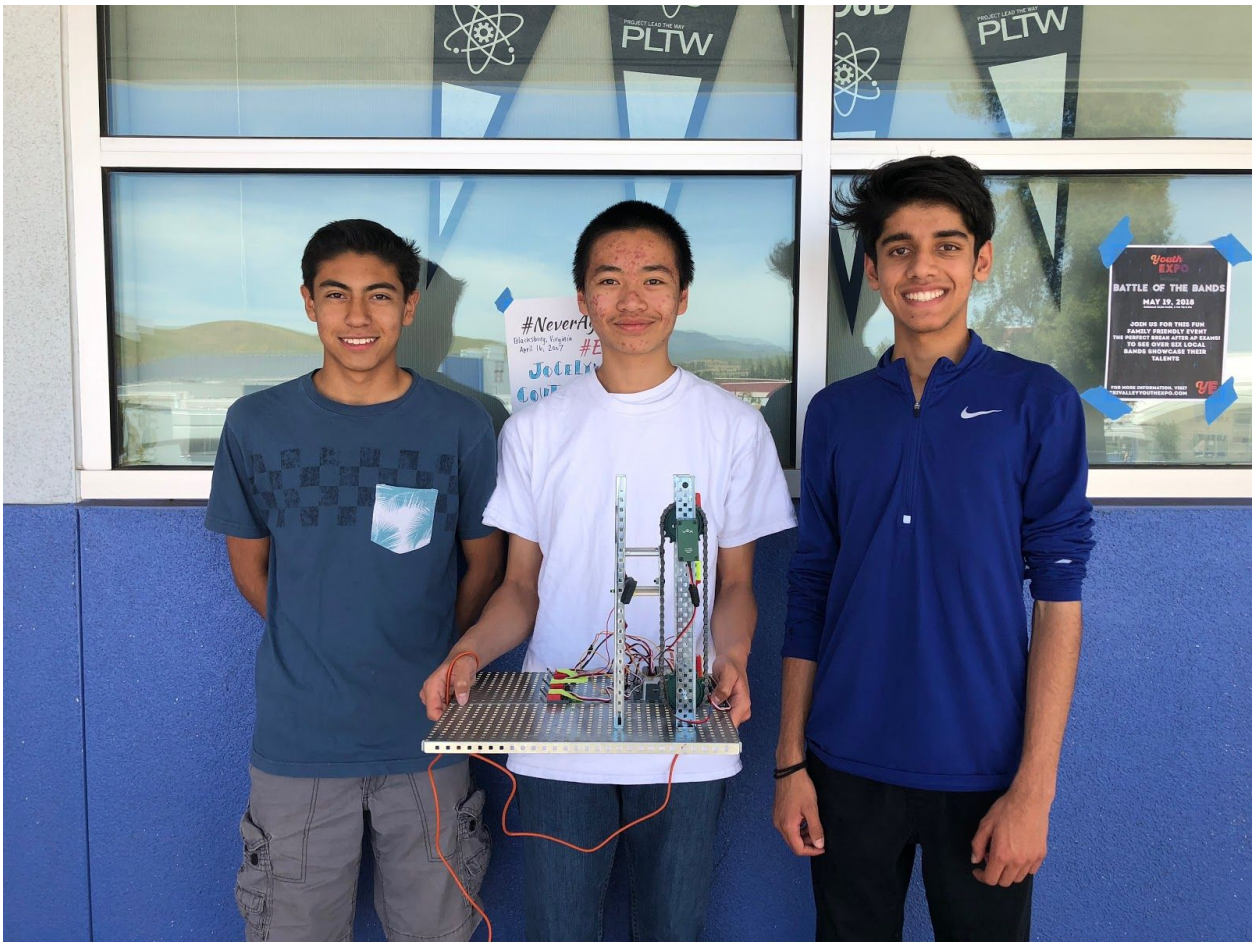
POE PERIOD 7

Team Elevator Music - Ekansh A, Adam G, Justin T

Elevator Project Documentation



Project 3.2.4 Machine Control Design

April 19th 2018- May 4th, 2018

Table of Contents

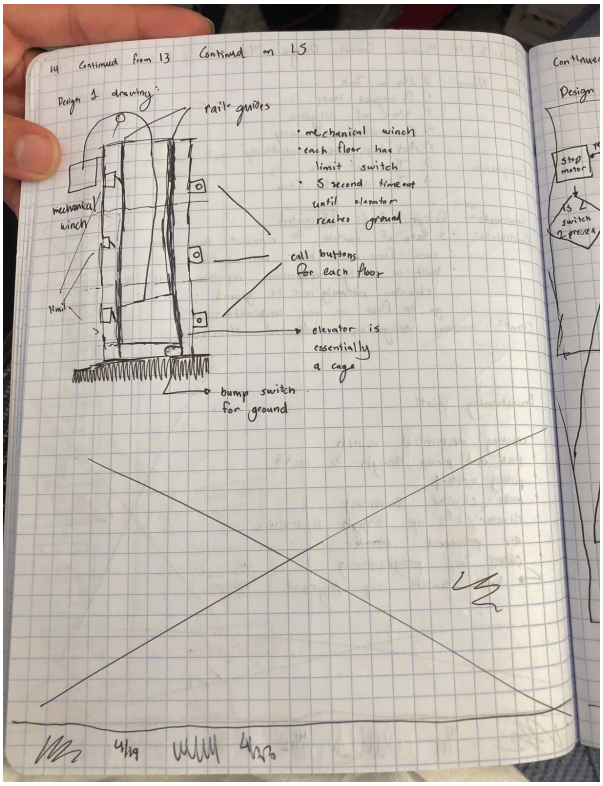
Design Brief

Client:	Elevator Corporation
Designers:	Ekansh Agrawal, Adam Garza, Justin Thai
Problem Statement:	It is troublesome to determine whether the software that allows an elevator to work does not have errors since applying them to actual elevators without prior testing can be dangerous for the user.
Design Statement:	Our team will create software for a residential elevator that can go between three different floors based on the floors requested by call buttons on each floor, as well as by a set of buttons inside the elevator. This elevator is also going to have a safety measure to return to the ground floor after a five seconds of nonuse. Additionally, a small-scale physical model of the three-floor elevator is going to be made to both troubleshoot and to demonstrate the functionality of our code.
Constraints:	The requirements of this project is that physical prototype must use the VEX cortex as well as VEX motors and sensors. For this reason the software powering our elevator must also be created in RobotC.
Deliverables:	Our team will deliver on the due date of this project: ● A physical prototype of the residential elevator with the required sensors and actuators; ● Software designed in RobotC for the use of the model; ● This team report documenting the development of our project; and ● Individually, each member will have their own documentation displaying ○ Their design brainstorming, ○ Their personal contributions to the team project, and ○ Their answers to the conclusion questions.

Brainstorming Sketches and Programs

Ekansh Agrawal

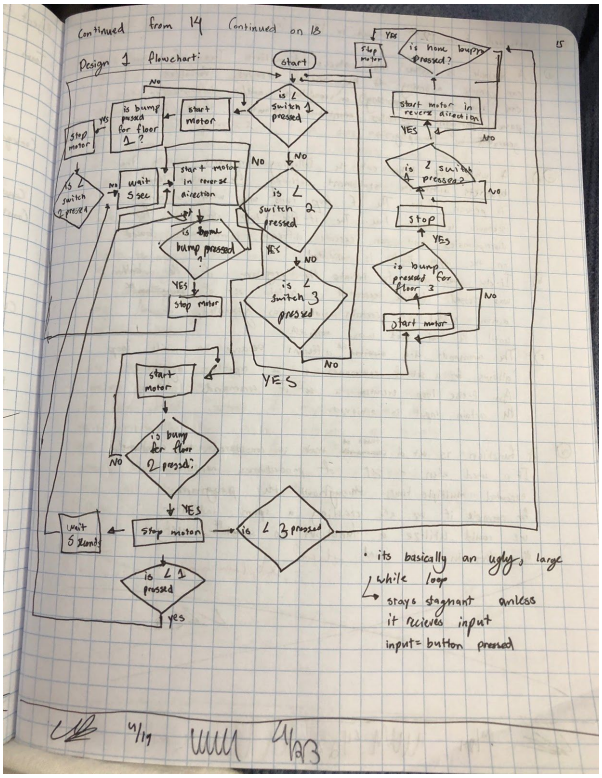
Physical Sketch:



Ekansh Agrawal

My brainstorm sketch was centered around the implementation of a mechanical winch with support railings. This winch would be powered by a motor anchored on the side of the railings. A pulley would transfer the rotational speed from the motor into the winch. Each floor has its own corresponding limit switches to indicate the motor to stop moving. Call buttons inside and outside the elevator would allow the user to call the elevator to a specific floor.

Program Sketch (Flowchart):

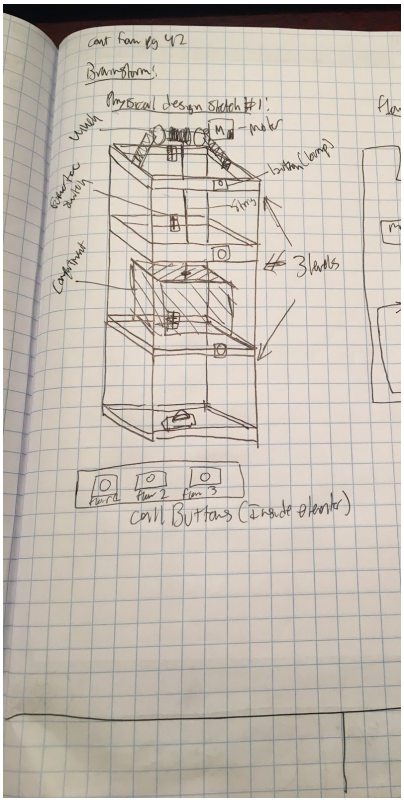


Ekansh Agrawal

The program begins by waiting for the user to call the elevator to the desired floor. It would begin by waiting until any call button had been pressed. The program would then initiate the winch to move towards the desired floor until the corresponding limit switch was pressed. Until the limit switch was pressed, the winch would keep rotating in that general area. Once the elevator reaches the floow it waits up to 5 seconds before returning home. If another floor is called within that time period, the loop is broken and the winch continues to rotate. This entire loop is never ending and keeps rotating as long as there is power to the system.

Adam Garza

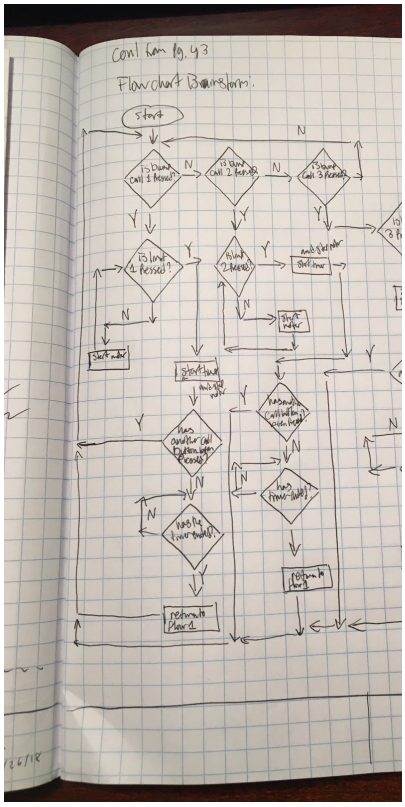
Physical Sketch:



Adam Garza

My brainstorm sketch was made to feature a rectangular prism cage that the elevator would ride inside attached to a string pulled by a winch. This winch would be powered by a motor stationed at the top of the cage. If called the motor would activate, as the elevator passes the correct limit switch, it would stop. Call buttons inside the elevator, but located outside in the physical model, would then tell the

Program Sketch (Flowchart):



Adam Garza

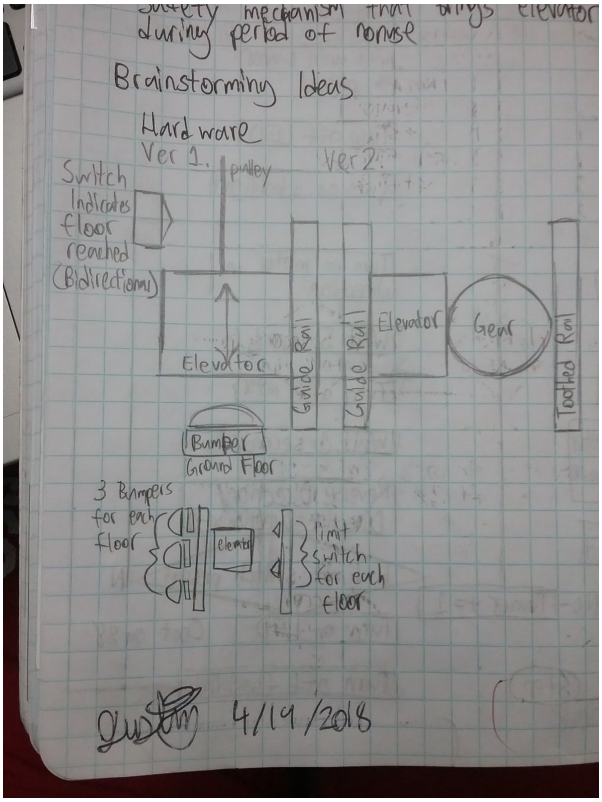
This flowchart design was a concept for what I thought the program would be accomplishing. It would begin by asking if any call button had been pressed. Then depending on which call button had been pressed, the program would ask if then corresponding limit switch had been pressed. If no, the program would start the motor until the limit switch had been pressed. If yes, the program would start a timer and wait

elevator where to go.

for another call button to be pressed. If no other button had been pressed, the elevator would return to the first floor and wait for a call button to be pressed. The program is a never-ending loop.

Justin Thai

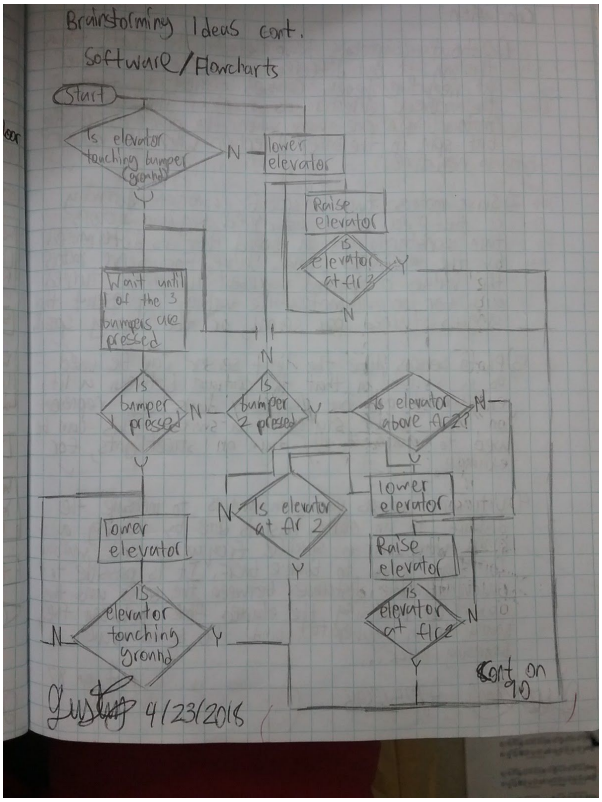
Physical Sketch:



Both my designs feature an elevator that moves with the assistance of guide rails, and sensors that detect when a floor has been called for and sensors that detect which floor the elevator is at. The main difference is that one design uses a winch to raise and lower the elevator, while the other is attached to a motorized gear with a toothed rail, used to “climb” up and down the three floors.

Justin Thai

Program Sketch (Flowchart):



This flowchart initializes by making sure the elevator is already on the first floor before proceeding. If the first floor is requested, the elevator moves down until sensors indicate it has touched the ground. If the third floor is requested, the elevator moves up until it reaches the third floor. If the second floor is requested, it checks if the elevator is above or below floor two and either raises or lowers the elevator until the second floor is reached. This repeats indefinitely.

Justin Thai

Design Matrix

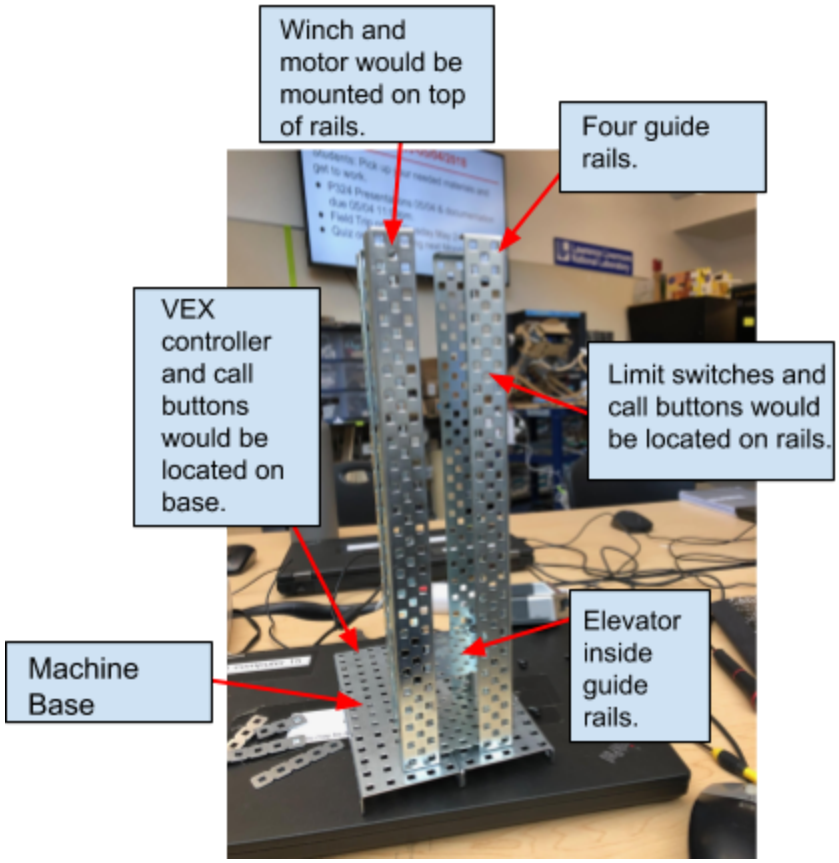
3-Most favored 1-Least favored				
Qualities / Attributes	Weight	Prototype #1 Ekansh	Prototype #1 Adam	Prototype #3 Justin
Functionality	**5**	3	3	2
Stability	**3**	3	3	2
Aesthetics	**2**	2	3	2
Time to create	**4**	2	2	1
Complexity	**4**	2	2	1
*Numbers of limit switches**	**-**	4	4	4
*Numbers of motors**	**-**	1	1	1
Number of bump switches	**-**	3	3	3
Total Score	**-**	44	46	28
**Does not factor into the design matrix. Instead, it serves as a reference to distinguish between prototypes.*				

X Eugene Chou 04/26/2018

The value of each design was determined by its functionality, complexity, the amount of time it would take to complete, its stability, and the aesthetic quality of each design. Weight was assigned based on the importance of each attribute, prioritizing how well the design solves the problem and the design's feasibility of being easily created. We created a weighted scale to prioritize which category was more important. This would skew the scoring to give the proper criteria higher weightage. Justin's design, revolving around a tooth gear train system, was deemed inefficient after group input. Ekansh's and Adam's designed weighed at a higher scale as their design was more feasible. Adam's overall design achieved greater points due to its overall aesthetics and complexity to build.

Modification Sketches

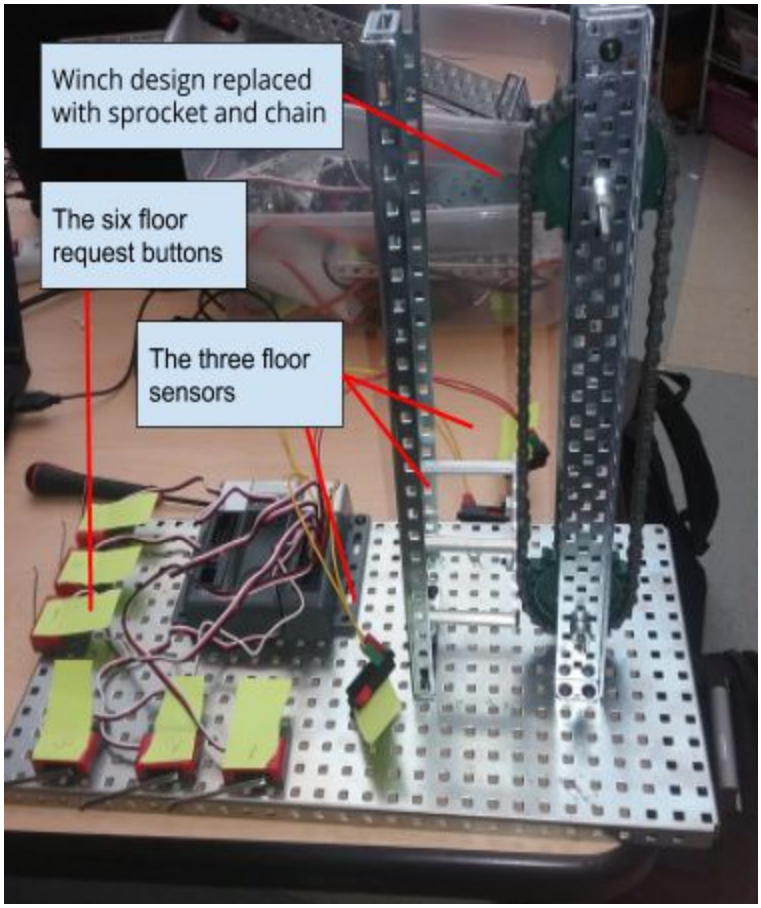
Monday, April 30th, 2018



The first design was based off of Adam's brainstorming sketch. This included the elevator fitting inside four glide rails and having a winch mounted at the top of the machine. Even though this early stage is only a shell of the design, this version would have had a motor attached to the winch that had a string attached to the elevator. When activated, the motor would cause the winch to turn and raise or lower the elevator. The elevator would pass limit switches that would activate and stop the winch when the elevator reached the correct floor. This design was soon thrown out due to friction issues and other, more efficient designs would replace it, so there isn't much to see in the picture shown. Comments and documentation were added to our program.

Ekansh Agrawal, *Adam Garza, Justin Thai* - 4/30/2018

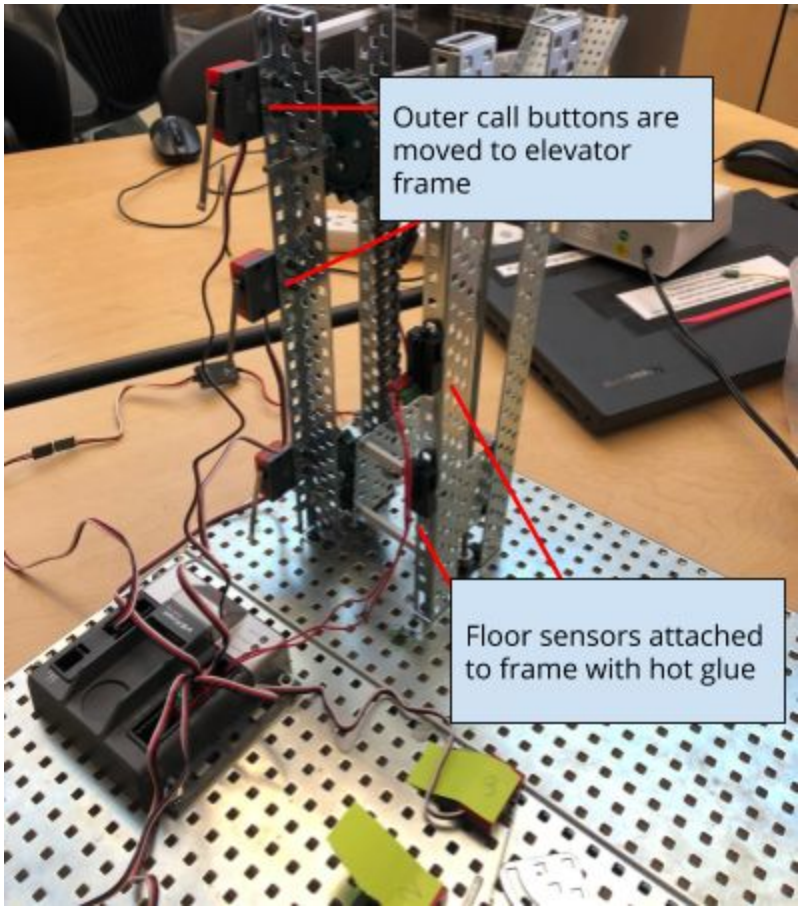
Tuesday, May 1st, 2018



We modified our design to move the elevator with a sprocket and chain instead of with a winch. We also added to our model the three floor sensors which would detect which floor the elevator is at, as well as the six limit switches which would act as the two sets of call buttons with one present on the inside of the elevator and the other outside on every floor. For the three floor sensors, wires were prepared to make the Fischer Tech switches compatible with the VEX Cortex, however we did not have a way of attaching the switches to the frame in such a way that the elevator would trigger the sensors when it arrived at the different floors.
We had our program troubleshot separately, and were able to resolve most of our errors. Most of the changes in the code involved what the elevator would do in periods of nonuse. Our design used a timer that would increase by one increment every millisecond, but would reset back to zero milliseconds everytime a call button was pressed to request movement of the elevator. When the equivalent of five seconds has passed the elevator automatically moves to the first floor if it is not already there.

Ekansh Agrawal, *Adam Garza, Justin Thai* - 5/1/2018

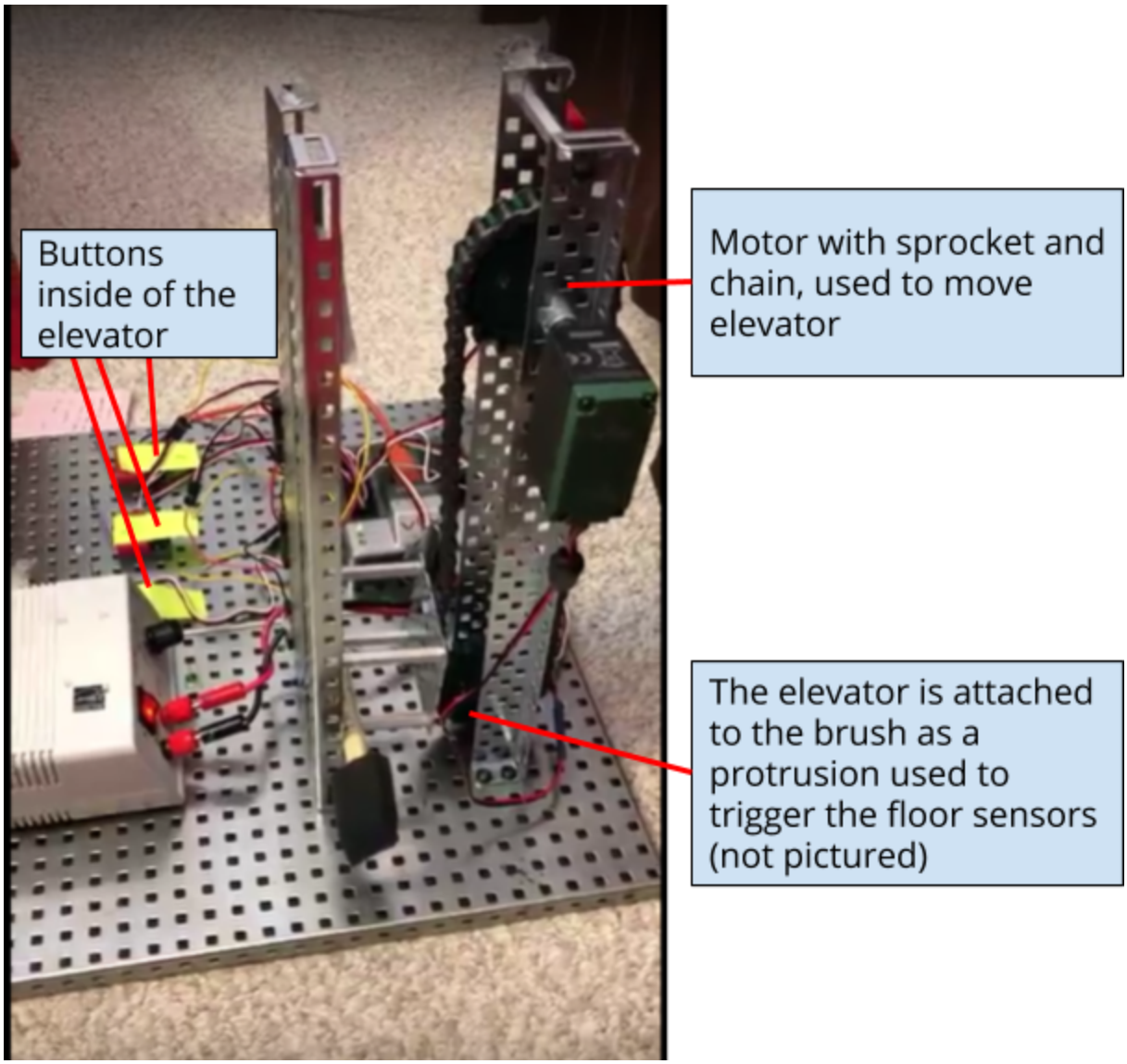
Wednesday, March 2nd, 2018

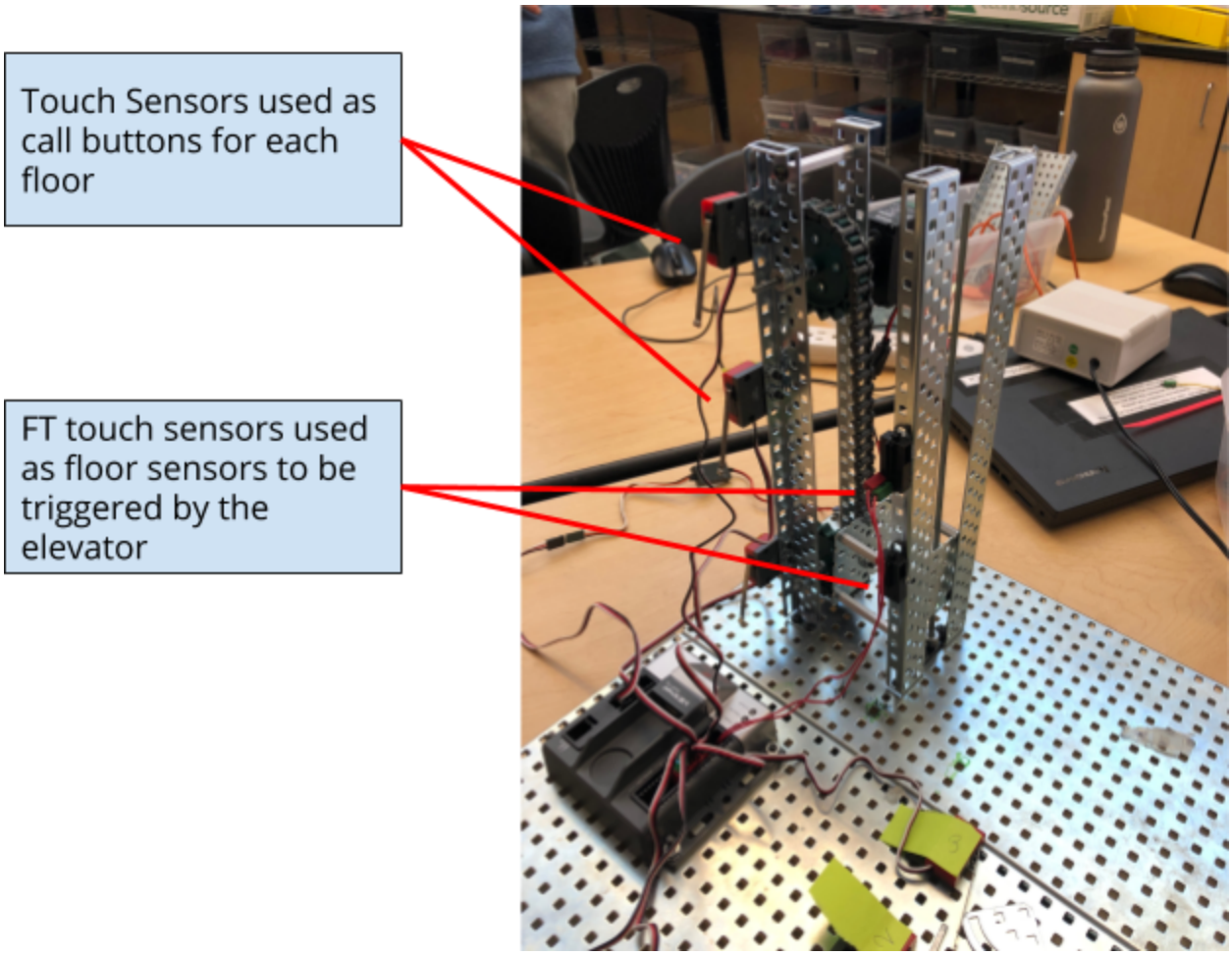


We had the three sensors that represented the call buttons outside the elevator moved to the frame of the elevator around the areas they would be if their respective floors were there. Additionally, the floor sensors were connected to the frame with hot glue, as well as a rod protruding from the elevator that would trigger those switches as the elevator moves. However, the rod was too flimsy, causing the elevator to not press against the sensors with enough force to register that it is at the correct floor. Additionally, the software makes the elevator move too fast, so that even if the elevator where to register that it has arrived at the desired floor, by the time it stopped it would no longer be pressing against the switch and the next time a floor was requested the elevator would not know which direction to move to arrive at the requested floor.

Ekansh Agrawal, *Adam Garza, Justin Thai* - 5/2/2018

Final Physical Solution





Our physical model a single motor and nine sensors. Six sensors are used as call buttons that request the elevator to move to each floor. There are two sets of buttons that represent the three floors of the elevator; One set represents the call buttons on each floor, and the other set at the base of our model represents the buttons that would be present on the inside of the elevator. The other three sensors are switches used as “floor sensors” which detect when they are pressed against the elevator, making the floor the elevator is at known. This information is useful so that the elevator will not move if it has been requested to move to a floor it is already at, and can also be used to determine which direction the elevator has to go in order to move to the second floor. The motor is attached to sprocket and chains, which connects to the elevator and allows it to move upward and downwards.

Final Program Solution

```
#pragma config(Sensor, dgtl4,  button1,        sensorTouch)
#pragma config(Sensor, dgtl5,  button2,        sensorTouch)
#pragma config(Sensor, dgtl6,  button3,        sensorTouch)
#pragma config(Sensor, dgtl7,  button1b,       sensorTouch)
#pragma config(Sensor, dgtl8,  button2b,       sensorTouch)
#pragma config(Sensor, dgtl9,  button3b,       sensorTouch)
#pragma config(Sensor, dgtl10, fl1,            sensorTouch)
#pragma config(Sensor, dgtl11, fl2,            sensorTouch)
#pragma config(Sensor, dgtl12, fl3,            sensorTouch)
#pragma config(Motor,  port3,           winchmotor,    tmotorVex393_MC29, openLoop)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

//fl1-fl3 are the touch sensors that will indicate for the elevator when a floor is reached
/*digital ports 4-9 represent the buttons that request the elevator to move to different floors
  each "floor request button" has 2 sensors associated with it because in an elevator, there are
  two sets of floor call buttons: one for inside the elevator and one for the buttons on each floor*/

/*
  Project Title: Project 3.2.4
  Team Members: Ekansh Agrawal, Adam Garza, and Justin Thai
  Date: 4/18/18
  Class and Section: 7 Period
  Task Description:
    Design the control system and a prototype of an elevator that can go between three floors in any combination.
    The prototype must include a set of three switches to represent each floor inside the elevator. Each floor
    the elevator stops at must have a call button. A built-in safety mechanism requires that the elevator
    normally rest on the ground floor and return to the ground floor after a user-determined period of nonuse.

  Pseudocode:
    When starting move the elevator to the 1st floor. Every millisecond check to see if either the buttons inside
    of the elevator or the outside buttons on each floor are pressed. If the 1st floor is requested (while the
    elevator is not on the 1st floor), move the elevator downwards until it triggers the sensor for the 1st floor.
    If the 3rd floor is requested (while the elevator is not on the 3rd floor), move the elevator upwards until
    it triggers the sensor for the 3rd floor. If the 2nd floor is requested (while the elevator is not on the 2nd
    floor), check where the elevator is at with the floor sensors. If the elevator is below the 2nd floor, move
    the elevator upwards. If the elevator is above the 2nd floor, move it downwards. Continue moving until the
    sensor for the 2nd floor is triggered. A timer is constantly counting the seconds that pass; every time a
    floor is requested, this timer resets. After 5 seconds of nonuse, the elevator calls itself to the 1st floor.

*/

int on = 1;       int off = 0;          //(For Sensors) 1 is on and 0 is off
int flr = 0;                            //Integer for current floor the elevator is at
int theTimer = 0;                       //Variable for timer intended to count how many seconds has passed since a floor was requested
int speed = 30;
int speeddown = -20;
```

104:38 C and C++ Spaces: 2

```
void moveFloor(int flrnum)              //input flrnum represents desired floor
{
  theTimer = 0;                         //Restart timer since a floor has been requested
  if (flrnum == 1)                      //Run if elevator is requested for on the 1st floor
  {
    if (SensorValue(fl1)==off)          //Run only if elevator is not already on the 1st floor
    {
      motor(winchmotor)=speeddown;            //Make motor move downwards
      untilTouch(fl1);                  //Wait until 1st floor is reached
      flr = 1;
    }
  }
  else if (flrnum == 2)                 //Run if elevator is requested for on the 2nd floor
  {
    if (SensorValue(fl1)==on)           //Run only if elevator is on 1st floor
    {                                   //Since the elevator is on the 1st floor and the 2nd floor is requested
      motor(winchmotor)=speed;            //Make motor move upwardsx
    }
    else if (SensorValue(fl3)==on)      //Run only if elevator is on 3rd floor
    {                                   //Since the elevator is on the 3rd floor and the 2nd floor is requested
      motor(winchmotor)=speeddown;            //Make motor move downwards
    }
    untilTouch(fl2);                    //Wait until 2nd floor is reached
    flr = 2;
  }
  else if (flrnum == 3)                 //Run if elevator is requested for on the 3rd floor
  {
    if (SensorValue(fl3)==off)          //Run only if elevator is not already on the 3rd floor
    {
      motor(winchmotor)=speed;            //Make motor move upwards
      untilTouch(fl3);                  //Wait until 3rd floor is reached
      flr = 3;
    }
  }
  motor(winchmotor)=off;                //Stop motor once elevator is at the requested floor
}

task main()
{
  motor(winchmotor)=off;                //Make sure motor has stopped before anything continues
  moveFloor(1);                         //Make sure elevator initiallizes on the 1st floor
```

```
while(true)                          //Infinite loop
{
  if (SensorValue[button1] == on || SensorValue[button1b] == on)      //If 1st floor is requested
    moveFloor(1);                                                     //Run function to move elevator to 1st floor
  else if (SensorValue[button2] == on || SensorValue[button2b] == on) //If 2nd floor is requested
    moveFloor(2);                                                     //Run function to move elevator to 2nd floor
  else if (SensorValue[button3] == on || SensorValue[button3b] == on) //If 3rd floor is requested
    moveFloor(3);                                                     //Run function to move elevator to 3rd floor
  else if (theTimer >= 5000 && SensorValue[fl1] == off)               //If 5 seconds (500 milliseconds) has passed
                                                                      //without a button being pressed and elevator is not on floor 1
    moveFloor(1);                                                     //Run function to move elevator to 1st floor
  else if (theTimer >= 5000 && SensorValue[fl1] == on)                //if 5 seconds (500 milliseconds) has passed
                                                                      //without a button being pressed but elevator is already on floor 1
    theTimer = 0;                                                     //Run function to move elevator to 1st floor
  else                               //Only executes if nothing is done and 5 seconds have yet to pass
    wait1Msec(1);                    //wait 1 millisecond
    theTimer += 1;                   //Increase timer by 1 (millisecond)
 }
}
```

Essentially this code take each limit sensor as an input for where the elevator is located. Each limit switch represents its corresponding floor. When a call button for the elevator is pressed, the elevator moves towards that floor until the limit switch is triggered. The elevator waits on that floor for 5 seconds until it is called to another floor. If another call button is initiated then the elevator moves to that location. If the timers times out, then the elevator returns to the ground floor. This entire system is looped so it is never-ending.

Key Contributors

Ekansh Agrawal -

During this project, I was in charge of building the physical model and assisting in optimizing the code for the robot. I helped implement design changes throughout the building process to ensure maximum efficiency for our model. I also assisted with increasing the efficiency of the code by utilizing more functions and user-generated inputs.

Adam Garza -

During this project I was tasked with helping build the team's physical model, assisting in assembling the team documentation, as well as creating and maintaining the team gantt chart. I completed all of these tasks as well as others when they arose, such as finding various solutions or brainstorming solutions to solve challenges in the design of our model.

Justin Thai -

I helped revise the flowchart we would use for the program of our design. Using this flowchart, I created the software of our elevator in RobotC, and also troubleshot it to fix the errors in the program.