Hazen Hamather
ECE497 – Embeddd 32 bit Linux
Homework 7

Reading and Writing GPIO Pins

This homework focused on the different methods to read and write to GPIO pins. The first method utilized Python, the second mmap with C, the third the kernel, and the fourth was the PRU of the BeagleBone Black. Python was surprisingly quick when it came to reading the input pin and outputting. Of all the methods and even though this was my favorite method, it was unfortunately the slowest method to respond to the square wave being generated. The next fastest method was mmap() in C. This resulted in a response time almost three times as fast as what Python could offer us. This huge discrepancy could potentially be caused by the Adafruit GPIO library I used with the Python code. Adafruit most likely had a debounce time built in to the method, leading to an increase in the overall response time. The next quickest time was the kernel method and this was expected. When operating at such a low level and directly accessing the starting and ending addresses, one expects "reaction" times to be a bit quicker. Finally, an unbelievable faster method than all of these was the PRU. Finishing in first place, approximately three orders of magnitude smaller than second place, was the PRU with a time of 20 nanoseconds. A summary of my findings can be found in Table 1 below.

Table 1: Summary of response times and percent of CPU used

Method	Response Time (us)	%CPU Used
Python	178	80.6
Mmap via C	60	98.6
Kernel	29.4	0.7
PRU	20(ns)	0.1

An sample oscilloscope output can be found in Figure 2, which shows the response time of the kernel process.

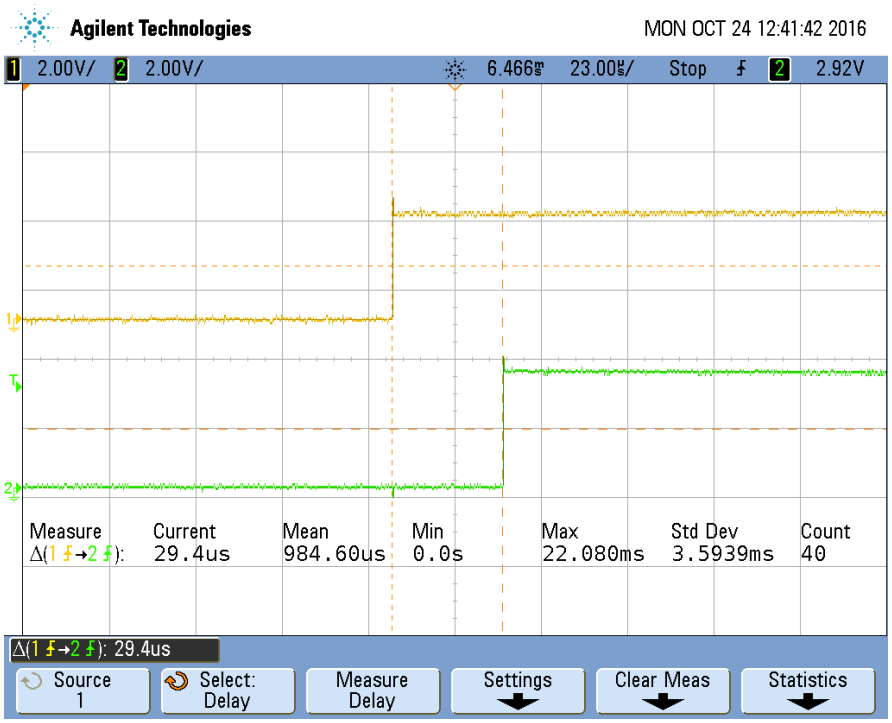

Figure 2: Oscilloscope output for the kernel method

Figure 3 below summarizes the CPU usage for the same method.

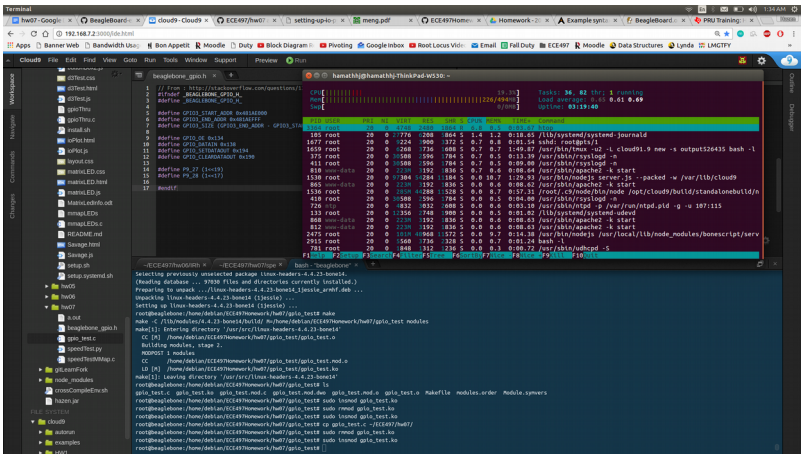

Figure 3: CPU usage for the kernel method

With only 0.7% CPU usage, the kernel method was quite efficient. But as great as the kernel approach was, the PRU smoked the competition with a CPU usage percentage of 0.1 or less with a response time of 20 nanoseconds.