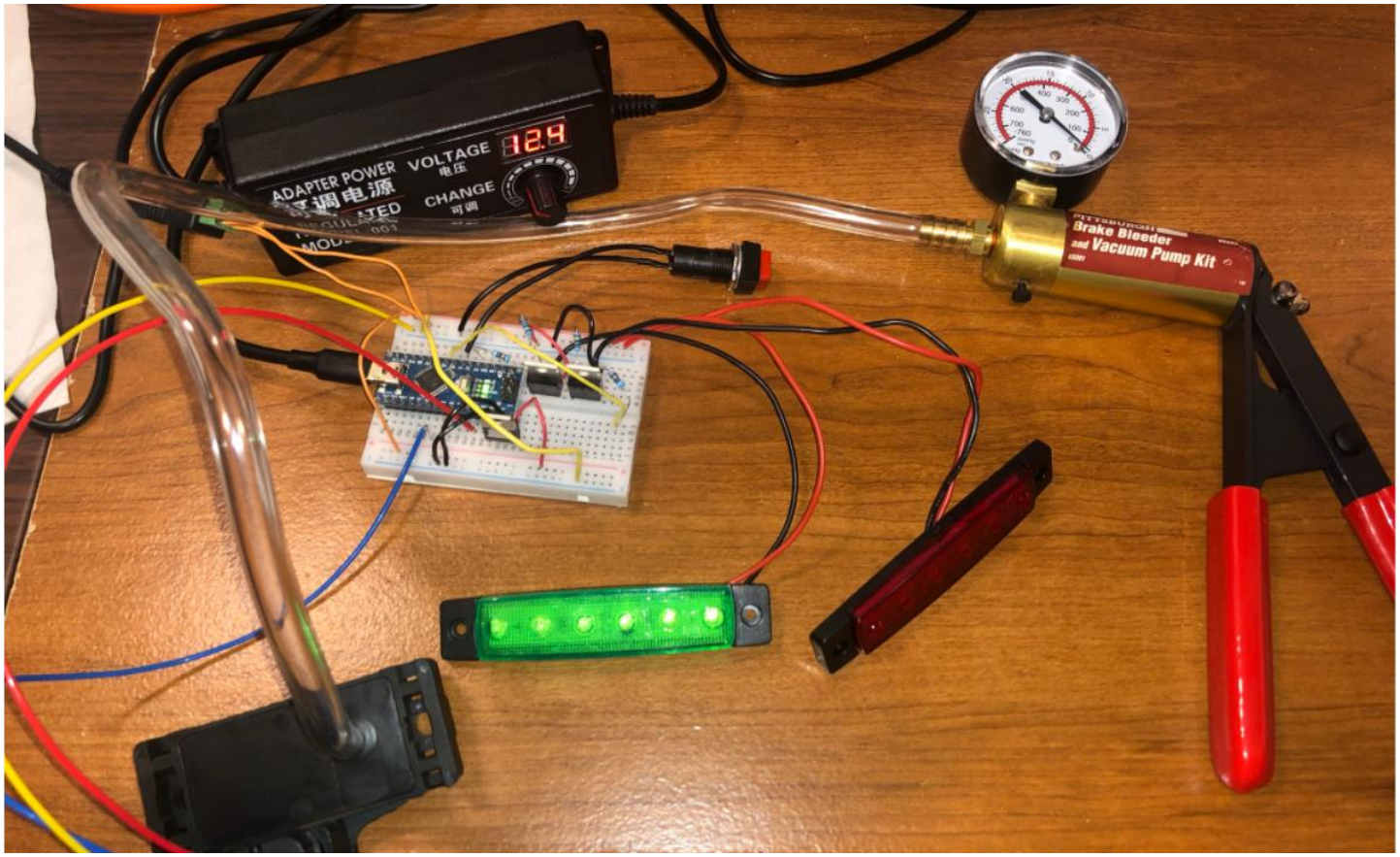
Final Project

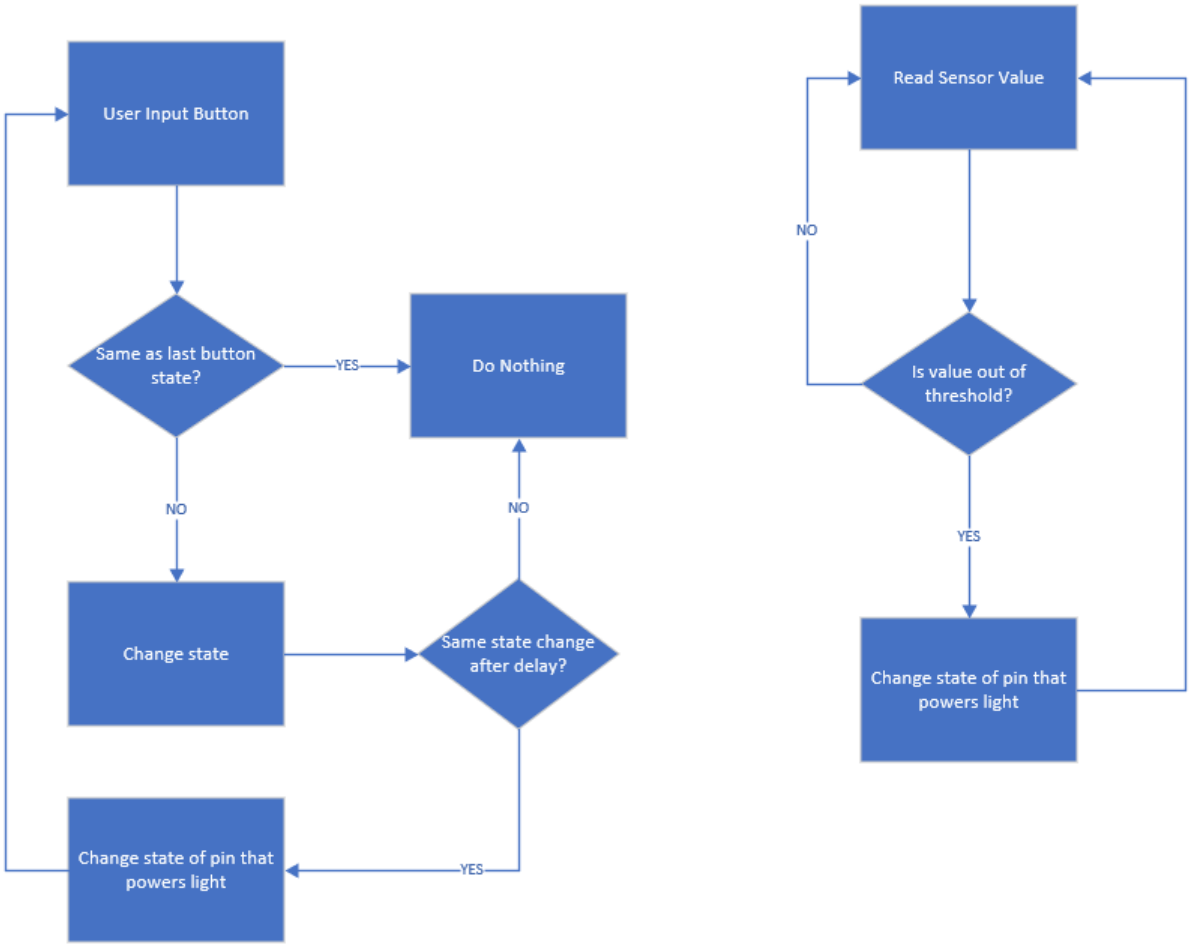
Ken Carr

The scope of my final project was a microcontroller that did 2 basic functions as a small system. It had 2 small state machines that were not linked together other than they cycle equally over time, switching back and forth between each state at lightning speed. The first state was a button being pushed was taken in and debounced. After being debounced a decision was made to change or not change state of the output. The second state machine was a sensor reading in analog data and causing an output change based on a defined threshold value. I picked this project because it was a simple representation of what I believe will happen at scale in the professional world.



(Figure 1: Image of working project)

After some jumping to far ahead in the process, I learned the first effective step was to layout the desired logic and output with a flowchart. The desired output was the result of 2 small state machines or independent functions which are represented in figure 2. The chart shows there is no connection between the 2 functions, they have an independent trigger that cause a state change of the system.

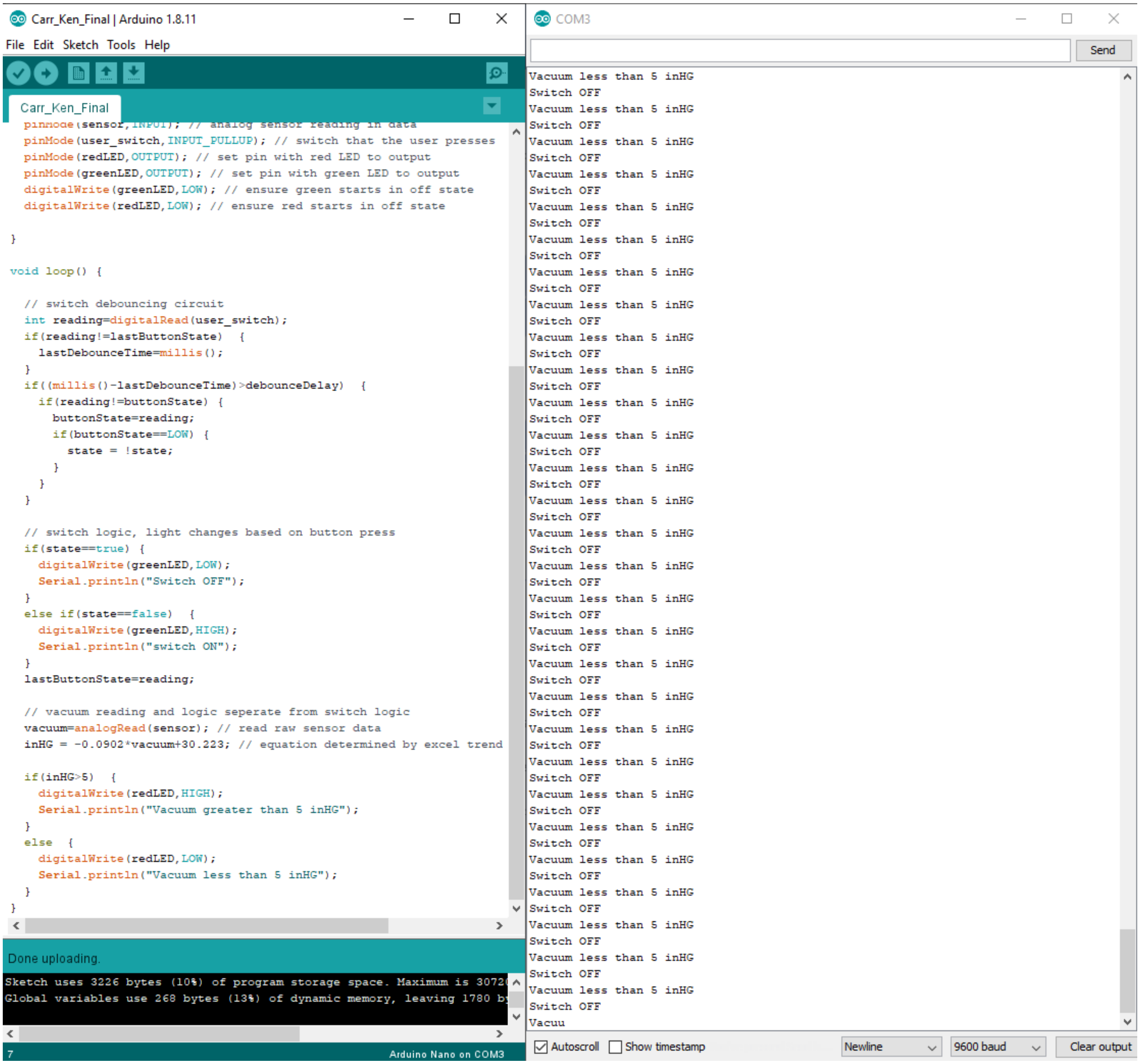


(Figure 2: Flowchart of parallel functions)

The next step was to write the code, this was much easier now that I had a clear picture of the input, output, and decisions that need to be made in between to make the state change. Below in figure 3 shows the code running with a serial monitor to do initial debug to make sure the states are changing as desired based on use input. There are 2 inputs in the system, the button and the MAP sensor. The serial monitor shows what state they are in which can be compared later to what is physically observed in the system to make sure it is all functioning as desired.

The language I selected was Arduino C. The selection was based on the large Arduino community for help and availability of controllers. I already had an Arduino Nano I was playing with so it was very available. The question that came up was in relation to embedded C for my major, what is the difference? After doing some digging, I found that there is a “Arduino.h” header file that basically adds the extra functions to make it easier to do certain functions.

Doing some further research, I learned that some basic functions like assigning a pin are simplified with the Arduino language but do take up more processing power and memory. With the program being so small it isn’t noticeable, but a larger program or smaller controller this may become critical to the design scope. If I was to take the project to the next level for optimization, I would write in the ports using the registers so it was directly accessed instead of through the simplified method with Arduino.



(Figure 3: Arduino code with serial monitor testing)

Last part of testing was putting the board together and testing it with lights to see if each state machine functioned as designed. The first learning that I had was the power up state causing undesired outputs. When I powered up, I wanted everything to be off, but the light kept turning on every time I pushed code or powered up. After debugging function, I found that I had a Boolean state opposite to what I needed it to be to perform as desired. After making the change it functioned as desired.

The next debug issue was the output of the state machine. It appeared to work at a quick test, but then I realized the state machines had switched outputs. The green and red light although changing state, were opposite to what I had codded in the software. This was a good lesson that it can be the small details that get you. Here it is just lights, but something larger it may have been a large dangerous motor. Confirming outputs before hooking things up is

critical to ensuring same operation of each state machine. Going forward I would think of using lights before hooking motors up just to functionally check if things are working in the sequence or output pin that I intended.

Lessons Learned:

There are 3 categories I really developed my learnings from. The first is in project and time management in respect to a dynamic schedule directly and indirectly project related. The second category is in code design and microcontrollers.

Project management over time isn't the easiest thing to manage especially with a full and dynamic schedule as mine. The full-time work and school with life balance is hard when hard deadlines come up. Setting realistic expectations and not getting caught up in the excitement of what can be done vs what needs be done to meet the minimal project requirements giving the most valuable resource constraint that money can't buy, which is time. There were instances where I quickly went from 50 hours work week to 90 overnight due to issues out of my control that resulted in some crazy travel while still trying to balance all my classes. The solution was look at the core of the scopes of both major projects and see where there is a parallel to try and make both projects 1 major effort instead of 2 separate linear efforts. With slight adjustments I believe I was able to meet the individual criteria of each course despite not being able to get back the time I lost from higher priorities with work. I really learned that not everything has to be linear in function and bridging efforts is a great way to accelerate projects. If this approach is done earlier then later as a reaction to survival of the semester, I may be able to learn much more much quicker by freeing up some time or getting ahead instead of behind.

It is a habit of digging right in and starting to write code without a flowchart or pseudo code. What I realized when doing this is I would miss critical decisions that would cause my code not to work. By mapping out what I wanted to achieve without the code gave me a clear picture that was much easier and less frustrating to execute. The debug stage was also much easier since I better thought out the logic with the not as obvious edge cases instead of trying to code off a quick idea that hasn't been flushed out. I also learned that using the timing of the clock cycle was a great way to debounce a circuit so it doesn't have to be done in hardware, and can be simply done in software.

The last step taught me a couple good lessons that were touched on up above. Debug doesn't just mean the code, but does the code functionally do as desired after it compiles in the IDE. When running it as if everything was fine, I realized there was a little detail I missed that could have gone unrecognized through all of the videos. Debug is finding a safe way to confirm that each state machine operates as desired. Doing my testing I have identified lights are a great way to see if function is as desired in a safe way. The first step I always see of getting a light to work seams silly at times, but I really appreciate how it could save time, money, if not lives by giving an extra level of diligent confidence that things are as they are scoped out to be.