Engineering Addendum

Hello Future Developers of Project Vulcan,

Project Vulcan was created as an ECE Senior Design Project for the 2016-2017 academic year for Professor Densmore and the CIDAR Lab. The ultimate goal was to advance the science of microfluidic devices and give them the capabilities to conduct biological experiments. This project was also made to be expanded upon by creating modular hardware primitives easily compatible with the current Neptune/Fluigi/MINT software as well as the physical chips. We have created two hardware modules, but this can be expanded to include more modules with a variety of different purposes. This letter presents a discussion of what we believe could be added to the project.

PCR:
Interface: More data can be shown, such as the steady state error, ramp up, and ramp down times.
Software: Any dramatic changes to the chip should cause for evaluation in the code for the PID controller. Namely, the P, I , and D values should be retuned. The maximum PWM value should be re done and ask CIDAR Lab if it warrants further re-evaluation of the mathematical model used in the code (the formula may not be accurate for a dramatically different chip)
Hardware: There is room for improvement, as the current 3D printed case can be cleaned up and make the module look neater. A cooling pad can be added and then manipulated in the code. This would help expand the range of temperature the module can do.

Peripheral Manager:
Software: The Manager is as close to functionally complete as can be - the only updates that can be performed are UI updates or a rewriting of the entire manager in a language such as Go to improve the performance of the manager and further reduce latency in communications.

Droplet Characterization:
The longest portion of effort and time put into the development of the Droplet Characterization went to the CV Algorithm. CV Algorithms basically breakdown and parse images in ways to expose areas of interest and information is gathered by looking at the pixels around the region. For the module, a successive order of filters and transformations are applied to every frame of the video to expose the droplets while reducing noise as much as possible. Specifically, the order is:

- Copy the frame
- Grayscale the frame
- Blur the frame
- Perform a binary threshold
- Draw contontours
- Identify static region(s) of interest on the frame
- Observe contour and pixel data when contours pass through the region(s) of interest
- Perform math on observed data
- Store the modified data and associate it with a contour
- Go to the next frame

It is relatively simple just talking about it, but figuring out the order and parameters was an endeavor. CV is general is extremely volatile to changes in lighting of the target video. Because everything is computed from the actual pixels of the medium, changes in lighting fundamentally alter the RGB distribution of the pixel. When performing a series of experiments, a standard lighting should be agreed upon and the CV should be adjusted to that agreed setting.

If the CV seems to be buggy even with a set lighting, the first place you should look is the parameters passed into the filtering functions, as the values ultimately control how much of a property is applied. The parameter tweaking should happen in the same order as the algorithm (blurring, then thresh, then contour, etc). If this still doesn't resolve issues, I have compiled a list of resources to begin looking at CV counting techniques.

http://www.pyimagesearch.com/
https://www.learnopencv.com/blob-detection-using-opencv-python-c/
http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html
https://www.youtube.com/watch?v=aEcBnD80nLg
http://www.femb.com.mx/blog/
http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html
http://docs.opencv.org/3.1.0/

Liquid Flow Relation (LFR)/muShroomMapper Software:

(Note: Shane McCormack will be continuing on with CIDAR lab as a Ph.D student. He will continue to develop his simplified microfluidic creation standard)

Connection to Neptune/Visualization: At the moment, the only connections to Neptune, the GUI for parametric design, has is the input LFR and UCF, and the output MINT file for Place and Route. In the future, I'd like to edit the graph of microfluidic components I make during LFR->MINT translation to make it compatible with a graph visualization tool. I've been suggested to work with D3. This visualization-compatible graph would then be passed to Neptune and the same visualization tool would be embedded into Neptune to preview the user's specification before the Place and Route occurs, which is somewhat time consuming dependent on the complexity of the microfluidic design.

Simplification of UCF User Input and Formatting: The JSON format of the UCF is somewhat complex for new users. I plan to work with another Ph.D student in CIDAR lab who is a microfluidics expert to simplify the information the user needs to submit to describe their microfluidic components. For example, instead of describing the dimensions of a droplet generator in MINT-style format, the user would instead input their desired droplet size. The formatting of this simplified UCF input could also be made graphical, and possibly integrated into Neptune, similar to the Vulcan Scheduler input format.

Automatic Channel Merging Standard in LFR: Similar to how if an output is defined twice, the output of the gate is split, I'd like overloaded inputs to a component to mean that the inputs are merged before entering the component. Currently, there is a component defined in the UCF that does just this, but making it automatic takes one more task off the user's shoulders.