

Senior Design 2018/2019
Maroon Five

Anas Alhamad

Arik Espineli

Jasmine Gill

Miranda Sweigert

Jeff Smith

Design Review 1.2 Documentation

12/4/18



Table of Contents

Executive Summary	2
Risk Reduction Prototype	3
Electrical RRP	3
Mechanical RRP	9
Specification Table	14
Engineering Analyses	15
Electrical	15
Mechanical	20
Risks Reduced or Remaining	22
Updated Project	23
Updated Electrical Block Diagram	23
Updated Mechanical Block Diagram	24
Rigorous winter break and first week schedule	25
Preliminary Full Year Schedule	27
Winter Quarter	27
Spring Quarter	34
Appendix	38
Specification Supporting Material	38
Engineering Analysis Supporting Material	53
References	62

Executive Summary

Research shows that employees and machinery in an industrial environment may not be sufficiently protected from machine fires. In industrial and manufacturing environments an average of 37,000 fires occur each year. This amounts to \$1 billion in property damage and hundreds of injuries [1]. Even though machines are designed with the highest safety standards, the materials they work with may be susceptible to fires. Fires can also start due to tool failures, programming mistakes for CNC machines and more [2]. These fires could be prevented by assigning someone to be constantly be on fire watch, however this is not always possible. While fire watchers may prevent or stop a fire they still come at a cost to companies. The goal of Maroon Five is to create a device that can protect a company's interests while not costing too much.

Maroon Five aims to accomplish this by creating an X-Y ceiling gantry system with an integrated fire extinguisher. This system would automatically detect fires using sensors integrated with a microprocessor. The system would move the extinguisher to the location of the fire and suppress the fire automatically, eliminating the need for a designated fire-watcher, and decreasing damage that would normally be caused by sprinkler systems. This system would be mounted to the existing ceiling infrastructure and use sensors on the system and placed around the shop area to detect fires.

The highest risks in this solution are being able to accurately detect a fire in a shop environment and the successful movement of the extinguishing housing to the area of the fire. It is also important that the system be able to decide if a fire or heat source is wanted (welding, plasma cutting, etc.) or unwanted (shop fires).

To demonstrate the feasibility of this solution, two risk reduction prototypes were built. One to demonstrate the ability to code sensors to detect fires. A combination of infrared, ultraviolet, thermal, and smoke sensors will be integrated with a microcontroller and programmed to successfully detect a fire. The other demonstrates the feasibility of designing and fabricating a movement system that can be mounted to the ceiling. A four foot by four foot stable two axis rail system mounted on ceiling supports will be constructed to successfully move in both X and Y directions. These prototypes will show that it is possible to build the full system to detect, locate, and extinguish a fire.

Risk Reduction Prototype

Electrical RRP

For the electrical RRP Maroon Five Engineering researched the best sensors to use to accurately detect a fire. The electrical engineers tested infrared (IR), ultraviolet (UV), thermal, and smoke sensors using a live fire. The selected sensors were integrated with a microcontroller where the output data was saved to be analyzed later.



Fig. 1: Experiment Site (Magnolia Park)

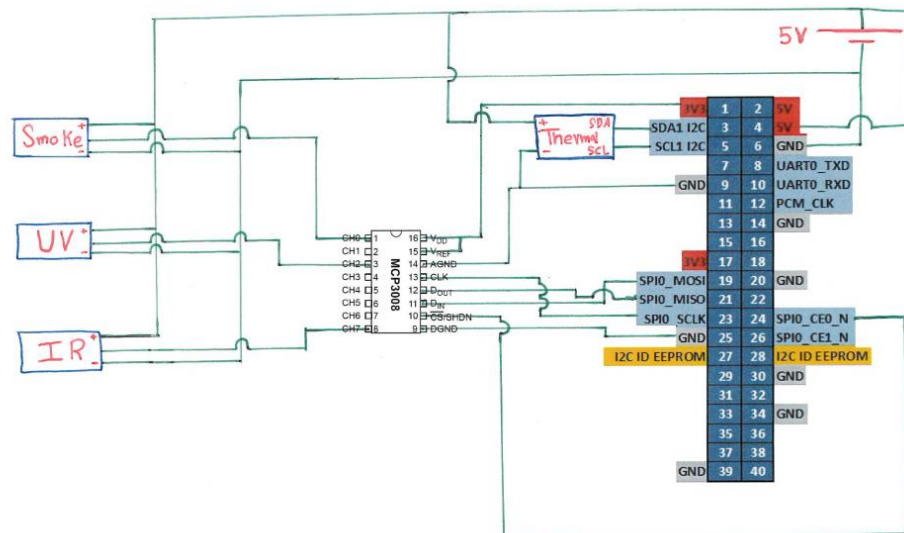


Fig 2: Raspberry Pi Circuit

Pictured above (Fig. 1) is the area where the electrical RRP (Fig. 2) testing was conducted. The barbecue pit allowed for semi-accurate placement of charcoal so that the size of the fire was near the size specified in D002. The main concern faced with testing outdoors was that the

height of the fire, due to wind conditions, varied uncontrollably. This led to, at certain points during the test, a flame that was larger than 8"x8"x8".

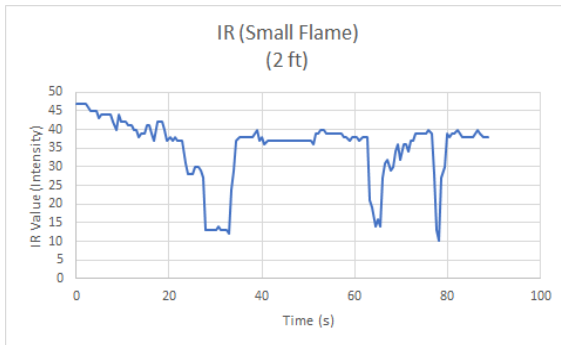


Fig 3: IR Values - Small Flame

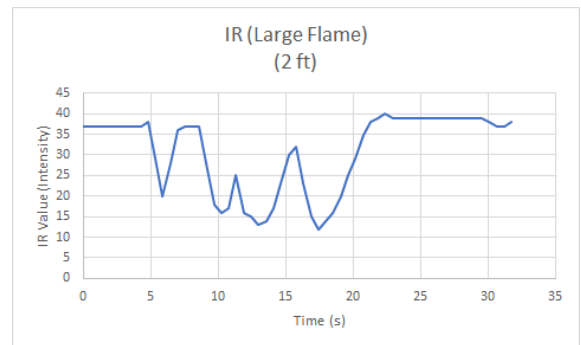


Fig 4: IR Values - Large Flame

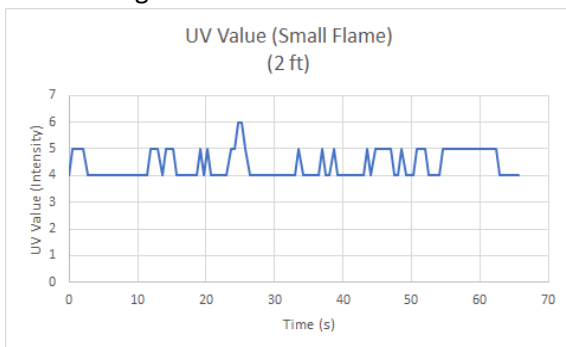


Fig 5: UV Values - Small Flame

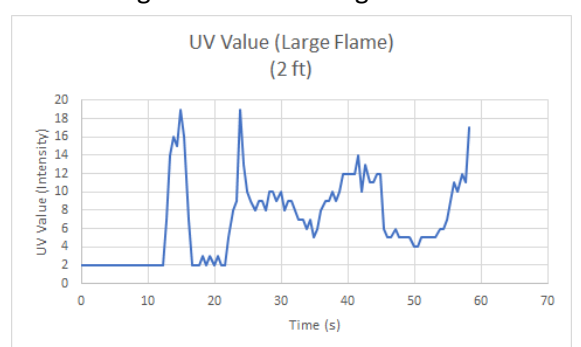


Fig 6: UV Values - Large Flame

****Note: Small Flame = ~8"x8"x8" && Large Flame = Flames larger than ~10"x10"x10"****

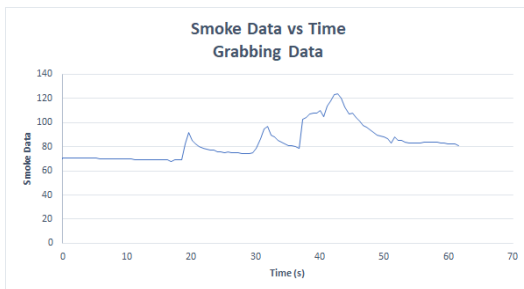


Fig 7: Smoke Data - Different sensor angles

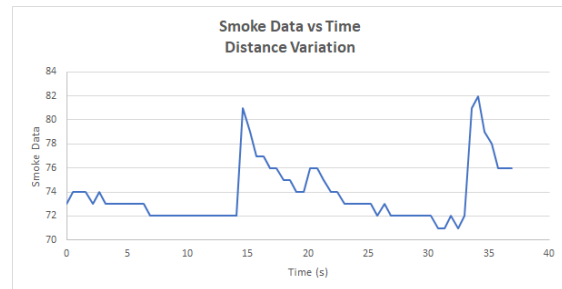


Fig 8: Smoke Data - Varying sensor distance

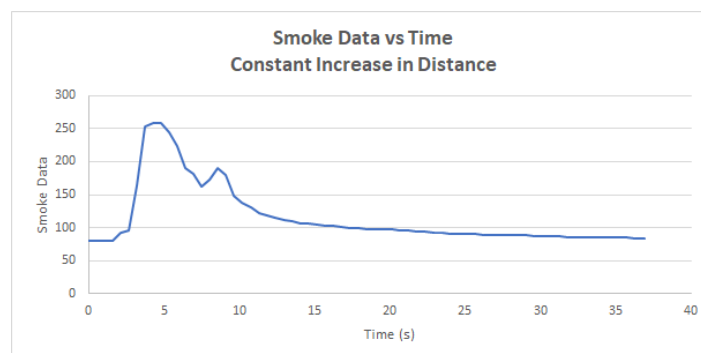


Fig 9: Smoke Data - Constant increase in distance directly in the path of the smoke

```

New Reading
7.2 7.2 6.5 6.8 6.5 3.5 1.8 511.8
7.2 7.8 6.8 6.8 6.2 5.2 2.5 511.2
8.0 6.8 6.5 6.2 6.0 6.0 4.2 1.0
8.0 8.0 6.5 6.0 6.0 5.8 5.0 5.0
7.5 12.2 7.0 8.2 6.8 503.8 5.8 0.0
8.0 7.0 6.5 6.0 6.8 6.0 6.8 6.0
8.2 7.5 6.2 6.2 6.2 6.0 6.2 6.2
7.8 7.8 6.8 6.2 6.5 6.0 6.2 6.0

New Reading
37.2 43.2 39.0 30.2 23.5 17.0 13.8 11.5
27.2 28.5 26.0 21.5 17.2 15.0 12.2 10.8
23.0 22.8 21.2 18.0 15.0 13.8 11.2 10.2
19.2 18.0 16.5 14.5 13.2 11.5 9.5 8.2
17.2 16.2 15.2 13.2 12.2 10.5 7.2 6.8
15.0 14.0 12.8 12.0 11.2 10.2 8.5 7.0
14.0 13.5 11.8 11.2 10.2 9.0 7.0 4.5
13.0 12.8 11.8 11.0 10.0 9.8 7.5 4.2

New Reading
450.8 389.2 100.5 63.5 48.0 40.8 36.5 32.5
401.2 350.2 107.5 82.5 53.5 44.5 37.8 35.0
386.5 347.8 105.5 80.2 53.8 44.5 38.0 35.2
385.5 337.5 104.8 101.2 62.0 49.2 40.5 37.5
414.0 358.8 106.8 105.2 61.5 49.2 40.0 37.0
415.2 372.0 109.2 75.5 50.0 42.0 36.5 34.0
104.0 103.2 81.2 60.0 44.0 37.8 34.0 31.2
85.0 86.8 66.8 52.5 41.5 36.5 32.5 31.2

```

Fig 10: Sample Thermal Camera Output (8x8 array of temperatures)

The charts (Fig. 1-10) above display the output from the sensors tested. The thermal camera output (Fig. 10) displays the change in temperatures as seen by the sensor when it was pointed at the flame. This will be useful in developing a fire detection algorithm by utilizing a rate of change formula for the the temperatures along with the outputs from other sensors leading to a higher fire detection rate.

For most of the sensor tests performed the sensor being tested was not immediately pointed at the flame. There was a delay of a few seconds so that baseline values which would be used distinguish the values for flame recognition could be recorded. This was repeated multiple times to test the consistency and sensitivity of any given sensor as evidenced by the multiple peaks and valleys of each graph.

D001: Fire Recognition - Timing: The system shall be able to recognize when a fire occurs within 90 seconds. The system should be able to recognize when a fire occurs within 60 seconds. The sensors will be able to detect when a fire occurs in the room. Ninety seconds has been determined to be sufficient for our RRP response time. This will be validated by using a timer that will record how long it took for the system to detect the fire within ignition.

It can be observed in the charts above (Fig.1-10) that the sensors recognize the fire well under 60 seconds. The valleys observed in the IR sensor vs time graph (Fig. 3 & 4) occur when the sensor is pointed at the flame. The smoke sensor data (Fig. 8) was gathered by moving the sensor close to the flame (higher density) and moving it away

slowly (lower density). The output from the UV sensors (Fig. 5 & 6) increases as it recognizes the fire, it was able to detect the flame within the first few seconds.

```

thermalData = thermalCamera.getThermalOutput()

for row in thermalData:
    formatTemp = ['{0:.1f}'.format(temp) for temp in row]
    for num in formatTemp:
        thermalText.write(str(num))
        thermalText.write(" ")
    thermalText.write("\n")

thermalTime.write(str(t.time()))
thermalTime.write("\n")

thermalText.write("\n")
thermalText.write("New Reading")
thermalText.write("\n")

"""
uvIndexData = uvIndex.getUVindexOutput()
writeToTxtFile(uvIndexText, uvIndexData)
"""

t.sleep(.5)

```

Fig 11: Thermal Camera Partial Data Output Code

The readings for the thermal camera were taken at half second intervals (Fig. 11) and it can be seen from Fig.10 that the camera recognizes the flame, indicated by the change in temperatures, within 1.5 seconds.

D002: Fire Recognition - Size: The system shall be able to recognize a fire that is 8 inches in diameter. The system should be able to recognize a fire that is 6 inches in diameter.

The system should be able to detect a wide range of fire sizes, the smallest of which is a six-inch fire. This size was chosen because the size of a welding & plasma cutter flames can range from one to six inches. The size of the flame for testing will be validated by building the fire on a measured piece of equipment.

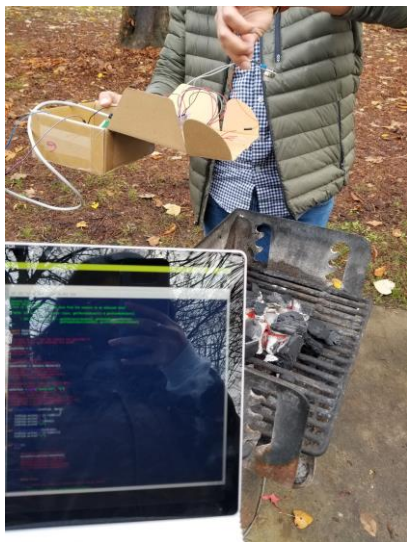


Fig 12: Testing smoke sensor with an ~8" fire



Fig 13: Testing sensors with a large fire

The size of a recognizable flame varied with each sensor and was not as accurate as the other tests due to wind conditions. The furthest a ~6" flame was detected was 14"-16" away. This was near enough to heat the sensor to unsafe levels and was deemed unsafe. An ~8" flame on the other hand was detectable from 2-3 feet away by all of the sensors except for the smoke sensor.

Fig. 9 shows the variation in the data collected by the smoke sensor as it was moved around the flame to find the ideal angle and distance. It was discovered that it only detected the smoke if it was placed directly in the path that the smoke was blowing. It was not sensitive enough to discover the flame even if the smoke was passing near it.

D003: Fire Recognition - Accuracy: **The system shall accurately detect a fire 9 out of 10 times. The system should accurately detect a fire 10 out of 10 time.** The accuracy of detection makes the system more reliable. This will be validated by testing the system repeatedly and recording results.

The tests for flame detection were repeated several times to determine the reliability of the sensors. Each time the tests were run the sensor was pointed at and away from an 8" flame at the maximum distance that they were able to detect the flame. This fulfilled D003 as the sensors were able to detect the flames every time they were pointed at it.

Mechanical RRP

For the RRP Maroon Five Engineering designed and constructed a stable two axis rail system that can be mounted to existing ceiling infrastructure that will not cause damage to the built environment around it.

M001: Middle Bar Movement - the middle bar of the 'H', or gantry, shall successfully move in the y axis between the other two beams. **The beam shall cover 3 total feet in the y direction, 1.5 feet in positive and negative directions. The beam movement should cover 4 total feet, 2 feet in positive and negative y directions.** Current plans are that 2 feet is expected to be sufficient to show movement is possible to reach the location of the fire, this means a 16 square foot area. This size has been determined based on equipment located in the Seattle Pacific University Machine Shop. This will be tested by measuring the full range of motion of the axis and compare it to the actual size.

M001 was not successfully met. This is because the manufacturing time for the motor mount was longer than anticipated due to lack of access to the shop. Maroon Five engineers are confident that with sufficient time to complete fabrication of the motor mount, that this specification would be met successfully. In the video (<https://drive.google.com/file/d/1W0AVCo4X4vSekmBRbSfLgAdlztOkwVGE/view?usp=sharing>) it can be seen that both motors run successfully together. This means that once the motor for bridge movement is mounted, the motors would work successfully together on the system. The motor mounts will be redesigned over winter break to ensure early manufacturing during winter quarter.

M002: Extinguisher Movement - the extinguisher housing will move along the rail system successfully in the x direction. **The housing shall cover 3 total feet in the x direction, 1.5 feet in positive and negative directions. The housing movement should cover 4 total feet, 2 feet in positive and negative x directions.** Current plans are that 2 feet is expected to be sufficient to prove movement and reach the location of the fire. This will be sufficient within the given area, see M001. This will be verified by measuring the full range of motion of the axis and compare it to the actual size.

Fig 14: Set-up for verifying M002



M002 was met successfully. The motor to move the housing moved over 3 feet in the x axis, which successfully meets the threshold value for this specification. Set up of the motor for testing can be seen in figure 14. In the video (<https://drive.google.com/file/d/1478e9Mn2jsDL4ofbgP7OOzRnnqv5GVUy/view?usp=sharing>) this can be seen.

M003: System Movement - the extinguisher housing will move to a specified position in the room. **The housing shall move within a 1-foot radius of specified location and should move within a 0.5-foot radius of specified location.** The fire extinguisher will be most effective with a sweeping motion. If the trolley is located within this range the extinguisher will be able to suppress our target fire size, see D002. This will be verified by inputting a desired location and physically measuring the actual position and compare it to the desired location.

M003 was partially met. This is because M001 was not met, so a location could not be prescribed for both x and y coordinates. Though when prescribing a coordinate through meeting M002, the housing moved successfully to within 1 inch of the desired location. Set up for this can also be seen in figure 14.

M004: Bridge Speed of Movement - The motor required for the movement of the bridge in the x-direction shall supply a torque that provides a speed of 4 feet per second, and should supply a torque that provides a speed of 8 feet per second. This was determined by measuring the average speed of human response to a fire and how long it would take to obtain an extinguisher and extinguish a fire. This took an average of 20 seconds to move 20 foot, which is a speed of 1 foot per second. Our system must be faster than human response. This will be validated by making the motor move the bridge a specified distance and recording the time.

This specification was not met because M001 was not met. Based on the movement from M002, Maroon Five engineers are confident that M004 can be met, since the speed of the motor for M002 met the threshold of this specification. See M001 for completion information.

P001: The power needed for all components shall not be greater than 15 Amps and should not be greater than 12 Amps. The device needs to adhere to code standards and easily be utilized in a standard outlet. At 120 volts in a standard outlet, the power supply should not be higher than 15 Amps.

The total current pulled in the system is equal to the current pulled by the Switching Power Supply which has a maximum power of 500W. This means the maximum current pulled from the electrical outlet is at 4.1A if ideal Switching Power Supply or 5A if accounting for 20% power lost. The motors were also tested to ensure enough power is being supplied when needed. To achieve that the circuit was opened and then the Ammeter was placed between the switching power supply and motor driver (Fig 14). The reading was recorded while X-Motor was at stop (0.62A) and while turning (1.09A). The method was then repeated with Y-Motor and recorded (0.77A) while turning. These numbers agree with the range given by the motor drivers' datasheet that has a maximum current of 5A at 48V. This fulfilled P001 as the maximum supplied current from the electrical 120VAC outlet will not exceed a total of 12A.

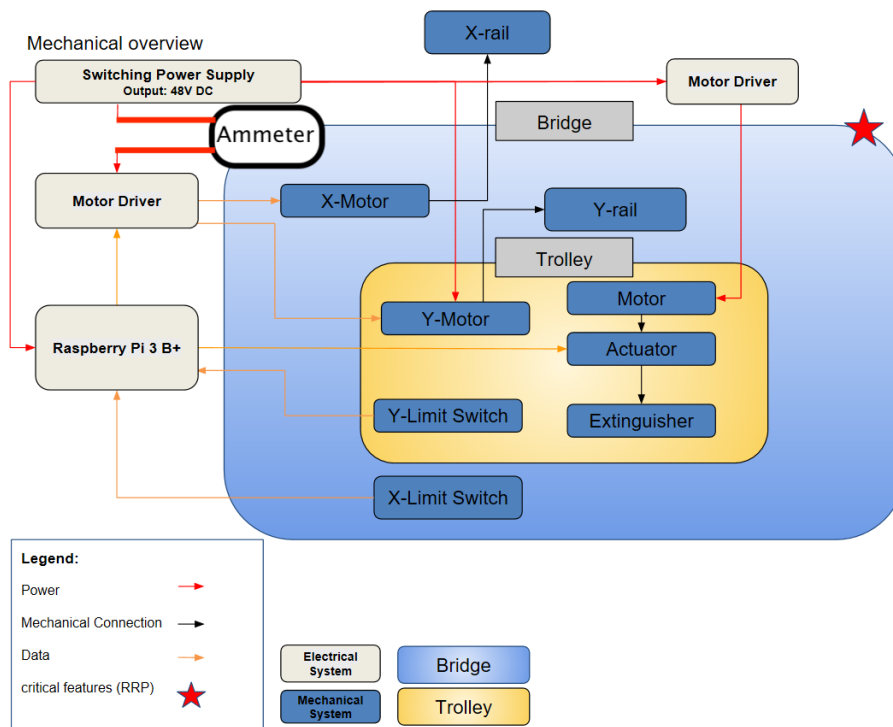


Fig 15: Mechanical Block Diagram

Specification Table

Spec ID	Requirement	Threshold (Shall)	Objective (Should)	Validation Method	Met?
D001	Fire Recognition - Timing	Recognize fire within 90 seconds	Recognize fire within 60 seconds	Testing with different fire scenarios and sensors in a controlled area.	Yes
D002	Fire Recognition - Size	Recognize a fire that is 8 inches in diameter	Recognize a fire that is 6 inches in diameter	Testing with small fires that are built and see if small or larger fires are detected.	Possibly
D003	Fire Recognition - Accuracy	Recognize a fire accurately 9/10 times	Recognize a fire accurately 10/10 times	Test the system repeatedly in the scenarios described in the other specs.	Yes
M001	Middle Bar (y) Movement	Move 1 foot in positive and negative y	Move 2 feet in positive and negative y.	Tape Measure.	No
M002	Extinguisher (x) Movement	Move 1 foot in positive and negative x	Move 2 feet in positive and negative x	Tape Measure	Yes
M003	Housing Movement	Move to within a 6 inch radius of fire location	Move to within 4 inch radius of fire location	Tape Measure	Partially
M004	Bridge Motor Speed	Shall move the bridge at 4 feet per second	Should move the bridge at 8 feet per second	Recording the time it takes to move to specified distances	No
P001	Power Supply	<15A	<12A	Measuring supplied current with Ammeter	Yes

Fig 16: Specification Table

Engineering Analyses**Electrical**

Power Consumption - Compute the maximum estimated power consumption for each element at its rated voltage to find the total needed power that help selecting a proper power supply equipment parts.

The main power supply for the unit is the electrical outlet plug (120VAC). Three more power supplies are supplied by the main outlet plug (120VAC) to provide rated voltage to the electrical elements. The max current that the main power supply sees is 2.08 Amps. In the table below (Fig 18) the highlighted numbers in orange indicate the changes after testing the sensors and analyzing their data. This is based on a positioning solution that fits the sensors characteristics.

Circuit Element	Voltage (Volts)	Current (Amps)	Peak Power (Watts)	Qty	Total Current (Amps)	Total Peak Power (Watts)	Supplying Power to Other Elements (Y/N)
Switching Power Supply	120 VAC	5 A	500 W (If full load)	1	--	100 W (accounting for 20% power lost)	Y (powering the unit)
Motor Driver 1	48 VDC	5 A	240 W	1	--	48 W (accounting for 20% power lost)	Y (powering 7.07Nm Motor)
Motor Driver 2	48 VDC	5A	240 W	1	--	48 W (accounting for 20% power lost)	Y (powering 8.5Nm Motor)
Raspberry Pi 3 B+	5 VDC	1 A	5 W	1	1 A	5 W	N (Only signal)
Stepper Motor (7.07Nm)	3.39 VDC	4.24 A	14.4 W	1	4.24 A	17.28 W (accounting for 20% power lost)	N
Stepper Motor (8.5Nm)	5 VDC	5 A	25 W	1	5 A	30 W (accounting for 20% power lost)	N
UV Sensor 1	5 VDC	16mA	0.08 W	1 10	16 mA 160 mA	0.08 W 0.8 W	N

UV Sensor 2	5 VDC	16 mA	0.08 W	1	16 mA	0.08 W	N
IR Thermal Camera	5 VDC	100 mA	0.5 W	1 4	100 mA 400 mA	0.5 W 2 W	N
IR Sensor	5 VDC	16 mA	0.08 W	1 10	16 mA 160 mA	0.08 W 0.8 W	N
Smoke Sensor	5 VDC	16 mA	0.08 W	1 8	16 mA 128 mA	0.08 W 0.64 W	N
Ultrasonic Sensor	5 VDC	16 mA	0.08 W	1 4	16 mA 64 mA	0.08 W 0.32 W	N
Total (Current is calculated from total power at 120VAC)	--	--	--	--	2.08 A 2.11 A	249.18 W 252.92 W	

Fig 16: Power Consumption - power and current needed to be supplied at maximum load

Fire Detection - Research the type and size of fire that we'll be trying to suppress and the sensors that will be used to detect that fire.

The project specification for fire detection is for class A, B & C fires. To determine whether these fires can be detected UV, IR, smoke, and thermal sensors will be used.

The following table describes the types of fire and the sensors that can be used to detect that specific type of fire.

Class of Fire	Materials	Sensors
A	Solid materials i.e. wood, paper or textiles	UV, IR & Thermal
B	Flammable liquids i.e. petrol, diesel or oils	Thermal
C	Electrical Fires i.e. short circuits or overloaded electrical cables	UV, IR & Thermal

Fig 17: Class of Fire and Relevant Sensor Chart

Below is a chart of the ideal distance and position for each sensor compiled by using the data collected during the specification testing. The testing for each sensor was terminated at the furthest distance it was unable to accurately detect an ~8"x8"x8" - 10"x10"x10" flame. The size of the flame may not be accurate as the wind either increased or decreased the volume of the flame.

Sensor	Ideal Distance (ft.)	Ideal Position
UV	2 - 3 ft.	45° - 60°
Backup UV	2 - 3 ft.	45° - 60°
Flame (IR)	2 - 3 ft.	45° - 60°
Thermal Camera	6 ft.	Any plane above the fire
Smoke	1.5 - 2 ft.	Directly above fire

Fig 18: Ideal Distance and Position Chart



Fig 19: Distance Testing (Smoke Sensor: Directly above flame ~1.5-2ft)

The range for the sensors to detect the fires (Fig. 18) were found to be unsuitable for the project. After a consultation with Dr. Lindberg the EE team was advised to use the same sensors as above with a double convex lens (Fig. 20 & 21). This lens would help increase the range that the sensor could detect the fire but it would narrow the field of view. This particular lens was only able to enhance the range of the UV sensor thus the increased range tests were completed with the UV sensors to prove that this method could be used with other sensors given the right lens.

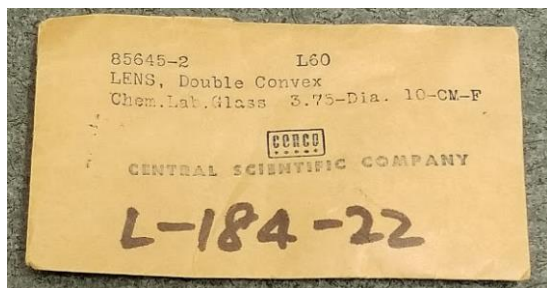


Fig 20: Double Convex Lens



Fig 21: Lens Measurement

The tests with the lens were completed in a new location with a 3D printed tube (Fig. 22 & 23) that held the lens and the sensor to remove any noise from the system. The tests yielded promising (Fig. 24-26) results. The UV sensor was able to detect the fire from 8-ft away. The intensity of the fire, as detected by the sensor, was lower than when it was closer; however, this may be modified during winter quarter by redesigning the tube to ensure having the sensors aligned with the lens.

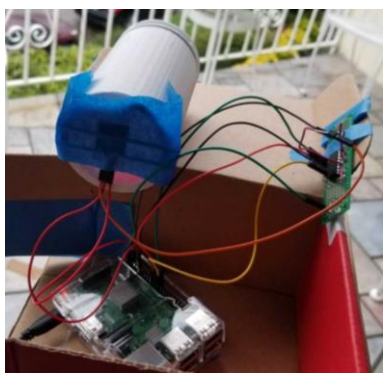


Fig 22: Lens Test Setup

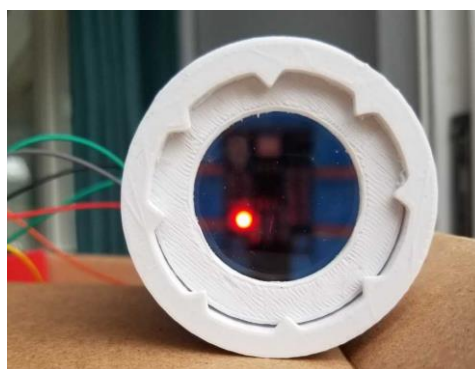


Fig 23: 3D Tube (Front)

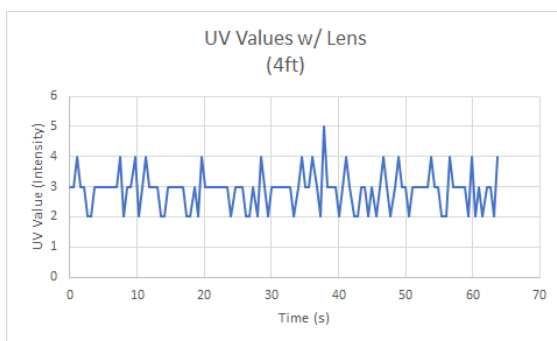


Fig 24: UV Sensor with Lens (4ft)

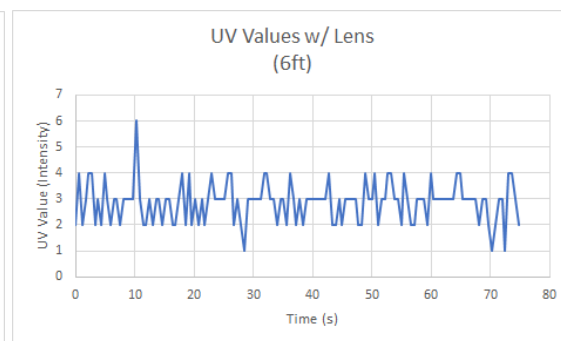


Fig 25: UV Sensor with Lens (6ft)

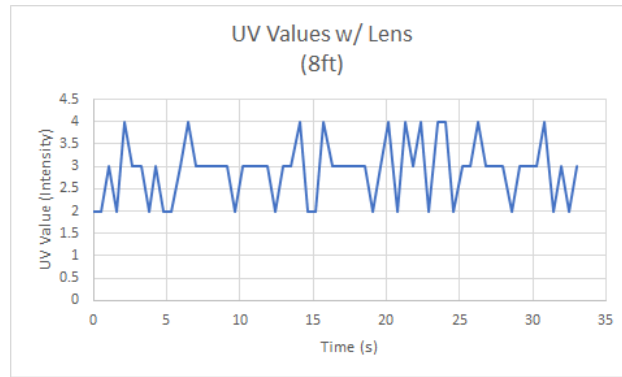


Fig 26: UV Sensor with Lens (8ft)

Safety of housing movement - How to detect if there are obstacles or people in the area where the fire suppression system will be deployed.

Utilizing an ultrasonic sensor (Fig. 27) the system will be able to detect any obstructions that will prevent the safe movement of the trolley from one point to another. It will halt the movement so that the housing will not introduce damages to the user when moving. The formula below will be integrated into the code that moves the system. Ultrasonic sensors have a wide range that can be used to ensure enough space and time for the unit to fully stop.



Fig 27: Ultrasonic Sensor

$$D = \frac{1}{2} \times (\text{Rx-Tx time of wave}) \times \text{Sonic Speed (1236 km/hr.)}$$

Mechanical

Structural Strength - Analyze the strength and rigidity of the rail system and extinguisher housing to ensure that it will be safe to have this system hanging above machinery.

This analysis was used to determine the strength of the ceiling mounts required to rigidly support the gantry. An analysis of the strength and rigidity of the rail system and extinguisher housing was calculated using statics to ensure that it will be safe to have this system hanging above machinery. From this analysis it was calculated that each ceiling connection must be able to support 80N of force and the x-gantry beams must support 150N of force.

Motor Distance - Determine how many steps in stepper motor are required for how much x or y movement.

The motor distance is required in order to determine how many steps in stepper motor are required for how much x or y movement. This analysis has been integrated into the code that drives the stepper motors. To calculate this a conversion from rotational distance to linear distance was calculated. Those calculations were put into a python code that would calculate the number of steps required given a desired coordinate. This code was integrated into the final motor control code.

Motor Power - Calculate the amount of power for the motors controlling the movement and select motors.

This analysis allowed Maroon Five to determine the force required to move our system using an estimated payload. This force was calculated to be:

$$F_{Req.} = 194.145 N$$

From the force calculation the required torque using the radius of the pinion (2 inches). This calculated torque was then used to select a motor that will be able to provide enough torque to move the payload.

System Dynamics - Evaluate the equation of motion of the trolley

This analysis along with the motor distance analysis will allow Maroon Five's engineers to accurately position the fire extinguisher housing above the fire. This analysis was completed by looking at the dynamics of each individual part of the system and combine them to give an equation of motion of the entire gantry system. Both axis use the same rack and pinion so the equation can be applied to both axes. The equation of motion was calculated to be:

$$T = (I_p + I_m) * \omega_m + r_p * F_{Tr}$$

Fire Safety - Calculate the heat transfer within the electronics housing to ensure system does not overheat

A heat transfer analysis was completed to determine the safe operation of the electronics needed to drive the motors. This analysis was conducted by using the thermal resistance network and using the thermal properties of the materials for the housing. The safe operating ambient internal temperature of the box is known based on the electronics, and the equations are set up to be completed next quarter when the orientation of this housing is fully designed. Currently it is not known where the full gantry system will be tested, and this will determine the location of the housing for motor electronics.

Risks Reduced or Remaining

After completing the risk reduction prototype and engineering analyses Maroon Five's electrical engineers were able to reduce multiple risks regarding the detection of a fire (Pg. 3), distinguishing a fire from welding (Appendix Pg. 45), safely moving the housing (Pg. 16) and extending the range of the sensors (Pg. 13). Also the power analysis was completed (Pg. 12) on all the equipment used so that the right power supply could be purchased for the project.

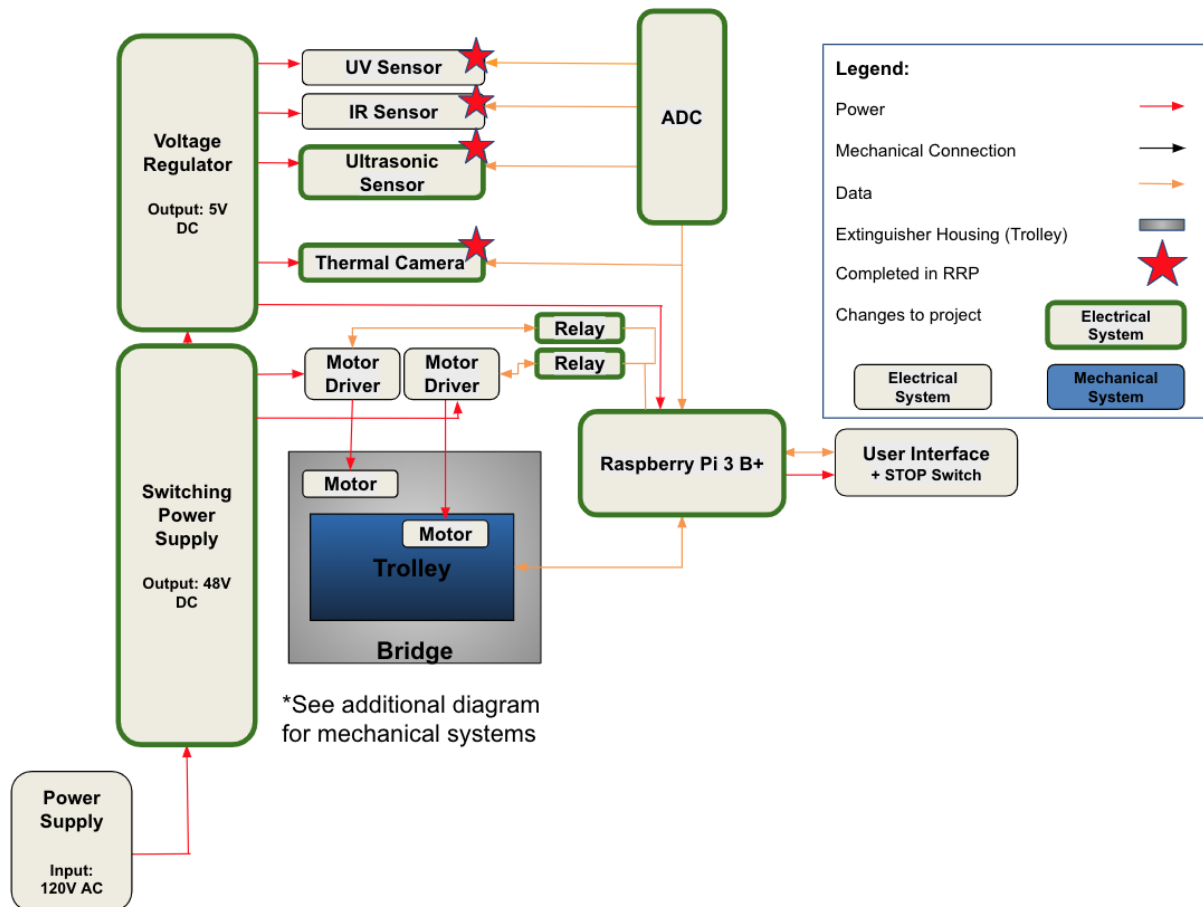
For the sensors used in the RRP, it was found that the UV, IR and Thermal data were the most reliable. These sensors, in addition to the ultrasonic sensor, will be used in the final project. The UV sensor with the lens was also able to distinguish between a welding arc and a fire. This means that moving forward, the system will be able to reliably detect a fire without false alarms and from a reasonable distance. There are no remaining risks from an electrical standpoint for this project.

After working on the mechanical risk reduction prototype the mechanical engineers were able to reduce a variety of risks. The engineers were able to design and build a ceiling connection system to connect the gantry system to the ceiling, develop mounting systems for the motors, and were able to accurately control both motors using a single micro controller. Through different engineering analyses, Maroon Five was able to determine factors of safety for the strength of the system. Analyses also finalized an equation of motion for the system, giving the engineers an understanding of the torque required from the motors and therefore the motor sizing.

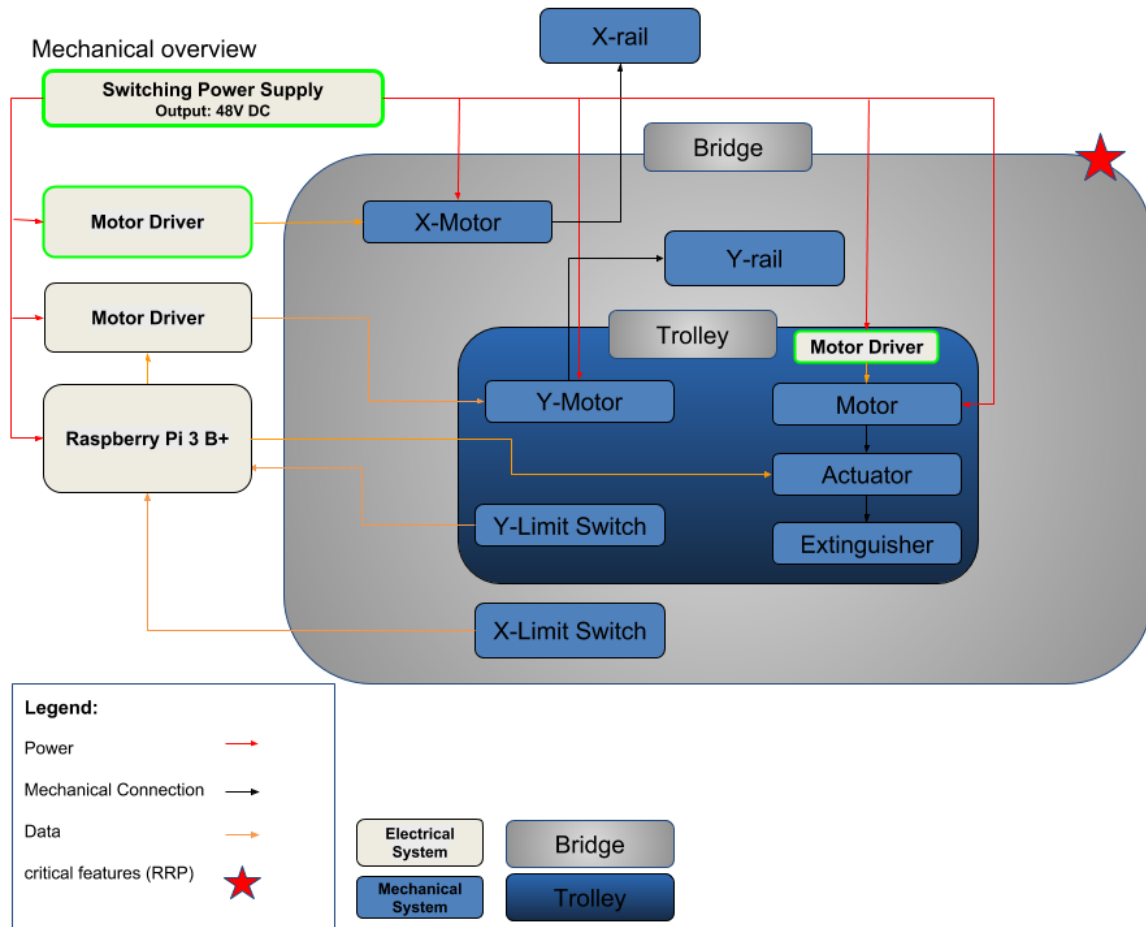
Moving forward there are a few risks that Maroon Five's engineers must address. One of those risks is developing more stable motor mounts that will allow for smoother movement of the entire system, and completing the motor mount for M001. Another risk that must be addressed is controlling the motors. The engineering team was able to successfully move the motors in a single direction, however they were unable to move the motor in both directions. This will be addressed through further research on the motors and the programming of the motors. There are not many available resources which is why this was not completed prior to this design review. Maroon Five has developed methods for addressing these issues and is currently working on reducing this final risks and is confident that these methods will provide for successful implementation of the final product.

Updated Project

Updated Electrical Block Diagram



Updated Mechanical Block Diagram



Rigorous winter break and first week schedule

Task Name	Duration	Start	Finish	Owner
Winter Break Tasks	18 days	Mon 12/10/18	Wed 1/2/19	All
Motor Mount Redesign	18 days	Mon 12/10/18	Wed 1/2/19	Arik
Rough Sketch	9 days	Mon 12/10/18	Thu 12/20/18	Arik
Initial CAD Draft	9 days	Fri 12/21/18	Wed 1/2/19	Arik
Order Parts	18 days	Mon 12/10/18	Wed 1/2/19	All
Gear Rack	18 days	Mon 12/10/18	Wed 1/2/19	Miranda
Aluminum Tubing	18 days	Mon 12/10/18	Wed 1/2/19	Jeff
Cable Management Track	18 days	Mon 12/10/18	Wed 1/2/19	Arik
Step Down Buck Converter	18 days	Mon 12/10/18	Wed 1/2/19	Jasmine
Sensors	18 days	Mon 12/10/18	Wed 1/2/19	Jasmine
Relays	18 days	Mon 12/10/18	Wed 1/2/19	Anas

PCB	18 days	Mon 12/10/18	Wed 1/2/19	EE Team
PCB Design	11 days	Mon 12/10/18	Sun 12/23/18	Jasmine
Order PCB	8 days	Mon 12/24/18	Wed 1/2/19	Anas
Winter Quarter	52 days	Thu 1/3/19	Fri 3/15/19	
Week 1	4 days	Thu 1/3/19	Tue 1/8/19	All
Motor Mount Redesign Final Draft	4 days	Thu 1/3/19	Tue 1/8/19	Arik
Solder PCB	4 days	Thu 1/3/19	Tue 1/8/19	Jasmine
Ceiling Mount	4 days	Thu 1/3/19	Tue 1/8/19	Miranda
Draft of Ceiling Mount Design	4 days	Thu 1/3/19	Tue 1/8/19	Miranda
Drill Holes for Rack Support	4 days	Thu 1/3/19	Tue 1/8/19	Jeff

Preliminary Full Year Schedule**Winter Quarter**

Task Name	Duration	Start	Finish	Owner
Winter Quarter	52 days	Thu 1/3/19	Fri 3/15/19	
Week 1	4 days	Thu 1/3/19	Tue 1/8/19	All
Motor Mount Redesign Final Draft	4 days	Thu 1/3/19	Tue 1/8/19	Arik
Solder PCB	4 days	Thu 1/3/19	Tue 1/8/19	Jasmine
Ceiling Mount	4 days	Thu 1/3/19	Tue 1/8/19	Miranda
Draft of Ceiling Mount Design	4 days	Thu 1/3/19	Tue 1/8/19	Miranda
Drill Holes for Rack Support	4 days	Thu 1/3/19	Tue 1/8/19	Jeff
Week 2	5 days	Wed 1/9/19	Tue 1/15/19	All
Develop Fire Sensing Algorithm	5 days	Wed 1/9/19	Tue 1/15/19	Anas
Finalize Specifications	5 days	Wed 1/9/19	Tue 1/15/19	All

Order Motor Mount Material	1 day	Wed 1/9/19	Wed 1/9/19	Arik
Address DR 1.2 Comments	5 days	Wed 1/9/19	Tue 1/15/19	All
Week 3	5 days	Wed 1/16/19	Tue 1/22/19	All
Sensor Communication	5 days	Wed 1/16/19	Tue 1/22/19	Jasmine
Back-up Power Supply	5 days	Wed 1/16/19	Tue 1/22/19	Anas
Power supply calculations	5 days	Wed 1/16/19	Tue 1/22/19	Anas
Order back-up Power Supply	5 days	Wed 1/16/19	Tue 1/22/19	Anas
Fabrication	5 days	Wed 1/16/19	Tue 1/22/19	ME Team
Cut Tubing to Length	5 days	Wed 1/16/19	Tue 1/22/19	Arik
Attach Gear Rack to Tubing	5 days	Wed 1/16/19	Tue 1/22/19	Jeff
Motor Mount Fabrication	5 days	Wed 1/16/19	Tue 1/22/19	Miranda
Actuation Design	5 days	Wed 1/16/19	Tue 1/22/19	ME Team
Design Actuation System/CAD Draft	5 days	Wed 1/16/19	Tue 1/22/19	Arik

Gantry System	5 days	Wed 1/16/19	Tue 1/22/19	ME Team
CAD Draft 3	5 days	Wed 1/16/19	Tue 1/22/19	Miranda
SolidWorks Finite Element Analysis	5 days	Wed 1/16/19	Tue 1/22/19	Miranda
Week 4	5 days	Wed 1/23/19	Tue 1/29/19	All
Testing	5 days	Wed 1/23/19	Tue 1/29/19	EE Team
Control and output location communication	5 days	Wed 1/23/19	Tue 1/29/19	Jasmine
Extinguisher Housing	5 days	Wed 1/23/19	Tue 1/29/19	ME Team
Design Housing/Extinguisher CAD Draft 1	5 days	Wed 1/23/19	Tue 1/29/19	Miranda
Actuation system	5 days	Wed 1/23/19	Tue 1/29/19	ME Team
Actuation CAD draft 2	5 days	Wed 1/23/19	Tue 1/29/19	Arik
Order Parts	5 days	Wed 1/23/19	Tue 1/29/19	Arik
Gantry System	5 days	Wed 1/23/19	Tue 1/29/19	ME Team

Integrate Motor Mount on rails	5 days	Wed 1/23/19	Tue 1/29/19	Jeff
Week 5	5 days	Wed 1/30/19	Tue 2/5/19	All
Design Review 2.1	5 days	Wed 1/30/19	Tue 2/5/19	All
DR 2.1 Documentation	5 days	Wed 1/30/19	Tue 2/5/19	Arik
DR 2.1 Presentation Slide	5 days	Wed 1/30/19	Tue 2/5/19	Miranda
DR 2.1 Practice Video	5 days	Wed 1/30/19	Tue 2/5/19	Jeff
Extinguisher Housing	5 days	Wed 1/30/19	Tue 2/5/19	All
Extinguisher CAD Draft 2	5 days	Wed 1/30/19	Tue 2/5/19	Jeff
Order Housing Material	5 days	Wed 1/30/19	Tue 2/5/19	Miranda
Sweeping Motion CAD Draft 1	5 days	Wed 1/30/19	Tue 2/5/19	Arik
Actuation Code Draft 1	5 days	Wed 1/30/19	Tue 2/5/19	Jasmine
Week 6	5 days	Wed 2/6/19	Tue 2/12/19	All
Testing	5 days	Wed 2/6/19	Tue 2/12/19	All

Accuracy of Motor Movement	5 days	Wed 2/6/19	Tue 2/12/19	Arik
Gantry System	5 days	Wed 2/6/19	Tue 2/12/19	ME Team
Order Limit Switches	5 days	Wed 2/6/19	Tue 2/12/19	Arik
Extinguisher Housing	5 days	Wed 2/6/19	Tue 2/12/19	ME Team
Fabricate Housing	5 days	Wed 2/6/19	Tue 2/12/19	Jeff
2 Sweeping Motion CAD Draft	5 days	Wed 2/6/19	Tue 2/12/19	Arik
Order parts for Sweeping motion	5 days	Wed 2/6/19	Tue 2/12/19	Arik
1 Sweeping motion Code Draft	5 days	Wed 2/6/19	Tue 2/12/19	Jasmine
Week 7	5 days	Wed 2/13/19	Tue 2/19/19	All
Safety	5 days	Wed 2/13/19	Tue 2/19/19	EE Team
Code	5 days	Wed 2/13/19	Tue 2/19/19	Jasmine
Wire	5 days	Wed 2/13/19	Tue 2/19/19	Anas
Test	5 days	Wed 2/13/19	Tue 2/19/19	Jasmine

Gantry System	5 days	Wed 2/13/19	Tue 2/19/19	ME Team
Integrate Extinguisher Housing	5 days	Wed 2/13/19	Tue 2/19/19	Jeff
Fabricate Sweeping Motion system	5 days	Wed 2/13/19	Tue 2/19/19	Arik
Week 8	5 days	Wed 2/20/19	Tue 2/26/19	All
Electrical	5 days	Wed 2/20/19	Tue 2/26/19	EE Team
Integrate Electrical system	5 days	Wed 2/20/19	Tue 2/26/19	Jasmine
Test Electrical system	5 days	Wed 2/20/19	Tue 2/26/19	Anas
Gantry System	5 days	Wed 2/20/19	Tue 2/26/19	ME Team
Test Sweeping motion system	5 days	Wed 2/20/19	Tue 2/26/19	Miranda
Week 9	5 days	Wed 2/27/19	Tue 3/5/19	All
Integrate Mechanical and Electrical System	5 days	Wed 2/27/19	Tue 3/5/19	All
Cable Management	5 days	Wed 2/27/19	Tue 3/5/19	Miranda

Attach system to ceiling	5 days	Wed 2/27/19	Tue 3/5/19	Jeff
Wire sensors	5 days	Wed 2/27/19	Tue 3/5/19	Jasmine
Wire motors	5 days	Wed 2/27/19	Tue 3/5/19	Arik
Connect Power	5 days	Wed 2/27/19	Tue 3/5/19	Anas
Full System Functionality Test	5 days	Wed 2/27/19	Tue 3/5/19	Jeff
Week 10	5 days	Wed 3/6/19	Tue 3/12/19	All
Specification Verification	5 days	Wed 3/6/19	Tue 3/12/19	All
Electrical	5 days	Wed 3/6/19	Tue 3/12/19	EE Team
Mechanical	5 days	Wed 3/6/19	Tue 3/12/19	ME Team
Design Review 2.2	5 days	Wed 3/6/19	Tue 3/12/19	All
DR 2.2 Documentation	5 days	Wed 3/6/19	Tue 3/12/19	Arik
DR 2.2 Presentation Slide	5 days	Wed 3/6/19	Tue 3/12/19	Jasmine
DR 2.2 Practice Video	5 days	Wed 3/6/19	Tue 3/12/19	Anas

Week 11 (Finals Week)	3 days	Wed 3/13/19	Fri 3/15/19	All
Design Review 2.2	3 days	Wed 3/13/19	Fri 3/15/19	All

Spring Quarter

Task Name	Duration	Start	Finish	Owner
Spring Quarter	54 days	Mon 3/25/19	Thu 6/6/19	All
Week 1	5 days	Tue 3/26/19	Mon 4/1/19	
Address DR 2.2 Comments	5 days	Tue 3/26/19	Mon 4/1/19	
Refine Motor mounts	5 days	Tue 3/26/19	Mon 4/1/19	Jeff
Update CAD Drawing	5 days	Tue 3/26/19	Mon 4/1/19	Jeff
Refine Motor Movement Code	5 days	Tue 3/26/19	Mon 4/1/19	Jasmine
Week 2	5 days	Tue 4/2/19	Mon 4/8/19	
Refine Extinguisher Housing	5 days	Tue 4/2/19	Mon 4/8/19	Arik

Refine sensing algorithm	5 days	Tue 4/2/19	Mon 4/8/19	Anas
Week 3	5 days	Tue 4/9/19	Mon 4/15/19	
Refine Sweeping actuation	5 days	Tue 4/9/19	Mon 4/15/19	Arik
Refine Sensor logic	5 days	Tue 4/9/19	Mon 4/15/19	Jasmine
Week 4	5 days	Tue 4/16/19	Mon 4/22/19	
Refine Ceiling Supports	5 days	Tue 4/16/19	Mon 4/22/19	Miranda
Refine Back-up Power	5 days	Tue 4/16/19	Mon 4/22/19	Anas
Week 5	5 days	Tue 4/23/19	Mon 4/29/19	
Design Review 3.1	5 days	Tue 4/23/19	Mon 4/29/19	All
DR 3.1 Documentation	5 days	Tue 4/23/19	Mon 4/29/19	Jasmine
DR 3.1 Presentation Slide	5 days	Tue 4/23/19	Mon 4/29/19	Miranda
DR 3.1 Practice Video	5 days	Tue 4/23/19	Mon 4/29/19	Arik
Week 6	5 days	Tue 4/30/19	Mon 5/6/19	

DR 3.1 revisions	5 days	Tue 4/30/19	Mon 5/6/19	Miranda
Refine Safety code	5 days	Tue 4/30/19	Mon 5/6/19	Jasmine
Week 7	5 days	Tue 5/7/19	Mon 5/13/19	
Test system with changes	5 days	Tue 5/7/19	Mon 5/13/19	Arik
Mechanical Testing	1 day	Tue 5/7/19	Tue 5/7/19	Jeff
Electrical Testing	1 day	Tue 5/7/19	Tue 5/7/19	Anas
Week 8	5 days	Tue 5/14/19	Mon 5/20/19	
Specification Verification	5 days	Tue 5/14/19	Mon 5/20/19	
Electrical	5 days	Tue 5/14/19	Mon 5/20/19	Jasmine
Mechanical	5 days	Tue 5/14/19	Mon 5/20/19	Miranda
Week 9	5 days	Tue 5/21/19	Mon 5/27/19	
Design Review 3.2 Documentation	5 days	Tue 5/21/19	Mon 5/27/19	All
Section 1	5 days	Tue 5/21/19	Mon 5/27/19	Anas
Section 2	5 days	Tue 5/21/19	Mon 5/27/19	Arik

Section 3	5 days	Tue 5/21/19	Mon 5/27/19	Jeff
Section 4	5 days	Tue 5/21/19	Mon 5/27/19	Jasmine
Section 5	5 days	Tue 5/21/19	Mon 5/27/19	Miranda
Week 10	5 days	Tue 5/28/19	Mon 6/3/19	
Design Review 3.2 Document Revisions	5 days	Tue 5/28/19	Mon 6/3/19	Miranda
Final System Tests	5 days	Tue 5/28/19	Mon 6/3/19	Jeff
Week 11 (Finals Week)	1 day	Tue 6/4/19	Tue 6/4/19	
Design Review 3.2 Presentation	1 day	Tue 6/4/19	Tue 6/4/19	All

Appendix

Specification Supporting Material

Sensor Testing Code: Main

Main Class: This class was used to test and record the data from the sensors. It imports the sensor class (below) which grabs the outputs from sensors.

```

"""
Jasmine Gill
Senior Design: RRP Testing
Class to test and record the data from the sensors to be analyzed later

**NOTE: I2C methods for sensor class, getThermalOutput() & getUVindexOutput()

: ADC Channel Methods - getUVOutput(channel), getSmokeOutput(channel),
                        getFlameOutput(channel), getUVS12SDOutput(channel)
"""

import Sensors
import time as t

#Channels on ADC that the sensors are attached to
#Currently only even channels are wired
#uv1Chan = 2
#smokeChan = 2
#flameChan = 2
#uv2Chan = 4

#Initialize sensors
#uv1Sensor = Sensors.Sensors()
#smokeSensor = Sensors.Sensors()
#flameSensor = Sensors.Sensors()
#uv2Sensor = Sensors.Sensors()
thermalCamera = Sensors.Sensors()
#uvIndex = Sensors.Sensors()

#Initialize text files to record data
#uv1Text = open("uvSmallflame.txt", "w")
#smokeText = open("smoke3.txt", "w")
#flameText = open("flame2.txt", "w")
#uv2Text = open("uv2.txt", "w")
thermalText = open("thermalFlame.txt", "w")
thermalTime = open("thermalTimeFlame.txt", "w")
#uvIndexText = open("uvIndex.txt", "w")

def writeToTxtFile(txtFile, data):

    txtFile.write(str(t.time()))
    txtFile.write(", ")
    txtFile.write(str(data))
    txtFile.write("\n")

def writeInitialTime(txtFile):
    txtFile.write(str(t.time()))

```

```

txtFile.write("\n")

def main():
    try:

        #writeInitialTime(uv1Text)
        #writeInitialTime(smokeText)
        #writeInitialTime(flameText)
        #writeInitialTime(uv2Text)
        writeInitialTime(thermalTime)
        #writeInitialTime(uvIndexText)

    while True:

        #Comment out whatever sensors aren't being tested
        """
        uv1Data = uv1Sensor.getUVOutput(uv1Chan)
        writeToTxtFile(uv1Text, uv1Data)

        smokeData = smokeSensor.getSmokeOutput(smokeChan)
        writeToTxtFile(smokeText, smokeData)

        flameData = flameSensor.getFlameOutput(flameChan)
        writeToTxtFile(flameText, flameData)

        uv2Data = uv2Sensor.getUVS12SDOutput(uv2Chan)
        writeToTxtFile(uv2Text, uv2Data)
        """
        thermalData = thermalCamera.getThermalOutput()

        for row in thermalData:
            formatTemp = ['{0:.1f}'.format(temp) for temp in row]
            for num in formatTemp:
                thermalText.write(str(num))
                thermalText.write(" ")
            thermalText.write("\n")

        thermalTime.write(str(t.time()))
        thermalTime.write("\n")

        thermalText.write("\n")
        thermalText.write("New Reading")
        thermalText.write("\n")

        """
        uvIndexData = uvIndex.getUVIndexOutput()
        writeToTxtFile(uvIndexText, uvIndexData)
        """

        t.sleep(.5)

        #Comment out the print lines for the sensors that aren't being used
        """
        print(uv1Data)
        print("")

        print(smokeData)
        print("")

```



```

        print(flameData)
        print("")

        print(uv2Data)
        print("")
        """
        for row in thermalData:
            print(['{0:.1f}'.format(temp) for temp in row])
            print("")
        print("New Reading")
        """
        print(uvIndexData)
        print("")
        """

    except KeyboardInterrupt:
        """
        uv1Text.close()
        smokeText.close()
        flameText.close()
        uv2Text.close()
        """
        thermalText.close()
        thermalTime.close()
        """
        uvIndexText.close()
        """
        print("")
        print("Program Stopped")

if __name__ == "__main__":
    main()

```

Sensor Class

```

"""
Jasmine Gill
Senior Design: RRP Testing
Method to call each sensor and test it using a raspberry pi
Sensors In Order: UV GUVVA S12D, Smoke Sensor, Flame Sensors 595391, IR Array Breakout MLX 90640,
GUVVA S12SD, IR Thermal Camera AMG8833, UV Index Sensor VEML 6075

**NOTE: RPi can't read values for analog sensors, it requires an adc. For
this class we used a MCP3008 chip which is capable of reading outputs
from 7 sensors at a time. The I2C sensors are unaffected, their
outputs can be read without the use of an ADC.**
"""

import RPi.GPIO as g #For the pins
import smbus #For I2C
import time
import adafruit_amg88xx
import adafruit_veml6075
import busio
import board
import Adafruit_GPIO.SPI as SPI
import Adafruit_MCP3008

```

```

#Setting board configurations (Hardware SPI - Using SPI0)
g.setmode(g.BCM)
SPI_PORT = 0
SPI_DEVICE = 0
mcp = Adafruit_MCP3008.MCP3008(spi=SPI.SpiDev(SPI_PORT, SPI_DEVICE))

class Sensors(object):

    def __init__(self):
        print('Sensor initialized')

    """
    Sensor 1: UV GUVA S12D
    Input: UV ray
    Output: Analog output
    Notes: Sensor must remain stationary to output reliable readings
           Power w/ 3.3V not 5, amazon page lies

    pin: Pin # that sensor is connected to
    """
    def getUVOutput(self, pin):

        val = mcp.read_adc(pin)
        return val

    """
    Sensor 2: MQ-2 Smoke Sensor
    Input: Smoke
    Output: Analog output, more smoke = higher voltage
    Notes: No Datasheet
           More smoke = Higher voltage

    pin: Pin # that sensor is connected to
    """
    def getSmokeOutput(self, pin):

        val = mcp.read_adc(pin)
        return val

    """
    Sensor 3: Phantom YoYo Flame Sensor: 595391
    Input: Light Source of flame
    Output: Analog or Digital output (Simple Binary, 1 for flame detected, 0 for not)
    Notes: May need to be ~50 cm to the fire, detection wavelength: 760-1100 nm
           Optimal mounting angle= 60 degrees

    pin: Pin # that sensor is connected to
    """
    def getFlameOutput(self, pin):

        val = mcp.read_adc(pin)
        return val

    """
    Sensor 4: MLX 90640 IR Array Breakout - SparkFun
    Input: Surface Temperature
    Output: Array containing temperatures in the cameras field of view

```

```

Notes:  Detects Surface Area Temperatures from a few feet
        +- 1.5C, Temp Range: -40C to 300C
        Baud Rate: 115200

busPin: Bus pin # (0 or 1) that the array is connected
address: I2C address of the array (0x33)
Chip doesn't work with raspberry pi
"""
def getIRarrayOutput(self, busPin, address, cmdSize):

    self.busPin = busPin
    self.address = address
    self.cmdSize = cmdSize

    bus = smbus.SMBus(busPin)
    time.sleep(1)

    data = bus.read_byte_data(address, 0x07)
    temp = data * .02 - 273.15
    """arraySize = len(data)
    print(arraySize)

    num = 0
    while(num != arraySize):
        print(data[num])
        num+= 1

    print()
    print()"""

    print(data)
    return temp

"""
Sensor 5: UV S12SD - UV Sensor
Input: UV Rays
Output: Analog output
Notes: Sensor must remain stationary to output reliable readings

pin: Pin # that sensor is connected to
"""
def getUVS12SDOutput(self, pin):

    val = mcp.read_adc(pin)
    return val

"""
Sensor 6: 2nd IR Thermal Camera (adafruit)
Input: Temperature
Output: Temp array
Notes: Need adafruit_adxx library
        Max Frame Rate = 10 Hz
        8x8 array of IR Thermal cameras
        Temp Range: 32-176F
        Address: 0x69
"""
def getThermalOutput(self):

```

```

i2c_bus = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_amg88xx.AMG88XX(i2c_bus)

time.sleep(0.1) #Wait for sensor boot

data = sensor.pixels

return data

"""
Sensor 7: UVA/UVB/UV Sensor
Input: UV Rays
Output: UV Index
Notes: Address: 0x10
"""

def getUVindexOutput(self):

    i2c = busio.I2C(board.SCL, board.SDA)

    sensor = adafruit_veml6075.VEML6075(i2c, integration_time = 100)
    data = sensor.uv_index

    return data

"""
Private helper function to scale output from sensors to
values that will be useful for analysis
"""
def __calcWavelength(self, num, minimum, maximum, a, b):

    """min & max are numbers to be scaled
    a & b are values to be scaled to"""
    scaledNum = (((b - a)*(num - minimum)) / (maximum - minimum)) + a
    return scaledNum

```

Motor Testing Code

Main: This class sets the motor pulse, frequency and sends it a position to move to.

```
"""
Test program to move the motor to the desired location
"""

import setepnerMotorClass
import time as t

#Run the motor
def main():
    try:

        motorTest = setepnerMotorClass.stepperMotorClass(40, 36) #position in inches

        motorTest.setPulseTime(500) #microseconds
        motorTest.getPulseTime()

        motorTest.setFrequency(4000) #hertz
        motorTest.getFrequency()

        motorTest.moveMotor()
        motorTest.stopMotors()

    except KeyboardInterrupt:
        g.stop()

if __name__=="__main__":
    main()
```

Stepper Motor Class: This code drives the motor and sets the values specified in main motor class above. Currently the motor does not turn counterclockwise so the motorRev function doesn't work. This will be addressed further in winter quarter.

```

"""
*****
Jasmine Gill
Motor Control Script - Test
Python 3.5.3
Stepper Motor: 34HS46-5004D1
Motor Driver: DM860T
Senior Design - Fall Qtr Mechanical RRP
*****

                DO THIS FIRST
NOTE: Before running the program open the terminal
      and run the following command:
      sudo pigpiod -s 1

*****
Motor Wiring:

Red/Black: A/C
Yellow/Blue: B/D

Steps:

  A B C D
1: + + - -
2: - + + -
3: - - + +
4: + - - +

CW : 1 -> 4
CCW: 4 -> 1

Driver Wiring:

DIR- : RPi GPIO Pin
DIR+ : Rpi GPIO PIn
PUL- : Wire to Ground
PUL+ : Rpi GPIO PIn

ENA- : Don't wire
ENA+ : Don't wire

*****
"""

#Import Libraries
import pigpio
import time as t
import math

#Connect to pigpiod daemon
g = pigpio.pi()

```

```

class stepperMotorClass(object):

    #Class Variables

    #Motor X pins
    motorXDirM = 21
    motorXDirP = 16
    motorXPulP = 12

    #Motor Y Pins
    motorYDirM = 26
    motorYDirP = 13
    motorYPulP = 6

    #Setting default pulse and frequency
    g.set_PWM_frequency(motorXPulP, 4000)
    g.set_PWM_frequency(motorYPulP, 4000)
    pulseTime = 500

    #Pinion Radius in inches
    radius = 1

    #Motor angle
    angle = 1.8

    def __init__(self, xPOS, yPOS):
        if(xPOS >=48 or yPOS >= 48 or (xPOS <=0 and yPOS <= 0)):
            print("INVALID COORDINATES")
        else:
            self.xPOS = xPOS
            self.yPOS = yPOS
            self.__setup()

    def __setup(self):
        #XMotor Pin Setup
        g.set_mode(self.motorXDirM, pigpio.OUTPUT)
        g.set_mode(self.motorXDirP, pigpio.OUTPUT)
        g.set_mode(self.motorXPulP, pigpio.OUTPUT)

        #UNCOMMENT THIS WHEN SECOND MOTOR IS HOOKED UP
        #YMotor Pin Setup
        g.set_mode(self.motorYDirM, pigpio.OUTPUT)
        g.set_mode(self.motorYDirP, pigpio.OUTPUT)
        g.set_mode(self.motorYPulP, pigpio.OUTPUT)

    """
    Method to set the pins for the motor that moves in the
    X-direction. Use this method to set the pins if original
    pins above don't work
    """
    def setXMotorPins(self, motorXDirM, motorXDirP, motorXPulP):

        self.motorXDirM = motorXDirM
        self.motorXDirP = motorXDirP
        self.motorXPulP = motorXPulP

```

```

#X-Movement Motor Pin Setup
g.set_mode(motorXDirM, pigpio.OUTPUT)
g.set_mode(motorXDirP, pigpio.OUTPUT)
g.set_mode(motorXPuLP, pigpio.OUTPUT)

def getXMotorPins(self):
    print("MotorXDirM Pin: ", self.motorXDirM)
    print("MotorXDirP Pin: ", self.motorXDirP)
    print("MotorXPuLP Pin: ", self.motorXPuLP)

"""
Method to set the pins for the motor that moves in the
Y-direction. Use this method to set the pins if original
pins above don't work
"""
def setYMotorPins(self, motorYDirM, motorYDirP, motorYPuLP):

    self.motorYDirM = motorYDirM
    self.motorYDirP = motorYDirP
    self.motorYPuLP = motorYPuLP

#Y-Movement Motor Pin Setup
g.set_mode(motorYDirM, pigpio.OUTPUT)
g.set_mode(motorYDirP, pigpio.OUTPUT)
g.set_mode(motorYPuLP, pigpio.OUTPUT)

def getYMotorPins(self):
    print("MotorYDirM Pin: ", self.motorYDirM)
    print("MotorYDirP Pin: ", self.motorYDirP)
    print("MotorYPuLP Pin: ", self.motorYPuLP)

"""
Set time for pulses
Shorter pulses = faster motor movement supposedly but some don't work
Limitations: 500micr for 800 pulses/rev setting so far
send time in microseconds
"""
def setPulseTime(self, micros):
    self.pulseTime = micros

def getPulseTime(self):
    print(self.pulseTime)

"""
Set the frequency: How fast the pulses are sent
sent frequency in hertz
"""
def setFrequency(self, hertz):
    g.set_PWM_frequency(self.motorXPuLP, hertz)
    g.set_PWM_frequency(self.motorYPuLP, hertz)

def getFrequency(self):
    print(g.get_PWM_frequency(self.motorXPuLP))
    print(g.get_PWM_frequency(self.motorYPuLP))

"""
Set Pinion Radius in inches
"""
def setPinionRadius(self, pinionRadius):

```



```

self.radius = pinionRadius

def getPinionRadius(self):
    print(self.radius)

def setMotorAngle(self, motorAngle):
    self.angle = motorAngle

def getMotorAngle(self):
    print(self.angle)

def stopMotors(self):
    g.wave_clear()

def moveMotor(self):

    g.wave_clear()
    #Motor movement to the coordinates given
    xSteps = self.__calcSteps(self.xPOS)
    self.__motorFWD(xSteps, self.motorXPulP)
    print('boop')
    #self.__motorRev(xSteps, self.motorXDirM)

    #UNCOMMENT THIS WHEN 2ND MOTOR IS HOOKED UP

    ySteps = self.__calcSteps(self.yPOS)
    self.__motorFWD(ySteps, self.motorYPulP)
    print('beep')
    """
    if(ySteps > 2700):
        num2 = ySteps/2700;
        intNum2 = int(num2)

        for i in range (intNum2):
            self.__motorFWD(2700, self.motorYPulP)

            remSteps2 = (ySteps - (2700*intNum2))
            self.__motorFWD(int(remSteps2), self.motorYPulP)

    """

#Private methods
def __dirCW(self):
    #Xmotor
    g.write(self.motorXDirP, 1)
    g.write(self.motorXDirM, 0)
    #Ymotor
    g.write(self.motorYDirP, 1)
    g.write(self.motorYDirM, 0)

def __dirCCW(self):
    #Xmotor
    g.write(self.motorXDirP, 0)
    g.write(self.motorXDirM, 1)
    #Ymotor
    g.write(self.motorYDirP, 0)
    g.write(self.motorYDirM, 1)

```

```

def __motorFWD(self, steps, motorPulsePin):
    self.__dirCW()
    t.sleep(.1) #wait
    g.wave_clear()

    limit = 1100

    if(steps > limit):

        num = steps/limit;
        intNum = int(num)
        print(intNum)

        chain= [0] * (2)
        wf = []
        #Create a chain of waves with 2700 steps
        for i in range(limit):

            wf.append(pigpio.pulse(1<<motorPulsePin, 0, self.pulseTime))
            wf.append(pigpio.pulse(0, 1<<motorPulsePin, self.pulseTime))

        g.wave_add_generic(wf)
        chain[0] = g.wave_create()

        for i in range(intNum):
            g.wave_chain(chain)
            while(g.wave_tx_busy() == True):
                t.sleep(g.wave_get_micros() * 10**-6)

        chain2 = [0] * 2
        #Add the remaining steps to the chain of waves
        remSteps = (steps - (limit*intNum))
        for i in range (remSteps):
            wf.append(pigpio.pulse(1<<motorPulsePin, 0, self.pulseTime))
            wf.append(pigpio.pulse(0, 1<<motorPulsePin, self.pulseTime))
        g.wave_add_generic(wf)
        chain2[0] = g.wave_create()

        #Trasmit the wave chain
        g.wave_chain(chain2)

    else:

        wf = []
        for i in range (steps):
            wf.append(pigpio.pulse(1<<motorPulsePin, 0, self.pulseTime))
            wf.append(pigpio.pulse(0, 1<<motorPulsePin, self.pulseTime))
        g.wave_add_generic(wf)
        wave = g.wave_create()
        g.wave_send_once(wave)

def __motorRev(self, steps, motorPulsePin):

    self.__dirCW()
    t.sleep(.1) #wait

    g.wave_clear()

    limit = 1100

```

```

if(steps > limit):

    num = steps/limit;
    intNum = int(num)
    print('intnum')
    print(intNum)

    chain= [0] * (2)
    wf = []
    #Create a chain of waves with 1100 steps
    for i in range(limit):
        wf.append(pigpio.pulse(0, 1<<motorPulsePin, self.pulseTime))
        wf.append(pigpio.pulse(1<<motorPulsePin, 0, self.pulseTime))

    g.wave_add_generic(wf)
    chain[0] = g.wave_create()

    for i in range(intNum):
        g.wave_chain(chain)
        while(g.wave_tx_busy() == True):
            t.sleep(g.wave_get_micros() * 10**-6)

    chain2 = [0] * 2
    #Add the remaining steps to the chain of waves
    remSteps = (steps - (limit*intNum))
    for i in range (remSteps):
        wf.append(pigpio.pulse(0, 1<<motorPulsePin, self.pulseTime))
        wf.append(pigpio.pulse(1<<motorPulsePin, 0, self.pulseTime))
    g.wave_add_generic(wf)
    chain2[0] = g.wave_create()

    #Trasmit the wave chain
    g.wave_chain(chain2)

else:

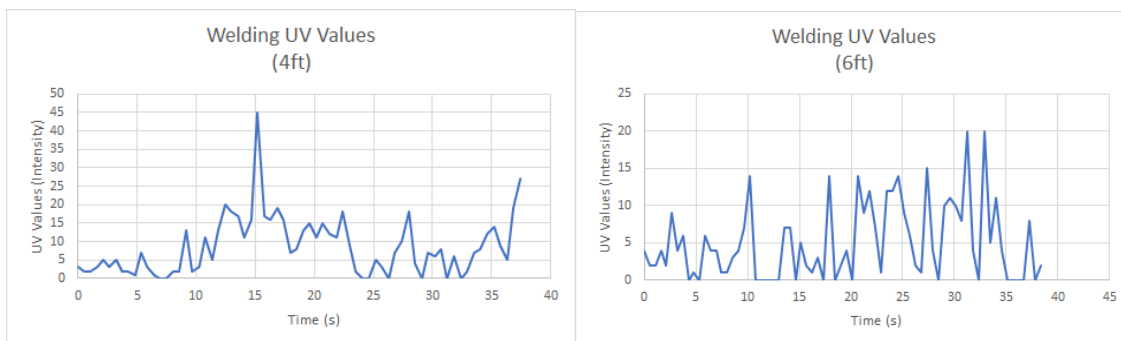
    wf = []
    for i in range (steps):
        wf.append(pigpio.pulse(0, 1<<motorPulsePin, self.pulseTime))
        wf.append(pigpio.pulse(1<<motorPulsePin, 0, self.pulseTime))
    g.wave_add_generic(wf)
    wave = g.wave_create()
    g.wave_send_once(wave)

def __calcSteps(self, position):

    turns = math.degrees(position/self.radius)
    steps = turns/0.225;
    print(steps)
    return int(steps)

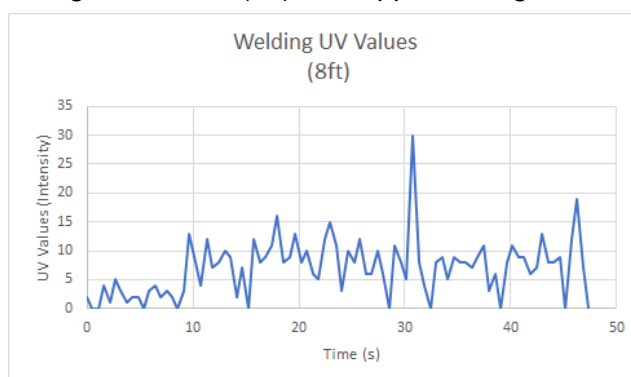
```

Welding Arc vs Fire Detection



Appendix Fig 1: Welding UV Values (4ft)

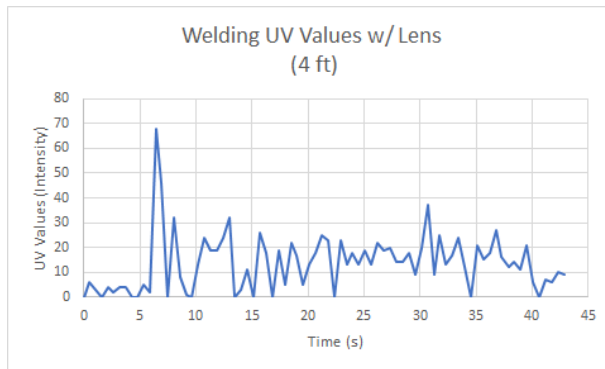
Appendix Fig 2: Welding UV Values (6ft)



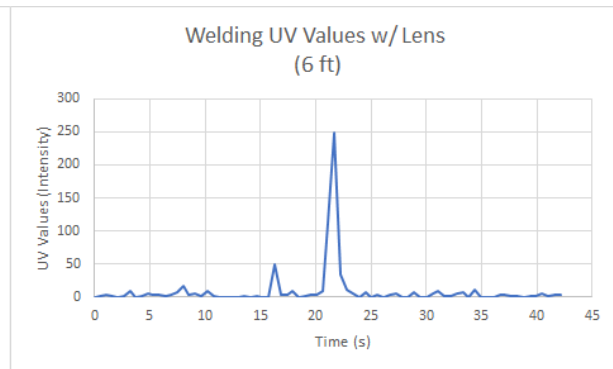
Appendix Fig 3: Welding UV Values (8ft)

One of the concerns that Maroon 5 engineers had was having the system distinguish between a welding arc and a flame. Upon testing the UV sensor on a welding arc and comparing the data gathered to the flame data, it was surmised that the sensors are able to distinguish between the two. However, this was only when the sensor was placed four feet away from the welding area. When the sensors were directed towards the welding arc the UV values (Appendix Fig. 1) were much larger than those of a normal flame (Fig. 5 & 6).

As the UV sensor was moved away from the welding area (Appendix Fig. 2 & 3) it was still able to detect welding but the intensity was much lower. To extend the range of the sensor a lens placed in front of the sensor (similar to the range testing done for flame testing) and it was found that the sensor was able to get values that were much higher (Appendix Fig. 4 & 5). The only problem that was found in this case was that the sensor had to be pointed exactly where the welding arc was being used due to the lens reducing the sensors field of view.



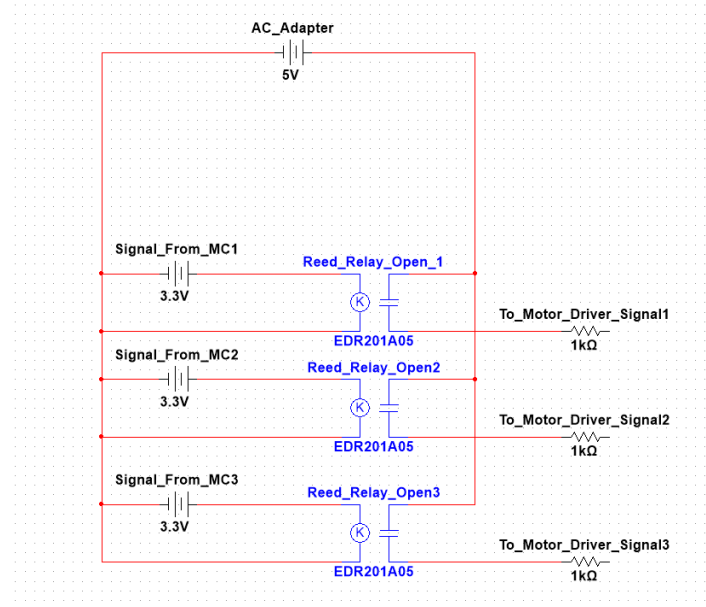
Appendix Fig 4: UV Values (4ft)



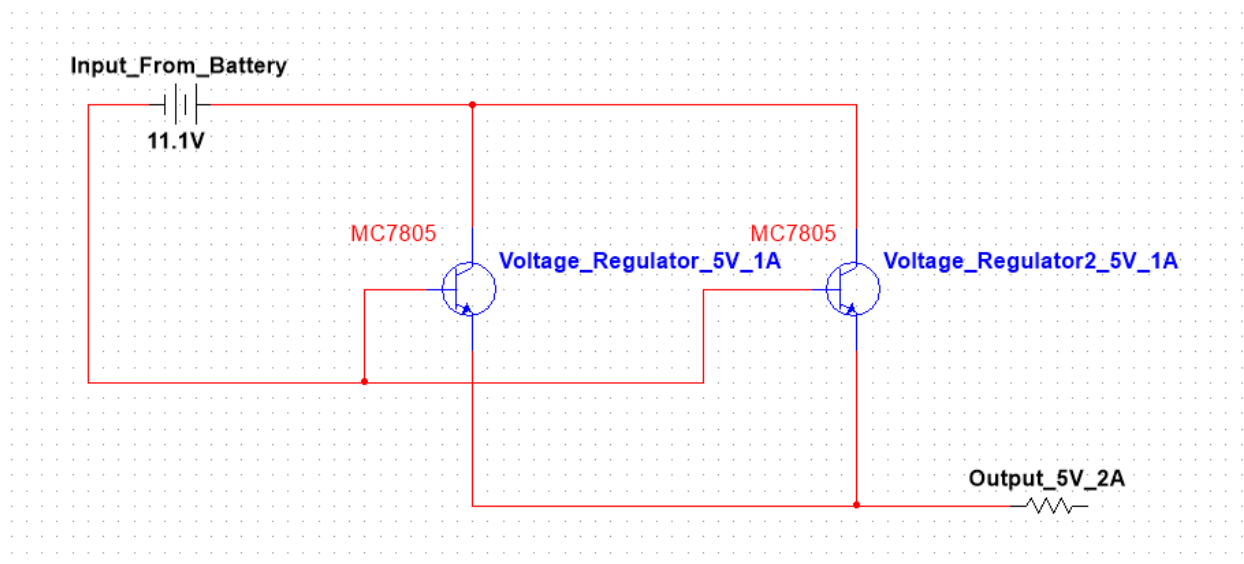
Appendix Fig 5: UV Values (6ft)

Engineering Analysis Supporting Material

Power Consumption

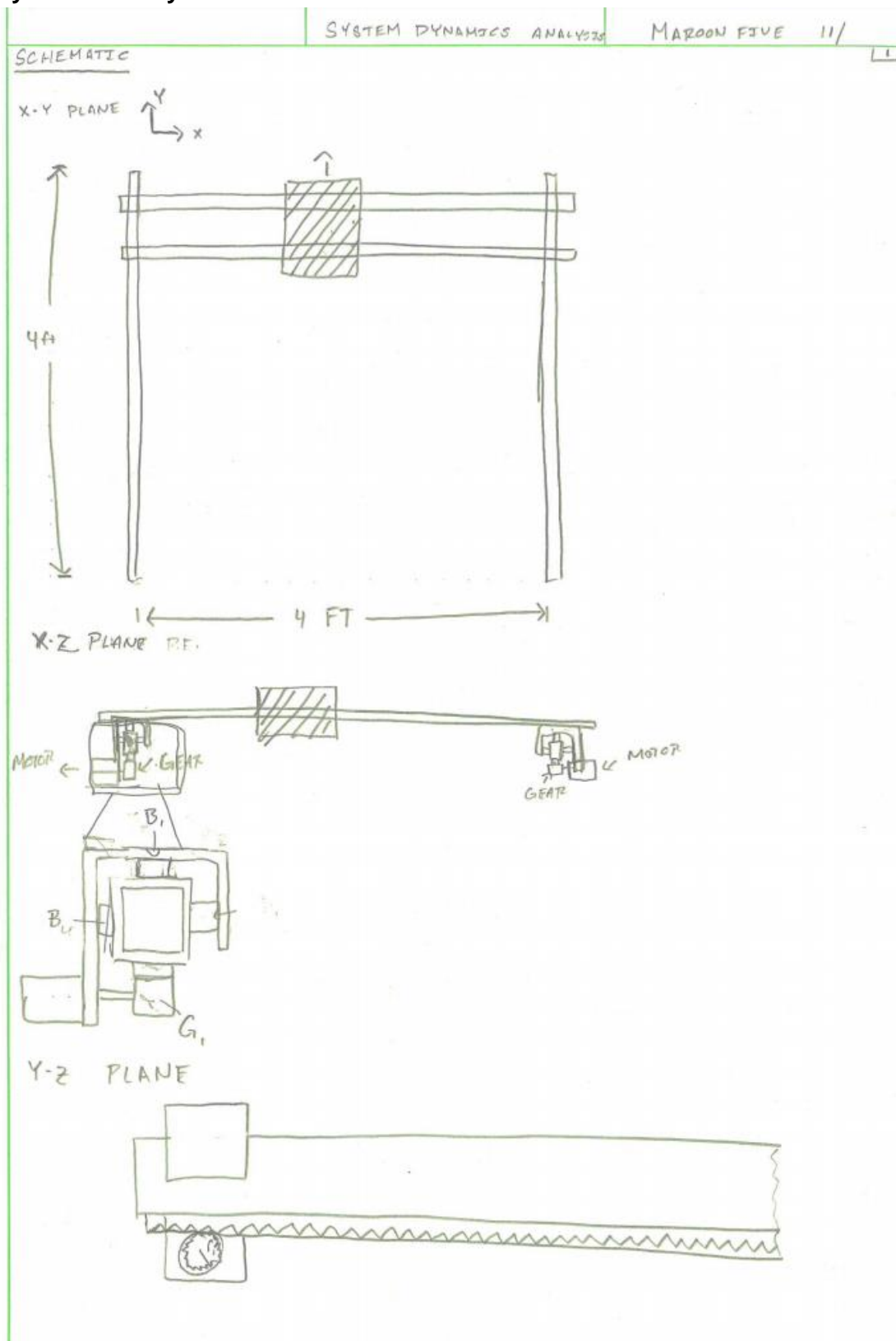


Relays to generate 5V signals



Converting 11.1V to 5V with max. of 2A

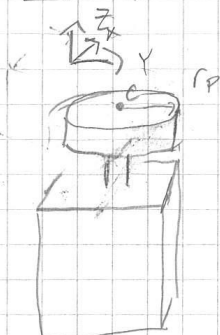
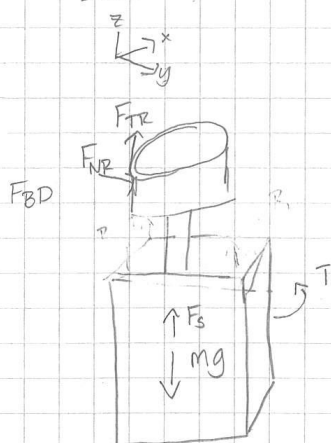
System Dynamics Analysis



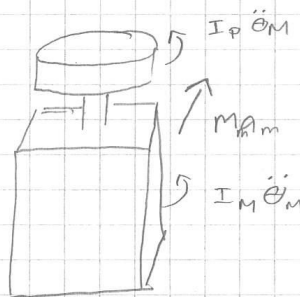
42-301 00 2025 BY SCIENCE & SOCIETY
 42-302 100 2025 BY SCIENCE & SOCIETY
 42-303 000 2025 BY SCIENCE & SOCIETY
 National Brand

SYSTEM DYNAMICS ANALYSIS

MARION FIVE

PULLEY AND MOTORSCHEMATICFBD & MAD

MAD



$$\sum F_z = 0 = F_s - mg$$

$$F_s = mg$$

$$\sum F_x = ma$$

$$F_{TR} = m_m a_m$$

$$(\sum M_c)_{FBD} = (\sum M_c)_{MAD}$$

$$T - r_p \cdot F_{TR} = I_p \ddot{\theta}_m + I_m \ddot{\theta}_m$$

$$T - r_p F_{TR} = (I_p + I_m) \ddot{\theta}_m$$

$$T = (I_p + I_m) \ddot{\theta}_m + r_p F_{TR}$$

Motor Distance Code

```
#Motor Movement Initial Code

from math import *

r = int(input("Enter Pinion Radius (in): "));

position_X = int(input("Enter Desired X coordinate (in): "));

position_Y = int(input("Enter Desired Y coordinate (in): "));

x_turns = degrees(position_X/r);

y_turns = degrees(position_Y/r);

x_steps = x_turns/1.8;

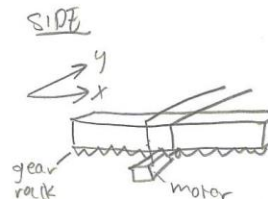
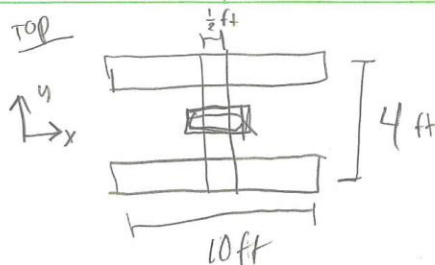
y_steps = y_turns/1.8

print ('Degrees required (x): ', x_turns);
print ('Degrees required (y): ', y_turns);

print ('Steps required (x): ', x_steps);
print ('Steps required (y): ', y_steps);
```

Structural Strength Analysis

Static Analysis



ASSUMPTIONS

- 1) mass of trolley = 40 lb \approx 18 kg
- 2) mass of bridge \approx 30 kg

ANALYSIS

$$\sum F_z = 0 = R_1 + R_2 - F_g$$

$$F_g = 9.81 \frac{m}{s^2} (30 \text{ kg}) = 294.3 \text{ N}$$

$$\therefore R_1 + R_2 = 294.3 \text{ N}$$

$$\sum M_1 = -F_g (-0.609 \text{ m}) + R_2 (1.21 \text{ m}) = 0$$

$$R_2 = \frac{177.5 \text{ N}\cdot\text{m}}{1.21 \text{ m}}$$

$$R_2 = 147.25 \text{ N}$$

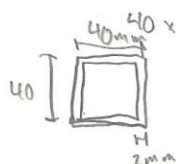
$$R_1 = 147.05 \text{ N}$$

Difference in value caused by conversions and rounding



R_{c1} and R_{c2} are the reaction forces that the ceiling supports must support

aluminium square tube 6061



$$4 \text{ ft} : \frac{.5443 \text{ lb}}{1 \text{ ft}} \cdot 4 \text{ ft} = 2.21 \text{ lb}$$

$$2.21 \text{ lb} = 1.005 \text{ kg}$$

$$\therefore F_g = 9.81 \frac{m}{s^2} (1.005 \text{ kg}) = 9.86 \text{ N}$$

$$\text{for } 10 \text{ ft: weight} = 1.69 \text{ kg}, F_g = 16.27 \text{ N}$$

$$\begin{aligned}\sum F_z = 0 &= R_{c1} + R_{c2} - P - F_g \\ &= R_{c1} + R_{c2} - 150\text{N} - 16.27\text{N} \\ R_{c1} + R_{c2} &= 166.27\text{N}\end{aligned}$$

* $P = 150\text{N}$
from reaction
force previously
calculated, and
is rounded for
safety.

$$\begin{aligned}\sum M_1 = 0 &= -P(1.254\text{m}) - F_g(1.254\text{m}) + R_{c2}(3.048\text{m}) \\ R_{c2} &= \frac{+150\text{N}(1.524\text{m}) + 16.27\text{N}(1.524\text{m})}{3.048\text{m}} \\ R_{c2} &= 83.135\text{N} \\ R_{c1} &= 83.135\text{N}\end{aligned}$$

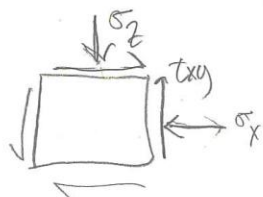
∴ $\left[\begin{array}{l} \text{x-gantry bars must support } 150\text{N} \\ \text{ceiling supports must support } 83\text{N} \end{array} \right]$

for aluminum 6060

$$S_{ut} = \frac{215\text{N}}{\text{mm}^2}$$

$$S_y = \frac{160\text{N}}{\text{mm}^2}$$

$$BA = 60 \times 60$$



$$\sigma_x = 0 \quad \tau_{xy} = 0$$

$$\sigma_z = \frac{83\text{N}}{(46\text{mm} \times 60\text{mm})} = 1.0375 \frac{\text{N}}{\text{mm}^2}$$

$$\sigma_{A,B} = \frac{\sigma_z}{2} \pm \sqrt{\left(\frac{\sigma_x - \sigma_z}{2}\right)^2 + \tau_{xy}^2}$$

$$\sigma_A = 1.0375 \frac{\text{N}}{\text{mm}^2}$$

$$\sigma_B = 0$$

$$\tau_{max} = 0.518 \frac{\text{N}}{\text{mm}^2} = \left(\frac{\sigma_A - \sigma_B}{2}\right)$$

factor of safety by DE (Distortion Energy)

$$\sigma_A \geq \sigma_B, \sigma_B = 0$$

$$So \quad \sigma_1 = \sigma \geq N$$

$$\sigma_2 =$$

$$\sigma_3 = 0 \text{ N}$$

$$\sigma' = (\sigma_A^2 - \sigma_A \sigma_B + \sigma_B^2)^{1/2} \geq S_y$$

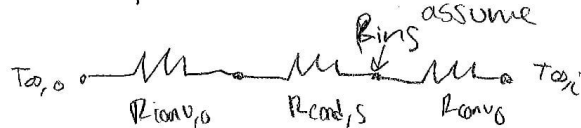
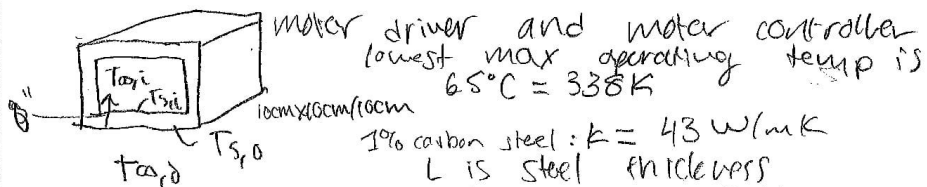
$$= (1.0375 \frac{\text{N}}{\text{m}^2})^{1/2} \geq S_y$$

$$\frac{S_y}{\sigma'} = n, \quad n = \frac{160}{1.0375} = 154 \text{ factor of safety}$$

* this means we may choose a cheaper material if this is a large factor of safety.

Heat Transfer Analysis

Heat Transfer Analysis



*assume
Steady state,
constant properties
and uniform
surface temps

$$R_{tot}'' = \frac{1}{h_o A_o} + \frac{L}{k_s A} + \frac{1}{h_i A_i}$$

find required resistance (R_{tot}) value
to have $T_{a,i} = 338\text{K}$

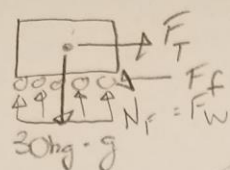
$$R_{tot} = \frac{\Delta T}{Q}$$

$$\Delta T = 700\text{K} - 338\text{K}$$

$$= 362\text{K}$$

plug in values when surface area is
defined

Motor Power Calculations



$\sum F_y = 0 = (30\text{kg})(9.81\text{m/s}^2) + N_f = 0$
 $\sum F_x = F_T - F_f$
 $\sum F_x = 30\text{kg} \cdot \left[\frac{(2.5\text{m/s})}{\frac{1}{2}\text{sec}} \right]$

max velocity
 acceleration desired
 achieve max velocity in 1/2 sec.

$\sum F_x = 150\text{N}$

$F_f = (30\text{kg}) \cdot (9.81\text{m/s}^2) \cdot (0.15)$

$F_f = 44.145\text{N}$

$150\text{N} = F_T - 44.145\text{N}$

$F_T = 194.145\text{N}$ → required force to meet speed specs

μ_s for steel on steel

Torque:
 $T_{\text{req}} = F_T \cdot r_{\text{pinion}}$
 $T_{\text{req}} = (194.145\text{N}) \cdot (0.0254\text{m}) = 4.9276\text{Nm}$
 $T_{\text{req}} = 4.9276\text{Nm}$
 x2 factor of safety
 $T_{\text{req}} = 9.8552\text{Nm}$

Torque Curve

Power vs Torque

References

[1]

07, 2017 Feb. "Preventing the Five Major Causes of Industrial Fires and Explosions." *Occupational Health & Safety*, ohsonline.com/articles/2017/02/07/preventing-the-five-major-causes-of-industrial-fires-and-explosions.aspx.

[2]

Wilhite, David. "The Case For Supplementary Fire Suppression." *Modern Machine Shop*, Modern Machine Shop, 11 Sept. 2009, www.mmsonline.com/articles/the-case-for-supplementary-fire-suppression.