Index

Introduction

Machine-to-machine (M2M) communication is automated data transmission technology. It provides data communication between the machines via wired or wireless channels. The data communication from one host to another is based on the Open Systems Interconnection Model (OSI). However, M2M is a subset of the Internet of Things, and compared to the Internet of Things (IoT) technology, machines to machines (M2M) have some limitations, but cost, performance, and reliability is better.

This report is mainly focused on two sections: the design of the M2M communication architecture and understanding the network configuration and related commands. Secondly, we designed a real-life M2M communication system with the help of the Raspberry Pi. The primary model of this system is the opening window in a wider and more complex network. During the implementation process, we discussed step by step how to reach the final goal. Finally, it will provide a clear understanding and methodology of which parameters and factors need to be considered to improve the system's efficiency.

My personal objective during the experiment was that it was one of the best opportunities to understand and gain hands-on experience with real-life network configuration for multi-homing and route setup in multiple local area networks (LANs). In addition to that, a clear approach to how the latency is affected in a closed-loop control system and how to overcome it with the continuous coefficient-varying method.

M2M system in the context of OSI model

Here we are going to discuss the how a machine-to-machine communication work and their transmission from one layer to another. M2M communication data transfer technology is reference of OSI model.

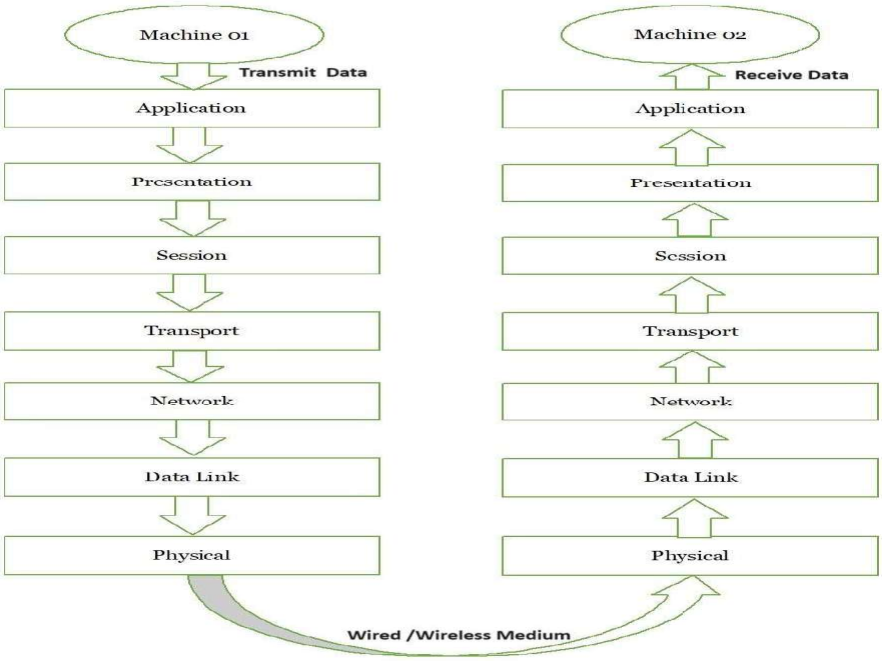

Figure .1 OSI model of M2M Communication.

How it Works

As per the above reference model, when data is transferred from source to destination, it should be going through seven layers, and we can go through a top-down approach. Firstly, a machine sends or receives the data from the application layer, which holds several applications programs like mail services, directory services, and network resources. The network-supporting application is carried out in this layer, and after the data is transferred to the presentation layer, it provides data compression, data encryption, and decryption and sends it to the session layers. The session layer performs the synchronisation of data communication between the two different hosts. However, also ensure the data resynchronization is done properly in opposite directions. The data from the session layer is transferred to the transport layer, which runs on a certain protocol. The transport layer converts the whole data into multiple data points (segmentation) and passes them to the network layer. The network layer directs the data through different channels based on their running protocol. The data link layer is divided into two layers: media access control (MAC) and logical access control (LLC). It performs flow control, synchronisation, and error checking. The last one is the physical layer, which means a medium of communication wired or wireless from one to another. Similarly, data is received from the physical layer of the destination and passed step by step to the application layer end user.

1.Target Architecture network

Design Part

It is one of the most complex network configurations when compared to others, and in this experiment, we're using four Raspberry Pi A, B, C, and D with their individual IP addresses, as indicated in the diagram below. Manager Pi 3 is the name assigned to the first Raspberry Pi A. It acts as the bridge between the server and host devices (MyRigB, MyRigC, and MyRigD), and Manager Pi can interact with the rest of MyRig B, C, and D via the SSH command (Linux preferred OS).

To begin, four Raspberry Pis (A, B, C, and D) are connected to the Network Switch_01 via an Ethernet (RJ45) connection. Furthermore, switch_01 and switch_02 are interconnected to build a firewall between us and others, thus it. Additionally, the Manager Pi is linked to the router, and the router is linked to the server (unix.ncl.ac.uk).

List of Equipment used for Network Design

- Raspberry Pi 3B or 4B: 4 sets
- Micro SD card 32G Samsung Class U1: 4
- Network switch Net gear GS305 with a power supply unit: 1 set
- RJ45 network patch cables, Cat 6: 4
- Router

Design of Target Architecture

.

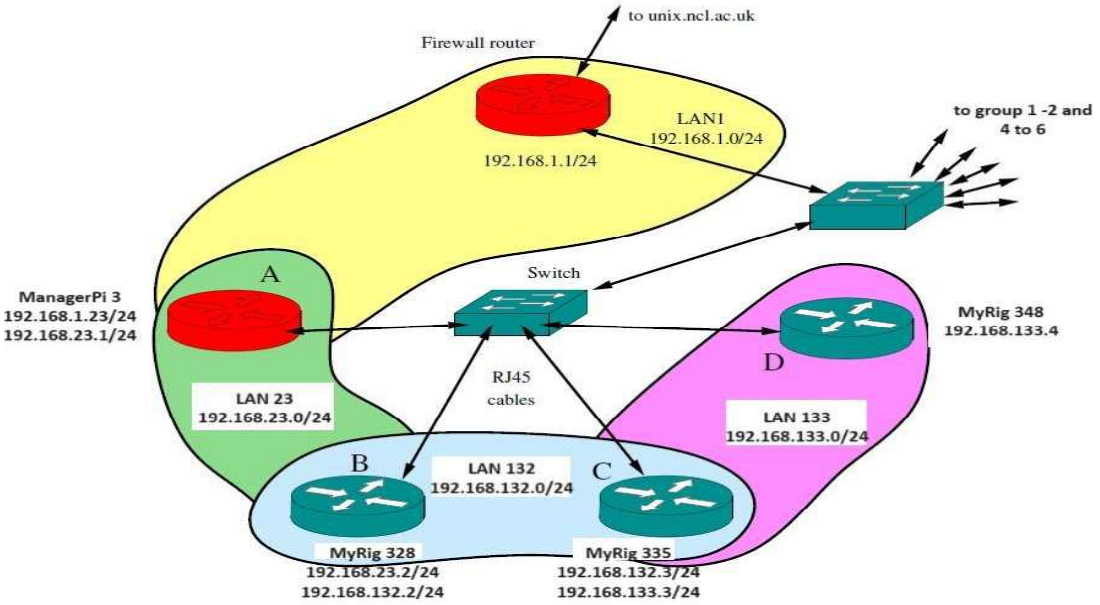

Figure .2 Target architecture

Experiments: A

1. Capture / Analysis of datagram packet

   For datagram packet analysis, we use the “tcpdump” command with the subset command as well as the Transmission Control Protocol (TCP).

Step 1: First, we will check if “tcpdump” is installed in our operating system (OS) by using the command “tcpdump --version" and the output as shown below.

pi@MyRig328:~$ tcpdump --version
tcpdump version 4.9.3
libpcap version 1.8.1
OpenSSL 1.1.1d 10 Sep 2019

Step:2- Secondly, we will check the command like "sudo tcpdump -c 5” for the real purpose of analysing and specifically capture only five packets due to the higher number of packets transferring. See the figure: 2.

```
pi@MyRig328:~$ sudo tcpdump -c 5
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
17:07:55.975532 IP 192.168.23.2.ssh > 192.168.23.1.47504: Flags [P.], seq 1798702323:1798702511, ack 1524907765, win 501, options [nop,nop,TS val 2383371604 ecr 1991523147], length 188
17:07:55.975741 IP 192.168.23.1.47504 > 192.168.23.2.ssh: Flags [.], ack 188, win 5921, options [nop,nop,TS val 1991523963 ecr 2383371604], length 0
17:07:55.979162 IP 192.168.23.2.ssh > 192.168.23.1.47504: Flags [P.], seq 188:408, ack 1, win 501, options [nop,nop,TS val 2383371608 ecr 1991523963], length 220
17:07:55.979269 IP 192.168.23.2.ssh > 192.168.23.1.47504: Flags [P.], seq 408:596, ack 1, win 501, options [nop,nop,TS val 2383371608 ecr 1991523963], length 188
17:07:55.979337 IP 192.168.23.1.47504 > 192.168.23.2.ssh: Flags [.], ack 408, win 5921, options [nop,nop,TS val 1991523967 ecr 2383371608], length 0
5 packets captured
11 packets received by filter
0 packets dropped by kernel
```

Figure :2- Specifically captured 5 packets/datagram (sudo tcpdump -c 5)

Step:3- For further study and analysis we took one datagram from above as shown below.

The captured packet of tcpdump information is formed in each new line, and each line contains a timestamp and packet information. The typical format of the TCP protocol is shown in figure 3 and the captured packet is shown in figure 4.

[Timestamp] [Protocol] [Src IP].[Src Port] > [Dst IP].[Dst Port]: [Flags], [Seq], [Ack], [Win Size], [Options], [Data Length]

Figure .3: Typical format of TCP Protocol.

17:07:55.979162 IP 192.168.23.2.ssh > 192.168.23.1.47504: Flags [P.], seq 188:408, ack 1, win 501, options [nop,nop,TS val 2383371608 ecr 1991523963], length 220

Figure .4 Capture packet.

- ✓ **17:07:55.979162 –** The local time of captured packet and the following format Hour: Minutes:Seconds.Frac.
- ✓ **192.168.23.2.ssh** – The IP address of source and last port is separated with dot(.) here ssh
- ✓ **192.168.23.1.47504** - The IP address of destination and last port is separated with dot(.) here 47504.
- ✓ **Flags [P.]** – Push Acknowledgment packet, it means the acknowledge the previous packet and send data.
- ✓ **seq 188:408** – The Sequence number means that in a data stream 188 to 408 data bytes contain.
- ✓ **ack 1** – it means only one packet of datagram is expected after receiving this packet.
- ✓ **win 501** – it is a window number of available bytes.
- ✓ **options [nop,nop,TS val 2383371608 ecr 1991523963]** – its mean TCP options (nop: no operation, TS: Timestamp of TCP, ecr: echo reply).
- ✓ **length 220** – it means the length of payload data is 220.

Experiment: B

How Does the Ping Command Work

The ping command uses the Internet Control Message Protocol (ICMP) to echo requests from the destination host while waiting for a response. The Ping command provides two important pieces of information: one is how many echo request messages are returned, and the second is how much time it takes to return them.

1. Find out the latency using the "ping" command from MyRig328 to 192.168.132.3(MyRig335) see below figure.5

```
pi@MyRig328:~$ ping 192.168.132.3
PING 192.168.132.3 (192.168.132.3) 56(84) bytes of data.
64 bytes from 192.168.132.3: icmp_seq=1 ttl=64 time=0.284 ms
64 bytes from 192.168.132.3: icmp_seq=2 ttl=64 time=0.216 ms
64 bytes from 192.168.132.3: icmp_seq=3 ttl=64 time=0.208 ms
64 bytes from 192.168.132.3: icmp_seq=4 ttl=64 time=0.203 ms
64 bytes from 192.168.132.3: icmp_seq=5 ttl=64 time=0.219 ms
^C
--- 192.168.132.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 198ms
rtt min/avg/max/mdev = 0.203/0.226/0.284/0.029 ms
```

Figure:5- Ping 192.168.132.3 (from MyRig328).

As per the data analysis, the above ping command (ping 192.168.132.3) requests an echo message from

destination host 192.168.132.3. In this data, five packets of data were transmitted and five of them received with zero percent loss, and finally, the average time taken to receive was 0.226 milliseconds.

Note: For the above data, the average latency is 0.226 milliseconds, and the throughput is 4.424 bytes/seconds. (Total number of data points / Total time taken) Latency and throughput are the two factors that determine network performance.

2.Similarly the Ping command from MyRig328 to Manager Pi(192.168.23.2) see the figure.6

```
pi@MyRig328:~$ ping 192.168.23.2
PING 192.168.23.2 (192.168.23.2) 56(84) bytes of data.
64 bytes from 192.168.23.2: icmp_seq=1 ttl=64 time=0.096 ms
64 bytes from 192.168.23.2: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 192.168.23.2: icmp_seq=3 ttl=64 time=0.068 ms
64 bytes from 192.168.23.2: icmp_seq=4 ttl=64 time=0.068 ms
64 bytes from 192.168.23.2: icmp_seq=5 ttl=64 time=0.066 ms
^C
--- 192.168.23.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 191ms
rtt min/avg/max/mdev = 0.066/0.073/0.096/0.014 ms
```

Figure.6- Ping 192.168.23.2 (from MyRig328).

Note: Average latency is 0.73 milliseconds, and throughput is 13.586 bytes/second.

3.Route determination using the command “traceroute”

Traceroute is a network tool to determine the packet route from source to destination host, and it also provides time for how long it took to reach the destination host. Traceroute using Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values. The TTL value is decreased with each hop, and finally it reaches zero, which means an error message is returned carrying the address of the last router. For example, see the below figure. 6 routes from Manager Pi3 to Facebook

```
c3054367@ManagerPi3:~ $ traceroute facebook.com
traceroute to facebook.com (157.240.221.35), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  0.326 ms  0.279 ms  0.244 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  ct-pop--ncl-site1.ncl.ac.uk (194.81.6.225)  289.049 ms  1.305 ms  1.210 ms
 7  ae30.glasss-sbr1.ja.net (146.97.37.213)  4.751 ms  4.505 ms  4.315 ms
 8  ae31.manckh-sbr2.ja.net (146.97.33.53)  8.789 ms  9.657 ms  9.578 ms
 9  ae29.erdiss-sbr2.ja.net (146.97.33.41)  11.135 ms  90.986 ms  86.581 ms
10  ae31.londpg-sbr2.ja.net (146.97.33.21)  90.136 ms  87.905 ms *
11  ae29.londhx-sbr1.ja.net (146.97.33.1)  15.301 ms * *
12  * ae0.londhx-ban2.ja.net (146.97.35.198)  25.998 ms  25.916 ms
13  ae0.pr02.lhr4.tfbnw.net (157.240.70.134)  49.941 ms  49.864 ms  49.788 ms
14  po152.asw02.lhr6.tfbnw.net (129.134.44.202)  14.982 ms po162.asw01.lhr3.tfbn
w.net (129.134.44.206)  14.656 ms po152.asw02.lhr6.tfbnw.net (129.134.44.202)  1
5.052 ms
15  psw04.lhr8.tfbnw.net (129.134.57.126)  15.445 ms psw01.lhr8.tfbnw.net (129.1
34.57.129)  16.190 ms  16.113 ms
16  173.252.67.195 (173.252.67.195)  15.486 ms 157.240.38.181 (157.240.38.181)
14.667 ms 157.240.38.185 (157.240.38.185)  14.372 ms
17  edge-star-mini-shv-01-lhr8.facebook.com (157.240.221.35)  15.510 ms * *
```

Figure.7. traceroute Facebook.com (from Manager Pi3)

Design of M2M communication framework

1. Network Design

Here we used the same target architecture network for experimenting with the machine-to (M2M) communication framework. see the figure. 7 Design of Target Architecture Network below. The objective of this framework design is mainly focused on two of them. The first is to configure each network device with their unique internet protocol (IP) address and examine the connectivity, datagram/packet analysis from different LAN networks, calculation of latency and throughputs, and finally SSH key generation and data pipeline creation between the devices. Secondly, we discussed which network devices are suitable for connecting things (IoT devices) as per the first objective experiments. The implementation and component's parts are the same as the Target architecture, so I didn't discuss them here.

M2M Communication Framework Network Modeling

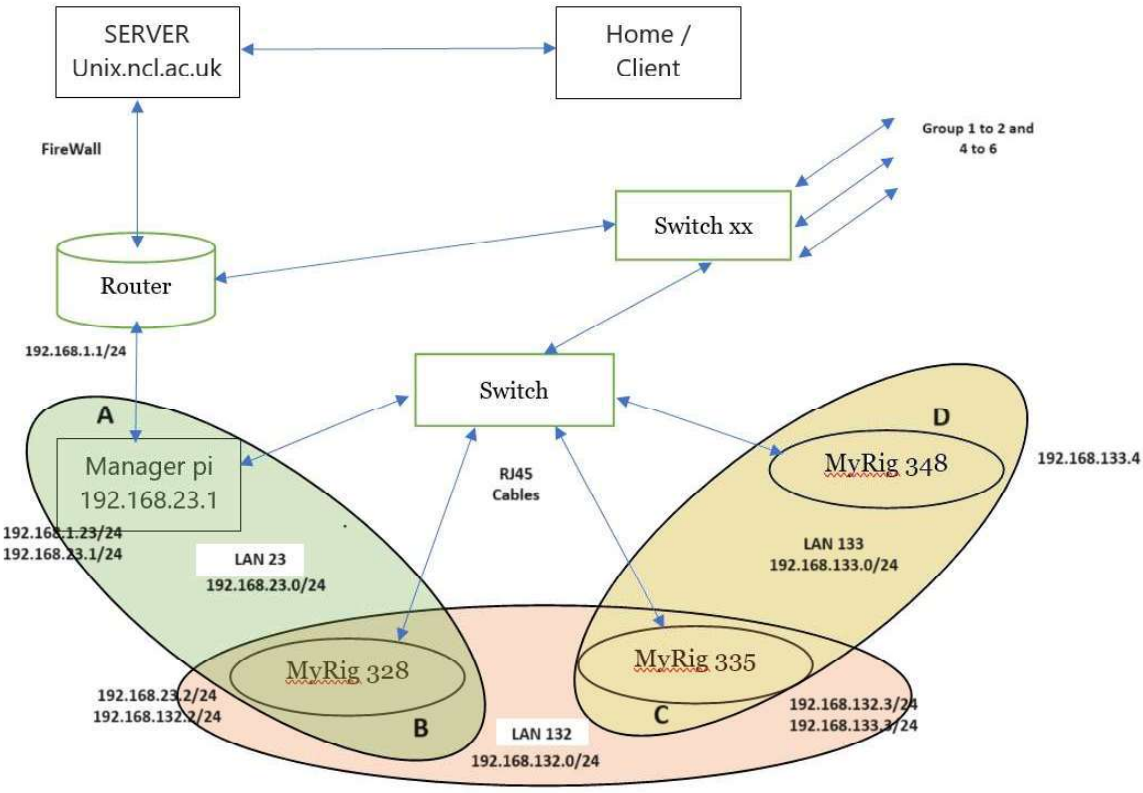

Figure.8- Design of Target Architecture Network

Experiment:01-Find the IP addresses, define the LANs, and determine which gateways to use.

Steps for Procedure

Step 01: Open the ssh terminal and login into the university server (unix.ncl.ac.uk) with username and provided password.

Step 02: After login to the University server map to port 10322 (see the below command) because our machine (ManagerPi) is connected to the Port 10322 terminal. After entering the command provide the assigned password to access the ManagerPi terminal.

Command : ssh -Y -p 10322 username@localhost

Step 03: Enter the below command and see the discussion with figure.9, 10 and 11.

Command: ifconfig, arp and route.

For this experiment, I understood how to find the IP address, LANs, and gateways on each host of the network, and before we go further, we need to clear the networking commands.

1. ifconfig (interface configuration): This command is used to find the configuration information of the connected networks, and it's also used to define the network address of the connected host or devices.
2. arp (Address Resolution Protocol): It is a communication protocol to find the MAC addresses of currently connected devices in a network. It also acts like an echo request to find the IP address of connected devices in a network.
3. route: This command is used to find the routing table of the current device and get information about which gateway and netmask are used.

See below figure.9, 10 and 11 to analyze the IP, MAC address and Route table.

```
c3054367@ManagerPi3:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.23.1  netmask 255.255.255.0  broadcast 192.168.23.255
        ether dc:a6:32:fe:1c:f4  txqueuelen 1000  (Ethernet)
        RX packets 1794292170  bytes 4052899821 (3.7 GiB)
        RX errors 21  dropped 21  overruns 0  frame 32
        TX packets 2769947283  bytes 2704522625 (2.5 GiB)
        TX errors 3  dropped 0 overruns 0  carrier 0  collisions 0

eth0:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.23  netmask 255.255.255.0  broadcast 192.168.1.255
        ether dc:a6:32:fe:1c:f4  txqueuelen 1000  (Ethernet)

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2078514551  bytes 44408772157930 (40.3 TiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2078514551  bytes 44408772157930 (40.3 TiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure.9-IP address of connected network devices

```
c3054367@ManagerPi3:~ $ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.1.1              ether   7c:05:07:0d:65:0f   C                     eth0
192.168.1.21             ether   dc:a6:32:fe:1c:35   C                     eth0
192.168.23.5                     (incomplete)                              eth0
192.168.23.4             ether   dc:a6:32:fe:1c:dc   C                     eth0
192.168.23.3             ether   dc:a6:32:fe:1c:7d   C                     eth0
192.168.23.2             ether   dc:a6:32:fe:1c:94   C                     eth0
192.168.123.4            ether   dc:a6:32:fe:1c:dc   C                     eth0
```

Figure.10 MAC address of the connected device

```
c3054367@ManagerPi3:~ $ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
192.168.0.0     192.168.23.2    255.255.0.0     UG    0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.23.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

Figure.11-MangerPi route.

Discussion: Here we find the IP, MAC, and gateway of the connected device, like ManagerPi3. Here, ManagerPi has two IP addresses: 192.168.23.1 and 192.168.1.23. It is a concept of multi-homing that means a single device has more than one IP address. Secondly, MAC addresses (HWaddress), as shown in Figure 9,

Finally, gateway 192.168.1.
Why Multi-homing is needed?

As per the M2M communication framework architecture (Figure 7), the MangerPi is connected to two LAN networks, which is why it has two IP addresses. The IP addresses of the devices must be different in every network; otherwise, the delivery of data would be a mess.

Experiment 02- Configure the devices, test them, debug any problems.

Here we are using the multi-homing method to configure the device or host, which is to create more than one network connection and IP address on a single host. The major part of a configuration setting depends on three parts: the network IP address, subnet mask, and default gateway. After configuring the host, we understood the separation of networks and subnetworks. Moreover, the method of multi-homing provides better data communication without compromising performance. In addition to that, the subnet mask divides the IP address into two parts. The first one identifies the device, and the other part identifies the network to which it belongs.

Step for the experiments

Objective: Here we are going to configure devices B, C, and D and split the LAN network into three parts and IP as well. Device Manager Pi and B are configured in the first LAN network. Similarly, B and C devices are configured for the second LAN network, and finally, C and D are configured for the third LAN network.

Note: Here ManagerPi is act as a host A (it is the gateway between server and other devices)
Host B: MasterRig 32x (four OS images such as rootfs5, rootfs6, rootfs7 and rootfs8)
Host C: MasterRig 33x (four OS images such as rootfs5, rootfs6, rootfs7 and rootfs8)
Host D: MasterRig 34x (four OS images such as rootfs5, rootfs6, rootfs7 and rootfs8)

Step 01: Login to host A (ManagerPi) using ssh command from unix.ncl.ac.uk server and map to host B (MasterRig32x) using this command c3054367@ManagerPi3:~ $ ssh -Y -p 22 pi@192.168.23.2 : initialize the below command as well.

Command:01 – ls (ls - list directory contents)

Command:02 – cd rootfs8 (as per the directory in MasterRig32x have four OS images like rootfs5,6,7 and 8) here we are configuring in rootfs8.

Command:03- cd etc/network/ interfaces (/etc/network/interfaces contains network interface configuration information for the ifup(8) and ifdown(8) commands. This is where you configure how your system is connected to the network.)

Command:03 Add Custom gateway – send a range of IP addresses to a particular router on your LAN route add -net XXX.XXX.X.X netmask XXX.XXX.XXX.X gwXXX.XXX.X.X

Note: open the file and write the interfaces such as IP address, netmask and gateway (Figure. 11)

Command:03: Finally, boot the file using touch boot_8 and sudo reboot. After finishing the booting process, it is automatically directed to the managerPi (Host A). Once we check the Ping command for testing the

connectivity, Continue the same procedure using the ssh command to login to the Host B device. Now the Host B (MasterRig32x) device is changed to MyRig328 mode. (see figure.10)

Step:02- Similarly continue above command in host devices C and D. (See the figure 12 and 13)

```
pi@MasterRig32x:~ $ ls
clean_conf.tar.gz  crig5678.tar.gz  interfaces.save  new  rootfs5  rootfs6  rootfs7  rootfs8  tmp
pi@MasterRig32x:~ $ cd rootfs8
pi@MasterRig32x:~/rootfs8 $ cd etc/network/
pi@MasterRig32x:~/rootfs8/etc/network $ cat interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto eth0
allow-hotplug eth0
iface eth0 inet static
address 192.168.23.2
netmask 255.255.255.0
gateway 192.168.23.1

auto eth0:0
allow-hotplug eth0:0
iface eth0:0 inet static
address 192.168.132.2
netmask 255.255.255.0

post-up ip route add 192.168.133.0/24 via 192.168.132.3
```

Figure.12- Configuration in Host B (MasterRig 32x)

```
pi@MasterRig33x:~ $ ls
01  clean_conf.tar.gz  crig5678.tar.gz  rig  rootfs5  rootfs6  rootfs7  rootfs8
pi@MasterRig33x:~ $ cd rootfs5
pi@MasterRig33x:~/rootfs5 $ cd etc/network/
pi@MasterRig33x:~/rootfs5/etc/network $ cat interfaces
auto eth0
allow-hotplug eth0
iface eth0 inet static
address 192.168.132.3
netmask 255.255.255.0
gateway 192.168.132.2

auto eth0:0
allow-hotplug eth0:0
iface eth0:0 inet static
address 192.168.133.3
netmask 255.255.255.0

post-up ip route add 192.168.133.0/24 via 192.168.132.3
post-up ip route add 192.168.23.0/24 via 192.168.132.2
pi@MasterRig33x:~/rootfs5/etc/network $ touch boot_5
pi@MasterRig33x:~/rootfs5/etc/network $ sudo reboot
```

Figure.13: Configuration in Host C (MasterRig 33x)

```
pi@MasterRig34x:~ $ ls
clean_conf.tar.gz  crig5678.tar.gz  rootfs5  rootfs6  rootfs7  rootfs8
pi@MasterRig34x:~ $ cd rootfs8
pi@MasterRig34x:~/rootfs8 $ cd etc/network/
pi@MasterRig34x:~/rootfs8/etc/network $ cat interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto eth0
allow-hotplug eth0
 iface eth0 inet static
 address 192.168.133.4
 netmask 255.255.255.0
 gateway 192.168.133.3
```

Figure.14-Configuration in Host D (MasterRig 34x)

Discussion

Note: See below table for conclusion of the which IP address is connected to respective LAN network and gateways. See the table .01

Devices	Network Name	IP Address	Netmask	Gateway
MyRig A	LAN 23	192.168.23.1	255.255.255.0	192.168.23.1
MyRig B		192.168.23.2	255.255.255.0	
MyRig B	LAN 132	192.168.132.2	255.255.255.0	192.168.132.2
MyRig C		192.168.132.3	255.255.255.0	
MyRig C	LAN 133	192.168.133.3	255.255.255.0	192.168.133.3
MyRig D		192.168.133.4	255.255.255.0	

Table.01 -IP address of LAN network

Design of DC motor control System

Here is discussed the implementation of a control system in a DC motor, which consists of two parts: a software part to analyse the speed, frequency, time period, and direction (clockwise or anti-clockwise). The second part is hardware, such as a DC motor, driver board, power supply, and encoder (sensor). The input part of the DC motor is connected to the driver board (control board) to control the speed of the motor rotation, and this driver board is controlled by one of the RigPi via an RS232 serial cable. The output (shaft) of the DC motor is connected to the other motor shaft for breaking purposes. In between the DC motors, a 500-slot disc is attached to measure the respective parameters via an encoder (sensor-IR LED and photo diode). The output of the encoder is connected to the Rigi Pi. Finally, the 20-volt power supply unit is connected to the driver board. See the below figure. 15.

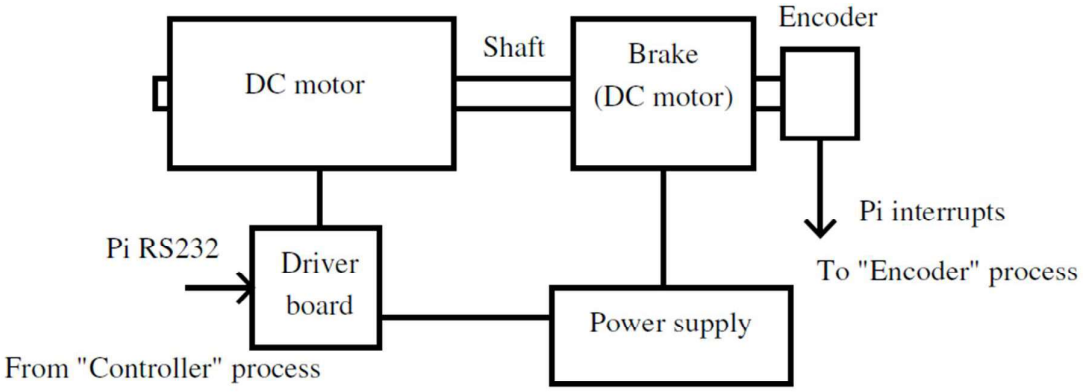

Figure. 15 Design of DC motor control System

Working Principal

Once the motor is started, the rotation speed is measured by the encoder and passed to the Rigi Pi for analysis with the help of the encoder program. The parameters of the encoder output signal are also analysed in the controller program. The desired or rectified data passes to the driver board via serial cable, and the driver board controls the required speed of the motor.

DC Motor Encoder

The 500-cut slot dick is attached to the motor shaft, and in between the light-emitting diode (LED) and photo diode (2-nos), it is attached. Once the disc is completed, a single rotation in one second produces 500 pluses on the encoder side. That means the frequency of disc rotation is 500 Hz. The direction is analysed for the signal detection of the photo diodes. See Figure 16.

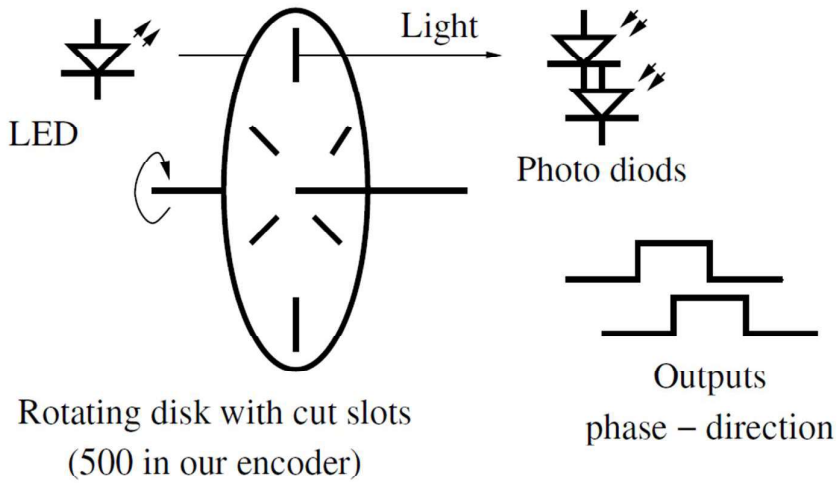

Figure.16- DC Motor Encoder

Design of Control System Architecture

The control system architecture mainly consists of three parts: software, hardware, and network. Latency is the major challenge in between these parts. The software part can be divided into two sections: one is the encoder, and the other is the controller. The hardware part is the motor shaft, and its frequency is measured in milliseconds. Finally, the network part is managed in Rigi Pi, and moreover, the communication part takes part in our configured path.

Latency It is an important term in our control system because, in between software and hardware, similarly, software and network have a small fraction of a second delay due to different physical conditions of the system and environmental factors like temperature. These types of delays or latency can affect the stability of the control system output. To minimise the latency and improve the stability of the output, we add a closed-loop system. The duties of the closed loop function are carried out by the controller programme, which always compares input and output with mathematical functions like proportional-integral-derivative (PID). The output of the PID controller is more stabilised as compared to the previous output. The continuous stabilisation of PID control provides the accurate output of the motor. See below the figure. 17

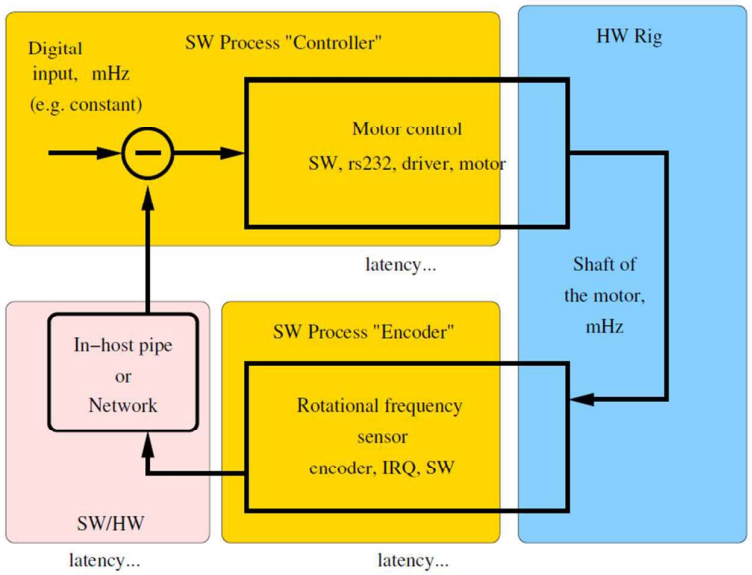

Figure.17-Control System Architecture

Implementation of controller system

Here we discussed the implementation of a closed-loop control system for an electrical motor connected to a single device with the same network, and similarly, the motor c reconnects to two different devices with different networks. The purpose of implementation in two different systems is only to analyse the parameters of the stability controller. Finally, we understood the working method and characteristics of the coefficient in a closed-loop system. See the figure. 15 and 16 models of closed-loop controller systems in experiments 01 and 02.

1. Model of Closed loop Control System Architecture (Experiment.01)

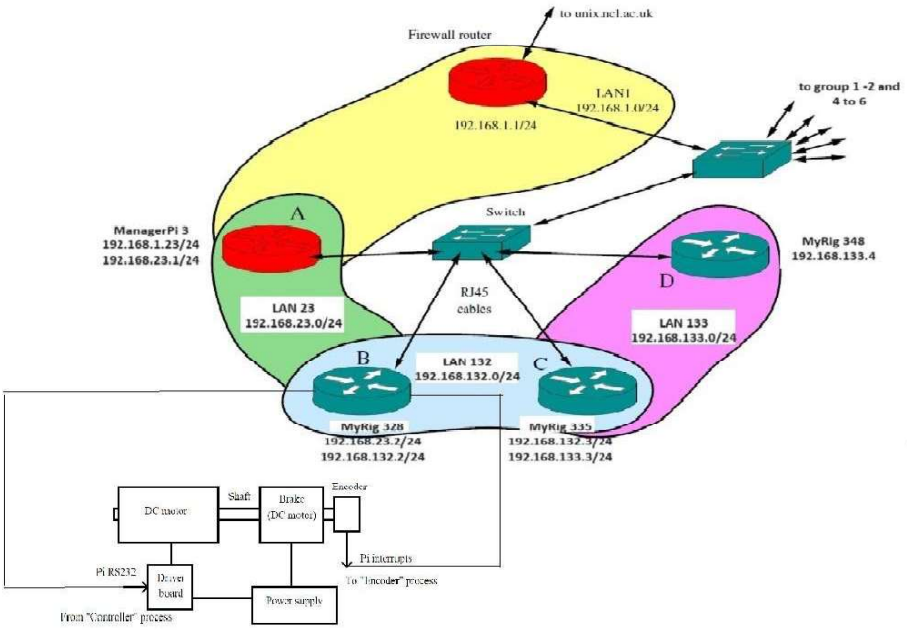

Figure 18-Closed loop system connection in Single device and network.

The closed-loop system connected to the MyRig328 via LAN network 132 (192.168.132.0/24). The host device MyRig328 is running two programs: one is an encoder program to measure the frequency, time period, and rotation, and the other is a controller program to maintain the stability of the motor function based on the Proportional -Integral-Derivative (PID).

2. Model of Closed loop Control System Architecture (Experiment.02)

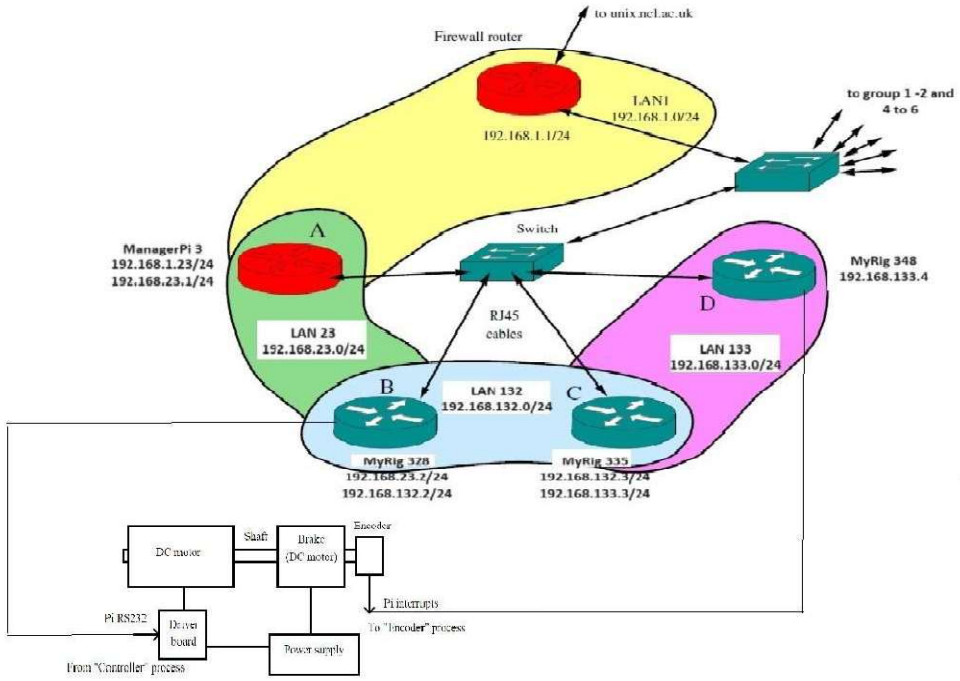

Figure.19- Closed loop system connection in different device and network

The closed-loop system was connected between MyRig328 and MyRig348 with different LAN networks (132 and 133). Now the encoder program is running in MyRig348, and this data is passed through the gateway of MyRig335 to MyRig328. After getting the data from MyRig348, it is going to process the controller program run in MyRig328. Finally, the stabilised output is going to the motor driver part to control the motor functions.

Interrupt for encoder

```
pi@MyRig328:~$ sudo su
root@MyRig328:/home/pi# echo 9 > /sys/class/gpio/unexport
root@MyRig328:/home/pi# echo 9 > /sys/class/gpio/export
root@MyRig328:/home/pi# echo in >
bash: syntax error near unexpected token `newline'
root@MyRig328:/home/pi# echo in > /sys/class/gpio/gpio9/direction
root@MyRig328:/home/pi# echo rising > /sys/class/gpio/gpio9/edge
root@MyRig328:/home/pi#
```

Figure.20- Configure IRQ9

```
pi@MyRig328:~$ sudo su
root@MyRig328:/home/pi# echo 10 > /sys/class/gpio/unexport
root@MyRig328:/home/pi# echo 10 > /sys/class/gpio/export
root@MyRig328:/home/pi# echo in > /sys/class/gpio/gpio10/direction
root@MyRig328:/home/pi# echo none > /sys/class/gpio/gpio10/edge
```

Figure.21- Configure IRQ10

Experiment 01-Implement the closed loop system in one device.

Pies 32 33 34

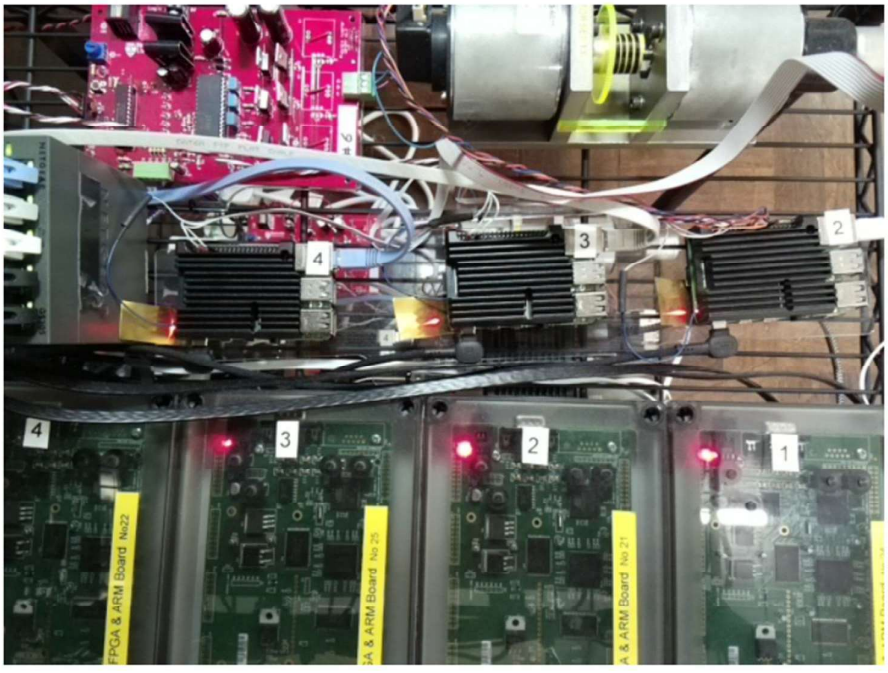

Figure.22- Connection diagram MyRig and Motor

The motor speed is controlled by sending the command from MyRig via serial cable RS232. The encoder part of the DC motor is connected to the general-purpose input and output pins (GPIO) of the MyRig 328. As per the above images (Figures 20 and 21), they show the configuration of GPIO pins 9 and 10.

The command of drive the motor (See figure.23)

```
root@MyRig328:/home/pi# echo 800 1\\ > /dev/serial0
root@MyRig328:/home/pi# echo 737 1\\ > /dev/serial0
root@MyRig328:/home/pi# echo 600 1\\ > /dev/serial0
```

Figure.23- Motor Drive command

- The value of 737 is stop the motor.
- The value of 800 rotates the motor in clockwise direction.
- The value of 600 rotates the motor in an anti-clockwise direction.
- The value is settled + or – 500 at threshold point 737.
- The value greater than 737 it rotates clockwise.
- The Value less than 737 it rotates anti-clockwise.

The output value is fetched in the respective files of GPIO pins (Figures 17 and 18), and after this data, we generate the graph using GnuPlot software for detailed analysis of PID coefficients.

Step 01- Measuring step response.

- In a controller program by replacing the constant input 2hz
- Fetching the output data from the controller file and plot the graph.
- If not stable change the value of PID coefficients.
- The value of PID coefficients is directly proportional to gain so that means control system unstable.
- The lower coefficient values provide better stability of the system.

Step 02-Frequency Measurement

The frequency measurement is taken from IRQ9 for the principle of fetching the rising signal count in a desired period (T sample). Once the rising ng signal is produced in the rise file, the timer will start for the desired period. Finally, we calculate the number of rising signals with their respective periods. See below figure 22 for frequency measurement and figures.24-programme code for frequency measurement.

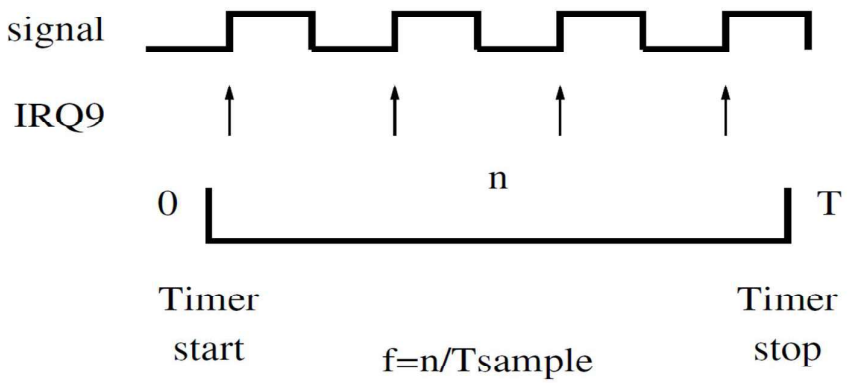

figure 24 for frequency measurement

```c
#include <stdio.h> // printf() fflush()
#include <unistd.h> // usleep() lseek() read()
#include <stdlib.h> // system()
#include <poll.h> // poll()
#include <fcntl.h> // open()
#include <time.h> // if you need a timer in case of interval measurement
#include <pthread.h> // pthread_mutex_lock() pthread_mutex_unl
#define GPIO_PIN_9 "/sys/class/gpio/gpio9/value"

volatile unsigned int freq_counter = 0;

int main() {
    int fd9;

    // Export GPIO pin 9
    system("echo 9 > /sys/class/gpio/unexport; \
            echo 9 > /sys/class/gpio/export; \
            echo in > /sys/class/gpio/gpio9/direction; \
            echo falling > /sys/class/gpio/gpio9/edge");

    // Open the file for GPIO pin 9
    if ((fd9 = open(GPIO_PIN_9, O_RDONLY)) == -1) {
        perror("Failed to open GPIO 9");
        return 1;
    }

    struct pollfd fds[1];
    fds[0].fd = fd9;
    fds[0].events = POLLPRI;

    printf("Monitoring GPIO 9 for falling edges. Press Ctrl+C to exit.\n");

    while (1) {
        int pollResult = poll(fds, 1, -1);  // Wait indefinitely for GPIO 9 events

        if (pollResult > 0) {
            if (fds[0].revents & POLLPRI) {
                // GPIO 9 falling edge event occurred
                lseek(fd9, 0, SEEK_SET);  // Rewind to the beginning of the file
                char str9;
                read(fd9, &str9, 1);
                if (str9 == '0') {
                    freq_counter++;
                }
```

Figure 25- program code for frequency measurement.

Step 03- Direction Measurement

The direction measurement is part of the encoder side, and it's processed from the signals generated from Photo Diodes 1 and 2. It is a simple calculation for checking the signal generated from both photo diodes at the same time. For example, the rising signal from photo diode 1 and the signal from photo diode 2 are both zero, which means the rotation of the disc is clockwise and vice versa anti-clockwise. See the below figure. 25.

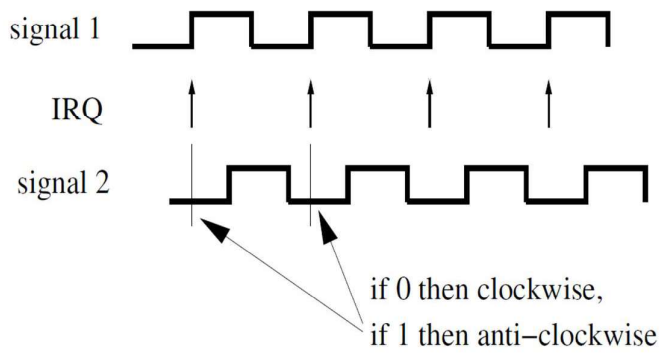

Figure.26-Direction Measurement

Step:04-Encoder program

Before going to the encoder program we need to discuss some functions.

1. Poll () - wait for some event on a file descriptor.
   The function poll is always waiting for some event in a file discreptor. Once the data arrived in a file it will fetch them excute in appropriate area and provide time buffer in that file discriptor.
2. Pthread_create() In the calling process, the pthread_create() technique starts a new thread. The new thread begins execution by performing start_routine (), which is passed to the sole arguments.

See below Encoder Program code.(Figure.27)

```
  GNU nano 3.2
#include <stdio.h> // printf() fflush()
#include <unistd.h> // usleep() lseek() read()
#include <stdlib.h> // system()
#include <poll.h> // poll()
#include <fcntl.h> // open()
#include <time.h> // if you need a timer in case of interval measurement
#include <pthread.h> // pthread_mutex_lock() pthread_mutex_unl

int i;
int fd1; // file descriptor
char str[4];

int main()
{
   // printf() displays the string inside quotation
   printf("Hello, World!\n");

   //call Setup_GPIO;
   system("echo 9 > /sys/class/gpio/unexport; \
           echo 9 > /sys/class/gpio/export; \
           echo in > /sys/class/gpio/gpio9/direction; \
           echo rising > /sys/class/gpio/gpio9/edge; \
           echo 10 > /sys/class/gpio/unexport; \
           echo 10 > /sys/class/gpio/export; \
           echo in > /sys/class/gpio/gpio10/direction; \
           echo none > /sys/class/gpio/gpio10/edge");

   i = 1000;

   while (i > 1)
   {
      //continue execution while i is i as loop
      fd1 = open("/sys/class/gpio/gpio9/value", O_RDONLY | O_NONBLOCK);
      lseek(fd1, 0, SEEK_SET);// rewind to the beginning of the file
      read (fd1, str, 1); // read the value
      printf("value of fdi = %d\n", fd1);
      sleep(1000)
      i--;
      printf("value of i = %d\n", i);
   };

   return 0;
}
```

figure.27-Encoder program code).

Stability Experiment between the components

As per the above experiment, we understand the system is more stable in a control system connected to a single host and a single network because the data communication speed is higher. Secondly, the same control system is connected to different devices and networks, so obviously, the data communication path is higher as a result of the poor stability factor. For the data transfer between the components via wired or wireless, we expect a delay (latency), so it directly affects the desired output. For this instance, we can use the proportional-integral and differential (PID) control system in the feedback loop. This latency factor is reduced with the help of the PID coefficients Kp, Ki, and Kd. If the PID coefficients have a higher value in a system, the gain is also higher, as the system is unstable. As a continuous process, the PID feedback system will overcome the latency, and the output value almost always tends to reach the desired value. The mathematical form of PID is shown below figure.24

The overall control function $u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(\tau)\,\mathrm{d}\tau + K_{\mathrm{d}} \frac{\mathrm{d}e(t)}{\mathrm{d}t},$

Figure .28- Mathematical Form of P-I-D (Source: Wikipedia)

u(t) = PID control Variable
Kp = proportional gain
e(t) = error value
Ki = integral gain
de(t) =change in error value
dt = change in time

working Principal

The common factor of the PID is the error value (e(t)), which is always computed simultaneously with the respective coefficients in proportional, integral, and derivative. The sum of these PID results is equal to the control variable, so we can say the system is stable. The PID coefficient is not a constant variable; it will vary depending on physical and other factors. In our practical experiment, we always plot the graph, analyse the pattern, and change the value of the PID coefficients according to the desired plot.

Conclusion

The step-by-step procedure of the above experiment I understood the M2M communication framework design, architecture, and data communication technology based on the OSI reference model. Secondly, detailed implementation of Target, Instrumental, and home architecture designs and their network configuration methods, including multi-homing, Moreover, this topic provides a wider experience of networking and routing through the Secure Shell (SSH) command. The third part is an understanding of control systems in open and closed loops. In addition to that, the experiment with a closed-loop control system provided more knowledge of deeper concept coefficient parameters. Finally, I understood the factors that affected the stability of the control system and gained good knowledge about key tools for improving the control system's stability.

Reference

1. Figure 2 to 28- Lecture Note , Dr. Alex Bystrov,

2. Latency and Throuhput, comparitech, available website https://www.comparitech.com/net-admin/latency-vs-throughput/, Accessed 17 October 2023.
3. IPv4 packet structure , Toutorial point, Available Website :https://www.tutorialspoint.com/ipv4/ipv4_packet_structure.htm, Accessed 15 October 2023
4. TcpCommand in Linux , https://linuxize.com/post/tcpdump-command-in-linux/, Accessed 10 october 2023.
5. Tracer route work example, Priyanka kumari, https://www.linkedin.com/pulse/traceroute-work-examples-using-command-priyanka-kumari/, Accessed 13 October 2023.
6. poll()--Wait for Events on Multiple Descriptors, IBM , https://www.ibm.com/docs/ja/i/7.3?topic=ssw_ibm_i_73/apis/poll.html , Accessed 16 October 2023.
7. Pthread Create function , Linux , https://linux.die.net/man/3/pthread_create, Accessed 18 October 2023.