ECEE 5623, Real-Time Systems:

Exercise #4 – Real-Time Continuous Media

DUE: As Indicated on D2L by midnight and Syllabus

Please thoroughly read Chapters 7 & 8 in RTECS with Linux and RTOS

Please see example code provided - Linux, FreeRTOS, VxWorks, Zephyr

This lab is written to be completed with embedded Linux running on the DE1-SoC, Jetson or Beagle board(s), but you are welcome to complete it using FreeRTOS, VxWorks or Zephyr as an option. Note that you will either have to adapt the example code or use equivalent examples from the links above for RT-Clock, use of tasks in place of pthreads, and the appropriate board to boot FreeRTOS, VxWorks, or Zephyr as described on the course main web page.

Exercise #4 Requirements:

1) [10 points] Obtain a Logitech C200 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.

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.

3) [10 points] Using your verified Logitech C200 camera on a Jetson and verify that it can stream continuously using to a raw image buffer for transformation and processing using example code from the computer-vision folder such as simple-capture, simpler-capture, or

simpler-capture-2. Provide a screen shot to prove that you got continuous capture to work. Note that simpler capture requires installation of OpenCV on your Jetson – this will likely already be available on your board, but if not, please follow simple instructions found here to install openCV [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].

4) [20 points] Choose a continuous transformation OpenCV example from computer-vision 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.

5) [50 points] Using a Logitech C200, choose 3 real-time interactive 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.

Overall, provide a well-documented professional report of your findings, output, and tests so that it is easy for a colleague (or instructor) to understand what you've done. Include any C/C++ source code you write (or modify) and Makefiles needed to build your code. I will look at your report first, so it must be well written and clearly address each problem providing clear and concise responses to receive credit. Note that it is up to you to pick an OOP for implementation and tools of your choice to develop and test your application – this is a large part of the challenge of this assignment.

Grading Rubric

[10 points] Camera and device driver verification:

[5 points] USB hot-plug ____

[5 points] UVC driver verify ____

[10 points] Camera streaming checkout:

[5 points] Use of tools such as camorama______________________________________

[5 points] Verification____

[10 points] Camera continuous streaming tests and applications:

[5 points] Build and test ____

[5 points] Streaming verification___________ ______________________________

[20 points] Continuous transformation tests and applications:

[10 points] Demonstration ____

[5 points] Description____

[50 points] Soft real-time analysis and conversion to SCHED_FIFO with predictability analysis:

[10 pts] Design concepts____

[10 pts] Algorithm analysis____

[10 pts] Prototype analysis____

[10 pts] Final predictable response jitter analysis_____________________________