ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

To: Dr. Mark Yoder

From: Michael McDonald

Subject: Week 5 (Kernel and memmap, Exercises 8, 8a, 11b, 23) Memo

Date: October 8, 2013

1 Exercise Summary

Homework 5 is complete.

Objective	**Exercise**	**Status**	**Notes**
Compile the 3.8 Kernel	Exercise 8	Completed	I did this in Memo 2, because it was on the calendar then.
Install necessary cross compilation tools	Exercise 8a	Completed	I did this in Memo 2, because it was on the calendar then.
Memmap GPIO access	Exercise 11	Completed	This is awesome. It also makes me want to write another layer of abstraction to do GPIO calls, because it's a bit of a pain.
Building the Linux Kernel	Exercise 23	Completed	Can't wait to start playing with the Kernel and changing it :)

I ran the experiments in the extras section and experienced some strange behavior. First off, the performance seemed to actually be on par with the shell file (if not worse). The output signals weren't clean, and they were definitely not accurately representing the input properly. They also delayed significantly longer than anticipated, from nearly zero to two or three ms, which is a significant delay that we should not be dealing with. Setting the nice priority (`nice -n .\gpioThru &`) didn't seem to change things that much either, even when I set the nice to -20 (on par with the Kernel level processes). I wasn't running any other processes in my Userspace, so I'm not entirely sure what is causing my problems, but I would like to test it on another bone to see if it's my bone or if it's my code.

The main points I took away from these exercises were the following:

- I'm really fascinated at how the Linux kernel was created, and how it accomplishes the tasks it needs to. I'd also be curious to know how small one could make it in order to put it on smaller and smaller platforms (possibly even cut into the market of Arduino [especially now that those are starting to grow in size]).
- I feel like taking this class, especially checking out all the low level code, makes computing feel less high level and abstract, and adds a concreteness that