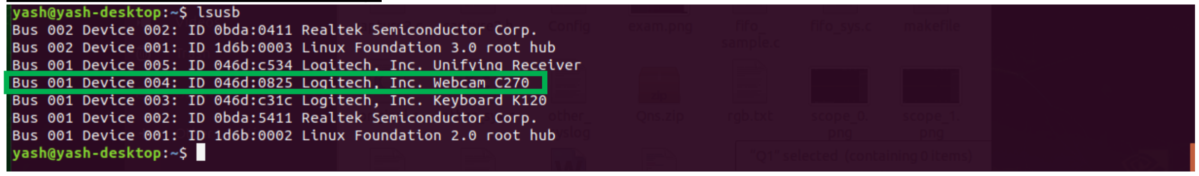
ECEN-5623 ASSIGNMENT-4

YASH GUPTE

Date: 19th July 2019

1) [10 points] Obtain a Logitech C270 camera (or equivalent) and verify that is detected by the Jetson board USB driver. Use lsusb, lsmod and dmesg kernel driver configuration tool to make sure your Logitech C200 USB camera is plugged in and recognized by your Jetson. Do lsusb | grep C200 and prove to me (and more importantly yourself) with that output (screenshot) that your camera is recognized. Now, do lsmod | grep video and verify that the UVC driver is loaded as well (http://www.ideasonboard.org/uvc/). To further verify, or debug if you don't see the UVC driver loaded in response to plugging in the USB camera, do dmesg | grep video or just dmesg and scroll through the log messages to see if your USB device was found. Capture all output and annotate what you see with descriptions to the best of your understanding.

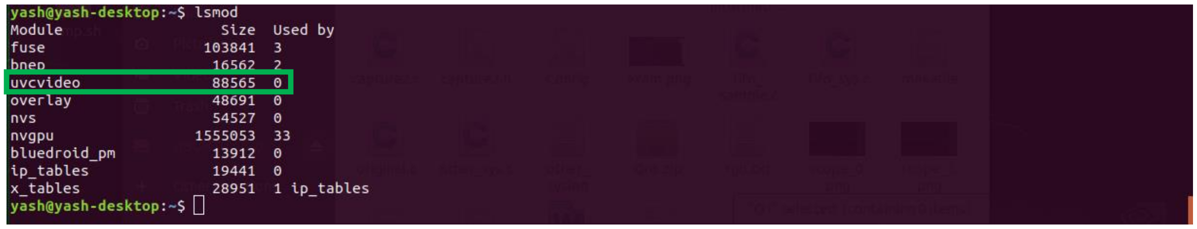
<u>**Commands and their meaning:**</u>



```
yash@yash-desktop:~$ lsusb
Bus 002 Device 002: ID 0bda:0411 Realtek Semiconductor Corp.
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 005: ID 046d:c534 Logitech, Inc. Unifying Receiver
Bus 001 Device 004: ID 046d:0825 Logitech, Inc. Webcam C270
Bus 001 Device 003: ID 046d:c31c Logitech, Inc. Keyboard K120
Bus 001 Device 002: ID 0bda:5411 Realtek Semiconductor Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
yash@yash-desktop:~$
```

Fig. 1-A Simple lsusb

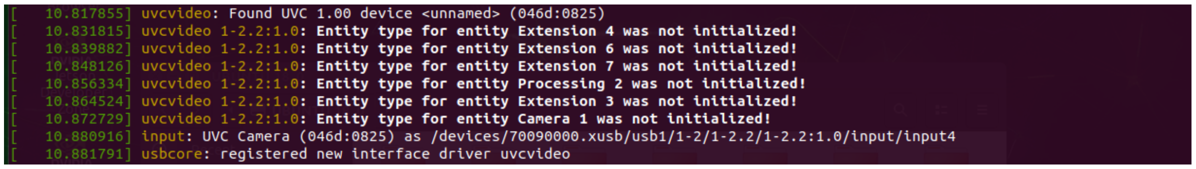
Lsusb -> Displays a list of connected USB devices. In the image above, webcam is denoted as device 004, with an ID 046d:0825.



```
yash@yash-desktop:~$ lsmod
Module                  Size  Used by
fuse                  103841  3
bnep                   16562  2
uvcvideo               88565  0
overlay                48691  0
nvs                    54527  0
nvgpu                1555053  33
bluedroid_pm           13912  0
ip_tables              19441  0
x_tables               28951  1 ip_tables
yash@yash-desktop:~$
```

Fig. 1-A Simple lsmod

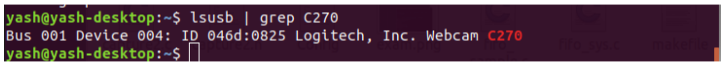
Uvcvideo -> Lsmod displays the information about the installed Kernel Modules. UVCVideo, is the camera driver installed for using the logitec C270 camera. Size of the module is 88565 bytes and currently unused by any module.



```
[   10.817855] uvcvideo: Found UVC 1.00 device <unnamed> (046d:0825)
[   10.831815] uvcvideo 1-2.2:1.0: Entity type for entity Extension 4 was not initialized!
[   10.839882] uvcvideo 1-2.2:1.0: Entity type for entity Extension 6 was not initialized!
[   10.848126] uvcvideo 1-2.2:1.0: Entity type for entity Extension 7 was not initialized!
[   10.856334] uvcvideo 1-2.2:1.0: Entity type for entity Processing 2 was not initialized!
[   10.864524] uvcvideo 1-2.2:1.0: Entity type for entity Extension 3 was not initialized!
[   10.872729] uvcvideo 1-2.2:1.0: Entity type for entity Camera 1 was not initialized!
[   10.880916] input: UVC Camera (046d:0825) as /devices/70090000.xusb/usb1/1-2/1-2.2/1-2.2:1.0/input/input4
[   10.881791] usbcore: registered new interface driver uvcvideo
```

Fig. 1-A Simple dmesg

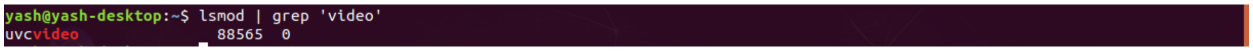
Dmesg displays all the kernel ring buffer messages when called without arguments. Here uvcvideo represents messages from the UVC kernel module. 046D:0825 is the device ID of the USB mouse.



```
yash@yash-desktop:~$ lsusb | grep C270
Bus 001 Device 004: ID 046d:0825 Logitech, Inc. Webcam C270
yash@yash-desktop:~$
```

Fig. 1-A Simple lsusb | grep

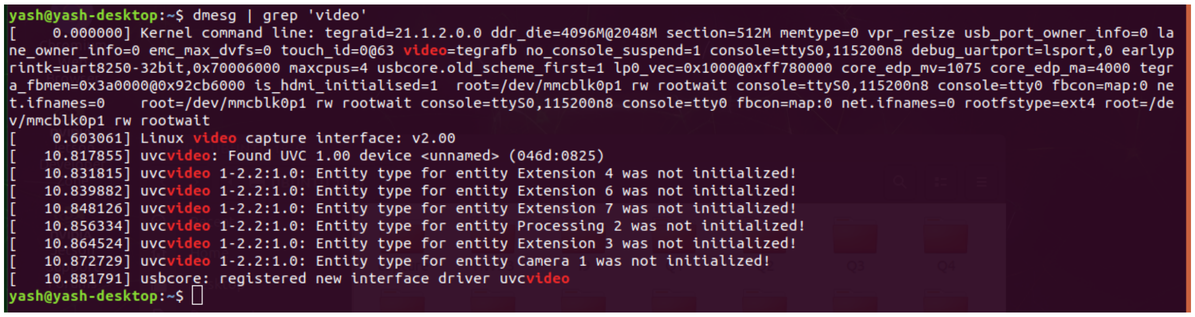
Grep searches for specific string in the listed options. In this case, lsusb list allthe connected USB devices and grep 270 represents only the specific listing with C270 in it.



```
yash@yash-desktop:~$ lsmod | grep 'video'
uvcvideo               88565  0
```

Fig. 1- lsmod | grep video

Lsmod lists the kernel modules and grep 'video' lists the kernel modules with 'video' present in their name. Here, uvcvideo represents the specific video string.

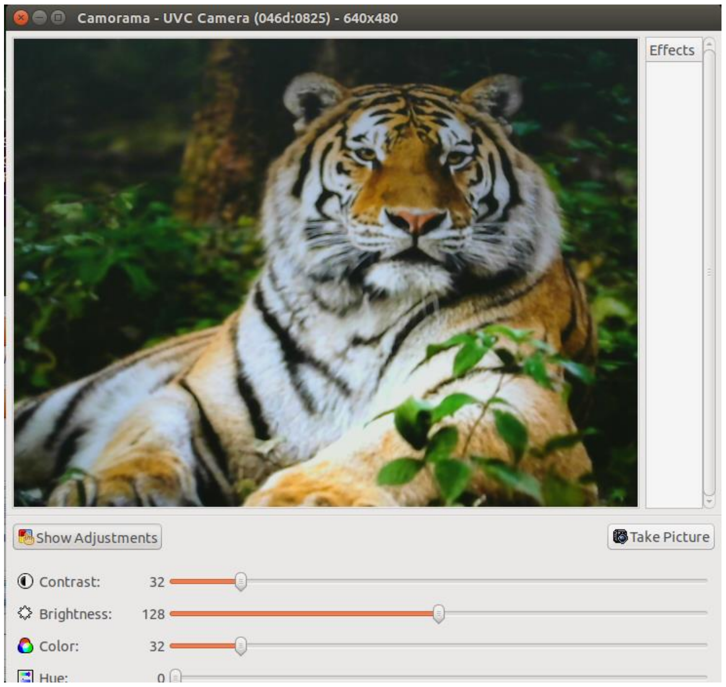


```
yash@yash-desktop:~$ dmesg | grep 'video'
[    0.000000] Kernel command line: tegraid=21.1.2.0.0 ddr_die=4096M@2048M section=512M memtype=0 vpr_resize usb_port_owner_info=0 la
ne_owner_info=0 emc_max_dvfs=0 touch_id=0@63 video=tegrafb no_console_suspend=1 console=ttyS0,115200n8 debug_uartport=lsport,0 earlyp
rintk=uart8250-32bit,0x70006000 maxcpus=4 usbcore.old_scheme_first=1 lp0_vec=0x1000@0xff780000 core_edp_mv=1075 core_edp_ma=4000 tegr
a_fbmem=0x3a0000@0x92cb6000 is_hdmi_initialised=1  root=/dev/mmcblk0p1 rw rootwait console=ttyS0,115200n8 console=tty0 fbcon=map:0 ne
t.ifnames=0    root=/dev/mmcblk0p1 rw rootwait console=ttyS0,115200n8 console=tty0 fbcon=map:0 net.ifnames=0 rootfstype=ext4 root=/de
v/mmcblk0p1 rw rootwait
[    0.603061] Linux video capture interface: v2.00
[   10.817855] uvcvideo: Found UVC 1.00 device <unnamed> (046d:0825)
[   10.831815] uvcvideo 1-2.2:1.0: Entity type for entity Extension 4 was not initialized!
[   10.839882] uvcvideo 1-2.2:1.0: Entity type for entity Extension 6 was not initialized!
[   10.848126] uvcvideo 1-2.2:1.0: Entity type for entity Extension 7 was not initialized!
[   10.856334] uvcvideo 1-2.2:1.0: Entity type for entity Processing 2 was not initialized!
[   10.864524] uvcvideo 1-2.2:1.0: Entity type for entity Extension 3 was not initialized!
[   10.872729] uvcvideo 1-2.2:1.0: Entity type for entity Camera 1 was not initialized!
[   10.881791] usbcore: registered new interface driver uvcvideo
yash@yash-desktop:~$ 
```

Fig. 1- dmesg | grep video

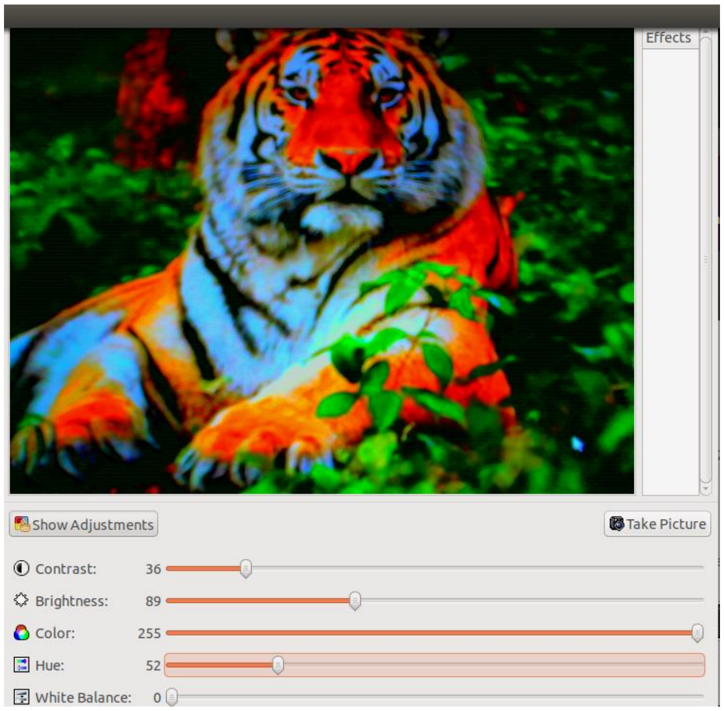
Dmesg represents all the kernel ring buffer messages and grep video represents the specific video containing messages.

2) [10 points] If you do not have camorama, do apt-get install camorama on your Jetson board [you may need to first do sudo add-apt-repository universe; sudo apt-get update]. This should not only install nice camera capture GUI tools, but also the V4L2 API (described well in this series of Linux articles - http://lwn.net/Articles/203924/). Running camorama should provide an interactive camera control session for your Logitech C2xx camera – if you have issues connecting to your camera do a "man camorama" and specify your camera device file entry point (e.g. /dev/video0). Run camorama and play with Hue, Color, Brightness, White Balance and Contrast, take an example image and take a screen shot of the tool and provide both in your report. If you need to resolve issues with sudo and your account, research this and please do so.



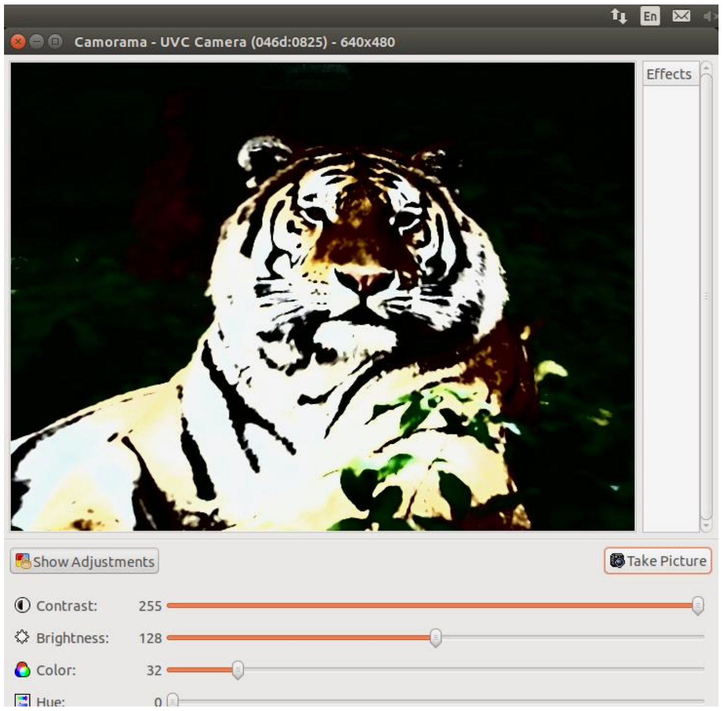
Contrast: 32
Brightness: 128
Color: 32
Hue: 0

Fig. 2-1- Original



Contrast: 36
Brightness: 89
Color: 255
Hue: 52
White Balance: 0

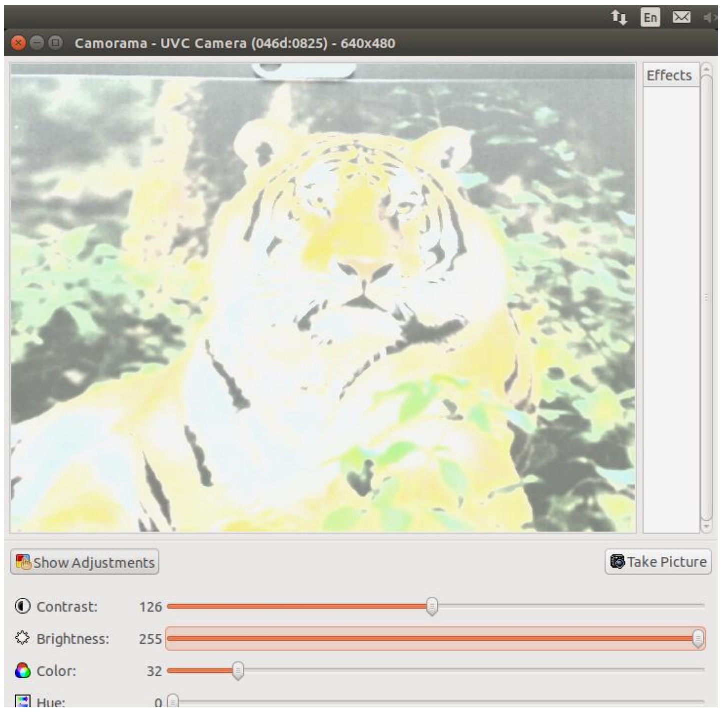
Fig. 2-2- Hue



Camorama - UVC Camera (046d:0825) - 640x480

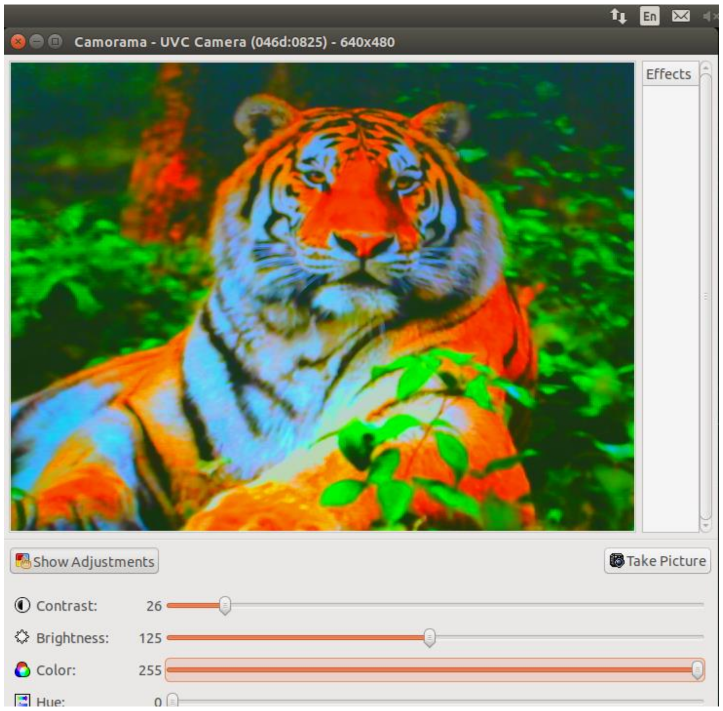
Contrast: 255
Brightness: 128
Color: 32
Hue: 0

Fig. 2-3- Contrast



Contrast: 126
Brightness: 255
Color: 32
Hue: 0

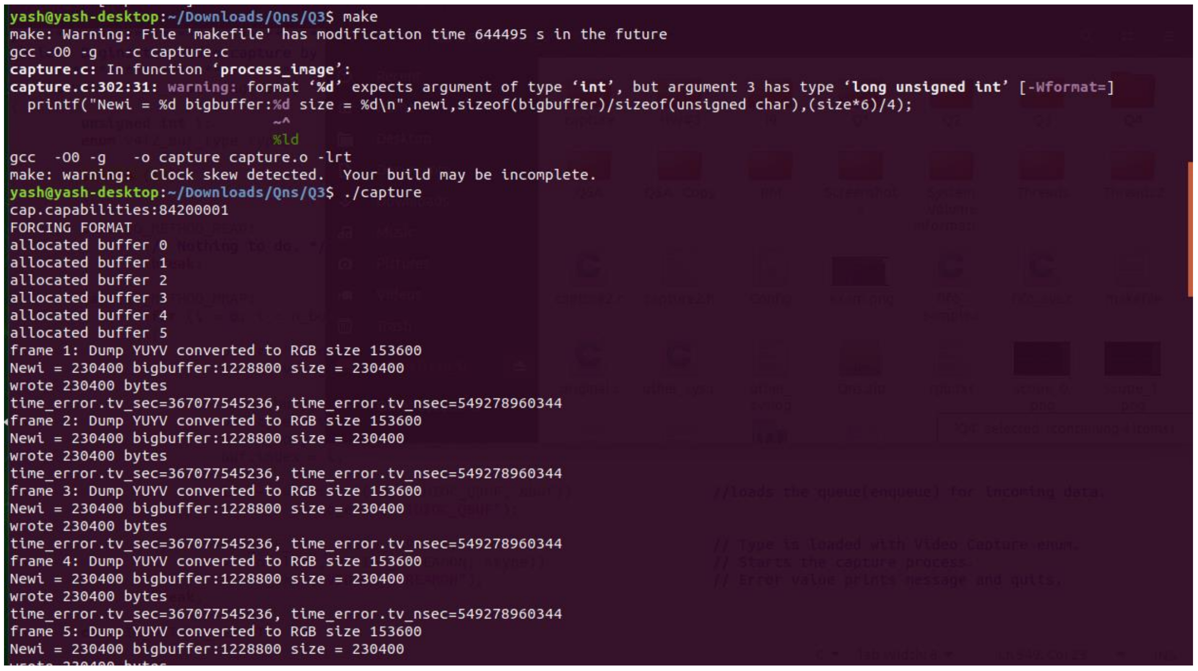
Fig. 2-4- Brightness



Contrast: 26
Brightness: 125
Color: 255
Hue: 0

Fig. 2-5- Color

3) [10 points] Using your verified Logitech C270 camera on an R-Pi or Jetson and verify that it can stream continuously using to a raw image buffer for transformation and processing using example code such as simple-capture. This basic example interfaces to the UVC and USB kernel driver modules through the V4L2API. Alternatively, you can use OpenCV, which abstracts the camera interface and provides useful library functions with code found here - simpler-capture-2/. Provide a screen shot to prove that you got continuous capture to work. Note that simpler capture may require installation of OpenCV on an R-Pi 3b (on the Jetson Nano it is pre-installed) – this will likely already be available on your board, but if not, please follow https://www.learnopencv.com/install-opencv-3-4-4-on-raspberry-pi/ on the RPi and simple instructions found here to install openCV on a Jetson [the "Option 2, Building the public OpenCV library from source" is the recommended approach with – DWITH_CUDA=OFF unless you install CUDA 6.0 from here, either way you must have a consistent CUDA install for your R19 or R21 board, so if in doubt, don't install CUDA and leave it off when you build OpenCV].



```
yash@yash-desktop:~/Downloads/Qns/Q3$ make
make: Warning: File 'makefile' has modification time 644495 s in the future
gcc -O0 -g    -c capture.c
capture.c: In function ‘process_image’:
capture.c:302:31: warning: format ‘%d’ expects argument of type ‘int’, but argument 3 has type ‘long unsigned int’ [-Wformat=]
  printf("Newi = %d bigbuffer:%d size = %d\n",newi,sizeof(bigbuffer)/sizeof(unsigned char),(size*6)/4);
                              ~^
                              %ld
gcc  -O0 -g   -o capture capture.o -lrt
make: warning:  Clock skew detected.  Your build may be incomplete.
yash@yash-desktop:~/Downloads/Qns/Q3$ ./capture
cap.capabilities:84200001
FORCING FORMAT
allocated buffer 0
allocated buffer 1
allocated buffer 2
allocated buffer 3
allocated buffer 4
allocated buffer 5
frame 1: Dump YUYV converted to RGB size 153600
Newi = 230400 bigbuffer:1228800 size = 230400
wrote 230400 bytes
time_error.tv_sec=367077545236, time_error.tv_nsec=549278960344
frame 2: Dump YUYV converted to RGB size 153600
Newi = 230400 bigbuffer:1228800 size = 230400
wrote 230400 bytes
time_error.tv_sec=367077545236, time_error.tv_nsec=549278960344
frame 3: Dump YUYV converted to RGB size 153600
Newi = 230400 bigbuffer:1228800 size = 230400
wrote 230400 bytes
time_error.tv_sec=367077545236, time_error.tv_nsec=549278960344
frame 4: Dump YUYV converted to RGB size 153600
Newi = 230400 bigbuffer:1228800 size = 230400
wrote 230400 bytes
time_error.tv_sec=367077545236, time_error.tv_nsec=549278960344
frame 5: Dump YUYV converted to RGB size 153600
Newi = 230400 bigbuffer:1228800 size = 230400
```

Fig. 3-1- Build and Run

Code Explanation:

The code is designed to convert the incoming data stream from the camera into RGB. The code provides a user input-based code where users can select the mode of operation (more on this in the code comments). It could be Read-I/O, Memory mapped, or user pointer based. By default, the code runs in Memory Mapped mode. The code includes separate functions for capturing a frame, processing the image and stopping the capture process.

1. The first function is open_device(). This function reads the string input by the user indicating the valid port of the camera. If data was not entered, 'dev0' is selected as the default port. ISCHR is used to check if the specified device exists on the com-port. It then opens the COM-Port with the camera on it.
2. The next function init_device() declares various V4L2 buffers which include cap, cropcap and crop. First the device capabilities are checked, primarily video capture and video streaming which are needed to process the images captured by the camera. This is followed by queries to check if cropping is available to allow for different frame sizes. The height and width for the frame are then stored followed by the image format to be recorded while streaming this includes YUYV in this example. The

respective initialization function is then called. This includes init_mmap() in memory mapped option or init_userp() in case of user pointer. In the init_mmap() function an instance of ‘v4l2_requestbuffers’ is populated. Frame count, memory mapped option and video capture options are selected. Fixed size of buffers are calloced. For each count of the frame count, mmap(), is called. This function is unique to memory map functionality and involves mapping of device memory into application address space with suitable permissions (read and write).

3. Next, the video capturing process begins with start_capture(). The Video queue is loaded onto an instance of ‘v4l2_buffer’ using the VIDIOC_QBUF enumerator. The type v4l2_buf_type enum is loaded with Video capture enum value. The streaming process then begins with writing VIDIOC_STREAMON onto the type enum. Similar approach is applied to the user pointer method except that instead of directly using the buf v4l2_buffer, starting address of buffers is loaded into buf.m.userptr.
4. Mainloop() now begins reading each frame and processing the images with the predefined transforms. The frame read and processing is performed on 30 frames. 30 is a predefined number in the code. Firstly, a timeout value of 2s is used in the code. This timeout provides an additional margin for the execution of higher resolution image processing. The select() function waits up-to 2s before proceeding to read_frame(). A nanosleep() function ensures that on completion of a frame read, the system waits for 30us. This is useful in scheduling mechanisms to ensure a fixed time of service, especially in cyclic executive systems which can reduce the use of semaphores.
5. The read_frame() function proceeds depending on the mode selected at the start of the program. Read-I/O, Memory mapped IO or user pointer. In case of Read-I/O method, the buffer is merely read and not processed since the write capability is not provided. In MMIO method, xioctl() is used to ensure VIDIOC_DQBUF is used to ensure an empty buffer is available for storing the processed image.
6. Process_image() is used to apply various transformations to the captured frames. Pixelformat field of pix structure is used to check which format was used to capture images. In case of Grey scale capture, the image is directly converted to a ‘.pgm’ format using the dump_pgm() function. In case of YUYV format, yuv2rgb is invoked to convert the YUYV to RGBRGB-6 byte format. Then the image is stored in dump_ppm function. In case of RGB24 format, the image is stored directly as ‘.ppm’
7. Dump_pgm() and dump_ppm() are used to store the images in .ppm and .pgm format respectively. Each of these functions make use of predefined strings which contain the standard name ‘test’ for each image. The other header contains the format string for .ppm and .pgm formats. This is typically the height and width of the string along with a ‘\n255\n’. Based on the dimensions of the image set in HRES and VRES.
8. The yuv2rgb() and yuv2rgb_float() are standard library functions used to convert yuv to rgb and rgb in float respectively.
9. The stop_capture function stops the video streaming by writing VIDIOC_STREAMOFF into the fd file descriptor using xioctl.
10. The unint_device() function frees all buffers either used by user pointer or memory mapped I/O methods. In case of memory map-i/o munmap() is used to un-map the applications drivers from the devices memory. In case of user pointer, the calloced memory is freed.
11. Close_device() is used to close the camera opened by the file descriptor fd.

<u>*Continuous Verification can be noted from the screenshots uploaded in the Q3 folder on git.*</u>

4) [20 points] Choose ONE continuous transformation for each acquired frame such as color conversion (Lecture-Cont-Media.pdf, slide 5), conversion to grayscale, or sharpening (/sharpen-psf/sharpen.c) if you are using simple-capture and the V4L2 API. If you want to use OpenCV (not required, but can be fun and interesting) look at examples from computer-vision then you can choose transformations such as the canny-interactive, hough-interactive, hough-eliptical-interactive, or stereo-transform-impoved. Show a screen shot to prove you built and ran the code. Provide a detailed explanation of the code and research uses for the continuous transformation and describe.

__Overview:__

The code is designed to convert the incoming data stream from the camera into RGB. The code provides a user input-based code where users can select the mode of operation (more on this in the code comments). It could be Read-I/O, Memory mapped, or user pointer based. By default, the code runs in Memory Mapped mode. The code includes separate functions for capturing a frame, processing the image and stopping the capture process.

12. The first function is open_device(). This function reads the string input by the user indicating the valid port of the camera. If data was not entered, 'dev0' is selected as the default port. ISCHR is used to check if the specified device exists on the com-port. It then opens the COM-Port with the camera on it.
13. The next function init_device() declares various V4L2 buffers which include cap, cropcap and crop. First the device capabilities are checked, primarily video capture and video streaming which are needed to process the images captured by the camera. This is followed by queries to check if cropping is available to allow for different frame sizes. The height and width for the frame are then stored followed by the image format to be recorded while streaming this includes YUYV in this example. The respective initialization function is then called. This includes init_mmap() in memory mapped option or init_userp() in case of user pointer. In the init_mmap() function an instance of 'v4l2_requestbuffers' is populated. Frame count, memory mapped option and video capture options are selected. Fixed size of buffers are calloced. For each count of the frame count, mmap(), is called. This function is unique to memory map functionality and involves mapping of device memory into application address space with suitable permissions (read and write).
14. Next, the video capturing process begins with start_capture(). The Video queue is loaded onto an instance of 'v4l2_buffer' using the VIDIOC_QBUF enumerator. The type v4l2_buf_type enum is loaded with Video capture enum value. The streaming process then begins with writing VIDIOC_STREAMON onto the type enum. Similar approach is applied to the user pointer method except that instead of directly using the buf v4l2_buffer, starting address of buffers is loaded into buf.m.userptr.
15. Mainloop() now begins reading each frame and processing the images with the predefined transforms. The frame read and processing is performed on 30 frames. 30 is a predefined number in the code. Firstly, a timeout value of 2s is used in the code. This timeout provides an additional margin for the execution of higher resolution image processing. The select() function waits up-to 2s before proceeding to read_frame(). A nanosleep() function ensures that on completion of a frame read, the system waits for 30us. This is useful in scheduling mechanisms to ensure a fixed time of service, especially in cyclic executive systems which can reduce the use of semaphores.
16. The read_frame() function proceeds depending on the mode selected at the start of the program. Read-I/O, Memory mapped IO or user pointer. In case of Read-I/O method, the buffer is merely read and not processed since the write capability is not provided. In MMIO method, xioctl() is used to ensure VIDIOC_DQBUF is used to ensure an empty buffer is available for storing the processed image.
17. Process_image() is used to apply various transformations to the captured frames. Pixelformat field of pix structure is used to check which format was used to capture images. In case of Grey scale capture, the image is directly converted to a '.pgm' format using the dump_pgm() function. In case of YUYV

format, yuv2rgb is invoked to convert the YUYV to RGBRGB-6 byte format. Then the image is stored in dump_ppm function. In case of RGB24 format, the image is stored directly as '.ppm'

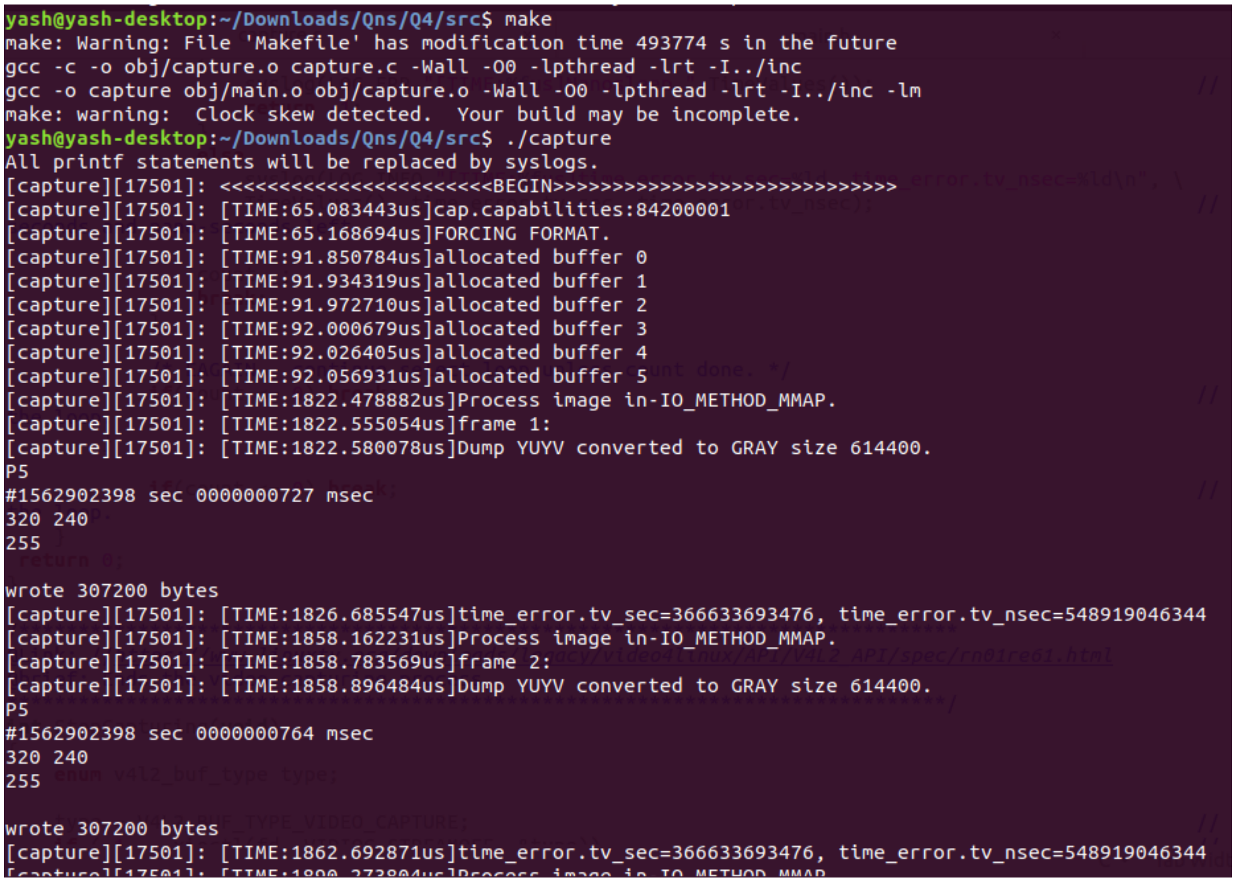
18. Dump_pgm() and dump_ppm() are used to store the images in .ppm and .pgm format respectively. Each of these functions make use of predefined strings which contain the standard name 'test' for each image. The other header contains the format string for .ppm and .pgm formats. This is typically the height and width of the string along with a '\n255\n'. Based on the dimensions of the image set in HRES and VRES.
19. The yuv2rgb() and yuv2rgb_float() are standard library functions used to convert yuv to rgb and rgb in float respectively.
20. The stop_capture function stops the video streaming by writing VIDIOC_STREAMOFF into the fd file descriptor using xioctl.
21. The unint_device() function frees all buffers either used by user pointer or memory mapped I/O methods. In case of memory map-i/o munmap() is used to un-map the applications drivers from the devices memory. In case of user pointer, the calloced memory is freed.
22. Close_device() is used to close the camera opened by the file descriptor fd.

<u>Modifications to 'simple_capture.c' for implementing conversion to greyscale.</u>

The functionality of 'simple_capture.c' is maintained for the most part of the code except for the following.

1. #define (MULTI_MODE) is used to run the code over 5 different resolutions. These include, 160x120, 320x240, 640x480, 800x600 and 960x720.
2. Yuvrgb is replaced with YUYV2GREY(). This function converts the YUYV to a greyscale image by only utilizing the YY component of the YUYV image.
3. Dump_ppm has not been used. Instead, only dump_pgm has been used. Since all conversions are neeed only in greyscale.

Screenshots:

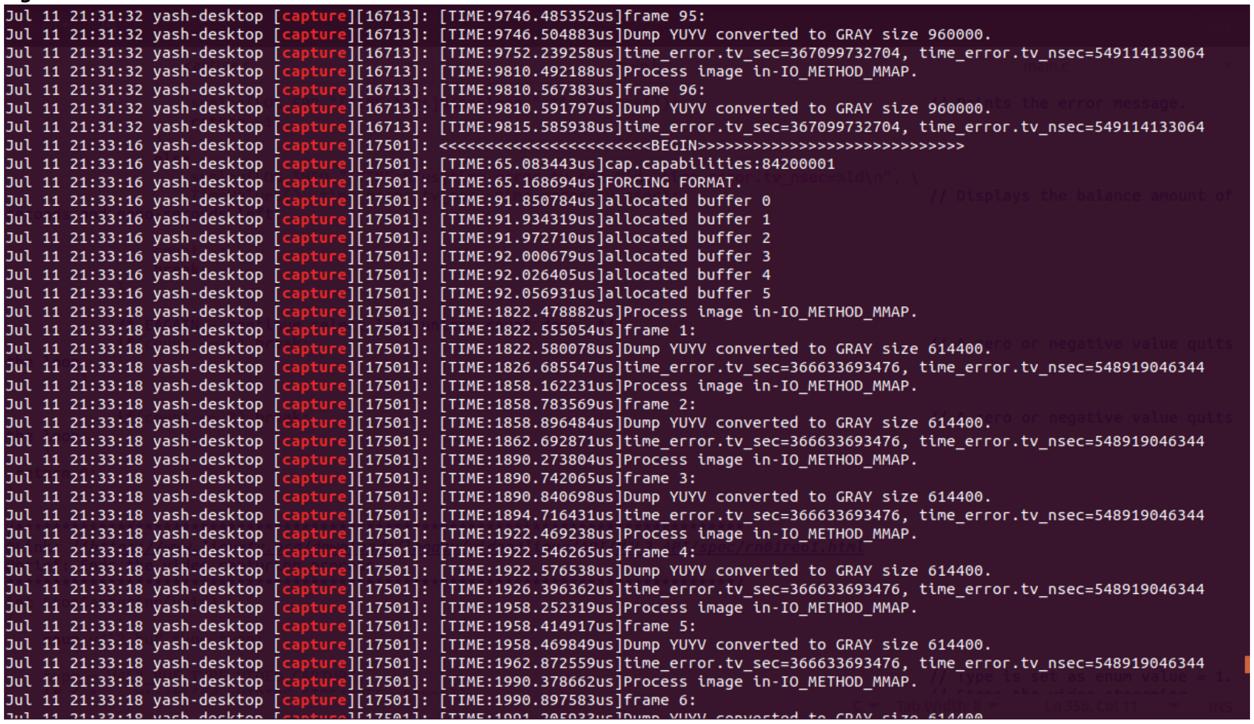


```
yash@yash-desktop:~/Downloads/Qns/Q4/src$ make
make: Warning: File 'Makefile' has modification time 493774 s in the future
gcc -c -o obj/capture.o capture.c -Wall -O0 -lpthread -lrt -I../inc
gcc -o capture obj/main.o obj/capture.o -Wall -O0 -lpthread -lrt -I../inc -lm
make: warning:  Clock skew detected.  Your build may be incomplete.
yash@yash-desktop:~/Downloads/Qns/Q4/src$ ./capture
All printf statements will be replaced by syslogs.
[capture][17501]: <<<<<<<<<<<<<<<<<<<<<<BEGIN>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
[capture][17501]: [TIME:65.083443us]cap.capabilities:84200001
[capture][17501]: [TIME:65.168694us]FORCING FORMAT.
[capture][17501]: [TIME:91.850784us]allocated buffer 0
[capture][17501]: [TIME:91.934319us]allocated buffer 1
[capture][17501]: [TIME:91.972710us]allocated buffer 2
[capture][17501]: [TIME:92.000679us]allocated buffer 3
[capture][17501]: [TIME:92.026405us]allocated buffer 4
[capture][17501]: [TIME:92.056931us]allocated buffer 5
[capture][17501]: [TIME:1822.478882us]Process image in-IO_METHOD_MMAP.
[capture][17501]: [TIME:1822.555054us]frame 1:
[capture][17501]: [TIME:1822.580078us]Dump YUYV converted to GRAY size 614400.
P5
#1562902398 sec 0000000727 msec
320 240
255

wrote 307200 bytes
[capture][17501]: [TIME:1826.685547us]time_error.tv_sec=366633693476, time_error.tv_nsec=548919046344
[capture][17501]: [TIME:1858.162231us]Process image in-IO_METHOD_MMAP.
[capture][17501]: [TIME:1858.783569us]frame 2:
[capture][17501]: [TIME:1858.896484us]Dump YUYV converted to GRAY size 614400.
P5
#1562902398 sec 0000000764 msec
320 240
255

wrote 307200 bytes
[capture][17501]: [TIME:1862.692871us]time_error.tv_sec=366633693476, time_error.tv_nsec=548919046344
```

Fig. 4-A -Make and execute commands.



```
Jul 11 21:31:32 yash-desktop [capture][16713]: [TIME:9746.485352us]frame 95:
Jul 11 21:31:32 yash-desktop [capture][16713]: [TIME:9746.504883us]Dump YUYV converted to GRAY size 960000.
Jul 11 21:31:32 yash-desktop [capture][16713]: [TIME:9752.239258us]time_error.tv_sec=367099732704, time_error.tv_nsec=549114133064
Jul 11 21:31:32 yash-desktop [capture][16713]: [TIME:9810.492188us]Process image in-IO_METHOD_MMAP.
Jul 11 21:31:32 yash-desktop [capture][16713]: [TIME:9810.567383us]frame 96:
Jul 11 21:31:32 yash-desktop [capture][16713]: [TIME:9810.591797us]Dump YUYV converted to GRAY size 960000.
Jul 11 21:31:32 yash-desktop [capture][16713]: [TIME:9815.585938us]time_error.tv_sec=367099732704, time_error.tv_nsec=549114133064
Jul 11 21:33:16 yash-desktop [capture][17501]: <<<<<<<<<<<<<<<<<<<<<<BEGIN>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Jul 11 21:33:16 yash-desktop [capture][17501]: [TIME:65.083443us]cap.capabilities:84200001
Jul 11 21:33:16 yash-desktop [capture][17501]: [TIME:65.168694us]FORCING FORMAT.
Jul 11 21:33:16 yash-desktop [capture][17501]: [TIME:91.850784us]allocated buffer 0
Jul 11 21:33:16 yash-desktop [capture][17501]: [TIME:91.934319us]allocated buffer 1
Jul 11 21:33:16 yash-desktop [capture][17501]: [TIME:91.972710us]allocated buffer 2
Jul 11 21:33:16 yash-desktop [capture][17501]: [TIME:92.000679us]allocated buffer 3
Jul 11 21:33:16 yash-desktop [capture][17501]: [TIME:92.026405us]allocated buffer 4
Jul 11 21:33:16 yash-desktop [capture][17501]: [TIME:92.056931us]allocated buffer 5
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1822.478882us]Process image in-IO_METHOD_MMAP.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1822.555054us]frame 1:
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1822.580078us]Dump YUYV converted to GRAY size 614400.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1826.685547us]time_error.tv_sec=366633693476, time_error.tv_nsec=548919046344
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1858.162231us]Process image in-IO_METHOD_MMAP.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1858.783569us]frame 2:
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1858.896484us]Dump YUYV converted to GRAY size 614400.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1862.692871us]time_error.tv_sec=366633693476, time_error.tv_nsec=548919046344
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1890.273804us]Process image in-IO_METHOD_MMAP.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1890.742065us]frame 3:
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1890.840698us]Dump YUYV converted to GRAY size 614400.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1894.716431us]time_error.tv_sec=366633693476, time_error.tv_nsec=548919046344
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1922.469238us]Process image in-IO_METHOD_MMAP.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1922.546265us]frame 4:
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1922.576538us]Dump YUYV converted to GRAY size 614400.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1926.396362us]time_error.tv_sec=366633693476, time_error.tv_nsec=548919046344
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1958.252319us]Process image in-IO_METHOD_MMAP.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1958.414917us]frame 5:
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1958.469849us]Dump YUYV converted to GRAY size 614400.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1962.872559us]time_error.tv_sec=366633693476, time_error.tv_nsec=548919046344
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1990.378662us]Process image in-IO_METHOD_MMAP.
Jul 11 21:33:18 yash-desktop [capture][17501]: [TIME:1990.897583us]frame 6:
```

Fig. 4-B – Syslog for Q4.

Conversion to grey scale has been performed. It can be viewed in the Q4 folder uploaded on it.

Need for continuous image transforms:

1. Grey scale helps reduce the variables that are needed to be observed on a color image. As a result, analysis of a grey scale image is easier. Modifications to the grey scale image can be made and tested followed by extending these changes to color images afterwards.
2. In images captured by the camera, especially RAW images can get blurry. This is because when images are captured by any camera lens even if HD, the sharpest portions are averaged out and slightly blurred. Successive image conversion to RGB can cause the carry forward of this blurred effect and makes it harder to resolve in a color channel-based image. Sharpening must be applied early on in the image processing steps.

5) [30 points] Using a Logitech C200, choose 3 real-time frame transformations to compare in terms of average frame rate at a given resolution for a range of at least 5 resolutions in a common aspect ratio (e.g. 4:3 for 1280x960, 640x480, 320x240, 160x120, 80x60) by adding time-stamps to analyze the potential throughput. Based on average analysis, pick a reasonable soft real-time deadline (e.g. if average frame rate is 12 Hz, choose a deadline of 100 milliseconds to provide some margin) and convert the processing to SCHED_FIFO and determine if you can meet deadlines with predictability and measure jitter in the frame rate relative to your deadline.

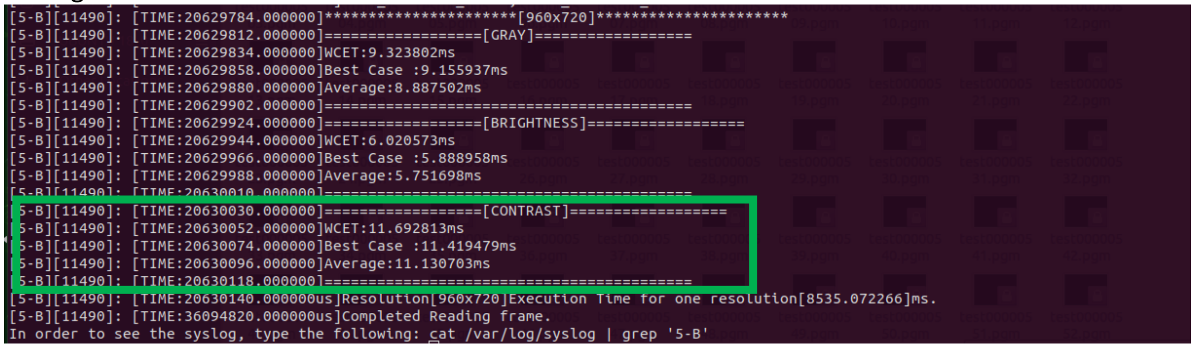
Ans-5A)

Structure of the program:

1. Three main threads run using FIFO. These are StartCapture, Mainloop and StopCapture.
2. The mainloop threads runs each transformation for 30 frames consecutively. That is, 30 frames for grey conversion, 30 for brightness and 30 for contrast.
3. The result includes WCET, best case execution time and average time of execution.
4. This helps us decide the deadline for the implementation with RM and scheduler in part-2 of this question.

Analysis:

It can be seen from the images below that the highest time of execution if for resolution 960x720. The highest time of execution for a resolution is for Contrast conversion. Which is at WCET of 11.69ms and average execution time of 11.13ms.



```
[5-B][11490]: [TIME:20629784.000000]*********************[960x720]*********************
[5-B][11490]: [TIME:20629812.000000]=================[GRAY]=================
[5-B][11490]: [TIME:20629834.000000]WCET:9.323802ms
[5-B][11490]: [TIME:20629858.000000]Best Case :9.155937ms
[5-B][11490]: [TIME:20629880.000000]Average:8.887502ms
[5-B][11490]: [TIME:20629902.000000]========================================
[5-B][11490]: [TIME:20629924.000000]=================[BRIGHTNESS]=================
[5-B][11490]: [TIME:20629944.000000]WCET:6.020573ms
[5-B][11490]: [TIME:20629966.000000]Best Case :5.888958ms
[5-B][11490]: [TIME:20629988.000000]Average:5.751698ms
[5-B][11490]: [TIME:20630010.000000]========================================
[5-B][11490]: [TIME:20630030.000000]=================[CONTRAST]=================
[5-B][11490]: [TIME:20630052.000000]WCET:11.692813ms
[5-B][11490]: [TIME:20630074.000000]Best Case :11.419479ms
[5-B][11490]: [TIME:20630096.000000]Average:11.130703ms
[5-B][11490]: [TIME:20630118.000000]========================================
[5-B][11490]: [TIME:20630140.000000us]Resolution[960x720]Execution Time for one resolution[8535.072266]ms.
[5-B][11490]: [TIME:36094820.000000us]Completed Reading frame.
In order to see the syslog, type the following: cat /var/log/syslog | grep '5-B'
```

Fig. 5A-1 – Values of WCET for Contrast at 960X720.

Considering we need 3 transforms in 100ms, with 11.69ms execution time for each, we get to a number of 11.69*3 = 35.07ms. A good deadline would be 100ms. This would give enough margin for the execution of the transforms.

Ans-5B)

Design of the program:

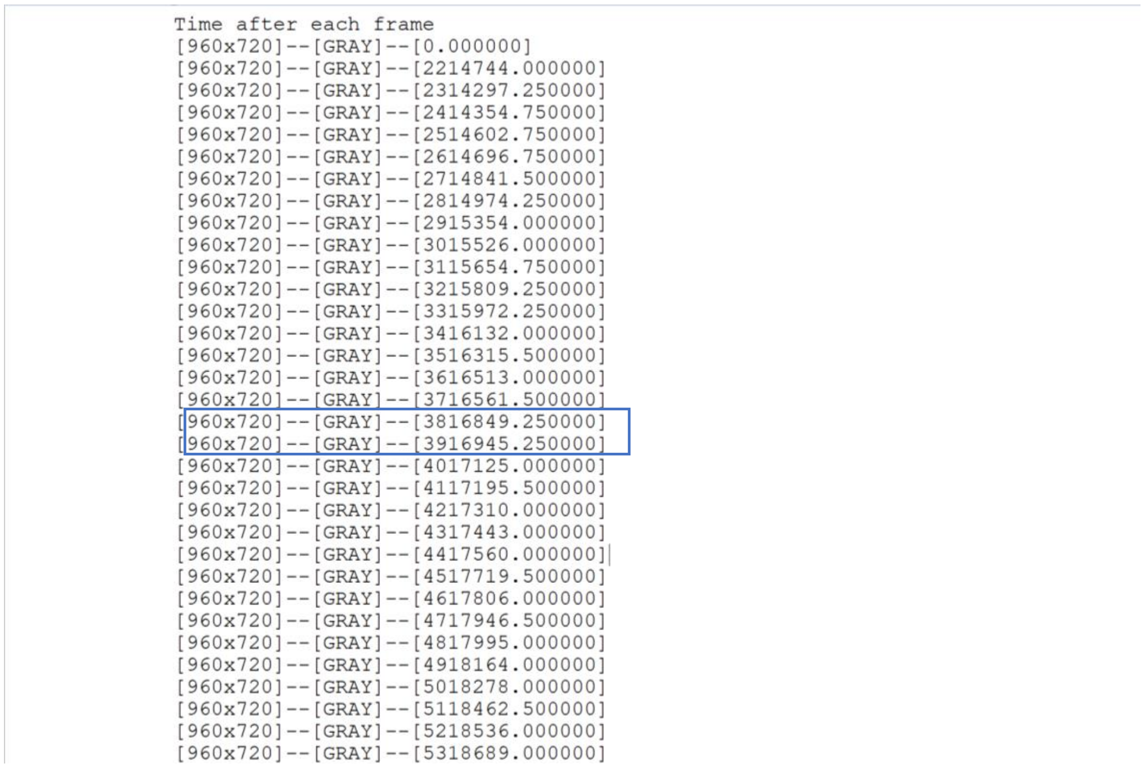
1. A scheduler running on CORE-0 of Nvidia Jetson. It implements RM for 3 tasks which run on CORE-3. – Highest priority.
2. Capture frame running on CORE-0 with second highest priority.
3. Based on time of execution, the priorities for the tasks were given as Grey with the smallest time and highest priority, Brightness with second highest priority and lastly contrast with lowest priority. – Priorities 3,4 and 5.
4. A deadline of 100ms to capture 3 frames. i.e. 1 from each transform at a set resolution.
5. The code runs for 50 frames for each type of transform.

Algorithm and working of the program:

1. Scheduler runs at the highest priority. It posts 3 semaphores. One for each conversion thread. Before it can release the semaphores it ensures once at the start of the program if the ReadFrame() thread has captured a frame or not. This check runs only once. Once semaphores are released, it enters a sleep of 100ms using the nanosleep() function. Once each of the 3 RM tasks has processed an image, they set a bit in TASK_BITMASK. In Scheduler after the 100ms sleep, a check for TASK_BITMASK is done. If it is complete with 3 bits set, the TotalFrameCount of 50 frames is reduced by 1. The semaphores are released again and the process continues till TotalFrameCounts does become 0. Once complete, the PROCESS_COMPLETE_BIT is set to 1 indicating the end of the processing followed by posting all the semaphores.
2. ReadFrame runs till PROCESS_COMPLETE_BIT is not set to 1. It sleeps for 20ms, hence providing a frame rate of 50fps. Once it awakens, it captures a new frame. Then it proceeds to sleep again.
3. Each of the 3 tasks run till PROCESS_COMPLETE_BIT!=1. Once in the task, they get blocked at semWait() if the sequencer has not woken from sleep and as a result not released the semaphores. Once they begin execution, each of the tasks perform their respective conversion and set the respective bit in TASK_BITMASK. At the same time, in each conversion, the time at each instant of the frame being converted is stored in an array.

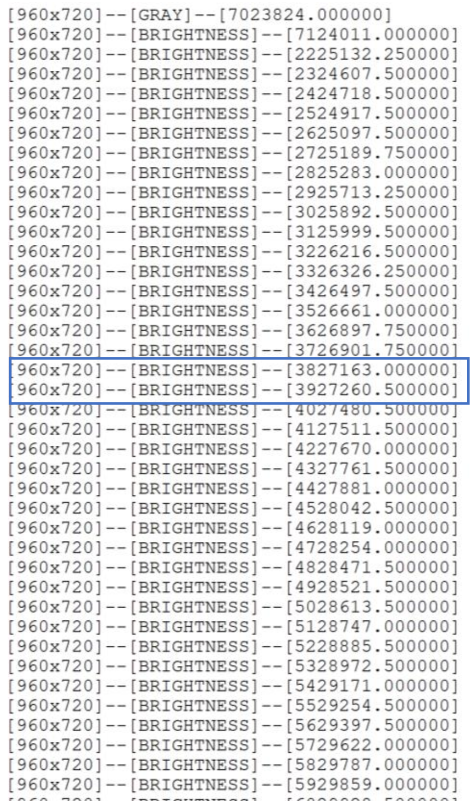
Prototype Analysis:

1. The first analysis involves determining that each conversion frame of the same type, occurs at a time difference of 100ms -- the set deadline. The image below is of the syslog of the system on Nvidia, in each of the frames in each transform, there is a time difference of 100ms. Indicating that the scheduler is running at the set frequency.



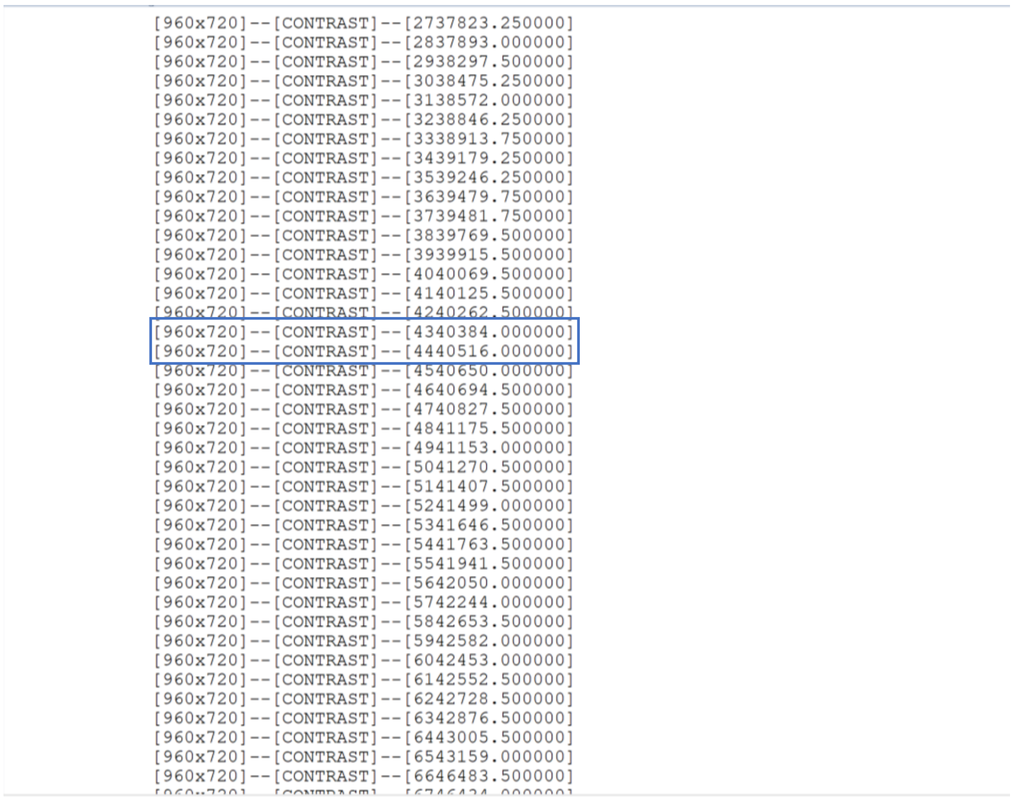
```
Time after each frame
[960x720]--[GRAY]--[0.000000]
[960x720]--[GRAY]--[2214744.000000]
[960x720]--[GRAY]--[2314297.250000]
[960x720]--[GRAY]--[2414354.750000]
[960x720]--[GRAY]--[2514602.750000]
[960x720]--[GRAY]--[2614696.750000]
[960x720]--[GRAY]--[2714841.500000]
[960x720]--[GRAY]--[2814974.250000]
[960x720]--[GRAY]--[2915354.000000]
[960x720]--[GRAY]--[3015526.000000]
[960x720]--[GRAY]--[3115654.750000]
[960x720]--[GRAY]--[3215809.250000]
[960x720]--[GRAY]--[3315972.250000]
[960x720]--[GRAY]--[3416132.000000]
[960x720]--[GRAY]--[3516315.500000]
[960x720]--[GRAY]--[3616513.000000]
[960x720]--[GRAY]--[3716561.500000]
[960x720]--[GRAY]--[3816849.250000]
[960x720]--[GRAY]--[3916945.250000]
[960x720]--[GRAY]--[4017125.000000]
[960x720]--[GRAY]--[4117195.500000]
[960x720]--[GRAY]--[4217310.000000]
[960x720]--[GRAY]--[4317443.000000]
[960x720]--[GRAY]--[4417560.000000]
[960x720]--[GRAY]--[4517719.500000]
[960x720]--[GRAY]--[4617806.000000]
[960x720]--[GRAY]--[4717946.500000]
[960x720]--[GRAY]--[4817995.000000]
[960x720]--[GRAY]--[4918164.000000]
[960x720]--[GRAY]--[5018278.000000]
[960x720]--[GRAY]--[5118462.500000]
[960x720]--[GRAY]--[5218536.000000]
[960x720]--[GRAY]--[5318689.000000]
```

Fig 5B-3: 50 Grey frame times at 960x720



```
[960x720]--[GRAY]--[7023824.000000]
[960x720]--[BRIGHTNESS]--[7124011.000000]
[960x720]--[BRIGHTNESS]--[2225132.250000]
[960x720]--[BRIGHTNESS]--[2324607.500000]
[960x720]--[BRIGHTNESS]--[2424718.500000]
[960x720]--[BRIGHTNESS]--[2524917.500000]
[960x720]--[BRIGHTNESS]--[2625097.500000]
[960x720]--[BRIGHTNESS]--[2725189.750000]
[960x720]--[BRIGHTNESS]--[2825283.000000]
[960x720]--[BRIGHTNESS]--[2925713.250000]
[960x720]--[BRIGHTNESS]--[3025892.500000]
[960x720]--[BRIGHTNESS]--[3125999.500000]
[960x720]--[BRIGHTNESS]--[3226216.500000]
[960x720]--[BRIGHTNESS]--[3326326.250000]
[960x720]--[BRIGHTNESS]--[3426497.500000]
[960x720]--[BRIGHTNESS]--[3526661.000000]
[960x720]--[BRIGHTNESS]--[3626897.750000]
[960x720]--[BRIGHTNESS]--[3726901.750000]
[960x720]--[BRIGHTNESS]--[3827163.000000]
[960x720]--[BRIGHTNESS]--[3927260.500000]
[960x720]--[BRIGHTNESS]--[4027480.500000]
[960x720]--[BRIGHTNESS]--[4127511.500000]
[960x720]--[BRIGHTNESS]--[4227670.000000]
[960x720]--[BRIGHTNESS]--[4327761.500000]
[960x720]--[BRIGHTNESS]--[4427881.000000]
[960x720]--[BRIGHTNESS]--[4528042.500000]
[960x720]--[BRIGHTNESS]--[4628119.000000]
[960x720]--[BRIGHTNESS]--[4728254.000000]
[960x720]--[BRIGHTNESS]--[4828471.500000]
[960x720]--[BRIGHTNESS]--[4928521.500000]
[960x720]--[BRIGHTNESS]--[5028613.500000]
[960x720]--[BRIGHTNESS]--[5128747.000000]
[960x720]--[BRIGHTNESS]--[5228885.500000]
[960x720]--[BRIGHTNESS]--[5328972.500000]
[960x720]--[BRIGHTNESS]--[5429171.000000]
[960x720]--[BRIGHTNESS]--[5529254.500000]
[960x720]--[BRIGHTNESS]--[5629397.500000]
[960x720]--[BRIGHTNESS]--[5729622.000000]
[960x720]--[BRIGHTNESS]--[5829787.000000]
[960x720]--[BRIGHTNESS]--[5929859.000000]
```

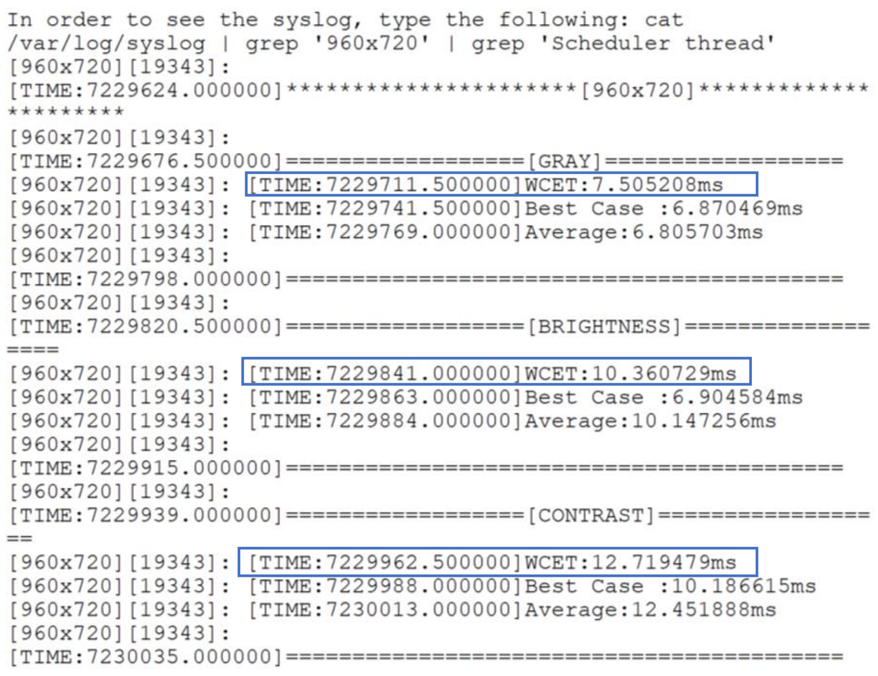
Fig 5B-3: 50 Bright frame times at 960x720



```
[960x720]--[CONTRAST]--[2737823.250000]
[960x720]--[CONTRAST]--[2837893.000000]
[960x720]--[CONTRAST]--[2938297.500000]
[960x720]--[CONTRAST]--[3038475.250000]
[960x720]--[CONTRAST]--[3138572.000000]
[960x720]--[CONTRAST]--[3238846.250000]
[960x720]--[CONTRAST]--[3338913.750000]
[960x720]--[CONTRAST]--[3439179.250000]
[960x720]--[CONTRAST]--[3539246.250000]
[960x720]--[CONTRAST]--[3639479.750000]
[960x720]--[CONTRAST]--[3739481.750000]
[960x720]--[CONTRAST]--[3839769.500000]
[960x720]--[CONTRAST]--[3939915.500000]
[960x720]--[CONTRAST]--[4040069.500000]
[960x720]--[CONTRAST]--[4140125.500000]
[960x720]--[CONTRAST]--[4240262.500000]
[960x720]--[CONTRAST]--[4340384.000000]
[960x720]--[CONTRAST]--[4440516.000000]
[960x720]--[CONTRAST]--[4540650.000000]
[960x720]--[CONTRAST]--[4640694.500000]
[960x720]--[CONTRAST]--[4740827.500000]
[960x720]--[CONTRAST]--[4841175.500000]
[960x720]--[CONTRAST]--[4941153.000000]
[960x720]--[CONTRAST]--[5041270.500000]
[960x720]--[CONTRAST]--[5141407.500000]
[960x720]--[CONTRAST]--[5241499.000000]
[960x720]--[CONTRAST]--[5341646.500000]
[960x720]--[CONTRAST]--[5441763.500000]
[960x720]--[CONTRAST]--[5541941.500000]
[960x720]--[CONTRAST]--[5642050.000000]
[960x720]--[CONTRAST]--[5742244.000000]
[960x720]--[CONTRAST]--[5842653.500000]
[960x720]--[CONTRAST]--[5942582.000000]
[960x720]--[CONTRAST]--[6042453.000000]
[960x720]--[CONTRAST]--[6142552.500000]
[960x720]--[CONTRAST]--[6242728.500000]
[960x720]--[CONTRAST]--[6342876.500000]
[960x720]--[CONTRAST]--[6443005.500000]
[960x720]--[CONTRAST]--[6543159.000000]
[960x720]--[CONTRAST]--[6646483.500000]
```

Fig 5B-3: 50 Contrast frame times at 960x720

2. To determine that the tasks completed well before deadline we can take the average and worst case execution times of each transform and subtract that value from the 100ms deadline. It can be seen from the image below that, this value is roughly, 69.595ms. This means that 82ms are still available with the scheduler after execution of each of the transforms. This provides a margin for coping with errors which could build over a period, or large frame numbers like in 1000s.

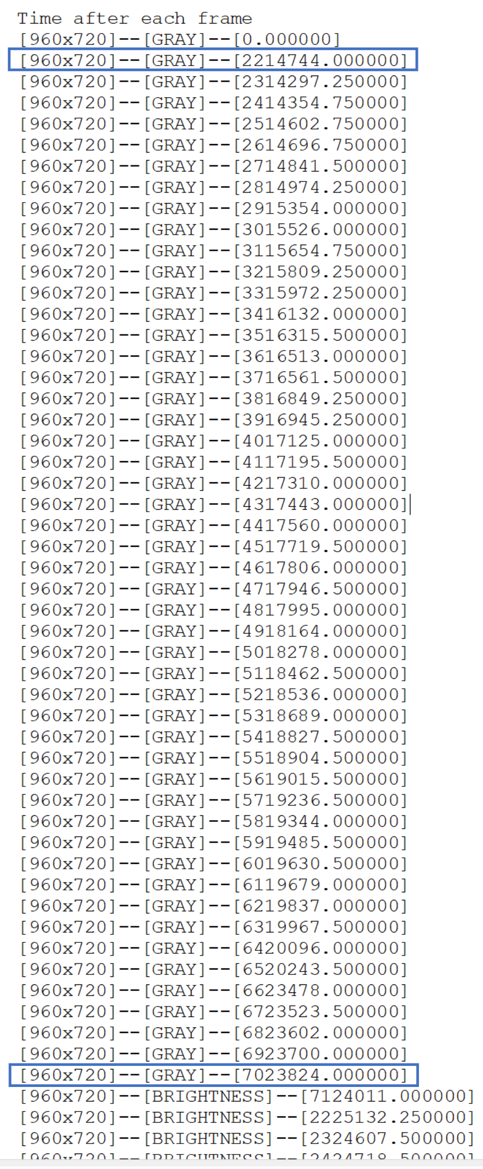


```
In order to see the syslog, type the following: cat
/var/log/syslog | grep '960x720' | grep 'Scheduler thread'
[960x720][19343]:
[TIME:7229624.000000]**********************[960x720]*************
*********
[960x720][19343]:
[TIME:7229676.500000]==================[GRAY]==================
[960x720][19343]: [TIME:7229711.500000]WCET:7.505208ms
[960x720][19343]: [TIME:7229741.500000]Best Case :6.870469ms
[960x720][19343]: [TIME:7229769.000000]Average:6.805703ms
[960x720][19343]:
[TIME:7229798.000000]==========================================
[960x720][19343]:
[TIME:7229820.500000]==================[BRIGHTNESS]==============
====
[960x720][19343]: [TIME:7229841.000000]WCET:10.360729ms
[960x720][19343]: [TIME:7229863.000000]Best Case :6.904584ms
[960x720][19343]: [TIME:7229884.000000]Average:10.147256ms
[960x720][19343]:
[TIME:7229915.000000]==========================================
[960x720][19343]:
[TIME:7229939.000000]==================[CONTRAST]================
==
[960x720][19343]: [TIME:7229962.500000]WCET:12.719479ms
[960x720][19343]: [TIME:7229988.000000]Best Case :10.186615ms
[960x720][19343]: [TIME:7230013.000000]Average:12.451888ms
[960x720][19343]:
[TIME:7230035.000000]==========================================
```

Fig 5B-4: WCET,BCET and Average times at 960x720

Time Left = Deadline – (Sum of WCET) = 100 – (7.5+10.36+12.719)

3. For the transformation of 50 frames, with a deadline of 100ms for a group of 3 frames, the time taken comes out to be approximately 50*100ms = 5s. It can be noted from the logs below that the time difference between first and last frame for a transform is approximately 4.8s. This difference is mainly due to the clock skew in the system.



```
Time after each frame
[960x720]--[GRAY]--[0.000000]
[960x720]--[GRAY]--[2214744.000000]
[960x720]--[GRAY]--[2314297.250000]
[960x720]--[GRAY]--[2414354.750000]
[960x720]--[GRAY]--[2514602.750000]
[960x720]--[GRAY]--[2614696.750000]
[960x720]--[GRAY]--[2714841.500000]
[960x720]--[GRAY]--[2814974.250000]
[960x720]--[GRAY]--[2915354.000000]
[960x720]--[GRAY]--[3015526.000000]
[960x720]--[GRAY]--[3115654.750000]
[960x720]--[GRAY]--[3215809.250000]
[960x720]--[GRAY]--[3315972.250000]
[960x720]--[GRAY]--[3416132.000000]
[960x720]--[GRAY]--[3516315.500000]
[960x720]--[GRAY]--[3616513.000000]
[960x720]--[GRAY]--[3716561.500000]
[960x720]--[GRAY]--[3816849.250000]
[960x720]--[GRAY]--[3916945.250000]
[960x720]--[GRAY]--[4017125.000000]
[960x720]--[GRAY]--[4117195.500000]
[960x720]--[GRAY]--[4217310.000000]
[960x720]--[GRAY]--[4317443.000000]
[960x720]--[GRAY]--[4417560.000000]
[960x720]--[GRAY]--[4517719.500000]
[960x720]--[GRAY]--[4617806.000000]
[960x720]--[GRAY]--[4717946.500000]
[960x720]--[GRAY]--[4817995.000000]
[960x720]--[GRAY]--[4918164.000000]
[960x720]--[GRAY]--[5018278.000000]
[960x720]--[GRAY]--[5118462.500000]
[960x720]--[GRAY]--[5218536.000000]
[960x720]--[GRAY]--[5318689.000000]
[960x720]--[GRAY]--[5418827.500000]
[960x720]--[GRAY]--[5518904.500000]
[960x720]--[GRAY]--[5619015.500000]
[960x720]--[GRAY]--[5719236.500000]
[960x720]--[GRAY]--[5819344.000000]
[960x720]--[GRAY]--[5919485.500000]
[960x720]--[GRAY]--[6019630.500000]
[960x720]--[GRAY]--[6119679.000000]
[960x720]--[GRAY]--[6219837.000000]
[960x720]--[GRAY]--[6319967.500000]
[960x720]--[GRAY]--[6420096.000000]
[960x720]--[GRAY]--[6520243.500000]
[960x720]--[GRAY]--[6623478.000000]
[960x720]--[GRAY]--[6723523.500000]
[960x720]--[GRAY]--[6823602.000000]
[960x720]--[GRAY]--[6923700.000000]
[960x720]--[GRAY]--[7023824.000000]
[960x720]--[BRIGHTNESS]--[7124011.000000]
[960x720]--[BRIGHTNESS]--[2225132.250000]
[960x720]--[BRIGHTNESS]--[2324607.500000]
[960x720]--[BRIGHTNESS]--[2424718.500000]
```

Fig 5B-5: Difference between start and end times at 960x720

Clock Skew and Jitter analysis:

In the images below as well as the syslog that, between the first and the last image of the same transform, there is a difference of about 4.85 – 5s. While this value is expected to be exactly 5s, it is possible that such an issue may be caused by clock skew. This difference in longer processing times could be become problematic.

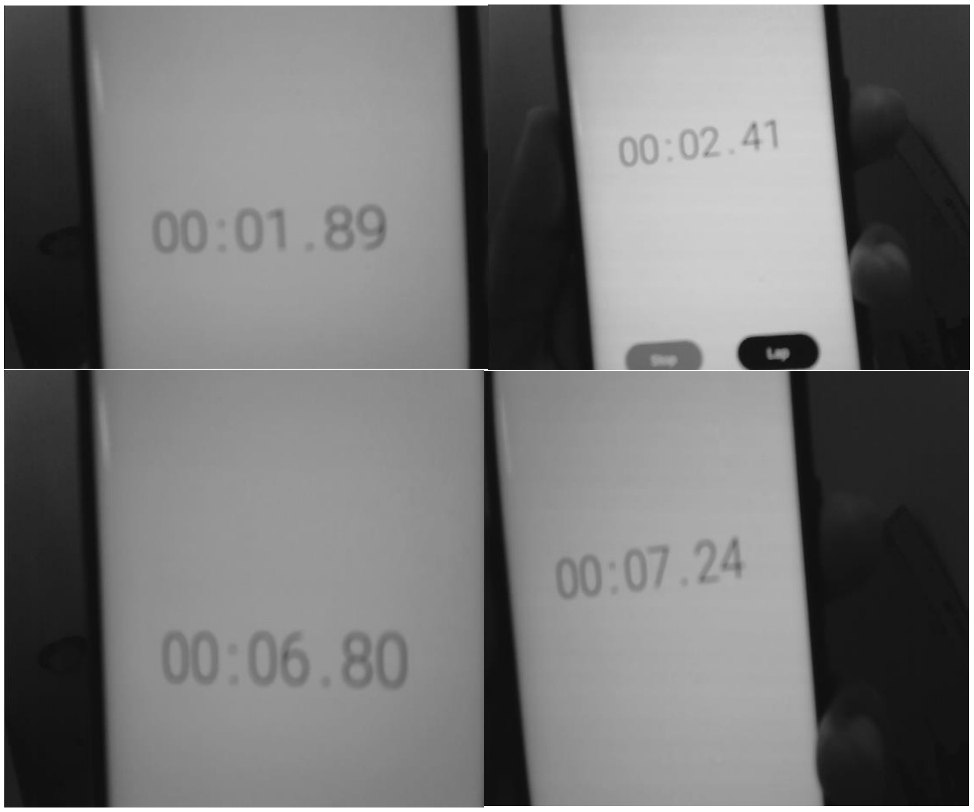


Fig 5B-5: Difference between start and end times at 960x720 and 800x600.