Project 3.2.4 Machine Control Design

The Gliding Ascenders

Kaitlyn Huynh, Anvitha Kachinthaya, Christy Koh, Loic Scomparin
Principles of Engineering 6°
April 20, 2017

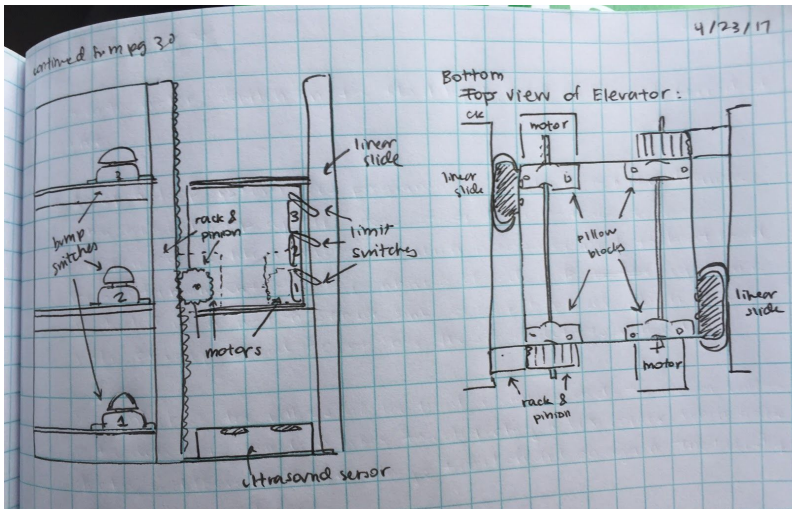
Table of Contents

Design Brief

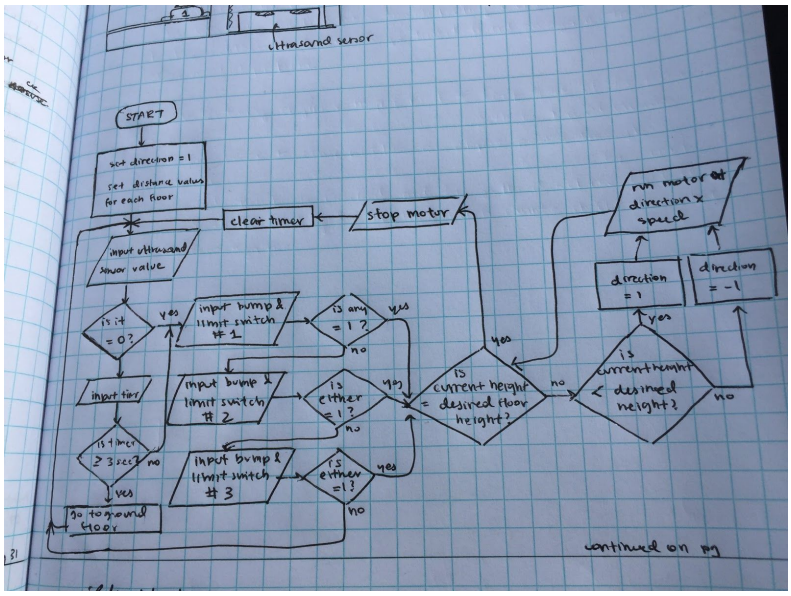
Client	Ascend Elevators
Designers	Kaitlyn Huynh, Anvitha Kachinthaya, Christy Koh, Loic Scomparin
Problem Statement	The problem we need to address is that there is no way for people to go between three floors quickly and safely in a building.
Design Statement	We will design, construct, and program a prototype of an elevator system that can transport people between three floors based on input from both inside the elevator and outside the elevator. The system will also include a built-in safety feature to rest on the ground floor.
Constraints	● Use VEX materials and program in RobotC ● Limited number of sensors to be used ● Elevator must go between three floors in any combination ● Elevator must have three switches ● Each floor must have a call button ● Built-in safety mechanism so that the elevator normally rests on the ground floor and returns to the ground floor after a user-defined period of nonuse
Deliverables	● Control system (RobotC program) and prototype ● Gantt chart and team responsibilities form ● Documentation due **May 4, 2017** ● Presentation on **May 5, 2017**

Brainstorming

Design 1: Rack and Pinion System



CK 04/24/2017

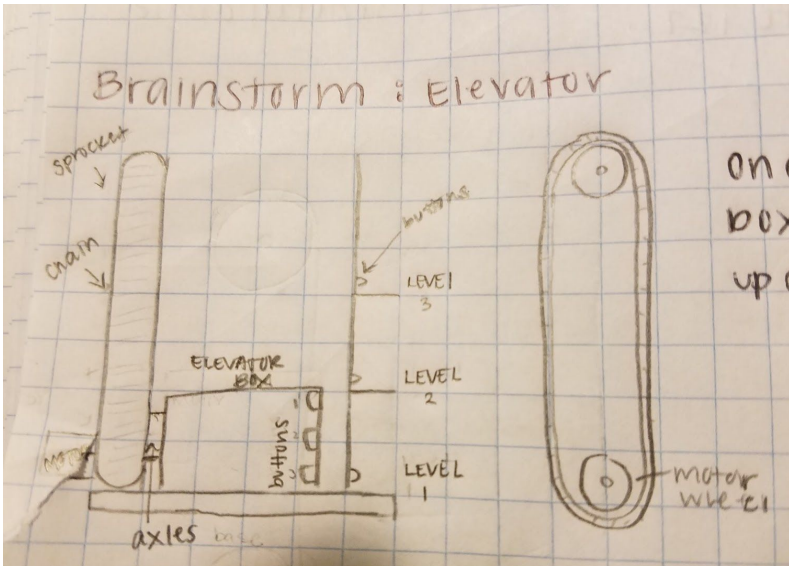


CK 04/24/2017

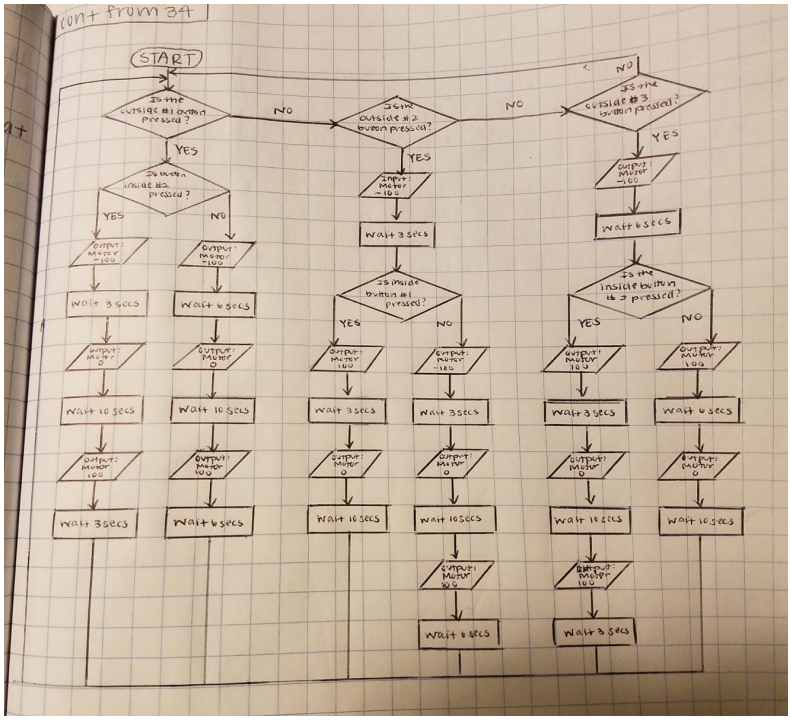
This design allows the elevator to travel up and down by rotating two motors. These motors are placed on opposite sides of the elevator. There are also two linear slides on either side of the elevator in order to guide it. An ultrasound sensor measures the position of the elevator, and there are call buttons on each floor as well as in the elevator.

The program for this elevator design first assigns some constants for the heights of the floors and creates a variable for the direction the elevator will travel. The main task begins by checking whether a timer, started automatically at the beginning of the program, has reached the maximum inactive period. If the time has not expired or the elevator is already at ground floor, the program will check the button inputs to see if the elevator has been called. If so, the current height will be compared to the destination height and the motors will drive the elevator to go up or down. Once the destination level is reached, the motor is stopped and the inactivity timer is reset. The program then loops back to the beginning by checking for inactivity.

Design 2: Traction Drum Pulley System



KH 4/24/17

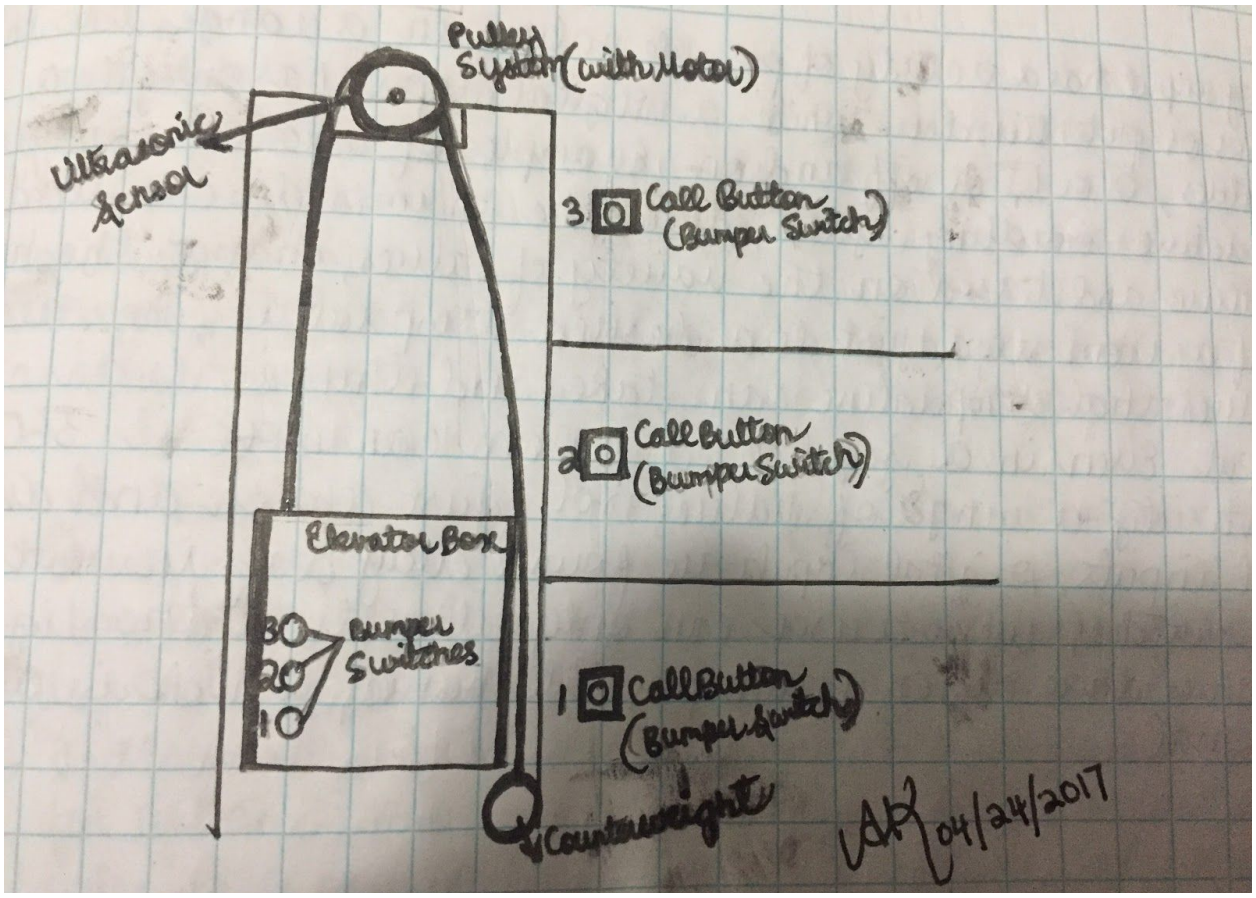


KH 4/24/17

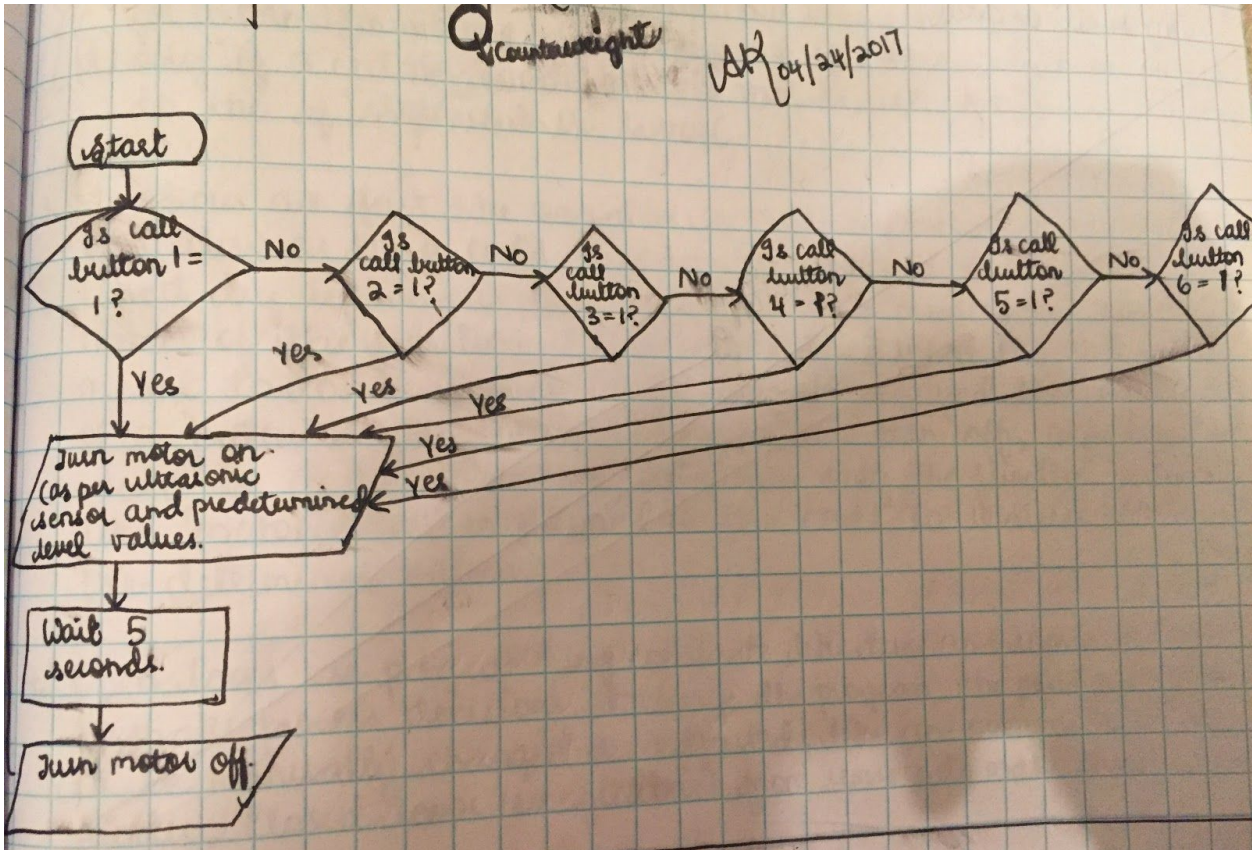
This design uses chain and sprocket and one motor. The motor will be connected to the base sprocket and the elevator box will be connected to one side of the chain so it will go up and down. There will be 3 switches on the outside and 3 switches in the inside of the elevator. Once the elevator picks up and drop off, it will return to the ground floor.

For the program, the elevator will begin at the ground level and then start with an outside button being pressed. The distance from level 1 to the other floors is measured in seconds, for example, the time to get to level 1 to level 2 is 3 seconds. So the motor will be turned on for a few seconds then stops at the level to wait for the passengers to enter into the elevator. It will then wait for one of the inside buttons to be pressed. Then, the motor will turn on for certain seconds to reach the level. After it drops off, the elevator will return to the ground level.

Design 3: Lifting Drum Pulley System

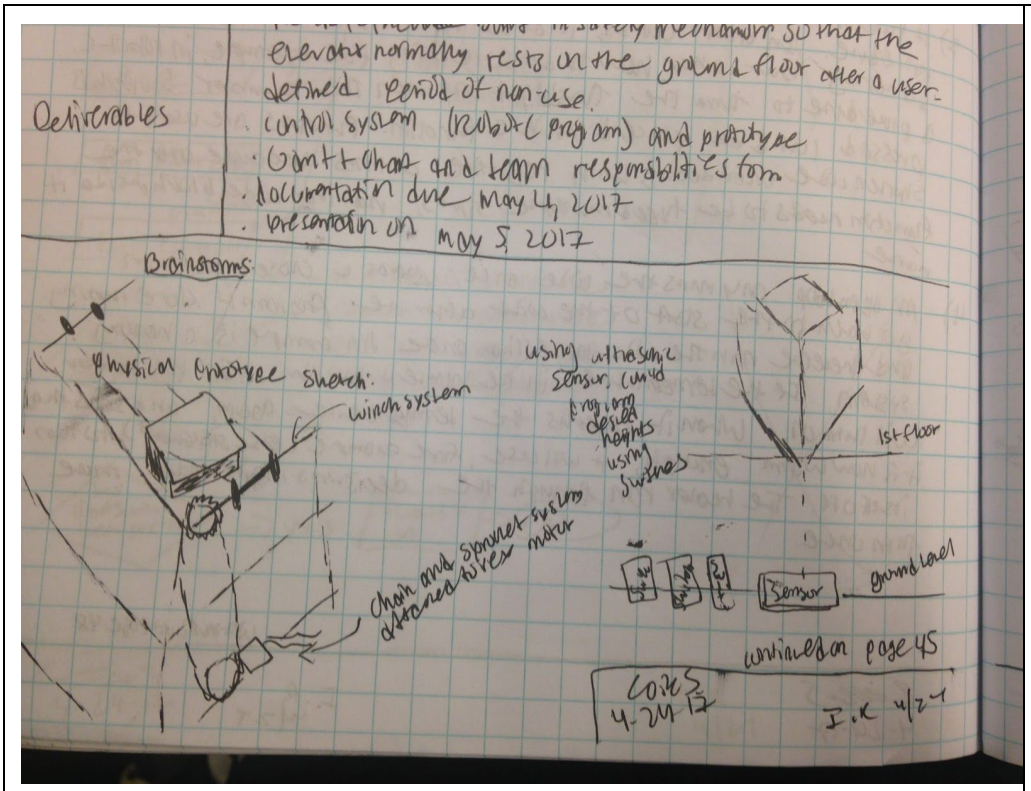
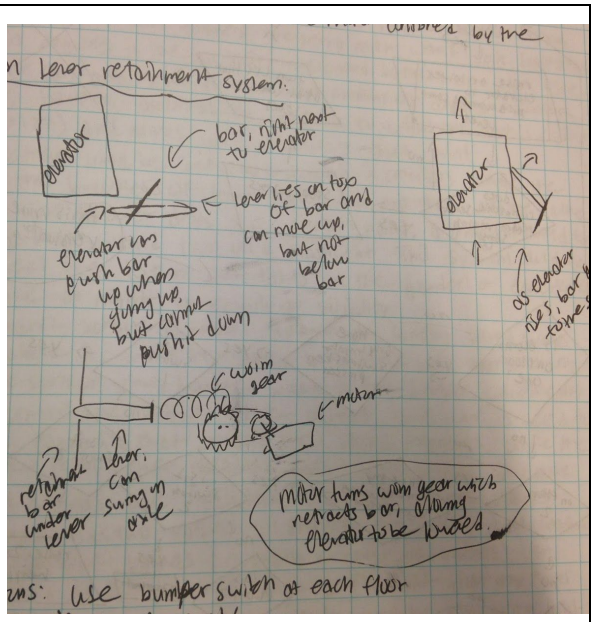


This design allows the elevator to move up and down using a pulley system with a counterweight to keep the system balanced and more stable. It requires one or two motors at the top, to turn the pulleys that hold the elevator. The system also uses six bump switches, as call buttons and within the elevator. An ultrasonic sensor mounted at the top, at the level of the pulley, measures the distance till the elevator box roof to determine the distance required to travel. It is attached above the elevator to allow the elevator to sit at the ground floor evenly.



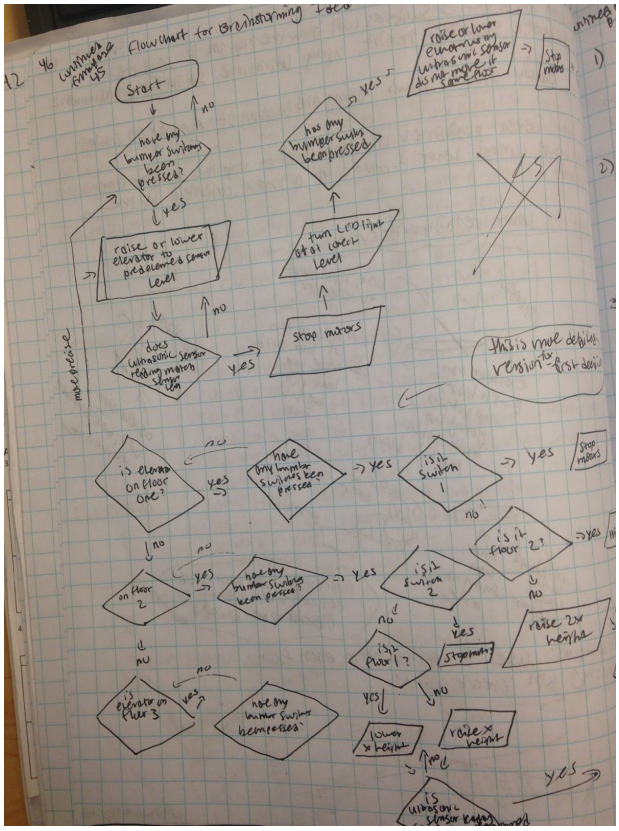
This program would constantly check the values of the touch sensors, and on finding them to be pressed, would move the motor accordingly, as the speed times the time needed to get to the required floor. After five seconds, the program would go back to checking the buttons, and repeat the process without end. There were many problems with this solution though, such as the differences between going from a higher floor to a lower floor and vice versa, which were not fully implemented in this brainstorm.

Design 4: Dual-Winch System

<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>This is the winch system and elevator shaft design. The building as a whole is essentially cubic, with the elevator located at a corner, with the ultrasonic sensor and first bump switch (used to call the elevator to that floor) located underneath.</td>
<td>This is a physical catchment system that could be added into the design in order to prevent the elevator from falling down after it has reached a certain level. Since the elevator is supposed to return to the ground floor after a certain period of time, the motor driving the worm gear could be programmed to retract the gear and let the elevator descend.</td>
</tr>
</table>

This design utilizes a winch system, composed of two winches driven by VEX motors, to raise and lower the elevator along the shaft. The winches would be positioned on opposite sides of the elevator. The winches themselves would be connected to the motors via chain and sprocket systems, as the motors would be towards the bottom of the building and the winches on top. An ultrasonic sensor, positioned at the bottom of the building, would accurately measure the height of the elevator. Bump switches would be positioned on each floor and programmed to send the elevator at their height when pressed, meaning that if the elevator is already on the current floor it will not move up or down. Lastly, a lever system would be used as a safety catch

for the elevator. A level positioned on top on a horizontal bar would extend out under the elevator to prevent downwards movement, but could then be retracted using a third motor and a worm gear.



Decision Matrix

Lifting Mechanism

Stability was determined by evaluating which design resulted in the least unnecessary movement of the elevator. We evaluated accuracy by how closely a motor could reach the desired level, minimizing error. Speed was evaluated by how fast an elevator design could theoretically move. Load and strength was determined by exploring how additional weight in the elevator would affect its performance. We evaluated flexibility by thinking about how easily the elevator system could be adapted and modified.

Scale: 1 - worst, 4 - best

Idea	Stability	Accuracy	Speed	Load/ Strength	Flexibility	Total
Design 1: Rack and Pinion	4	4	3	3	4	18
Design 2: Pulley System II (Traction Drum)	3	3	4	4	3	17
Design 3: Pulley System (Lifting Drum)	1	2	2	1	1	7
Design 4: Dual-Winch System	2	1	1	2	2	8

X Eugene Chou 04/25/2017

Floor Level Sensor

Accuracy was determined by how well the sensor could measure the heights of the different floors. Ease of programming was determined based on how easily the proposed

programs for the sensor combinations could be programmed and understood. We ranked the sensor combinations in resistance to damage by discussing how easy it would be for a sensor to be broken.

Scale: 1 - worst, 3 - best

Sensors	Accuracy	Programming Ease	Resistance to Damage	Total
Ultrasonic	2	3	3	9
Limit Switches	3	1	1	5
Encoder	1	1	3	5

Call Button & Elevator Button Sensors

We determined ease of use by how simple it would be to press the button combinations based on where they are placed (elevator or respective floor). Space was evaluated by how much volume the buttons take up. Lastly, resistance to damage was determined by how easily a sensor could be broken.

Scale: 1 - worst, 3 - best

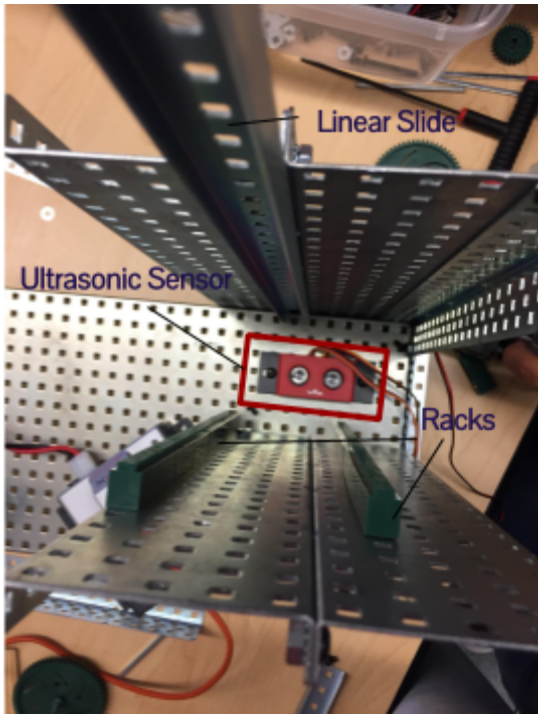
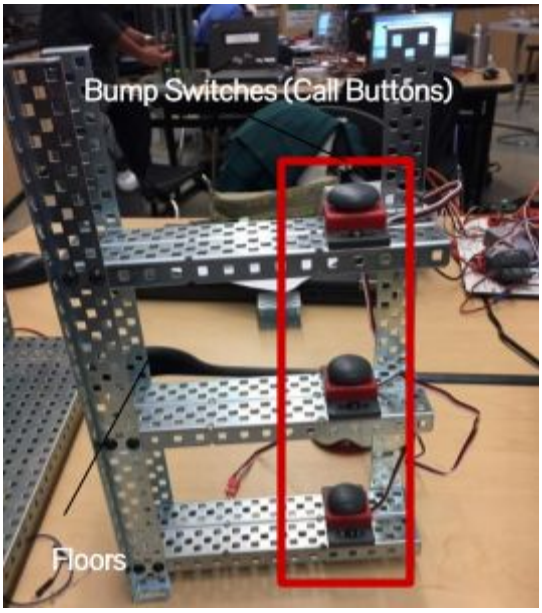
Sensors	Ease of Use	Space	Resistance to Damage	Total
Limit Switches for Elevator, Bump Switches for Call	3	2	2	7
All Bump Switches	1	1	3	5
All Limit Switches	2	3	1	6

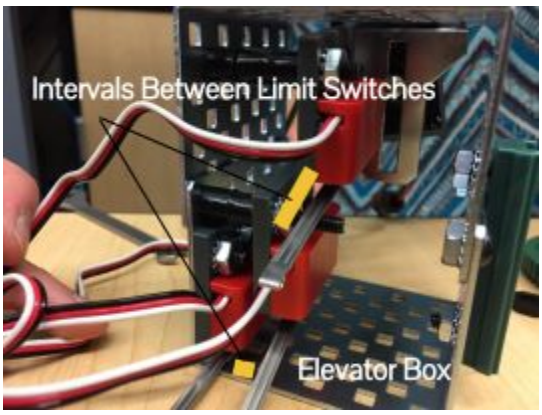
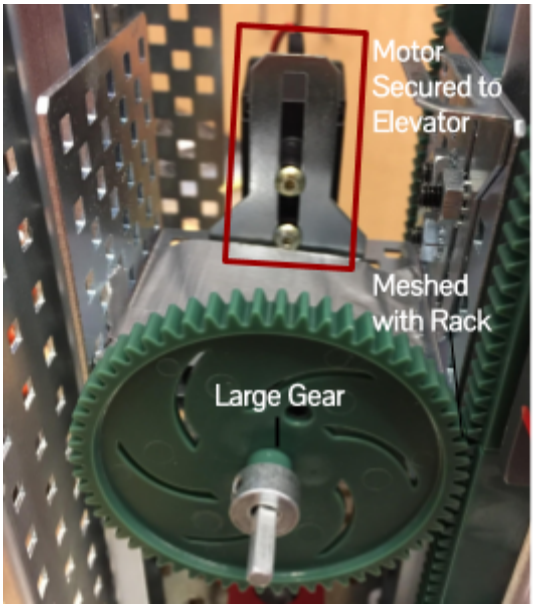
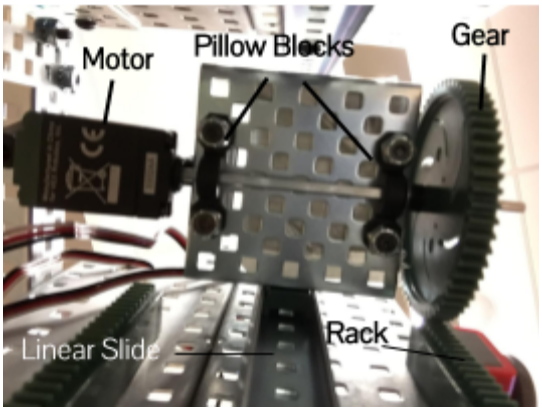
Based on the decision matrix, we selected a rack and pinion system with an ultrasound sensor and three bump switches and limit switches. This design was clearly more stable, accurate, and stronger, as well as resistant to damage.

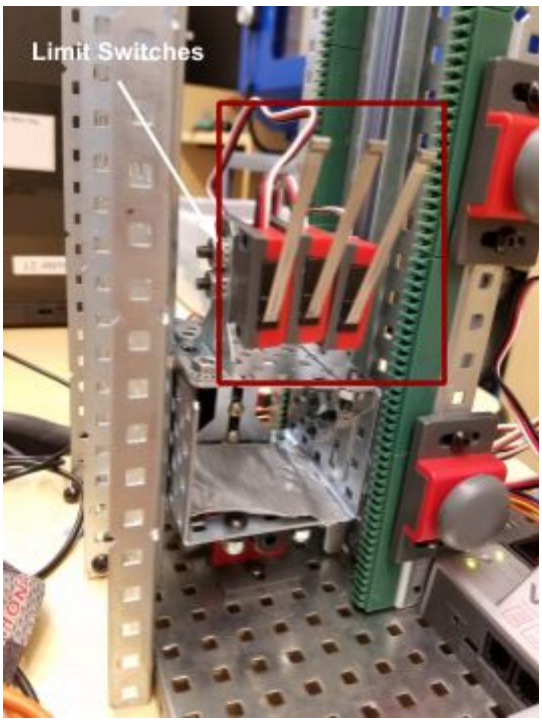
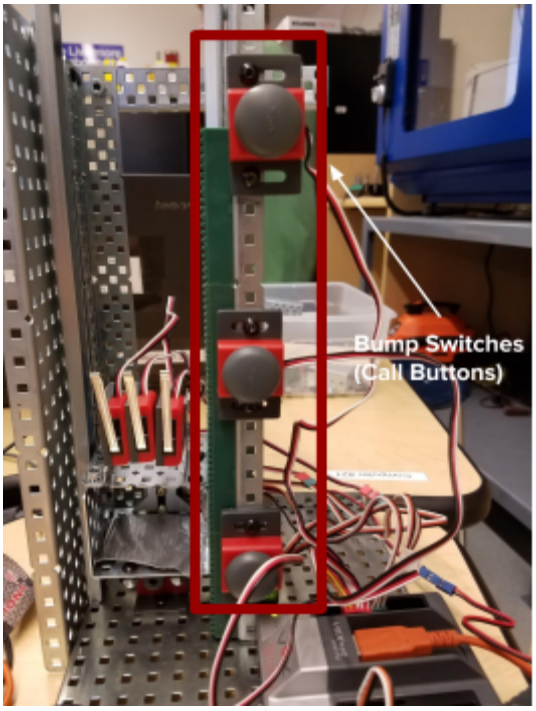
Selected Design: Design 1 - Rack and Pinion

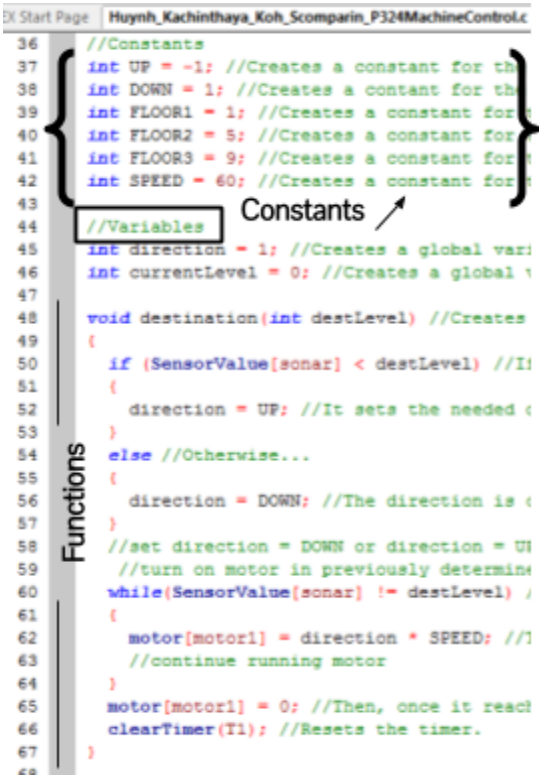
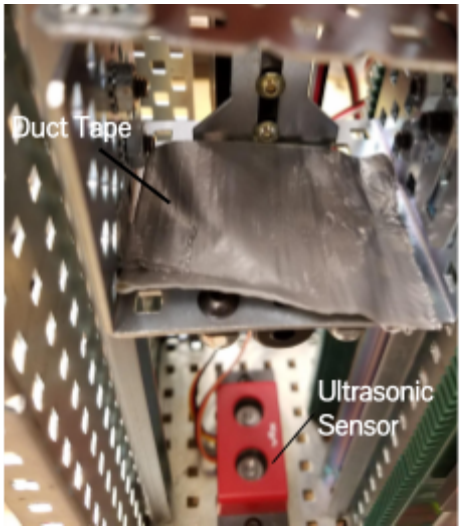
- 1 Ultrasonic
- 3 Bump Switches
- 3 Limit Switches
- 2 Motors

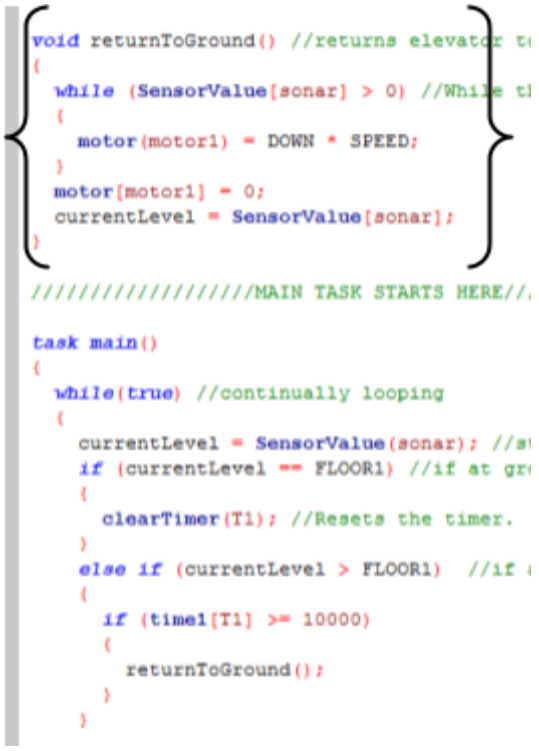
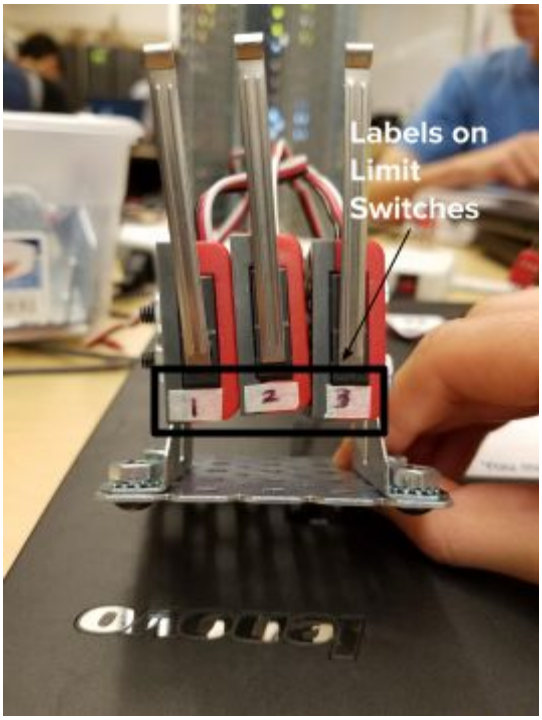
Design Modifications

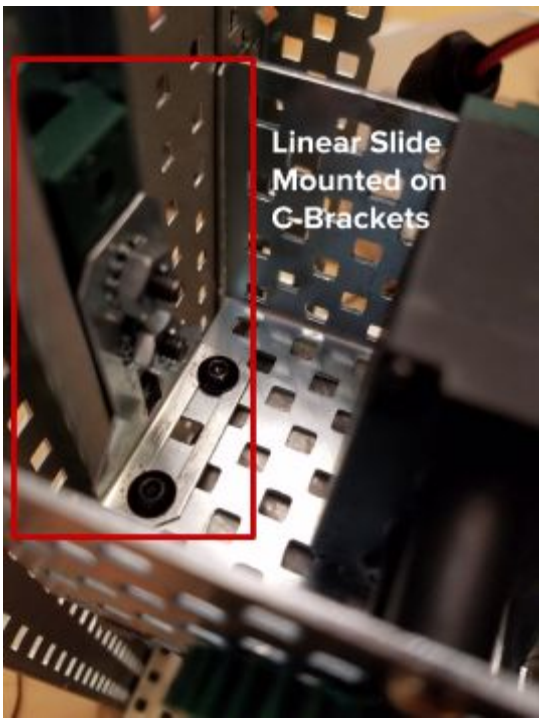
Picture/Diagram	Explanation	Initial/Date
	We constructed the elevator shaft today, mostly according to the initial design selected in the decision matrix. The one change we made was that we put only one linear slide on one side and two racks on the other, because there was only space on our elevator for one motor and having two racks would require gears spinning in different directions. Having both racks on one side would allow us to use one motor driving two gears in the same direction. We secured the ultrasound sensor to the bottom of the elevator shaft.	KH, AK, CK, LS 4/25/17
	We began constructing the three floors as well, with call buttons on each. This is going to be attached next to the elevator shaft once the construction for that is complete.	KH, AK, CK, LS 4/25/17

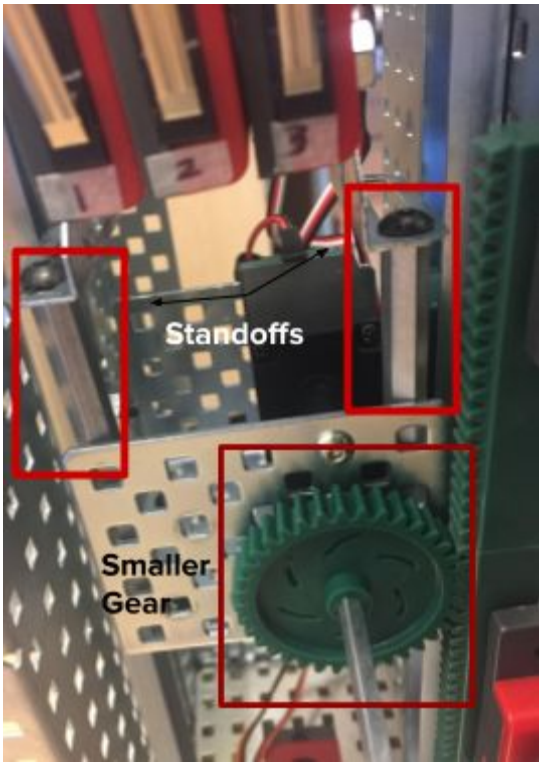
	The elevator itself needed to be redesigned in order to fit into the shaft we designed, as well as for the limit switches to be more accessible. In our initial design, the limit switches were all assembled next to each other, but would have been difficult to press a particular switch. In this modification, the switches were screwed in at different intervals, allowing for the elevator to respect size constraints while respecting the accessibility of the switches. In addition, the new configuration allowed for the wires attached on each of the switches to move smoothly up and down with the elevator, rather than becoming hung up on various other parts.	KH, AK, CK, LS 4/26/17
 	We modified the elevator to be smaller and more even. The overlapping metal in our previous design caused the linear slide inserts to be slanted, creating a lot of friction and slowing our elevator. This new design uses a large c-bracket with straighter sides. Instead of using a small gear for the pinion, we ended up using a larger gear because it meshed perfectly with the rack. This gave us a larger angular velocity, greatly increasing the speed of the elevator. In addition, instead of two motors, we decided to use only one because of space constraints. We positioned the motor centrally beneath the elevator, and secured the axle with two pillow blocks on the bottom of the elevator.	KH, AK, CK, LS 4/28/17

	We placed the three lift switches in a row on top of the elevator since we did not have enough space inside the elevator. It is not too difficult to press only one switch, as long as you are careful. This also allowed for the elevator to be more balanced than when all the lift switches were staggered using spacers before.	KH, AK, CK, LS 4/28/17
	Due to space constraints, we placed the bump switches directly on the elevator shaft rather than on separate floors (second picture from the top). That way, our material use was also more efficient and the switches were truly in the same position as they would be in real life - right next to the elevator doors.	KH, AK, CK, LS 4/28/17

	We began coding today, generally following the structure of the flow chart. To make the code more efficient and less repetitive, we created a function for the elevator to go up or down to its destination with the desired height as an integer parameter. We also added several more variables to control the speed, direction, and current height of the elevator. This would allow us ease of programming and increase other people's ease of understand our code.	KH, AK, CK, LS 5/1/17
	Since there were holes in the bottom of the elevator, the values obtained by the ultrasound sensor fluctuated widely and could not be used to accurately determine the height of the elevator. We had to put some duct tape on the floor of the elevator in order to allow better reflection of the ultrasound pulse for a more accurate sensor reading.	KH, AK, CK, LS 5/1/17

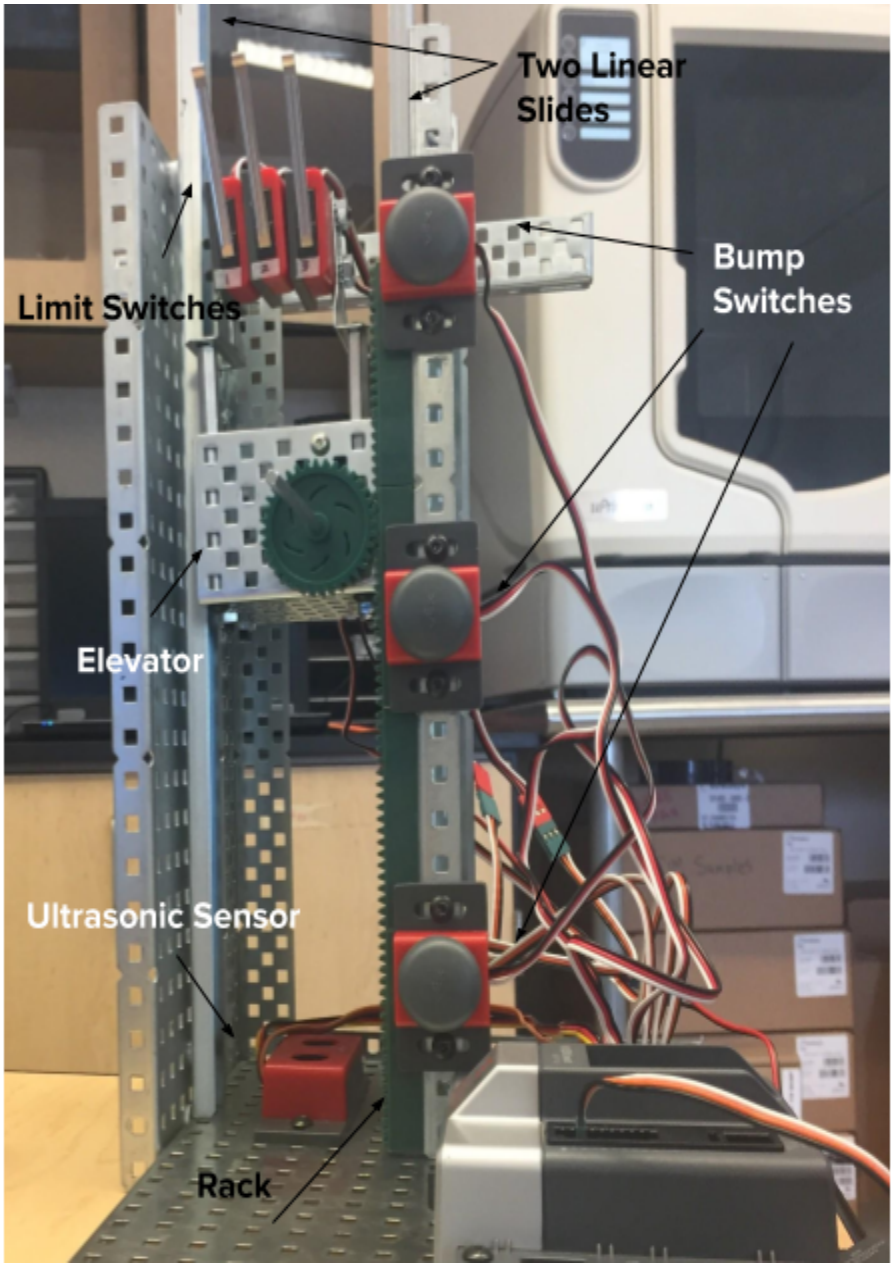
	We added a function for the safety feature, called returnToGround(). When the inactivity time period is passed, the function is called. It simply tells the motor to go down until it reaches the ground floor. Although this segment of code is not repeated, placing it in a different function allows our task main to be more understandable. It also allows us to visualize the flow of our program better and identify errors, if any.	KH, AK, CK, LS 5/2/17
	The problem with the way we stacked the limit switches is that it could be confusing for the user to figure out which switch corresponds to which floor. To solve this, we put strips of masking tape on the switches and labeled them with their respective floor numbers.	KH, AK, CK, LS 5/3/17

 	For the past few days, we have been trying to fix friction issues with the linear slides. Because our elevator is driven by only one rack and pinion on one side, the elevator tends to twist when going in one direction (sometimes up, sometimes down), and will begin to stall. We solved this problem by turning the elevator 90 degrees and mounting the linear slide inserts on right angle brackets instead. This allowed the lift to have some freedom to twist a little and still be able to travel up and down without stalling. We also had to change the position of our motor and use a smaller gear. In order to keep the smaller gear meshed with the rack, we adjusted the spacing for the linear slides as well. Moving the motor from the bottom of the elevator allowed us to receive more accurate values from our ultrasound sensor as well. Finally, the entire lift was working! We tested out the program and all our features worked.	KH, AK, CK, LS 5/4/17

	Lastly, we mounted the three limit switches to the top of the elevator. At first we attached the square plate with standoffs to the elevator, but the plate rubbed against the linear slide and caused our motor to stall. We ended up removing the plate and using standoffs to connect the right angle brackets holding the limit switches directly to the elevator, completely removing the square plate. This reduced weight, which greatly improved the accuracy of our elevator when going down and increased the load it could carry. We also placed the duct tape on the bottom of the elevator.	KH, AK, CK, LS 5/4/17

Final System Solution

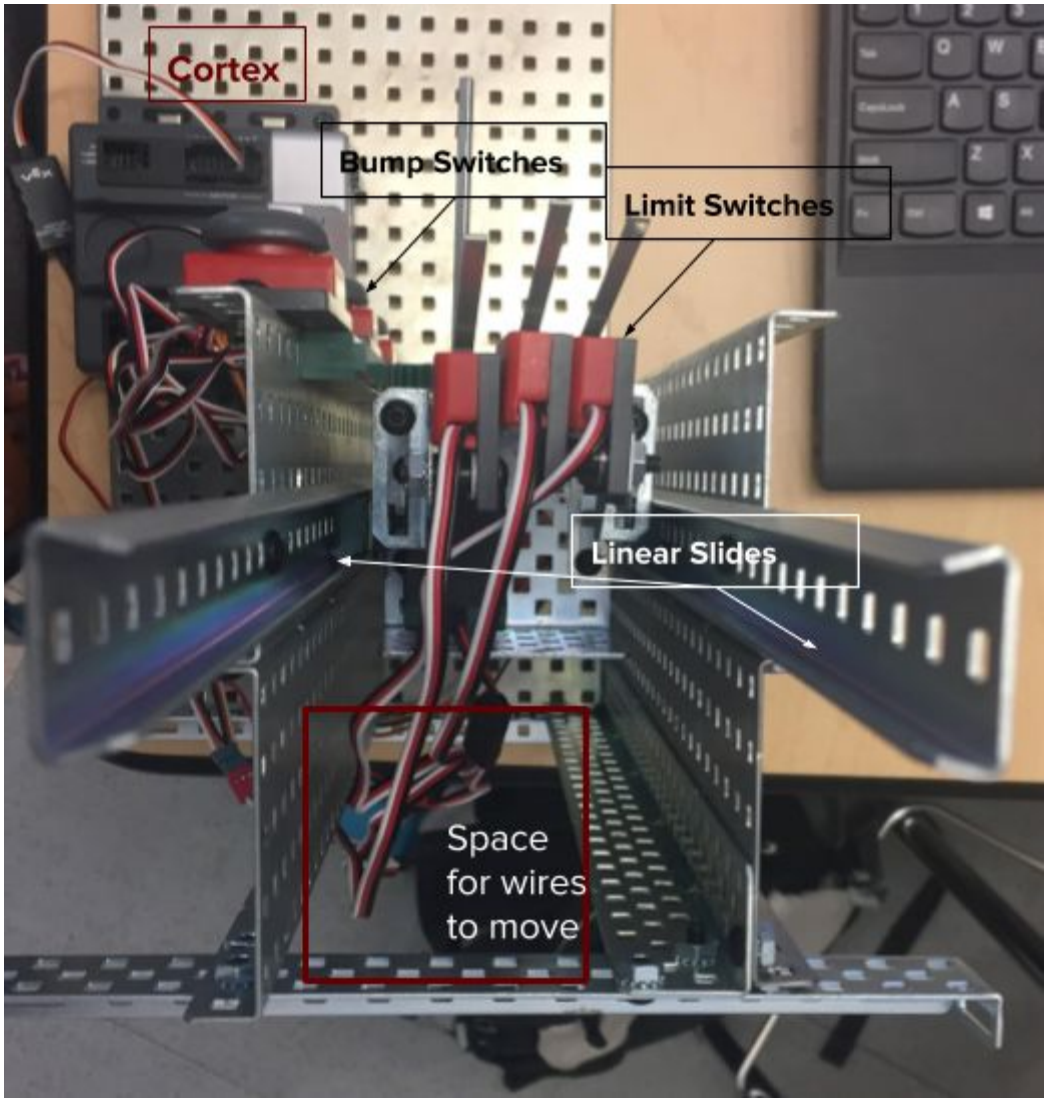
Physical Solution



Front View

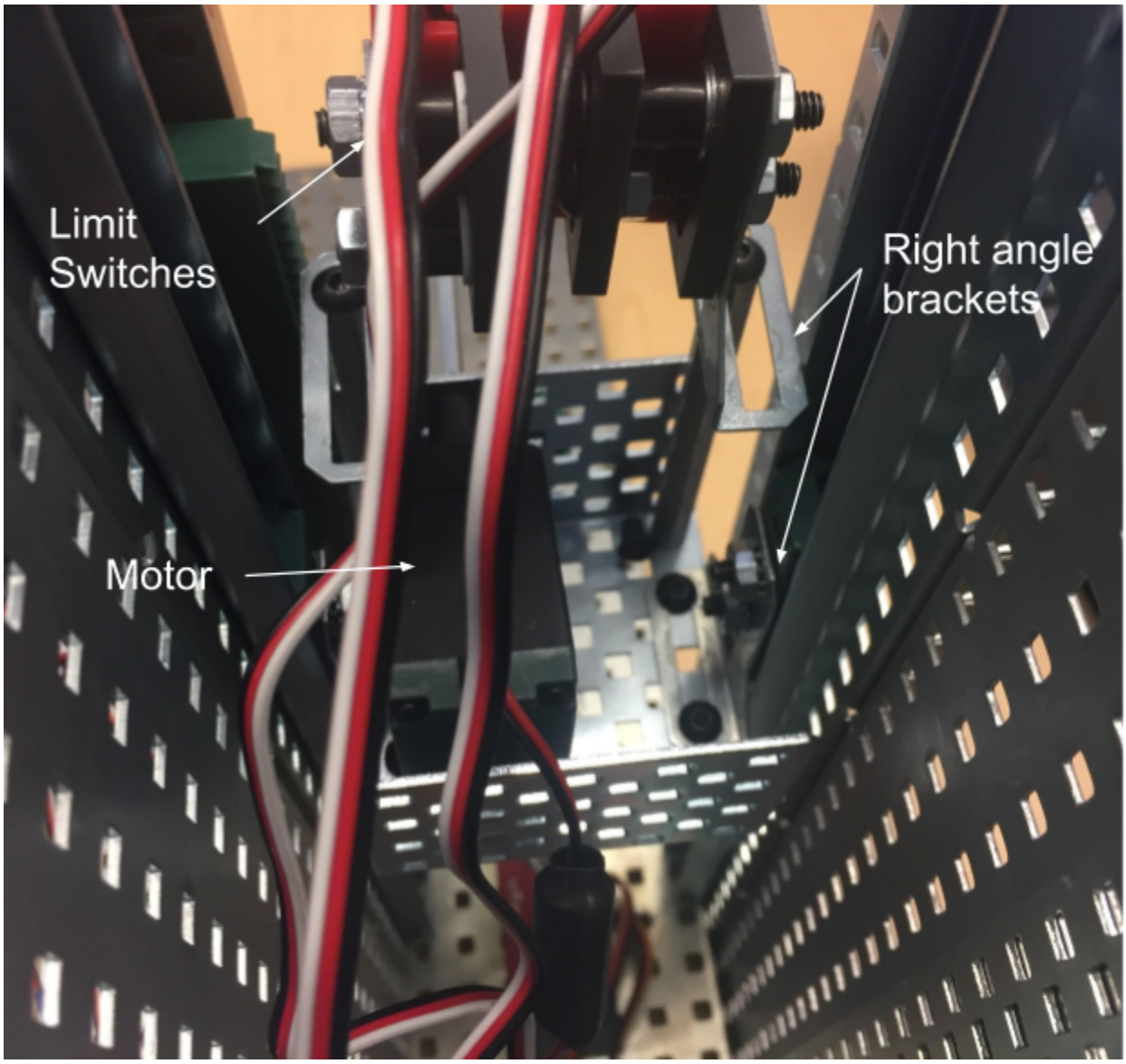
KH, AK, CK, LS
5/4/17

Our elevator uses a rack and pinion system with two linear slides for stabilization. The elevator uses three limit switches, three bump switches, an ultrasonic sensor and a VEX 393 motor. The ultrasonic sensor measures the distance of its position and uses the information to respond to the next command from either one of the bumper switches or limit switches. The motor will then turn on and go to the designated level.
This final design for our elevator is speedy, accurate, and very safe. It is a simple and intuitive design.



Top View

KH, AK, CK, LS
5/4/17



Inside View

KH, AK, CK, LS
5/4/17

Program Solution

Huynh_Kachinthaya_Koh_Scomparin_P324MachineControl.c*

```
#pragma config(Sensor, dgtl1,  lift1,          sensorTouch)
#pragma config(Sensor, dgtl2,  lift2,          sensorTouch)
#pragma config(Sensor, dgtl3,  lift3,          sensorTouch)
#pragma config(Sensor, dgtl4,  call1,          sensorTouch)
#pragma config(Sensor, dgtl5,  call2,          sensorTouch)
#pragma config(Sensor, dgtl6,  call3,          sensorTouch)
#pragma config(Sensor, dgtl7,  sonar,          sensorSONAR_inch)
#pragma config(Sensor, dgtl9,  stopBtn,        sensorTouch)
#pragma config(Motor,  port8,           motor1,        tmotorVex393_MC29, openLoop)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

/*

Project Title: Project 3.2.4 Machine Control - Elevator

Team Members: Kaitlyn Huynh, Anvitha Kachinthaya, Christy Koh, and Loic Scomparin
*
Date: May 2, 2017

Class & Section: POE Period 6

Task Description: Controls the movement of the elevator in response to button inputs on different floors.

Pseudocode:
- Each floor has a value set to it based on its distance from the ultrasound sensor.
- There are also constants set for going up or down, depending on where the elevator is in relation to the ultrasound sensor.
- The variable currentLevel measures the location of the elevator.
- When either a call button or lift button is pressed, outside or within the elevator, the elevator first checks to see if it is already at the
- If the elevator is not at the required floor, it checks to see if it must move up or down to get there, and assigns a postive or negative dire
- The elevator moves in the direction required of it at the speed assigned until the ultrasonic sensor senses that the elevator has reached its
- A timer is set every time the motor stops, and if it remains idle for more than sixty seconds, it moves down to the ground floor again for saf
- The timer is reset whenever the elevator reaches its floor.

*/
```

KH, AK, CK, LS
5/4/17

Above are the pragma statements and program title and description. The program utilizes three limit switches, three bump switches, an ultrasonic sensor and a VEX 393 motor. The pseudocode explains what the program does, and is a summary of all the comments thereafter.

Huynh_Kachinthaya_Koh_Scomparin_P324MachineControl.c*

```
35
36  //Constants
37  int UP = -1; //Creates a constant for the direction of the motor when the elevator goes up.
38  int DOWN = 1; //Creates a contant for the direction of the motor when the elevator goes down.
39  int FLOOR1 = 1; //Creates a constant for the value of the ultrasonic sensor at the first floor.
40  int FLOOR2 = 5; //Creates a constant for the value of the ultrasonic sensor at the second floor.
41  int FLOOR3 = 8; //Creates a constant for the value of the ultrasonic sensor at the third floor.
42  int SPEED = 100; //Creates a constant for the speed of the motor regardless of the direction it is travelling in.
43
44  //Variables
45  int direction = 1; //Creates a global variable for the direction of the elevator that can be changed with the constants "up" and "down."
46  int currentLevel = 0; //Creates a global variable for the level that the elevator is at, that changes constantly as it moves.
47
48  void destination(int destLevel) //Creates a function that allows the elevator to reach its destination by checking whether it has to go up or do
49  {
50    if (SensorValue[sonar] < destLevel) //If the ultrasonic sensor senses that the elevator is below the desired destination...
51    {
52      direction = UP; //It sets the needed direction as "up," or a negative motor value.
53    }
54    else //Otherwise...
55    {
56      direction = DOWN; //The direction is changed to "down," or a positive motor value.
57    }
58    //set direction = DOWN or direction = UP
59     //turn on motor in previously determined direction
60    while(SensorValue[sonar] != destLevel) //while the current level is not the destination level
61    {
62      motor[motor1] = direction * SPEED; //The motor runs in the direction determined above at the global variable speed.
63      //continue running motor
64    }
65    motor[motor1] = 0; //Then, once it reaches its floor, the motor stops.
66    clearTimer(T1); //Resets the timer.
67  }
68
```

KH, AK, CK, LS
5/4/17

The constants and variables are defined as above: the direction of the elevator, the heights of the floor levels, the speed of the motor and the distance of the elevator from the ultrasonic sensor at any given time. The screenshot displays the function “destination” as well, which checks whether the elevator must move up or down, changing the value of the variable “direction” as needed, and then turns the motor until the ultrasonic sensor value is equal to that of the floor to which the elevator must travel.

Huynh_Kachinthaya_Koh_Scomparin_P324MachineControl.c*

```
65    motor[motor1] = 0; //Then, once it reaches its floor, the motor stops.
66    clearTimer(T1); //Resets the timer.
67  }
68
69  void returnToGround() //Returns elevator to ground floor.
70  {
71    while (SensorValue[sonar] > FLOOR1) //While the ultrasonic sensor is greater than zero...
72    {
73      motor(motor1) = DOWN * SPEED; //The motor moves down in the speed that is required until it reaches the ground floor.
74    }
75    motor[motor1] = 0; //Then, the motor stops.
76    currentLevel = SensorValue[sonar]; //And the current level of the elevator is equal to that of the ultrasonic sensor.
77  }
78
```

The function “returnToGround” is a built-in safety feature that allows the elevator to return to the ground floor, or first floor. This function is run if the elevator has been sitting idle for more than three seconds, without being called by a button.

Huynh_Kachinthaya_Koh_Scomparin_P324MachineControl.c*

```
78
79  ////////////////////MAIN TASK STARTS HERE//////////////////////////////
80
81  task main()
82  {
83    while(true) //continually looping
84    {
85      currentLevel = SensorValue(sonar); //store the current elevator level
86      if (currentLevel == FLOOR1) //if at ground level...
87      {
88        clearTimer(T1); //Resets the timer.
89      }
90      else if (currentLevel > FLOOR1)  //if above ground level..
91      {
92        if (time1[T1] >= 3000) //If the elevator has been idle for more than three seconds.
93        {
94          returnToGround(); //Calls the function returnToGround, stated above.
95        }
96      }
97
98      //Regardless of position of elevator, check buttons
99      if ((SensorValue[lift1] == 1 || SensorValue[call1] == 1) && currentLevel != FLOOR1)
100       //check if ground floor buttons are pressed, and that the elevator is not already there
101     {
102       destination(FLOOR1); //call function for elevator to go to ground floor
103     }
104     else if ((SensorValue[lift2] == 1 || SensorValue[call2] == 1) && currentLevel != FLOOR2)
105       //check if 2nd floor buttons are pressed, and that the elevator is not already there
106     {
107       destination(FLOOR2); //call function for elevator to go to 2nd floor
108     }
109     else if ((SensorValue[lift3] == 1 || SensorValue[call3] == 1) && currentLevel != FLOOR3)
110       //check if 3rd floor buttons are pressed, and that the elevator is not already there
111     {
112       destination(FLOOR3); //call function for elevator to go to 3rd floor
113     }
114   }
115 }
```

KH, AK, CK, LS
5/4/17

The main task of the program lies in identifying where the elevator is at any given moment, and deciding where it has to go based on the buttons pressed and the function “destination.” First, the program checks whether the safety feature should be run. If the timer has not exceeded three seconds or the elevator is already at ground floor, it continues. When a button for a floor is pressed, either from inside the elevator or outside, the program checks whether it is already at the floor it needs to be at. Only if the levels of the elevator and where it needs to go do not match, the elevator moves. Since the program is on a forever loop, there is no end to how long it can be run.