



## Controlling Raspberry Pi 3 Model B Using PING Commands

### A. Objectives

1. An introduction to Shell and shell scripting
2. Starting a program at the Auto-start
3. Knowing your distro version
4. Understanding tcpdump command
5. Introducing tshark utility
6. Interfacing RPI to an LCD
7. Understanding PING command

### B. Time of Completion

This laboratory activity is designed for students with some knowledge of Raspberry Pi and it is estimated to take about 5-6 hours to complete.

### C. Requirements

1. A Raspberry Pi 3 Model 3
2. 32 GByte MicroSD card → Give your MicroSD card to the lab instructor for a copy of Ubuntu.
3. USB adaptor to power up the Pi
4. Read Lab 2 – Interfacing with Pi carefully.

### D. Pre-Lab

Lear about ping and ICMP protocols.

## E. Lab

This lab has two separate parts. Please make sure you read each part carefully. Answer all the questions. Submit your codes via Canvas.

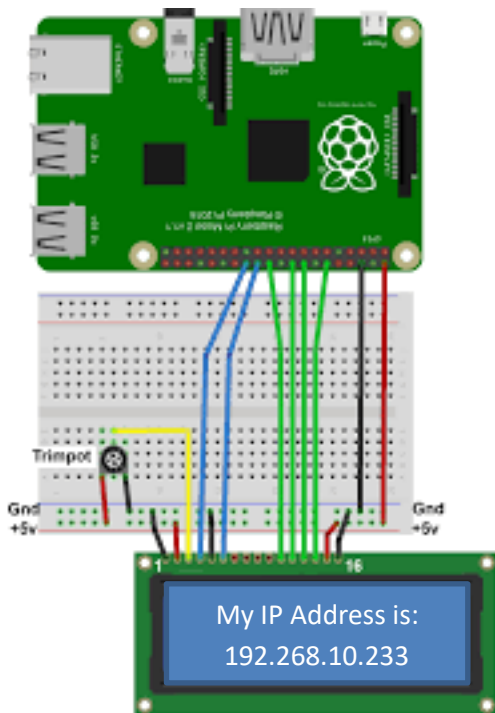
### 1) Part I - Showing IP Addresses on the LCD

In this section we learn how to interface an LCD to the Pi and run a program automatically at the boot up.

#### a) Interfacing your RPI to an LCD

In this section you need to interface your 16×2 LCD with Raspberry Pi using 4-bit mode. Please note that you can choose any type of LCD and interface it to your Pi, including OLED.

Below is the wiring example showing how to interface a 16×2 LCD to RPI. The sample code to program the RPI is provided here: <https://www.electronicshub.org/interfacing-16x2-lcd-with-raspberry-pi/>.



#### b) Auto-Launch

In this section we learn how to auto-start an executable using Auto Lunch in Ubuntu. We assume the Raspberry Pi is using **Ubuntu Mate** (Note that these steps are only tested on Ubuntu). In order to do this we need to create two files: (1) the actual executable program; (2) the start\_up program which runs the executable program. We start with creating the auto-startup program.

First let's make sure we are using Ubuntu: (note that you can do the following using ssh)

```
cat /etc/issue
```

A more detailed information about the OS can be obtained by: `cat /etc/os-release`

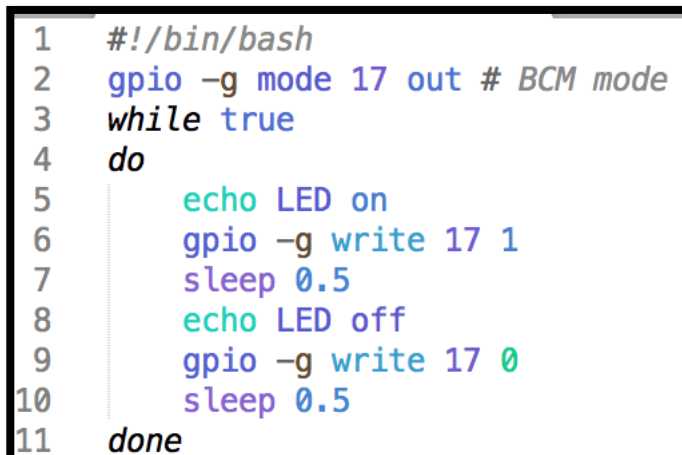
Go to `.config/autostart/` (**don't forget "."**) directory and create a file called `auto_start_led.desktop`:

```
cd .config/autostart/. # don't forget the "dot" in front of the .config
nano auto_start_led.desktop # you can use any other editor
```

Change the content of auto\_start\_led to the following

```
[Desktop Entry]
Type=Application
Name=auto_start_led
Exec=/home/ssuee/led.sh
Comment="Optional"
X-MATE-Autostart-enabled=true
```

Now we need to go to `/home/ssuee/led.sh` and edit that to flash an LED on the PI. In this case the LED is assumed to be connected to GPIO.0. Change the content of `/home/ssuee/led.sh` file as shown below:



```
1  #!/bin/bash
2  gpio -g mode 17 out # BCM mode
3  while true
4  do
5      echo LED on
6      gpio -g write 17 1
7      sleep 0.5
8      echo LED off
9      gpio -g write 17 0
10     sleep 0.5
11 done
```

Make sure the files are properly edited:

```
cd
more /home/ssuee/led.sh
more .config/autostart/auto_start_led.desktop
```

Connect GPIO.0 to an LED and a resistor, as you did in the previous lab:

[http://web.sonoma.edu/users/f/farahman/sonoma/courses/es465/lab/basiclab/RPi\\_Released/Sonoma\\_Lab\\_Raspberrypi3\\_Lab2\\_v2.pdf](http://web.sonoma.edu/users/f/farahman/sonoma/courses/es465/lab/basiclab/RPi_Released/Sonoma_Lab_Raspberrypi3_Lab2_v2.pdf)

Reboot your Raspberry Pi: `sudo reboot`

Alternatively, you can reset your PI. After the rebooting process is completed the LED should blink.

**Congratulations!** You just auto started a new program from your PI.

When a program in the autostart directory starts it continues running in the background. In order to stop it we can first find out its PID: `ps aux | grep led`

You can kill the process using `kill` command. After killing the process, reboot your PI and the LED should blink again.

## c) Part I Project

**1.a.1)** Modify the code example such that when the RPI is powered up, its IP address is displayed on the LCD. Submit your code. **You must show this to the instructor to receive credit for this part.** Note that in order for your program to automatically run you need to refer to Lab 2 – Interfacing with Pi and learn how to perform **Auto Lunch**.

## 2) Part II – Using PING Command to Control the RPI

The purpose of this section is to learn how to use the PING utility to control a RPI. Let's first make sure you understand the basics.

### a) Understanding TCPDUMP

TCPDUMP utility is the premier network analysis tool. It is very similar to Wireshark but it is considered to be a command-line packet analyzer tool. Using tcpdump command we can also capture the live TCP/IP packets and save them in file. The saved captured packets can be further analyzed via advanced tcpdump commands. This utility becomes very handy when it comes to troubleshooting on network level. Before you continue make sure it is installed on your RPI. In order to know if tcpdump has already been installed on your machine, make sure you can run the following command:

```
sudo tcpdump --ver
```

If nothing returned, install tcpdump. First, do `sudo apt-get update` to make sure you are working with the latest files for installing the latest version of tcpdump. Once the updating is completed, you can start the process of installing TCPDump:

```
sudo apt-get install tcpdump
```

To test if tcpdump is working, do `sudo tcpdump` and you should start seeing packets on the screen shortly. If you want to capture just a few packets, you can use something like this: `sudo tcpdump -c 10`. TCPDump will stop capturing after getting approximately 10 packets. You can capture just ICMP packets by using `sudo tcpdump icmp -c 10`.

You can direct the packets you are seeing on the screen by using the `-w` flag and send the packets to a file with a specific name: `sudo tcpdump -w test.pcap -c 10`. This will write the data in pcap format and captures the first 10 packets so that it is readable in Wireshark.

you can verify that the file was in fact created using the following command: `ls *.pcap`. To look at the content of the pcap file, you will need to use Wireshark. An alternative way is use tshark. We learn about it in the next section.

There are many good tutorials on how to use tcpdump commands. Refer to this tutorial and **do all the examples**: <https://hackertarget.com/tcpdump-examples/>. Make sure you understand how the examples work. Note that we need to use sudo to run tcpdump commands.

Submit your answers to the following questions:

**2.a.1)** What is the tcpdump command to filter only packets with source port address of 1025?

**2.a.2)** What is the tcpdump command to filter only ping packets initiated from a machine with IP address

**2.a.3)** What is the tcpdump command to filter packets originated from a machine with IP address 10.0.0.2 directed to a machine with IP address 10.0.0.3?

**2.a.4)** What is the tcpdump command to list interfaces that tcpdump can listen on?

**2.a.5)** What does the following command do? `tcpdump -nni <your_interface> icmp`. Take a snapshot of the response **and explain it**.

## b) Introducing to TSHARK

*TShark* is a terminal oriented version of Wireshark designed for capturing and displaying packets when an interactive user interface isn't necessary or available. Without any options set, TShark will work much like tcpdump. It will also use the pcap library to capture traffic from the first available network interface (or specified interface) and displays a summary line on the standard output for each received packet. The choice of using tshark or tcpdump is very subjective and many people have different opinion. In general, however, tshark supports more protocols. For now we stick with using tcpdump. You can learn tshark on your own.

Tshark supports the same options as Wireshark. So, if you know Wireshark it is very easy to understand how tshark works. For more information on tshark consult your local manual page (`man tshark`) on your PI or [the online resources](#). In this tutorial we do not use tshark. However, a useful tool in tshark is that it can convert pcap files from *binary* format into text file so we can actually view the file using a text editor. The following command converts a pcap file into a text file using tshark command:

```
tshark -r capture_file.pcap -V -x > Readable_File.txt
```

In the above command we have

- `-x` add output of hex and ASCII dump (Packet Bytes)
- `-r <infile>` set the filename to read from (- to read from stdin)
- `-V` add output of packet tree (Packet Details)

You can learn more about tshark command options by doing `tshark -help`. Below is an example of a readable txt file obtained from a pcap file:

```

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Encapsulation type: Ethernet (1)
Arrival Time: Mar 12, 2013 06:37:44.156914000 PDT
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1363095464.156914000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 98 bytes (784 bits)
Capture Length: 98 bytes (784 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:icmp:data]
Ethernet II, Src: Dell_b3:55:05 (00:1a:a0:b3:55:05), Dst: HewlettP_5e:71:4e (1c:c1:de:5e:71:4e)
Destination: HewlettP_5e:71:4e (1c:c1:de:5e:71:4e)
Address: HewlettP_5e:71:4e (1c:c1:de:5e:71:4e)
.... ..0. .... .. = LG bit: Globally unique address (factory default)
.... ...0 .... .. = IG bit: Individual address (unicast)
Source: Dell_b3:55:05 (00:1a:a0:b3:55:05)
Address: Dell_b3:55:05 (00:1a:a0:b3:55:05)
.... ..0. .... .. = LG bit: Globally unique address (factory default)
.... ...0 .... .. = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.0.101, Dst: 192.168.0.103
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
0000 00.. = Differentiated Services Codepoint: Default (0)
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 84
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
0... .... = Reserved bit: Not set

```

**Readable\_File.txt**

### c) Reviewing Shell Scripts

In general, operating systems use two types of interface: the **GUI**, which is the desktop that windows, etc. boots into AND the **command line**. In Linux command lines are executed using a Shell. A *Linux Shell* is a program, which facilitates the interaction between the user and the Linux operating system (Kernel). There are many shells available including sh, bash, csh, zsh and must more.

For example using bash Shell we can execute the following Linux commands :

- ls; pwd. #two commands on the same line
- rm junk\_file
- cp junk1 jun2
- compgen -c #dispalys all the commands
- uname -a #Linux Dist. Version
- dpkg -l | grep wget it is installed
- apt list | grep wget
- wget <http://aitislab.com/tecmint-14-09-12.tar>
- ps -u ssuee | grep ssh
- who -q
- users | wc -w

- `tar -xvf tecmint-14-09-12.tar`

*Shell scripting* is a way of automating execution of a *collection of commands*. In such cases, the execution of the commands is implemented based on the predefined control statements. In general many different text editors can be used to edit a shell script including `xed`, `vi`, `nano`, `vim`, `GNU emacs`, and so on.

Two of the most common shells are *SH* and *Bash Shell*. *SH* is the original (Bourne) Shell, having its root in the old Unix. Another type of shell is *Bash Shell* (stands for Bourne Again SHell). It was created by Brian Fox using *SH* but it has many advanced features. *Bash* is typically runs in a text window. A very simple *BASH* shell script called `simple.sh` is shown below:

```
#!/bin/bash
echo "HELLO WORLD"
```

Note that `#!/bin/bash` tells the system what Shell to use. The hash (`#`) denotes a comment line, one that is ignored by the system, the exclamation mark (`!`) means the comment is bypassed and will force the script to execute the line a command. This is also called *Hash-Bang*.

This is an example of commenting a line or commenting an entire block.

```
: <<'END'
    comments' here
    and here
END
```

Here is an example of getting an input form the user:

```
#!/bin/bash
echo -n "Is this a good question (y/n)? "
read answer
if
    echo "$answer" | grep -iq "^y" ;then
    echo Yes
else
    echo No
fi
```

Assuming the name of above file is `simple.sh`, we can make it executable using `chmod 777 simple.sh` and then run the program using `./simple.sh`. In the above file, `simple.sh`, we can add more Linux commands, including something like `tcpdump icmp`. Add the following commands in your script and see what happens:

```
now=$(date "+%H:%M:%S %d/%m/%y")
echo "Current time is: $now"
```

An infinite while loop can be implemented in the following way:

```
while true
do
    echo hello
done
```

It is possible to use IF and CASE statements in a shell script. For example consider the following examples (see appendix for more examples):

```
#!/bin/bash
firstname=$1
secondname=$2
echo Hello $firstname $secondname

if [ "$firstname" == "Farid" ]
then echo $firstname is an awesome name!
else
echo Hello $firstname
fi
```

```
#!/bin/bash
printf 'Which Linux distribution do you know? '
read DISTR
case $DISTR in
ubuntu)
echo "I know it! It is an operating system based on Debian."
;;
centos|rhel)
echo "Hey! It is my favorite Server OS!"
;;
windows)
echo "Very funny..."
;;
*)
echo "Hmm, seems i've never used it."
;;
esac
```

#### d) PING Utility Review

Ping is a diagnostic tool that uses the **Internet Control Message Protocol (ICMP)** protocol. ICMP is a supporting protocol in the Internet protocol suite. It is used by network devices, including routers to send error messages and operational information. For example using ping utility we can identify if is not is reachable or not.

The figure below shows the ICMP packet format. In this figure, there are several interesting points that must be notes:

- 1- ICMP is embedded in IP and has no transport layer.
- 2- The Ethernet header is 14 bytes long, the IP header is 20 bytes long, and the ICMP header is 8 bytes long.
- 3- Note that the payload size of the ICMP in this case is 32 bytes.
- 4- The total number of bytes in this Ethernet frame is  $42+32=74$  bytes, which is consistent with the size of the frame 9, as shown below.

Using PING utility we can change the characteristics of the ICMP packets, such as size, the number of times we send the ICMP packets, the data pattern of the ICMP payload, etc. For example consider the following



examples:

```
ping -c 1 -s 10 -p abcd 10.0.0.23
```

In this case we send only one packet (-c 1). We limit the size of the data payload to 10 bytes (remember in the figure below the size of the ICMP payload (data) is 64 bytes) using -s 10. The data pattern in this case will be abcd as defined by -p option.

```

▶ Frame 9: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▶ Ethernet II, Src: Dell_02:94:89 (5c:26:0a:02:94:89), Dst: CameoCom_03:47:56 (00:18:e7:03:47:56)
▶ Internet Protocol Version 4, Src: 192.168.1.102, Dst: 130.157.5.226
▼ Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x46ec [correct]
  [Checksum Status: Good]
  Identifier (BE): 1 (0x0001)
  Identifier (LE): 256 (0x0100)
  Sequence number (BE): 1647 (0x066f)
  Sequence number (LE): 28422 (0x6f06)
  [Response frame: 10]
▶ Data (32 bytes)

```

Answer the following questions:

**2.d.1)** Using `tcpdump -X icmp` command capture at least one ICMP packet and save the results in `Capture_File.pcap`.

**2.d.2)** Using Wireshark show the content of the file you just created. Take a snapshot of the Wireshark GUI.

**2.d.2)** Using `tshark` convert the binary file called `Capture_File.pcap` into a text file and show its content. What is the size of the ICMP packet, including its header. Explain your answer.

### e) Part II Project: Control an LED Remotely using TCPDUMP

This project must be implemented on the RPI using a shell script. In this project you must write a shell script such that when the PI receives a ping command with a specific byte pattern then it must turn on or off an LED. For example if the ping command's pattern is "aa ae" then the PI must turn on the LED. If the ping command's pattern is "ab ae" then the PI must turn off the LED. If the ping command's pattern is anything else, then the PI should do nothing and only **print** on the terminal that a PING message was just received! For extra credit you can interface the LCD with your PI to print the appropriate message as you ping the PI. **You must show the working program to the instructor to receive credit for this part.** Submit your code.

## F. Credits

Special thanks to online resources and all SSU students who assisted putting together this lab.

## E. References

- About tcpdump: <http://www.ronnutter.com/raspberry-pi-intro-to-tcpdump/>
- Learn about tshark: [https://www.wireshark.org/docs/wsug\\_html\\_chunked/AppToolstshark.html](https://www.wireshark.org/docs/wsug_html_chunked/AppToolstshark.html)

## Appendix

## Shell Script Examples

<pre>#Example 1 - General printing #!/bin/bash echo hello world! echo "Hello world!" echo 'Hello World' echo "Bash version \${BASH_VERSION}..."  #Example - Comment a block : &lt;&lt;'END'     comments' here     and here END</pre>	<pre>#Example 2 - usage ./filename a b #!/bin/bash firstname=\$1 secondname=\$2 echo Hello \$firstname \$secondname  if [ "\$firstname" == "Farid" ]     then echo     \$firstname is an awesome name! else     echo Hello \$firstname fi  #Example 3 - - usage ./filename 2 3 #!/bin/bash firstnumber=\$1 secondnumber=\$2 echo Sum of the numbers are: \$(( \$firstnumber+\$secondnumber))</pre>
<pre>#Example 6 - case statement #!/bin/bash printf 'Which Linux distribution do you know? ' read DISTR case \$DISTR in     ubuntu)         echo "I know it! It is an operating system based on Debian."         ;;     centos rhel)         echo "Hey! It is my favorite Server OS!"         ;;     windows)         echo "Very funny..."         ;;     *)         echo "Hmm, seems i've never used it."         ;; esac</pre>	<pre>#Example 4 - while loop #!/bin/bash count=0 # continue until \$n equals 5 while [ \$count -lt 5 ] # -lt is a comparison command do     echo "Welcome \$count times."     count=\$(( count+1 )) # increments \$count done  #Example 5 - for loop #!/bin/bash for i in {0..10}; do     echo "Welcome \$i times" done</pre>
<pre>#Example 7 - coloring #!/bin/bash #Bold High Intensity BIBlack='\033[1;90m'      # Black BIRed='\033[1;91m'       # Red BIGreen='\033[1;92m'     # Green BIYellow='\033[1;93m'    # Yellow BIBlue='\033[1;94m'      # Blue BIPurple='\033[1;95m'    # Purple BICyan='\033[1;96m'     # Cyan BIWhite='\033[1;97m'     # White NC='\033[0m' # No Color printf "I \${BIGreen}love\${NC} the Networking Class!\n" #Set background colors echo -e "\x1B[31;43m a \x1B[0m b \x1B[0;92m c \x1B[103;33m d \x1B[0;94m\${NC}"</pre>	<pre>#Example 8 - Reading the input and time #!/bin/bash echo -n "What is your first number?" #leaves the cursor at the same line read firstnumber echo -n "What is your second number?" read secondnumber clear echo "The sum of the numbers is \${(\$firstnumber+\$secondnumber)}" now=\$(date +%A) #shows today's date - try %B and %Y echo "Today is: \$now"</pre>

**ANSWER TO THE QUESTIONS:**

**2.a.1)** What is the command to filter only packets with source port address of 1025?

```
tcpdump src port 1025
```

**2.a.2)** What is the command to filter only ping packets initiated from a machine with IP address 192.168.0.123?

```
tcpdump -X icmp src 192.168.0.123
```

**2.a.3)** What is the command to filter packets originated from a machine with IP address 10.0.0.2 directed to a machine with IP address 10.0.0.3? `tcpdump src 10.0.0.2 and src 10.0.0.3`

**2.a.4)** What is the tcpdump command to list interfaces that tcpdump can listen on? `Tcpdump -D`

**2.a.5)** What does the following command do? `tcpdump -nni <your_interface> icmp`

```
19:50:51.916937 IP 172.217.2.14 > 172.19.1.205: ICMP 172.217.2.14 udp port 443 unreachable, length 36
```