SE3910 Lab 7: GStreamer in C

Due: April 30, 2014 (Dr. Schilling's section)

Due: Week 8, Monday (Dr. Yoder's section)

1. Introduction

Two weeks ago, we started to capture images using GStreamer. This week, we are going to do a more thorough analysis of the timing of script-based and C algorithms.

2. Lab Objectives

- Practice writing in a low-level compiled language
- Compare the run-time of compiled and scripted code
- Compare timing results obtained internally (using the time utility) and externally (using an oscilloscope)

3. Prelab

None for a change.

4. Part 1: Removing the script from Week #5

In week #5, we worked on taking pictures with the camera. As a first step toward integrating this into a Qt app, we would like to do this in C. Here's a mechanism we can use to do that.

On the course website, there is a program called testcode which essentially mimics the script we wrote a few weeks ago except that it is implemented entirely in C. Using this code for ideas, modify your code from two weeks ago to be written entirely in C. You will only be able to build the code on the beaglebone, as the libraries are not setup for cross compilation on your virtual machine (nor can they easily be setup.) You will need to use or modify the included makefile to make the code.

In particular, your app should:

- Provide documentation of usage when run without arguments, ***for example…***

```
Usage: camera option [file.png]
     option – one of "snap" or "timer"
        snap - take picture immediately
        timer – take picture after some amount of time
     file.png – the file to be written
```

- Be documented internally – all methods, global variables, etc.
- Allow the user to either take a picture immediately or wait until the timer has gone off
- Save the user's pictures to filenames sequentially rather than overwriting files
- Set a GPIO pin high right before taking the picture, and set it low after

5. Part 2: Timing tests

First use the "time" utility to measure the amount of time the process takes to run, as in Dr. Yoder's section of Lab 4. Then, measure the time on an oscilloscope.

6. Deliverables / Submission

Each team will be responsible for submitting one report with the following contents:

1. Introduction -> What are you trying to accomplish with this lab? This section shall be written IN YOUR OWN WORDS. DO NOT copy directly from the assignment.
2. Timing Analysis
   a. Present a table (or tables) of timing results including
      i. five sample times gathered by the "time" utility
      ii. five sample times gathered by the oscilloscope
      iii. the mean & sample std deviation of these
      iv. your results from the bash script program last week
      v. (optional: Get the 10 samples for the bash script, too)
   b. For the modified program, is the timing any different than it was when you simply used a script? Does putting everything in C offer any performance advantages?
   c. Do the two timing approaches yield the same times?
3. Things gone right / Things gone wrong -> This section shall discuss the things which went correctly with this experiment as well as the things which posed problems during this lab.
4. Conclusions -> What have you learned with this experience? What improvements can be made in this experience in the future?
5. Appendix -> Source code
   a. For each program, include the source code you used. Code should be well commented and documented. Source code should show which portion of the lab it was intended for.
   b. (See the documentation requirements above)

Dr. Schilling's Section (tentative)

This material should be submitted as a single pdf file.

Additionally, all of your source code shall be submitted in a tar.gz file. When the tar file is extracted, your code shall cleanly build without warnings on the Beaglebone by issueing a simple make all command. If you use any arguments for make, please document these in your report.

Dr. Yoder's Section

The source code should be included as a ZIP. The source code should be structured as follows:

- lab7.zip
  - lab7
    - Makefile
    - source.c
    - source.h
    - (optinal) any scripts

- etc.

When make is run (with appropriate arguments, e.g. make all –f makefile), the program should build for the Beaglebone. If you use any arguments for make, please document these.

If you have any questions, consult your instructor.