns", and add PROCESSOR under "Available Columns.

Raspberry Pi 3B CPU isolation and ROOT user

1) For the NanoPi NEO2 requires **“isolcpus”** command to be added at the very end of /boot/cmdline.txt, isolating of fourth core can be done with argument **isolcpus=3**
   pi@raspberrypi:~ $ sudo nano /boot/cmdline.txt

   wc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=/dev/mmcblk0p7 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait splash plymouth.ignore-serial-consoles **isolcpus=3**

   pi@raspberrypi:~ $ sudo reboot

2) Repeat step 3,4,5 as in NanoPi NEO2 in section 3.5.1.
3) Adding user root on Raspberry you have to change password and add “**PermitRootLogin yes**” at the end of the file /etc/ssh/sshd_config and reboot

   pi@raspberrypi:~ $ sudo passwd root
   pi@raspberrypi:~ $ sudo nano /etc/ssh/sshd_config

CPU frequency adjustment

Adjust serial connection latency and set up the highest frequency of the CPU. []

```
# echo "performance" > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
setserial /dev/ttyAMA0 low_latency
# echo FREQ > /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq
```

Appendix E Traffic test [34]

1) Install TCP dump for NTP packets monitoring .
   pi@raspberrypi:~ $ sudo apt-get install tcpdump
   pi@raspberrypi:~ $ tcpdump udp port 123

2) Download little c program ntpload3.c
   pi@raspberrypi:~ $ gcc ntpload3.c -o ntpload3
   pi@raspberrypi:~ $ ./ntpload3 192.xxx.x.xxx

3) These lines below has to be commented out in ntp.conf for allowing traffic to go through.
   #restrict -4 default kod notrap nomodify nopeer noquery
   #restrict -6 default kod notrap nomodify nopeer noquery

Appendix F Sample code

C function, the PPS driver generates 1 PPS 500 µs length at the each fractional second by timeout on defined GPIO output on the single board computer through system call function in kernel gettimeofday(&tv).

```
/* pulse-generator.c
 * Created on: May 01, 2019
 * Copyright (C) 2019
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program; if not, write to the Free Software Foundation, Inc.,
 * 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * pulse-generator.ko must be copied to
 *  /lib/modules/`uname -r`/kernel/drivers/misc/pulse-generator.ko
 */
/**
 * Asserts a 500 usec pulse at the fractional second time
 * requested by timeout on the gpio given by gpio_out.
 */
void generate_pulse(int timeout, int *gpio_out)
{
        struct timeval tv;
        int startTime, lastTime, time;

        do_gettimeofday(&tv);
        startTime = tv.tv_usec;
        if (startTime > timeout){                               // At the rollover of the second,
                timeout += 1000000;                             // extend the timeout value
        }

        lastTime = startTime;

        do {                                                    // Spin until time of day >= timeout
                do_gettimeofday(&tv);

                time = tv.tv_usec;
                if (time < lastTime){                           // If the second rolled over,
                        time += 1000000;                                // extend the time value.
                }
                lastTime = time;

        } while (time < timeout);

        pulseTime[0] = tv.tv_sec;                               // Record time of rising edge
        pulseTime[1] = tv.tv_usec;

        if (pulseTime[1] == timeout){                           // Omit pulse if pulseTime[1] > timeout

                gpio_set_value(*gpio_out, 1);

                timeout += 500;  //pulse lenght

                do {                                            // Spin until time of day >= timeout
                        do_gettimeofday(&tv);

                        time = tv.tv_usec;
                        if (time < lastTime){                   // If the second rolled over,
                                time += 1000000;                // extend the time value.
                        }
                        lastTime = time;

                } while (time < timeout);

                gpio_set_value(*gpio_out, 0);
        }
}
```

*Figure 35:**PPS driver generator code sample***

Appendix G Displayes example of two NTP servers and one NTP client through PUTTY interface

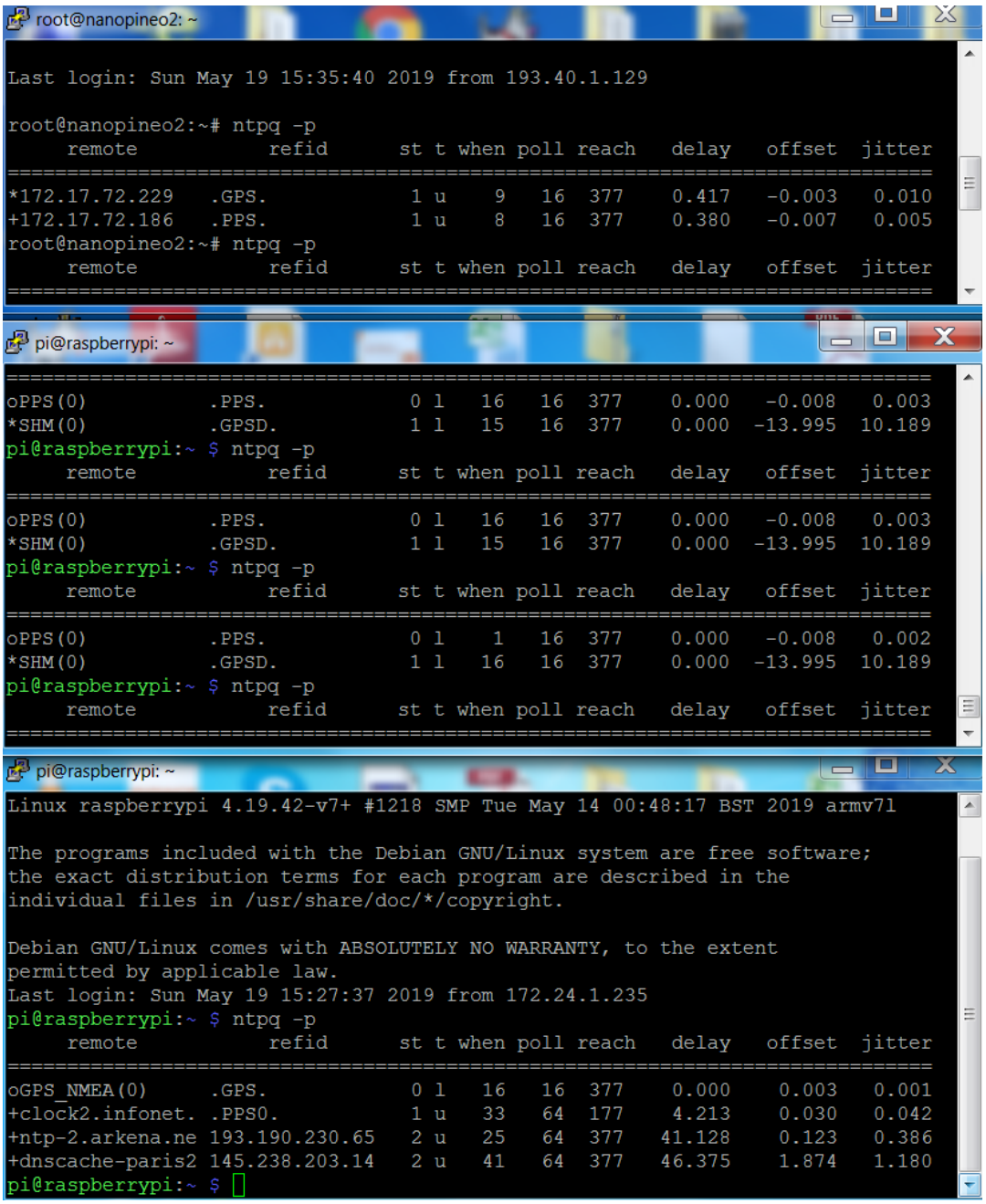


Figure 36: ***On the top NanoPI NEO2 NTP client in the middle NTP TSIP server and on the bottom NTP NMEA server.***

I, Miroslav Rolko,

1. here with grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,
"**Development and performance analysis of the Network Time Protocol Server**" supervised by Viljo Allik and Indrek Sünter.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons license CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive license does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.
Miroslav Rolko
21.05.2019