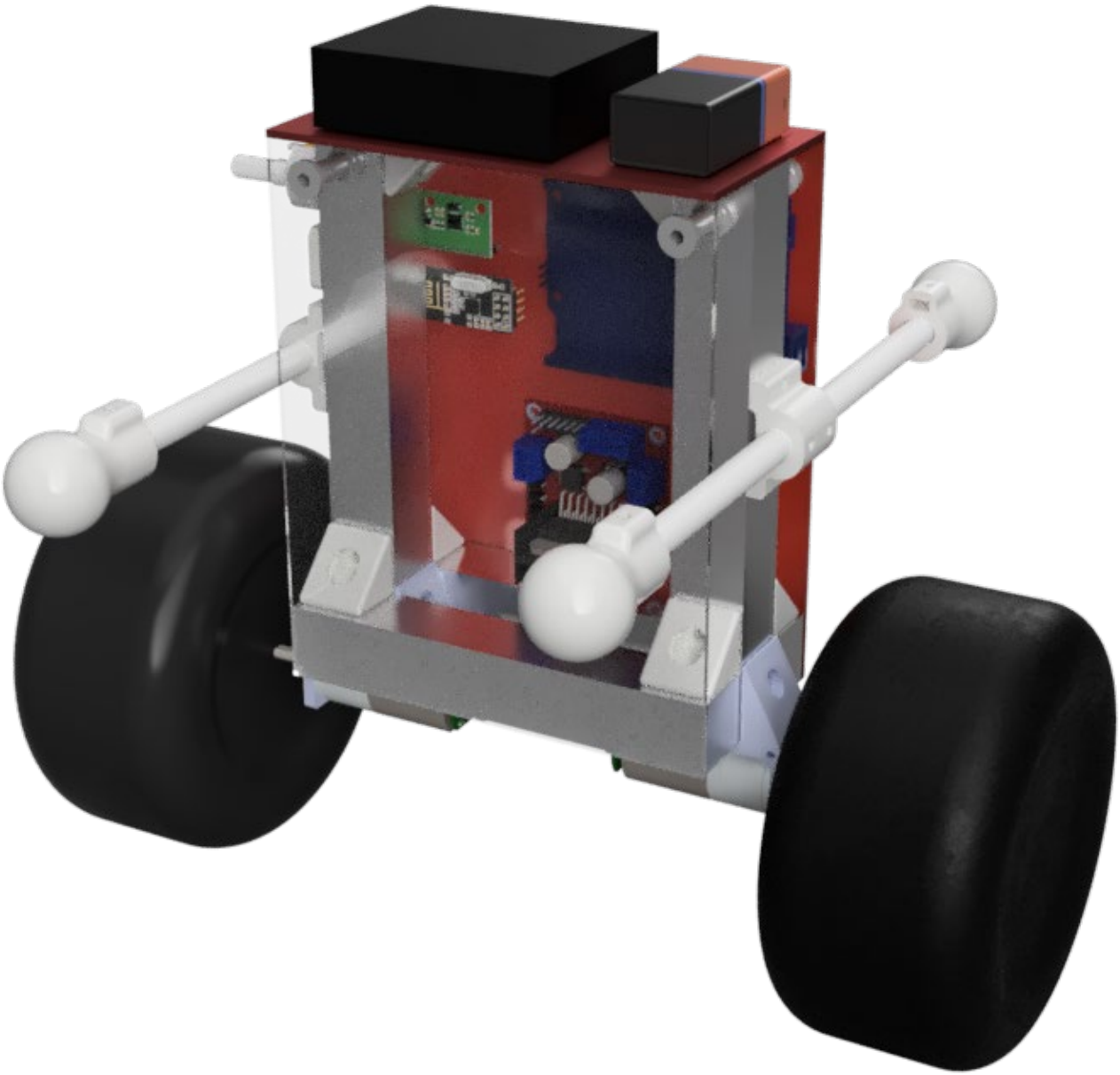


Noah Zipin

Engineering Portfolio

noahzipin@gmail.com

TABLE OF CONTENTS

INTRODUCTION

Hello! My name is Noah Zipin and I am currently a second-year student pursuing a Mechanical Engineering degree at the University of Maryland, College Park. This portfolio is designed to supplement my resume and demonstrate my practical experience in engineering. I have had the opportunity to participate in many projects, both in and out of the classroom, which have not only improved my technical skills but also my ability to think critically and solve problems efficiently. It is my hope that this will allow you to better assess how my skills can be applied to your company. Thank you for your time and consideration!

ABOUT ME

I've always had an innate curiosity to figure out how things work. From my time as a kid taking apart everything in the house, to my teenage years building robots for high stakes competition, my love of creating has always been the driving force. After overcoming a tragic accident in 2015 that resulted in a chronic neck injury, I learned the meaning of humility and found what it takes to persevere. I began to love teaching, and often spent much of my time mentoring young robotics teams where I eventually founded and coached my own team at a local elementary school. My insatiable curiosity and passion for helping others is what fuels my love of engineering. I want to use my knowledge and skills to make the world a better place.

Soft Robotic Gripper (current project)

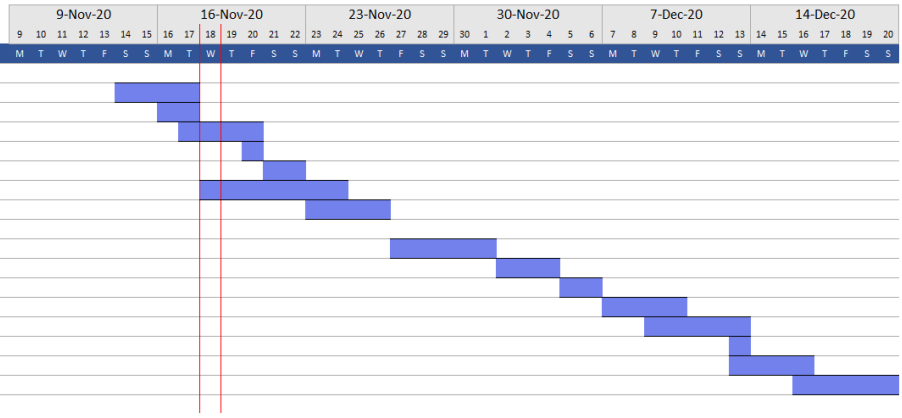
After taking courses like Mechanics of Materials and Materials Science in the A. James Clark School of Engineering this semester, I became highly interested in the nonlinear elastic behavior of cross-linked polymers, like rubber or silicone.

I created this project so that I could investigate the material properties of soft materials and to see if I could simulate complex behaviors such as the visco-elastic effect using finite element analysis.

The goal of the project is to design a soft robotic pneumatically actuated gripper that is capable of grasping objects with complex geometries that would be difficult for traditional "hard" robotic manipulators to deal with (examples include pencil, water bottle, glasses, etc). I'd like to be able to finish the project prior to the start of 2021.

Project Start: Sat, 14-Nov-2020
Display Week: 1

TASK	START	END
Phase I: Making an Actuator		
CAD Mold in SolidWorks	11-14	11-17
Create BOM	11-16	11-17
Acquire Materials	11-17	11-20
Print Mold	11-20	11-20
Assemble Actuator	11-21	11-22
Conduct FEA	11-18	11-24
Record Video	11-23	11-26
Phase II: Making Robotic Gripper		
Research + Make Design	11-27	12-1
CAD Mold in SolidWorks	12-2	12-4
Create BOM	12-5	12-6
Acquire Materials	12-7	12-10
Conduct FEA	12-9	12-13
Print Mold	12-13	12-13
Assemble Actuator	12-13	12-16
Do Electronics	12-16	12-20
Record Video	12-21	12-21



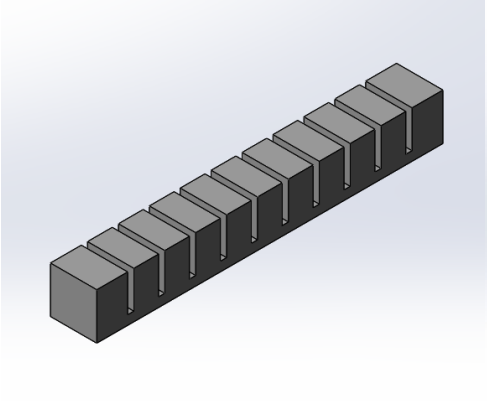
Above is a dynamic Gantt chart I made in Excel to be able to able to adhere to a timeline. The project is designed to have two phases:

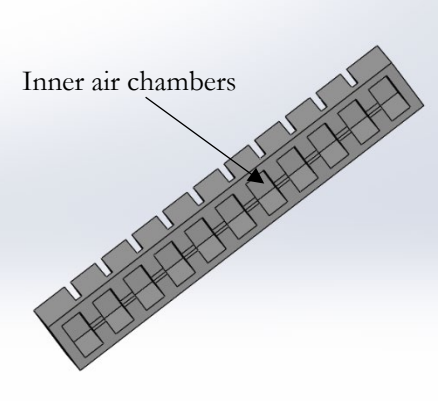
Phase I - To create a single soft actuator made of ECOFLEX 00-35 platinum cure silicone rubber. Here, I'll be utilizing resources from the soft robotics toolkit to create a PneuNets bending actuator, and simulating its deflection as a function of pressure using the Abaqus software suite.

Phase II - To design a gripper based on the actuator from Phase I. I'll be using Abaqus to adjust geometries and conduct FEA to achieve the deflection I want without having to make several prototypes. I'll also be using Arduino to control air flow to the gripper so it can be electronically actuated.

The PneuNets actuator was originally developed by the Whitesides Research Group at Harvard University. It operates by inflating an elastomer with inner air chambers. The elastomer will expand in areas that are the thinnest, and deflection will compound to create a bending motion.

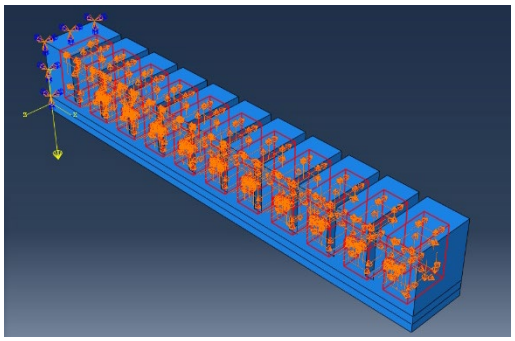
This concept can be seen in the cross section of an inflated PneuNets actuator at left. Here the thinnest walls of the elastomer expand which results in a net bending motion.

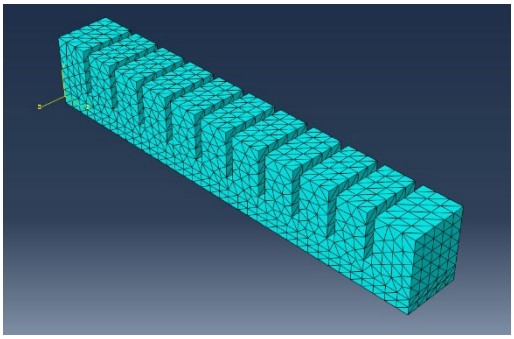




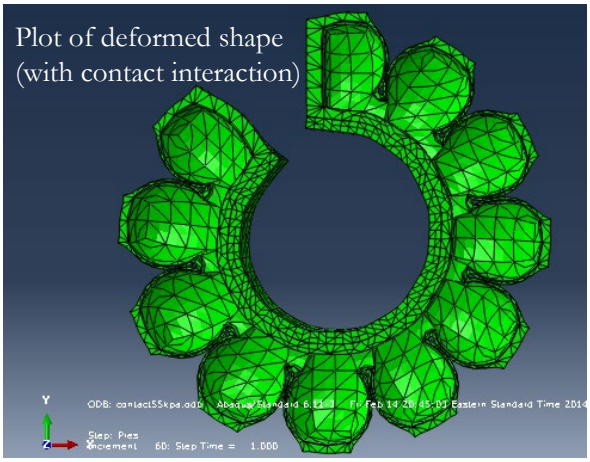
I started by creating simple geometries for a soft actuator with dimensions suggested by the soft robotics toolkit in SolidWorks. Section views of the internal air chambers can be seen in the images at left.

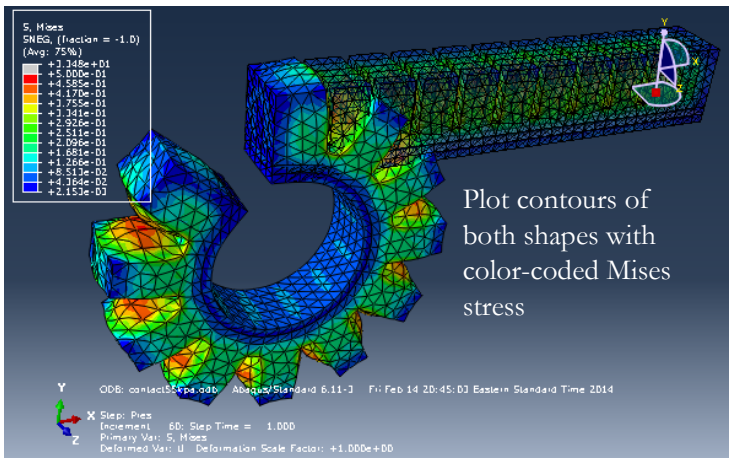
I then used Abaqus CAE to conduct finite element analysis (FEA) to see the actuator deflection with pressure and gravitational loads. It is important to simulate these actuators to see their behavior before fabrication so there's less of a need to make multiple prototypes. However, simulating the non-linear elastic behavior of the EcoFlex silicone is quite difficult, and is the main reason I'm doing this project. I've only ever conducted FEA on rigid components, so I'm excited to learn this new skill.



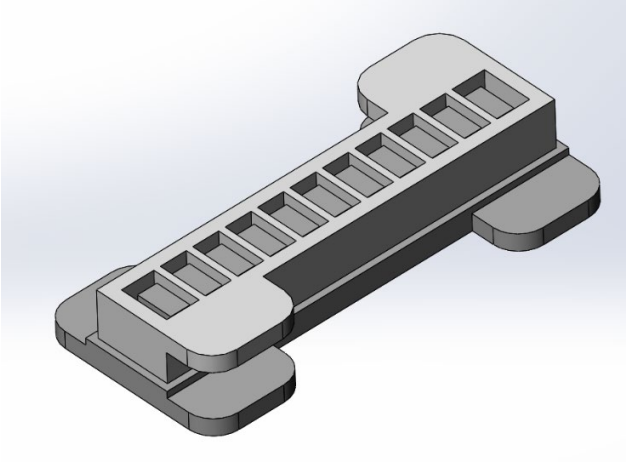


I created boundary conditions and applied loads. The left end of the beam is fixed. Uniform pressure (8 psi) acts on all surfaces of the internal chamber. Since the actuator acts in the vertical plane, I've also included gravity as a force that acts on the actuator. I created the mesh (right, above) and ran the analysis:

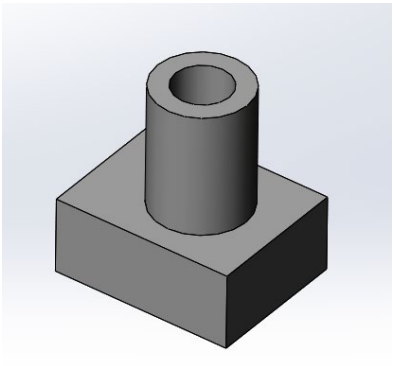




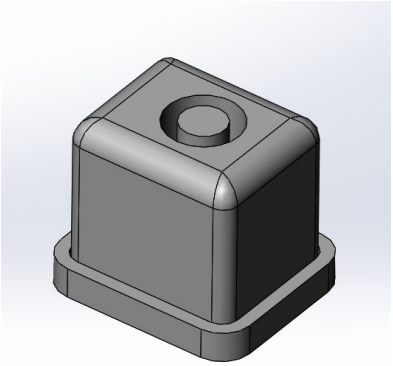
Above are the results of the simulation. It is worth noting that I included a strain-limiting layer (in this case, paper) to ensure uniform bending. I will include a paper layer in the actual model.



Since I was happy with the geometry and morphology of the current model, I moved forward and created molds in SolidWorks, which can be seen above. This is based on the original actuator with slightly more complex geometry so that the silicone can more easily bond with itself in the casting process.



Adapter for air source



Mold

To attach the soft robotic actuator to an air source, I designed a part in SolidWorks to be cast out of silicone that would snugly fit the 6 mm outer diameter tubing.

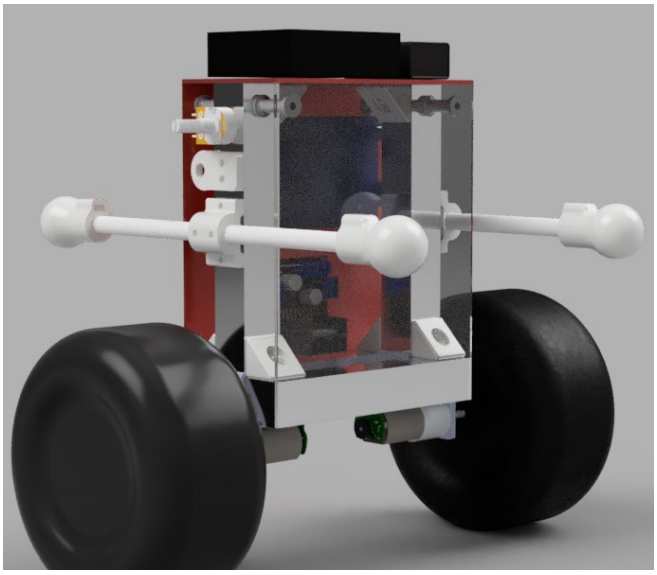
Below are the results of the completion of Phase I of the project:

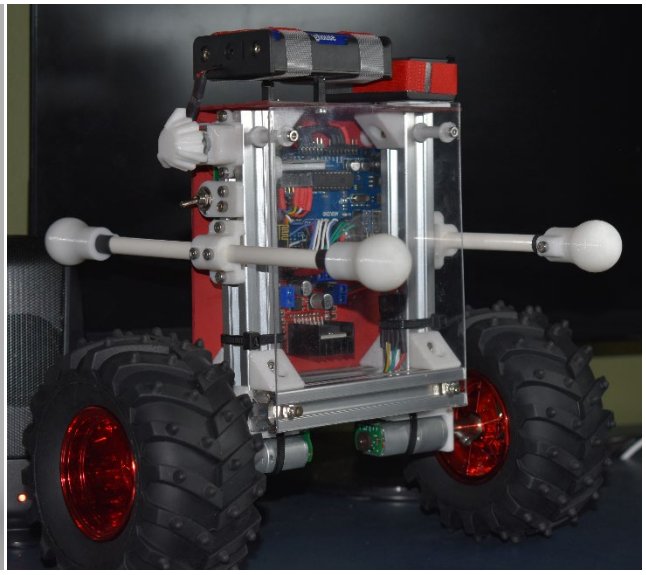
Due to casting difficulties with ECOFLEX 00-35 silicone, I had to reduce the total number of sections from 11 to 8. But deflection is the same with increased pressure.

A **demonstration** of the single actuator working can be seen be clicking this link:

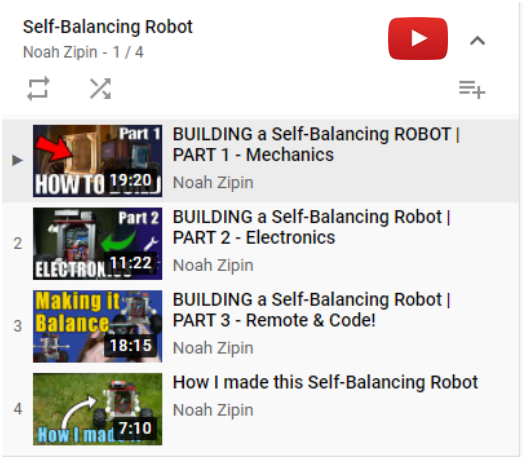
https://www.youtube.com/watch?v=M8f4WjTySG0&ab_channel=NoahZipin

2 Wheeled Self-Balancing Robot





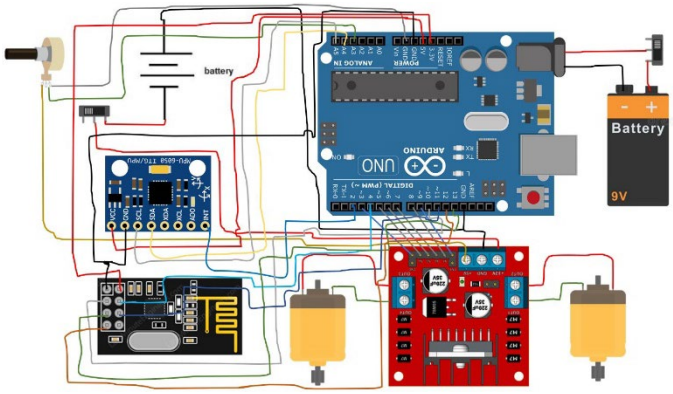
I chose to start this project to learn new skills during the extra time I had due to the pandemic. Here, I've designed, manufactured, and controlled 100% custom systems to make a robot that balances on two wheels. This was a highly multidisciplinary project, and I've learned so much about circuit design, control theory, dynamic systems theory, systems integration, and more. And, I've been able to improve on some of my other skills, like CAD modeling, project planning, and presentational speaking.

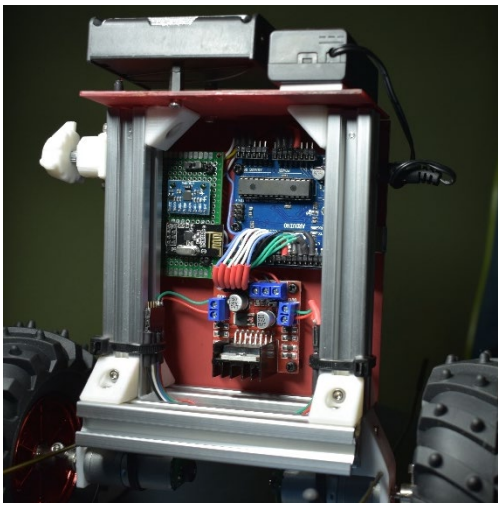


On top of making the robot, I've also made a full in-depth YouTube series that describes the entire process and explains all aspects of the design and theory. My latest video, which showcases the robot fully working, can be found here:

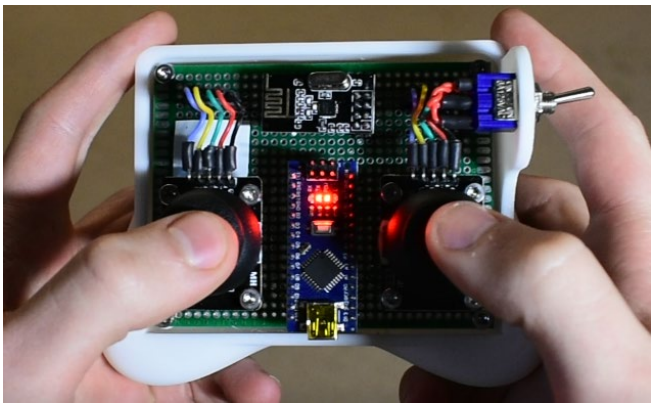
https://www.youtube.com/watch?v=PNKXGorAnBg&list=PLmEi2OoxXIQ88akwk1nRoXV6LqhXQp3kH&index=4

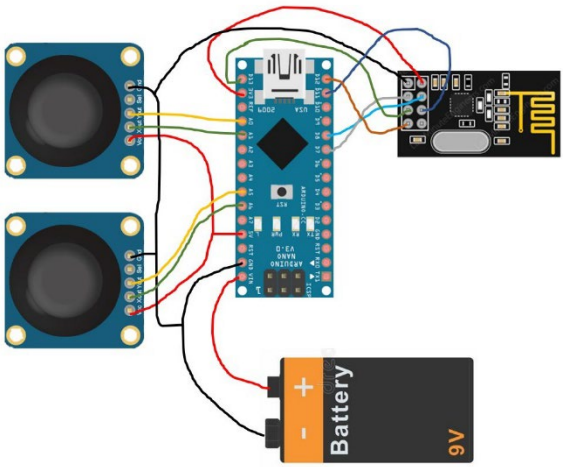
I've made also this project entirely open-source and posted all CAD, code, and circuit schematics on my GitHub: https://github.com/nzipin



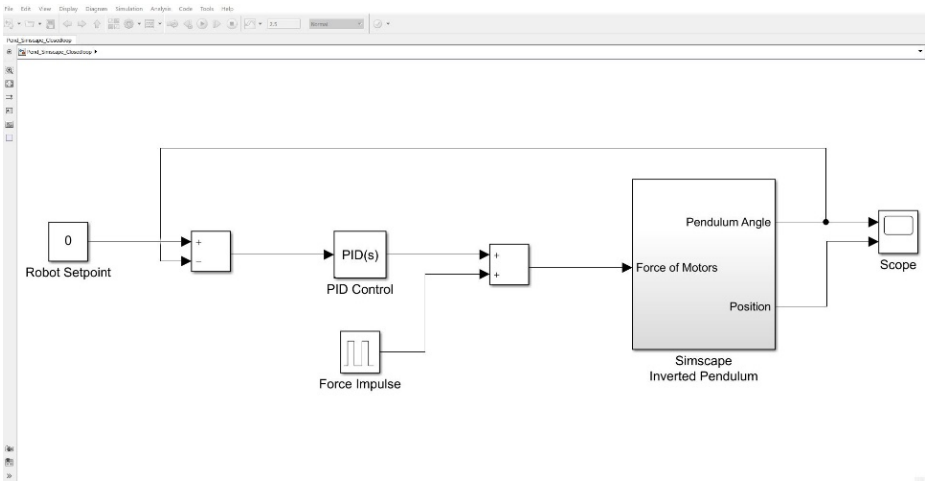


Rudimentary wiring diagrams soon turned into a highly complex system that needed to be integrated with the existing mechanical design. I was able to learn about how each board worked, and how each communicated to the Arduino microcontroller.





I researched and implemented 12C, UART, and SPI connections and used PWM on the dual H-bridge of the motor driver to control the motors. I was also able to practice different soldering techniques and learn about PCB design while assembling some components on dot prototyping board.



To design control systems for the robot, I utilized Simulink and created the block diagram seen at left. Luckily, the self-balancing robot mimics the inverted pendulum on a cart dynamic system, which is a classic control problem for stabilizing non-linear dynamics. I was able to use a controls plant provided through a collaboration between the University of Michigan, Ann Arbor and Carnegie Mellon University, which can be found here.

I was able to use this plant to get a general sense of optimal PID values, and further tuned the system via trial and error.

Programming the robot was done using C++ (Arduino) to set motor velocity based on the angle of the robot via a PID control algorithm. Driving the robot manually is done by altering the setpoint value of the robot (see block diagram) and letting the robot "fall" forwards or backwards.

All code is available in the open source on my GitHub, but snippets can be seen below:

```
/********** PID **********/
#include <PID_v1.h>

double kP = 24;
double kI = 350;
double kD = 1.2;

double setpoint, input, output;   // PID variables
PID pid(&input, &output, &setpoint, kP, kI, kD, DIRECT); // PID setup

/********** radio **********/
#include <nRF24L01.h>
#include <SPI.h>
#include <RF24.h>
#include <nRF24L01.h>

RF24 radio(10, 4);    // CE, CSN
const byte address[6] = "00001";
float angleV = 0, turnV = 0; // values from remote
float controlValues[2]; // array to receive both data values

/********** L298N **********/
#include <L298N.h>

// Pin definition
const unsigned int EN_A = 3;
const unsigned int IN1_A = 5;
const unsigned int IN2_A = 6;

const unsigned int IN1_B = 7;
const unsigned int IN2_B = 8;
const unsigned int EN_B = 9;

// Create motor instances
L298N rightMotor(EN_A, IN1_A, IN2_A);
```

Above, the beginning of the code on the Arduino Uno, setting up PID control algorithms, radio interface, and motor instances

At right, the interrupt service routine to read accurate data from the 6DoF inertial measurement unit

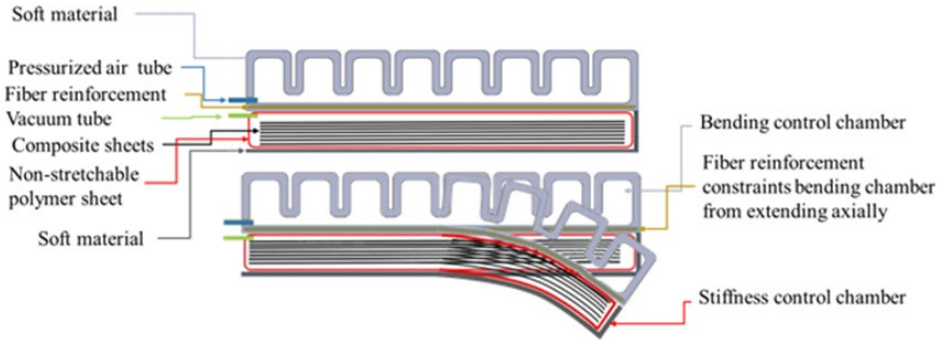
```
void loop() {
  int forwardValue = analogRead(A1);
  int turnValue = analogRead(A5);

  int mapVal1 = map(forwardValue, 0, 1023, -512, 512);
  float angleValue = (float) mapVal1/200;

  int mapVal2 = map(turnValue, 0, 1023, -512, 512);
  float turnAngleValue = (float) mapVal2/5;

  controlValues[0] = angleValue;
  controlValues[1] = turnAngleValue;

  radio.write(&controlValues, sizeof(controlValues));

  Serial.print("forward: ");
  Serial.print(angleValue);
  Serial.print( " turn: ");
  Serial.println(turnAngleValue);
}
```

Above, the code uploaded to the remote control and data read via SPI connection.

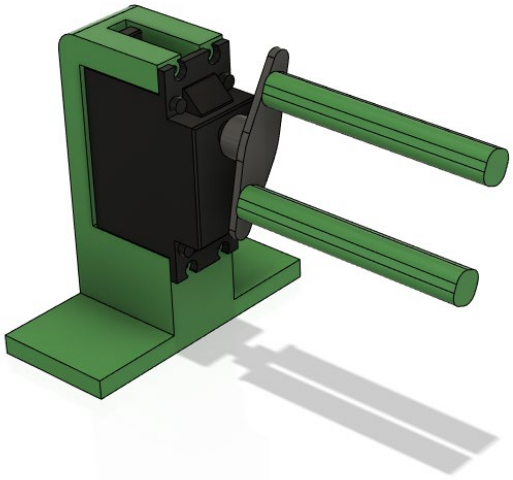
```
void readAngles()  {

    mpuIntStatus = mpu.getIntStatus();
    fifoCount = mpu.getFIFOCount();

    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));
     }

     else if (mpuIntStatus & 0x02) {
        // wait for correct available data length, should be a VERY short wait
        while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1 packet available
        // (this lets us immediately read more without waiting for an interrupt)
        fifoCount -= packetSize;

        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

        IMUdataReady = 0;
        //count = count + 1;
    }
}
```

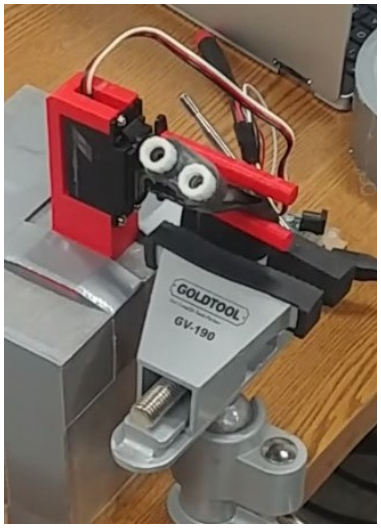
BAM Lab Flexible Actuators

This project is currently on hiatus due to the pandemic. Since the fall of 2019, I've been a member of the Bio-Inspired Advanced Manufacturing (BAM) Lab at the University of Maryland, College Park. Prior to the outbreak, my team and I worked on developing novel flexible actuators that utilized laminar jamming to achienve variable stiffness. The actuators were 100% 3D printed on the Connex3 poly jet printer, which is capable of integrating both rigid and flexible materials for an all-in-one design.



At left is the concept of laminar jamming and how it can vary the stiffness of a soft actuator. Layers are vacuum sealed which artificially creates higher bending yield.



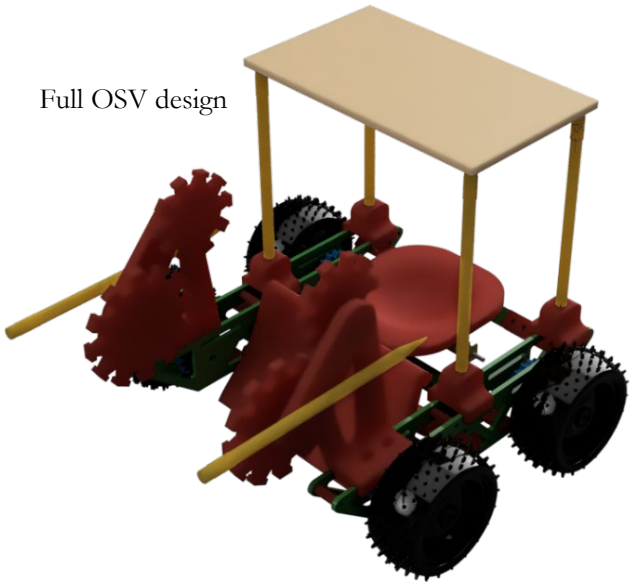



```
#include <Servo.h> //includes Servo library

int cycle = 5; //sets speed of servo
int wait = 500; //time between oscillation
int value; //sets value of servo position
Servo forkServo; //declares servo

void setup() {
forkServo.attach(11); //attaches servo to pin 11
}

void loop() {
  for(value = 0; value<=180; value++){ //turns servo to 180 degrees
    forkServo.write(value);
    delay(cycle);
  }
  delay(wait);
  for(value = 180; value>=0; value--){ //turns servo to 0 degrees
    forkServo.write(value);
    delay(cycle);
  }
  delay(wait);
```

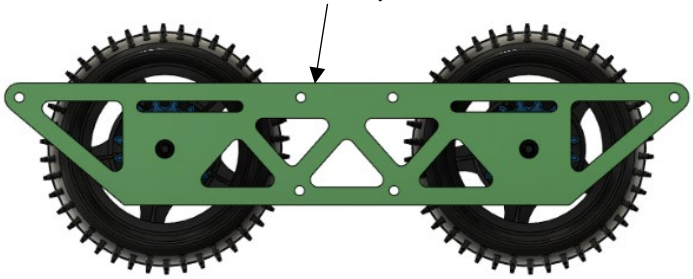
Above is the custom agitation mechanism I designed and simulated in SolidWorks. Excess support material clogs the actuator a result of the printing process on the poly jet 3D printer, and it needs to be drained out with a sodium hydroxide solution. Due to the intricate nature of the jamming layers, the actuator needed to be continuously agitated to prevent layers from sticking together. Pictured above is initial tests of the agitation mechanism working.

OSV Design Project

Full OSV design



Modular acrylic chassis

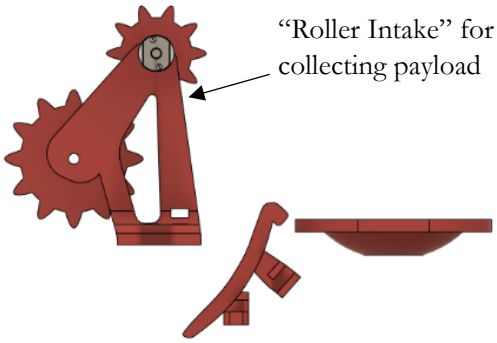


This was an in-class project for *ENES100: Introduction to Engineering Design* that I did during the Spring 2020 semester. Due to the pandemic, we were unfortunately unable to fully realize the design, although we had everything ready for fabrication. The goal of the project was to design an over-sand vehicle (OSV) capable of autonomously navigating a randomized course to a specific site, and able to identify the weight and material of a payload (a sphere, in this case).

Acting as mechanical team lead, myself and 8 other students collaborated to design and manufacture a WIFI controlled robot for a competition between around 50 other groups within the university. Being one of two members on my team with previous CAD experience, a large part of my job was teaching other students about modeling in SolidWorks and Fusion 360.

I designed a modular chassis, seen above, in which components could be clipped and zip-tied on, making design iterations simple and efficient for rapid prototyping. It uses two direct-drive motors at the rear of the OSV and chain running between the parallel plates to drive the front wheels. Here I was able to provide .dxf files using Adobe Illustrator for fabrication on the laser cutter.

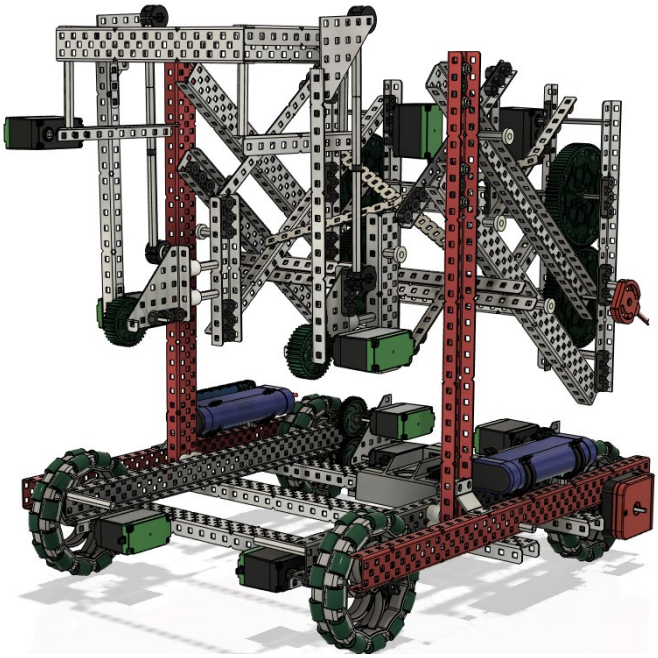
"Roller Intake" for collecting payload

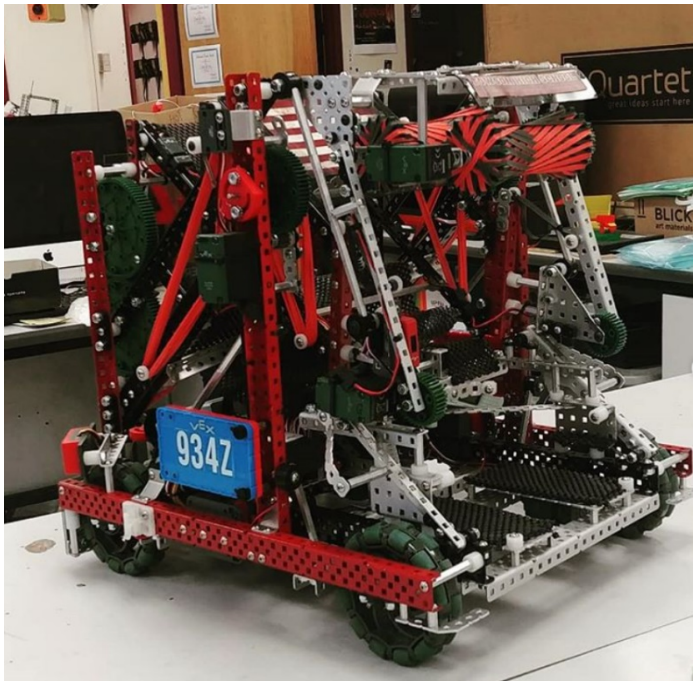


Our team also designed a novel "roller intake," seen at left, which was used to collect the payload with high tolerance. All of these components are 3D printed and simply clipped to the modular acrylic chassis, making it easy to iterate the design.

Apart from the mechanical team, I also played an integral role in designing logic for autonomous navigation. I designed a pseudocode flowchart which describes a clever method of navigation, without using any on-board sensors. Since our OSV's position can be determined using a Wi-Fi-enabled overhead computer vision system, we were able to determine if our OSV ran into an obstacle by calculating the OSV's current velocity and comparing it to the expected velocity of our OSV. If our current velocity was much less than our expected velocity, we knew we ran into an obstacle and acted accordingly.

VEX Robotics Competition Robot





The VEX Robotics competition is a high stakes tournament where robots compete in a point-based game against each other. Robots compete with a randomized opponent in several 2-minute matches throughout the tournament. The robot seen above is my team's VEX "In the Zone" robot. The objective of this game is to stack as many yellow cones as possible on "mobile goals" and place the mobile goals (with stacked cones on them) into your team's scoring zone.

I acted as the founder and president of the Towson Robotics program from mid-2015 – 2019. I performed most of the mechanical design and assembly of the robot seen above over the course of a competition season, which is about 8 months. I also acted as programming lead, where we coded in a variant of C called RobotC for both driver and autonomous modes.

The drive train is composed of six "VEX 393" motors mechanically linked together with 1:1 sprockets, chain, and gears. The drive train uses two hall effect quadrature encoders for positional control using PID algorithms. The frame has two ultrasonic distance sensors that are used during autonomous navigation to align with the perimeter of the field. The lifting mechanism is two reversed, cascaded parallel linkages that are capable of reaching a minimum height of 16 inches and a maximum height of 57 inches. The lift uses 2 potentiometers that utilize PID algorithms to achieve accurate preset heights that are useful when creating large stacks of cones.

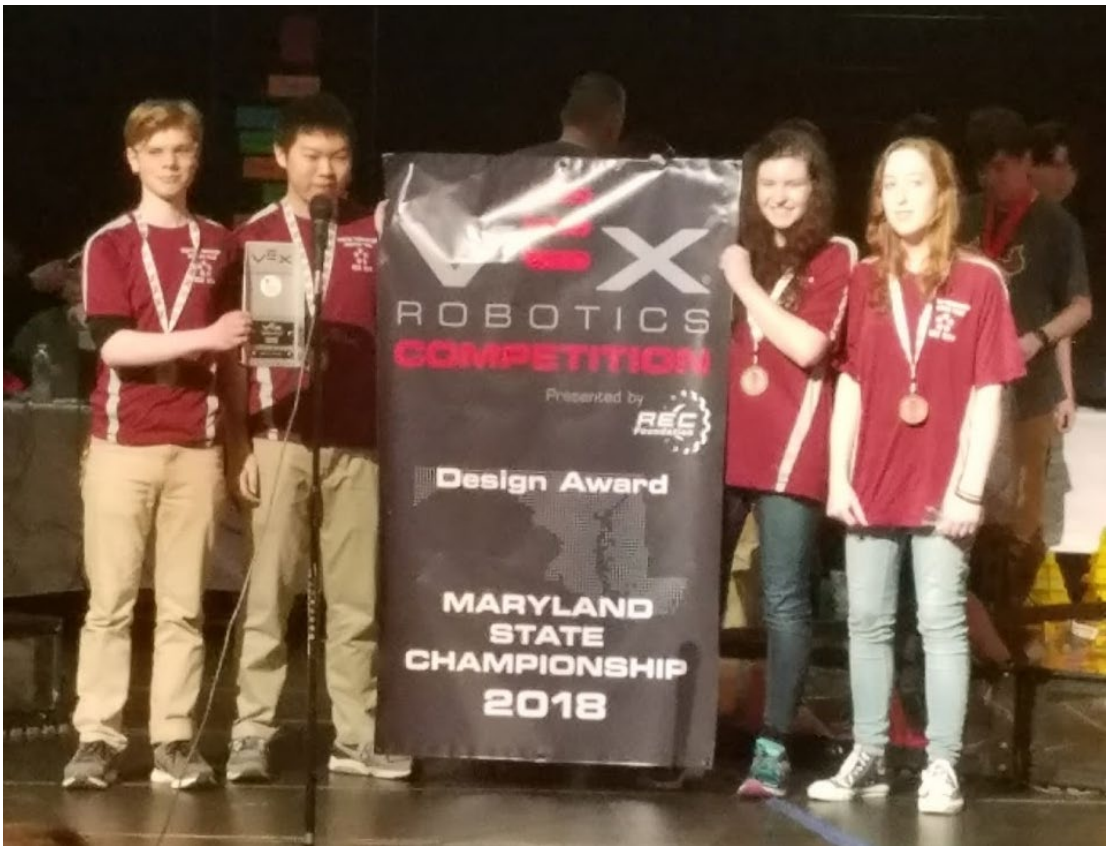
A YouTube video showcasing the robot in action can be found here:
https://www.youtube.com/watch?v=FwU0mx70mjg&t=73s&ab_channel=NoahZipin

Above is the robot fully extended, which reaches a height of almost 5' tall. CAD vs. Reality

```
task forwardLeft_pid()
{
  SensorValue(leftDrive) = 0;
  while(true)//(SensorValue(leftDrive) < FLsetpoint)
  {
    errorFL = FLsetpoint - SensorValue(leftDrive); //calculate for error

    integralFL = integralFL + errorFL;//integral calculation
    if (errorFL == 0 || (FLsetpoint < SensorValue(leftDrive))){//? if error is 0 or you
      integralFL = 0;
    }
    if (errorFL > integralActiveZoneFL){//zone for which integral can accumulate error
      integralFL = 0;
    }

    derivativeFL = errorFL - prevErrorFL; //derivative calculation
    prevErrorFL = errorFL;

    speedFL = errorFL*kp + integralFL*ki + derivativeFL*kd; //power sent to motors

    motor[backLeft] = motor[frontLeft] = speedFL; //assign new motor value to motors

    wait1Msec(20); //time for each cycle
  }
}
```

```
task whichOne()  //auton switcher with Potentiometer
{
  if(SensorValue(LeftRight) <= 2047)
    Left = true;
  else if(SensorValue(LeftRight) > 2047)
    Right = true;

  if(SensorValue(TwTtSg) <= 1365)
    Twenty = true;
  else if(SensorValue(TwTtSg) > 1365 && SensorValue(TwTtSg) <= 2730)
    Ten = true;
  else if(SensorValue(TwTtSg) > 2730)
    Stationary = true;
}

void pre_auton()
{
  SensorValue(leftDrive) = 0;
  SensorValue(rightDrive) = 0;
  startTask(whichOne);
}
```

Here are two short code segments written in RobotC for our competition robot. The left image shows one of our PID algorithms that uses the hall effect positional sensor on the drive train so that we don't overshoot target distance values during autonomous operation. The right image shows a task that allows us to choose which autonomous routine to run without re-downloading the program, which helped us out a great deal for strategic purposes during tournaments.



Pictured above (myself, at left) is our victory at the Maryland VEX Robotics State Championship in 2018, where our team was awarded the "Design Award" for outstanding documentation of the engineering design process, excellent on-field performance, and quality code.

This was a proud moment, and is the highest award given to any team in the tournament. The award qualified our team to compete in the VEX Robotics World Championships, in which we competed in Louisville Convention center in head-to-head matches against over 50 other countries from around the world. We were proud to represent the state of Maryland.

Over the course of my VEX Robotics career, my team and I were able to compete in the VEX World Championships three times, in 2017, 2018, and 2019. It was a rewarding experience.