EE405 Electronic Design Lab

Lab 4. Communication and Camera

I. Purpose

This lab aims to understand socket programming, implement a remote video streaming system using a Webcam on the Bone, and design a remote keyboard commander for Beaglebone on PC. Finally, you will use this part for the teleoperation of the robot. You will see the webcam (which is attached to the robot) on the PC, and teleoperate arm.

II. Problem Statement

Problem 4A.

Test server and client example using datagram socket(UDP).

Problem 4B.

Implement a remote keyboard commander on PC to teleoperate a robot manipulator using WiFi and datagram socket.

Problem 4C.

Test Webcam streaming via FFmpeg and Implement video functionality composed of

- Camera on Beaglebone (capture from webcam and send video to network)
- Viewer on PC (Receive video from network and display video to the user)

III. Lab Procedures

**** WEEK 1 only ****

A. Check Webcam and Ethernet connection

0. Install ffmpeg ~~(Already done by TAs)~~

(The TAs were expected to complete this part because it requires a significant amount of time, but we were unable to access it due to a login issue with Debian. We sincerely apologize for any inconvenience caused)

To use FFmpeg for Webcam streaming(later), we need to install a high-version of FFmpeg. It may take 1-2 hours, so install FFmpeg now to save time.

DO NOT TURN OFF BEAGLEBONE BEFORE THE INSTALLATION IS COMPLETE.

ON PC Shell:

$ wget http://ffmpeg.org/releases/ffmpeg-4.2.3.tar.gz

Transfer file from PC to Bone (Still on PC shell):

$ scp ffmpeg-4.2.3.tar.gz debian@192.168.7.2:~/ffmpeg.tar.gz

Now, move on the ***Bone shell.***

$ tar -xvzf ffmpeg.tar.gz

$ cd ffmpeg

$ ./configure && make

$ sudo make install

Open another shell and follow the below procedures!

1. Connect Loigitech C270 Webcam on the Bone before booting.

2. Check the Webcam is connected well

Check the Webcam is successfully detected on the Bone by typing 'lsusb'.

```
debian@beaglebone:~$ lsusb

Bus 001 Device 003: ID 046d:0825 Logitech, Inc. Webcam C270
Bus 001 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

3. Check WiFi configuration on Bone.

Test ifconfig:

```
debian@beaglebone:~$ ifconfig
eth0: flags=-28605<UP,BROADCAST,RUNNING,MULTICAST,DYNAMIC>  mtu 1500
        inet 192.168.50.128
...
...
```

This shows wired Ethernet connection with **IP 192.168.50.128.** (Your IP address will be different from this, but don't worry and use yours.)

B. Test Stream Socket & Datagram Socket

11. Create a working directory

$ mkdir -p ~/DesignLab/Network/Socket

12. Download and build files

Please refer https://beej.us/guide/bgnet/html/split/, Section 6. Download server.c, client.c, listener.c, and talker.c.

Move server.c and listener.c file to the Bone and build, and build client and talker files, too(on the PC).

Test server on Bone, client on PC, via wired Ethernet.

13. Test server-client example

Run client without server

Bone	**PC**
	$ ./client 192.168.50.128 client: connect: Connection refused client: failed to connect

PC cannot connect because the server(on the Bone) is not working.

***192.168.50.128* in the above example is an IP address of the Ethernet(of the Bone).**

Run the server first, and then run the client.

Bone	**PC**
$ ./server	
server: waiting for connections…	
	// Client via wired Ethernet
	$ ./client 192.168.50.128
	client: connecting to 192.168.50.128
server: got connection from 192.168.100.112	
	client: received 'Hello, world!'

Notice that server_pc is still waiting for another client.
Note that you may use 192.168.50.128 (IP of the Bone) instead of PC IP.
Input Crtl+C to quit the server.

14. Test listener-talker example

Test server on Bone, client on PC, via wired Ethernet.

Bone	**PC**
# ./listener_bone	
listener: waiting to recvfrom...	
	$ ./talker_pc 192.168.50.128 "Can you hear me?"
	talker: sent 16 bytes to 192.168.50.128
listener: got packet from 192.168.100.112	
listener: packet is 16 bytes long	
listener: packet contains "Can you hear me?"	

C. Remote Commander via UDP

15. Create a working directory
$ mkdir -p ~/DesignLab/WiFi/Remote_Commander

16. Implement commander and control method

Download Skeleton code for Lab4 on the KLMS.

Implement Remote_Commander_PC_skeleton.c and WiFi_Control_TMR_skeleton.c with UDP to send keyboard input. It will be used for robot teleoperation.

Build and test the codes. Does it work properly?

Tip. You can reuse most parts of the listener-talker file for implementation.

D. Stream Webcam Screen via V4L2 and FFmpeg

18. Create a working directory

$ mkdir -p ~/DesignLab/Webcam

19. Check for Webcam device and UVC device driver

First issue “lsusb” with connecting Webcam to Bone. (We already checked in the section 2)

$ lsusb

Bus 001 Device 003: ID 046d:0829 Logitech, Inc.

…

Check UVC device driver.

Linux 2.6.26 and newer includes the Linux UVC driver natively. You will not need to download the driver sources manually unless you want to test a newer version or help with development.

 # lsmod | grep uvc

uvcvideo 57013 0

videobuf2_vmalloc 2490 1 uvcvideo

When you do “ls /dev”, you should see video0 at the bottom:

 # ls /dev

/dev/audio1

…...

/dev/video0

20. Install Video4Linux2 and Test

Video4Linux, or V4L, is a video capture application programming interface for Linux, supporting many USB webcams, TV tuners, and other devices. Video4Linux is closely integrated with the Linux kernel.

Install v4l2 on Bone:

$ sudo apt-get update

$ sudo apt-get install v4l-utils

$ sudo apt-get install libv4l-dev

If it doesn't work, please edit */etc/apt/source/list* :

deb http://archive.debian.org/debian wheezy main contrib non-free
deb http://archive.debian.org/debian-security wheezy/updates main

and try again.

You can see or control camera detail info.

To list the devices:

 # v4l2-ctl --list-devices

To list the formats available:

 # v4l2-ctl --list-formats

Get all of the information available for the camera:

 # v4l2-ctl --all

If we wish to modify the resolution:

 # v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=1

Check current format

 # v4l2-ctl -V

21. Download capture.c, modify, and test

Download capture.c from https://github.com/derekmolloy/boneCV and modify Line 493 from:

```
if (force_format==2){
        fmt.fmt.pix.width       = 1920;
        fmt.fmt.pix.height      = 1080;
        fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_H264;
        fmt.fmt.pix.field       = V4L2_FIELD_INTERLACED;
        }
```

To:

```
if (force_format==2){
        fmt.fmt.pix.width       = 1280;
        fmt.fmt.pix.height      = 720;
        fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_MJPEG;
        fmt.fmt.pix.field       = V4L2_FIELD_NONE;
        }
```

And fix line 191, too, as below.:

```
tv.tv_sec = 20;
// for longer timeout
```

Build capture.c with v4l2 package.

$ gcc capture.c -lv4l2 -o capture

22. Streaming the Webcam

Now we will stream the Webcam screen via FFmpeg.

On the **Bone**, type below:

$ ./capture -F -c 0 -o | ffmpeg -vcodec mjpeg -i pipe:0 -f mjpeg udp://224.0.0.1:1234

It means, streaming the Webcam screen through udp network with input as ***capture*** binary file.

On the **PC**, type:

$ ffplay -i udp://224.0.0.1:1234

Note that you typed the same network address on the Bone and the PC.

Can you see the webcam screen on the PC well?

Note. You may see some warning messages on the Bone, but you can ignore it.