ECEE 5623, Real-Time Systems:

Exercise #4 – Real-Time Continuous Media

DUE: As Indicated on Canvas

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 R-Pi3b (order or borrow and use NOOB image)**. **It is possible to also use a Jetson Nano (Getting started)**, Jetson TK1/TX1/TX2 (JetPack install) or Beagle board(s). **The standard final project requires use of a camera, which is simple with embedded Linux using the UVC driver**. Linux examples can be found here - http://ecee.colorado.edu/~ecen5623/ecen/ex/Linux/code/

Exercise #4 Requirements:

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.

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.

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 R-Pi 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].

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.

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.

[20 points] 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 and make sure your code is well commented, documented and follows coding style guidelines. 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: Linux manual pages can be found for all system calls (e.g. fork()) on the web at http://linux.die.net/man/ - e.g. http://linux.die.net/man/2/fork

In this class, you'll be expected to consult the Linux manual pages and to do some reading and research on your own, so practice this in this first lab and try to answer as many of your own questions as possible, but do come to office hours and ask for help if you get stuck.

Upload all code and your report completed using MS Word or as a PDF to Canvas and include all source code (ideally example output should be integrated into the report directly, but if not, clearly label in the report and by filename if test and example output is not pasted directly into the report). ***Your code must include a Makefile so I can build your solution on an embedded Linux system (R-Pi 3b+ or Jetson). Please zip or tar.gz your solution with your first and last name embedded in the directory name and/or provide a GitHub public or private repository link. Note that I may ask you or SA graders may ask you to walk-through and explain your code. Any code that you present as your own that is "re-used" and not cited with the original source is plagiarism. So, be sure to cite code you did not author and be sure you can explain it in good detail if you do re-use, you must provide a proper citation and prove that you understand the code you are using.***

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 _____

[10 points] Description_____

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

[5 pts] Design concepts_____

[5 pts] Algorithm analysis_____

[10 pts] Prototype analysis_____

[10 pts] Final predictable response jitter analysis_____

[20 points] Quality of reporting and code quality and originality:

[10 pts] Professional quality of reporting, testing and analysis (0…6 is below average, 7 is average, 8 is good, 9 excellent, and 10 is best overall.)_____

[10 pts] Code quality including style, commenting, originality, proper citation for re-used code, modified code, etc._____