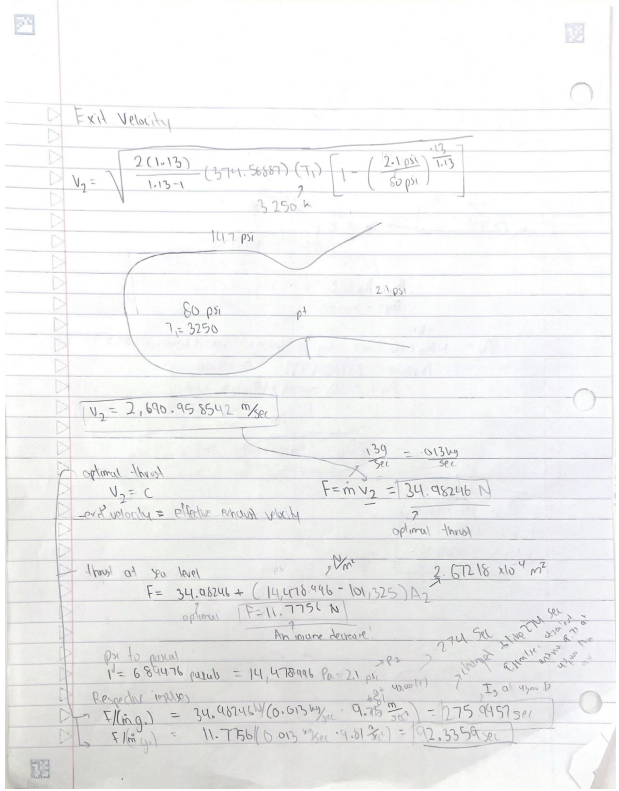
White Horizon PDR

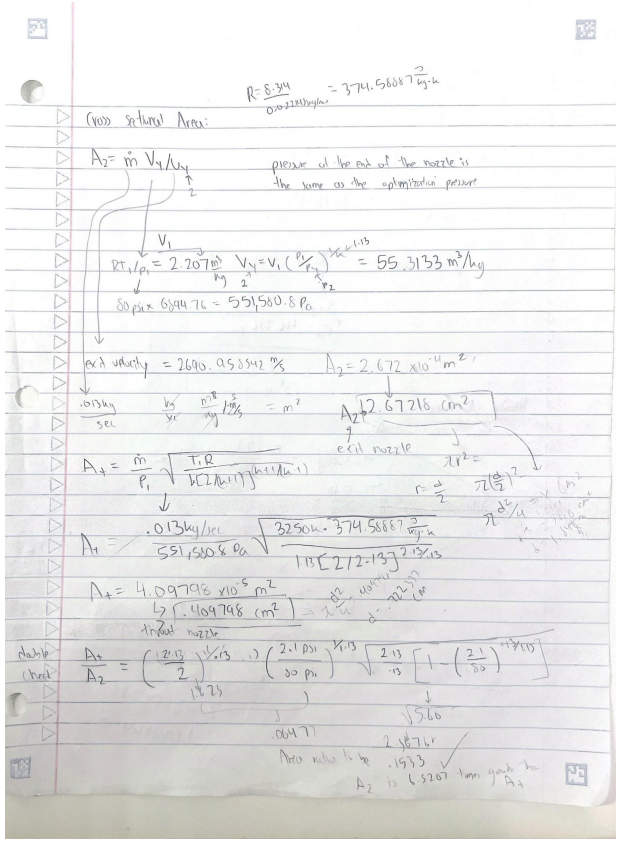
By Anvesh, Moataz, Nicole and Connor

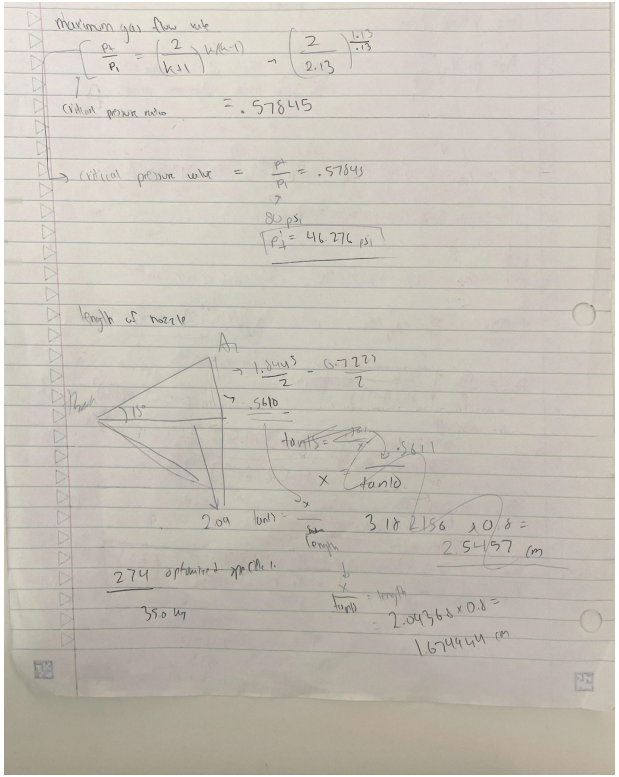
Calculations and Critical Values

	A	B	C	D	E	F
1	**Parameter**	**Metric**	**Units**	**Imperial**	**Units**	**Notes**
2	Optomized Specific Impulse	274	s			
3	Specific Impulse at sea level	92.3359	s			
4	Optimal Thrust	35.0	kgm/s^2	7.864	lbf	
5	Sea Level Thrust	11.8	kgm/s^2	2.647	lbf	
6	Exit Velocity (V2)	2690.95842	m/s	8829.035	ft/s	
7	Critical Pressure Ratio	0.58	unitless			
8	Pressure Value	319.06	kPa	46.276	psi	@ Mach 1
9	Area Ratio	0.153	no units lol			
10	Throat Area	0.410	cm^2	0.064	in^2	circular
11	Throat Diameter	0.722	cm	0.284	in	
12	Exit Area	2.67	cm^2	0.414	in^2	A2
13	Exit Diameter	1.84	cm	0.726	in	
14	Nozzle Length	1.6749	cm	0.659	in	Bell Nozzle

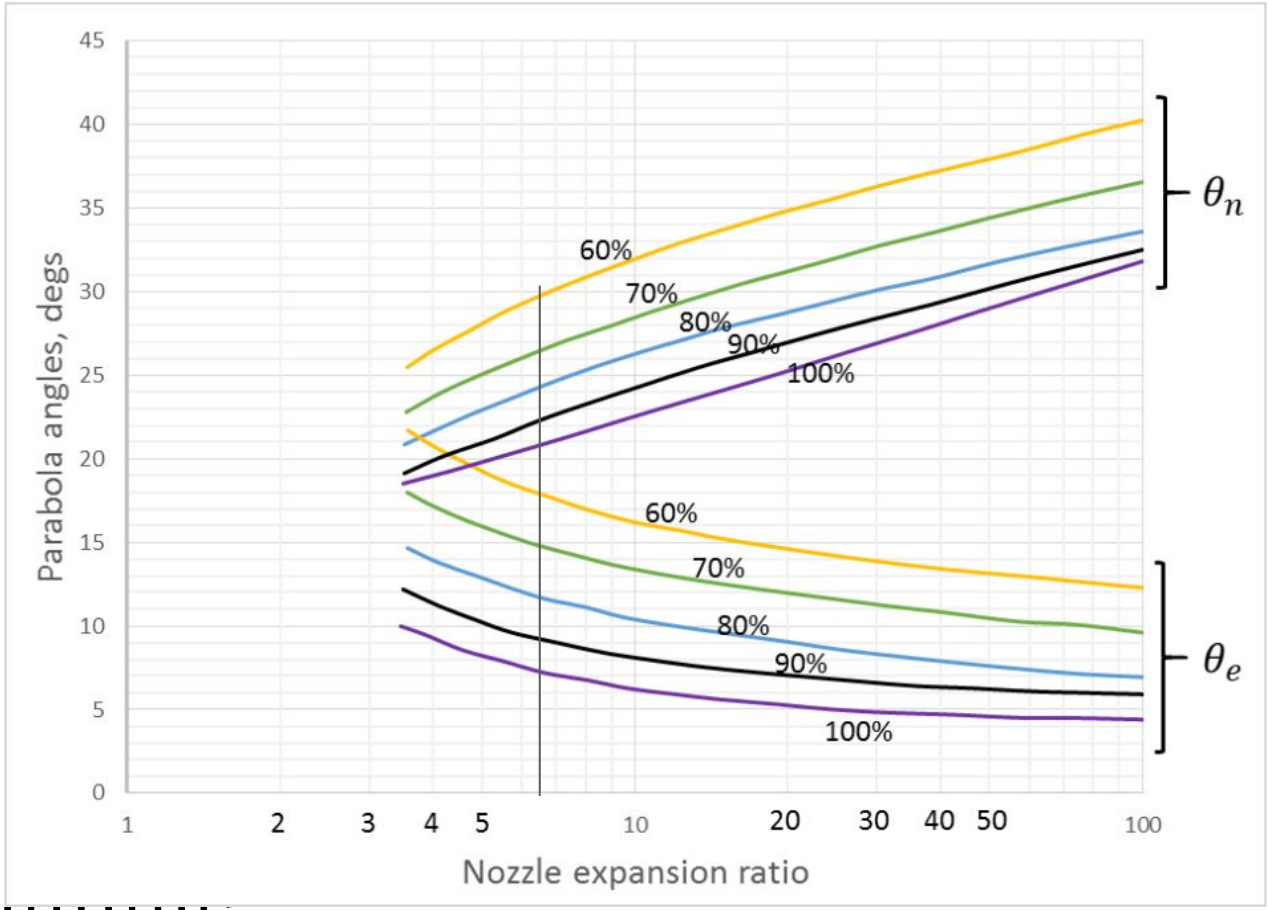
Calculations







Creating a Rao Nozzle and Choosing Approximation Angles



Our Expansion Ratio is: 6.51

Thus we can choose approximation angles of:

Exit Angle: 11.6 degrees

Initial Angle: 24.2 degrees

The Math Behind the Points Found

Technical papers

The mathematical approach

The throat

The equations of the above circular arcs defining the throat are defined trigonometrically, defining the origin of the coordinates as the centre of the narrowest part of the throat:

For the entrant section:

$$x = 1.5\,R_t \cos\theta$$
$$y = 1.5\,R_t \sin\theta + 1.5\,R_t + R_t \quad \textbf{equ.s 4}$$

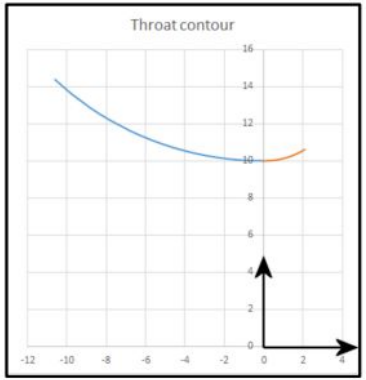
where: $-135 \le \theta \le -90$

(The initial angle isn't defined and is up to the combustion chamber designer, -135 degrees is typical.)

For the exit section:

$$x = 0.382\,R_t \cos\theta$$
$$y = 0.382\,R_t \sin\theta + 0.382\,R_t + R_t \quad \textbf{equ.s 5}$$

where: $-90 \le \theta \le (\theta_n - 90)$



The bell

The bell is a quadratic Bézier curve, which has equations (see Wikipedia):

$$x(t) = (1-t)^2 N_x + 2(1-t)t\,Q_x + t^2 E_x \quad 0 \le t \le 1$$
$$y(t) = (1-t)^2 N_y + 2(1-t)t\,Q_y + t^2 E_y \quad 0 \le t \le 1 \quad \textbf{equ.s 6}$$

Selecting equally spaced divisions between 0 and 1 produces the points described earlier in the graphical method, for example 0.25, 0.5, and 0.75.

Selecting equally spaced divisions between 0 and 1 produces the points described earlier in the graphical method, for example 0.25, 0.5, and 0.75.

Equations 6 are defined by points N, Q, and E (see the graphical method earlier for the locations of these points).

Point N is defined by equations 5 setting the angle to ($\theta n - 90$).

Coordinate E_x is defined by equation 3, and coordinate E_y is defined by equation 2.

Point Q is the intersection of the lines: $\overrightarrow{NQ} = m_1 x + C_1$ and: $\overrightarrow{QE} = m_2 x + C_2$ **equ.s 7**

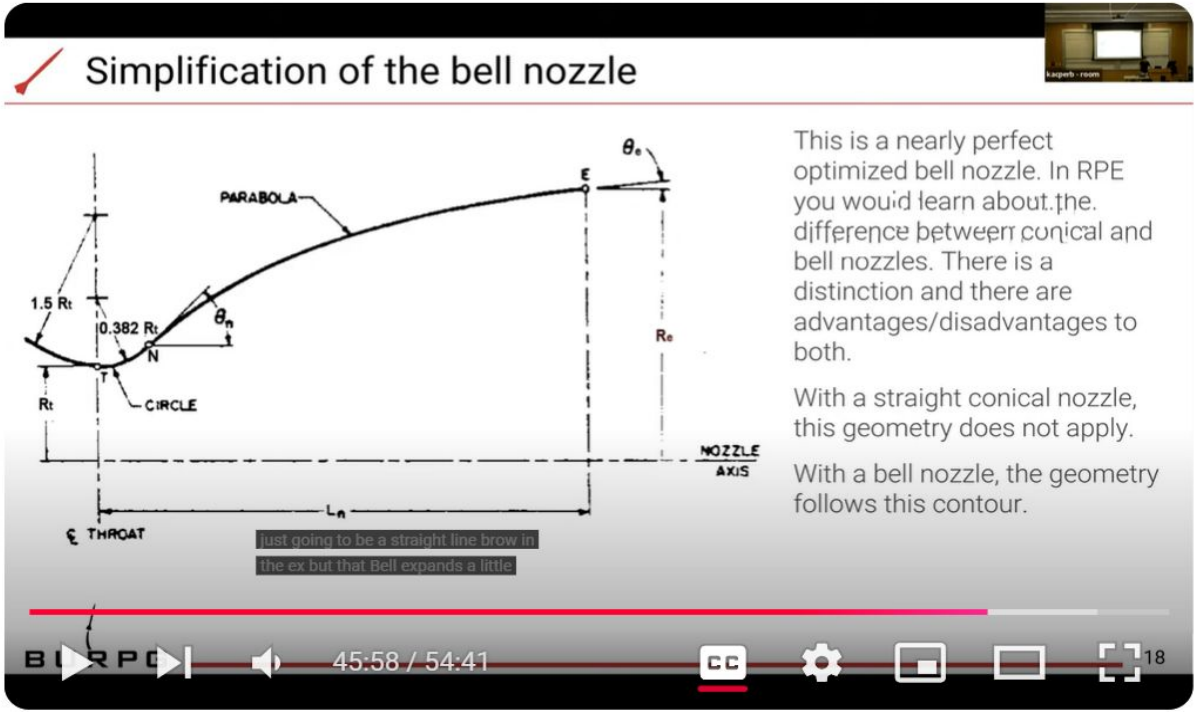
where: gradient $m_1 = \tan(\theta_n)$, gradient $m_2 = \tan(\theta_e)$ **equ.s 8**

and: intercept $C_1 = N_y - m_1 N_x$, intercept $C_2 = E_y - m_2 E_x$ **equ.s 9**

The intersection of these two lines (at point Q) is given by:

$$Q_x = \frac{(C_2 - C_1)}{(m_1 - m_2)},\ Q_y = \frac{(m_1 C_2 - m_2 C_1)}{(m_1 - m_2)} \quad \textbf{equ.s 10}$$

Design of the Bell Nozzle



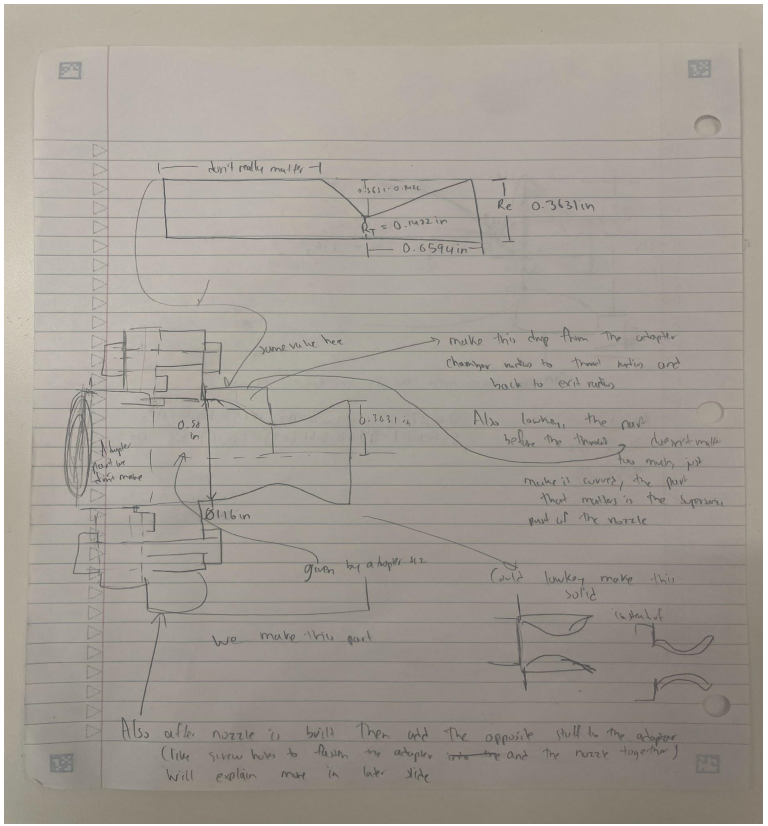
Nozzle Lecture Fall 2024

Unlisted

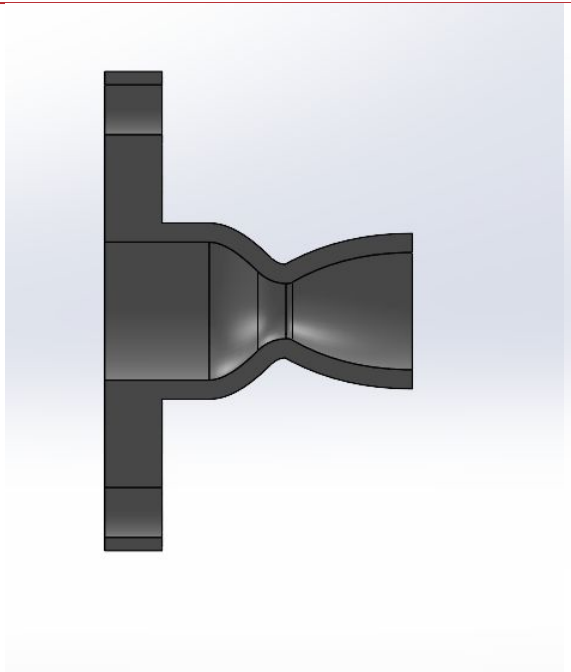
Angles at nozzle and exit as well as the radius of exit, throat, and the length of the nozzle were specified in the spreadsheet above and slide 6

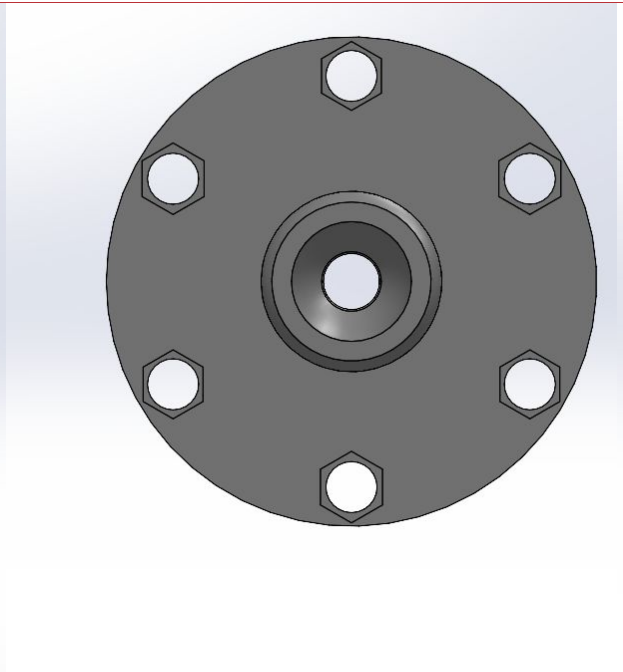
Simple Nozzle Design

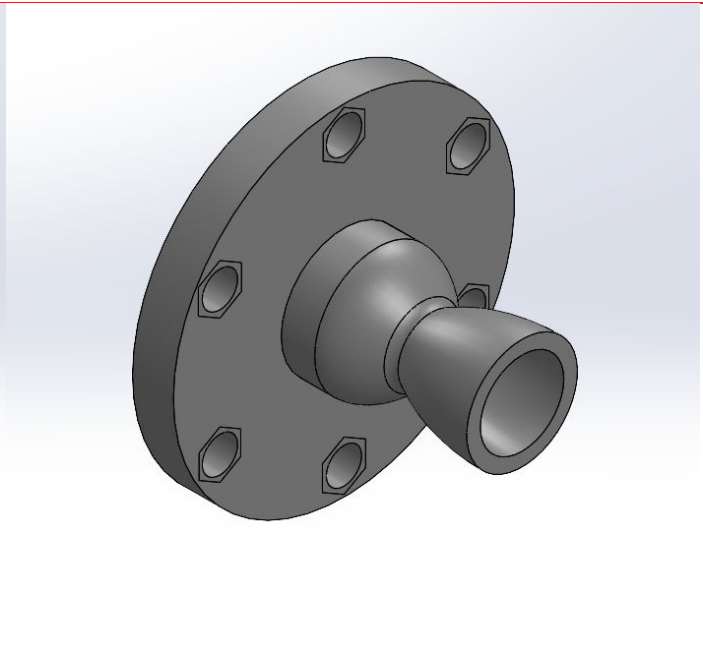
CAD video help



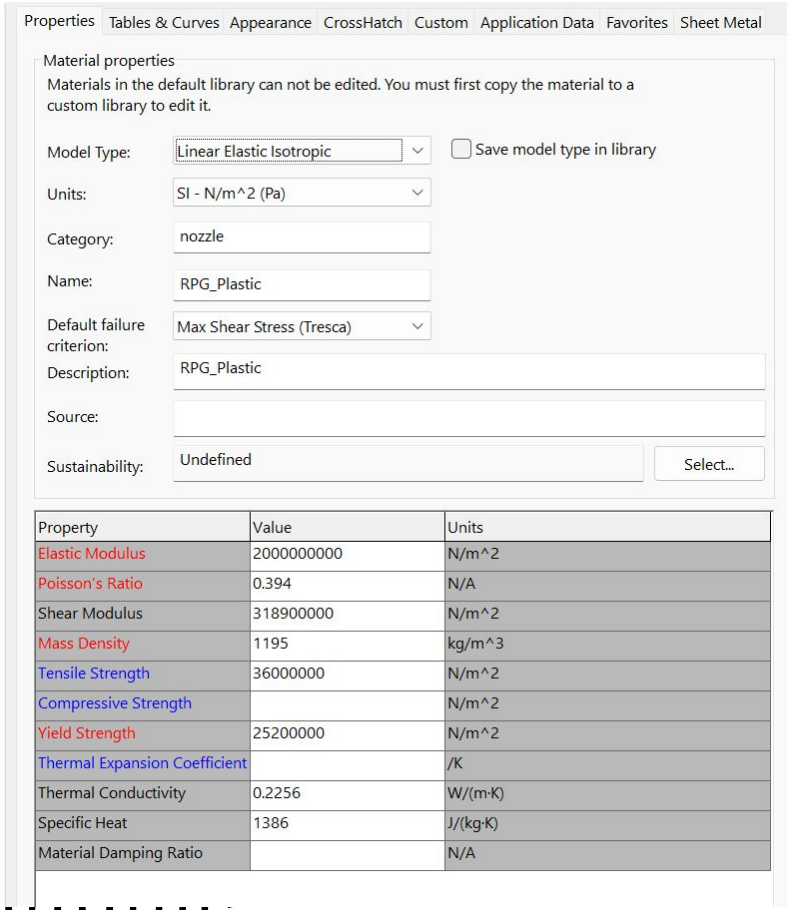
CAD Model of Nozzle







Material Selection and Properties

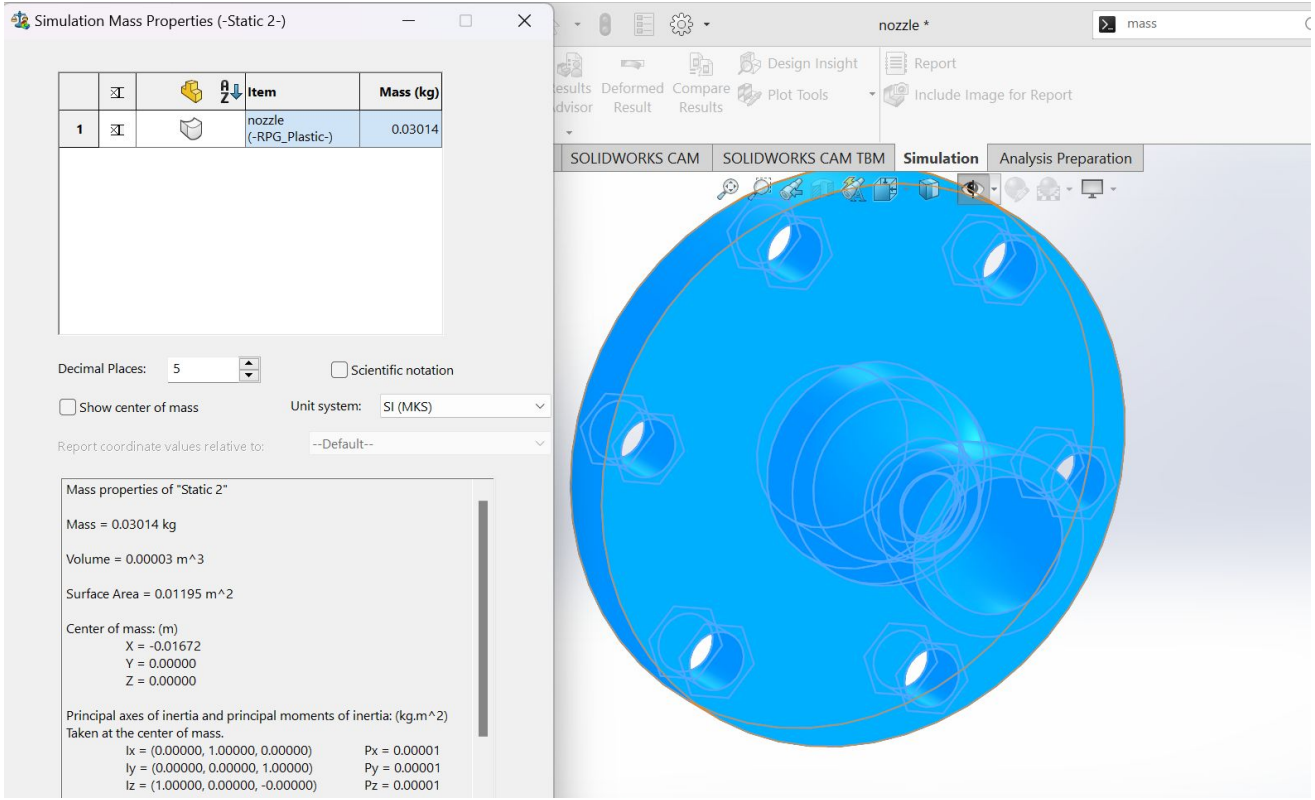


Property	Value	Units
Elastic Modulus	2000000000	N/m^2
Poisson's Ratio	0.394	N/A
Shear Modulus	318900000	N/m^2
Mass Density	1195	kg/m^3
Tensile Strength	36000000	N/m^2
Compressive Strength		N/m^2
Yield Strength	25200000	N/m^2
Thermal Expansion Coefficient		/K
Thermal Conductivity	0.2256	W/(m·K)
Specific Heat	1386	J/(kg·K)
Material Damping Ratio		N/A

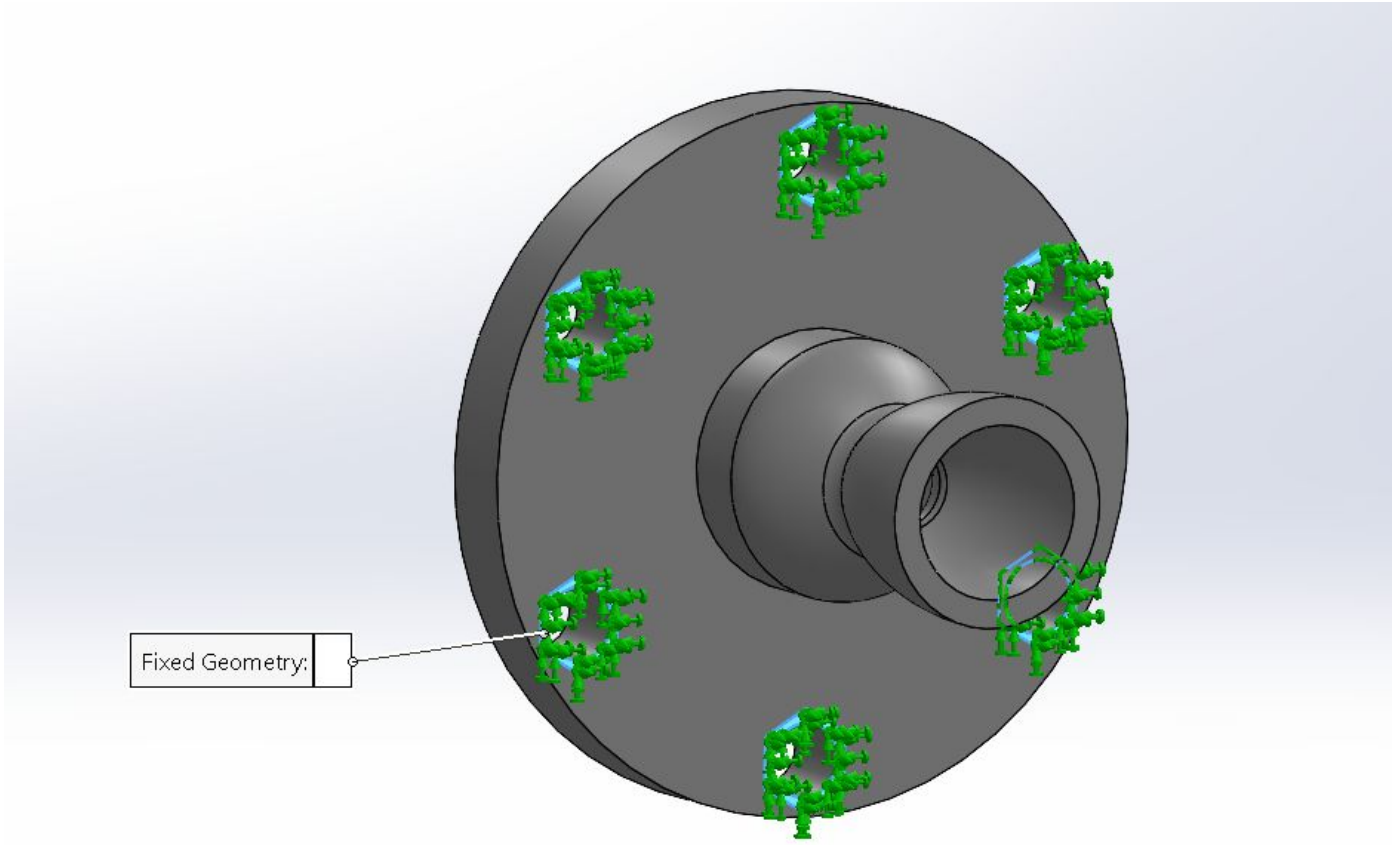
Resin Parameters:

- Resin Type: ABS-like resin V2.0
- Hardness: 84 D; Shrinkage: 7.1 %
- Viscosity (25°C): 150-200 mPa.s
- Liquid Density: 1.100 g/cm³
- Solid Density: 1.195 g/cm³
- Flexure Strength: 59-70 Mpa
- Extension Strength: 36-53 Mpa
- Elongation at Break: 14.2
- Shelf Life: 2 year

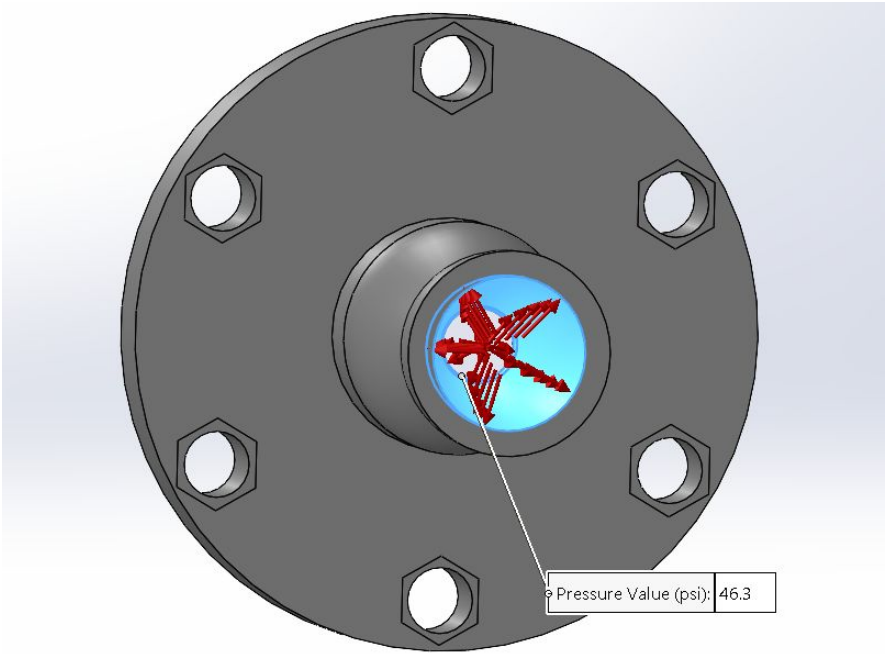
Applied Materials

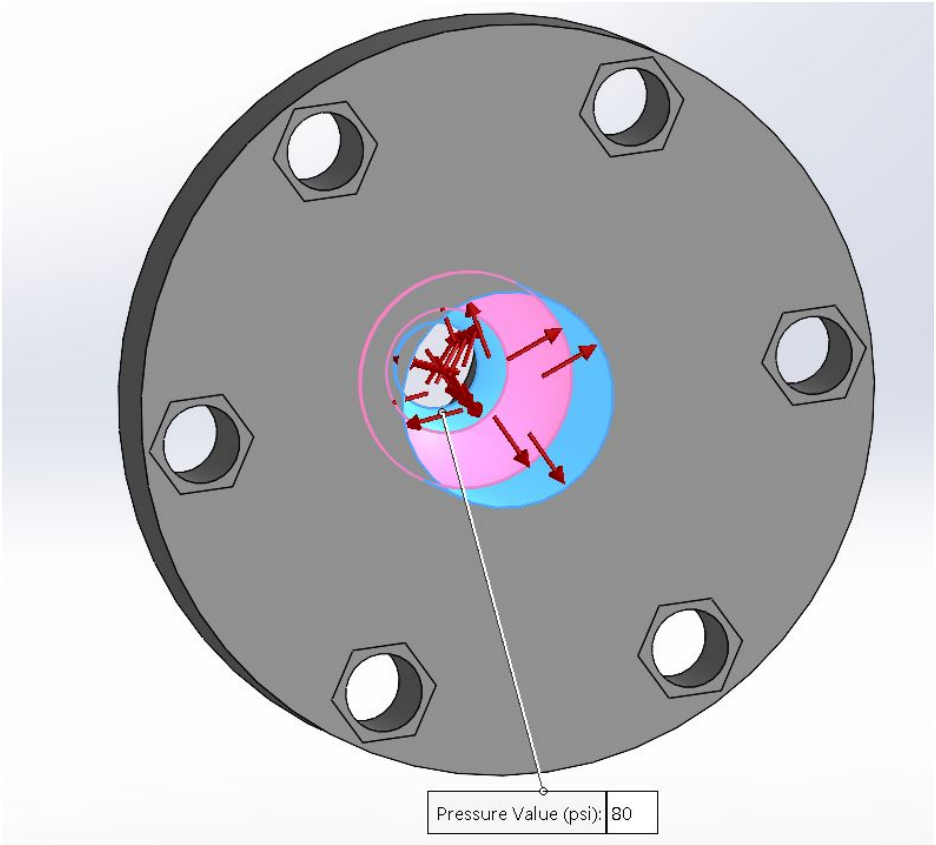


Fixed Geometry



Loads



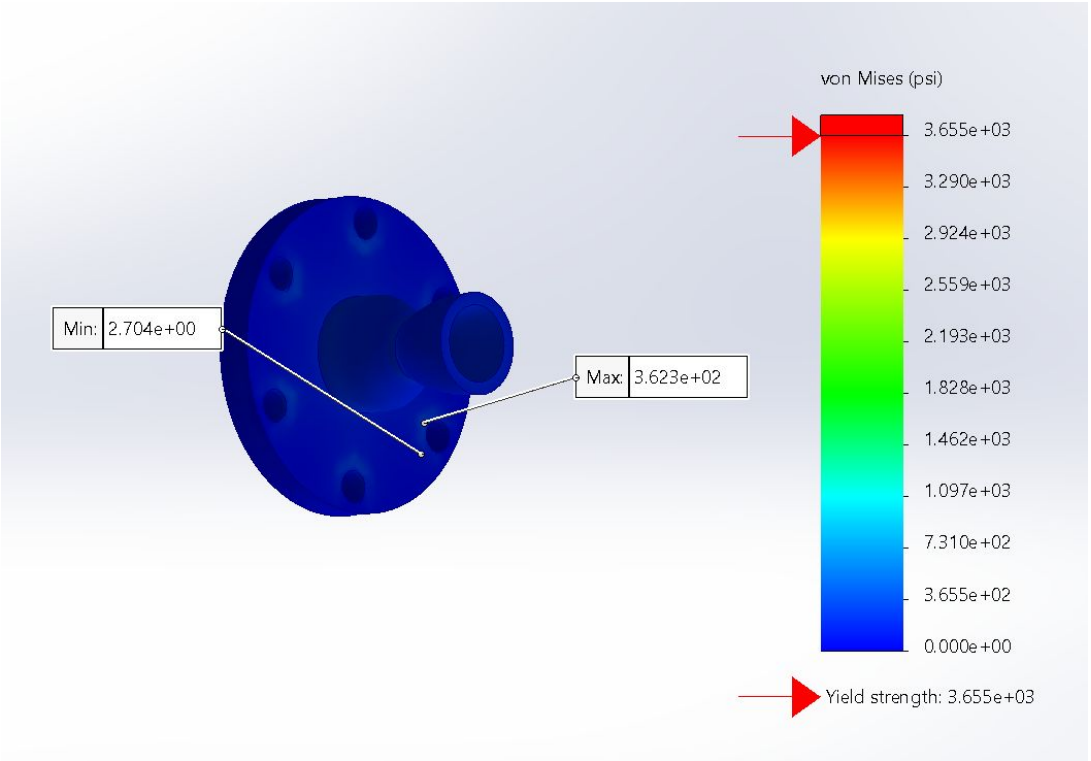


Stress Testing

Yield Strength: 25.2 Mpa or 3654.951 psi

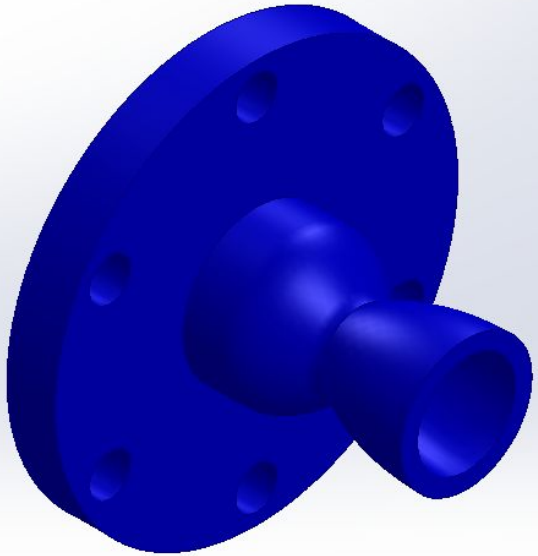
Ultimate Tensile Strength: 36Mpa or 5221.36 psi

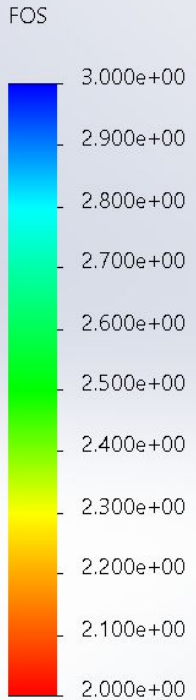
Went for the lowest possible in the strength ranges of the material and made sure that the forces on it would not exceed that strength minimum



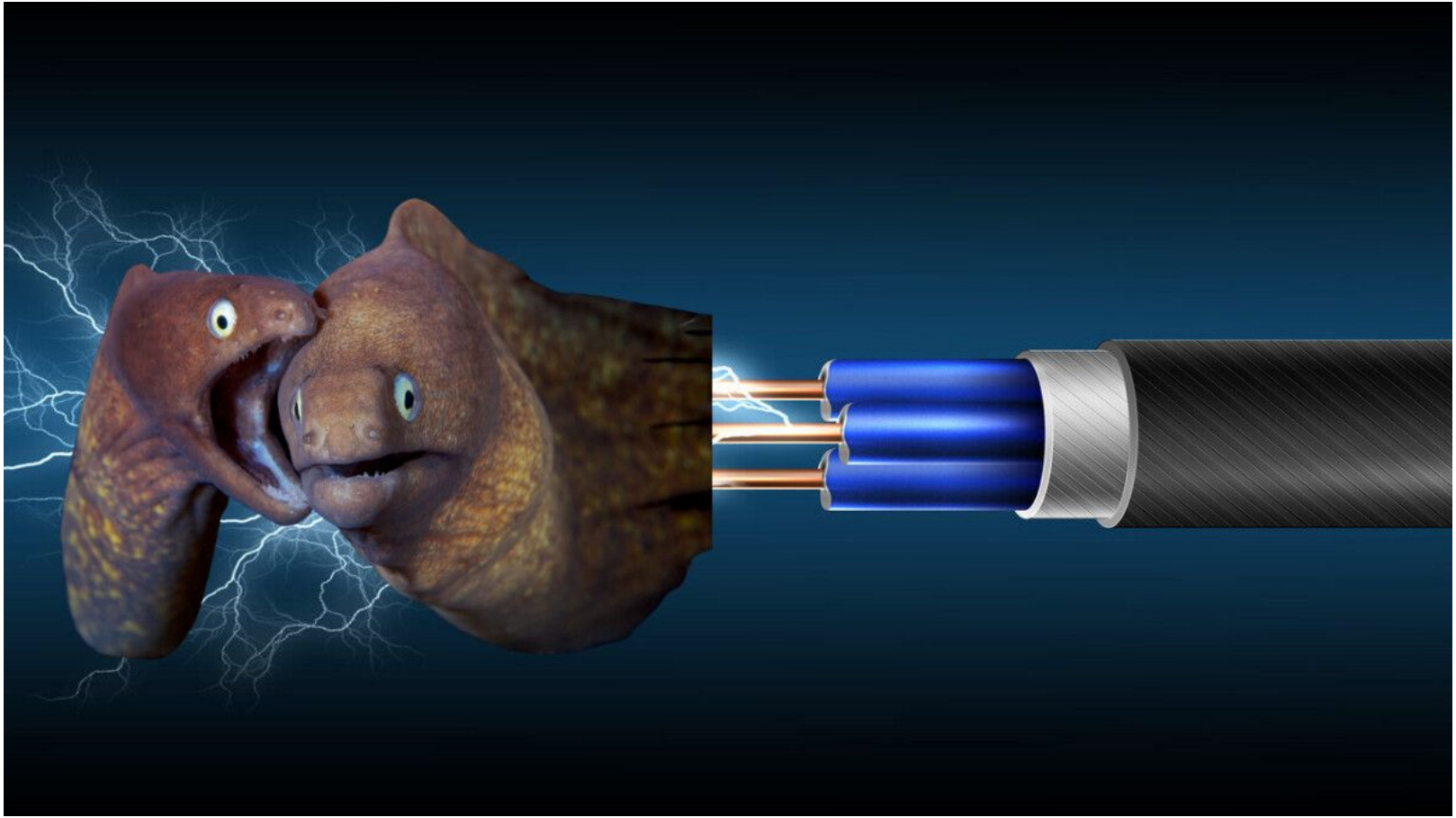
FOS Plot

Model name: nozzle
Study name: Static 2(-Default-)
Plot type: Factor of Safety Factor of Safety2
Criterion : Automatic
Factor of safety distribution: Min FOS = 10

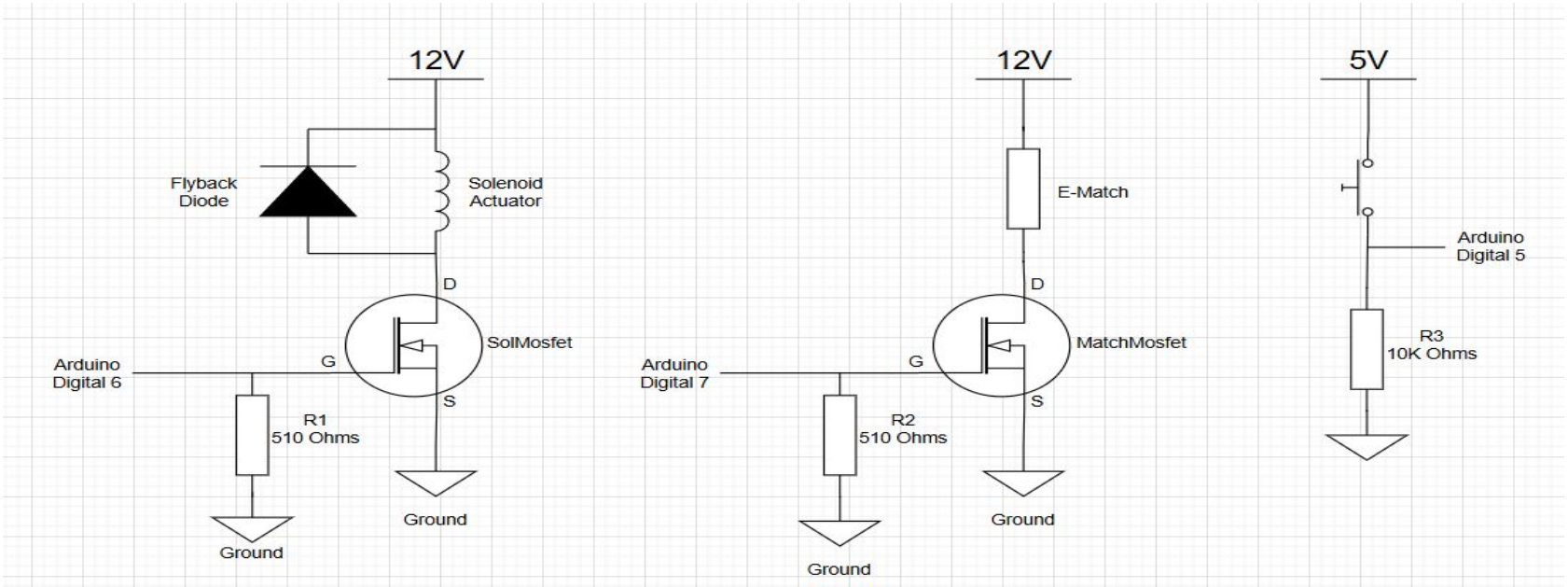




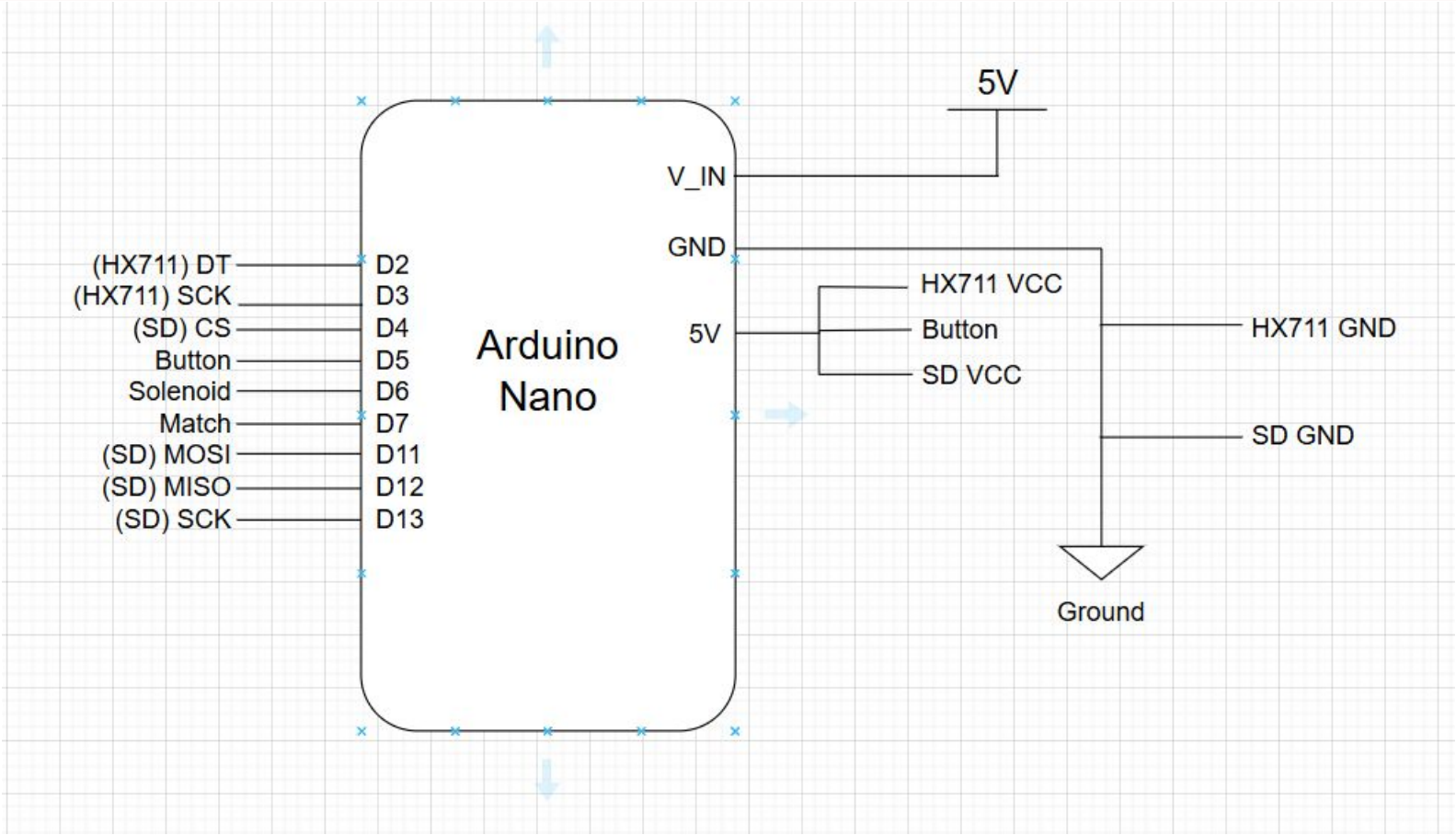
Ewectwonics



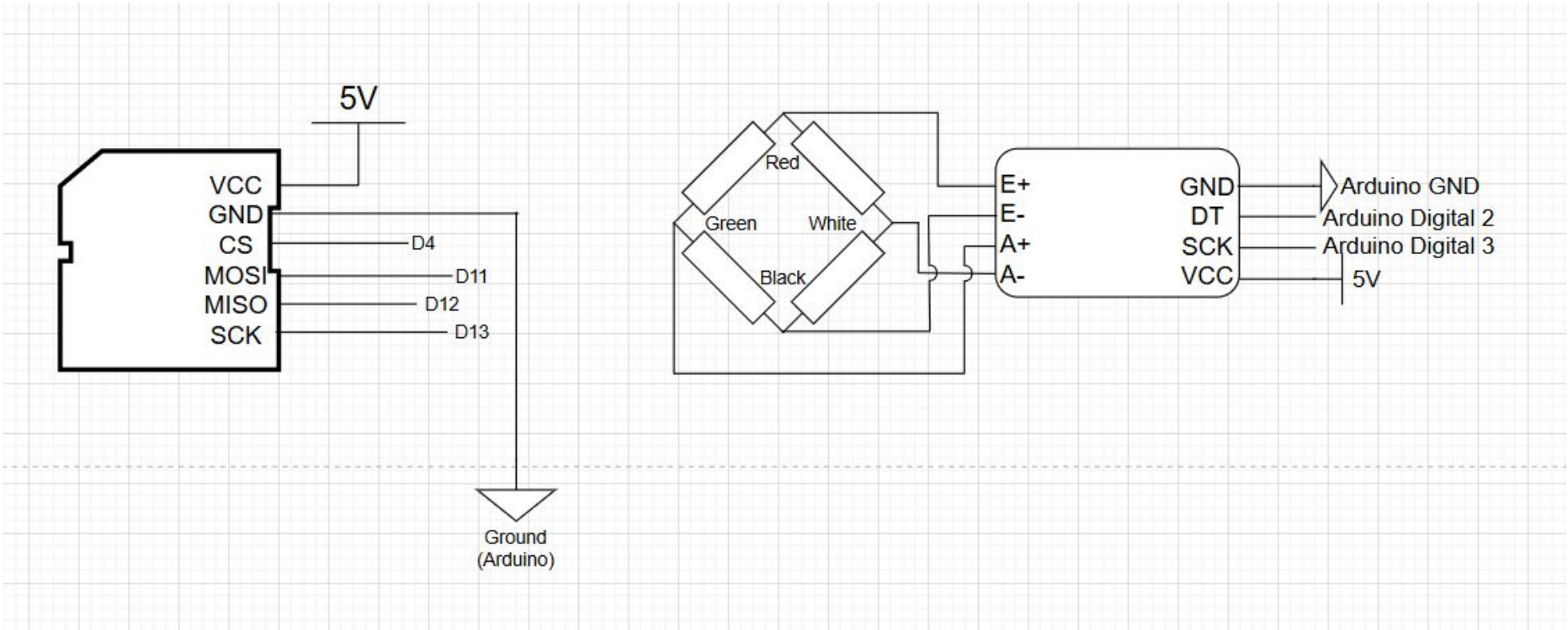
Schematic (Solenoid, E-match and Button)



Schematic- Arduino



Schematic (SD Card and HX711)



Code

```
#include "HX711.h"  //Add library of HX711 so can use
#include <SPI.h>    //Includes so pins 4,11,12 and 13 work as intended
#include <SD.h>     //Add library of SD so can open file


//Pin Designations
const int buttonPin = 5;                //Set button pin to pin 5
const int SolMosfetPin = 6;             //Set connector pin to MOSFET gate of solenoid path to pin 6
const int MatchMosfetPin = 7;           //Set connector pin to MOSFET gate of e-match path to pin 7
const int DTPin = 2;                    // set pin for Data from HX711 Amp to pin 2
const int SCKPin = 3;                   //set pin for clock from HX711 Amp to pin 3
const int chip = 4;                     //set pin which decides which to commuicate from to SD Card module,
used 4 cuz uncreative
const long interval = 200;              //sets the interval length to 200 milliseconds
unsigned long previousReadingTime = 0;  //begins the program with having read at 0 seconds
```

Code

```
//Initializing Scale
HX711 scale;    //Initializes HX711
File dataFile;  //Declares dataFile globally so it works in function and loop
void InitializeSDCard() {
  // Initializes the SD card and makes sure it works
  Serial.print("Initializing SD Card...");
  if (!SD.begin(chip)) {                    //checks to see if the SD Card is communicating properly is
functional
    Serial.println("Initializing Failed");  //prints that something occured and Pin 4 is malfunctioning
    return;}                                 //Stops process from going any further, halting program
  dataFile = SD.open("data.txt", FILE_WRITE);  //initializes data file in setup (open once)
  if (dataFile) {
    Serial.println("File has successfully opened");  //prints that the file opened successfully
  } else {
    Serial.println("data.txt did not open successfully");  //prints that the file failed to open
    return;                                                //stops the process
  } Serial.println("Initializing Complete");  //if nothing happens, it prints that the sd card has initialized
}
```

Code

```
void InitializePins() {
  //Sets the pins up and turns them to low automatically
  pinMode(buttonPin, INPUT);         //Sets button as input
  pinMode(SolMosfetPin, OUTPUT);     //sets MOSFET pin on Solenoid path as output
  pinMode(MatchMosfetPin, OUTPUT);   //sets MOSFET pin on Ematch path as output
  Serial.println("The Pin settings are complete");
  //Setting pins to low so it starts out off
  digitalWrite(SolMosfetPin, LOW);     //Sets Mosfet pin on solenoid path to 0, so starts at 0
  digitalWrite(MatchMosfetPin, LOW);   //same as above, but for the one on the ematch
  Serial.println("The Output Pins have been set to Low");
} enum State {  //declares variable state and the 3 different kinds states possible, IDLE, SOLENOID_ON and
MATCH_ON, used enum bc easier to read name than look at boolean
  IDLE,
  SOLENOID_ON,
  MATCH_ON,
};
State currentState = IDLE;          //initiates program with state IDLE
unsigned long stateStartTime = 0;   //Sets the initial start time of the first IDLE section to 0
```

Code

```
void setup() {
  Serial.begin(9600);  //Begin card comms at baud rate 9600, cuz arduino nano can't get much higher
  while (!Serial) {    // waits for the serial port to connect (I.E computer)
    ;
  }
  Serial.println("USB Connection is Good");

  //Running Initialization Functions for SD Card Module and Nano Pins
  InitializeSDCard();  // Runs the SD Card Initialization Function
  InitializePins();    // Runs the Pin Initialization Function

  //Starting the loadcell up
  scale.begin(DTPin, SCKPin);  //initializes the Loadcell, DT pin signals which pin the data comes from and
SCK controls the timing of data collection
  Serial.println("The Loadcell is Initialized");
}
```

Code

```
void loop() {
  // Deals with button mechanism turning on and and off with button
  unsigned long currentTime = millis();  //Checks how many milliseconds have passed since the loop began each time the loop is run

  switch (currentState) {  //sets up a switch() case: statement so the program swaps when currentState changes
    case IDLE:
      if (digitalRead(buttonPin) == HIGH) {
        delay(50);  //Debounce
        if (digitalRead(buttonPin) == HIGH) {
          digitalWrite(SolMosfetPin, HIGH);
          stateStartTime = currentTime;  //sets the stateStartTime for the SOLENOID_ON to whatever time it is
          currentState = SOLENOID_ON;    //swaps case to SOLENOID_ON
        }
      }
      break;  //prevents falling-through (moving to next case)
```

Code

```
    case SOLENOID_ON:
      if (currentTime - stateStartTime >= 200) {  //if 1/5 of a second has passed since the solenoid mosfet
turned on
        digitalWrite(MatchMosfetPin, HIGH);       //sets ematch to ignite
        stateStartTime = currentTime;             //sets the start time for MATCH_ON
        currentState = MATCH_ON;                  //sets current state to MATCH_ON
      }
      break;
    case MATCH_ON:
      if (currentTime - stateStartTime >= 5000) {  //after 5 seconds since the ematch turned on
        digitalWrite(SolMosfetPin, LOW);           //Turns off solenoid MOSFET
        digitalWrite(MatchMosfetPin, LOW);         //Turns off match MOSFET
        dataFile.close();                          //Saves it and closes file
        Serial.print("File Closed");               //Communicates that the data has been filed and saved
        currentState = IDLE;                       //Resets to IDLE state
      }
      break;
  }
```

Code

```
  if (currentTime - previousReadingTime >= interval) {  //Checks if 1/5 of a second has passed since the data was last read
    previousReadingTime = currentTime;                  //sets the previous reading time to the current time as it is being checked
    int reading = scale.read();                         //reads whatever loadcell says and stores as reading
    Serial.print("Timestamp: ");
    Serial.print(currentTime);
    Serial.print(" Value: ");  // prints opening
    Serial.println(reading);   // prints value loadcell reads on the previous line
    if (dataFile) {
      dataFile.print("Timestamp: ");
      dataFile.print(currentTime);
      dataFile.print(", HX711 Reading: ");
      dataFile.println(reading);  // writes into data file
      dataFile.flush();           //pushes value from buffer to file if value gets stuck
      Serial.println("Data saved in SD Card");
    } else {
      Serial.println("Error opening data.txt");  //prints that file opening failed if it did
    }
  }
```

Shopping List

- 1 Arduino Nano + A perfboard
  - Kinda necessary, otherwise the last 7 slides were for no reason, need the perfboard to put stuff on
- 2 N-Channel MOSFETs Low Capacitance, Form channel w/ gate mean fast switch time
- 1 Solenoid Actuator
  - To open and close fuel line for nozzle to work
- 1 Flyback Diode
  - Flyback Voltage Bad
- 1 E-match
  - someone could pull out a Bic and light it. might work.
- Red Button
- 3 10K ohm resistors (to prevent floating MOSFETs and button)(Dunno how to do the math to find exact value, but according to the HIP chat, this is good)
- SD Card Module and SD Card itself
  - Collect data for this
- Loadcell and HX711 Loadcell amp
  - arduino calibrates the loadcell, loadcell collects data, amp sends data to arduino
- 12V battery
  - D r u g s