Milestone Two Report

Group Members: Colin Thackston, Jacob Rogers, Geri Rista

Our task for this milestone was to create a STM32 based system to collect and log power data to RAM. Eventually, we should be able to build a separate system to work parallel with the trailcounter to monitor what state it is in and how much power it is drawing. The power system is a not deliverable to the customer, but rather a tool for our lab to use in order to analyze power usage.

Our system require two different discovery board because one must run a program (and thereby a voltage that we can measure) and the other measures the flow of this power. We are still utilizing the µCurrnent to convert the voltage using Ohm's Law. The µCurrnent device is a high-precision resistor that then returns a voltage that we know is directly proportional to the amount of the power the system is using. On the running board, we plugged the µCurrnent leads into the discovery board IDD pins to draw the current out of the board. On the other end of the µCurrent, we connected the leads to the PC0 and PC1 for our GPIO pins to receive the voltage. In order to transform the voltages from the µCurrnent to useful data, we initialize the analog-digital converter (ADC) which encodes the readings into a binary number. The process works by indicating the number of discrete values the ADC can produce over the range of analog values. The minimum charge required to produce a signal is the least significant bit. The number of voltage intervals is defined by N=2^M where M is the ADC resolution in bits. The successive approximation algorithm then computes a summation that resembles the continuous voltage. Once the system is initialized and running, all one needs to do in order to read the power in the running board is pull the values off of pins PC0 and PC1.

Implementation of this application proved to be more difficult than we initially thought. Aside from some appalling github issues that were entirely self-inflicted, we encountered problems from the very beginning. Initially, we could not read data off of the GPIO pins and spent a couple hours debugging it. After changing batteries in the μCurrnent, we still could not read data. We determined that this was because our ADC and pins were not configured correctly because we kept reading the same value regardless of whether or not we connected the μCurrnent device to the board. Consulting the data sheet revealed we skipped several steps in setting up the ADC. Eventually, we had to boilerplate code that Bryce gave us because the initialization process proved to be too complicated for us to understand with our limited background and documentation. Overall, our struggles with the ADC further revealed our unfamiliarity with the hardware and the immense volume of material. Once we got the ADC working, extracting data proved to be quite straightforward.We wrote the data to a large buffer and then looped through it to get the readings. We opened up the two channels and measured the difference which was then written to the buffer.

The results we obtained for this milestone were similar to those that we obtained for the previous milestone for when our board was in a running state and blinking all of the LEDs. However, before measuring the amount of milliamps drawn by a STM32 blinking all LEDs, we first conducted a test to see if the values being output were reasonable. This consisted of measuring the milliamps drawn by our first discovery board by reading from the 3V pin on it and making sure the values were in the 30mA range. We then read the milliamps from the GND pin, where we saw a drop in milliamps down to the 2.5 range. These are about the values we'd expect being output when wiring up to these two pins. Finally, we then used our initially STM32 board and µCurrent device to read the amount of power being drawn by our second STM32 when all of its lights are blinking. Sampling the power drawn while this is going on gave us values between the range of 38 to 44 milliamps drawn. These values were then saved to flash memory by storing them into an array. The values stored are close to the value read from our oscilloscope which was 45 milliamps.

There are various possible reasons for the discrepancy between the reading we got from the oscilloscope and the reading we got from our STM32 system. One reason might be due to individual variance between the boards themselves. One board might draw slightly more power or slightly less power than another. Another possible source of error consists of some loss of precision due having to convert our values from a float into two separate ints. This is so that we could actually print out our results since ChibiOS does not support printing floating point numbers. Finally, we did not calibrate our oscilloscope prior to use and so there is some margin of error from our oscilloscope measurement from this. The combination of these factors plus general electrical interference are possible contributing factors as to why the values output from our system do not match up to the values output from the oscilloscope. However, due to the difference being roughly only 1 milliamp, we hope this is within the acceptable margin of error.

Proceeding forward from this position, our next goal is to be able to save power data onto an SD card on top of the current capability to store our data in flash. Furthermore, we should also be able to have the capability to capture whether the system we're reading power data for is in a run, sleep, or standby state. We hope that the accomplishment of this next task will proceed more smoothly than our current task since now we already know how to setup our system for measuring collecting and logging power data. While this lab did not initially go as we wanted, we were able to successfully complete our task and accomplished all of our required goals.