* A high-level description of the problem
* A relevant background information to understand the problem and to get more information on the problem
* A description of your design in a form that people who aren’t familiar with the work can understand how the design was achieved. Imagine one of your classmateswas to use your project and extend it. What would they need to know about how you built your project
* A presentation of any results. This can include some metrics in terms of your design and how you improved it, results generated by your design, etc.
* A conclusion summarizing the project and what was presented in this document
* Pictures and/or movies of your system and design and how it works process
* Citations for which design elements are borrowed from others in and out of the class. Includes links to these users and highlighted code.

Manipulation of a Rubik’s Cube

Our goal was to create a Rubik’s out of LED’s with input controls. We originally planned on making a 3x3 cube with several inputs to manipulate the various faces of the cube as well as the cube as a whole. We intended to implement a function linking the various inputs to the lights with a function in VHDL on an FPGA. The 6 LED grids and necessary buttons would be connected physically and powered by a logic board, in this case an Altera DE2. We planned 19 inputs, 1 reset, 12 face rotations, and 6 cube rotations. Outputs were dependent on how the LEDs we chose received signals.

VHDL is a hardware language that is used devices like field programmable gate arrays (FPGA) to create circuits. This setup can be used to control hardware as well as run programs at the gate level. It is capable of implementing all common types of logic blocks, and can therefore be used to control anything that has a direct relationship between input and output. These implementations are made by writing code in VHDL and programming it directly to an FPGA. This makes the chip on the FPGA create the gate level circuit dictated by the code. FPGA’s like the DE2 have onboard clocks that send a signal in as well as switches and buttons. For outputs, onboard LED’s can be triggered and data can be sent out from pins to other devices.

The application of VHDL and an FPGA to the Rubik’s Cube involves programming the circuitry to trigger the RGB LEDs on the cube matrix to display the correct color and make the correct changes when user input’s updated the cube. This can be done by storing data for current color displays in registers and loading new values into those registers when certain changes are triggered to the cube. This means that each user input will set off a unique pattern of register shifts and loads in order to manipulate the cube accordingly.

We originally just planned to test the LEDs using an Arduino.

Fig 1. Shows our test bench.

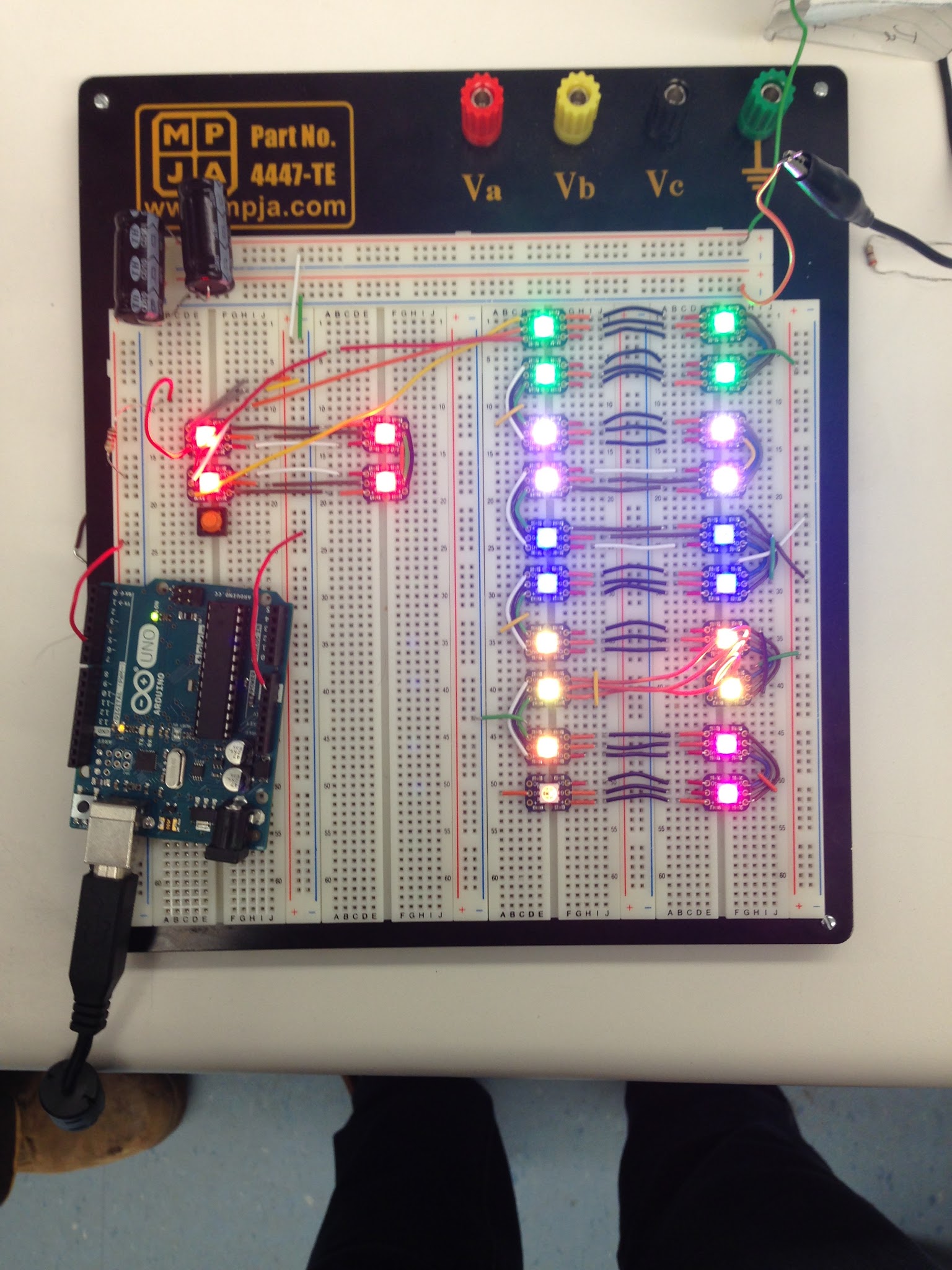
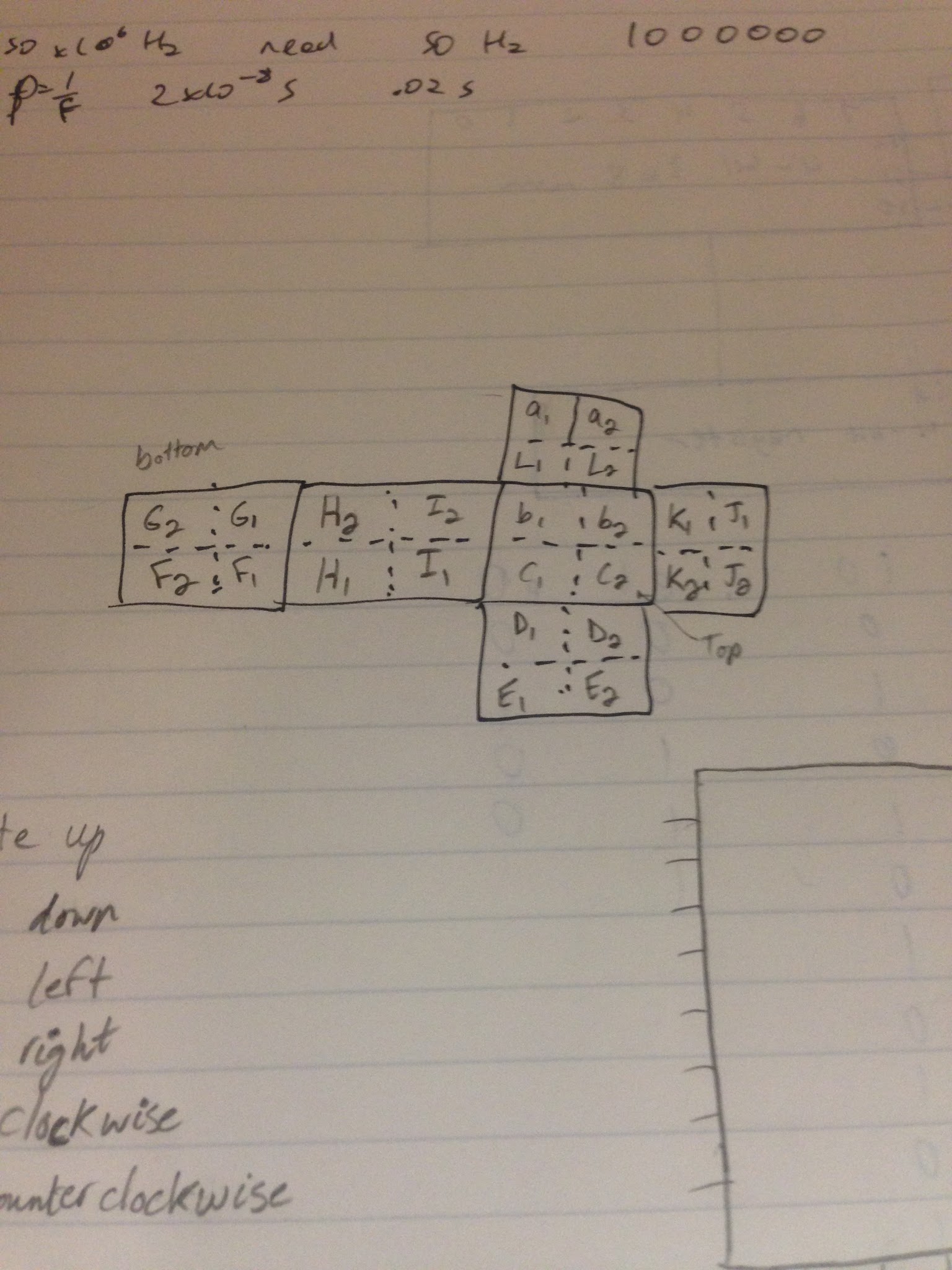
The LEDs are Neopixels from Adafruit. They can be chained together because data is sent through them sequentially in a data stream. We ended up scaling back to a 2x2x2 Rubik’s Cube due to time and budget constraints. Our design was achieved by mapping the cube and assigning a value to each of the 25 Neopixels in the cube. This was done by creating 6 matrices (1 per face) that were laid out relative to one another in such a way that we could actively manipulate each individual LED by just referencing its position in the matrix. This can be seen in the figure below. 

Fig 2. A birds eye view of the cube layout in an unfolded position.



After mapping out the cube, a test bench was assembled as seen in Figure 1. Immediately after this step, a pseudocode was constructed in order to get a general understanding of what kind of functionality we wanted our cube to have. These included rotate up, down, left, right, clockwise, and counterclockwise. As well, we outlined options to select which part of the cube that we wanted to apply each function to. These selections included choosing between, the front or back side, left or right half, and top or bottom half. Rather than implementing a separate third control with its own functionality to rotate the whole cube at once, we decided that it would be better to have our third control that would just select two sections together and perform the task by manipulating each side at the same time (a copy of the pseudo code is attached). Continuing, once we had an overall layout, we decided to try to assemble the 3 dimensional cube structure by taking 12 miniature breadboards and linking them together. However, we ran into significant issues when attempting to mount the LEDs to the boards. Each LED had 6 pin holes on it. These included “Power In”, “Power Out”, 2 grounds, and Signal In, Signal Out. Unfortunately, each pinhole was not very sensitive and needed phenomenal contact with its connecting wires in order to work properly. This proved to be the most significant challenge for us as assembly of the cube hindered achieving any functionality.

We tossed around a few ideas to overcome this challenge but ended up soldering wires to each of the LEDs to act as headers to improve reliability and contact consistency. Using actual headers would have saved us a lot of time in two ways. First, we would not have wasted so much time connecting wires to the LEDs in bread boards with limited success. Second, we would not have had to make our own pins to solder to the LEDs. Eventually, we did get all of them mounted onto the small breadboards and we put the 6 2x2 matrices on a cube to form the full prototype.

By the time all the LEDs on the prototype cube were functioning properly, we had run out of time to implement the function on an FPGA. Due to a blatant misunderstanding on our part, we attempted to implement the entire function using only the Arduino we were using to test the LEDs. It was our full intention to make use of control inputs via the FPGA to choose which operations that we wished to perform but the numerous issues that we encountered in the first phase alone prevented us from doing so. We were able to get some shifting to execute on the cube but we were unable to make it controlled with user input. We also could not have the correct functions for each of the manipulations we had planned.

Overall, the project was a tremendous learning experience. Regardless of all of the failures, we have taken much from this which will be applied to our lives. First, we have learned that while we should set our goals high, we should not try to be too ambitious. Setting an attainable goal would have been a good first step for us. We had not realized initially, but the level of difficulty of this project seemed to prove too much for us as we had invested 70+ man hours into the project and could not even achieve design functionality that we had originally hoped for. Additionally, we learned that it is important to do things right the first time. We would have had a good shot at making our project fully functional if we had not had to spend so much time getting the hardware to work at the most basic level. These experiences will be useful in future projects and work.

References:

<http://stackoverflow.com/questions/13295011/altering-the-values-of-array-elements-from-within-a-function>

<http://stackoverflow.com/questions/12307935/iterating-object-names-in-c>

<http://stackoverflow.com/questions/2342114/extracting-rgb-color-components-from-integer-value>

<https://learn.adafruit.com/adafruit-neopixel-uberguide/neomatrix-library>

Below is our Video link:

<https://youtu.be/06RE4fi0yCE>

We have uploaded a set of codes that we have pulled from and used to implement functionality into the cube: