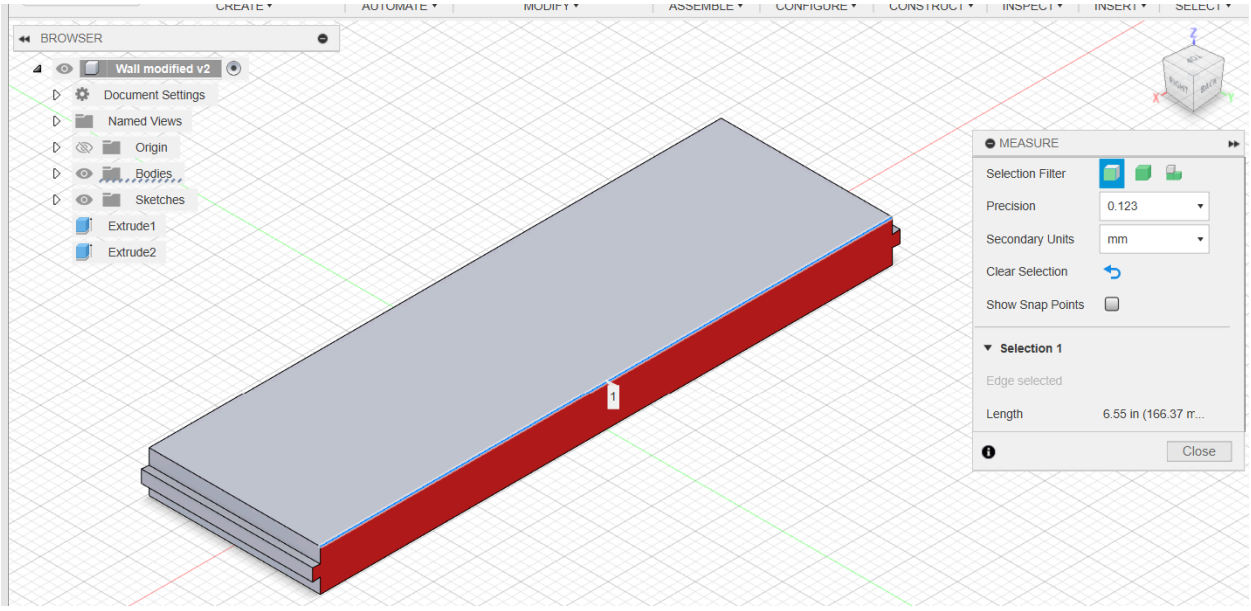
Mouse Version 7.0

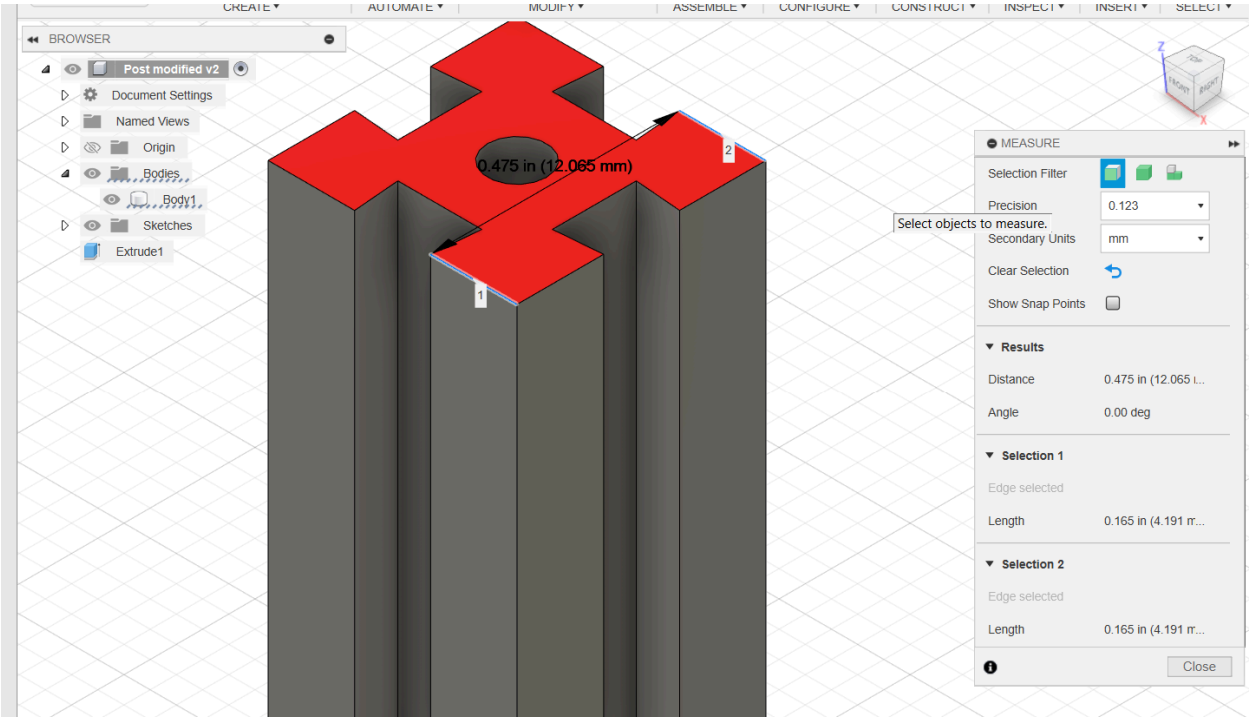
Objectives:

- Hardware
    - Serial console over bluetooth
        - Look into TI chips
    - IR sensor replacement
        - Test different circuits and design a custom replacement to SHARP sensor while still using their enclosure
    - Vacuum
        - Switch to MOSFET switch instead of driver
    - Misc fixes
        - Remove reset switch
        - Fix low battery comparator circuit
        - Swap out long IR sensor wires and crimp instead, or ditch wires completely if IR sensor replacement works out
- Software
    - Maze algorithm
        - Incorporate parameters for time taken for different movements
    - Command line interface
        - Debugging interface to calibrate mouse to provided maze
            - Provide wall detection thresholds and PID constants
        - Manual movements for debugging
    - Clean up
        - Decide on source code template to use across all firmware and other high level software
        - Finish SOLID principles implementation for embedded firmware

6/24/2024:

- Back on the grind
- Soon sent us 3D models





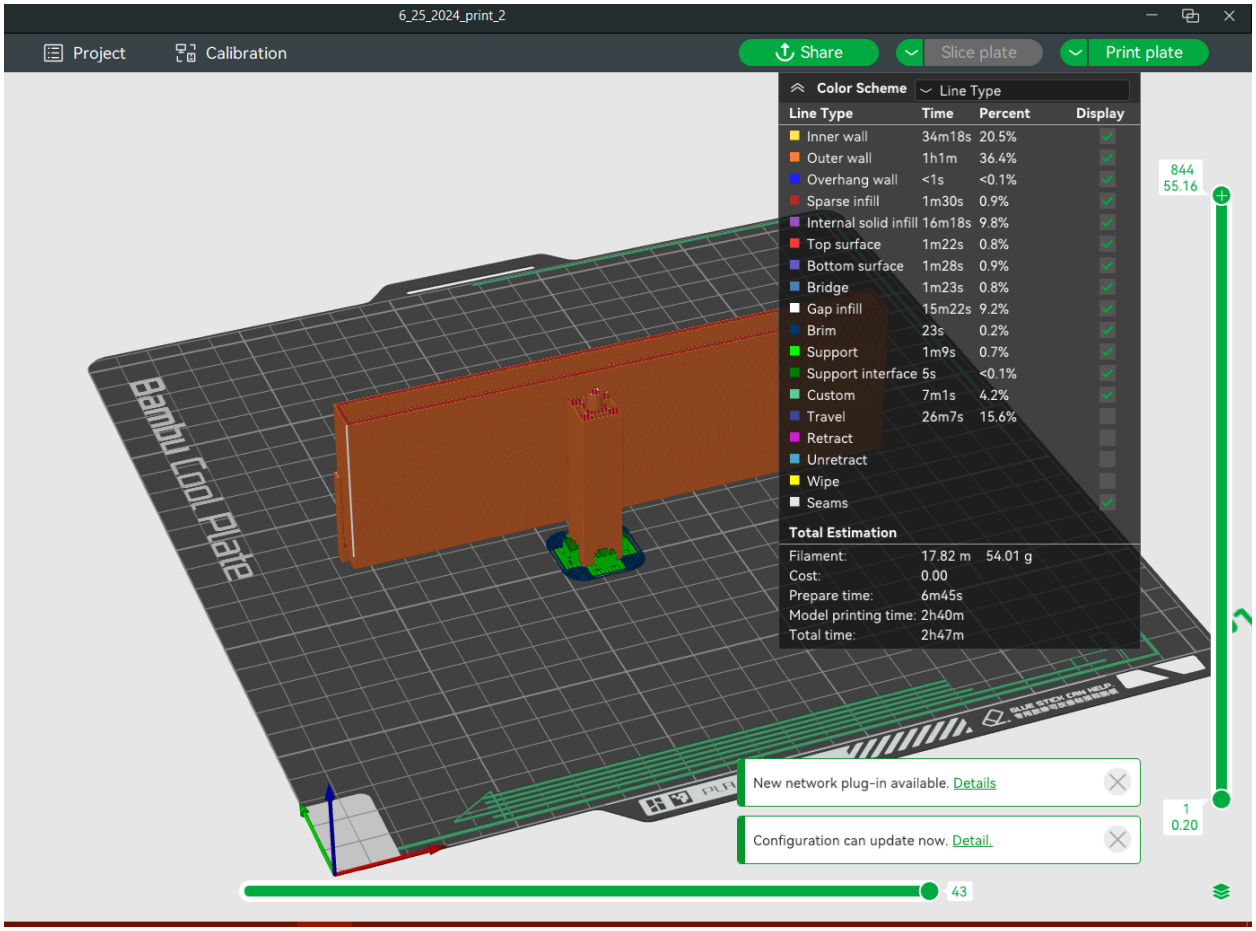
- So (6.55in) 166.37mm + (0.475in) 12.065mm = (7.025in) 178.435mm

- Let's ask dad if he's down to make a bunch of panels with holes in them
- If not buying wooden planks and drilling holes in them doesn't sound too bad…

- 0.2mm on both sides is too much, 0.02mm on both sides is not enough
- 0.1mm on both sides next

6/25/2024:

- Aight so wall and post are done



- Made sleeve versions of wall and post for easy fit
- So this costs about 55g of filament
- Each roll has 1000g of filament, so if we want a new 5x5 board:

- 5x5 maze means 6walls*5 * 2 = 60 walls and posts needed
- 1000g/55g = 18.18 walls and posts per roll
- 60 walls and posts needed / 18 walls and posts per roll = 3.3 rolls needed
- If we have 4 rolls we can make a 5x5

- Wall is 6.55in, post is 0.475in, so 6.55in * 5 + 0.475in * 6 = 35.6in each side of maze
- Fits on this cork board: HERE 48in x 36in hopefully…

7/1/2024:

- We need to figure out what the heck we have and don't have before we order more specific parts
- Let's get metallic spacers: HERE
- Our main objective is to build a new maze, so let's order filament for now

7/4/2024:

- Aight so we have the cork board for the 5x5
- We can punch holes into it fine, but it's not as sturdy as wood
- Gonna try to make sleeves for the posts to slit into

- Aight finally finished updating inventory list we've been procrastinating
- Now we're ready to consider new parts

8/31/2024:

- It's been 2 months
- Here we finally are

Todo:

- Finish maze
    - Print wall mounts
    - If wall mounts are good, spam print walls, posts, and mounts
- Make new mouse rev and order
    - ~~Change power node naming scheme to 3V3, etc~~
    - ~~Find tiny 0402 decoupling caps for MCU~~
    - ~~Find tiny 0402 LEDs~~
    - ~~Find smaller inductors for 5V and 3.3V regulators~~
    - ~~Find smaller pushbutton for config~~
    - ~~Remove reset circuit~~
    - ~~Fix comparator circuit~~
        - ~~Need two extra resistors to scale down output~~
    - ~~Remove vacuum driver circuit and create mosfet switch instead~~
    - Make holes for IR sensor mount
    - Any other changes to try and reduce PCB size
- Upgrade mounts
    - Make pockets for screw nut on wheel motor mount, vacuum mount, and IR sensor mount
    - Extend IR sensor mount to plug into PCB
- Other new materials
    - Wires to crimp IR sensors
    - Conformal coating

- Voltage divider for comparator output:

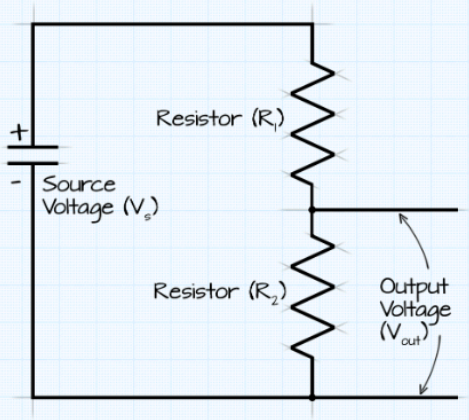
Voltage Divider Calculator

A voltage divider circuit is a very common circuit that takes a higher voltage and converts it to a lower one by using a pair of resistors. The formula for calculating the output voltage is based on Ohms Law and is shown below.

$$V_{out} = \frac{V_s \times R_2}{(R_1 + R_2)}$$

where:

- V_S is the source voltage, measured in volts (V),
- R_1 is the resistance of the 1st resistor, measured in Ohms (Ω),
- R_2 is the resistance of the 2nd resistor, measured in Ohms (Ω),
- V_{out} is the output voltage, measured in volts (V),



Enter any three known values and press "Calculate" to solve for the other.

Voltage Source (V_S)	12	Volts (V)
Resistance 1 (R_1)	10000	ohms (Ω)
Resistance 2 (R_2)	3793.103	ohms (Ω)
Output Voltage (V_{OUT})	3.3	Volts (V)
	Calculate	Click "Calculate" to update the field with orange border.

  - 3.74kOhms gives us like 3.2V across R2, so we'll go w/ that

9/1/2024:

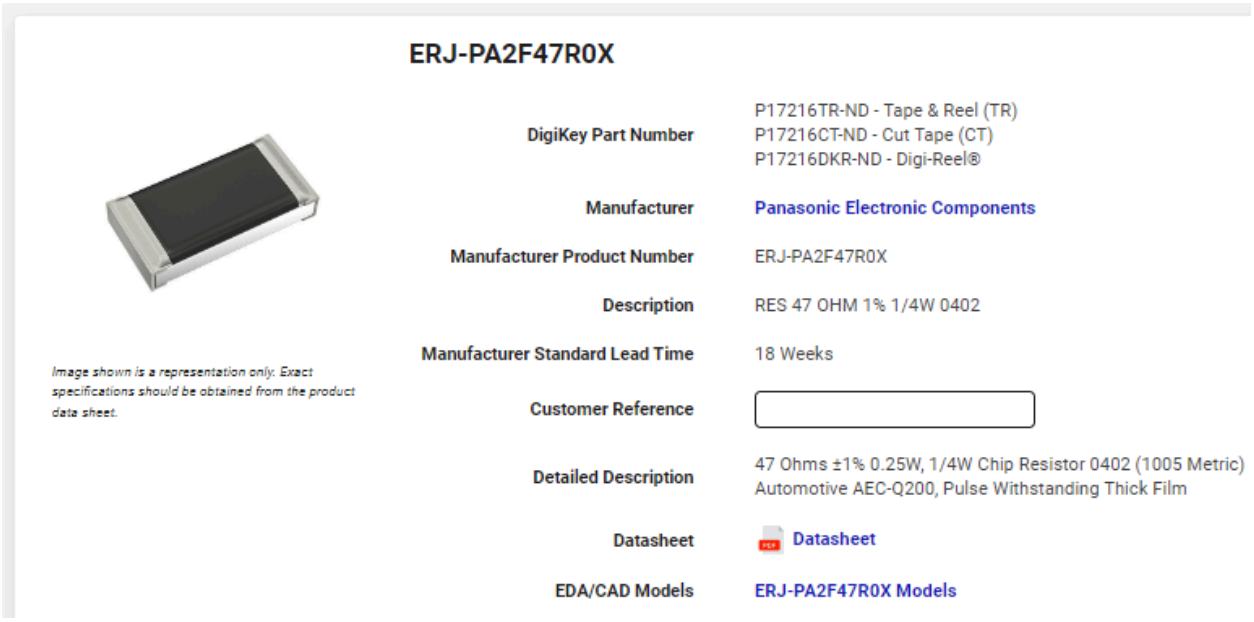
- Let's finish this

- Calculating gate resistor for mosfet:
  - Gate current = gate capacitance * (gate voltage / switching time desired):
    - Ig = Ciss * (Vg / tsw)
  - Gate resistor = gate voltage / gate current:
    - Rg = Vg / Ig, Rg = tsw / Ciss

- So plugging in numbers:

Diode Forward Voltage (Note 6)	V_{SD}	—	—	0.9	V	V_{GS} = 0V, I_S = 1A
DYNAMIC CHARACTERISTICS (Note 7)						
Input Capacitance	C_{iss}	—	290	—	pF	V_{DS} = 10V, V_{GS} = 0V
Output Capacitance	C_{oss}	—	66	—	pF	

- Let’s say we want to switch at 10ns, or 100MHz freq
- Ciss for our mosfet is 290pF
- 10ns / 290pF = 34 ohms
- This means that current is 3.3V / 34 Ohms = 95mA
- …that means that power is 95mA*3.3V = 0.31581W
- Our goto quarter Watt resistors aren’t going to do the trick?
- That’s annoying so going to reverse
- 3.3V / (0.25W / 3.3V) = 43 Ohms
- We’re switching at 13nS w/ this?
- That’s fine

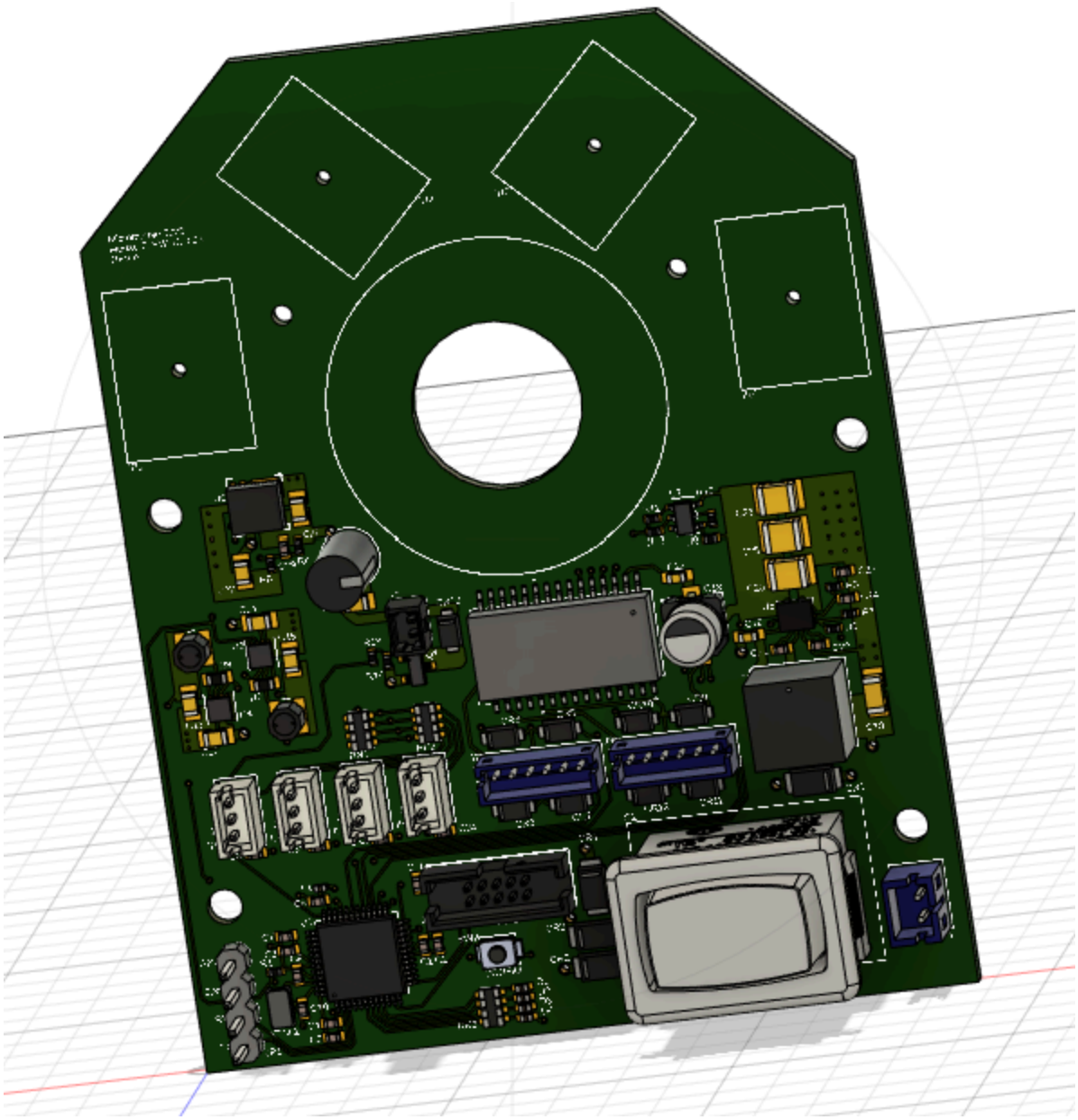


ERJ-PA2F47R0X

Image shown is a representation only. Exact specifications should be obtained from the product data sheet.

DigiKey Part Number	P17216TR-ND - Tape & Reel (TR) P17216CT-ND - Cut Tape (CT) P17216DKR-ND - Digi-Reel®
Manufacturer	Panasonic Electronic Components
Manufacturer Product Number	ERJ-PA2F47R0X
Description	RES 47 OHM 1% 1/4W 0402
Manufacturer Standard Lead Time	18 Weeks
Customer Reference	
Detailed Description	47 Ohms ±1% 0.25W, 1/4W Chip Resistor 0402 (1005 Metric) Automotive AEC-Q200, Pulse Withstanding Thick Film
Datasheet	Datasheet
EDA/CAD Models	ERJ-PA2F47R0X Models

- We’ll go w/ this



- Alright donezo
- Moving on to 3D printing mode
- We'll queue up our wall mount and see how it goes
- In the meantime we'll add holes in our PCB for IR sensor mounts, and then PCB'll be good to go

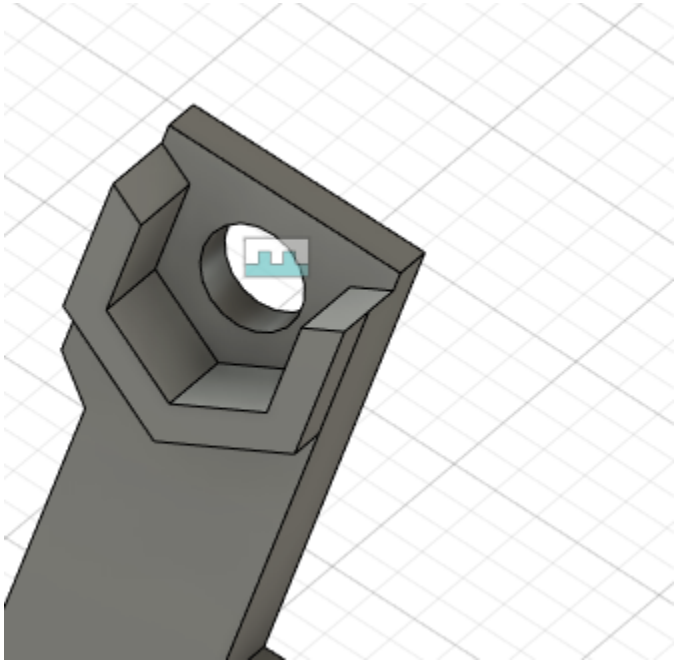
- Alright the maze post sleeve is good to go
- New pcb and digikey materials are good to go too

Next steps:

- Upgrade mounts
    - ~~Wheel motor mount & battery mount~~
        - ~~Add pockets for hex nuts (M3)~~
    - ~~IR sensor mount~~
        - ~~Add pockets for hex nuts (M2)~~
    - ~~Vacuum mount~~
        - ~~Add pockets for hex nuts (M1.6)~~
    - Print all and test
- Build maze
    - Mark up cork board for where walls and posts should go
    - Print posts, walls, post sleeves
    - Put everything together and consider adding a cut in the maze for easy traveling
- Order missing materials
    - ~~Wires to crimp and attach to IR sensors~~
    - ~~Conformal coating~~
- Bluetooth
    - Look into the new SoC from before
        - Check out IDE, SDK and look for bluetooth serial support
        - If it looks good, order dev board and test bluetooth serial capability
        - Design bluetooth module board for serial and attach to mouse 7 and test

9/2/2024:

- Alright out we go



- Let's see if this does the job
- Aight it's pretty solid
- Could add extra tab on back, but probably better to just super glue the nuts on the mounts to maintain tolerance

- Motor is conflicting w/ new motor mount design
- After fixing that we can move on to the maze

Next steps:
- Upgrade mounts
    - Print all parts and test
- Build maze
    - Mark up cork board for where walls and posts should go
    - Print posts, walls, post sleeves
    - Put everything together and consider adding a cut in the maze for easy traveling
- Create PID simulation script
    - Put together python script to simulate mouse going through a maze corridor
- Bluetooth
    - Look into the new SoC from before
        - Check out IDE, SDK and look for bluetooth serial support
        - If it looks good, order dev board and test bluetooth serial capability
        - Design bluetooth module board for serial and attach to mouse 7 and test

9/7/20204:
- Back
- Objectives for this weekend:
    - Finish building physical maze
    - ~~Design new mouse rev~~
        - ~~Drill smaller holes for M1.6 screws instead of M3 for motor mounts~~
        - ~~Move pushbutton closer to inside of mouse~~
        - ~~Bulk cap~~
        - ~~Fuse~~
    - ~~Mouse mounts~~
        - ~~Change board to wheel mount screw size from m3 to m1.6 and put slot on the bottom side instead of top~~
        - ~~Switch slot on vacuum mount from top to bottom~~
    - Put together mouse PID simulation in Python
    - Design bluetooth module

9/8/2024:

- Maze building in progress
- Cork board is so damn fragile
- We need a post removing tool to avoid ripping the post sleeve out of the maze each time the posts are removed

- Let's finish up mounts while we wait for things to dry

- Mounts updated
- We'll quickly put together our post remover next and then look into bluetooth

9/15/2024:
- Walls updated try to make them a bit less fragile
- All post sleeves hammered into maze, waiting to dry

- Bluetooth
    - Screw all of these SoC's we're gonna go w/ the ESP32 since it has a whole lot of support and it looks like it can definitely serve as a serial bridge
    - Has UART support and classic bluetooth serial port profile support

- ~~ESP32-C6 WROOM~~ ESP32-PICO-MINI-02-N8R2 Module
    - Digikey link for module: HERE
    - Fusion files
        - 3D model on mouser: HERE
        - UL PCB layout and symbol: HERE
    - ESP32 IDF SDK documentation for ESP32 (generic for all ESP32)
        - UART: HERE
        - Bluetooth Serial Port Profile: HERE
        - Arduino serial example for ESP32-PICO-MINI-02: HERE
    - Hardware references
        - Sparkfun dev board for SPP: HERE

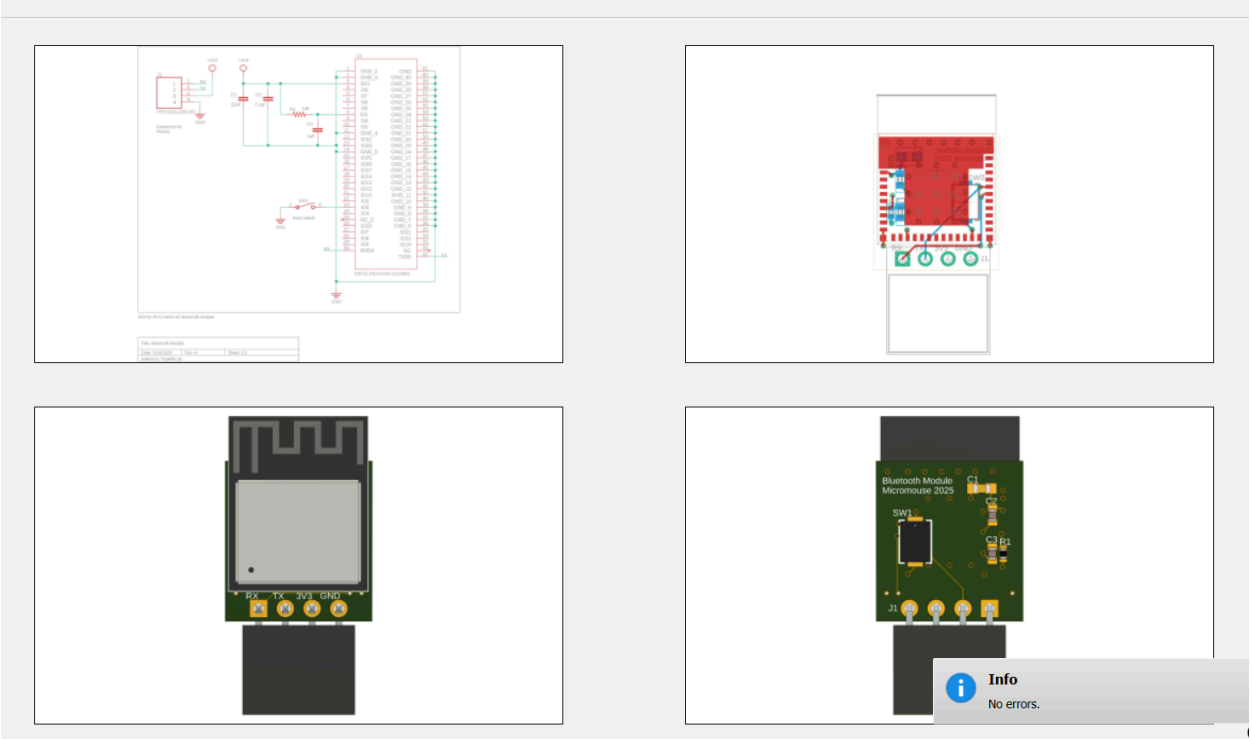
        - Sparkfun dev board for generic use: HERE
        - USB to serial breakout to program the ESP32: HERE
        - Video of ES32-S3 basic setup: HERE
        -

Next steps:

- Finish building maze - **Ryuichi**
    - Finish walls and posts
- Build new mouse 7 and test - **Ryuichi**
    - Clean firmware and create new Git repository
    - Create documentation and guides
        - Fusion 360 guide
        - Micromouse startup guide
        - Bare minimum Git guide (conventional Git commits too)
        - Semantic versioning, Doxygen, source code formatting guide
        - Block diagram of firmware
- Bluetooth board
    - Test w/ development board - **Chris F**
        - Does the boot pushbutton need to be pressed to upload code or run code that's been uploaded?
        - Can the ESP32 be programmed w/ USB easy?
    - ~~Design board and circuit- **Ryuichi**~~
        - ~~Right angle TH pins to get UART and 3.3V from mouse~~
        - ~~Other parts as needed using module datasheet and Sparkfun dev board schematic as references~~
- Python simulation - **Chris L**
    - After building mouse get numbers so Chris can put together simulation
        - Wheel positions
        - Mouse max and min speed as a function of wheel diameter
        - Mouse simplified rectangular hitbox
        - Mouse IR sensor locations on rectangular hitbox
        - Mouse IR sensor polling frequency
        - IR sensor piecewise equation
        - Mouse wall and post dimensions
    - Put together Sphinx template to use
    - Generating PID constants
        - One wall PD
        - Two wall PD
        - Turning PD
        - Diagonals

    - Generating IR sensor thresholds for wall detection
- Command line interface - **Andy**
    - Manual mouse movement
    - Mouse maze traversal progress and maze printing
    - PID constant assignment
    - IR sensor threshold assignment
- Maze algorithm updates- **Chris F**
    - Prepare maze algorithm for diagonal movements and time taken for each type of movement

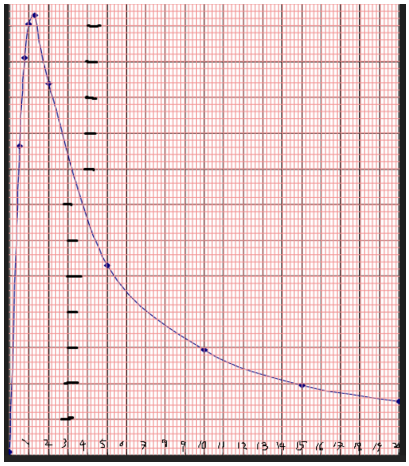
- Aight now we finally have our bluetooth SoC
- This thing is perfect for our application
- Let's make a quick board anticipating Chris's testing to work

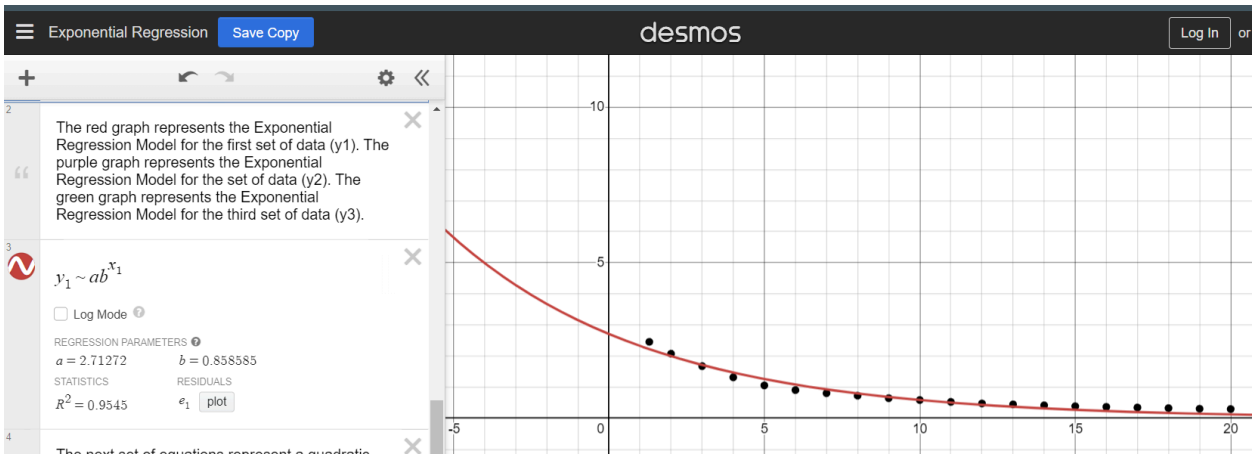


- Fantastic

9/26/2024:

- Working on maze
- Getting regression fit for SHARP IR sensor:





- Graph: HERE
- So the equation is: $y = ab^x$ where a = 2.71272, b = 0.858585
- 95% is aight
- If we want to get the ADC value we account for the voltage divider and 10 bit ADC:
- **ADC reading = ((((2.71272)(0.858585)^x) * (2.2 / (1+2.2))) / 1.8V) * 1024**

9/27/2024:

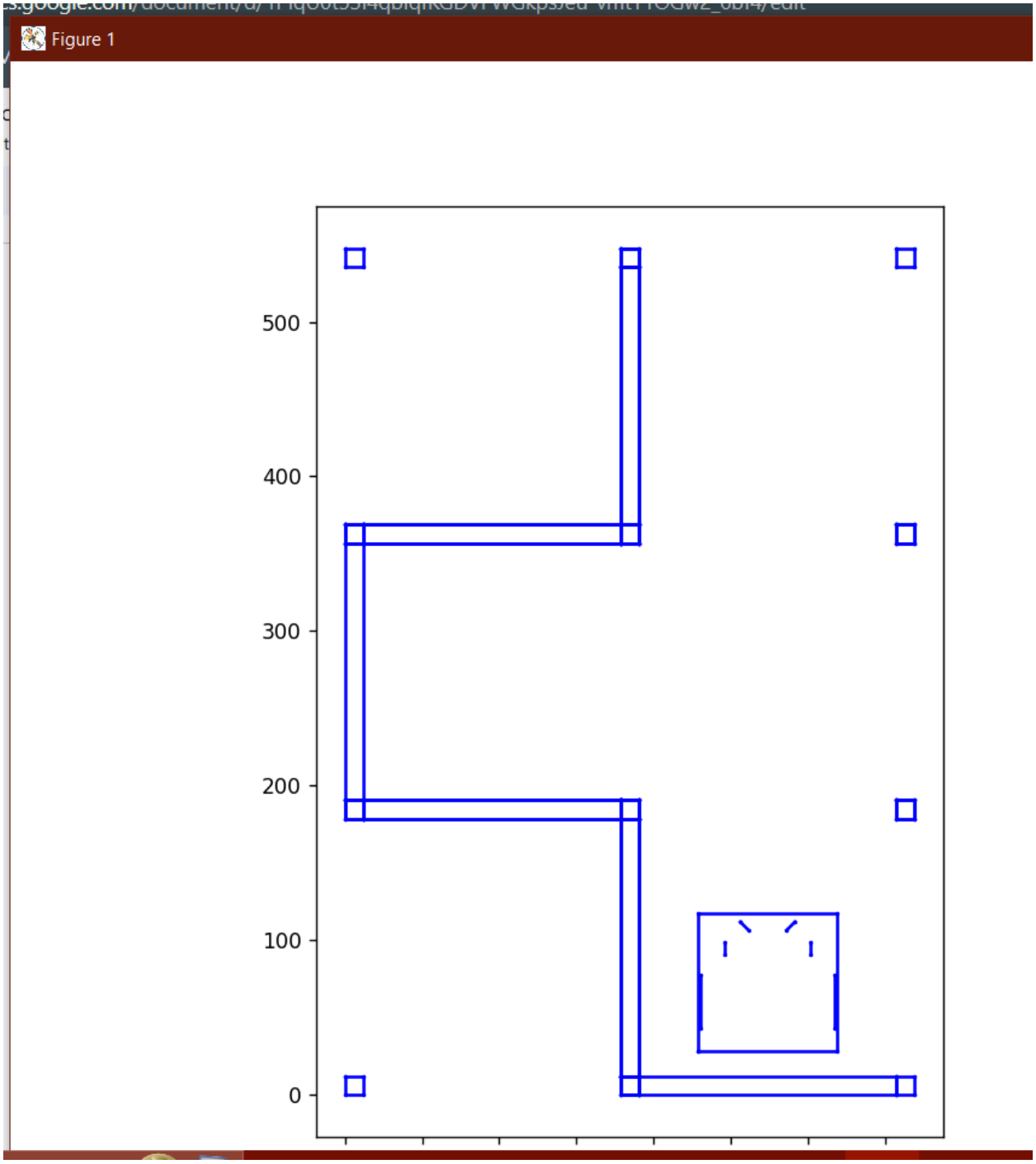
- Mouse itself is 69mm x 89mm (w x l) or so
- 2mm spacers
- 6.8mm from spacer to center of wheel
- Wheel location from bottom of mouse: 14.53mm and 49.53mm
- Top left corner of front IR sensors about 1.27mm from sides of mouse, 19.05mm from top of mouse
- Top right corner of diagonal sensors about 20.83mm from sides of mouse, 1.52mm from top of mouse
- IR sensor w/ mount is 11.3mm x 15mm (w x l)

10/6/2024:

- Aight physical maze is done
- Working on Python simulation, left off on maze class

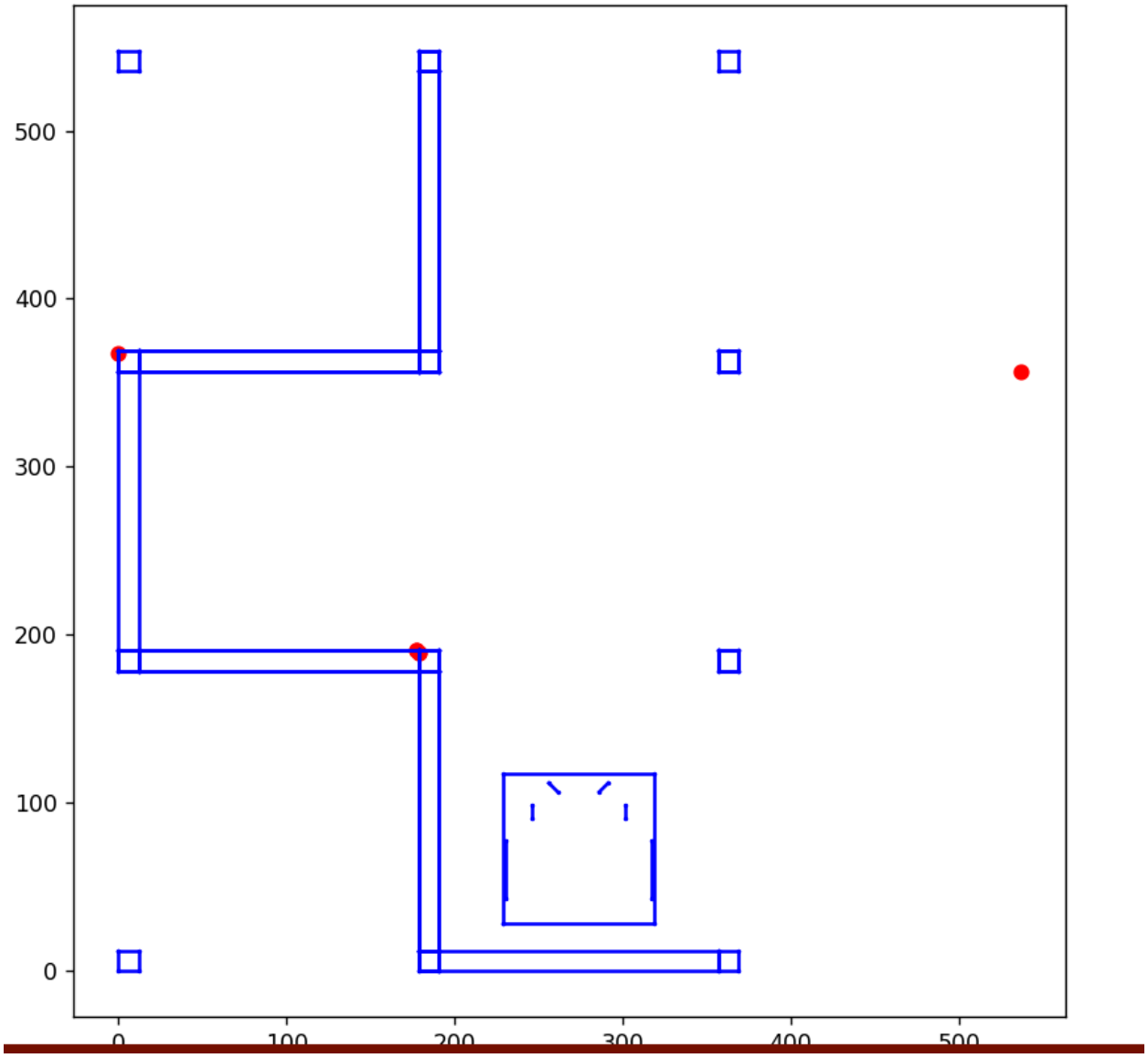
10/7/2024:

- Left off on get_shortest_sensor_distances() to check every line segment in a square against mouse sensor in mazesquare class to then port that over to maze class to check all squares around a mouse
- Need to remember to check for visibility to minimize comparisons

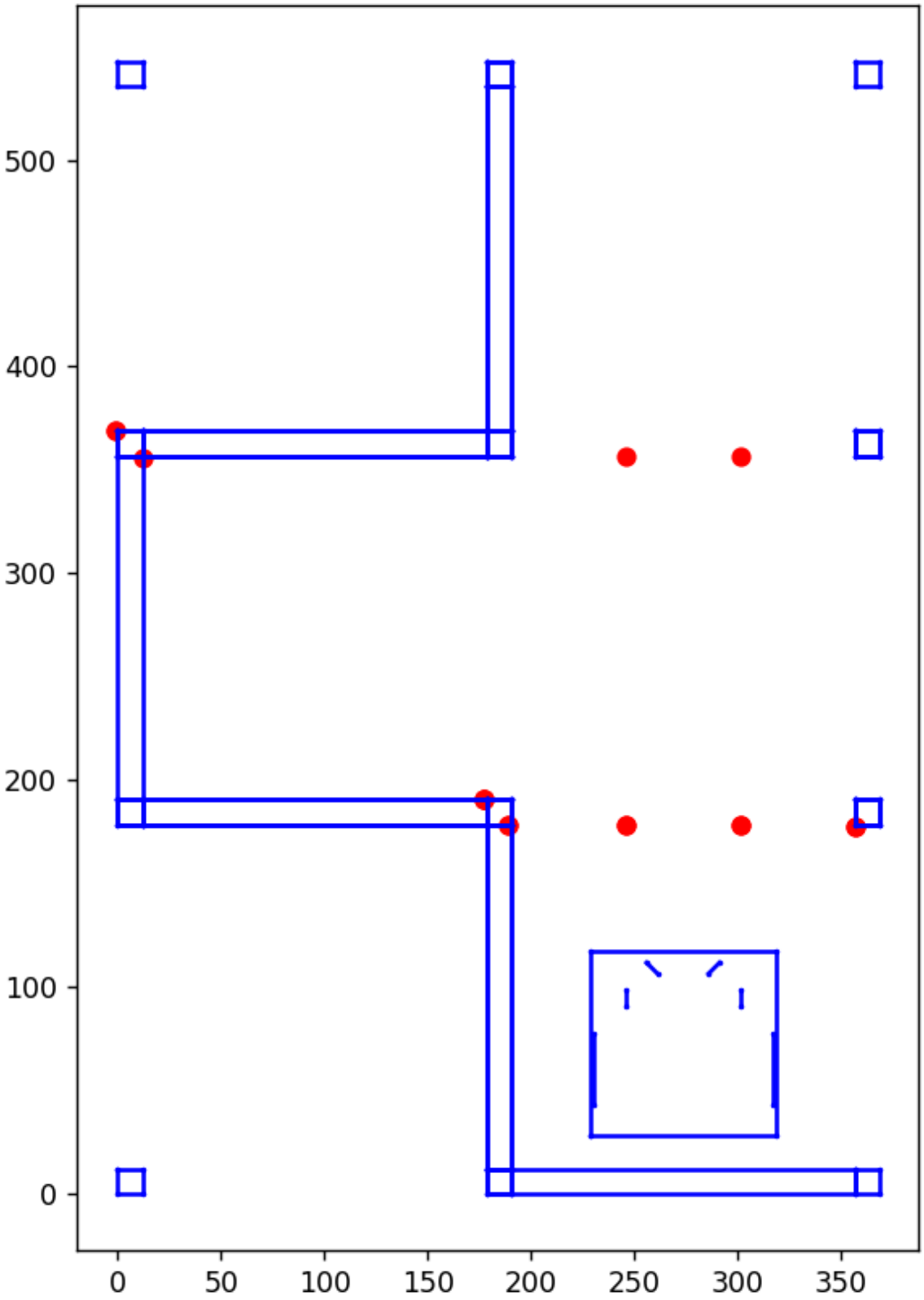


- Fantastic though
- Very cool
- We’ll get a graph going w/ this shi asap

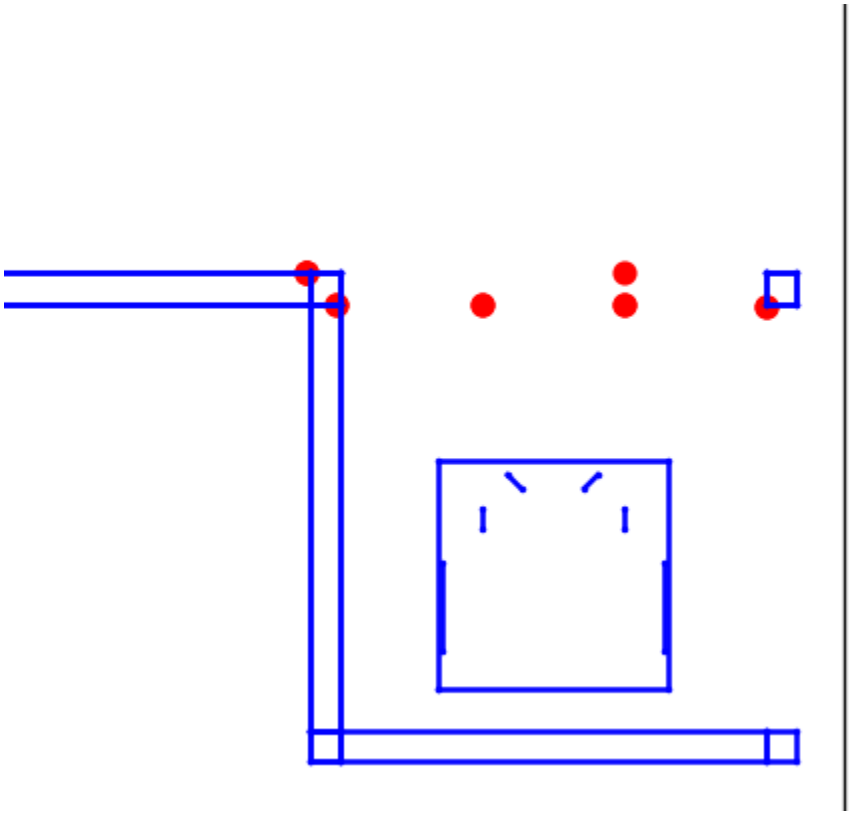
10/8/2024:



- Not bad
- There’s something bugged in mazeSquare’s get_shortest_sensor_distances() or deeper


500
400
300
200
100
0
0
50
100
150
200
250
300
350

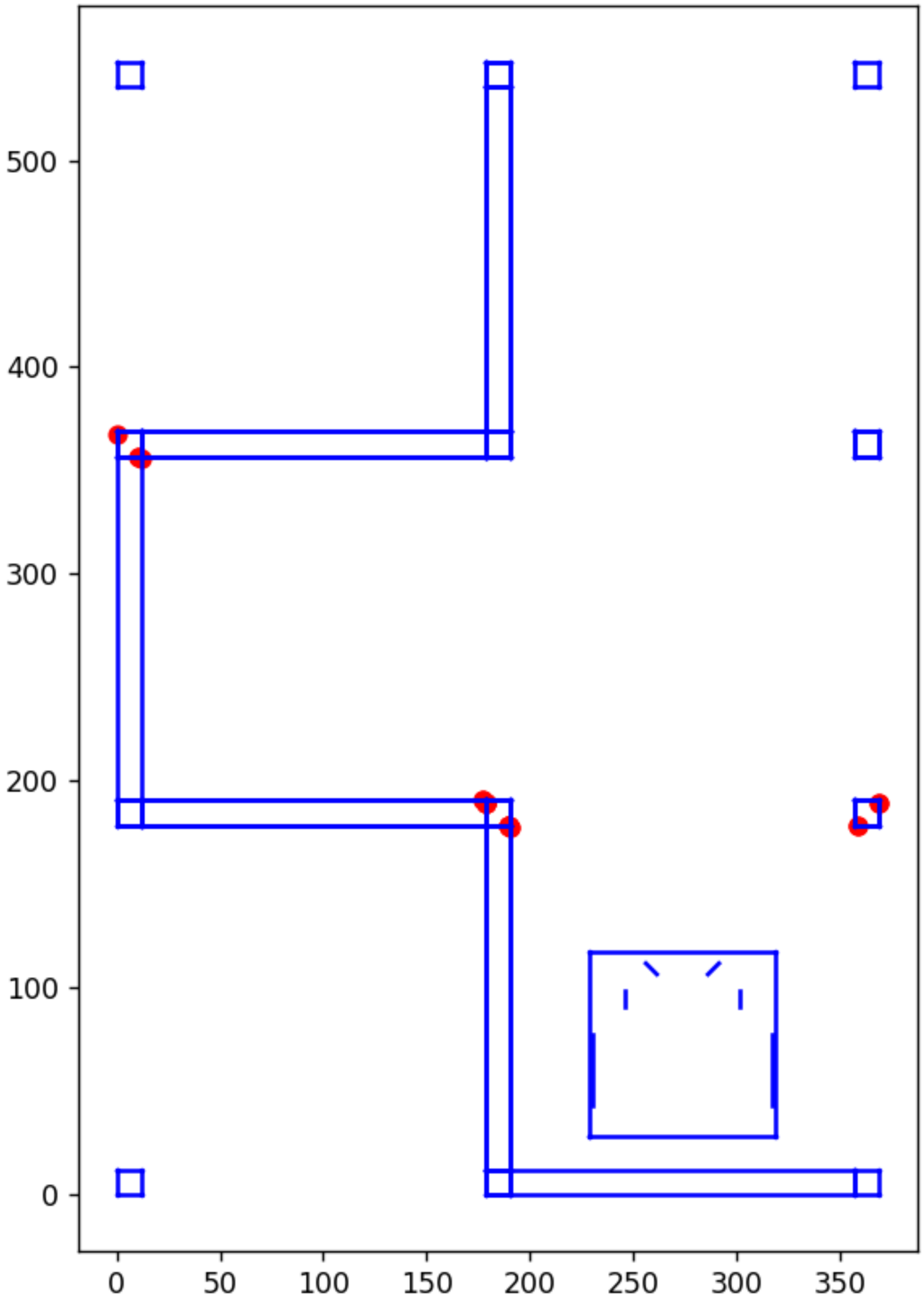
- Almost there
- Something weird's going on w/ reading walls that should be invisible



```
print(w4_2)
s4_distance_list.extend([
    mouse.get_right_forward_sensor_distance(w4_1, w4_2),
    mouse.get_right_forward_sensor_distance(w4_3, w4_2),
    mouse.get_right_forward_sensor_distance(w4_3, w4_4),
    mouse.get_right_forward_sensor_distance(w4_4, w4_1)
])
```

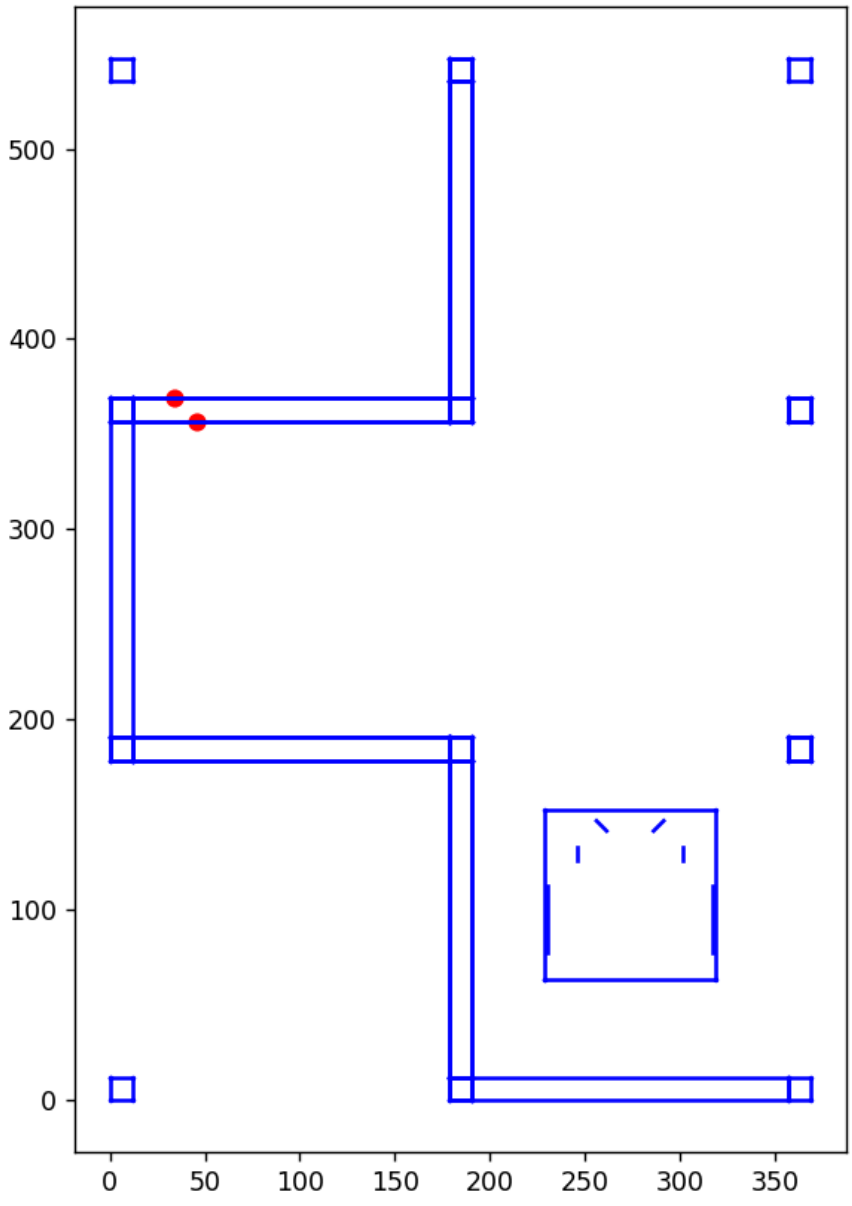
- Changing the order of the points makes a new intersection appear
- There's something going wrong w/ the intersection calculation

10/12/2024:


500
400
300
200
100
0
0
50
100
150
200
250
300
350

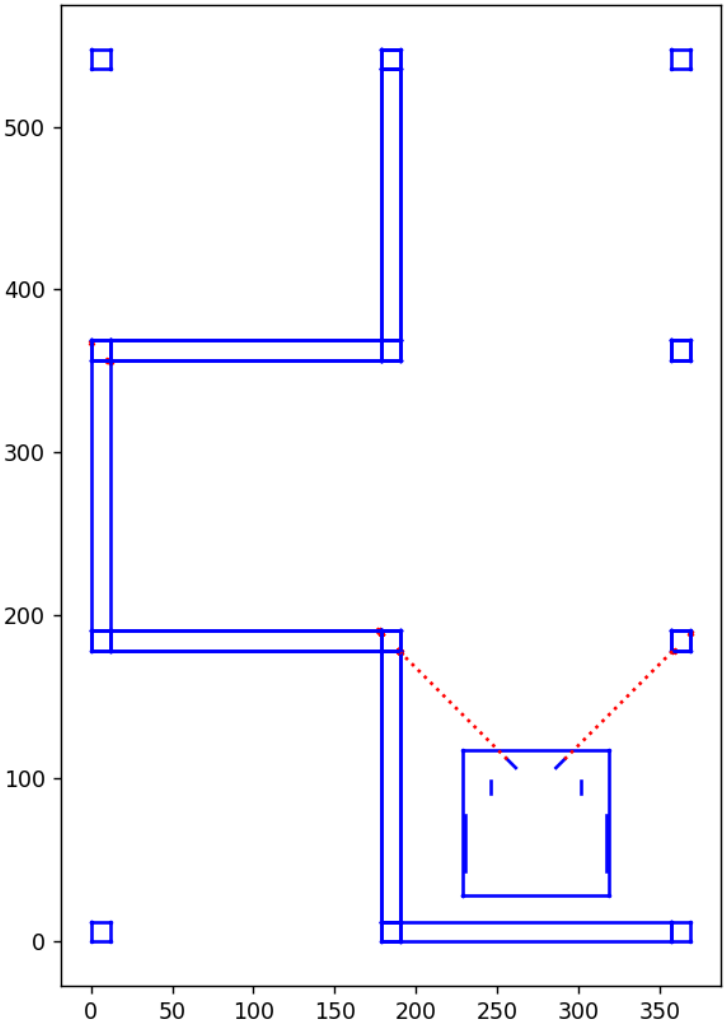
```
None, 92.6522015388733, 94.36339994934478, None
```

- Fantastic
- Thank you Python and numpy

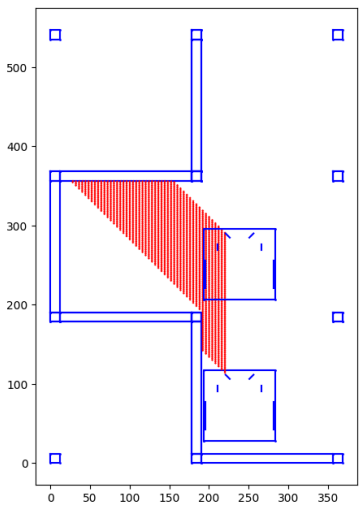


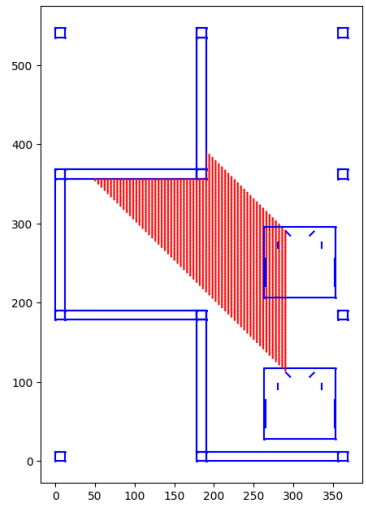

```
None, 297.21112226832963, None, None
```

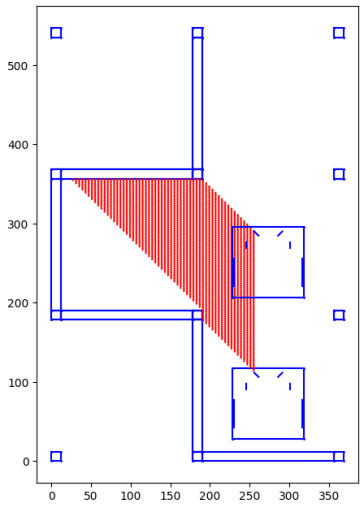
- Subarashiii

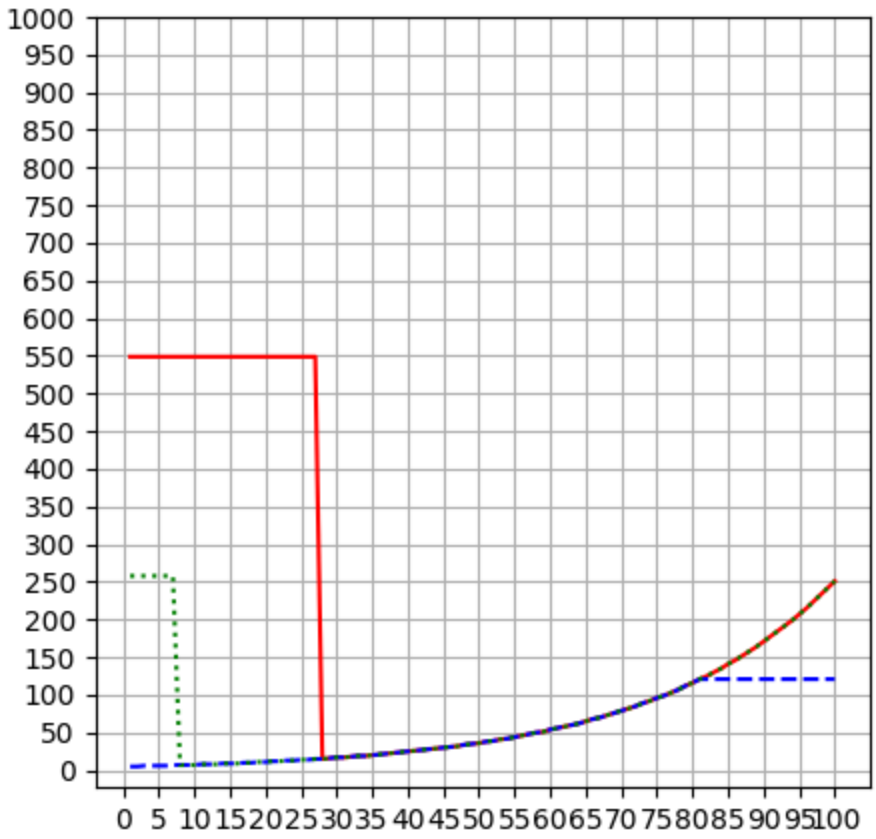


- Now it's just a matter of spamming it

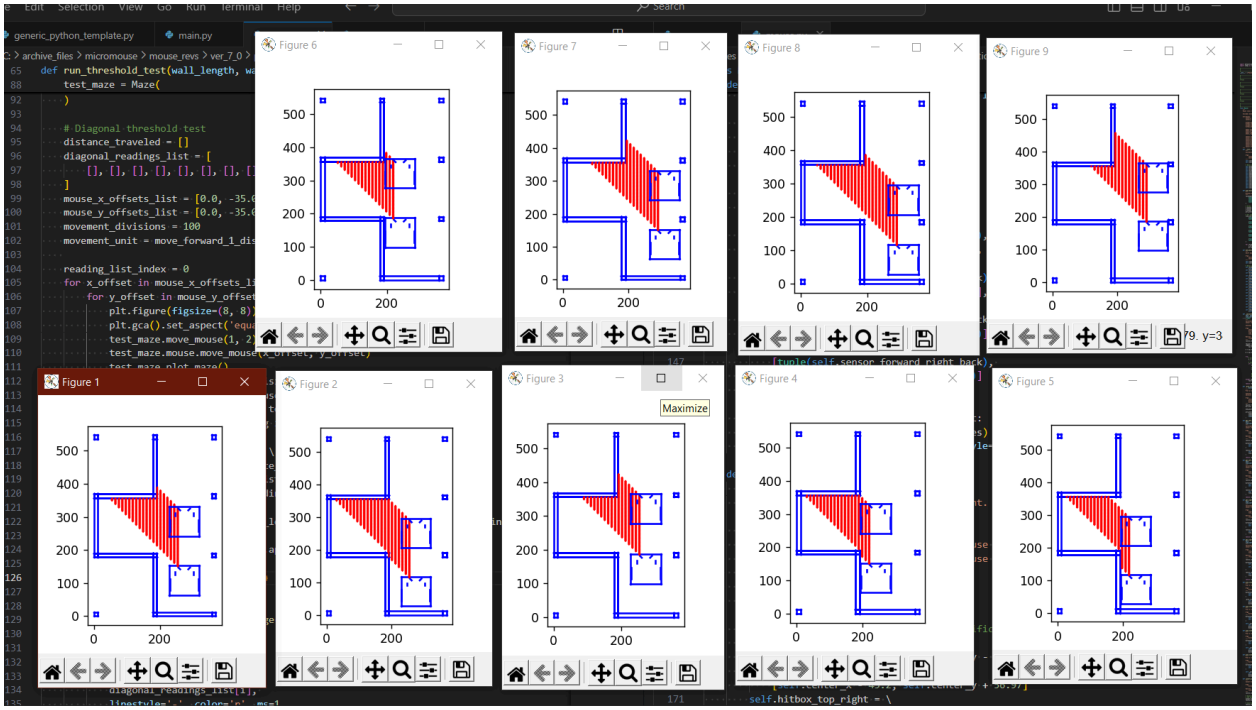




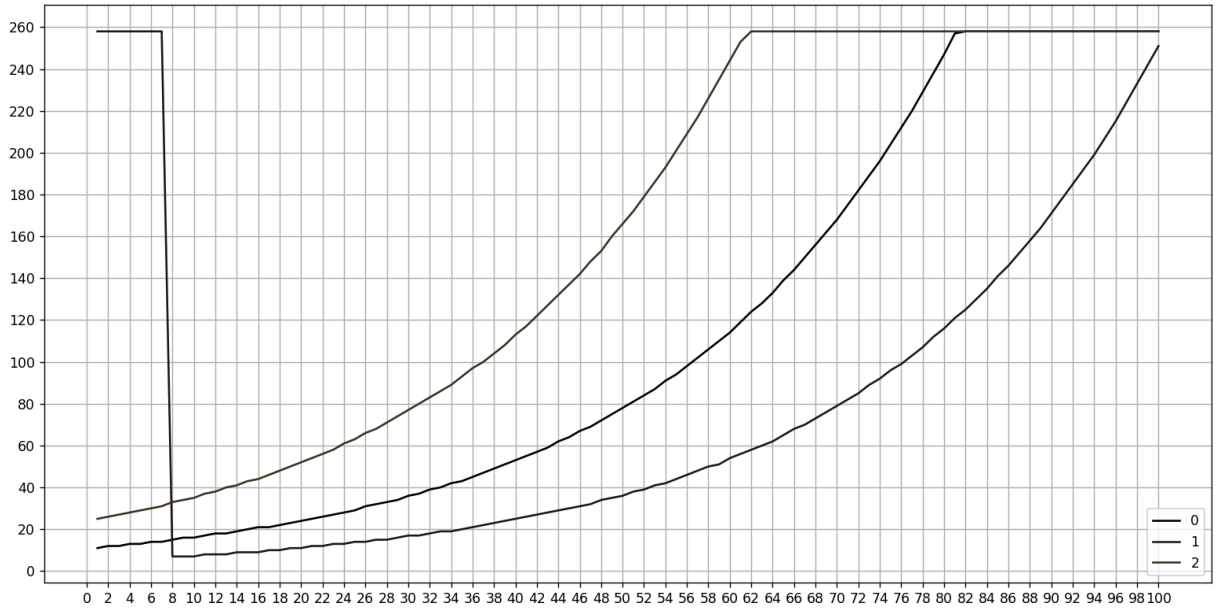




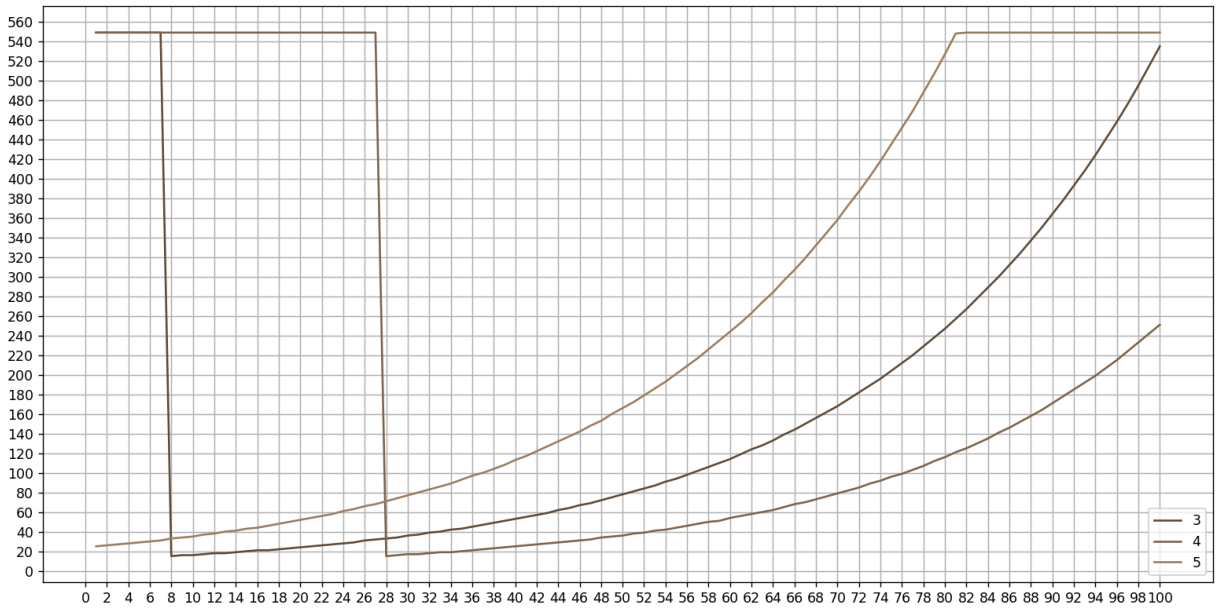
- Let’s add two vertical distance extremes too



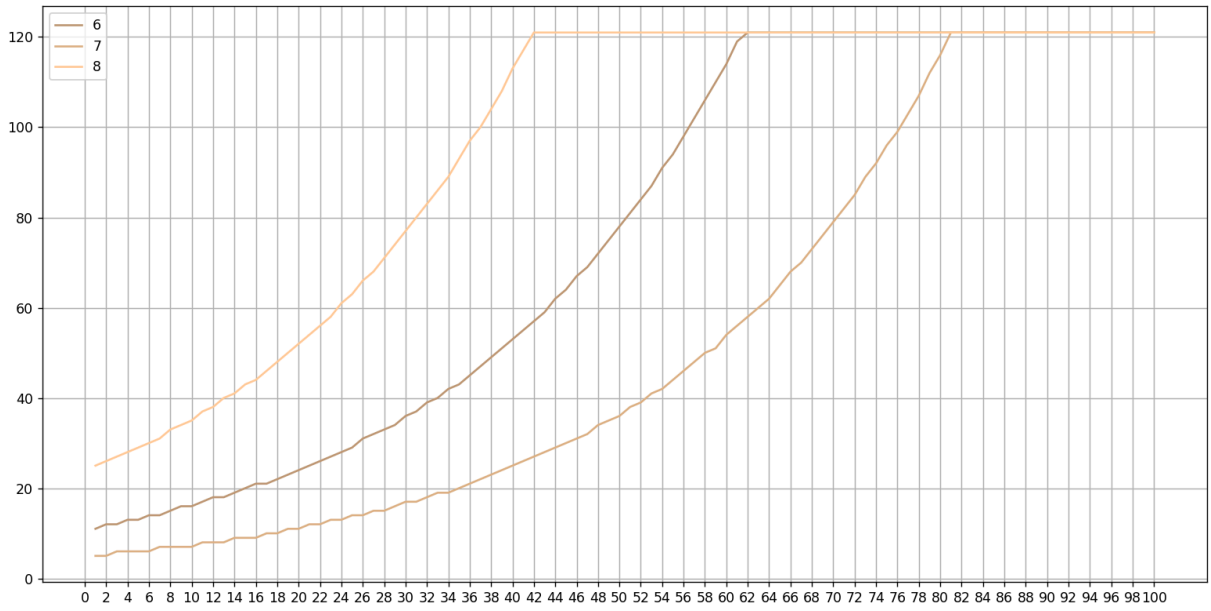
- Here we are



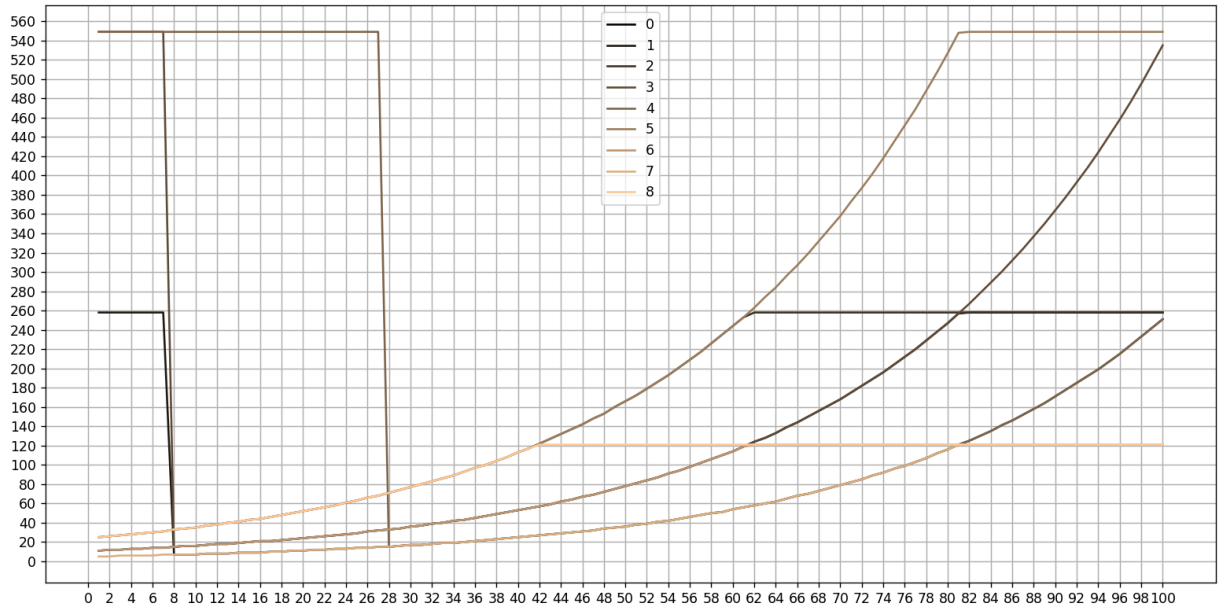
- Through the center
- Ideal scenario
- The left wall is absent from 10% -60% of move forward
- Gives us ADC readings from 40-250 or so

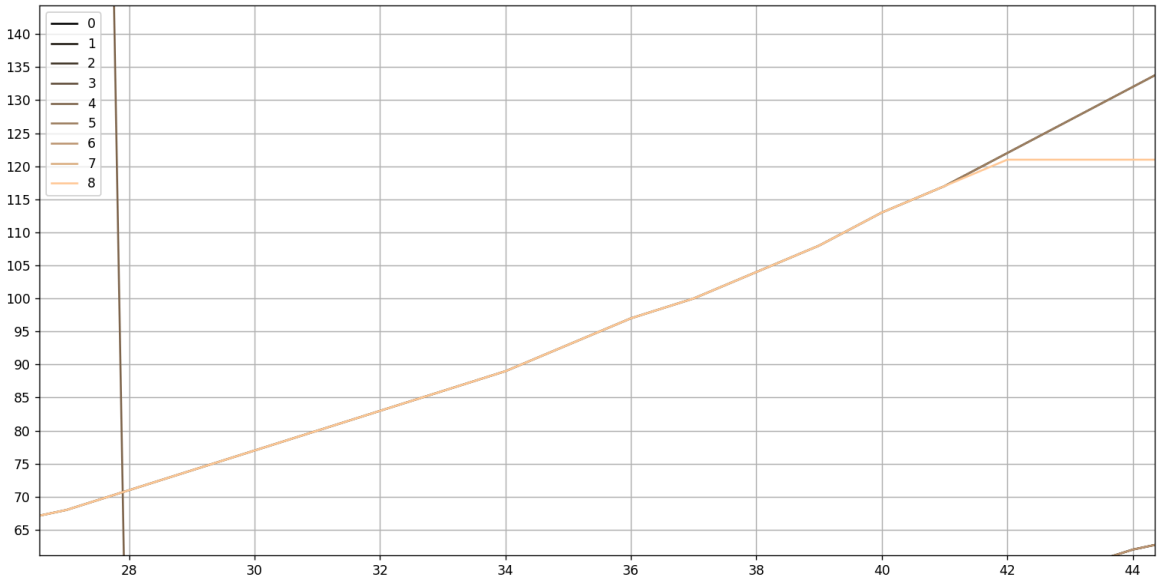


- Hugging left wall
- Gives us yay high ADC readings
- The left wall is absent from 30%-80% of move forward
- Gives us ADC readings from 80-540 or so



- Hugging right wall
- Gives us yay low ADC readings
- The left wall is absent from 0%-40% of move forward
- Gives us ADC readings from 30-120 or so

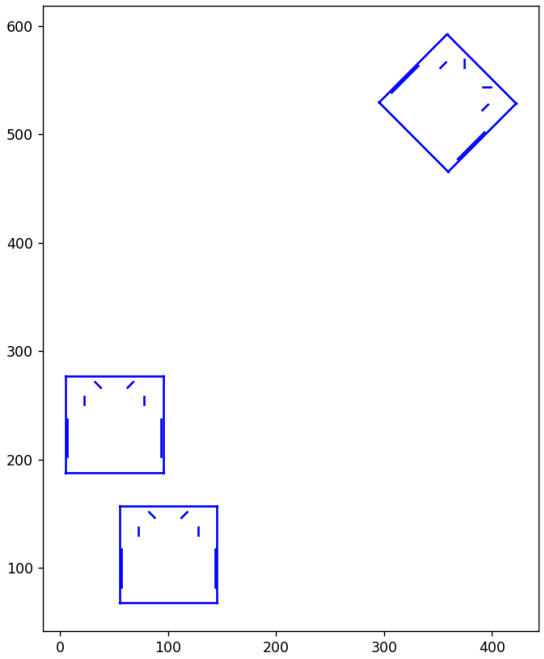




- Combined graph
- The sweet spot tells us to:
    - **Read between 30%-40% of the move forward**
    - **Check that readings are below 120** or so
- This will make sure that we see the opening no matter where the mouse is and how it's moving along the corridor
- 120 is pretty darn low
- ADC reading = ((((2.71272)(0.858585)^{x}) * (2.2 / (1+2.2))) / 1.8V) * 1024

- This means:
- $\text{Log}_{(0.858585)}$ ((((Reading / 1024) * 1.8) * (3.2/2.2)) / (2.71272)) = distance
- Desmos: HERE
- 120 correlates to 14cm away from the sensor
- 250 correlates to 9.4cm away from the sensor
- 540 correlates to 4.4cm away from the sensor

- Eh well the SHARP sensor is rated 2-15cm so guess it's fine
- Let's put together a test to see how much of an angle the mouse can be until the new scheme to check left/right openings starts to fail



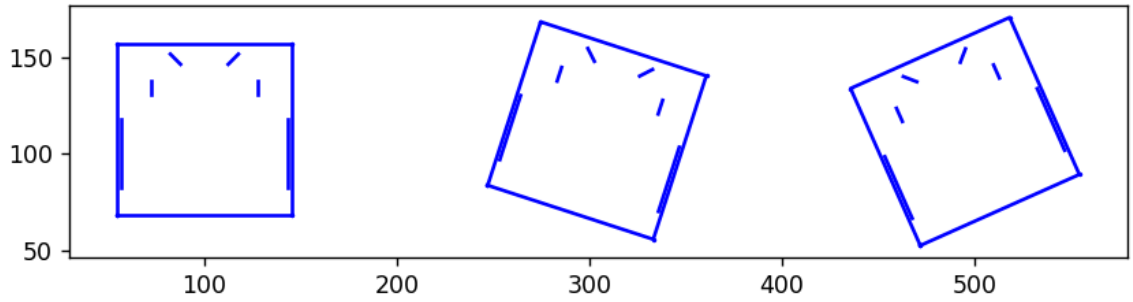
- Aight rotations are in

- We'll put together our jiggle test tomorrow maybe

10/13/2024:

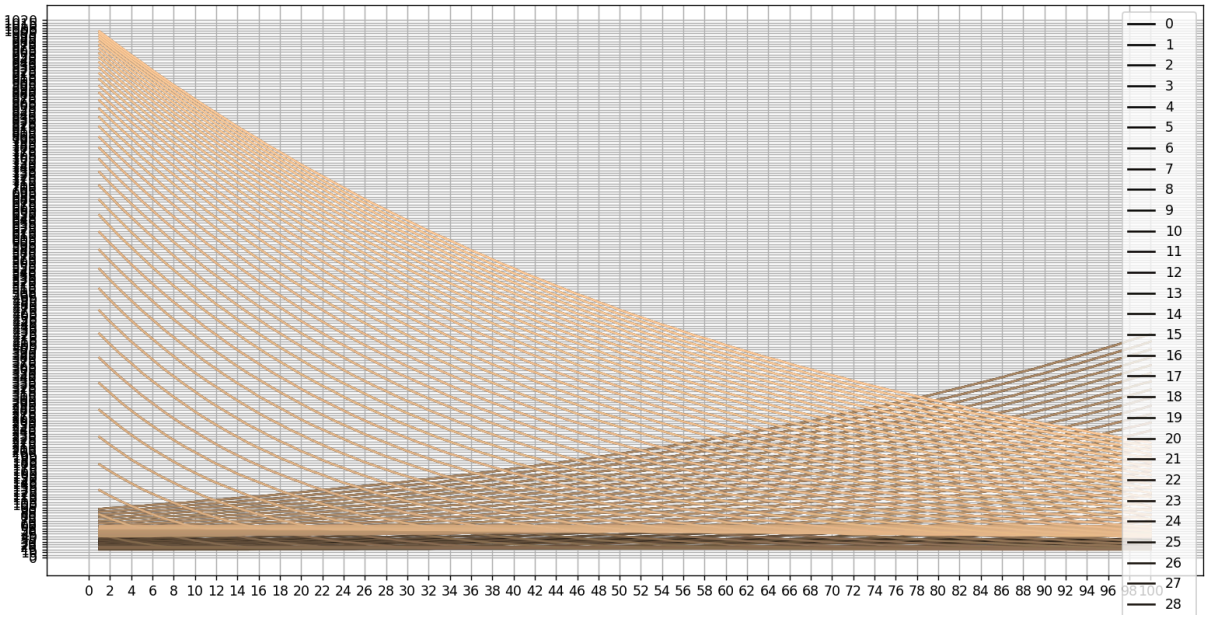
```
3] - Summary of passed cases [x_offset, y_offset, angle]:
3] - x: 0.0, y: 0.0- 31 angles passed, min angle: -21, max angle: 9
3] - x: 0.0, y: -35.0- 23 angles passed, min angle: -24, max angle: -2
3] - x: 0.0, y: 35.0- 43 angles passed, min angle: -18, max angle: 24
3] - x: -35.0, y: 0.0- 26 angles passed, min angle: -29, max angle: -4
3] - x: -35.0, y: -35.0- 17 angles passed, min angle: -30, max angle: -14
3] - x: -35.0, y: 35.0- 41 angles passed, min angle: -26, max angle: 14
3] - x: 35.0, y: 0.0- 34 angles passed, min angle: -15, max angle: 18
3] - x: 35.0, y: -35.0- 27 angles passed, min angle: -18, max angle: 8
3] - x: 35.0, y: 35.0- 43 angles passed, min angle: -11, max angle: 31
3] - [0.0, 0.0, -21]
```

- Here we are
- The most balanced and optimal position for the mouse if we want to use the 30%-40% and 120 threshold checking scheme is **the middle of the corridor towards the top/front of the square**
- It can tolerate up to -18 degrees and 24 degrees

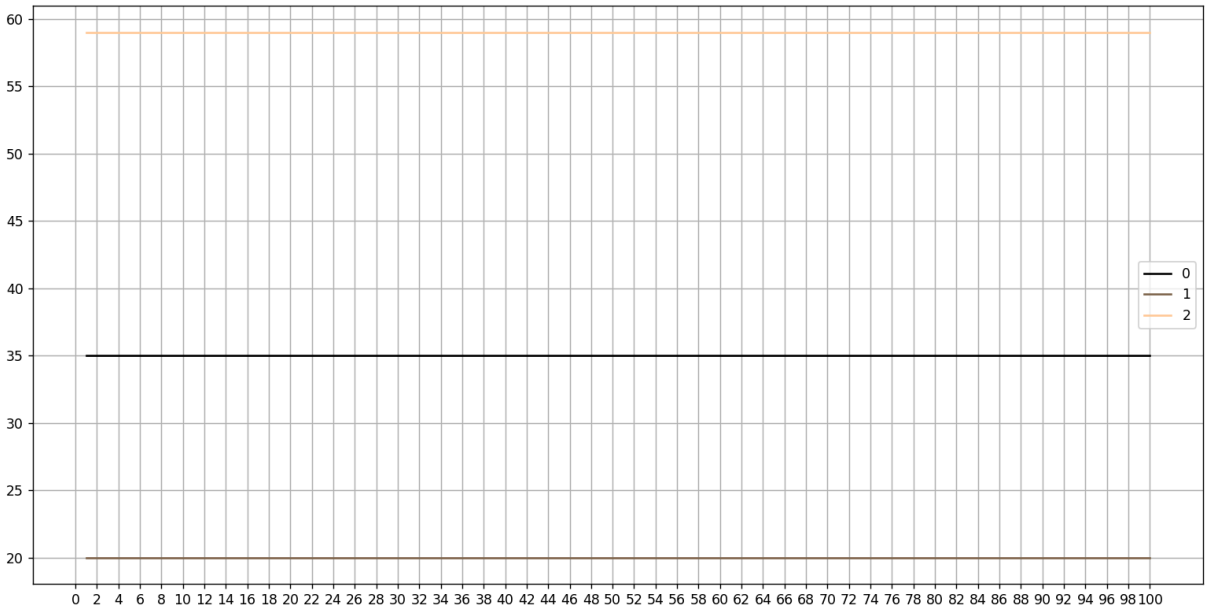


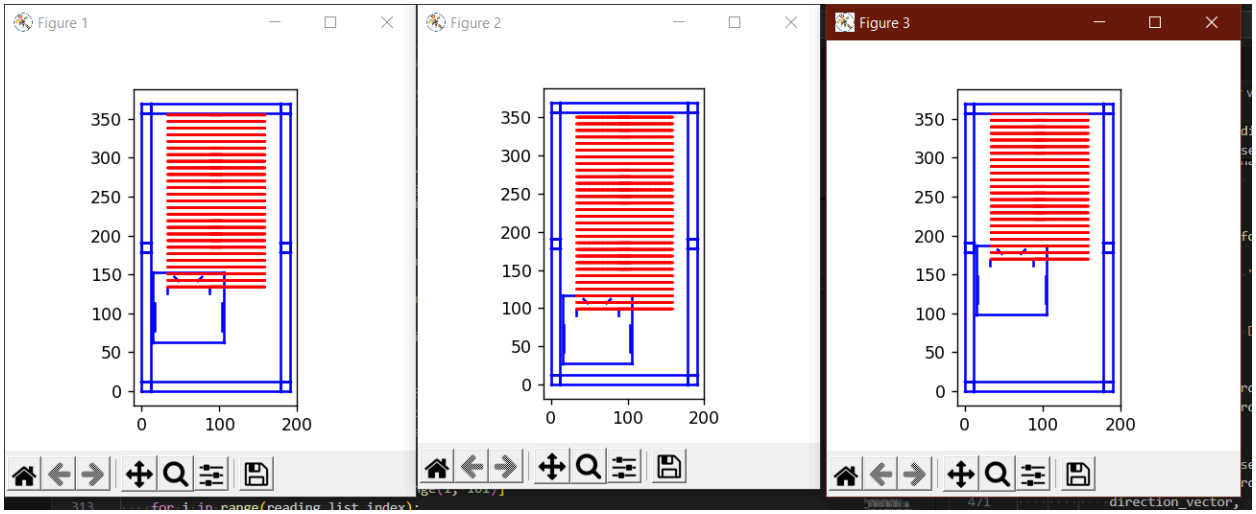
- This is what -18 degrees and 24 degrees looks like
- That's decent jiggle tolerance

- Let's move on to forward sensor threshold tests

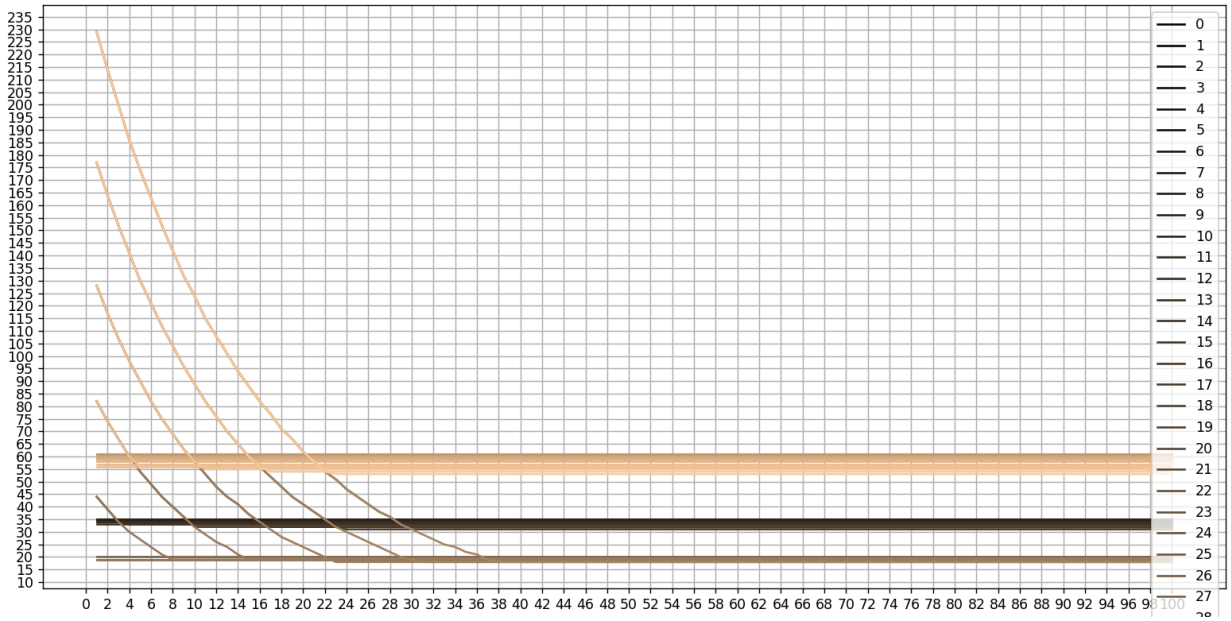


- Goodness gracious
- -44 degrees and 44 degrees is too much





- These are readings without angles



- Here's -10 degrees to 10 degrees
- The curves arise from hugging left/right walls and starting at it due to being at an angle
- Otherwise we can probably say that **anything greater than 70 indicates a wall in front**

walldetection_mci.h | movement_mci.c | movement_mci.h

```
52  #define MCI_READING_MM_TO_RAW(x) (896 * pow(0.98, x))
53
54  /* sensor threshold hard coded raw values- prepared since math is extremely slow */
55  /* values found by running mci_PrintWallPresence() w/ thresholds printed after */
56  /* changing wall length MCI_MAZE_WALL_LENGTH_MM */
57  #define MCI_FRONT_SENSOR_READING_THRESHOLD_RAW_165MM_WALLS_HARD_CODED    (174)
58  #define MCI_LEFT_SENSOR_READING_THRESHOLD_RAW_165MM_WALLS_HARD_CODED     (63)
59  #define MCI_RIGHT_SENSOR_READING_THRESHOLD_RAW_165MM_WALLS_HARD_CODED    (63)
60
61  #define MCI_FRONT_SENSOR_READING_THRESHOLD_RAW_180MM_WALLS_HARD_CODED    (128)
62  #define MCI_LEFT_SENSOR_READING_THRESHOLD_RAW_180MM_WALLS_HARD_CODED     (101)
63  #define MCI_RIGHT_SENSOR_READING_THRESHOLD_RAW_180MM_WALLS_HARD_CODED    (101)
64
65  /* hard coded values for 30mm to detect front wall too close */
66  #define MCI_FRONT_WALL_TOO_CLOSE_THRESHOLD_RAW_30MM_HARD_CODED           (160)
67
68  /* front sensor threshold raw value */
69  #define MCI_FRONT_SENSOR_READING_THRESHOLD_RAW \
70      MCI_FRONT_SENSOR_READING_THRESHOLD_RAW_180MM_WALLS_HARD_CODED
71      //MCI_READING_MM_TO_RAW(MCI_FRONT_SENSOR_READING_THRESHOLD_MM)
```

74 %

- These are thresholds we were working w/ before…
- That's a pretty high threshold on the front reading
- Mouse could've been seeing front openings where there wasn't?
- left/right thresholds were trickier because we had to get both timing and thresholds right

- We'll put together a front wall detection test provided a threshold parameter next

```
FAILED: y: 35.0, ang: 43 average_reading: 324
FAILED: y: 35.0, ang: 44 average_reading: 333
Summary of passed cases [y_offset, angle]:
y: 0.0- 35 angles passed, min angle: -17, max angle: 17
y: -35.0- 37 angles passed, min angle: -18, max angle: 18
y: 35.0- 31 angles passed, min angle: -15, max angle: 15
[0.0, -17]
```

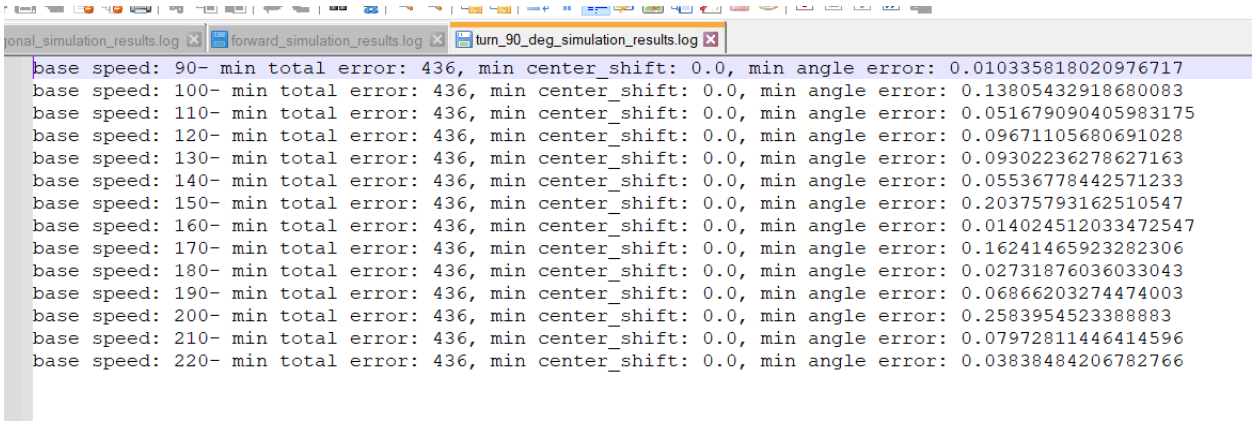
- Front wall check tolerates around -15 degrees and 15 degrees if it's up close towards the front of the square

- We'll continue tomorrow and put together a simulation to find optimal PD constants for turning, and then for moving forward

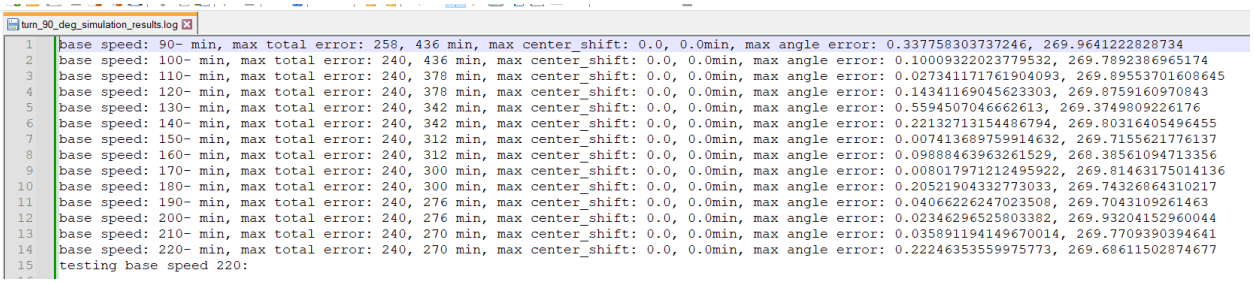
10/14/2024:

- Left off writing method to simulate PID turning

- Encoder counts per turn can be a hardcoded attribute on the mouse class, but we need a method to calculate the encoder counts per square to square after taking parameters for wall and post length
- Let's make independent motor speed offsets the test we run provided chosen PD constants and base motor speed



- Interesting



- Bad code, updated

turn_90_deg_simulation_results.log

```
[240.0, 0.0, 110.53721018083849, 6.4, 0.2]
[240.0, 0.0, 1.0016177666085184, 6.4, 0.30000000000000004]
[240.0, 0.0, 251.4660253523786, 6.4, 0.4]
[240.0, 0.0, 141.93043293814867, 6.4, 0.5]
[240.0, 0.0, 32.39484052391874, 6.4, 0.6000000000000001]

top 50 lowest angle_error:
[240.0, 0.0, 0.09632323232162321, 0.8, 0.1]
[240.0, 0.0, 0.1608499099443037, 4.9, 0.7000000000000001]
[240.0, 0.0, 0.3202432938944355, 6.2, 0.2]
[240.0, 0.0, 0.3611311788195479, 6.0, 0.1]
[240.0, 0.0, 0.8834685198725936, 3.1, 0.8]
[240.0, 0.0, 1.0016177666085184, 6.4, 0.30000000000000004]
[240.0, 0.0, 1.042505651531414, 5.800000000000001, 0.0]
[240.0, 0.0, 1.4825397039715114, 3.9000000000000004, 0.30000000000000004]
[240.0, 0.0, 1.6829922393199155, 6.6000000000000005, 0.4]
[240.0, 0.0, 1.7238801242431094, 5.5, 1.0]
[240.0, 0.0, 2.104415706589876, 5.1000000000000005, 0.8]
[240.0, 0.0, 2.3595234903571054, 2.2, 0.7000000000000001]
[240.0, 0.0, 2.36436671203181, 6.800000000000001, 0.5]
[240.0, 0.0, 2.4052545969564676, 5.300000000000001, 0.9]
[240.0, 0.0, 2.6372511068475717, 3.3000000000000003, 0.5]
[240.0, 0.0, 2.7709403395830066, 10.0, 0.8]
[240.0, 0.0, 2.783183582647098, 2.5, 0.5]
[240.0, 0.0, 2.8068360946379585, 4.6000000000000005, 0.4]
[264.0, 0.0, 2.908967633567144, 0.5, 0.30000000000000004]
[240.0, 0.0, 3.0457411847435907, 7.0, 0.6000000000000001]
[240.0, 0.0, 3.452314812294432, 9.8, 0.7000000000000001]
[240.0, 0.0, 3.508715244569103, 1.7000000000000002, 1.0]
[240.0, 0.0, 3.7271156574553572, 7.2, 0.7000000000000001]
[240.0, 0.0, 4.133689285005843, 9.600000000000001, 0.6000000000000001]
[240.0, 0.0, 4.408490130166669, 7.4, 0.8]
[262.0, 0.0, 4.557164391431201, 0.30000000000000004, 0.8]
[240.0, 0.0, 4.575653507119654, 2.7, 0.8]
[240.0, 0.0, 4.641348052834019, 1.6, 0.9]
[240.0, 0.0, 4.8150637577178514, 9.4, 0.5]
[240.0, 0.0, 4.970682676911949, 1.0, 0.5]
[240.0, 0.0, 5.089864602878535, 7.6000000000000005, 0.9]
[240.0, 0.0, 5.496438230432759, 9.200000000000001, 0.4]
[240.0, 0.0, 5.771239075589548, 7.800000000000001, 1.0]
[240.0, 0.0, 6.071230272526847, 1.2000000000000002, 0.6000000000000001]
[240.0, 0.0, 6.072754392942485, 3.0, 0.0]
```

- If we go w/ a base speed of 220 these are the optimal P,D constants
- Goodness gracious it's not linear at all

turn_90_deg_simulation_results.log

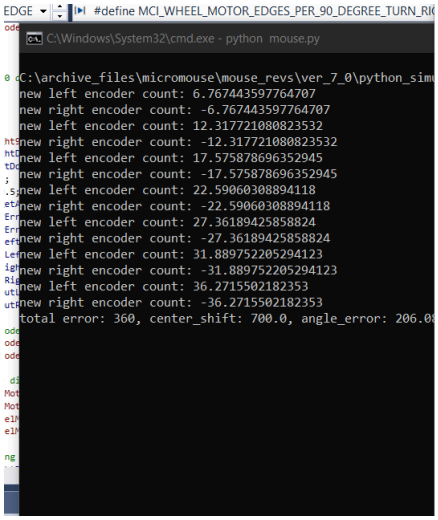
```
base speed: 90- min, max total error: 258, 436 min, max center_shift: 0.0, 0.0min, max angle error: 0.5461340613328645, 43.43611758255446
base speed: 100- min, max total error: 240, 436 min, max center_shift: 0.0, 0.0min, max angle error: 0.10867484738804478, 42.45390421947303
base speed: 110- min, max total error: 240, 378 min, max center_shift: 0.0, 0.0min, max angle error: 0.10867484738810163, 44.09092649127541
base speed: 120- min, max total error: 240, 378 min, max center_shift: 0.0, 0.0min, max angle error: 0.10867484738808741, 42.45390421947309
base speed: 130- min, max total error: 240, 342 min, max center_shift: 0.0, 0.0min, max angle error: 0.10867484738808741, 42.45390421947303
base speed: 140- min, max total error: 240, 342 min, max center_shift: 0.0, 0.0min, max angle error: 0.10867484738805899, 45.07313985435681
base speed: 150- min, max total error: 240, 312 min, max center_shift: 0.0, 0.0min, max angle error: 0.10867484738808741, 44.745735399996335
base speed: 160- min, max total error: 240, 312 min, max center_shift: 0.0, 0.0min, max angle error: 0.10867484738810163, 43.10871312819401
base speed: 170- min, max total error: 240, 300 min, max center_shift: 0.0, 0.0min, max angle error: 0.2187296069723459, 46.382757671798686
base speed: 180- min, max total error: 240, 300 min, max center_shift: 0.0, 0.0min, max angle error: 0.10867484738810163, 42.45390421947303
base speed: 190- min, max total error: 240, 276 min, max center_shift: 0.0, 0.0min, max angle error: 0.10867484738811584, 46.05535321743824
base speed: 200- min, max total error: 240, 276 min, max center_shift: 0.0, 0.0min, max angle error: 0.5461340613328076, 44.41833094563589
base speed: 210- min, max total error: 240, 270 min, max center_shift: 0.0, 0.0min, max angle error: 5.019741662795141, 42.781308673833536
base speed: 220- min, max total error: 240, 270 min, max center_shift: 0.0, 0.0min, max angle error: 17.788515382853532, 41.47169085639166
testing base speed 220:
```

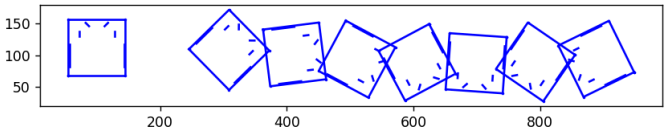
- Bad code again, here we are
- That makes more sense
- 220 is too high of a base speed
- 190 is pretty solid

turn_90_deg_simulation_results.log

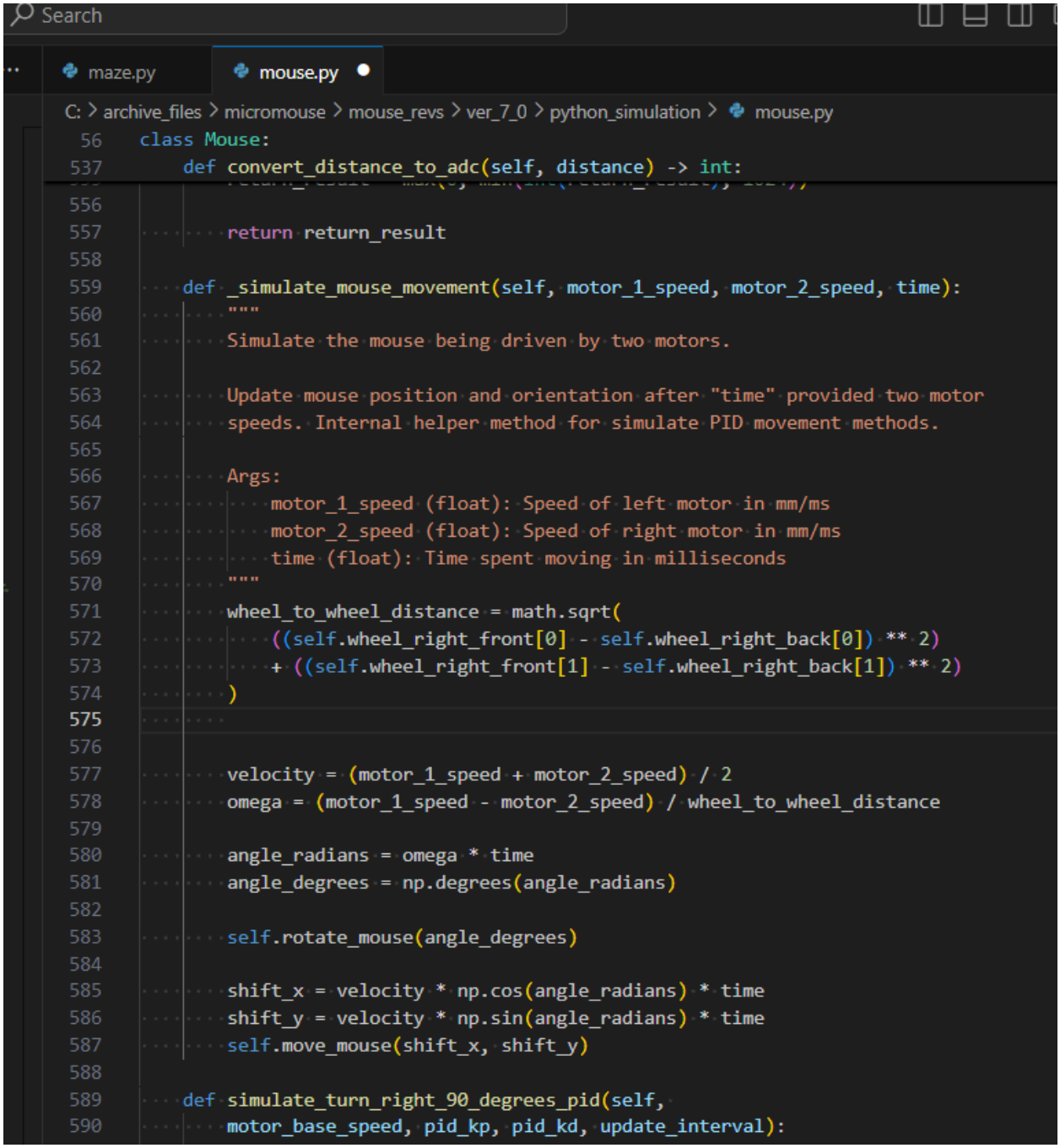
```
[262.0, 0.0, 27.720703773251856, 6.4, 0.5]
[262.0, 0.0, 28.702917136333255, 6.4, 0.6000000000000001]

top 50 lowest angle_error:
[314.0, 0.0, 0.5461340613328645, 3.1, 0.30000000000000004]
[314.0, 0.0, 0.5461340613328645, 3.1, 0.1]
[350.0, 0.0, 0.7634837561090535, 1.8, 0.4]
[314.0, 0.0, 0.8735385156932693, 3.1, 0.4]
[314.0, 0.0, 0.873538515693312, 3.1, 0.2]
[390.0, 0.0, 0.8735385156933404, 0.8, 0.2]
[436.0, 0.0, 1.2009429700537453, 0.1, 0.2]
[390.0, 0.0, 1.200942970053788, 0.8, 0.0]
[306.0, 0.0, 1.2009429700538021, 3.2, 0.8]
[436.0, 0.0, 1.2009429700538021, 0.1, 0.0]
[436.0, 0.0, 1.2009429700538021, 0.1, 0.1]
[342.0, 0.0, 1.4182926648299201, 1.9000000000000001, 0.9]
[306.0, 0.0, 1.4182926648299912, 3.3000000000000003, 0.8]
[384.0, 0.0, 1.5283474244142496, 0.8, 0.7000000000000001]
[342.0, 0.0, 1.5283474244142923, 1.8, 1.0]
[384.0, 0.0, 1.8557518787746972, 0.8, 0.5]
[306.0, 0.0, 1.8557518787746972, 3.2, 0.9]
[306.0, 0.0, 1.8557518787747114, 3.2, 0.7000000000000001]
[342.0, 0.0, 1.8557518787747114, 1.8, 0.7000000000000001]
[384.0, 0.0, 1.8557518787747114, 0.8, 0.6000000000000001]
[384.0, 0.0, 1.8557518787747114, 0.8, 0.30000000000000004]
[384.0, 0.0, 2.0731015735509004, 0.9, 0.5]
[384.0, 0.0, 2.073101573550929, 0.9, 0.7000000000000001]
[384.0, 0.0, 2.073101573550929, 0.9, 0.6000000000000001]
[342.0, 0.0, 2.183156333135173, 1.8, 0.9]
[384.0, 0.0, 2.1831563331351873, 0.8, 0.8]
[384.0, 0.0, 2.1831563331351873, 0.8, 0.4]
[380.0, 0.0, 2.1831563331352015, 0.8, 0.9]
[342.0, 0.0, 2.1831563331352157, 1.8, 0.8]
[390.0, 0.0, 2.18315633313523, 0.8, 0.1]
[314.0, 0.0, 2.400506027911433, 3.2, 0.30000000000000004]
[380.0, 0.0, 2.510560787495649, 0.8, 1.0]
[390.0, 0.0, 2.7279104822718665, 0.9, 0.0]
[314.0, 0.0, 2.837965241856139, 3.0, 0.0]
[350.0, 0.0, 2.8379652418561676, 1.7000000000000002, 0.30000000000000004]
[390.0, 0.0, 3.0553149366323, 0.9, 0.1]
```

- If we're going w/ 90 base speed supposedly 3.1 and 0.3 is the best



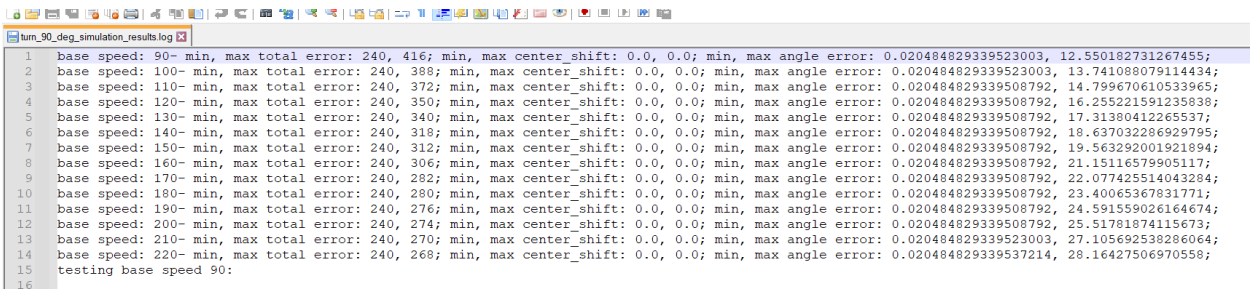


- Mm
- That’s not right



- This is our problem
- Forgot to translate PWM -> mm/ms
- Need to do that next next weekend
- And then hopefully it'll work w/ the simulate_get_encoder_count properly

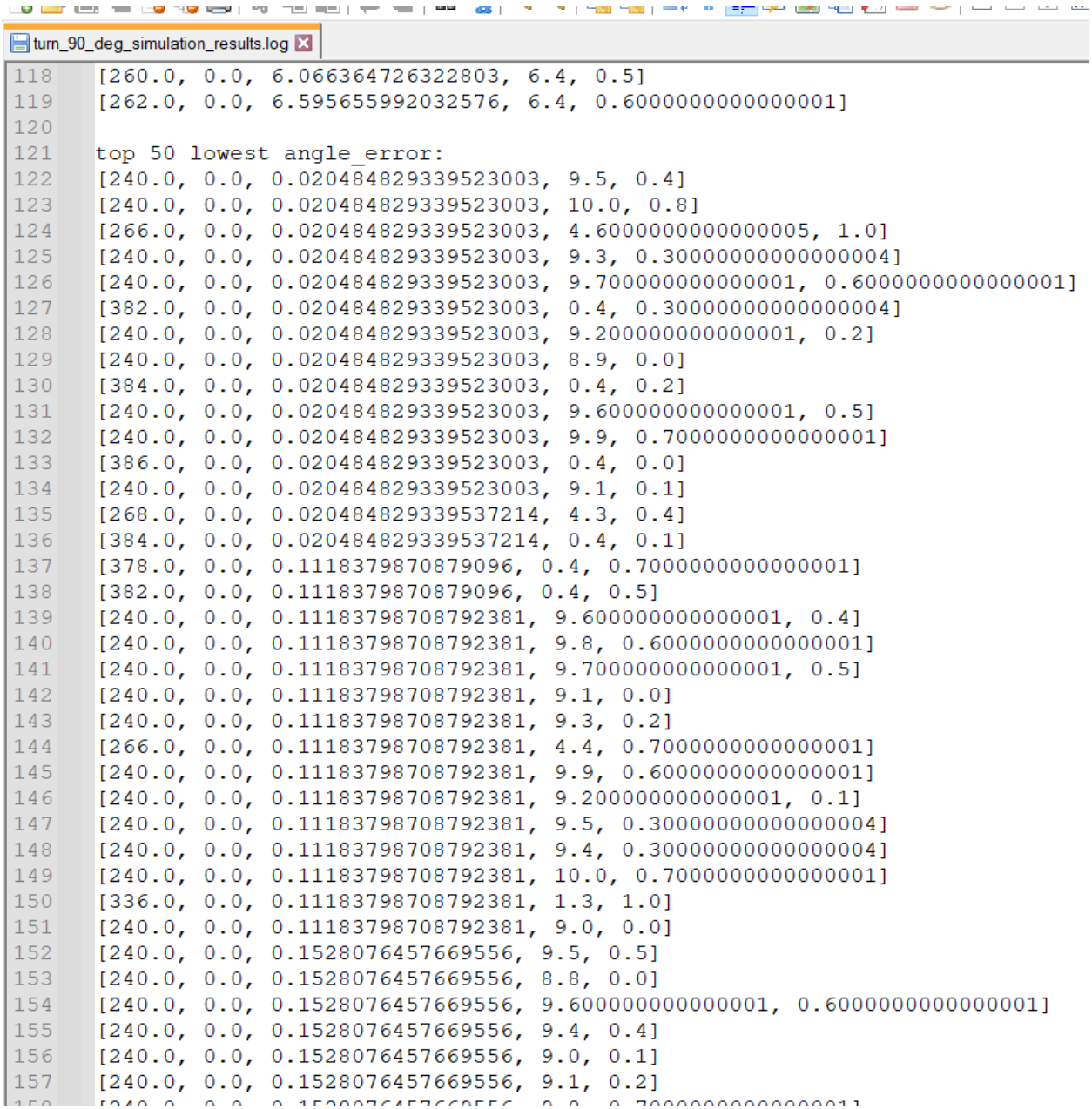
- Nah screw that let's do it now



turn_90_deg_simulation_results.log

```
base speed: 90- min, max total error: 240, 416; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339523003, 12.550182731267455;
base speed: 100- min, max total error: 240, 388; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339523003, 13.741088079114434;
base speed: 110- min, max total error: 240, 372; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 14.799670610533965;
base speed: 120- min, max total error: 240, 350; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 16.255221591235838;
base speed: 130- min, max total error: 240, 340; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 17.31380412265537;
base speed: 140- min, max total error: 240, 318; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 18.637032286929795;
base speed: 150- min, max total error: 240, 312; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 19.563292001921894;
base speed: 160- min, max total error: 240, 306; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 21.15116579905117;
base speed: 170- min, max total error: 240, 282; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 22.077425514043284;
base speed: 180- min, max total error: 240, 280; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 23.40065367831771;
base speed: 190- min, max total error: 240, 276; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 24.591559026164674;
base speed: 200- min, max total error: 240, 274; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339508792, 25.51781874115673;
base speed: 210- min, max total error: 240, 270; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339523003, 27.105692538286064;
base speed: 220- min, max total error: 240, 268; min, max center_shift: 0.0, 0.0; min, max angle error: 0.020848429339537214, 28.16427506970558;
testing base speed 90:
```

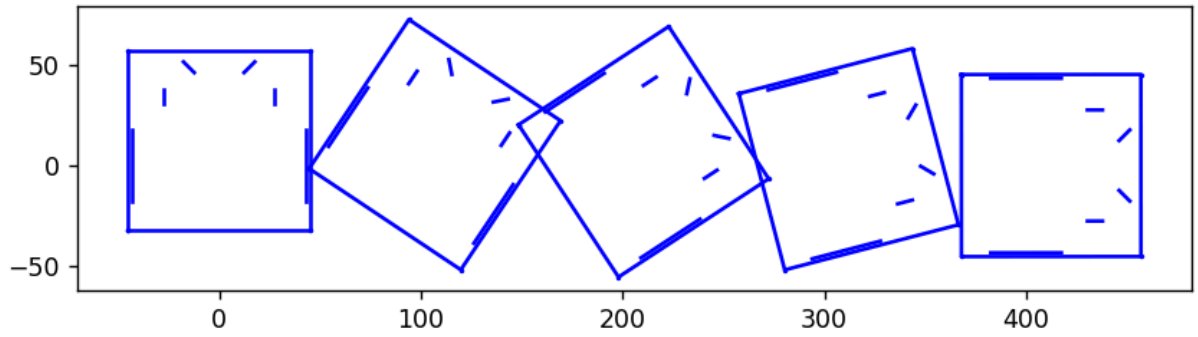
- Aight bet here we are
- Goodness



turn_90_deg_simulation_results.log

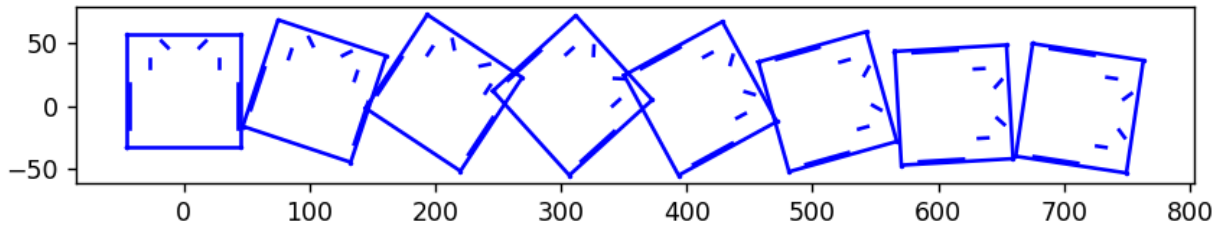
```
[260.0, 0.0, 6.066364726322803, 6.4, 0.5]
[262.0, 0.0, 6.595655992032576, 6.4, 0.6000000000000001]

top 50 lowest angle_error:
[240.0, 0.0, 0.020848429339523003, 9.5, 0.4]
[240.0, 0.0, 0.020848429339523003, 10.0, 0.8]
[266.0, 0.0, 0.020848429339523003, 4.6000000000000005, 1.0]
[240.0, 0.0, 0.020848429339523003, 9.3, 0.30000000000000004]
[240.0, 0.0, 0.020848429339523003, 9.700000000000001, 0.6000000000000001]
[382.0, 0.0, 0.020848429339523003, 0.4, 0.30000000000000004]
[240.0, 0.0, 0.020848429339523003, 9.200000000000001, 0.2]
[240.0, 0.0, 0.020848429339523003, 8.9, 0.0]
[384.0, 0.0, 0.020848429339523003, 0.4, 0.2]
[240.0, 0.0, 0.020848429339523003, 9.600000000000001, 0.5]
[240.0, 0.0, 0.020848429339523003, 9.9, 0.7000000000000001]
[386.0, 0.0, 0.020848429339523003, 0.4, 0.0]
[240.0, 0.0, 0.020848429339523003, 9.1, 0.1]
[268.0, 0.0, 0.020848429339537214, 4.3, 0.4]
[384.0, 0.0, 0.020848429339537214, 0.4, 0.1]
[378.0, 0.0, 0.1118379870879096, 0.4, 0.7000000000000001]
[382.0, 0.0, 0.1118379870879096, 0.4, 0.5]
[240.0, 0.0, 0.11183798708792381, 9.600000000000001, 0.4]
[240.0, 0.0, 0.11183798708792381, 9.8, 0.6000000000000001]
[240.0, 0.0, 0.11183798708792381, 9.700000000000001, 0.5]
[240.0, 0.0, 0.11183798708792381, 9.1, 0.0]
[240.0, 0.0, 0.11183798708792381, 9.3, 0.2]
[266.0, 0.0, 0.11183798708792381, 4.4, 0.7000000000000001]
[240.0, 0.0, 0.11183798708792381, 9.9, 0.6000000000000001]
[240.0, 0.0, 0.11183798708792381, 9.200000000000001, 0.1]
[240.0, 0.0, 0.11183798708792381, 9.5, 0.30000000000000004]
[240.0, 0.0, 0.11183798708792381, 9.4, 0.30000000000000004]
[240.0, 0.0, 0.11183798708792381, 10.0, 0.7000000000000001]
[336.0, 0.0, 0.11183798708792381, 1.3, 1.0]
[240.0, 0.0, 0.11183798708792381, 9.0, 0.0]
[240.0, 0.0, 0.1528076457669556, 9.5, 0.5]
[240.0, 0.0, 0.1528076457669556, 8.8, 0.0]
[240.0, 0.0, 0.1528076457669556, 9.600000000000001, 0.6000000000000001]
[240.0, 0.0, 0.1528076457669556, 9.4, 0.4]
[240.0, 0.0, 0.1528076457669556, 9.0, 0.1]
[240.0, 0.0, 0.1528076457669556, 9.1, 0.2]
```

- If we're talking 90 base speed these constants are the best




```
C:\archive_files\micromouse\mouse_revs\ver_7_0\python_simulation>python mouse.py
total error: 266, center_shift: 400.0, angle_error: 0.020484829339565636
```

- This is what 4.6 and 1 looks like
- Very clean



- 1.0 and 0.5 looks like this

Next steps:

- ~~Finish building maze - **Ryuichi**~~
    - ~~Finish walls and posts~~
- Build new mouse 7 and test - **Ryuichi**
    - Clean firmware and create new Git repository
    - Create documentation and guides
        - Fusion 360 guide
        - Micromouse startup guide
        - Bare minimum Git guide (conventional Git commits too)
        - Semantic versioning, Doxygen, source code formatting guide
        - Block diagram of firmware

- Bluetooth board
    - Test w/ development board - **Chris F**
        - Does the boot pushbutton need to be pressed to upload code or run code that's been uploaded?
        - Can the ESP32 be programmed w/ USB easily?
    - ~~Design board and circuit- **Ryuichi**~~
        - ~~Right angle TH pins to get UART and 3.3V from mouse~~
        - ~~Other parts as needed using module datasheet and Sparkfun dev board schematic as references~~
- Python simulation - ~~**Chris L**~~ **Ryuichi**
    - (reassigned to match Chris joining December after school)
    - After building mouse get numbers so Chris can put together simulation
        - ~~Wheel positions~~
        - Mouse max and min speed as a function of wheel diameter
        - ~~Mouse simplified rectangular hitbox~~
        - ~~Mouse IR sensor locations on rectangular hitbox~~
        - Mouse IR sensor polling frequency
        - ~~IR sensor regression equation from datasheet to simulate sensor~~
        - ~~Mouse wall and post dimensions~~
    - Put together Sphinx template to use
    - Generating PID constants
        - One wall PD
        - Two wall PD
        - ~~Turning PD~~
        - Diagonals
    - ~~Generating IR sensor thresholds for wall detection~~
- Command line interface - **Andy**
    - Manual mouse movement
    - Mouse maze traversal progress and maze printing
    - PID constant assignment
    - IR sensor threshold assignment
- Maze algorithm updates- **Chris F**
    - Prepare maze algorithm for diagonal movements and time taken for each type of movement

4/19/2025:

- Back we are

- Aight overhaul time

- To-do items:
    - GitHub organization (Mouse Unit 07)
    - GitHub org project for Kanban practices
        - Flush out list of tasks
    - New split repositories
        - references
            - Flush out all files and documents into this archive
        - team-guides
            - Migrate existing guides to this repo
            - Generate new guides
            - New c template
        - infrastructure-sketches
            - UML sketch for team-wide structure
            - UML sketch for all packages and repos needed
        - hardware-motherboard
        - software-motherboard-at32uc3l0256-drivers
        - software-motherboard-hal
        - software-motherboard-
    - Build micromouse project w/ CMake
    - Create empty packages, post to JFrog Artifactory and test CMake/Conan integration

- Software development entails:
    - GitHub Organizations
        - GitHub projects (Kanban)
        - GitHub (repos)
        - Jenkins
        - JFrog Artifactory
        - Docker
            - Conan
                - CMake
                - CppUTest
                - CppCheck
                - gcov ~~lcov~~
                - Clang-format
            -