Time & Frequency

Sagar Pradhan

JB KNOWLEDGE PARK(MDU)

sagar.pradhan

MENTOR-

1. Dr. Poonam Arora

2. Dr. Vijay Narain Ojha

**Design and Development of Network Time Display for Dissemination of Indian Standard Time**

An Embedded System to display Indian Standard Time

Through Network Time Protocol

**CERTIFICATE**

This is to certify that the project entitled “**Design and Development of Network Time Display for Dissemination of Indian Standard Time**” is the authentic record of work carried out by **SAGAR PRADHAN**, student of **B.Tech(Electronics and Communication), JB KNOWLEDGE PARK** at Time and Frequency Division, **CSIR- National Physical Laboratory**, New Delhi under the guidance of Dr. Poonam Arora and Dr. V.N. Ojha, during the period of June 22, 2018 to Sept05, 2018.

**Dr. Poonam Arora Dr. Vijay Narain Ojha**

Sr. Scientist Chief. Scientist

Time & Frequency Metrology Time & Frequency Metrology

CSIR-National Physical Laboratory CSIR-National Physical Laboratory

New Delhi, Delhi-110012 New Delhi, Delhi-110012

**ACKNOWLEDGEMENT**

First of all I would like to thank the Almighty for his constant strength, guidance and his abundant grace which enabled me to finish my project work successfully.

I would like to thank **Dr. Harshit Sinha** (**Dean**, **Maharishi Dayanand University**) for allowing me to do my internship at CSIR-National Physical Laboratory.

I would also like to thank all my faculty members of Delhi Technological University for their guidance and help.

The internship opportunity I had with **CSIR-National Physical Laboratory, NewDelhi** was a great chance for learning and professional development.

I would like to express my gratitude towards my Guide **Dr. Poonam Arora** (**Sr. Scientist**, Time and Frequency Division, CSIR-National Physical Laboratory) and **Dr. V.N Ojha** (**Chief Scientist**, Time and Frequency Division, CSIR-NPL) for giving me the opportunity to carry out this project in their supervision. I would like to thank them genuinely for encouraging for the project, thetasks and challenges that were given to me which helped me to increase significantlyin my knowledge. I sincerely thank him for guiding me and keeping me on thecorrect path to carry out the project.

I am highly thankful to the Ph.D scholar **Mrs. Shalu Goel and Mr. Vattikonda Bharath** (CSIR-National Physical Laboratory) for all their kind help throughout the internship.

My acknowledgement would be incomplete without expressing my sincere thanks to all the scientists and all the staff of CSIR-National Physical Laboratory, who actively helped me in every aspect by providing relevant materials, data andinformation pertaining to the successful completion of my project which wasextremely valuable for my study both theoretically and practically.

I perceive this opportunity as a big milestone in my career development. I will strive to use the gained skills and knowledge in the best possible way and I will continue to work on their improvement in order to attain the desired career objectives.

I am indebted to my loving parents "**Pramod Pradhan**", "**Sumitra Pradhan** immensely for their love and support without whom everything is incomplete and I would alsolike to thanks my brother "**Manoj Pradhan**" for his constant help, emotional support, love and concern.

Finally, I would like to thank all my friends and well-wishers for their support throughout my course and project work.

**SAGAR PRADHAN**

About CSIR-NPL

CSIR-National Physical Laboratory (CSIR-NPL) is the National Measurement Institute (NMI) of India and has the mandate for establishment, maintenance and dissemination of the units of physical measurements based on the International System of units (SI units). One of the seven base SI units, second, is realized (at par to the international level), maintained and disseminated by CSIR-NPL to the whole nation as the *Indian Standard Time*(IST).

CSIR-NPL (internationally known as NPLI) is the *Timekeeper of the Nation and*has the “Primary Time Scale Ensemble”, which gives time that is traceable to the Coordinated Universal Time (UTC) provided by International Bureau of Weights and Measures (BIPM) located in Sevres, France. The IST (i.e. UTC-NPLI plus 5:30 hours) generated with an ensemble of a bank of caesium clocks and hydrogen maser has current uncertainty of  7.2 nano-seconds with respect to UTC. CSIR-NPL disseminates IST through various mechanisms, such as Common View Global Navigation Satellite System (CVGNSS), Network Time Protocol (NTP), and Telephone lines. The present App is a network time dissemination mode and shows the IST maintained by CSIR-NPL through NTP service.

**MOTIVATION**

CSIR- National Physical Laboratory is one of the most prestigious labs in India where I got the opportunity to do my summer internship program. I got to know about the different standards that are maintained here, and was really motivated to work under one of those standards. Under the guidance of Dr. Poonam Arora in Time and Frequency Metrology Division, I was acquainted with a totally new field of research that related to dissemination of time.

In Time and Frequency Division, Indian Standard time is disseminated through three ways which are via internet, landline connection and satellite link. But, time transfer through network time protocol using Raspberry Pi was a real challenge for me. This kept me motivated and also gave me a platform to learn PYTHON coding.

**ABSTRACT**

Among all the physical quantities, time is the one which can be measured to the highest precision and accuracy. The base unit of time in System international (SI) is ‘second’ which is being realized to with an uncertainty of 10-15. The current definition of the SI second, as given by the International Bureau of Weights and Measures (BIPM) is;

“The second is duration of 9,192,631,770 periods of radiation corresponding to the transition between the two hyperfine levels of the ground state of the cesium 133 atoms.”

CSIR- National Physical Laboratory, New Delhi is an authorized signatory of the BIPM in India. As a time keeper of India, it generates, maintains and disseminates the time across the nation. Presently CSIR- NPL has five atomic clock which generates 1 pulse per second i.e. based on above definition of SI second. These atomic clocks are traceable via satellite link to BIPM which generates Universal Coordinated Time (UTC). There are many techniques through which CSIR - NPL disseminates time and frequency to all over India, Network Time Protocol is one of them.

Network Time protocol (NTP) is commonly used to synchronize clocks over network to UTC(NPLI) in wide area network such as public network. The delay asymmetry in wide area network due to routing or band width saturation is usually a dominating source of error. The uncertainty in time transfer using this method is in milliseconds.NTP is an application layer protocol and was designed by David L. Mills of University in 1985.NTP works on User Datagram protocol((UDP), Its default port number is 123 and it uses its own binary format. NTP is used to synchronize the clocks of server and clients to Universal CoordinatedTime (UTC) within few milliseconds error and maintain consistent time where high accuracy is required. Primary sources of time to NTP server can be GPS time or UTC synchronized atomic clocks.

It is the most widely used method for maintaining the accurate time over the world. NTP is one of the oldest and most used protocols.

**CHAPTER 1**

* 1. **RASPBERRY PI 3**

It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

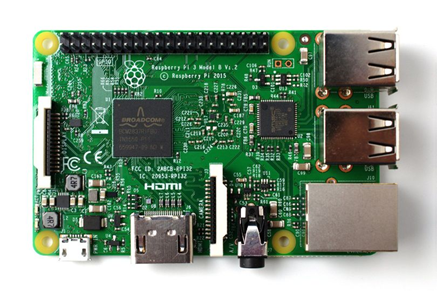
Raspberry Pi is a small device and it cost is low, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse.

The Raspberry Pi 3 Model B is the latest version of the $35 Raspberry Pi computer. The Pi isn't like your typical machine, in its cheapest form it doesn't have a case, and is simply a credit-card sized electronic board -- of the type you might find inside a PC or laptop but much smaller.

### **HOW DO I GET STARTED WITH THE RASPBERRY PI 3?**

One thing to bear in mind is that the Pi by itself is just a bare board. You'll also need a power supply, a monitor or TV, leads to connect to the monitor--typically HDMI, and a mouse and keyboard.

Once you've plugged in all the cables, the easiest way for new users to get up and running on the Pi is to download [The NOOBS (New Out-Of-Box Software) installer](https://www.raspberrypi.org/downloads/noobs/). Once the download is complete, follow some instructions and it will walk you through how to install an OS on the Pi. The installer makes it simple to set up various operating systems, although a good choice for first time users is the official OS Raspbian--although other operating systems are listed below.

The look and feel of Raspbian should be familiar to any desktop computer user. The OS, which is constantly being improved, recently had a graphical overhaul, and includes an optimized web browser, an office suite, programming tools, educational games, and other software.

*Fig. 1: Raspberry Pi 3*

**1.3 Interface Ports of Raspberry pi**

* USB mouse
* microSD card
* microSD USB card reader
* A monitor or TV that supports HDMI or composite video
* An HDMI cable or composite video cable
* An ethernet cable (or Wi-Fi dongle)  Raspberry Pi
* Power supply
* USB keyboard

**1.4 Steps to use raspberry Pi**

* Reformat your microSD card
* Download NOOBS onto the microSD card
* Set up your Raspberry Pi
* Download the Raspbian operating system on the Raspberry Pi
* Configure your Raspberry Pi

**1.5 How to Configure Raspberry Pi?**

* Reformat your microSD card
* Download NOOBS onto the microSD card
* Set up your Raspberry Pi
* Download the Raspbian operating system on the Raspberry Pi
* Configure your Raspberry Pi

**1.6 Re-format MicroSD Card**

* Insert your **microSD card** into the USB card reader.
* Connect the **card reader** to your computer
* **Dock** to install SD Formatter 5.0.
* Click the launched pad icon in your Dock
* Click on the **SD Formatter 5.0** app to open it
* Under **Select Card** select your microSD card from the dropdown menu.
* Click **Format** in the bottom right corner.

**1.7 Download Noobs into the MicroSD Card**

* Download the ZIP file of [NOOBS Version 2.4.5](https://www.raspberrypi.org/downloads/noobs/). It is a large file and will take a while to complete.
* Double-click on the NOOBS file from the **Downloads folder** in your **Dock** to open it.
* Drag and drop all selected NOOBS files into the **SD card icon** on your desktop. You don't have to open the SD card drive.
* Select **"Eject SD card**
* Remove the card reader from your computer.

**1.8 Set to up your Raspberry Pi**

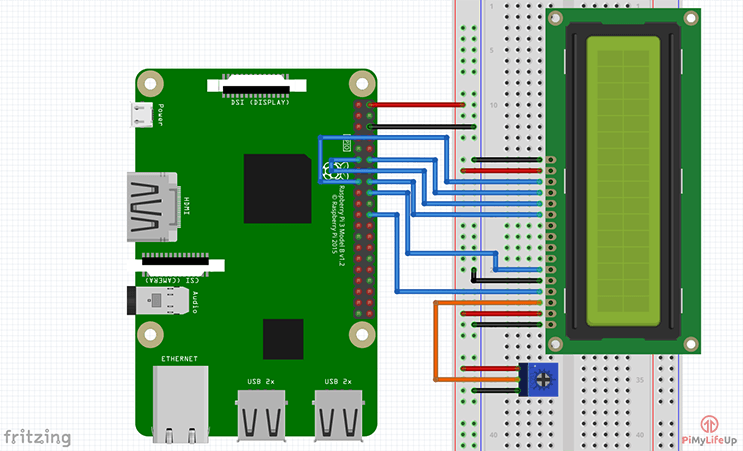
* Insert the microSD card into the card slot on the underside of the Raspberry Pi.
* Plug the USB keyboard and USB mouse into one of the USB ports.
* Turn on your monitor or TV set and Plug the HDMI or video component cable into the monitor or TV set
* Connect an ethernet cable to your router for internet
* Connect the power supply to the Raspberry Pi.
* Plug the power supply into the power outlet**.**

**1.9 Download the Raspbian operating system on the Raspberry Pi**

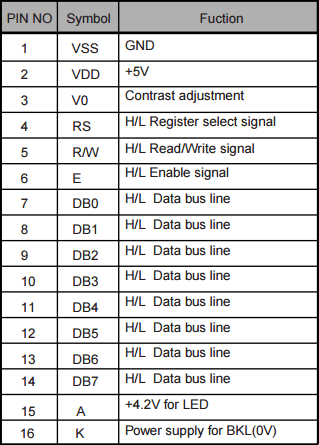
* Select Raspbian.
* Click Install
* When the warning window pops up. Click Yes to confirm. This is just letting you know that the microSD card will be overwritten with an uncompressed version of the Raspbian operating system.
* Wait for the installation process to complete.

*Fig. 2: GPIO Pins of raspberry Pi3*

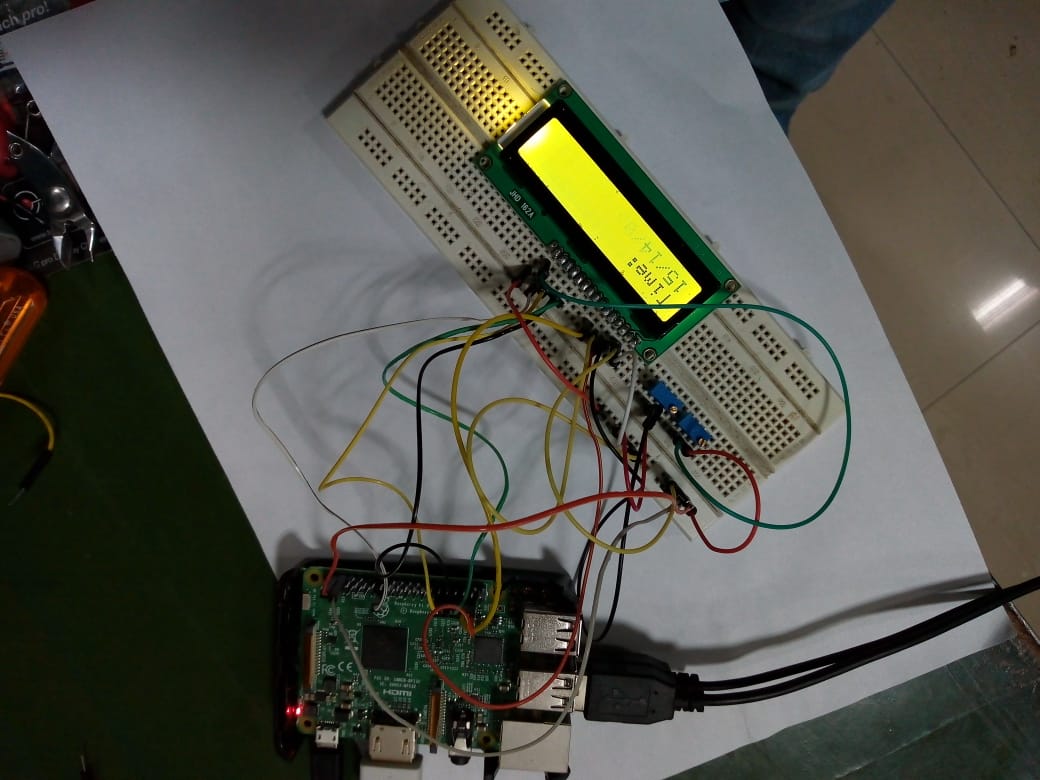
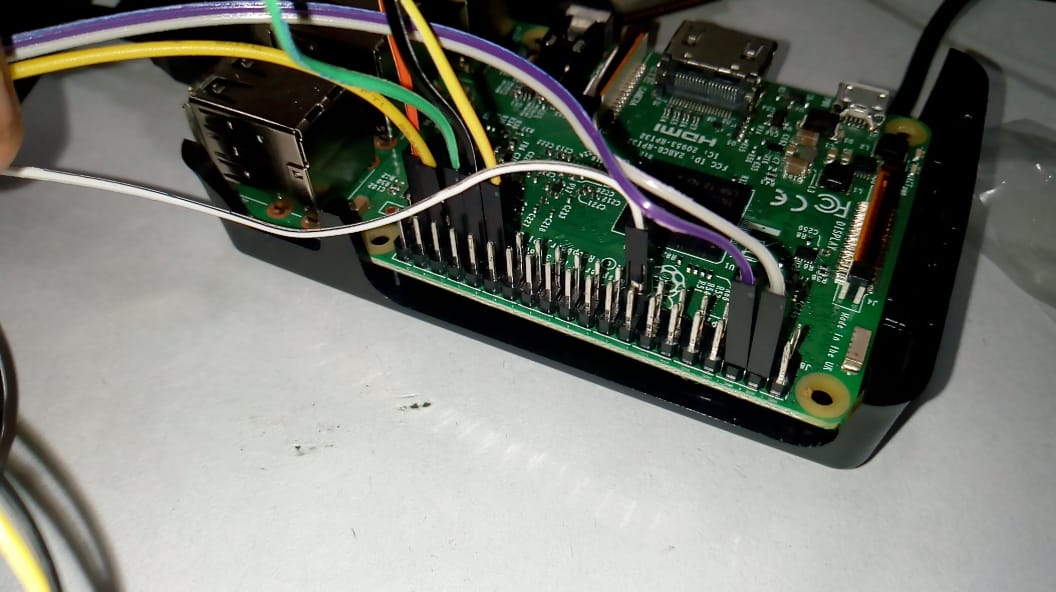
*Fig. 3: JHD 16\*2 M7 LCD*

**

*Fig. 4: Circuit Diagram of Raspberry Pi connected to LCD*



*Fig. 9: LCD Pins functions*

*****Fig. 5: Pins connected with jumper wires*

*Fig. 6: Raspberry Pi 3 connected with LCD to Display time*

**CHAPTER 2**

**2.1**

NTP ( Network Time Protocol)

In 1979, Network [time synchronization](https://en.wikipedia.org/wiki/Time_synchronization) technology was used in what was possibly the first public demonstration of [Internet](https://en.wikipedia.org/wiki/Internet) services running over a trans-Atlantic satellite network, at the [National Computer Conference](https://en.wikipedia.org/wiki/National_Computer_Conference) in New York. The technology was later described in the 1981 Internet Engineering Note (IEN) 173[[6]](https://en.wikipedia.org/wiki/Network_Time_Protocol#cite_note-6) and a public protocol was developed from it that was documented in [RFC](https://en.wikipedia.org/wiki/Request_for_Comments_(identifier)) [778](https://tools.ietf.org/html/rfc778). The technology was first deployed in a local area network as part of the Hello routing protocol and implemented in the [Fuzzball router](https://en.wikipedia.org/wiki/Fuzzball_router), an experimental operating system used in network prototyping, where it ran for many years.

Other related network tools were available both then and now. They include the [Daytime](https://en.wikipedia.org/wiki/Daytime_Protocol) and [Time](https://en.wikipedia.org/wiki/Time_Protocol) protocols for recording the time of events, as well as the [ICMP Timestamp](https://en.wikipedia.org/wiki/ICMP_Timestamp) and IP Timestamp option ([RFC](https://en.wikipedia.org/wiki/Request_for_Comments_(identifier)) [781](https://tools.ietf.org/html/rfc781)). More complete synchronization systems, although lacking NTP's data analysis and clock disciplining algorithms, include the Unix daemon *timed*, which uses an election algorithm to appoint a server for all the clients;[[7]](https://en.wikipedia.org/wiki/Network_Time_Protocol#cite_note-7) and the **Digital Time Synchronization Service** (DTSS), which uses a hierarchy of servers similar to the NTP stratum model.

In 1985, NTP version 0 (NTPv0) was implemented in both Fuzzball and Unix, and the NTP packet header and round-trip delay and offset calculations, which have persisted into NTPv4, were documented in [RFC](https://en.wikipedia.org/wiki/Request_for_Comments_(identifier)) [958](https://tools.ietf.org/html/rfc958). Despite the relatively slow computers and networks available at the time, accuracy of better than 100 [milliseconds](https://en.wikipedia.org/wiki/Millisecond) was usually obtained on Atlantic spanning links, with accuracy of tens of milliseconds on [Ethernet](https://en.wikipedia.org/wiki/Ethernet) networks.

In 1988, a much more complete specification of the NTPv1 protocol, with associated algorithms, was published in [RFC](https://en.wikipedia.org/wiki/Request_for_Comments_(identifier)) [1059](https://tools.ietf.org/html/rfc1059). It drew on the experimental results and clock filter algorithm documented in [RFC](https://en.wikipedia.org/wiki/Request_for_Comments_(identifier)) [956](https://tools.ietf.org/html/rfc956) and was the first version to describe the [client-server](https://en.wikipedia.org/wiki/Client-server_model) and [peer-to-peer](https://en.wikipedia.org/wiki/Peer-to-peer) modes. In 1991, the NTPv1 architecture, protocol and algorithms were brought to the attention of a wider engineering community with the publication of an article by [David L. Mills](https://en.wikipedia.org/wiki/David_L._Mills) in the [IEEE Transactions on Communications](https://en.wikipedia.org/wiki/IEEE_Transactions_on_Communications)

**2.2 CLOCK SYNCHRONISATION**

NTP uses a hierarchical, semi-layered system of time sources. Each level of this hierarchy is termed a *stratum* and is assigned a number starting with zero for the reference clock at the top. A server synchronized to a stratum *n* server runs at stratum *n* + 1. The number represents the distance from the reference clock and is used to prevent cyclical dependencies in the hierarchy. Stratum is not always an indication of quality or reliability; it is common to find stratum 3 time sources that are higher quality than other stratum 2 time sources. A brief description of strata 0, 1, 2 and 3 is provided below.

**Stratum 0**

These are high-precision timekeeping devices such as [atomic clocks](https://en.wikipedia.org/wiki/Atomic_clock), GPS or other [radio clocks](https://en.wikipedia.org/wiki/Radio_clock). They generate a very accurate [pulse per second](https://en.wikipedia.org/wiki/Pulse_per_second) signal that triggers an [interrupt](https://en.wikipedia.org/wiki/Interrupt) and timestamp on a connected computer. Stratum 0 devices are also known as reference clocks.

**Stratum 1**

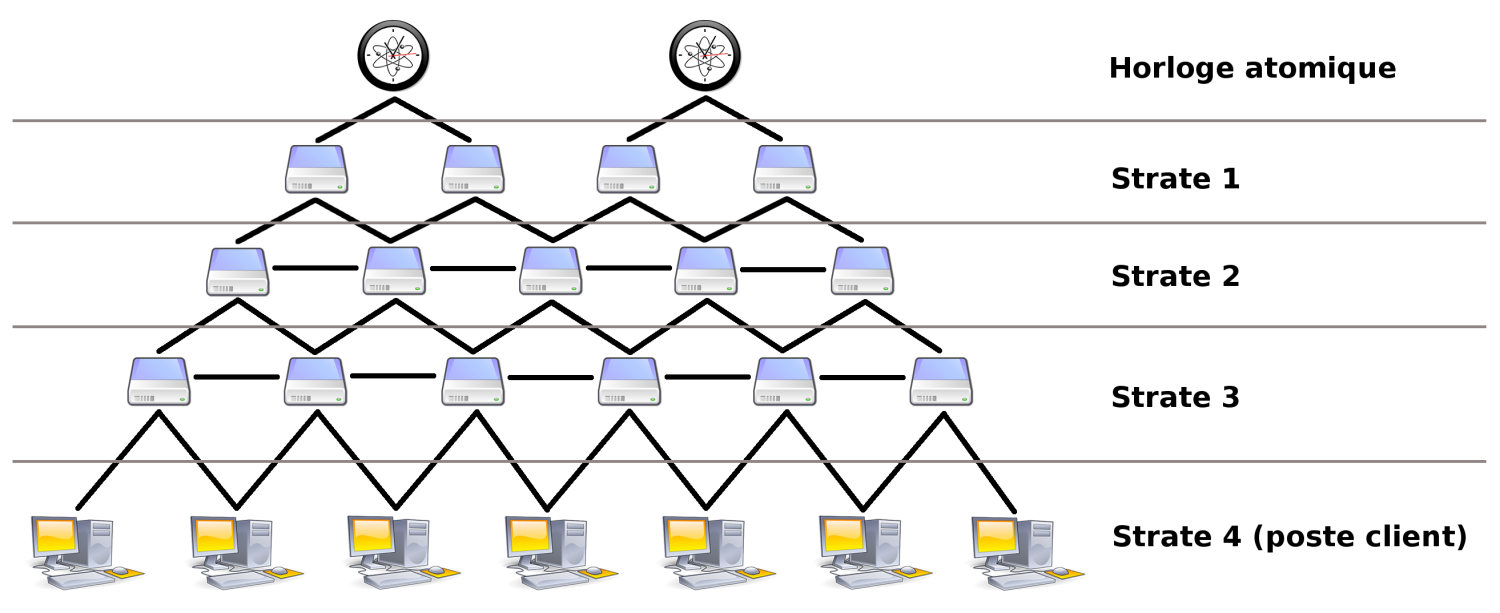
These are computers whose [system time](https://en.wikipedia.org/wiki/System_time) is synchronized to within a few microseconds of their attached stratum 0 devices. Stratum 1 servers may peer with other stratum 1 servers for [sanity check](https://en.wikipedia.org/wiki/Sanity_check) and backup. They are also referred to as primary time servers.

**Stratum 2**

These are computers that are synchronized over a network to stratum 1 servers. Often a stratum 2 computer will query several stratum 1 servers. Stratum 2 computers may also peer with other stratum 2 computers to provide more stable and robust time for all devices in the peer group.

**Stratum 3**

These are computers that are synchronized to stratum 2 servers. They employ the same algorithms for peering and data sampling as stratum 2, and can themselves act as servers for stratum 4 computers, and so on.

****The upper limit for stratum is 15; stratum 16 is used to indicate that a device is unsynchronized. The NTP algorithms on each computer interact to construct a [Bellman-Ford](https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm) shortest-path [spanning tree](https://en.wikipedia.org/wiki/Spanning_tree), to minimize the accumulated round-trip delay to the stratum 1 servers for all the clients.

*Fig. 7: NTP Architecture*

**2.3 Clock Synchronization algorithm**

A typical NTP client will regularly [poll](https://en.wikipedia.org/wiki/Polling_(computer_science)) one or more NTP servers. To synchronize its clock, the client must compute its time offset and [round-trip delay](https://en.wikipedia.org/wiki/Round-trip_delay_time). Time offset *θ* is defined by

{\displaystyle \theta ={(t\_{1}-t\_{0})+(t\_{2}-t\_{3}) \over 2}}

and the round-trip delay *δ* by

*δ*  **= (t3 – t0) + (t2 – t1)**

{\displaystyle \delta ={(t\_{3}-t\_{0})-(t\_{2}-t\_{1})}}

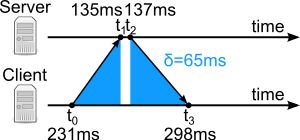
where

*t*0 is the client's timestamp of the request packet transmission,

*t*1 is the server's timestamp of the request packet reception,

*t*2 is the server's timestamp of the response packet transmission and

*t*3 is the client's timestamp of the response packet reception

****

*Fig. 8: Round Trip Delay Time*

Time offset is given by following formula

**Ɵ = (t1 – t2) + (t2 – t3)**

**2**

**CHAPTER 3**

**3.1 Implementation code apply**

* Open LXterminal in raspberry Pi
* Update current version in raspberry pi(using command [sudoraspi-config])

**3.2 Code to connect LCD through raspberry pi**

-(git clone <https://github.com/adafruit/Adafruit_Python_CharLCD.git>)

- (sudo python setup.py install)

**3.3 Connect with NTP Server**

- (sudo apt-get install ntp)

-(sudo /etc/init.d/ntp restart)

**3.3 Python command code for LCD time display**

* Cd /Adafruit\_Python\_CharLCD
* Sudonanochar\_lcd.py(code write here)
* After that it save with ctrl+x and press Y to save it
* Sudo python char\_lcd.py(Run the programme)

**3.4 Autorun folder**

-sudonano .baschr

**RESULT & SCOPE**

IST (Indian Standard Time) show successfully on LCD Screen using Raspberry Pi3 through NTP Server ([www.time.nplindia.org](http://www.time.nplindia.org)).

Scope is that to reduce delay of showing Indian Standard Time because reaching NTP server and getting time from it to display on a receiver produces some delay too.

****

*Fig. 10: Final Complete Showing IST (Indian Standard Time)*

**3.5 Run code**

Then time display automaticaly

###!/usr/bin/python

# Example using a character LCD connected to a Raspberry Pi

from socket import AF\_INET, SOCK\_DGRAM # For setting up the UDP packet.

import thread

import sys

import socket

import binascii

import struct, time # To unpack the packet sent back and to convert the seconds to a string.

import datetime

import pytz

import Adafruit\_CharLCD as LCD

global open\_ntd\_count, var\_log\_file, timestamp

global ntp\_server\_name = "time.nplindia.org"

global sync\_time = 60

global bias = 0

'''NTP Class Declaration'''

class NTPException(Exception):

"""Exception raised by this module."""

pass

class NTP:

"""Helper class defining constants."""

\_SYSTEM\_EPOCH = datetime.date(\*time.gmtime(0)[0:3])

"""system epoch"""

\_NTP\_EPOCH = datetime.date(1900, 1, 1)

"""NTP epoch"""

NTP\_DELTA = (\_SYSTEM\_EPOCH - \_NTP\_EPOCH).days \* 24 \* 3600

"""delta between system and NTP time"""

REF\_ID\_TABLE = {

"GOES": "Geostationary Orbit Environment Satellite",

"GPS\0": "Global Position System",

"GAL\0": "Galileo Positioning System",

"PPS\0": "Generic pulse-per-second",

"IRIG": "Inter-Range Instrumentation Group",

"WWVB": "LF Radio WWVB Ft. Collins, CO 60 kHz",

"DCF\0": "LF Radio DCF77 Mainflingen, DE 77.5 kHz",

"HBG\0": "LF Radio HBG Prangins, HB 75 kHz",

"MSF\0": "LF Radio MSF Anthorn, UK 60 kHz",

"JJY\0": "LF Radio JJY Fukushima, JP 40 kHz, Saga, JP 60 kHz",

"LORC": "MF Radio LORAN C station, 100 kHz",

"TDF\0": "MF Radio Allouis, FR 162 kHz",

"CHU\0": "HF Radio CHU Ottawa, Ontario",

"WWV\0": "HF Radio WWV Ft. Collins, CO",

"WWVH": "HF Radio WWVH Kauai, HI",

"NIST": "NIST telephone modem",

"ACTS": "NIST telephone modem",

"USNO": "USNO telephone modem",

"PTB\0": "European telephone modem",

"LOCL": "uncalibrated local clock",

"CESM": "calibrated Cesium clock",

"RBDM": "calibrated Rubidium clock",

"OMEG": "OMEGA radionavigation system",

"DCN\0": "DCN routing protocol",

"TSP\0": "TSP time protocol",

"DTS\0": "Digital Time Service",

"ATOM": "Atomic clock (calibrated)",

"VLF\0": "VLF radio (OMEGA,, etc.)",

"1PPS": "External 1 PPS input",

"FREE": "(Internal clock)",

"INIT": "(Initialization)",

"\0\0\0\0": "NULL",

}

"""reference identifier table"""

STRATUM\_TABLE = {

0: "unspecified or invalid",

1: "primary reference (%s)",

}

"""stratum table"""

MODE\_TABLE = {

0: "reserved",

1: "symmetric active",

2: "symmetric passive",

3: "client",

4: "server",

5: "broadcast",

6: "reserved for NTP control messages",

7: "reserved for private use",

}

"""mode table"""

LEAP\_TABLE = {

0: "no warning",

1: "last minute of the day has 61 seconds",

2: "last minute of the day has 59 seconds",

3: "unknown (clock unsynchronized)",

}

"""leap indicator table"""

class NTPPacket:

"""NTP packet class.

This represents an NTP packet.

"""

\_PACKET\_FORMAT = "!BBBb 11I"

"""packet format to pack/unpack"""

def \_\_init\_\_(self, version=2, mode=3, tx\_timestamp=0):

"""Constructor.

Parameters:

version -- NTP version

mode -- packet mode (client, server)

tx\_timestamp -- packet transmit timestamp

"""

self.leap = 0

"""leap second indicator"""

self.version = version

"""version"""

self.mode = mode

"""mode"""

self.stratum = 0

"""stratum"""

self.poll = 0

"""poll interval"""

self.precision = 0

"""precision"""

self.root\_delay = 0

"""root delay"""

self.root\_dispersion = 0

"""root dispersion"""

self.ref\_id = 0

"""reference clock identifier"""

self.ref\_timestamp = 0

"""reference timestamp"""

self.orig\_timestamp = 0

"""originate timestamp"""

self.recv\_timestamp = 0

"""receive timestamp"""

self.tx\_timestamp = tx\_timestamp

"""tansmit timestamp"""

defto\_data(self):

"""Convert this NTPPacket to a buffer that can be sent over a socket.

Returns:

buffer representing this packet

Raises:

NTPException -- in case of invalid field

"""

try:

packed = struct.pack(NTPPacket.\_PACKET\_FORMAT,

(self.leap<< 6 | self.version<< 3 | self.mode),

self.stratum,

self.poll,

self.precision,

\_to\_int(self.root\_delay) << 16 | \_to\_frac(self.root\_delay, 16),

\_to\_int(self.root\_dispersion) << 16 |

\_to\_frac(self.root\_dispersion, 16),

self.ref\_id,

\_to\_int(self.ref\_timestamp),

\_to\_frac(self.ref\_timestamp),

\_to\_int(self.orig\_timestamp),

\_to\_frac(self.orig\_timestamp),

\_to\_int(self.recv\_timestamp),

\_to\_frac(self.recv\_timestamp),

\_to\_int(self.tx\_timestamp),

\_to\_frac(self.tx\_timestamp))

except struct.error:

raise NTPException("Invalid NTP packet fields.")

return packed

deffrom\_data(self, data):

"""Populate this instance from a NTP packet payload received from

the network.

Parameters:

data -- buffer payload

Raises:

NTPException -- in case of invalid packet format

"""

try:

unpacked = struct.unpack(NTPPacket.\_PACKET\_FORMAT,

data[0:struct.calcsize(NTPPacket.\_PACKET\_FORMAT)])

except struct.error:

raise NTPException("Invalid NTP packet.")

self.leap = unpacked[0] >> 6 & 0x3

self.version = unpacked[0] >> 3 & 0x7

self.mode = unpacked[0] & 0x7

self.stratum = unpacked[1]

self.poll = unpacked[2]

self.precision = unpacked[3]

self.root\_delay = float(unpacked[4])/2\*\*16

self.root\_dispersion = float(unpacked[5])/2\*\*16

self.ref\_id = unpacked[6]

self.ref\_timestamp = \_to\_time(unpacked[7], unpacked[8])

self.orig\_timestamp = \_to\_time(unpacked[9], unpacked[10])

self.recv\_timestamp = \_to\_time(unpacked[11], unpacked[12])

self.tx\_timestamp = \_to\_time(unpacked[13], unpacked[14])

class NTPStats(NTPPacket):

"""NTP statistics.

Wrapper for NTPPacket, offering additional statistics like offset and

delay, and timestamps converted to system time.

"""

def \_\_init\_\_(self):

"""Constructor."""

NTPPacket.\_\_init\_\_(self)

self.dest\_timestamp = 0

"""destination timestamp"""

@property

def offset(self):

"""offset"""

return ((self.recv\_timestamp - self.orig\_timestamp) +

(self.tx\_timestamp - self.dest\_timestamp))/2

@property

def delay(self):

"""round-trip delay"""

return ((self.dest\_timestamp - self.orig\_timestamp) -

(self.tx\_timestamp - self.recv\_timestamp))

@property

deftx\_time(self):

"""Transmit timestamp in system time."""

return ntp\_to\_system\_time(self.tx\_timestamp)

@property

defrecv\_time(self):

"""Receive timestamp in system time."""

return ntp\_to\_system\_time(self.recv\_timestamp)

@property

deforig\_time(self):

"""Originate timestamp in system time."""

return ntp\_to\_system\_time(self.orig\_timestamp)

@property

defref\_time(self):

"""Reference timestamp in system time."""

return ntp\_to\_system\_time(self.ref\_timestamp)

@property

defdest\_time(self):

"""Destination timestamp in system time."""

return ntp\_to\_system\_time(self.dest\_timestamp)

class NTPClient:

"""NTP client session."""

def \_\_init\_\_(self):

"""Constructor."""

pass

defrequest(self, host, version=2, port='ntp', timeout=5):

"""Query a NTP server.

Parameters:

host -- server name/address

version -- NTP version to use

port -- server port

timeout -- timeout on socket operations

Returns:

NTPStats object

"""

# lookup server address

addrinfo = socket.getaddrinfo(host, port)[0]

family, sockaddr = addrinfo[0], addrinfo[4]

# create the socket

s = socket.socket(family, socket.SOCK\_DGRAM)

try:

s.settimeout(timeout)

# create the request packet - mode 3 is client

query\_packet = NTPPacket(mode=3, version=version,

tx\_timestamp=system\_to\_ntp\_time(time.time()))

# send the request

s.sendto(query\_packet.to\_data(), sockaddr)

# wait for the response - check the source address

src\_addr = None,

while src\_addr[0] != sockaddr[0]:

response\_packet, src\_addr = s.recvfrom(256)

# build the destination timestamp

dest\_timestamp = system\_to\_ntp\_time(time.time())

except socket.timeout:

raise NTPException("No response received from %s." % host)

finally:

s.close()

# construct corresponding statistics

stats = NTPStats()

stats.from\_data(response\_packet)

stats.dest\_timestamp = dest\_timestamp

return stats

def \_to\_int(timestamp):

"""Return the integral part of a timestamp.

Parameters:

timestamp -- NTP timestamp

Retuns:

integral part

"""

return int(timestamp)

def \_to\_frac(timestamp, n=32):

"""Return the fractional part of a timestamp.

Parameters:

timestamp -- NTP timestamp

n -- number of bits of the fractional part

Retuns:

fractional part

"""

return int(abs(timestamp - \_to\_int(timestamp)) \* 2\*\*n)

def \_to\_time(integ, frac, n=32):

"""Return a timestamp from an integral and fractional part.

Parameters:

integ -- integral part

frac -- fractional part

n -- number of bits of the fractional part

Retuns:

timestamp

"""

return integ + float(frac)/2\*\*n

defntp\_to\_system\_time(timestamp):

"""Convert a NTP time to system time.

Parameters:

timestamp -- timestamp in NTP time

Returns:

corresponding system time

"""

return timestamp - NTP.NTP\_DELTA

defsystem\_to\_ntp\_time(timestamp):

"""Convert a system time to a NTP time.

Parameters:

timestamp -- timestamp in system time

Returns:

corresponding NTP time

"""

return timestamp + NTP.NTP\_DELTA

defleap\_to\_text(leap):

"""Convert a leap indicator to text.

Parameters:

leap -- leap indicator value

Returns:

corresponding message

Raises:

NTPException -- in case of invalid leap indicator

"""

if leap in NTP.LEAP\_TABLE:

return NTP.LEAP\_TABLE[leap]

else:

raise NTPException("Invalid leap indicator.")

defmode\_to\_text(mode):

"""Convert a NTP mode value to text.

Parameters:

mode -- NTP mode

Returns:

corresponding message

Raises:

NTPException -- in case of invalid mode

"""

if mode in NTP.MODE\_TABLE:

return NTP.MODE\_TABLE[mode]

else:

raise NTPException("Invalid mode.")

defstratum\_to\_text(stratum):

"""Convert a stratum value to text.

Parameters:

stratum -- NTP stratum

Returns:

corresponding message

Raises:

NTPException -- in case of invalid stratum

"""

if stratum in NTP.STRATUM\_TABLE:

return NTP.STRATUM\_TABLE[stratum] % (stratum)

elif 1 < stratum < 16:

return "secondary reference (%s)" % (stratum)

elif stratum == 16:

return "unsynchronized (%s)" % (stratum)

else:

raise NTPException("Invalid stratum or reserved.")

defref\_id\_to\_text(ref\_id, stratum=2):

"""Convert a reference clock identifier to text according to its stratum.

Parameters:

ref\_id -- reference clock indentifier

stratum -- NTP stratum

Returns:

corresponding message

Raises:

NTPException -- in case of invalid stratum

"""

fields = (ref\_id>> 24 & 0xff, ref\_id>> 16 & 0xff,

ref\_id>> 8 & 0xff, ref\_id& 0xff)

# return the result as a string or dot-formatted IP address

if 0 <= stratum <= 1:

text = '%c%c%c%c' % fields

if text in NTP.REF\_ID\_TABLE:

return NTP.REF\_ID\_TABLE[text]

else:

return "Unidentified reference source '%s'" % (text)

elif 2 <= stratum < 255:

return '%d.%d.%d.%d' % fields

else:

raise NTPException("Invalid stratum.")

defget\_ntp\_time(host):

port = 123; # Port.

read\_buffer = 1024; # The size of the buffer to read in the received UDP packet.

address = ( host, port ); # Tuple needed by sendto.

data = '\x1b' + 47 \* '\0'; # Hex message to send to the server.

epoch = 2208988800L; # Time in seconds since Jan, 1970 for UNIX epoch.

client = socket.socket( AF\_INET, SOCK\_DGRAM ); # Internet, UDP

client.sendto( data, address ); # Aend the UDP packet to the server on the port.

data, address = client.recvfrom( read\_buffer ); # Get the response and put it in data and put the send socket address into address.

t = struct.unpack( "!12I", data )[ 10 ]; # Unpack the binary data and get the seconds out.

return t - epoch; # Calculate seconds since the epoch.

defsync\_ntd():

global timestamp

# print "Please wait while getting time from " + server + "\r\n"

response = call.request(server, version=3)

timestamp = round(response.tx\_time + bias)

**My code**# print "NPLI NTP TIME: " + time.ctime(timestamp) + "\r\n"

ntp\_date = datetime.datetime.fromtimestamp(timestamp)

lcd.message(ntp\_date.strftime("%a %d/%m/%Y\nIST %H:%M:%S"))

# Raspberry Pi pin setup

lcd\_rs = 26

lcd\_en = 19

lcd\_d4 = 13

lcd\_d5 = 06

lcd\_d6 = 05

lcd\_d7 = 22

lcd\_backlight = 4

# Define LCD column and row size for 16x2 LCD.

lcd\_columns =16

lcd\_rows = 2

lcd = LCD.Adafruit\_CharLCD(lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7, lcd\_columns, lcd\_rows, lcd\_backlight)

lcd.message('Hello! dear\n how are you?')

# Wait 5 seconds

time.sleep(2.0)

lcd.clear()

#START NEW CODE

call = NTPClient()

while True:

var\_log\_file = open(log\_file, "a")

sync\_ntd()

time.sleep(sync\_time)

lcd.clear()

#End NEW CODE

**7. python char\_lcd.py**