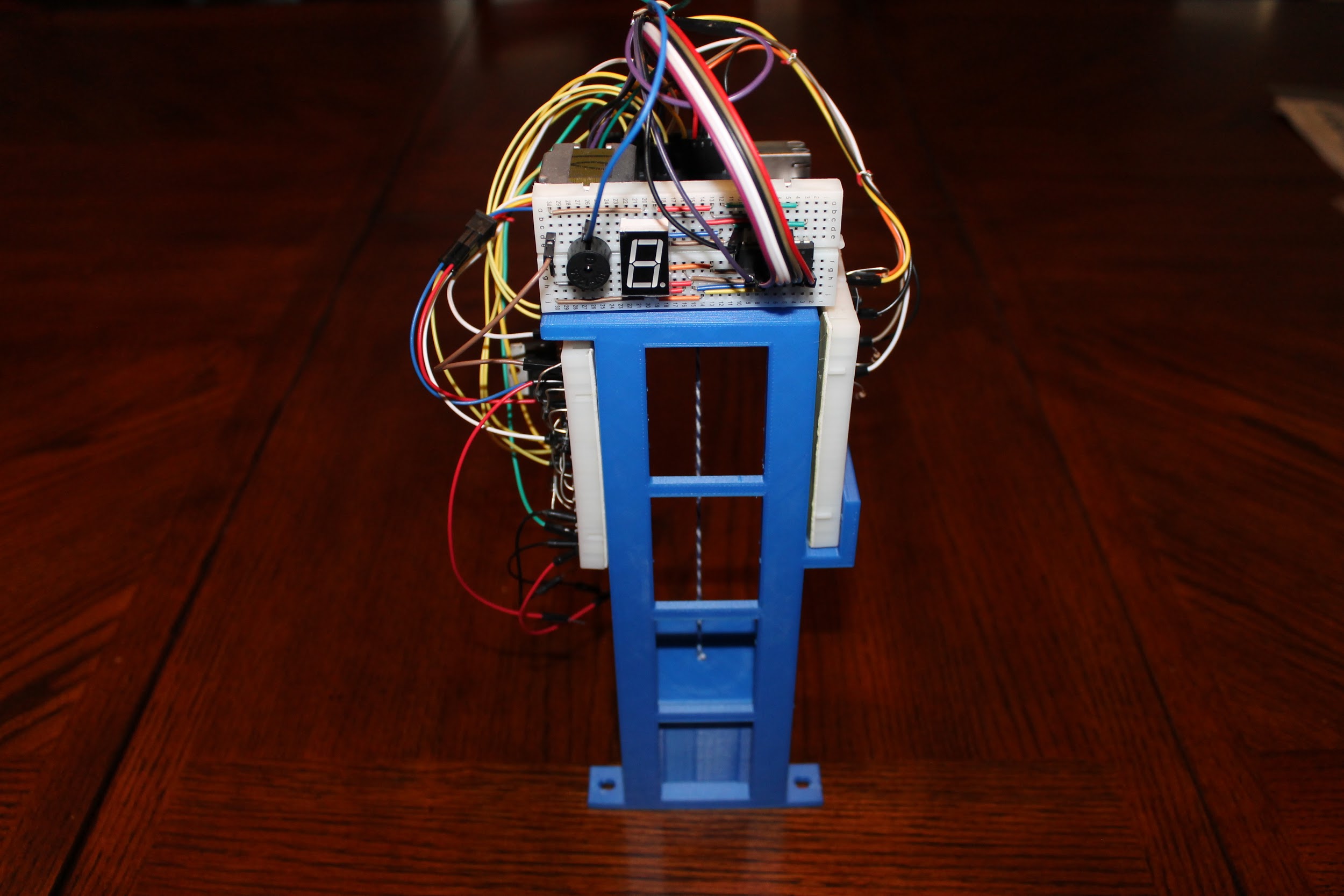
**EE 128 Mini Project Report**

**- Tiny Elevator -**



**Designed by: Gaurav Adlakha and**

**Kathryn Hammar**

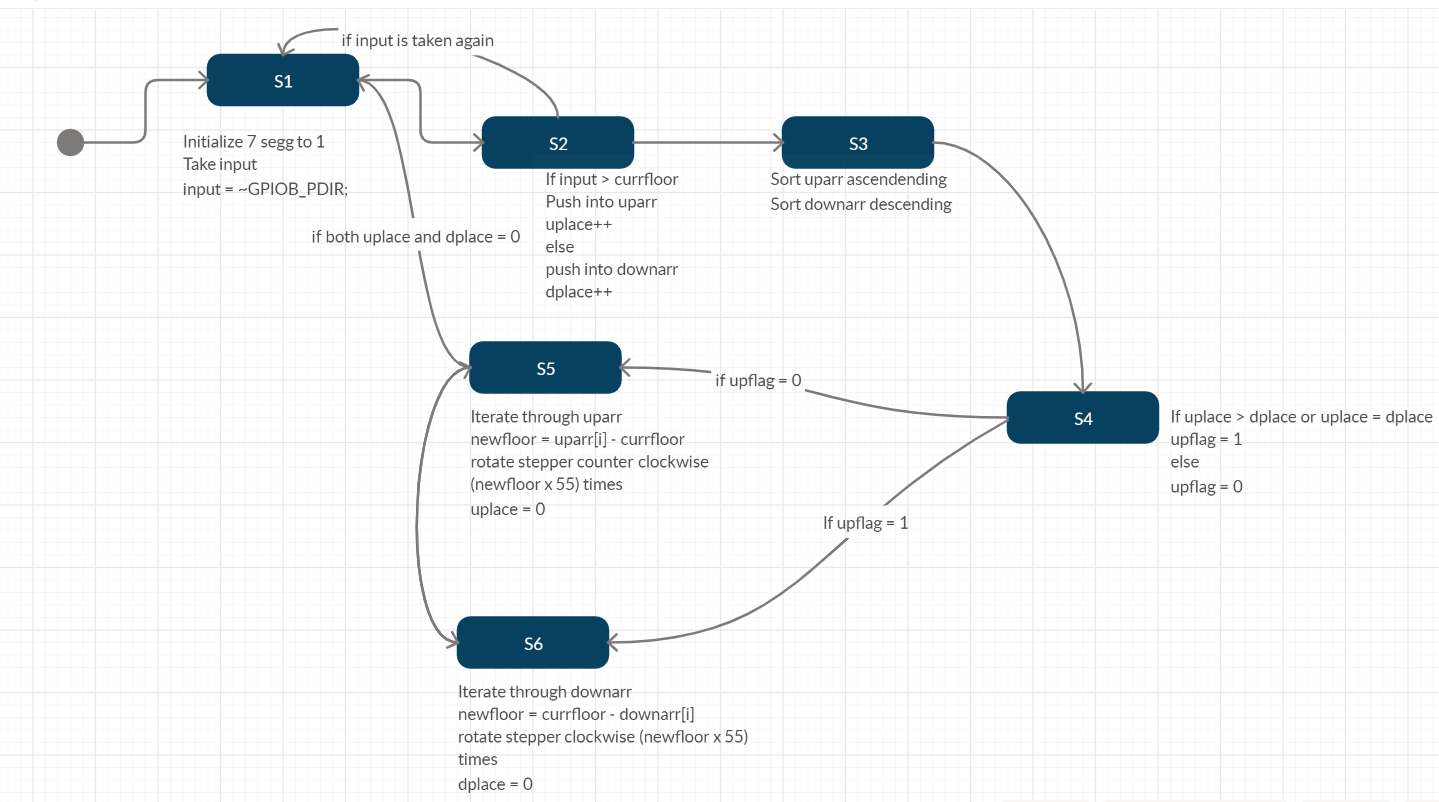
**-Section 021-**

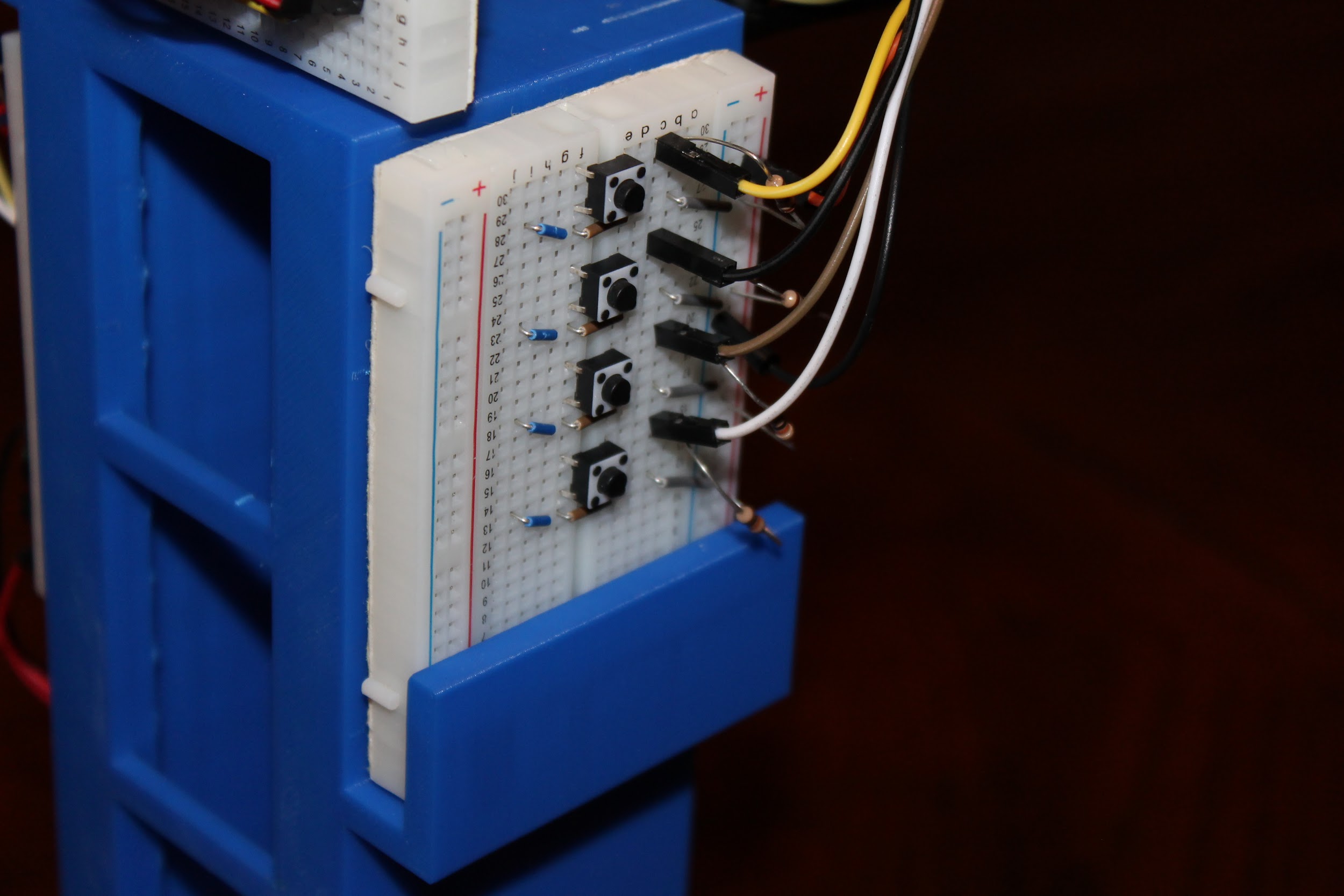
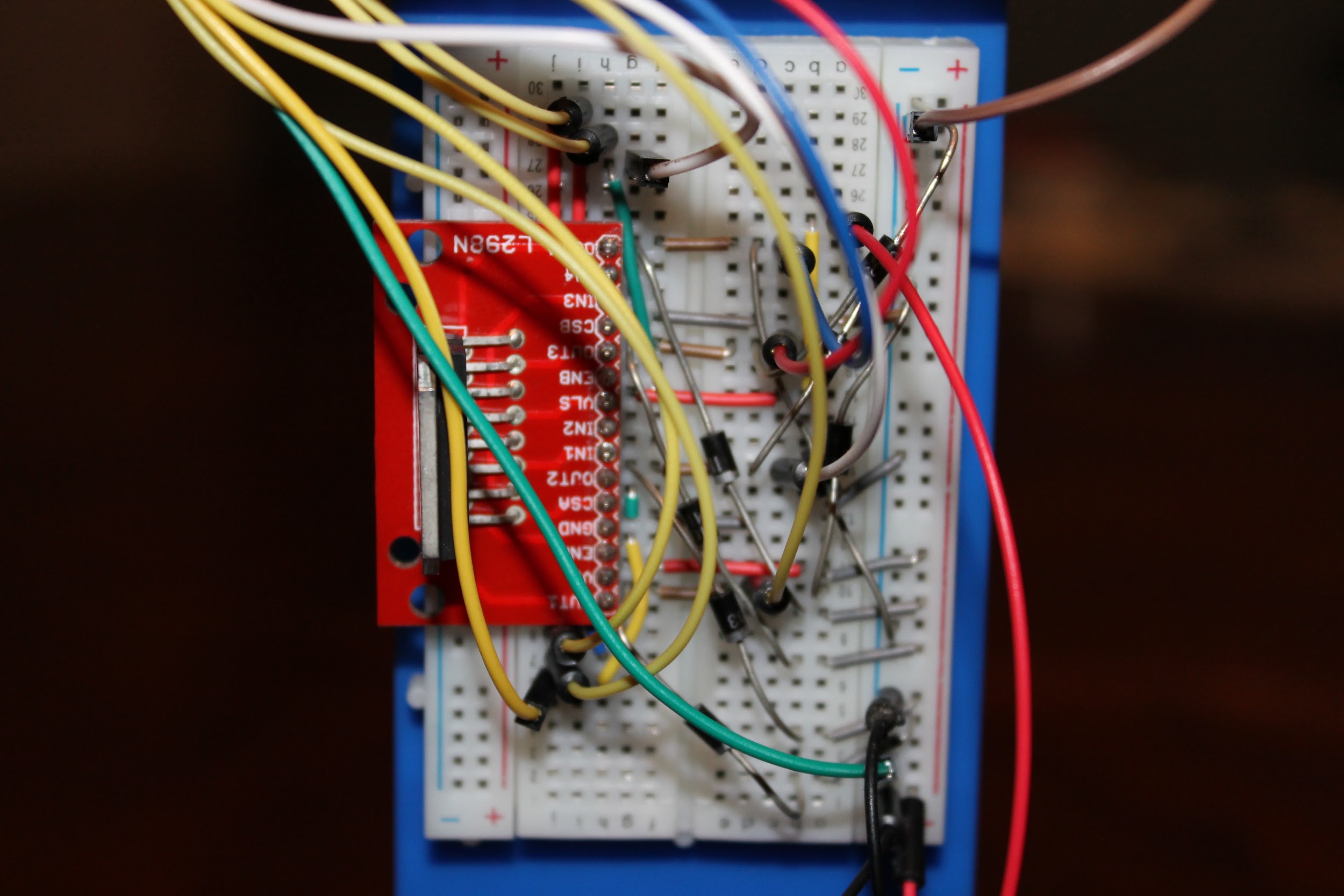
**Project Description**

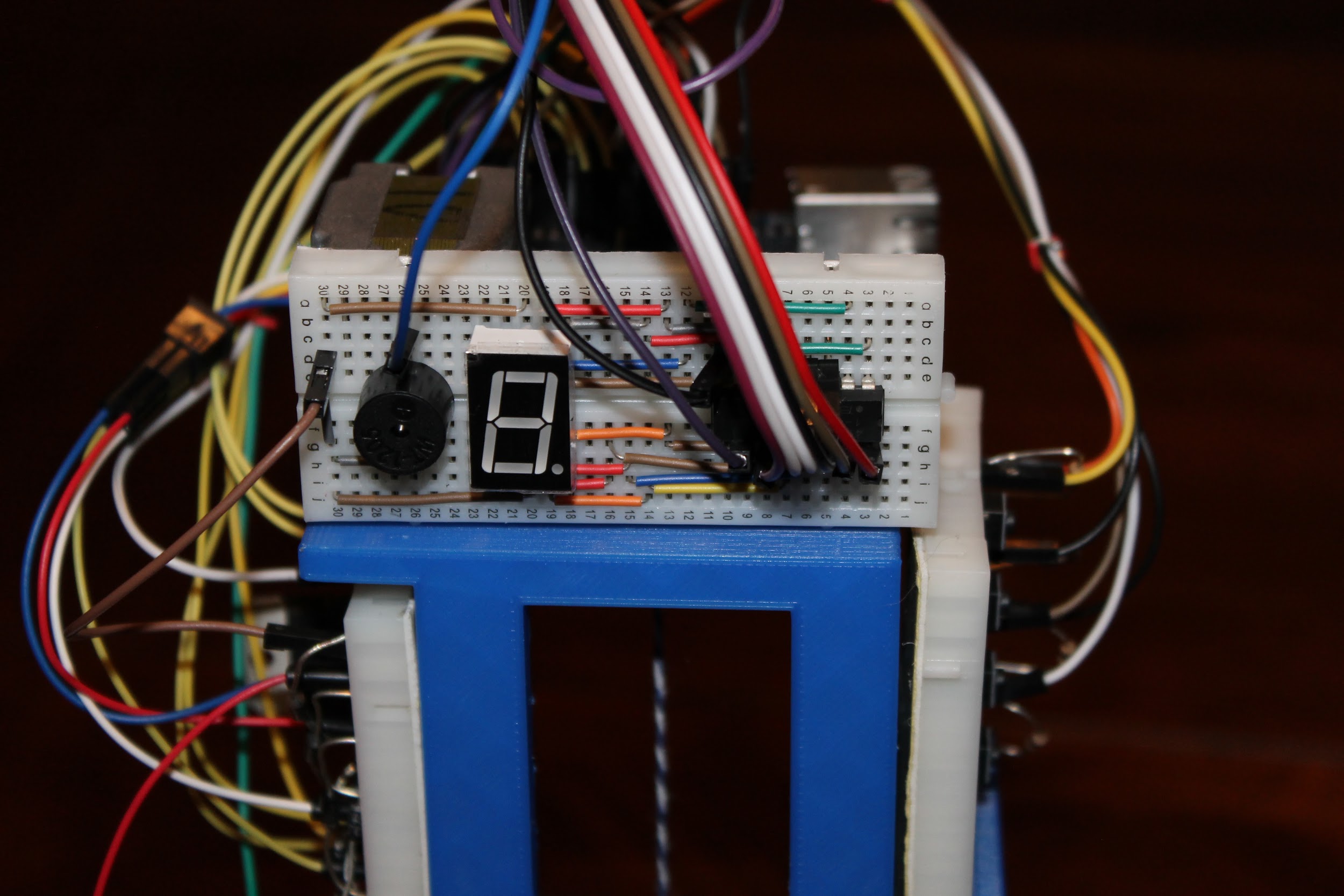
This project is based on a simple elevator system applied in a small scale. Both the elevator shaft and the elevator car were 3D printed. The elevator contains 4 floors, and can freely move between any of the floors with the press of a button. The elevator car is attached to a stepper motor, through a string, which allows for this movement. The distance between each respective floor is measured by one full revolution of the stepper motor. A 7 segment display is used to output which floor the car is located on, even while moving (it updates in real time). A small speaker is used to output a sound when a target floor has been reached. The system was also meant to be able to take multiple inputs at a time.

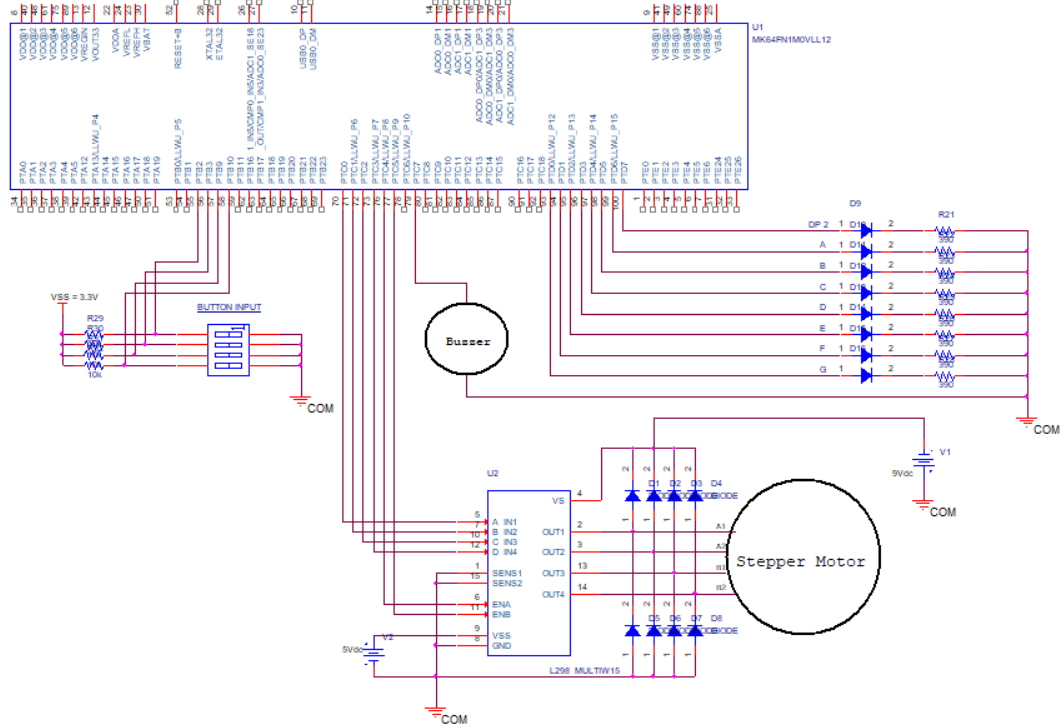
The idea and requirements for this project were to truly emulate a real elevator with proper use of input and practical use of output in order to enhance the user experience.

**System Design**



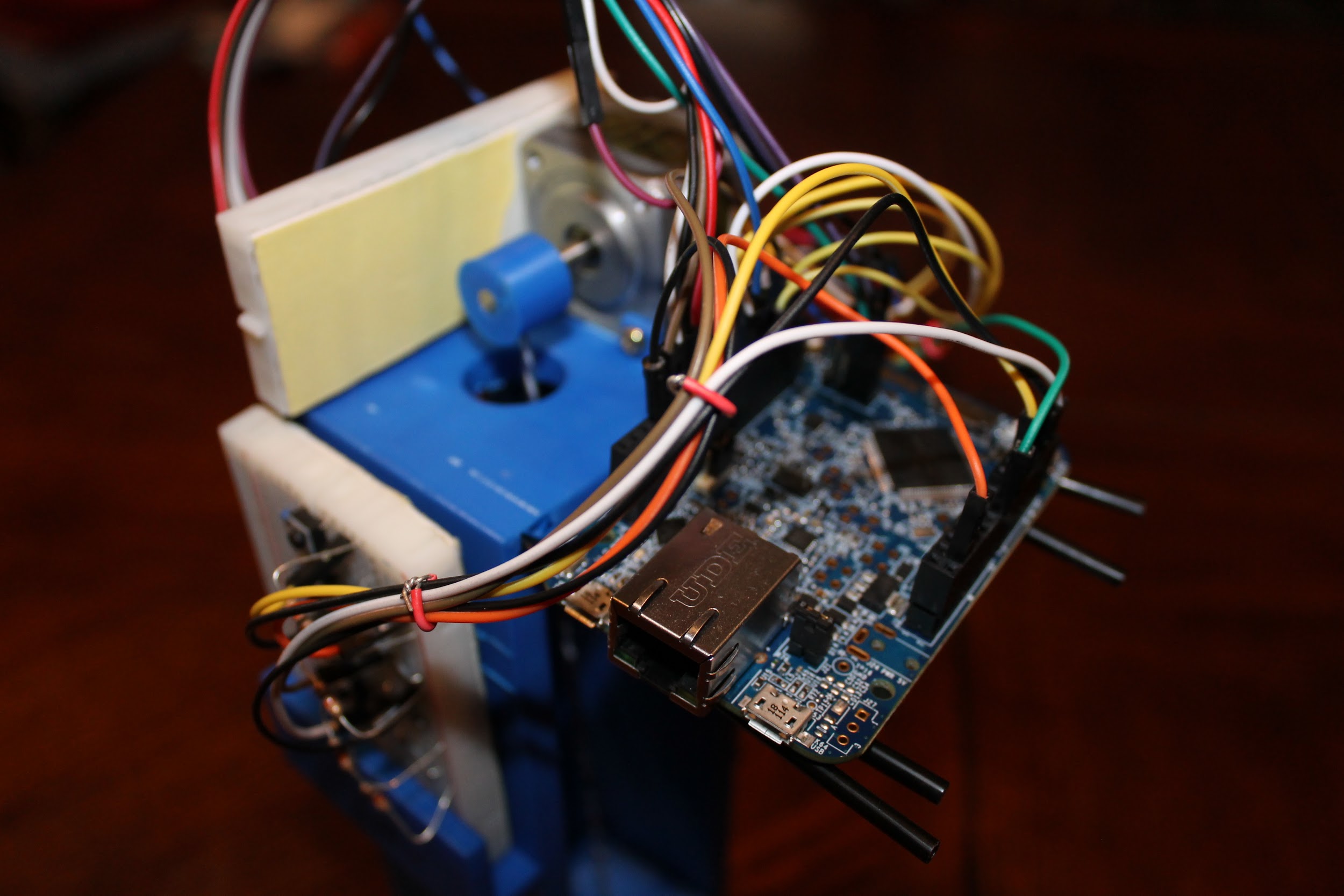
**Implementation Details**

**- Hardware Connection Photos -**

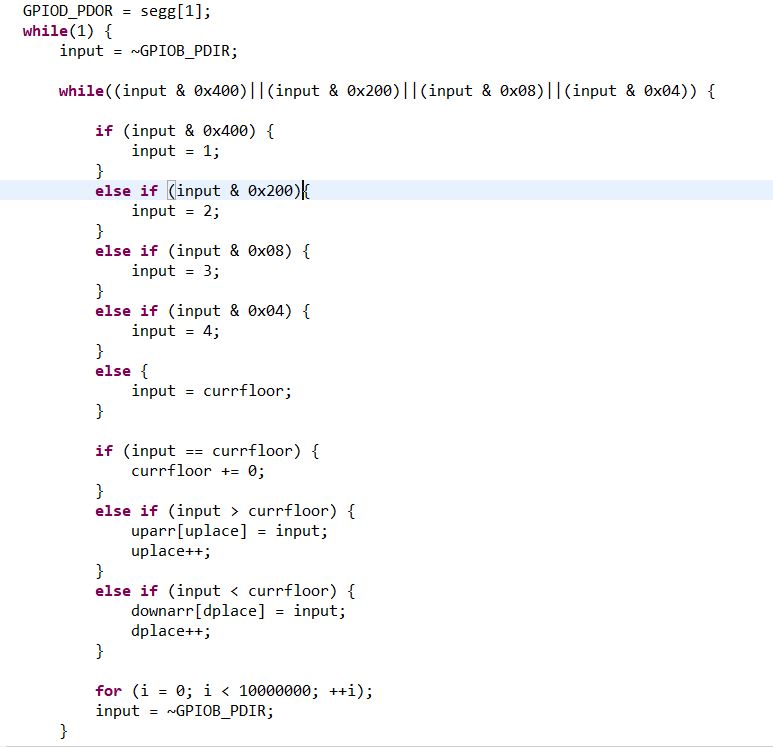


**- Hardware Schematic →**

**Hardware Connection Cont.**

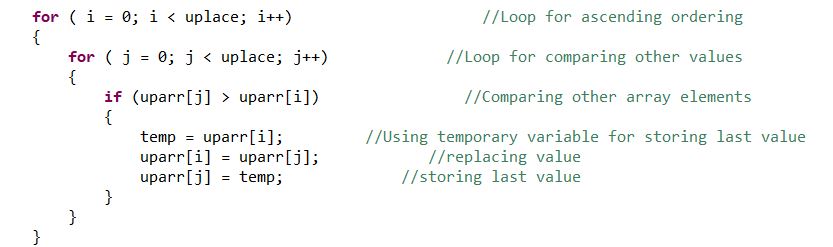


**Software Outline**

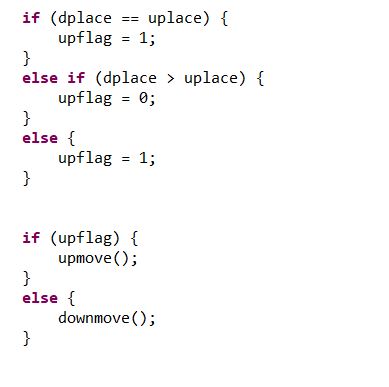
The following code outline includes sections that were aimed specifically toward the implementation of multiple outputs, a function that we were unable to fully complete.

This portion of our code checked for input on any of our wired buttons. Based on the input and the current floor saved, we push the floor value into the upwards movement array if larger or the downward movement array if smaller. An input equivalent to the current floor saved effectively does nothing. The uplace/dplace integer variables are used to keep track of the next free index within each array.

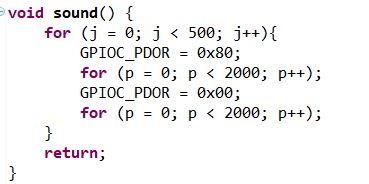
The final for loop is a delay before the code allows for input to be taken again. This part of the code is very inefficient and does not accomplish the goal of taking multiple inputs. Based on this implementation, a button would have to be pressed as soon as the for loop delay ended, which is a task that is nearly impossible to do consistently. In the end, the code still easily takes one input at a time and follows the rest of the logic.



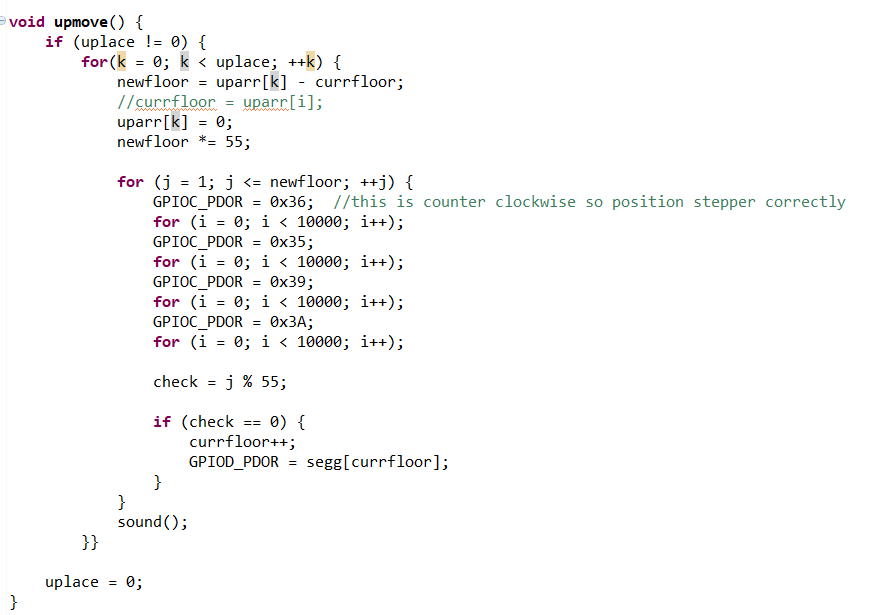
This block of our code is the logic used to sort our arrays. The idea was to efficiently hit each floor within our arrays. For example, if our input was floors 4 and 2 respectively, and our current floor was 1, we wanted the elevator to go to floor 2 then floor 4, regardless of the order of the inputs. The upwards array prioritizes lower floors over higher floors and the downwards array prioritizes higher floors above lower floors. Shown above is the upwards array sort logic. The upward array is sorted in ascending order based on the idea stated earlier. The downward array sort is done in descending order and is implemented extremely similarly, with some simple logic changes within the if statement.



This block of code essentially determines which direction the elevator will move after all inputs have been taken. The array with the most inputs will take priority, otherwise if both are equal size, upwards is prioritized. Since dplace and uplace correspond to each array’s index, they can also be used to see which has more inputs based on the value of each variable.



This portion of our code is a function designed to output sound. We simply send input through a single pin to our speaker and the frequency is dependent on the edge values within the two inner for loops. This code is simply a pulse width modulation implementation with 50% duty cycle. The outer for loop determines how long the sound is output for. Adjusting the inner delay values would result in a different frequency or differently pitched sound.

This block of code is where the movement of the motor takes place. The first thing to be done is to begin iterating through the movement functions corresponding array. Based on the value of the first index, we find out the amount of floors we need to traverse through to reach our destination by either subtracting the current floor from our target floor (in the case of upward movement) or subtracting our target floor from the current floor (in the case of downward movement). This value (newfloor) is then multiplied by 55. 

The value of 55 was unique as it was the number of steps required by our motor to make a full revolution. Thus multiplying the newfloor value by 55 will rotate the stepper the current amount until the target floor is reached. This is done in the second for loop, where we used the same code we used in Lab 5. Counter clockwise for upwards and clockwise for downwards.

Within the second for loop, we also check to see if our incrementer has reached a multiple of 55, in which case we increment or decrement the current floor depending on direction, and update the 7 segment display. After the second loop has finished its iteration, the sound function is called to show a target floor has been reached. If all indices of the array have been checked and all movement is complete, we exit the first for loop and set the index variable (uplace or dplace) to 0, so that old values can be overwritten and the code can correctly follow its logic.

**Evaluation**

The original requirements for our project involved using photoresistors with LED’s to determine if a floor was reached, we were not able to implement this aspect of the project due to our struggle with both the logic of multiple photoresistors (ADC inputs) and the physical/mechanical integration of such sensors into our system. Instead of light sensing we decided to use the step measurements as the constraint for floor movement. Another requirement was multiple input functionality, this is another aspect that we were unable to recreate in our system, this problem will be discussed later in the report. We added required features for our system including a 7 segment display to display the floor movement of the elevator and a speaker for auditory indication of a floor being reached.

For the 7 segment display, we implemented the code for it and simply tested whether or not it correctly changed the display when a floor was passed and/or reached. We achieved behavior that was suitably synchronized with the physical movement of the elevator, this can be seen in the demo video.

In order to implement the speaker, we tested various duty cycle values to hear the pitch and volume of each frequency, and made the decision for which frequency to use based on desirability and similarity to what you might hear from and actual elevator. We also made iterative changes to the outer for loop value in the code to find a desirable sound length.

The bulk of our motor testing came from iteratively observing how far the elevator moved based on different multiplier values in the for loops within the movement functions. With some mathematical calculations, we were able to narrow down how many steps our specific stepper motor took to make a whole revolution and how much of that revolution was necessary to move the elevator from floor to floor on our physical design.

For all of these tests and every other debugging situation that we came across, the step through function in the Kinetis debugger was what enabled us to find many solutions along with desirable implementation values for our outputs.

Demo Video: <https://www.youtube.com/watch?v=WMzIkT1-Ejg>

**Discussion**

The main implementation issue that we faced was the addition of multiple input functionality. As the code was being developed we went step by step trying to implement the more basic functionality first and then working on adding in more complex behavior after. With the limited amount of time that we had to create this project, we were not able to draft and debug the code for this feature quick enough to showcase it in the final design. We did, however, use a method of implementation within the final source code that would be adaptable to multiple inputs. The use of sorted arrays for the floors/direction make it so that if we were able to add the ability to accept multiple inputs at a time then the system for sorting those inputs and converting them to a physical response would already be complete.

Another challenge that we faced was in the use of photoresistors. This issue has been mentioned earlier in the report, we worked around this problem by using step measurements to decide where to stop for each floor instead of light sensing.

The main two limitations of our design correspond with the aforementioned challenges, the ability of our elevator to only accept one input at a time may be useful in a private penthouse elevator but, in the real world, that is considered a large limitation within the system. Also, due to variation in each floors height, using light sensors and LEDs would have led to a much more precise floor arrival location.

Additional improvement would be found in other functionalities that would enhance our project’s resemblance of its real life counterpart. This includes the implementation of call buttons which could be on each floor calling the elevator to whichever floor the button is pressed on. These would simulate the outer buttons outside of the elevator on the wall of the building. In conjunction with the call buttons, having LEDs corresponding to elevator direction would also serve to make our elevator more realistically functional. A final improvement that we began to design but were not able to complete was that of a fire sensor, this would use a temperature sensor to sense the heat (indicating flames) and shut down the elevator after bringing it to the first floor. This hard shutdown could be reset by a hypothetical firefighter pressing a button implemented to resume normal operation.

**Roles and Responsibilities of Group Members**

The role of each group member was separated cleanly between hardware and software. Kathryn, an electrical engineering student who has experience with computer automated design (CAD) and 3D printing, was in charge of creating and printing the design for the elevator along with assembling and debugging the various breadboard circuits of the system in an organized and efficient way. She utilized Autodesk Inventor for the CAD drawing and supplies from both the EE 128 lab kit and the EE 120B lab kit to assemble the finished project. Gaurav, a computer engineering student, focussed on writing the code for the project based on the desired implementation for an elevator system. Using the labs and lectures as a guide, he was able to develop the system’s software to cleanly interact with the various hardware inputs and outputs. He drafted the code using Kinetis Design Studio.

While developing the hardware and software aspects of the system, we worked together to debug any issues that arose in the interface between hardware and software. This collaboration led to us both having a deeper understanding of all aspects of the project.

**Conclusion**

The objective of this project was to implement a system with real life applications while learning how to develop designs implemented with a microcontroller and peripheral devices. Throughout the design process for this project, we were able to explore the various aspects necessary for creating a functioning system. We saw how important it is to design both our circuits and code to be organized for the sake of debugging. As we observed the necessary elements of a real life system, though we may not have been able to emulate them all, we gained a great appreciation for those who create such systems in the industry today and great hope for what we will be able to create in the future.