The external hardware monitor project is designed be a tool for those who are gaming, overclocking, or just interested in keeping track of information about their PC. It is external so that it does not require a window to be open on the main computer at all time to see the information. Even with 3 monitors myself I don’t like to make half of one of them unusable for other tasks. Most functionality that was planned was implemented successfully with some of the customization options for more users missing. The raspberry pi screen displays rolling graphs by the second and has an LED function as a quick temperature meter. At this point the project is done and below are the details of the process. I have added citations at the end for all the websites that were useful to in some form or another.

I had a very slow start with development. Being new to python I had a hard time jumping right into something more than basic. My first issue was that I was convinced that I needed to scrape the website itself to get ahold of the data I needed, something I’ve never done before either. After much trial and error, I finally figured out that the data wasn’t even on the website, and that there were java scripts that were displaying everything. I felt like this was the end of the road, if I was not able to see any of the data by viewing the page’s source code then this project didn’t seem possible. More reading showed that I needed to turn my attention to the json file that I found by going to the network tab of the browser source code viewer. This is the file that had everything I needed, only in a strange format I had never experienced before. After learning how to scrap data from a website I learned how to read data from a json file. The formatting was different that most examples online and it turned out I need to read it using recursion because of the nesting structure. This gave me a Unicode list/string of everything that I saved to a text file.

Once I could get my hands on the numbers that I needed I felt a lot better about my progress. At this point it had been around 15 hours and I finally had something useful. Quickly I figured out how to pick out specific attributes that I wanted, like CPU package temp instead of separate temperatures for each core. I put everything in an infinite loop with print statements and a 1 second sleep. This means I have a terminal window that displays data and updates once per second by requesting the website each time. Next, I figured out how to scrub the % and °C from the Unicode using regular expressions. This allowed me to convert each entry into a float type to be used for graphing or other purposes. I have only just started using matplotlib to create a simple graph, but it seems promising.

I have not tested functionality with the LED yet but GPIO programming seems to be very easy in python. I did plug it into the grove port on the Pimoroni and its PCB got extremely hot, so I am considering using a plain LED instead. I was planning on having the RGB LED color shift depending on temperatures but an off, solid, and blinking led could serve the same purpose if needed. From here I will continue to add graphing and then an acceptable GUI to contain everything. Once that is done, I will add the LED to the mix and should be close to finished. The final thing that I want to add is a config file where a user would be able to decide what information is displayed, what order, and how often it is updated. Due to the nature of the json layout most computers will not have the same data order. Currently it is read by what line number I know needs to be used, I may be able to change this to recognize specific components. This would make the user setup easier.

I spent some time learning about matplotlib and decided that it would function well enough for what I needed without adding a Tkinter GUI. I did have some issues implementing it in a simple way though. The way that the live updating graphs run halts any other progress for the python code. “update graph” isn’t a function that can just be called on your own terms like I expected it to be. The function starts an infinite loop where it runs at a set interval, so I ended up using 2 scripts together. The first script that I had already been working on can call the one with graph functionality to begin but I found running them individually is more stable. With 2 separate scripts I needed a way for them to communicate so I had the data needed written to a file by the first script and read into the graph by the second script. I don’t think this is the most elegant solution and with more time I may have been able to do everything in one script using lists. Using files made it easy to delete the first line, which is the first data point, after 60 had already been written. This is what gives the graph the continuous rolling appearance. Formatting was a bit tricky, but I found a way to get rid of the menu bar, change the window title, remove x axis labels, and get 4 graphs to display at the same time.

The LED that I had purchased did not seem to work in any meaningful way for me because of it using a I2C interface. I needed additional driver board and software. I decided to see if I could make anything happen with the LED, it was a 4 pin RGB led so it should run without the board that the connector is attached to. After using an Arduino board’s 3.3v and ground to test it was surprising to see that the LED was a common anode. This meant that instead of hooking the 4th pin to a ground and using the remaining to express the color I wanted, I needed to do the opposite. The Hyper pixel display has a 3.3v, ground, and 2 pins that are mapped to 10 and 11. At this point the screen software needed to be installed or the passthrough pins would not work, that went smoothly, and the graphs could be seen on the screen. I desoldered the LED from its board and wrote the code for the LED to change color based on temperature using a breadboard. It changes from green to yellow to red as the temperature increases and this combination is possible without needed a pin for blue. The light ended up being so bright that I had to add 2 1k resistors to each of the 2 pins just to be able to look at it. Now that it was all working correctly, I soldered everything that was on the breadboard to a prototype soldering board that would fit nicely on top of the pi.

Everything is functional and has gone according to plan for the most part. For testing I put the pi on Wi-Fi since it had been until this point hard wired with ethernet. This showed a problem were after around 10 seconds the script would end because of connection errors when fetching data, I almost noticed that the VNC viewer I was using was also timing out and lagging badly. I had not had any of these issues before starting this project and they seem to persist even after the program has stopped running and the pi has been restarted. This is disappointing because much of the reason for this project was to have something that could be placed almost anywhere. In the future I think that cleaning up the code and adding a way to handle a disconnect could help. The file IO that is performed keeps the CPU at around 33% usage and adding the graphs bumps it up to 55% or so. I would like to reduce this as well, keeping the pi working that hard constantly is probably not a good idea either.

As for the stretch goals I had mentioned like a custom user configuration, I ran out of time to spend given the guideline of about 4 labs worth. I am happy with the core functionality and I intend to improve upon it later. After cleaning up the code, fixing the file IO, and adding user configurations my plan is to find a slightly better screen to use. The size is a bit small to read from a glance at more than arm’s length away and 480p means that only a small amount of information can be read. I think something like a 1080p 8-10-inch screen with the pi mounted to the back would be a good direction to go. Using HDMI would free up many more options for LEDs and other hardware. I want to try the I2C/Grove system that my LED was a part of, multiple LEDs along the side could be used like a thermometer. A temperature sensor could be added to the system to read the ambient room temperature as well.

Overall, I learned a lot with this project. It is something that I have been wanting with no products available to buy. Being able to build this and explore other project ideas is satisfying and I already have reasons to buy a few more raspberry pi’s. Python was the biggest hurdle and I wish we would have talked about that in class a bit for the sake of the projects. It ended up being much more different than the other languages I have used in the past, mainly C++ and java. The library it uses for controlling the pi’s GPIO was very streamlined and easy to use, however. I look forward the building more tools for myself and family.

Youtube Link <https://www.youtube.com/watch?v=e5MK5dp6yCo&feature=youtu.be>

gpio - connecting common anode rgb led. (n.d.). Retrieved April 25, 2019, from Raspberry Pi Stack Exchange website: https://raspberrypi.stackexchange.com/questions/53588/connecting-common-anode-rgb-led

GPIO in Python - Raspberry Pi Documentation. (n.d.). Retrieved April 25, 2019, from https://www.raspberrypi.org/documentation/usage/gpio/python/README.md

Graph Sensor Data with Python and Matplotlib - learn.sparkfun.com. (n.d.). Retrieved April 25, 2019, from https://learn.sparkfun.com/tutorials/graph-sensor-data-with-python-and-matplotlib/all

how to select data from a JSON file in python? (n.d.). Retrieved April 25, 2019, from Stack Overflow website: https://stackoverflow.com/questions/19153009/how-to-select-data-from-a-json-file-in-python

How to use python parse json with nested children. (n.d.). Retrieved April 25, 2019, from Stack Overflow website: https://stackoverflow.com/questions/55679293/how-to-use-python-parse-json-with-nested-children/55679449?noredirect=1

KingDrakorex. (n.d.). *External hardware monitor with LED*. Retrieved from https://www.youtube.com/watch?v=e5MK5dp6yCo&feature=youtu.be

Python matplotlib Cairo error. (n.d.). Retrieved April 25, 2019, from Stack Overflow website: https://stackoverflow.com/questions/27749664/python-matplotlib-cairo-error

Python Programming Tutorials. (n.d.). Retrieved April 25, 2019, from https://pythonprogramming.net/python-matplotlib-live-updating-graphs/

Python, R. (n.d.). Working With JSON Data in Python – Real Python. Retrieved April 25, 2019, from https://realpython.com/python-json/

Python Subprocess – Python Tutorial. (n.d.). Retrieved April 25, 2019, from https://pythonspot.com/python-subprocess/

Soren. (2018, January 12). Making a LED blink using the Raspberry Pi and Python. Retrieved April 25, 2019, from Raspberry Pi HQ website: https://raspberrypihq.com/making-a-led-blink-using-the-raspberry-pi-and-python/