Thomas Winter – thwinter

ECE 785 Project 1 Report

Hardware Configuration

- Raspberry Pi 4
- 4GB of RAM
- 16GB SD Card
- No cooling used on the SoC
- Not using a case on the Raspberry Pi
- Using 480p touchscreen display to view the system

Temperature, Frequency, and Voltage

Monitor Program

The maximum update rate that had significant improvement on the rate was setting the interval to 200 ms. I set the maximum frames to run through to be 100. Setting the interval below 200 ms still resulted in completing the 100 frames in 79-80 seconds.

Data

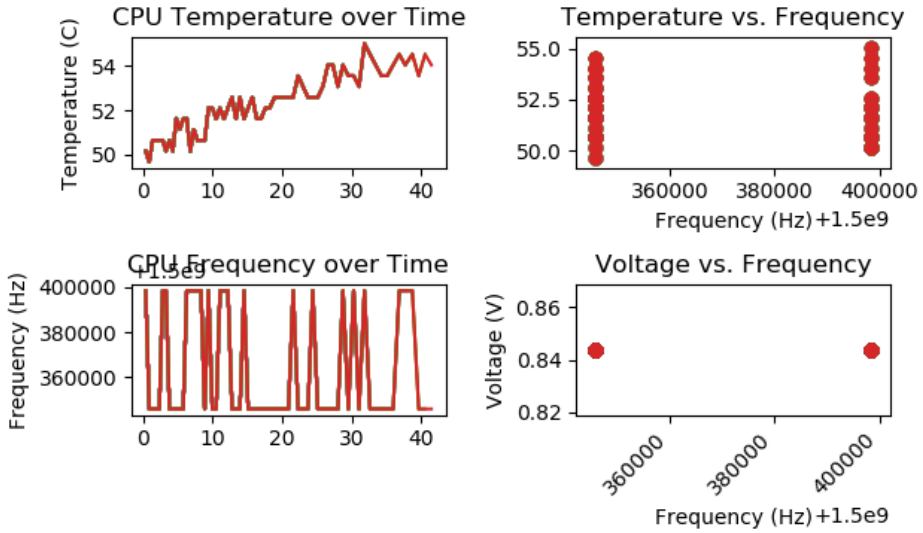

Figure 1. Baseline Data with no browser or benchmark running

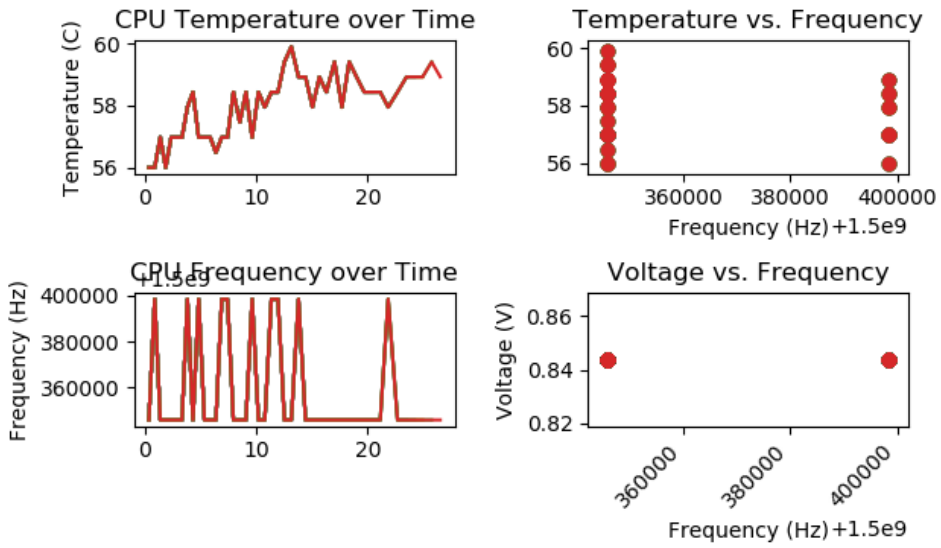

Figure 2. On Demand Governor with the C/C++ benchmark running around the 5 second mark. Note the jump in temperature and then sudden drop after the benchmark completed around the 7 second mark.

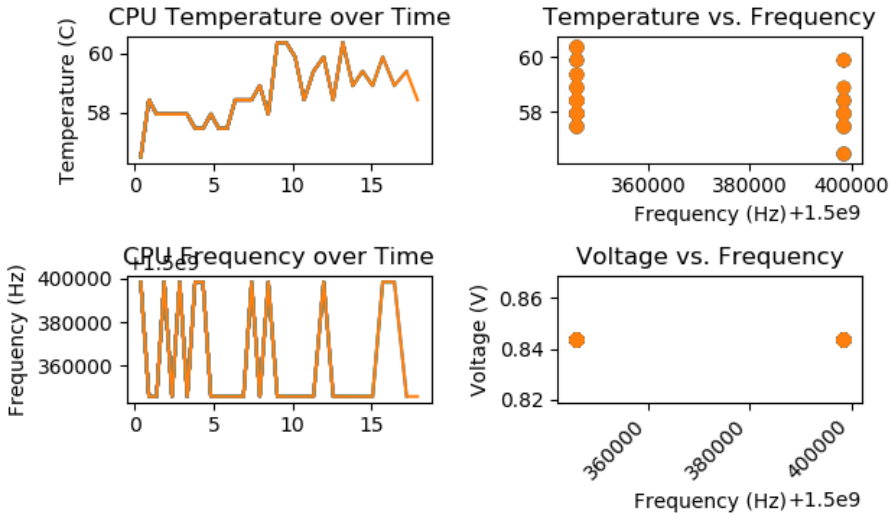

Figure 3. Performance Governor with the C/C++ benchmark running around the 8 second mark. Note the jump in temperature and then sudden drop after the benchmark completed around the 10 second mark.

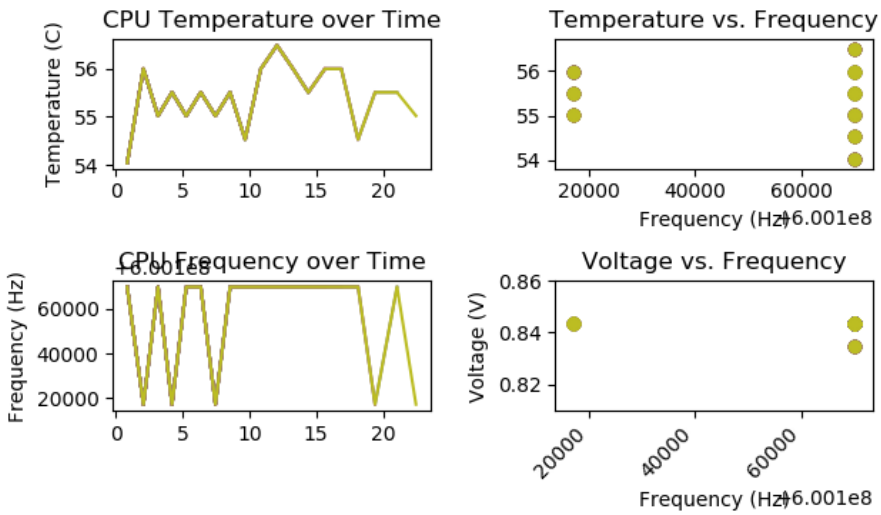

Figure 4. Powersave Governor with the C/C++ benchmark running around the 10 second mark. Note the jump in temperature and then sudden drop after the benchmark completed around the 15 second mark.

Analysis

The CPU was not throttled. I did notice that the Voltage remains pretty much constant no matter the CPU frequency. The "vcgencmd measure_voltage core" command actually measures the GPU voltage instead of the ARM voltage, so it is expected to remain pretty constant even with a higher CPU frequency because the voltage does not correspond to the CPU. I noticed that the Powersave Governor resulted in much lower temperature readings compared to the readings of the Performance Governor.

Benchmark Run-Time Performance

Instrumentation Overhead

Table 1. C/C++ Completion times depending on FuncAnimation Interval and cpufreq Governor

	cpufreq Governor		
Interval	**On-Demand (sec)**	**Performance (sec)**	**Powersave (sec)**
200	1.763	1.773	4.386
1000	1.796	1.748	4.392

Analysis

The FuncAnimation Interval does not seem to produce any significant difference between the completion times of the C/C++ benchmark test. I also ran the shell program that runs all of the benchmark tests, and I still did not see a significant difference between the completion times depending on the FuncAnimation Interval.

Data

Table 2. Benchmark Completion times depending on cpufreq Governor

	cpufreq Governor			
Language	**On-Demand (sec)**	**Performance (sec)**	**Powersave (sec)**	**Powersave : Performance**
C/C++	1.808	1.747	4.373	2.503
C++11	1.515	1.513	3.795	2.508
Haskell	2.661	2.653	6.635	2.501
Java	2.911	2.475	6.140	2.481
Python	156.782	156.771	387.100	2.469

Analysis

Because the ratio of Powersave completion time to Performance completion time was about 2.5 for every programming language, the benchmark tests must be close to purely being CPU-bound. This is because the Performance CPU frequency is 1500 MHz while the Powersave CPU frequency is 600 MHz, so the Performance frequency is also 2.5 times the Powersave frequency.