SE3910 Lab 5: Building a Camera using GStreamer

Due: April 16, 2014

1. Introduction

In last week's lab, we built a simple networked application, which bounced signals back and forth over the network, exploring the concept of latency. This week, we are going to move further into the realm of software development and begin interfacing with the camera.

2. Lab Objectives

- Interface with the camera port
- Interface with GStreamer to capture an image.

3. Part 1: Understanding GStreamer

GStreamer (http://gstreamer.freedesktop.org/) is library that is available to for developing applications which process audio and video streams. It can interfaces exist for using the library from both the command line and from within a C program.

To begin with, you will need to install the libraries on your beaglebone image. From the command line shell, issue the following commands:

sudo apt-get install gstreamer-tools
sudo apt-get install gstreamer0.10-plugins-base
sudo apt-get install gstreamer0.10-plugins-good
sudo apt-get install gstreamer0.10-dev

These three commands will download and install the gstreamer libraries on your beaglebone device. Note that you must be on the network for these commands to work, and the network must have access to the outside world.

Next, plug in the camera to the USB connection on your Beaglebone. With the camera plugged in, enter the following command on the shell. This will cause the beaglebone to capture a single image from the camera and store it in the file test.png. You can view the file by transferring it to your Linux virtual machine by sftp, and then view it on your Linux VM.

```
Gst-launch v4l2src device=/dev/video0 num-buffers=1 ! ffmpegcolorspace
! pngenc ! filesink location=test.png
```

4. Part 2: Building a controlled camera

Part 2 involves creating a controlled camera. In essence, for this part of the program, you want to connect a push button to the board to trigger taking a picture. To do this, use the techniques from week 1 and 2 to build a camera and write the code to take a picture.

To do this, you will need to write a simple shell script encapsulating the commands necessary to take a photo and store it to a given file. A good resource on this is http://linuxcommand.org/writing_shell_scripts.php.

Once you have successfully written and tested the shell script, you will need to write the camera application such that it calls and invokes the shell script to take pictures. For this, you will use the circuitco cape and two buttons. One of the buttons will "immediately" snap a picture. The other button will provide a 10 second countdown before snapping the picture by blinking the LED once per second for the first 7 seconds and then 4 times per second for the last 3 seconds. Additionally, this second picture will be stored in jpg format instead of png format. (Hint: You'll have to look a bit at the documentation for GStreamer to see how to do this.)

Pictures should be numbered incrementally (i.e. 1, 2, etc.)

5. Part 3: Measuring how long it takes to snap a photo.

As a last piece, we want to know how long it takes for the photo to be snapped. To do this, select a pin that you are going to use as a GPIO pin. Right before you invoke the script to take the picture, set the pin high. As soon as the photo is done being taken, set the pin back low, and measure the difference. This is the amount of time it takes to capture an image.

6. Deliverables / Submission

Each person 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. 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.
3. Example photos
    a. Include in your lab report a selfie showing your camera system working for both the jpg and png formats. With these photos, also include how long it took for the photos to be taken.
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.

This material should be submitted as a single pdf file.

If you have any questions, consult your instructor.