EXERCISE #4 – REAL-TIME CONTINUOUS MEDIA

REAL TIME EMBEDDED SYSTEMS (ECEN 5623)

SUMMER 2019

BOARD USED:

NVIDIA's JETSON NANO DEVELOPMENT BOARD

REPORT BY:

OM RAHEJA

PROFESSOR:

DR. SAM SIEWERT

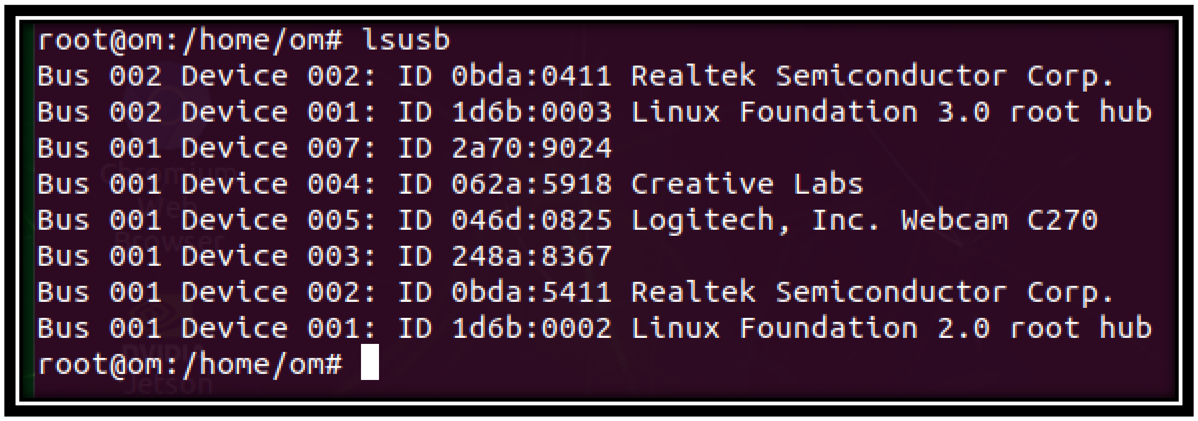
1. Obtain a Logitech C270 camera (or equivalent) and verify that is detected by the Jetson board USB driver. Use lsusb, lsmod and dmesg kernel driver configuration tool to make sure your Logitech C200 USB camera is plugged in and recognized by your Jetson. Do lsusb | grep C200 and prove to me (and more 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.

Ans: <u>Camera and device driver verification:</u>

<u>USB hot-plug:</u>

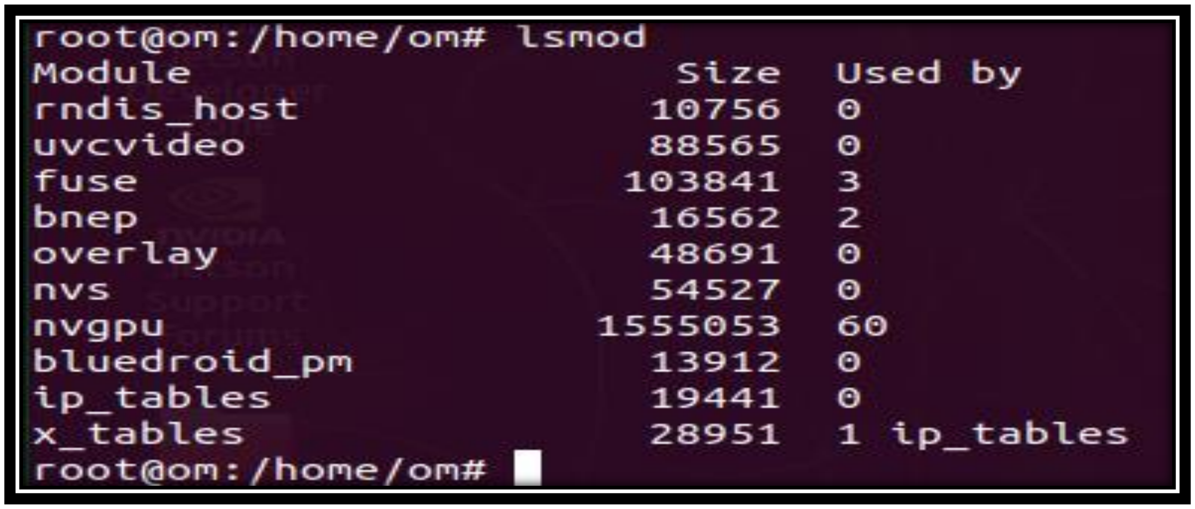
The following three commands have been run to confirm that the Logitech C270 webcam is recognized by Jetson Nano.

- ***lsusb***
  Logitech C270 Webcam is detected successfully.



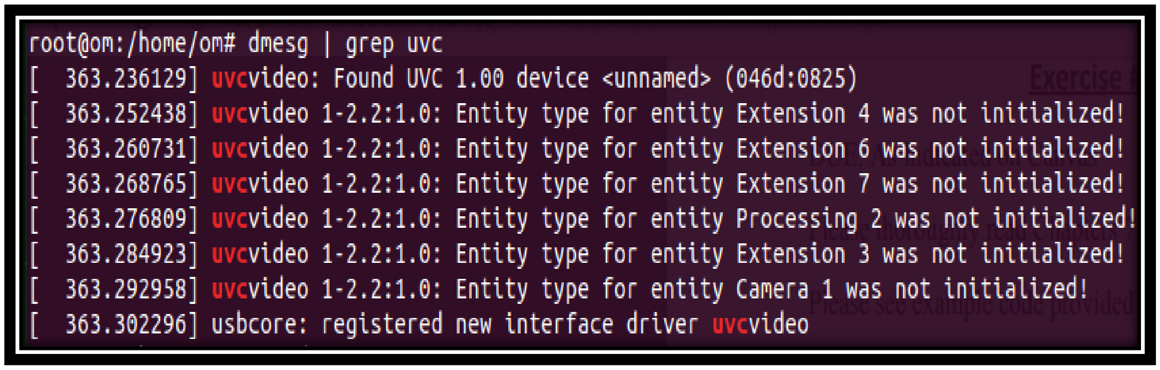
```
root@om:/home/om# lsusb
Bus 002 Device 002: ID 0bda:0411 Realtek Semiconductor Corp.
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 007: ID 2a70:9024
Bus 001 Device 004: ID 062a:5918 Creative Labs
Bus 001 Device 005: ID 046d:0825 Logitech, Inc. Webcam C270
Bus 001 Device 003: ID 248a:8367
Bus 001 Device 002: ID 0bda:5411 Realtek Semiconductor Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@om:/home/om#
```

- ***lsmod***



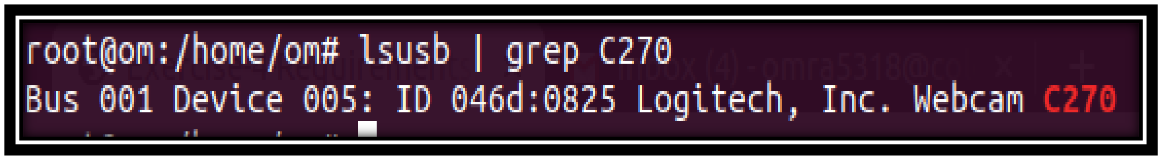
```
root@om:/home/om# lsmod
Module                  Size  Used by
rndis_host             10756  0
uvcvideo               88565  0
fuse                  103841  3
bnep                   16562  2
overlay                48691  0
nvs                    54527  0
nvgpu                1555053  60
bluedroid_pm           13912  0
ip_tables              19441  0
x_tables               28951  1 ip_tables
root@om:/home/om#
```

- ***dmesg***



```
root@om:/home/om# dmesg | grep uvc
[  363.236129] uvcvideo: Found UVC 1.00 device <unnamed> (046d:0825)
[  363.252438] uvcvideo 1-2.2:1.0: Entity type for entity Extension 4 was not initialized!
[  363.260731] uvcvideo 1-2.2:1.0: Entity type for entity Extension 6 was not initialized!
[  363.268765] uvcvideo 1-2.2:1.0: Entity type for entity Extension 7 was not initialized!
[  363.276809] uvcvideo 1-2.2:1.0: Entity type for entity Processing 2 was not initialized!
[  363.284923] uvcvideo 1-2.2:1.0: Entity type for entity Extension 3 was not initialized!
[  363.292958] uvcvideo 1-2.2:1.0: Entity type for entity Camera 1 was not initialized!
[  363.302296] usbcore: registered new interface driver uvcvideo
```

- ***lsusb | grep C270***

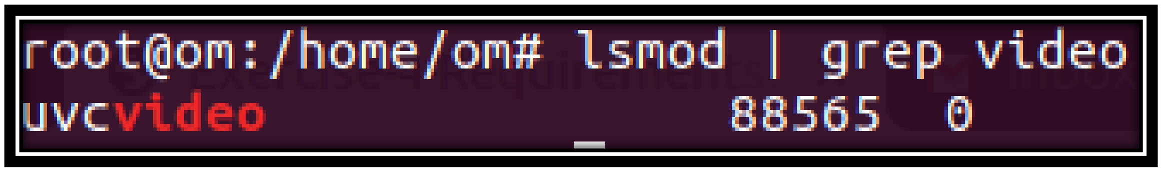


```
root@om:/home/om# lsusb | grep C270
Bus 001 Device 005: ID 046d:0825 Logitech, Inc. Webcam C270
```

<u>UVC driver verify:</u>

The following command was run to verify that the UVC driver was loaded successfully.

- ***lsmod | grep video***

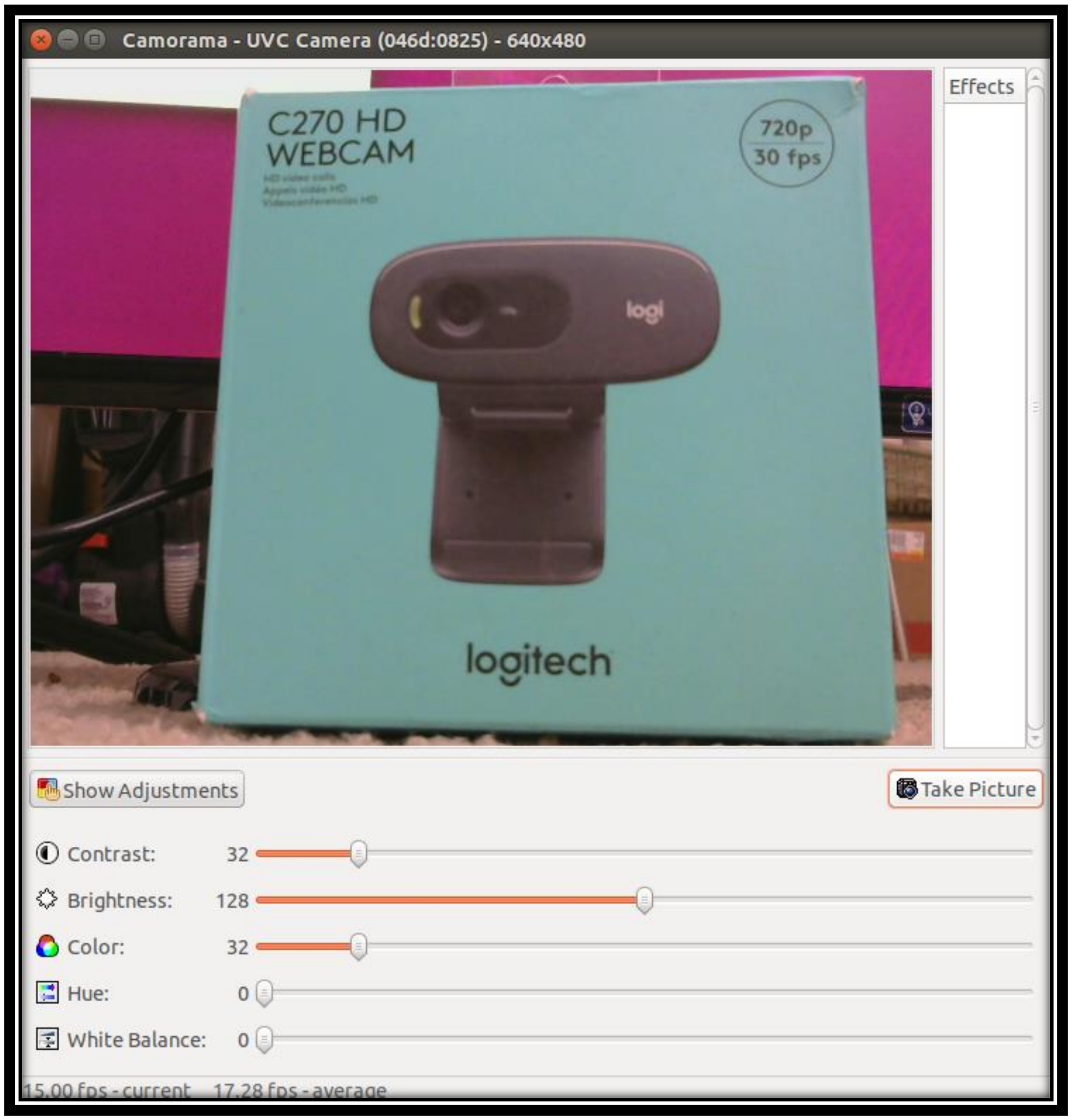


```
root@om:/home/om# lsmod | grep video
uvcvideo               88565  0
```

2. 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.

Ans: <u>Camera streaming checkout:</u>

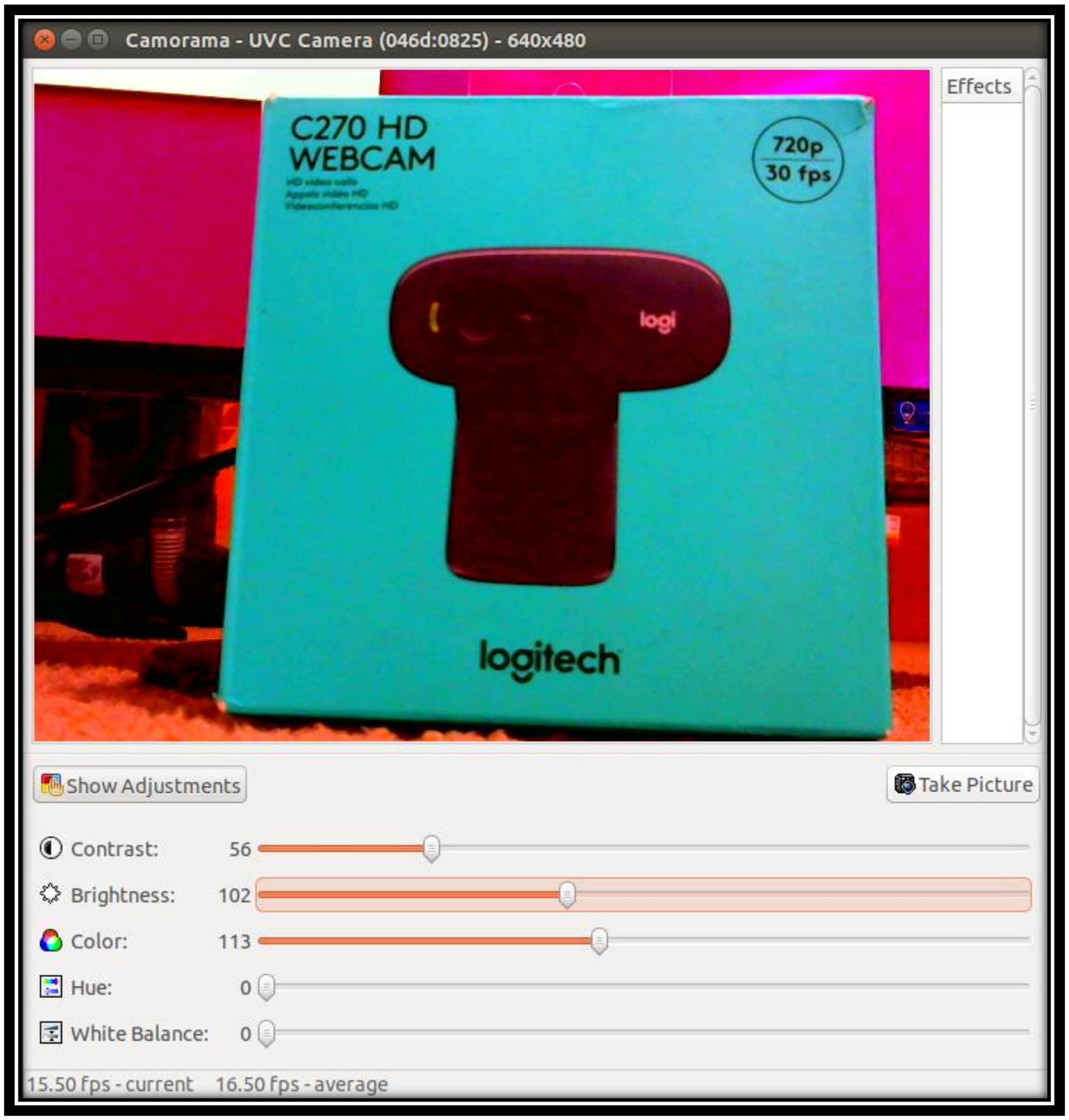
Use of tools such as camorama & Verification:



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

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

15.00 fps - current 17.28 fps - average


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

Contrast: 56
Brightness: 102
Color: 113
Hue: 0
White Balance: 0

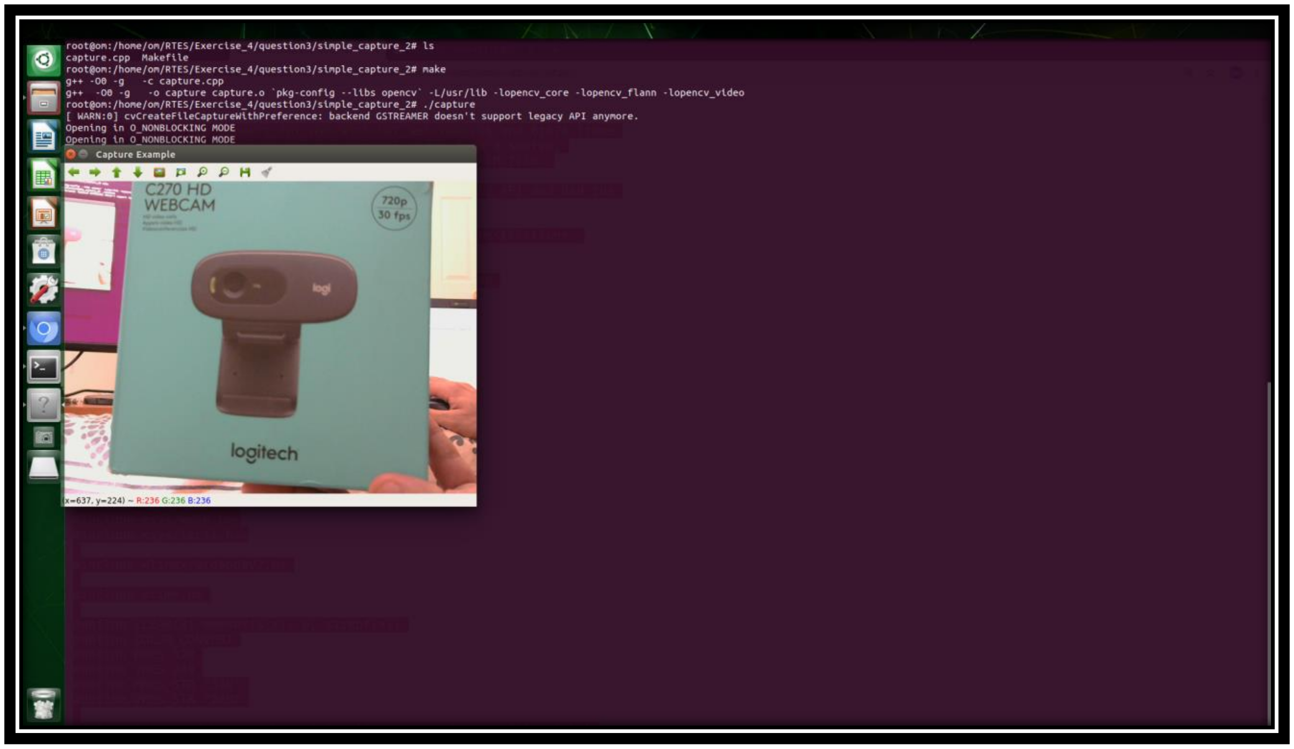
15.50 fps - current 16.50 fps - average

3. 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

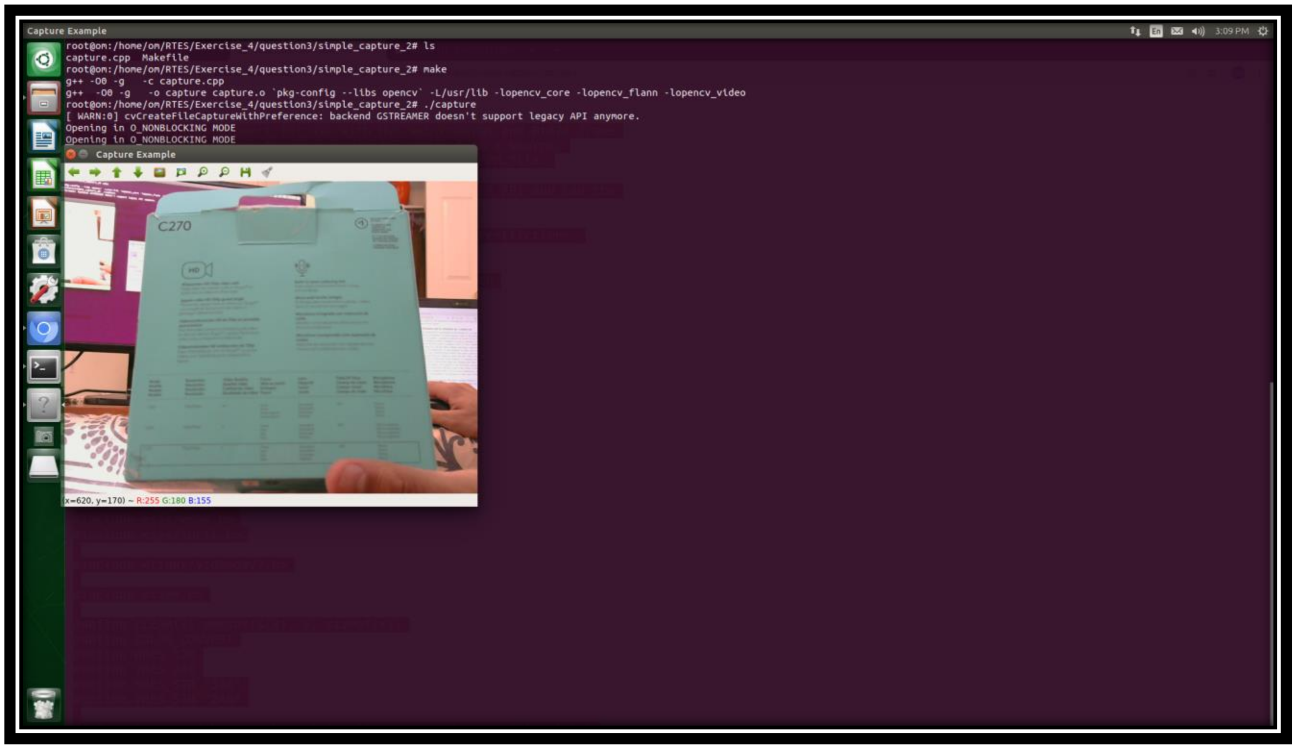
have a consistent CUDA install for your R19 or R21 board, so if in doubt, don't install CUDA and leave it off when you build OpenCV].

Ans: <u>Camera continuous streaming tests and applications:</u>

<u>Build and test:</u>

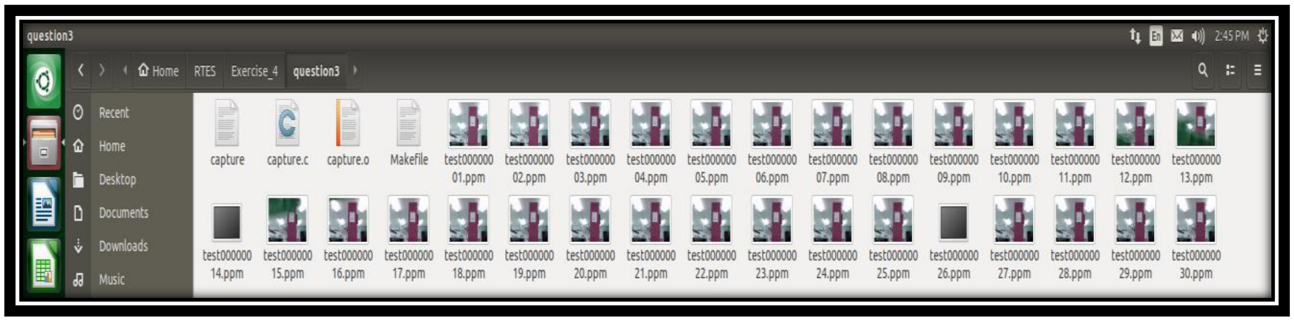


```
root@om:/home/om/RTES/Exercise_4/question3/simple_capture_2# ls
capture.cpp  Makefile
root@om:/home/om/RTES/Exercise_4/question3/simple_capture_2# make
g++ -O0 -g   -c capture.cpp
g++  -O0 -g   -o capture capture.o `pkg-config --libs opencv` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video
root@om:/home/om/RTES/Exercise_4/question3/simple_capture_2# ./capture
[ WARN:0] cvCreateFileCaptureWithPreference: backend GSTREAMER doesn't support legacy API anymore.
Opening in O_NONBLOCKING MODE
Opening in O_NONBLOCKING MODE
```



```
root@om:/home/om/RTES/Exercise_4/question3/simple_capture_2# ls
capture.cpp  Makefile
root@om:/home/om/RTES/Exercise_4/question3/simple_capture_2# make
g++ -O0 -g   -c capture.cpp
g++  -O0 -g   -o capture capture.o `pkg-config --libs opencv` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video
root@om:/home/om/RTES/Exercise_4/question3/simple_capture_2# ./capture
[ WARN:0] cvCreateFileCaptureWithPreference: backend GSTREAMER doesn't support legacy API anymore.
Opening in O_NONBLOCKING MODE
Opening in O_NONBLOCKING MODE
```

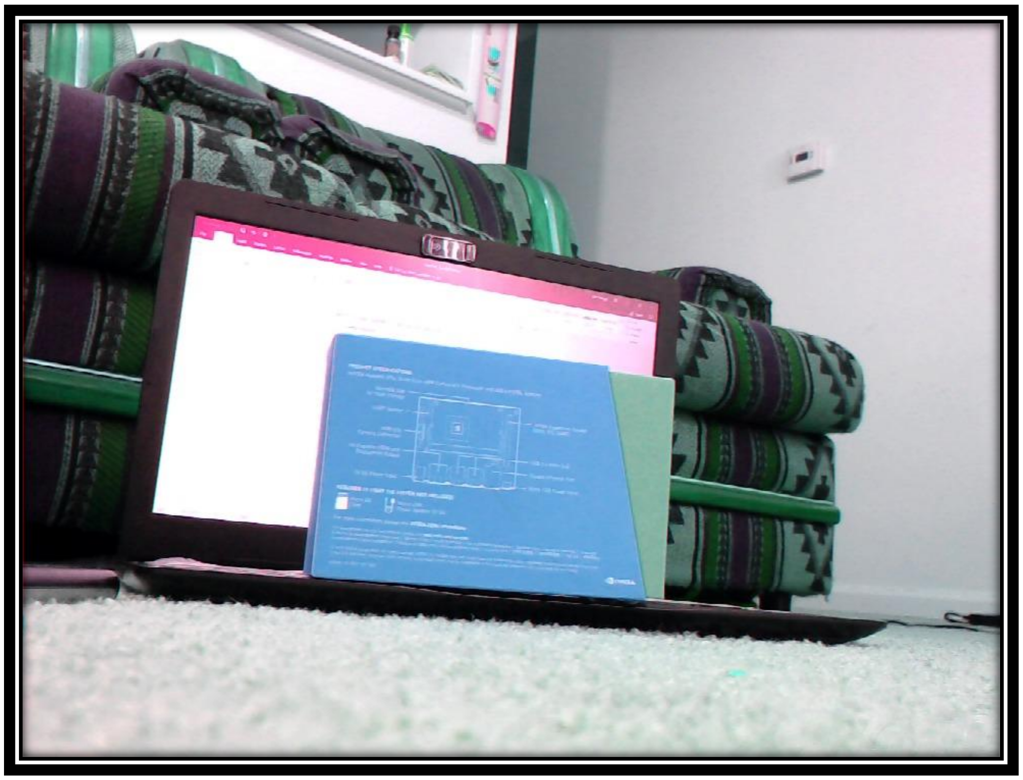
Streaming Verification:



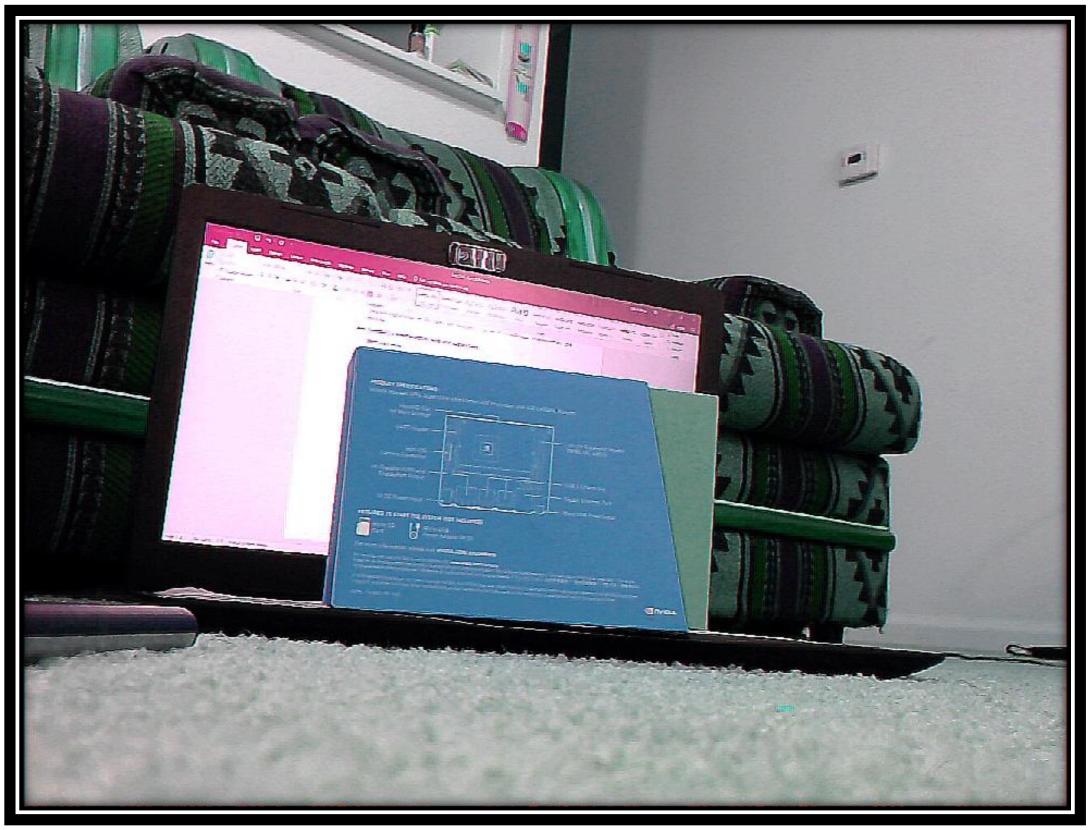
4. Choose ONE continuous transformation for each acquired frame such as color conversion (Lecture-Cont-Media.pdf, slide 5), conversion to grayscale, or sharpening (/sharpen-psf/sharpen.c) if you are using simple-capture and the V4L2 API. If you want to use OpenCV (not required but can be fun and interesting) look at examples from computer-vision then you can choose transformations such as the canny-interactive, Hough interactive, Hough-elliptical-interactive, or stereo-transform-improved. 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.

Ans: Continuous transformation tests and applications:

Demonstration:



Original Image



Sharpened Image

Description:

The original image was obtained as a result of execution of the simple-capture code. The sharpened image was obtained by using simple capture code which has the sharpening algorithm embedded into it. The program makes use of the V4L2 (Video4Linux) APIs.

a) **open_device()**
The open device function checks the status of the camera module connected. It also checks if the device file type is character special or block. After checking these, this function opens the device and checks for its return status. If the file descriptor returns a -1, then it means that the device was not opened successfully.

b) **init_device()**
The first call to the xioctl function is to check if the device that we are trying to open is V4L2 compatible. Then, it is checked it the V4L2 device connected has the capability to capture videos. After that, it is checked that whether the connected device has video streaming capabilities. One more call to xioctl function is made to check if the device has cropping and scaling capabilities. Here, the cropping rectangle is set to VIDIOC_S_CROP and the data stream is set to V4L2_BUF_TYPE_VIDEO_CAPTURE. The format of the frame is set, where the height and width of the frame depends upon the vertical and horizontal resolution we set in the program. The pixel coding format is set to YUYV. Lastly, a function call to init_mmap() is made.

In the init_mmap() function, mmap buffers are allocated which are located in the device memory. Blocks of memory are allocated using calloc function call.

c) **Start_capturing()**
The empty buffers are queued to collect the device data. VIDIOC_STREAMON is used to start the streaming I/O.

d) **Mainloop()**
This function calls the read_frame function which reads the frame from the device. This image/frame read is passed to a process_image function which converts the yuyv image into a rgbrgb image format by calling the yuv2rgb() function.
By altering the frame_count parameter, we can get the desired number of frames for analysis.

e) **Sharpen_image()**
This function is called to convert the RGB image into a sharper version of the same image. The algorithm for this function was referenced from Professor Sam Siewert's code (sharpen.c). Link for the sharpening algorithm is provided in the reference section.

f) **Stop_capturing()**
The I/O streaming is stopped by making use of VIDIOC_STREAMOFF.

g) **Uninit_device()**
The buffers allocated for the device are unmapped in this function.

h) **Close_device()**
The device is closed using the appropriate file descriptor.

5. 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: <u>Soft Real-time analysis and conversion to SCHED_FIFO with predictability analysis:</u>

<u>Design concepts:</u>

This section of the report describes the general flow of my code. For this question, I have used the Linux V4L2 driver code as the base and used three different transformations (sharpening, black & white and gray scaling) to accomplish the requirements.

The code has three threads, but only one will be created at any given time, based on the command line arguments entered during run-time. The command line takes the following arguments:

a) Executable name (./Question5)
b) /dev/video0
c) Frame count
d) Type of transformation (sharpen, Bnw or grayscale)
e) Horizontal resolution (HRES)
f) Vertical resolution (VRES)

In order to run the code with appropriate arguments, the instructions are displayed after typing a "make" in the command line window. Once appropriate arguments are passed in the command line, the thread associated with the transformation entered gets created. The thread is created with SCHED_FIFO scheduling policy so that it does not get preempted by any tasks running in the background of Linux. Once the thread is created, the appropriate transform is applied to all the frames. The number of frames to be analyzed can be entered during run-time. Jitter is calculated for each frame along with the time elapsed for the entire analysis of frames.

The code runs in two cycles. The first cycle is to calculate the deadline. Once, the deadline is calculated, the second cycle runs to calculate the jitter and check if any of the frames miss their deadline or not. After these two cycles are run, the thread exits and joins thereby providing the image frames in the appropriate form. The time taken per frame to capture and get transformed is tracked in a .csv file along with positive, negative and total jitter. A graph has been plotted for the number of frames vs (positive jitter and negative jitter).

Algorithm Analysis:

The code was analyzed for a total of 100 frames. This data can be made more accurate by running the program with a greater number of frames. This way, even the deadline calculated can be more accurate and result in a more reliable system.

Frame Rate:

Frame rates for all the three continuous transformations have been analyzed for five supportable resolutions.

Type of Transformation	Resolution				
	960x720	**800x600**	**640x480**	**320x240**	**160x120**
Sharpening	3.401255 Hz	4.801998 Hz	6.891533 Hz	19.582207 Hz	19.732920 Hz
Black and White	8.819565 Hz	14.606260 Hz	19.705780 Hz	19.616559 Hz	19.760839 Hz
Gray Scaling	8.327848 Hz	14.624923 Hz	19.685980 Hz	19.822063 Hz	19.824121 Hz

It can be observed that as the resolution decreases, the frame rate increases.

Time Elapsed in Execution [This is calculated for 1st cycle only]:

Time taken by all the three continuous transformations have been analyzed for five supportable resolutions.

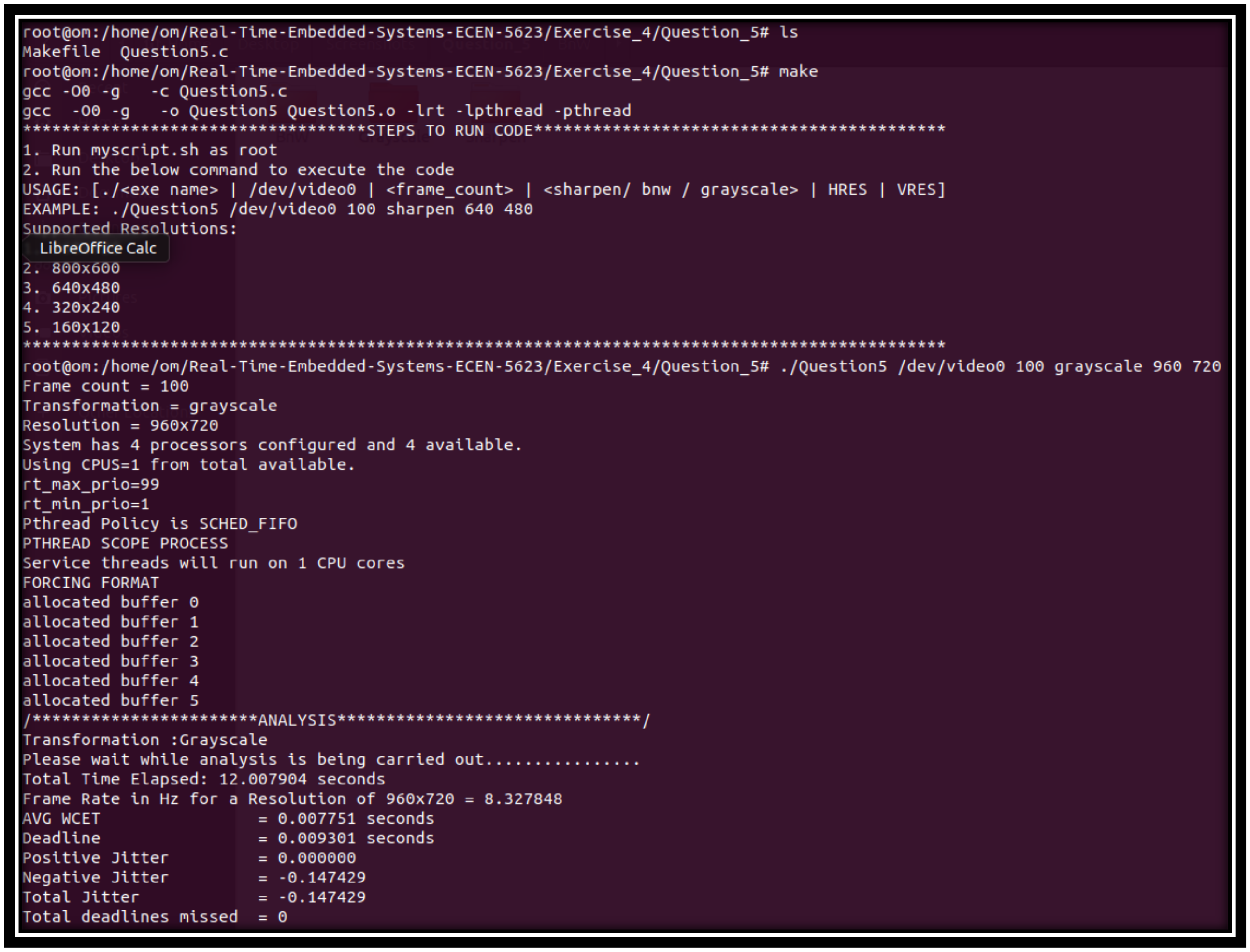
Type of Transformation	Resolution				
	960x720	**800x600**	**640x480**	**320x240**	**160x120**
Sharpening	30 secs	20.82 secs	14.51 secs	5.10 secs	5.06 secs
Black and White	12.21secs	6.84 secs	5.07 secs	5.09 secs	5.06 secs
Gray Scaling	12.07 secs	6.83 secs	5.07 secs	5.04 secs	5.04 secs

As the resolution goes on decreasing, the time taken for each transformation reduces.

<u>Prototype Analysis:</u>

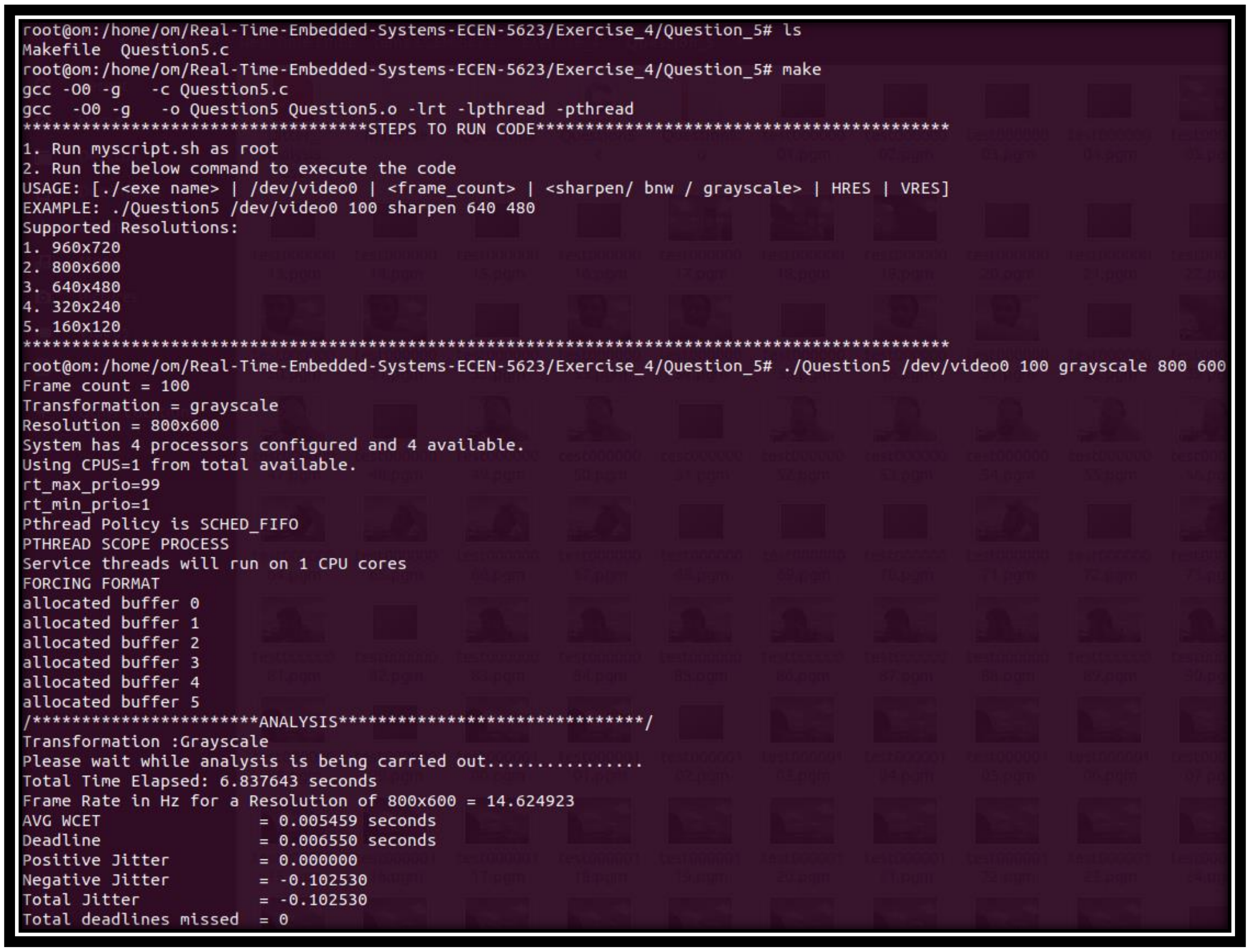
By executing the codes with different combinations of resolution and transformations the following data was collected for grayscale transformation with various resolutions. For convenience, only grayscale transformation analysis has been shown in the report. However, screenshots of analysis for Black and White transformation and Sharpening transformation has been attached in the Exercise_4 screenshots folder on my GitHub (https://github.com/omraheja/Real-Time-Embedded-Systems-ECEN-5623/tree/master/Exercise_4).

<u>Analysis for 960x720 resolution:</u>



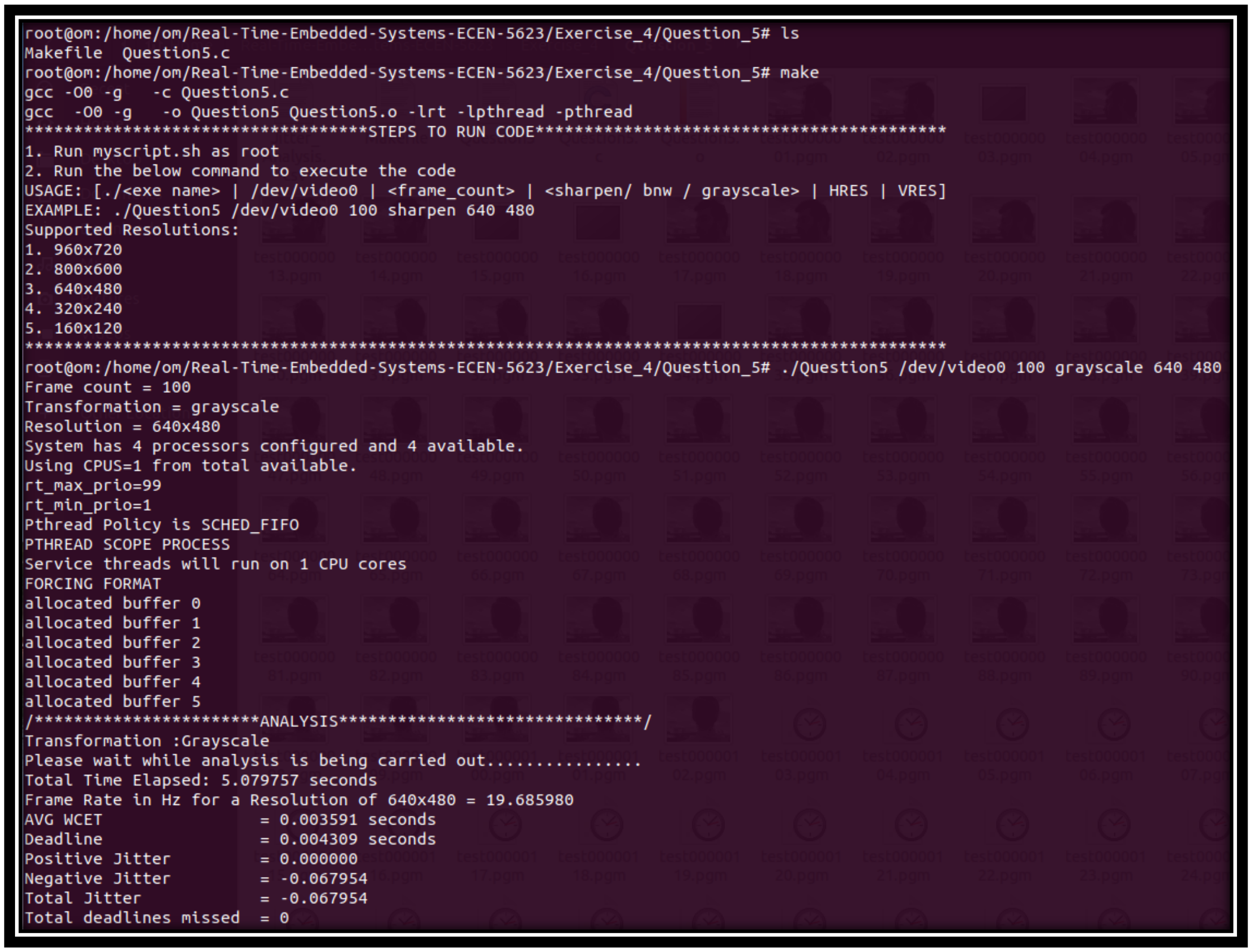
```
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ls
Makefile  Question5.c
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# make
gcc -O0 -g   -c Question5.c
gcc  -O0 -g   -o Question5 Question5.o -lrt -lpthread -pthread
***********************************STEPS TO RUN CODE******************************************
1. Run myscript.sh as root
2. Run the below command to execute the code
USAGE: [./<exe name> | /dev/video0 | <frame_count> | <sharpen/ bnw / grayscale> | HRES | VRES]
EXAMPLE: ./Question5 /dev/video0 100 sharpen 640 480
Supported Resolutions:
  LibreOffice Calc
2. 800x600
3. 640x480
4. 320x240
5. 160x120
**********************************************************************************************
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ./Question5 /dev/video0 100 grayscale 960 720
Frame count = 100
Transformation = grayscale
Resolution = 960x720
System has 4 processors configured and 4 available.
Using CPUS=1 from total available.
rt_max_prio=99
rt_min_prio=1
Pthread Policy is SCHED_FIFO
PTHREAD SCOPE PROCESS
Service threads will run on 1 CPU cores
FORCING FORMAT
allocated buffer 0
allocated buffer 1
allocated buffer 2
allocated buffer 3
allocated buffer 4
allocated buffer 5
/**********************ANALYSIS*******************************/
Transformation :Grayscale
Please wait while analysis is being carried out...............
Total Time Elapsed: 12.007904 seconds
Frame Rate in Hz for a Resolution of 960x720 = 8.327848
AVG WCET                = 0.007751 seconds
Deadline                = 0.009301 seconds
Positive Jitter         = 0.000000
Negative Jitter         = -0.147429
Total Jitter            = -0.147429
Total deadlines missed  = 0
```

<u>Analysis for 800x600 resolution:</u>



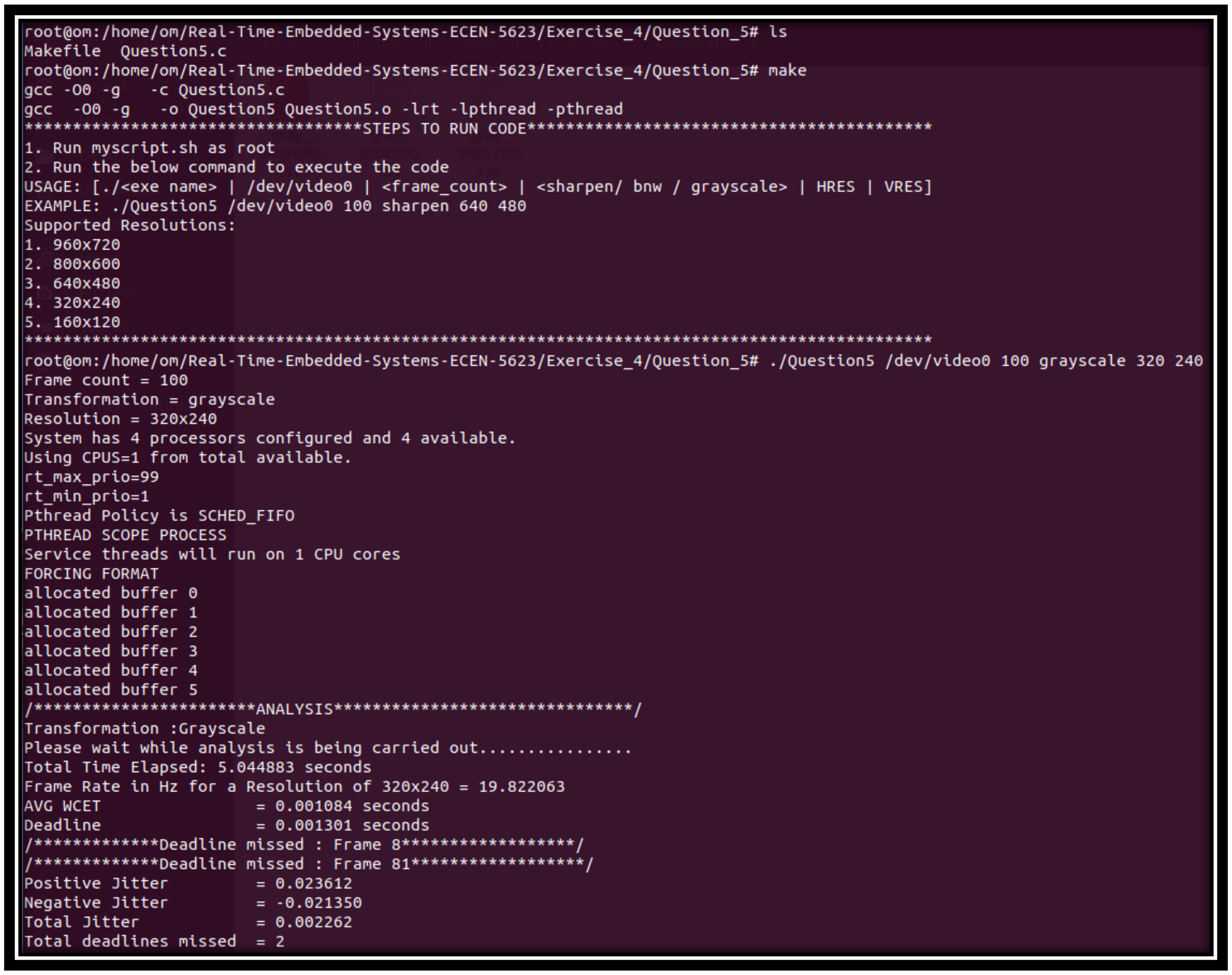
```
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ls
Makefile  Question5.c
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# make
gcc -O0 -g   -c Question5.c
gcc  -O0 -g   -o Question5 Question5.o -lrt -lpthread -pthread
**********************************STEPS TO RUN CODE*******************************************
1. Run myscript.sh as root
2. Run the below command to execute the code
USAGE: [./<exe name> | /dev/video0 | <frame_count> | <sharpen/ bnw / grayscale> | HRES | VRES]
EXAMPLE: ./Question5 /dev/video0 100 sharpen 640 480
Supported Resolutions:
1. 960x720
2. 800x600
3. 640x480
4. 320x240
5. 160x120
**********************************************************************************************
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ./Question5 /dev/video0 100 grayscale 800 600
Frame count = 100
Transformation = grayscale
Resolution = 800x600
System has 4 processors configured and 4 available.
Using CPUS=1 from total available.
rt_max_prio=99
rt_min_prio=1
Pthread Policy is SCHED_FIFO
PTHREAD SCOPE PROCESS
Service threads will run on 1 CPU cores
FORCING FORMAT
allocated buffer 0
allocated buffer 1
allocated buffer 2
allocated buffer 3
allocated buffer 4
allocated buffer 5
/**********************ANALYSIS*******************************/
Transformation :Grayscale
Please wait while analysis is being carried out................
Total Time Elapsed: 6.837643 seconds
Frame Rate in Hz for a Resolution of 800x600 = 14.624923
AVG WCET                = 0.005459 seconds
Deadline                = 0.006550 seconds
Positive Jitter         = 0.000000
Negative Jitter         = -0.102530
Total Jitter            = -0.102530
Total deadlines missed  = 0
```

Analysis for 640x480 resolution:



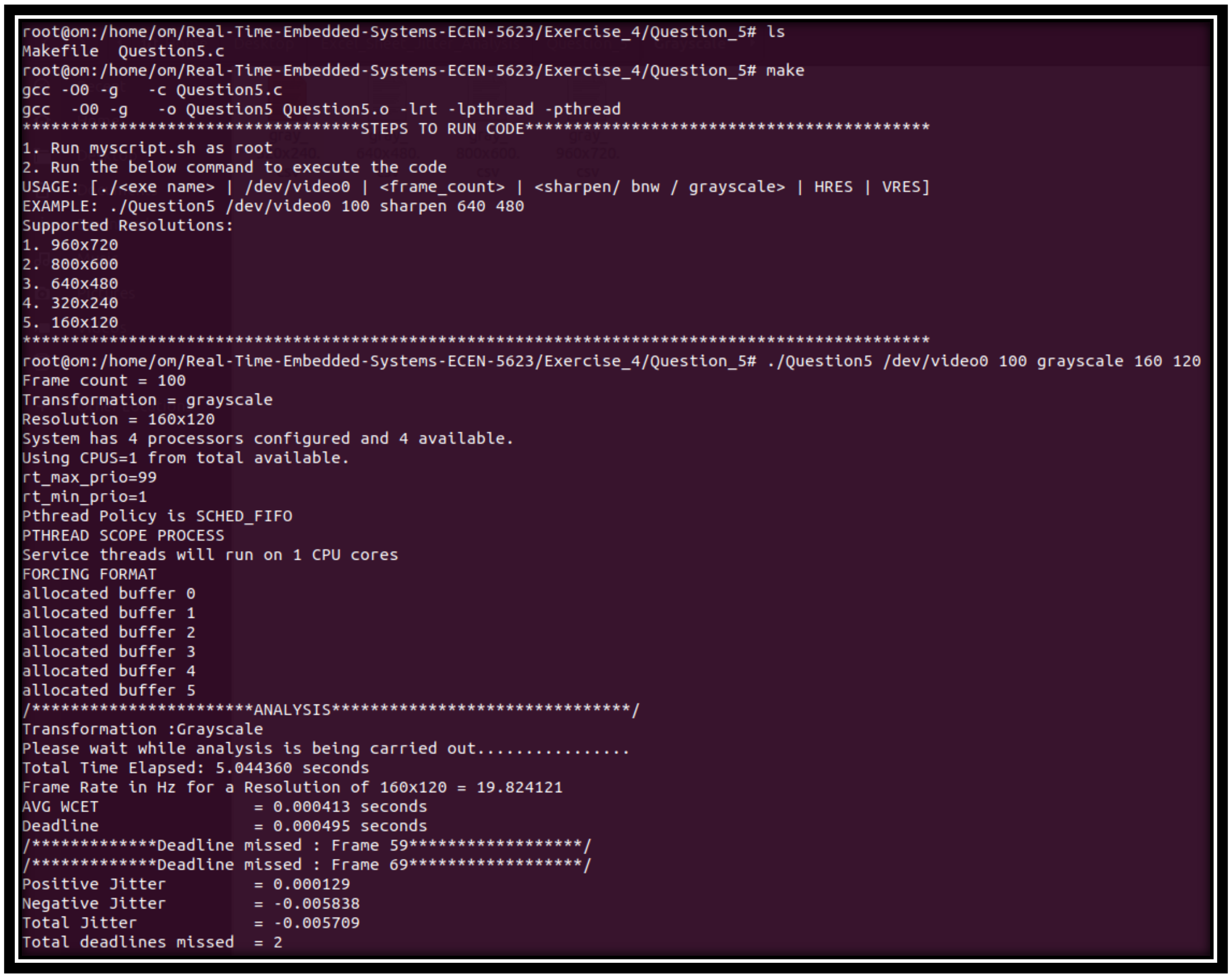
```
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ls
Makefile  Question5.c
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# make
gcc -O0 -g   -c Question5.c
gcc  -O0 -g   -o Question5 Question5.o -lrt -lpthread -pthread
***********************************STEPS TO RUN CODE****************************************
1. Run myscript.sh as root
2. Run the below command to execute the code
USAGE: [./<exe name> | /dev/video0 | <frame_count> | <sharpen/ bnw / grayscale> | HRES | VRES]
EXAMPLE: ./Question5 /dev/video0 100 sharpen 640 480
Supported Resolutions:
1. 960x720
2. 800x600
3. 640x480
4. 320x240
5. 160x120
********************************************************************************************
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ./Question5 /dev/video0 100 grayscale 640 480
Frame count = 100
Transformation = grayscale
Resolution = 640x480
System has 4 processors configured and 4 available.
Using CPUS=1 from total available.
rt_max_prio=99
rt_min_prio=1
Pthread Policy is SCHED_FIFO
PTHREAD SCOPE PROCESS
Service threads will run on 1 CPU cores
FORCING FORMAT
allocated buffer 0
allocated buffer 1
allocated buffer 2
allocated buffer 3
allocated buffer 4
allocated buffer 5
/**********************ANALYSIS*******************************/
Transformation :Grayscale
Please wait while analysis is being carried out...............
Total Time Elapsed: 5.079757 seconds
Frame Rate in Hz for a Resolution of 640x480 = 19.685980
AVG WCET                = 0.003591 seconds
Deadline                = 0.004309 seconds
Positive Jitter         = 0.000000
Negative Jitter         = -0.067954
Total Jitter            = -0.067954
Total deadlines missed  = 0
```

Analysis for 320x240 resolution:



```
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ls
Makefile  Question5.c
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# make
gcc -O0 -g   -c Question5.c
gcc  -O0 -g   -o Question5 Question5.o -lrt -lpthread -pthread
***********************************STEPS TO RUN CODE*******************************************
1. Run myscript.sh as root
2. Run the below command to execute the code
USAGE: [./<exe name> | /dev/video0 | <frame_count> | <sharpen/ bnw / grayscale> | HRES | VRES]
EXAMPLE: ./Question5 /dev/video0 100 sharpen 640 480
Supported Resolutions:
1. 960x720
2. 800x600
3. 640x480
4. 320x240
5. 160x120
***********************************************************************************************
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ./Question5 /dev/video0 100 grayscale 320 240
Frame count = 100
Transformation = grayscale
Resolution = 320x240
System has 4 processors configured and 4 available.
Using CPUS=1 from total available.
rt_max_prio=99
rt_min_prio=1
Pthread Policy is SCHED_FIFO
PTHREAD SCOPE PROCESS
Service threads will run on 1 CPU cores
FORCING FORMAT
allocated buffer 0
allocated buffer 1
allocated buffer 2
allocated buffer 3
allocated buffer 4
allocated buffer 5
/***********************ANALYSIS*******************************/
Transformation :Grayscale
Please wait while analysis is being carried out...............
Total Time Elapsed: 5.044883 seconds
Frame Rate in Hz for a Resolution of 320x240 = 19.822063
AVG WCET                = 0.001084 seconds
Deadline                = 0.001301 seconds
/*************Deadline missed : Frame 8******************/
/*************Deadline missed : Frame 81******************/
Positive Jitter         = 0.023612
Negative Jitter         = -0.021350
Total Jitter            = 0.002262
Total deadlines missed  = 2
```

Analysis for 160x120 resolution:


```
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ls
Makefile  Question5.c
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# make
gcc -O0 -g   -c Question5.c
gcc  -O0 -g   -o Question5 Question5.o -lrt -lpthread -pthread
**********************************STEPS TO RUN CODE******************************************
1. Run myscript.sh as root
2. Run the below command to execute the code
USAGE: [./<exe name> | /dev/video0 | <frame_count> | <sharpen/ bnw / grayscale> | HRES | VRES]
EXAMPLE: ./Question5 /dev/video0 100 sharpen 640 480
Supported Resolutions:
1. 960x720
2. 800x600
3. 640x480
4. 320x240
5. 160x120
*********************************************************************************************
root@om:/home/om/Real-Time-Embedded-Systems-ECEN-5623/Exercise_4/Question_5# ./Question5 /dev/video0 100 grayscale 160 120
Frame count = 100
Transformation = grayscale
Resolution = 160x120
System has 4 processors configured and 4 available.
Using CPUS=1 from total available.
rt_max_prio=99
rt_min_prio=1
Pthread Policy is SCHED_FIFO
PTHREAD SCOPE PROCESS
Service threads will run on 1 CPU cores
FORCING FORMAT
allocated buffer 0
allocated buffer 1
allocated buffer 2
allocated buffer 3
allocated buffer 4
allocated buffer 5
/**********************ANALYSIS*******************************/
Transformation :Grayscale
Please wait while analysis is being carried out...............
Total Time Elapsed: 5.044360 seconds
Frame Rate in Hz for a Resolution of 160x120 = 19.824121
AVG WCET                = 0.000413 seconds
Deadline                = 0.000495 seconds
/*************Deadline missed : Frame 59******************/
/*************Deadline missed : Frame 69******************/
Positive Jitter         = 0.000129
Negative Jitter         = -0.005838
Total Jitter            = -0.005709
Total deadlines missed  = 2
```

Final predictable response jitter analysis:

The jitter was calculated by subtracting the execution time of each frame with the deadline. If the difference between execution_time_per_frame and deadline is positive, it constitutes to positive jitter. On the other hand, if the difference comes out to be negative then the jitter is considered as a negative jitter. Jitter calculated is relative to the deadline. In my code, I set the deadline as average case execution time multiplied by a margin for safety. The average case execution time was multiplied by this factor to ensure that frames do not miss their deadlines. Following is the analysis of total jitter (positive jitter + negative jitter) for various resolutions and transforms. The unit of jitter is seconds here. The negative jitter indicates that the frames did not miss their deadlines, while positive jitter indicates that deadlines were missed by a few frames. This does not mean that if the total jitter is negative, then all frames have been captured before their deadline. Similarly, total jitter being positive does not mean that all frames have missed their deadlines. Overall jitter will heavily depend on the deadline being set for that application. Deadlines must be accurately determined in order to avoid a lot of jitter.

Type of Transformation	**Resolution**				
	960x720	**800x600**	**640x480**	**320x240**	**160x120**
Sharpening	-5.472957	-3.778228	-2.574138	-0.612099	-0.147898
Black and White	-0.147429	-0.102530	-0.067954	0.002262	-0.005709
Gray Scaling	-0.190684	-0.177666	-0.112426	-0.235752	0.080305

REFERENCES:

1) Sharpen code: [http://ecee.colorado.edu/~ecen5623/ecen/ex/Linux/computer-vision/sharpen-psf/sharpen.c]
2) Linux Media Kernel API documentation: https://linuxtv.org/docs.php
3) **GitHub Link:** https://github.com/omraheja/Real-Time-Embedded-Systems-ECEN-5623