ROBOTIC CARICATURE ARTIST

TEAM 2

DYLAN HUANG, PETER KUIMELIS, & SOUMITHRI BALA

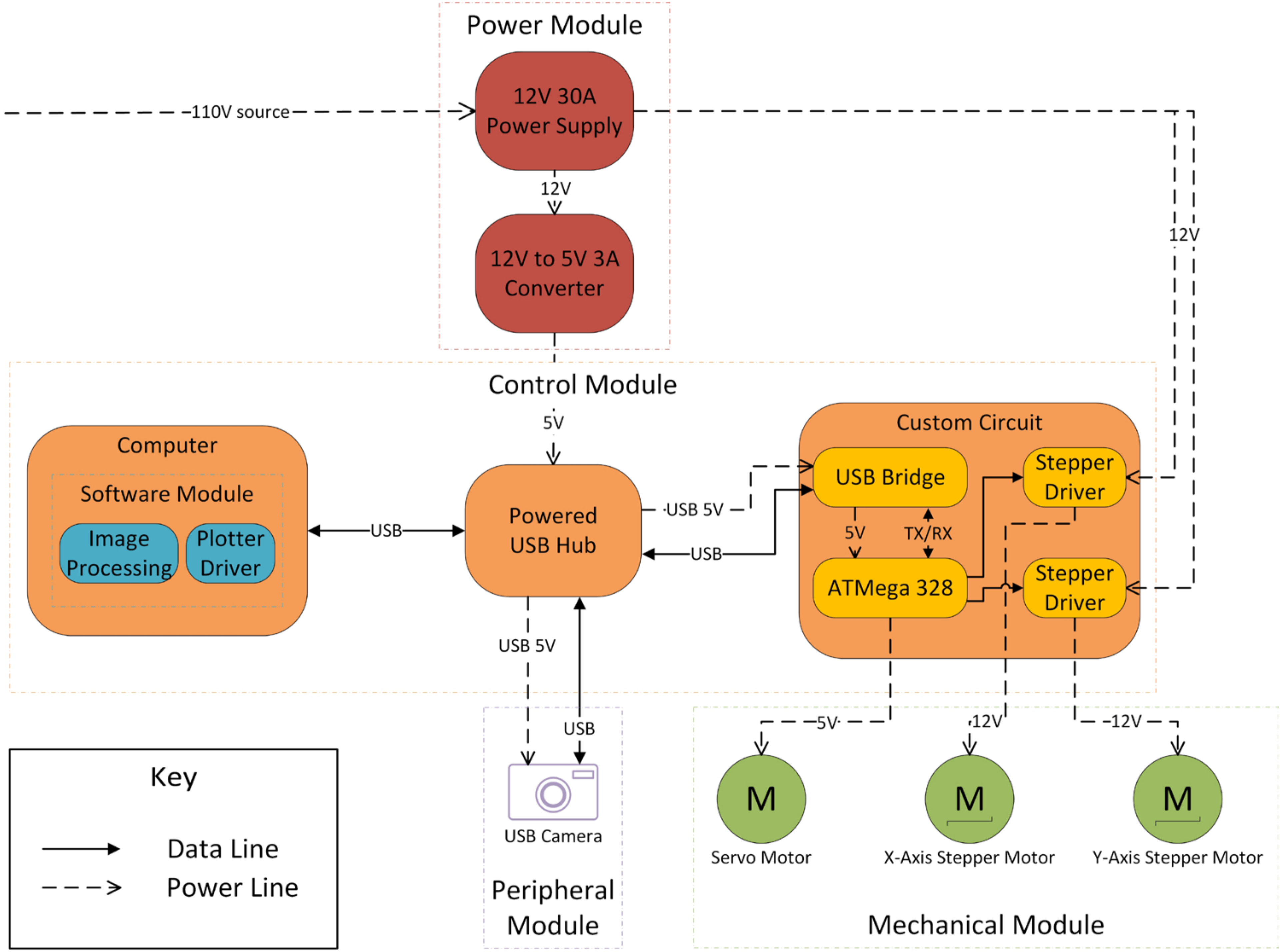
ECE 445, SENIOR DESIGN, SPRING 2018

INTRODUCTION

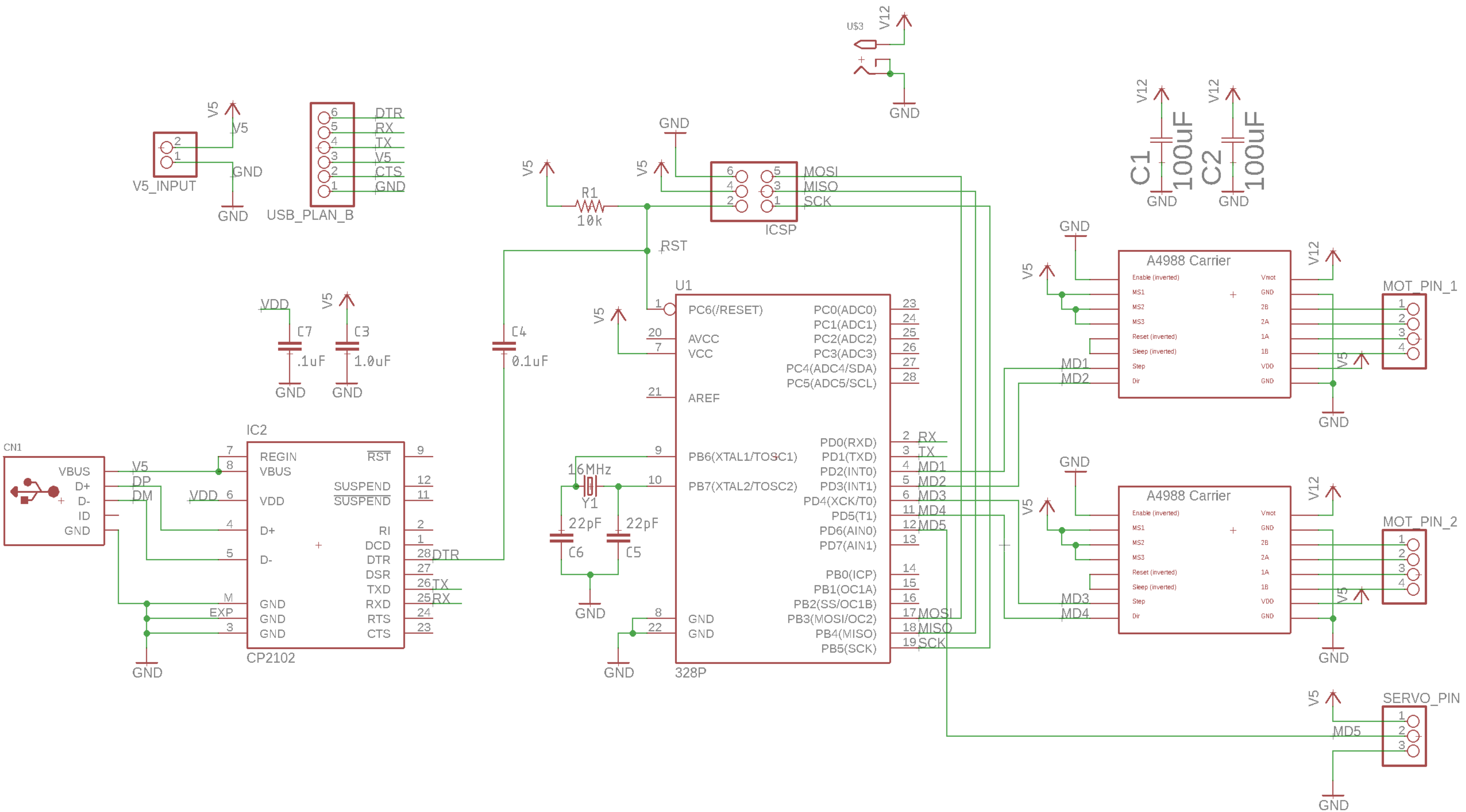
- Our goal: replicate the experience provided by a caricature artist

OBJECTIVES

- Capable of operating in a variety of conditions
- Drawing should take less than 20 minutes
- Entire process should be autonomous

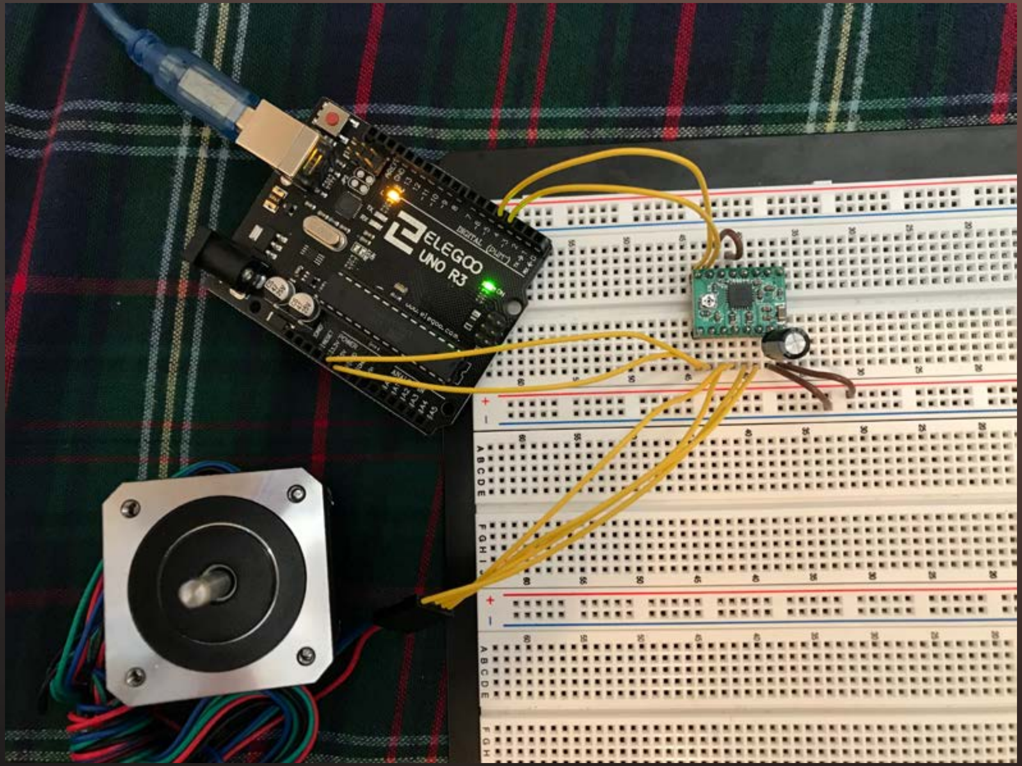


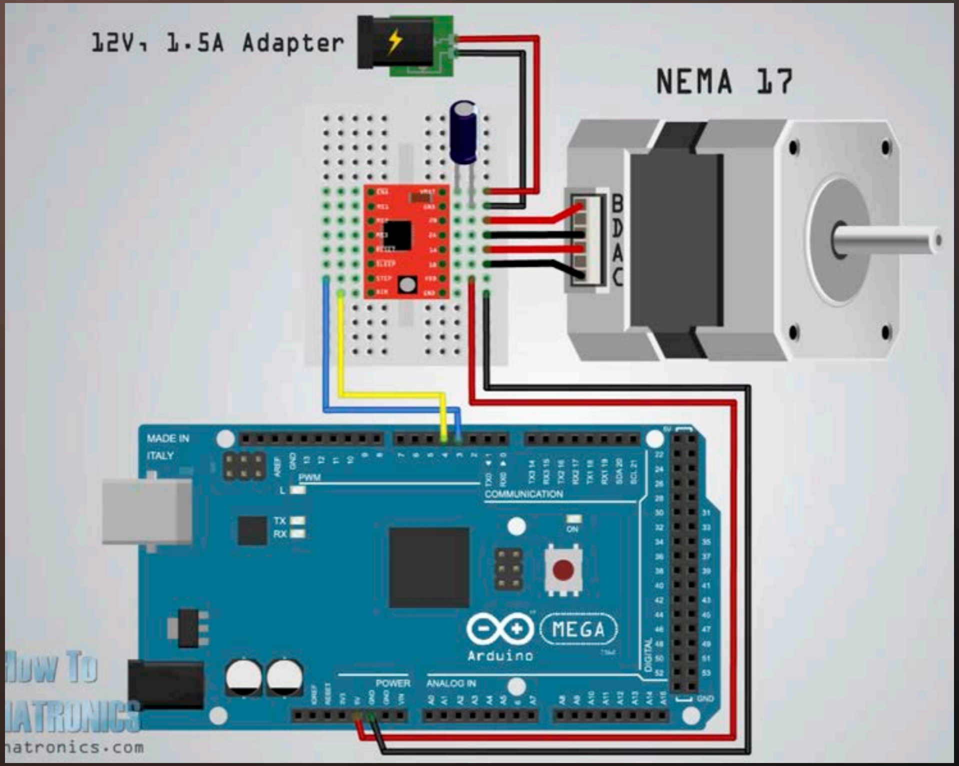
Power Module
110V source
12V 30A
Power Supply
12V
12V to 5V 3A
Converter
12V
Control Module
5V
Computer
Software Module
Image
Processing
Plotter
Driver
USB
Powered
USB Hub
USB 5V
USB
Custom Circuit
USB Bridge
5V
TX/RX
ATMega 328
Stepper
Driver
Stepper
Driver
USB 5V
USB
USB Camera
Peripheral
Module
5V
12V
12V
M
M
M
Servo Motor
X-Axis Stepper Motor
Y-Axis Stepper Motor
Mechanical Module
Key
Data Line
Power Line

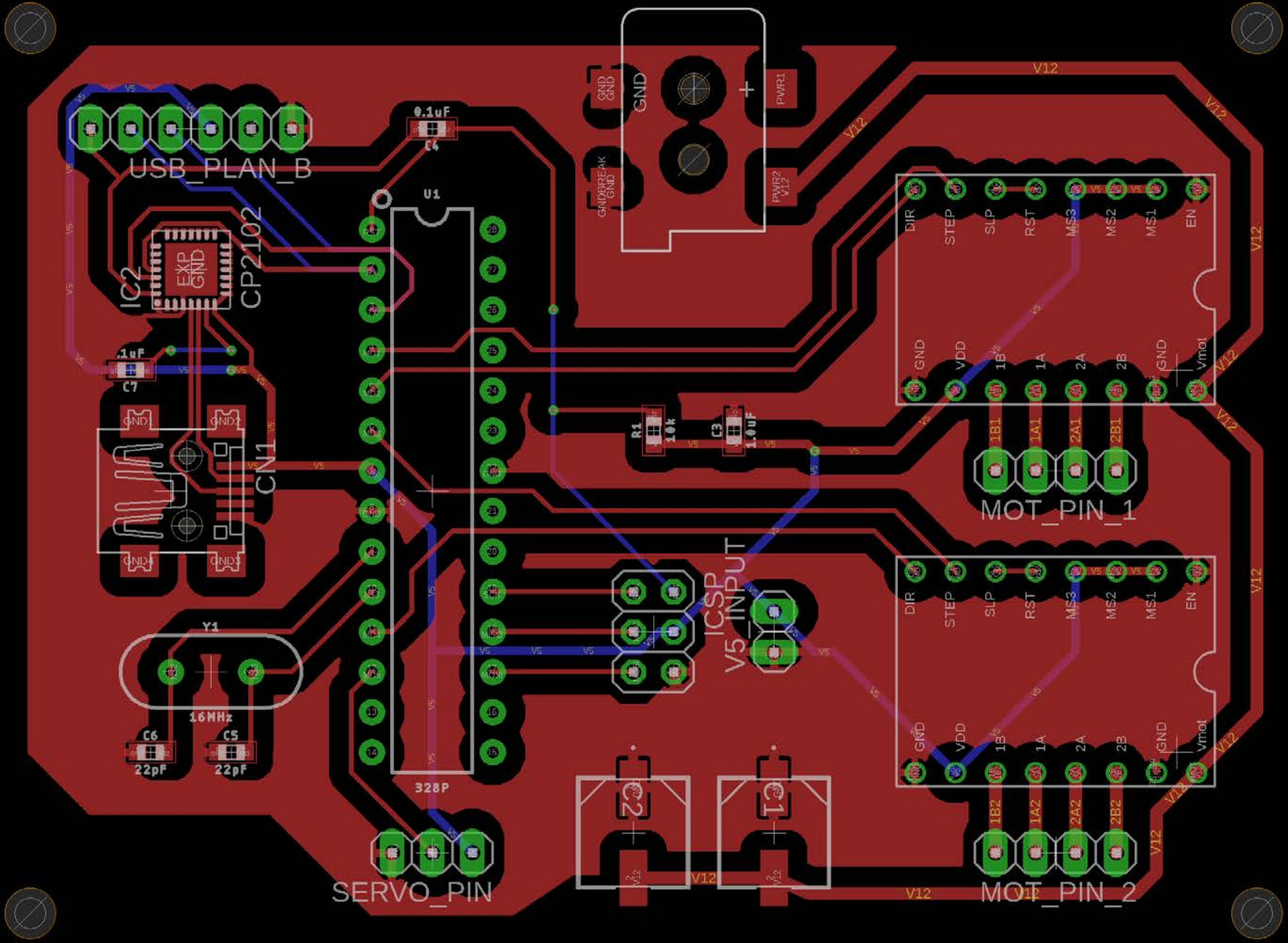


U$3
V12
GND
C1 100uF
C2 100uF
V5_INPUT
USB_PLAN_B
DTR
RX
TX
V5
CTS
GND
R1
10k
ICSP
MOSI
MISO
SCK
RST
U1
328P
PC6(/RESET)
AVCC
VCC
AREF
PB6(XTAL1/TOSC1)
PB7(XTAL2/TOSC2)
PC0(ADC0)
PC1(ADC1)
PC2(ADC2)
PC3(ADC3)
PC4(ADC4/SDA)
PC5(ADC5/SCL)
PD0(RXD)
PD1(TXD)
PD2(INT0)
PD3(INT1)
PD4(XCK/T0)
PD5(T1)
PD6(AIN0)
PD7(AIN1)
PB0(ICP)
PB1(OC1A)
PB2(SS/OC1B)
PB3(MOSI/OC2)
PB4(MISO)
PB5(SCK)
MD1
MD2
MD3
MD4
MD5
VDD
C7 .1uF
C3 1.0uF
C4 0.1uF
16MHz
Y1
22pF
C6
C5
CN1
VBUS
D+
D-
ID
IC2
CP2102
REGIN
RST
SUSPEND
RI
DCD
DSR
TXD
RXD
RTS
M
EXP
A4988 Carrier
Enable (inverted)
MS1
MS2
MS3
Reset (inverted)
Sleep (inverted)
Step
Dir
Vmot
2B
2A
1A
1B
MOT_PIN_1
MOT_PIN_2
SERVO_PIN

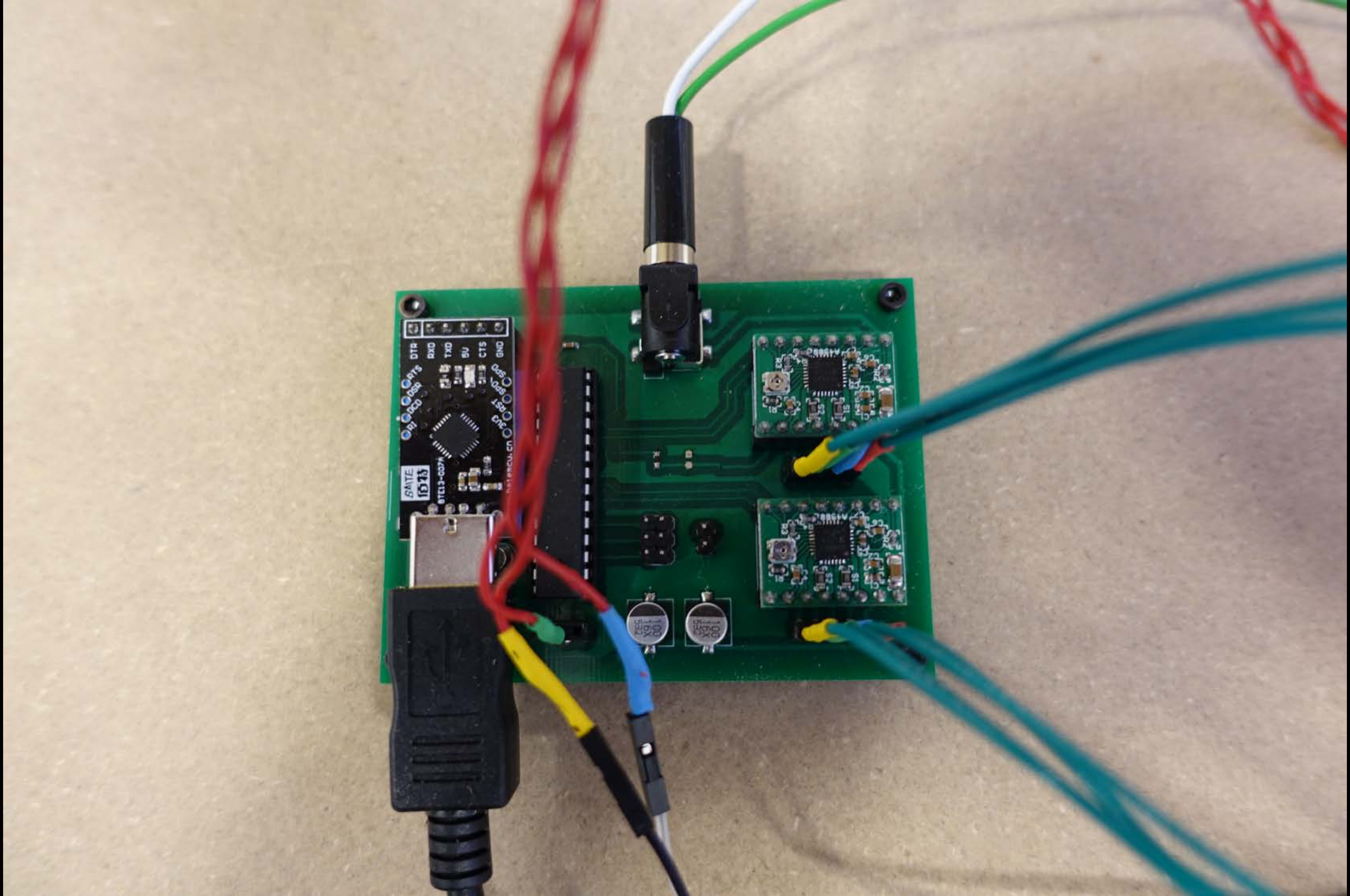
CIRCUIT – PROTOTYPE





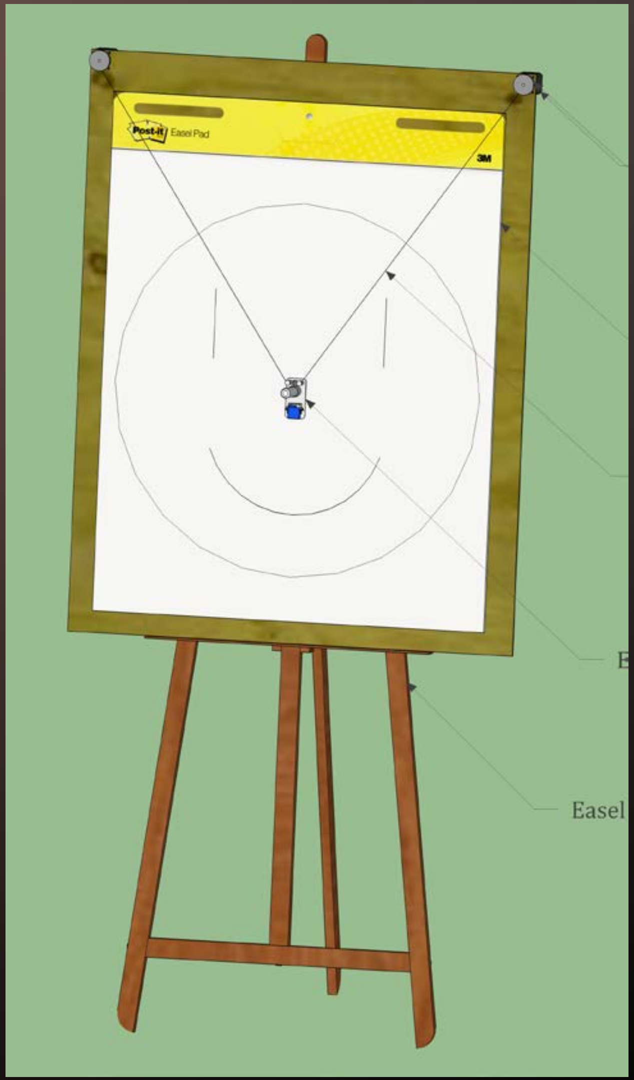


USB_PLAN_B
0.1uF
C4
U1
IC2
EXP GND
CP2102
.1uF
C7
GND1
GND2
CN1
GND4
GND3
Y1
16MHz
C6
22pF
C5
22pF
328P
SERVO_PIN
GND
PWR1
PWR2
V12
R1
10k
C3
1.0uF
ICSP
V5_INPUT
C2
C1
V12
V5
DIR
STEP
SLP
RST
MS3
MS2
MS1
EN
GND
VDD
1B
1A
2A
2B
GND
Vmot
1B1
1A1
2A1
2B1
MOT_PIN_1
1B2
1A2
2A2
2B2
MOT_PIN_2


DTR
RXD
TXD
5V
CTS
GND

MECHANICAL DESIGN – REQUIREMENTS

- Greatly influences rest of project
- Simple
- Support drawings > 8.5" X 11"
  - (Pad is 20" x 23")
- No special tools
- Correct and reliable!



MECHANICAL DESIGN – DIMENSIONS

Accuracy Equations

$$x = \cos(\alpha) * d$$

$$y = \sin(\alpha) * d$$

$$d = \sqrt{x^2 + y^2}$$

$$\alpha = atan2(y, x)$$

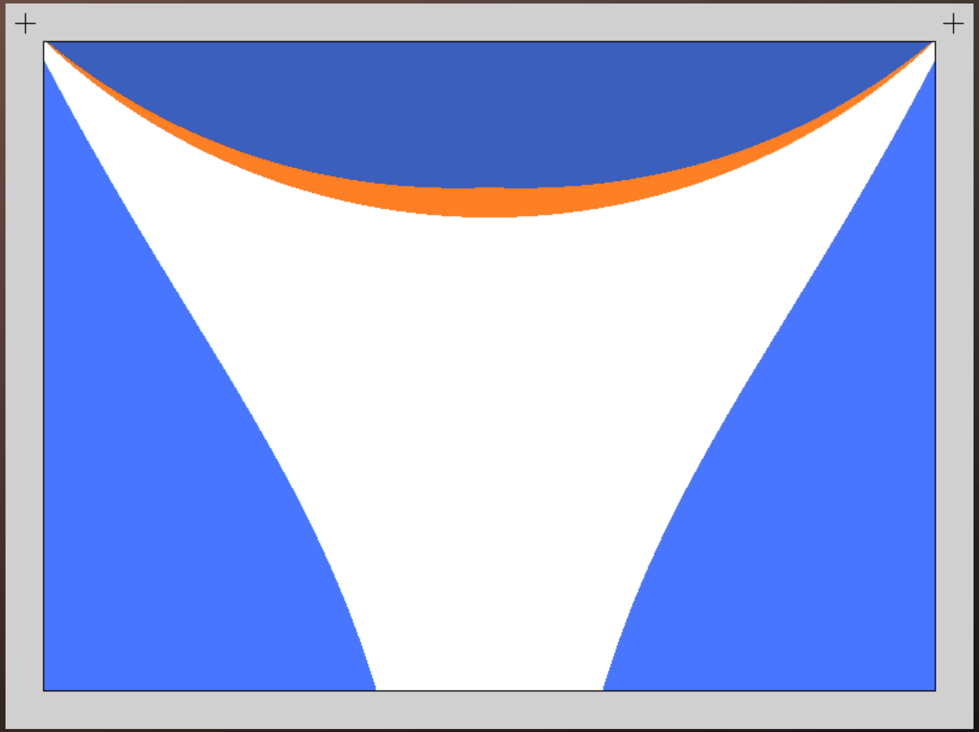
Tension Equations

$$t_{left} = \frac{\cos(\alpha_{right}) * m}{\cos(\alpha_{left}) * \sin(\alpha_{right}) + \sin(\alpha_{left}) * \cos(\alpha_{right})}$$

$$t_{right} = \frac{\cos(\alpha_{left}) * m}{\cos(\alpha_{left}) * \sin(\alpha_{right}) + \sin(\alpha_{left}) * \cos(\alpha_{right})}$$

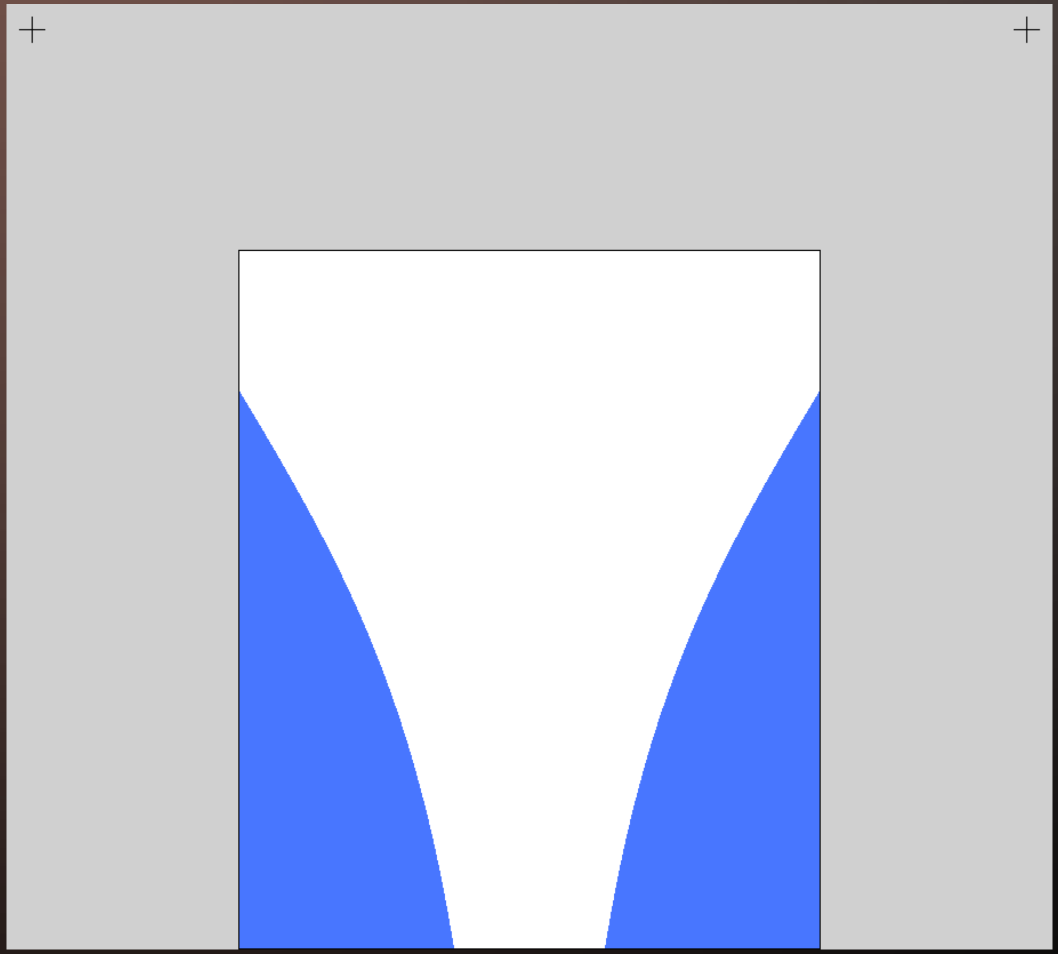
MECHANICAL DESIGN – DIMENSIONS

- Program calculates tension/accuracy across area
- Orange = Poor Accuracy
  - (1 unit deviates more than 1.4 units)
- Blue = Poor Tension
  - (T < .5 or T > 1.5)



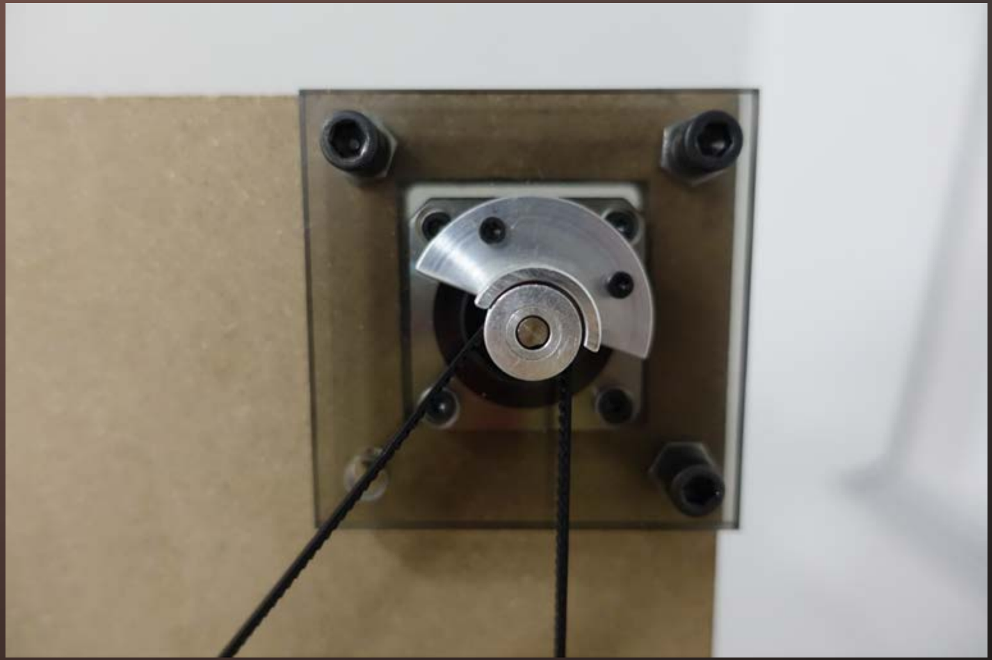
MECHANICAL DESIGN – DIMENSIONS

- 1mm = 1px
- Fix motors, allow border resizing
- Calculate plywood dimensions
  - 36” x 32.5”
- Calculate optimal draw area
  - 13.66” x 10.79”



MECHANICAL DESIGN – DRIVE SYSTEM

- Many different options
  - (wire, beads, belts)
- Drive system should be consistent
- GT2 Belts, 16-Tooth Pulleys

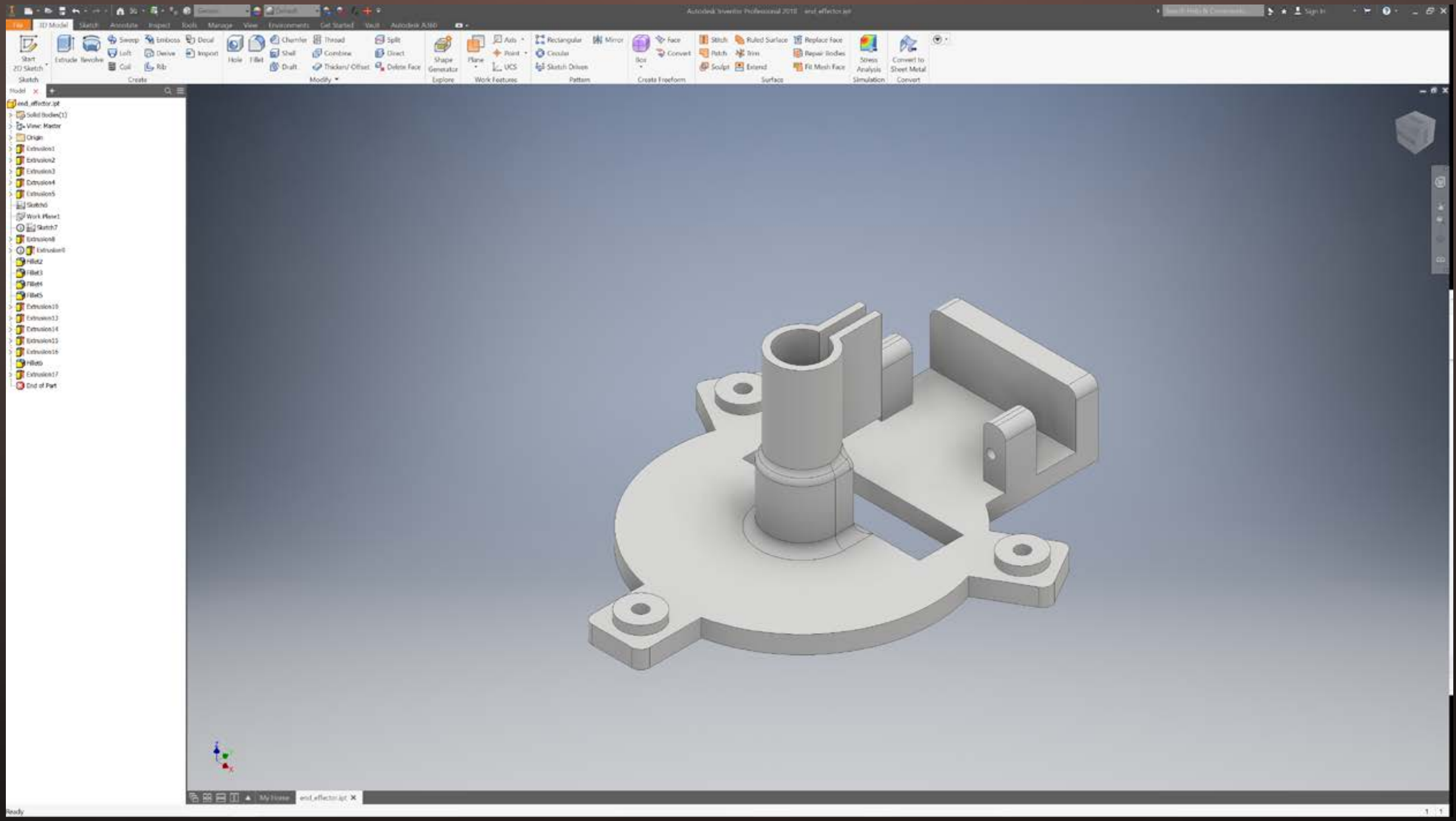


$$STEPS\_PER\_MM = \frac{STEPS\_REV * MICRO\_STEPS}{BELT\_PITCH * PULLEY\_TEETH} = \frac{200*16}{2*16} = 100$$

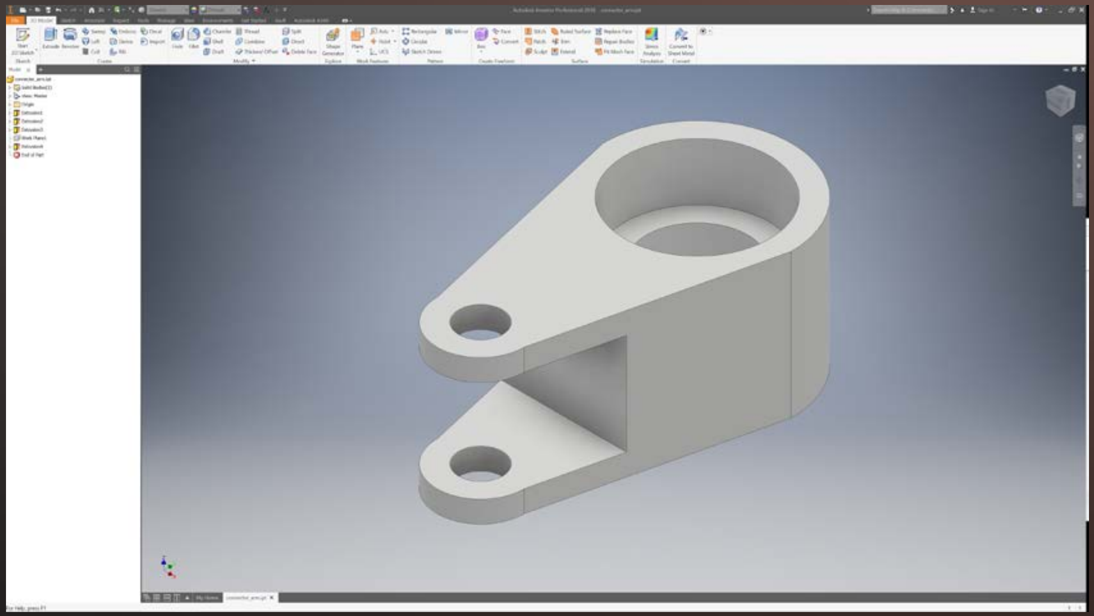
MECHANICAL DESIGN – END EFFECTOR

- Servo lifts end effector
- Screwless pen holder
- No plastic on plastic contact
- Adjustable weight

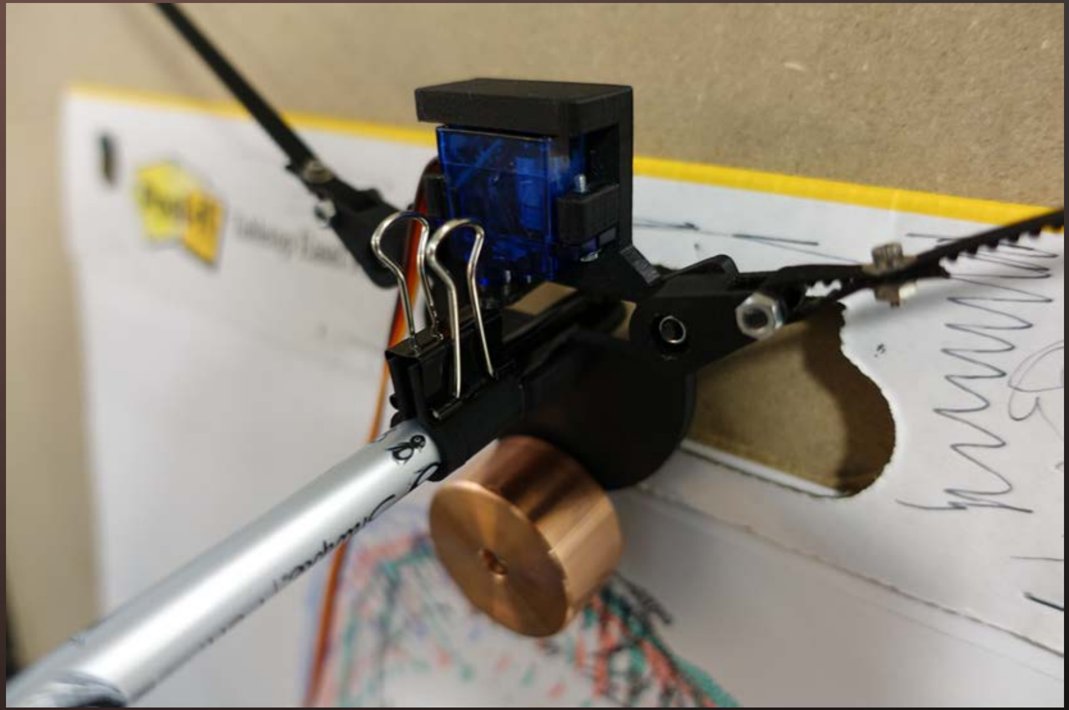
MECHANICAL DESIGN – END EFFECTOR



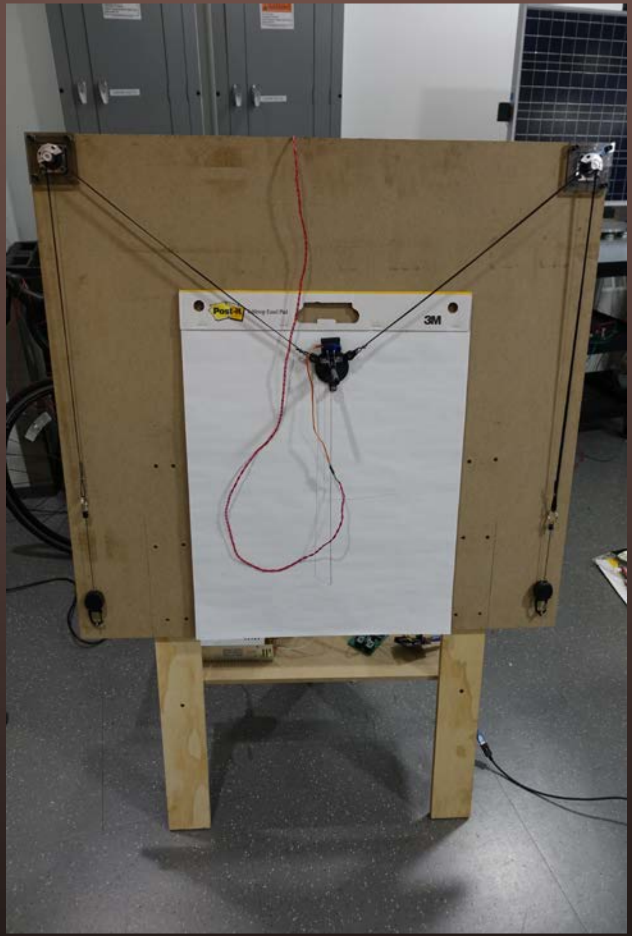
MECHANICAL DESIGN – END EFFECTOR



MECHANICAL DESIGN – END EFFECTOR



MECHANICAL DESIGN



FIRMWARE – REQUIREMENTS

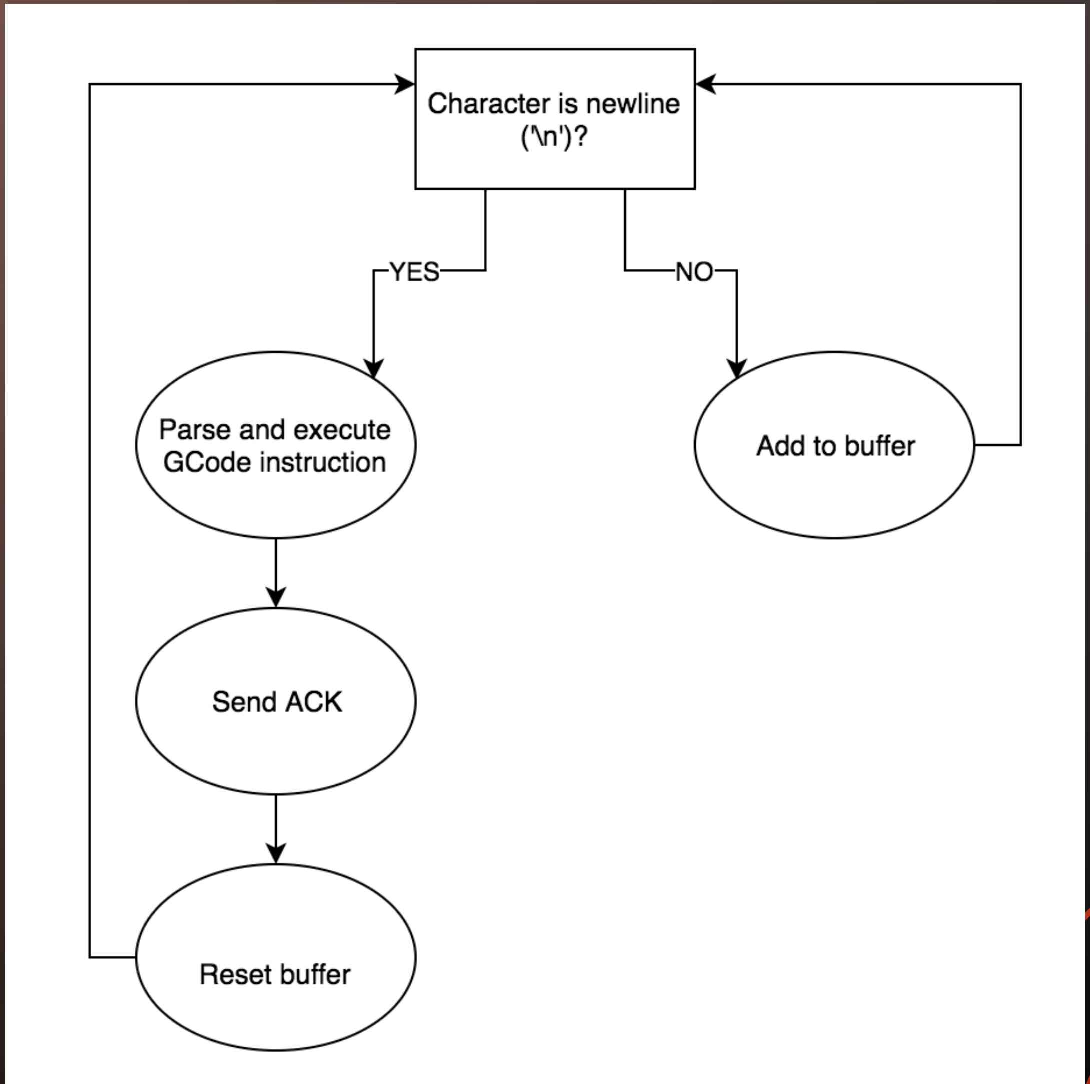
- Interface with computer
- Execute G-Code instructions
- Fault-tolerant
- Fast and reliable

FIRMWARE – CHALLENGES

- How do we quickly and reliably communicate G-Code instructions?
- How do we draw a line?

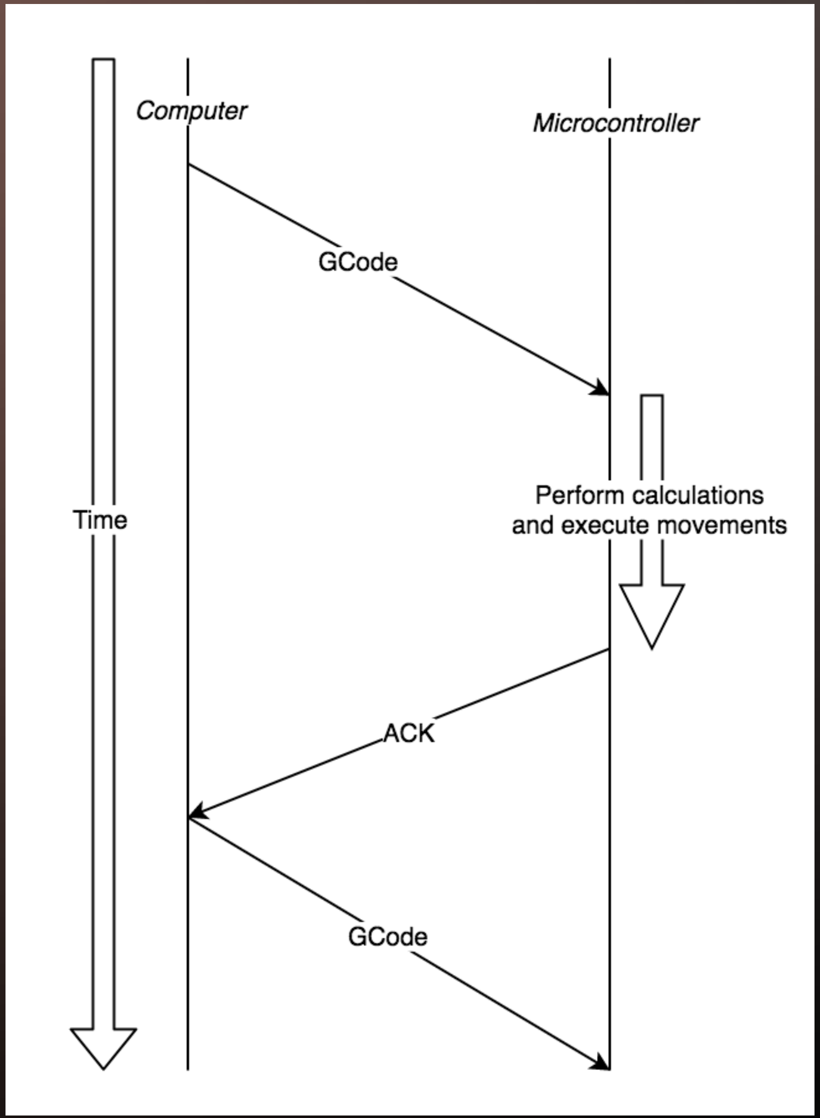
FIRMWARE – COMMUNICATE G-CODE?

- G-Code is delimited by new line characters
- G-Code example:
  - "G0 X25 Y93 Z1 F32\n"
  - X – X position
  - Y – Y position
  - Z – servo position
  - F – stepper motor speed
- Python for communication on the sender side



FIRMWARE – SMALL BUFFER SIZE ☹

- 64 byte serial buffer size on ATMega328 (too small!)
- Solution: communicate acknowledgement of executed instruction



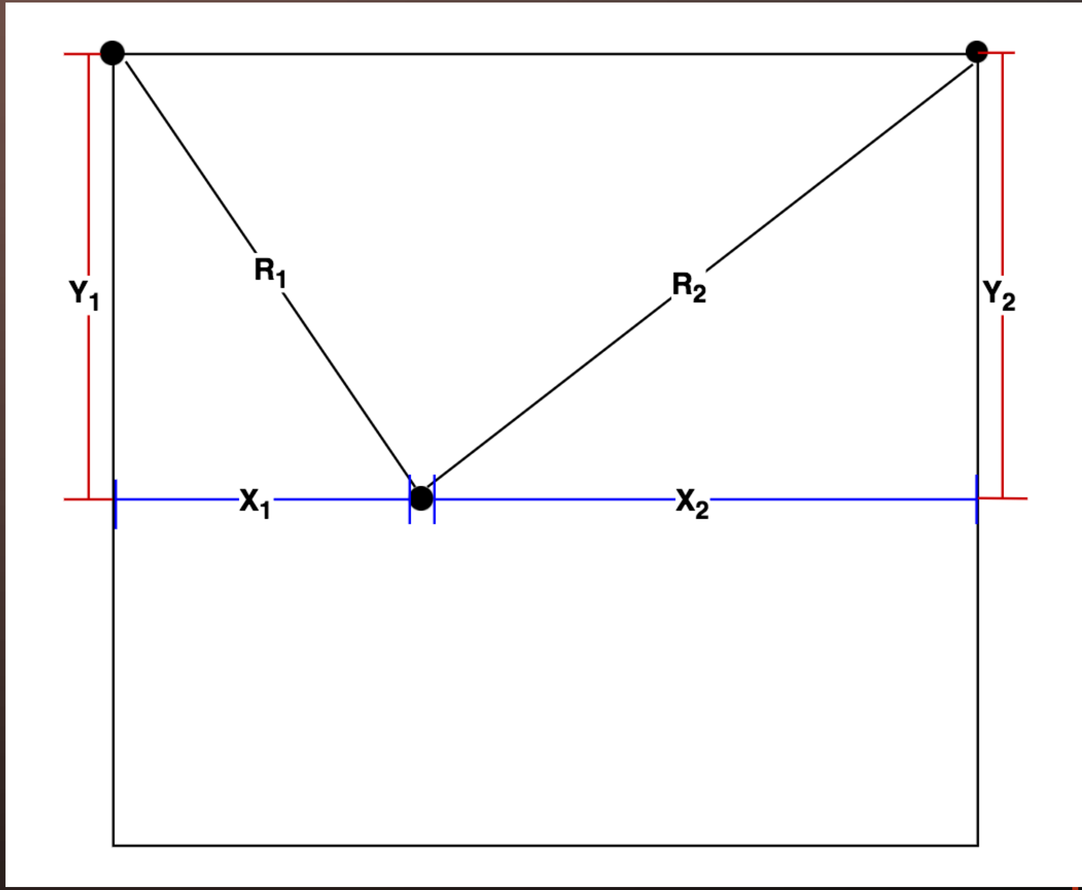
FIRMWARE – HOW DO WE DRAW A LINE?

$$\Delta R_1 = \sqrt{X_{1,final}^2 + Y_{1,final}^2} - \sqrt{X_{1,initial}^2 + Y_{1,initial}^2}$$

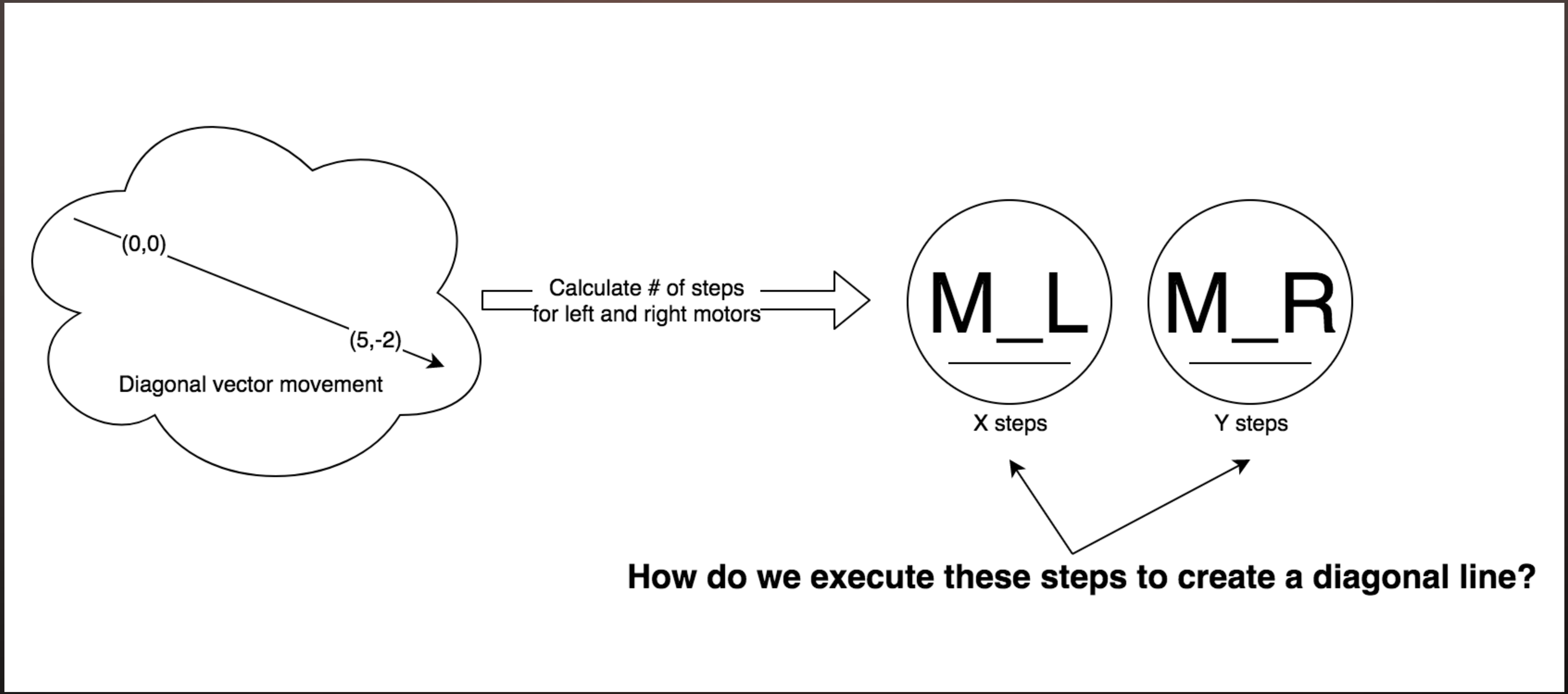
$$\Delta R_2 = \sqrt{X_{2,final}^2 + Y_{2,final}^2} - \sqrt{X_{2,initial}^2 + Y_{2,initial}^2}$$

$$MSteps_1 = STEPS\_PER\_MM * \Delta R_1$$

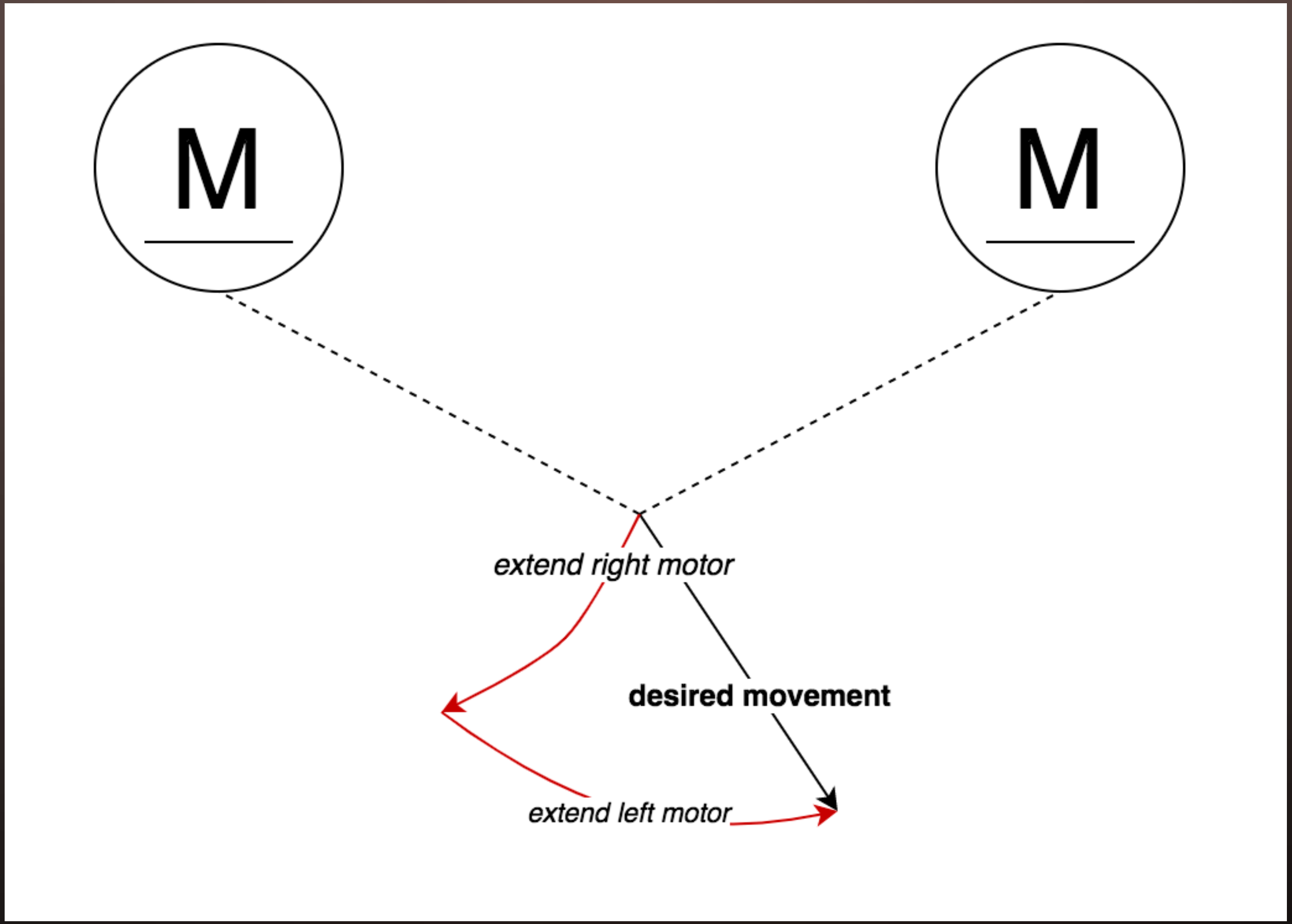
$$MSteps_2 = STEPS\_PER\_MM * \Delta R_2$$



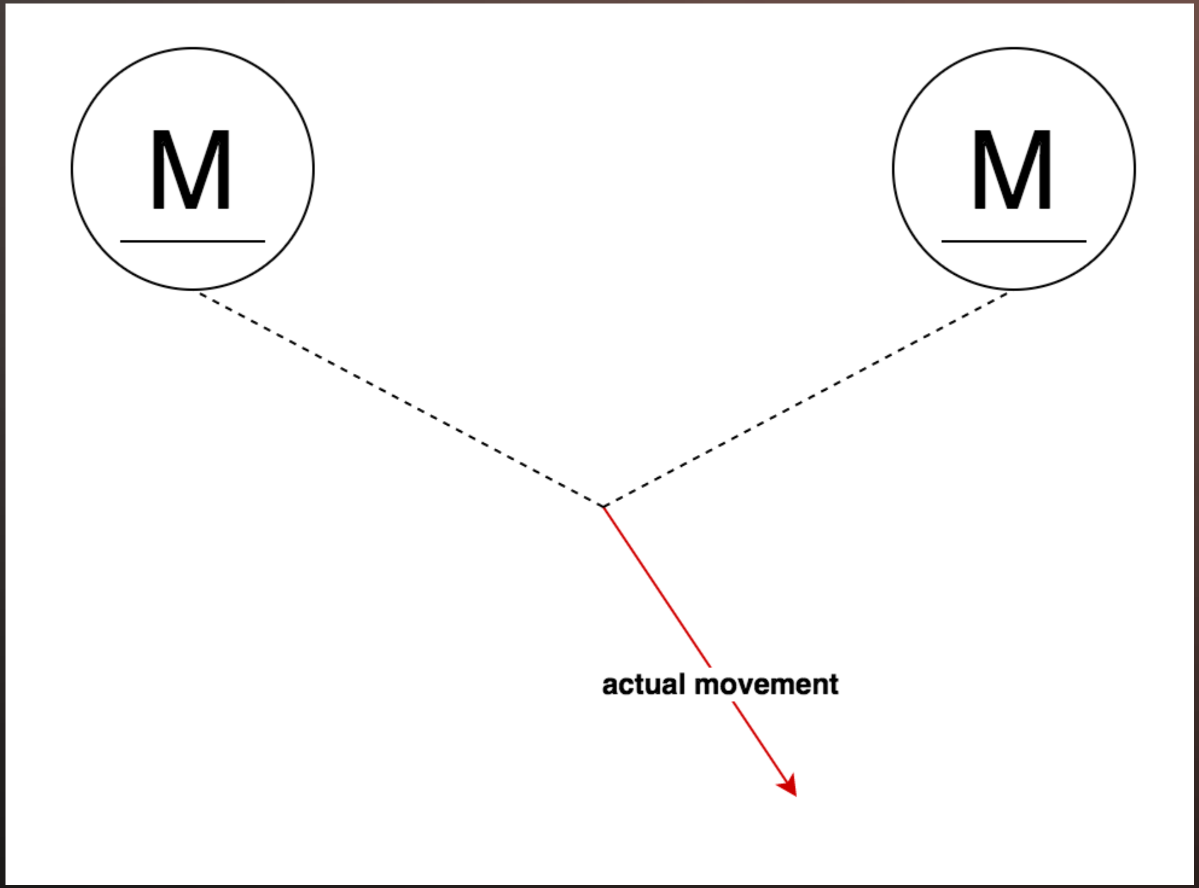
FIRMWARE – HOW DO WE DRAW A LINE?

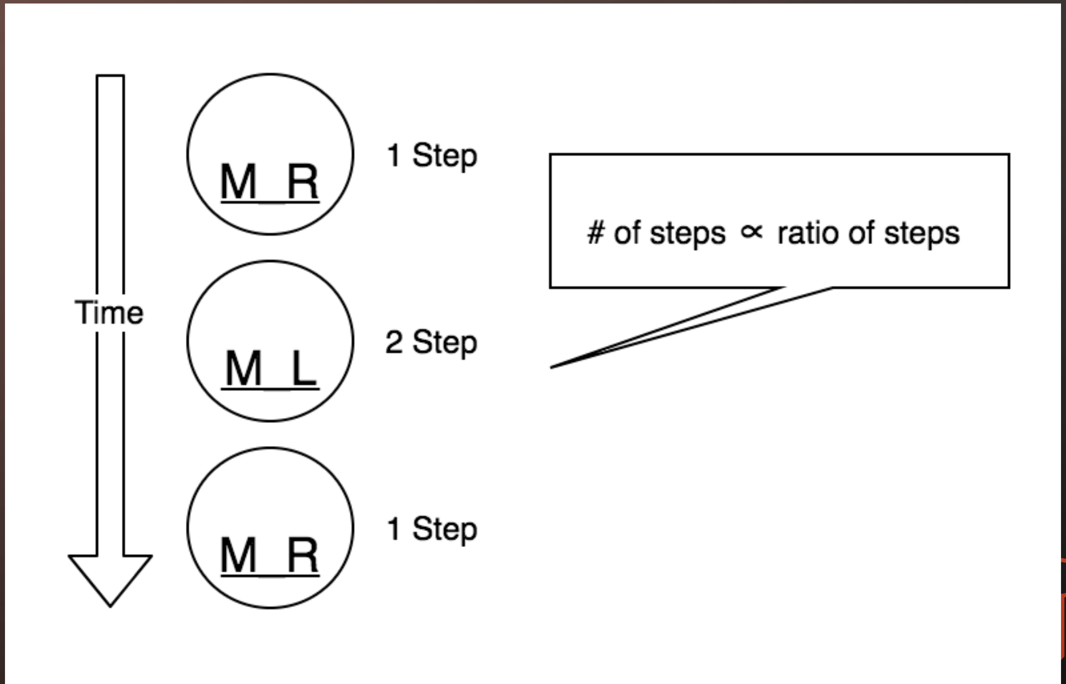


FIRMWARE – PROBLEM



FIRMWARE – SOLUTION

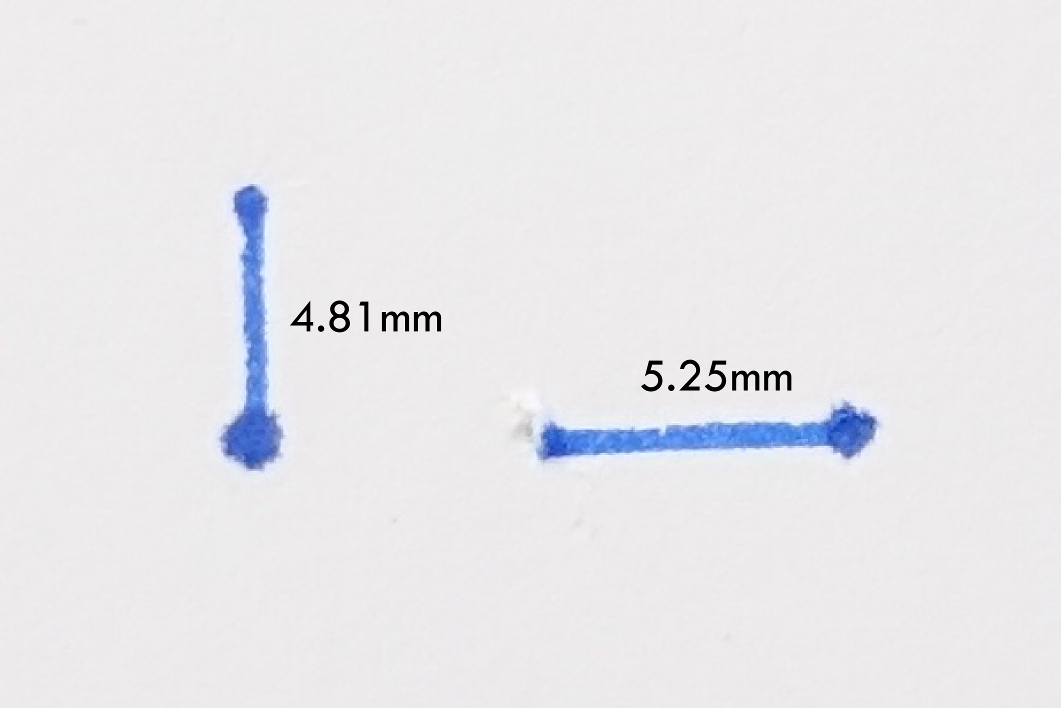




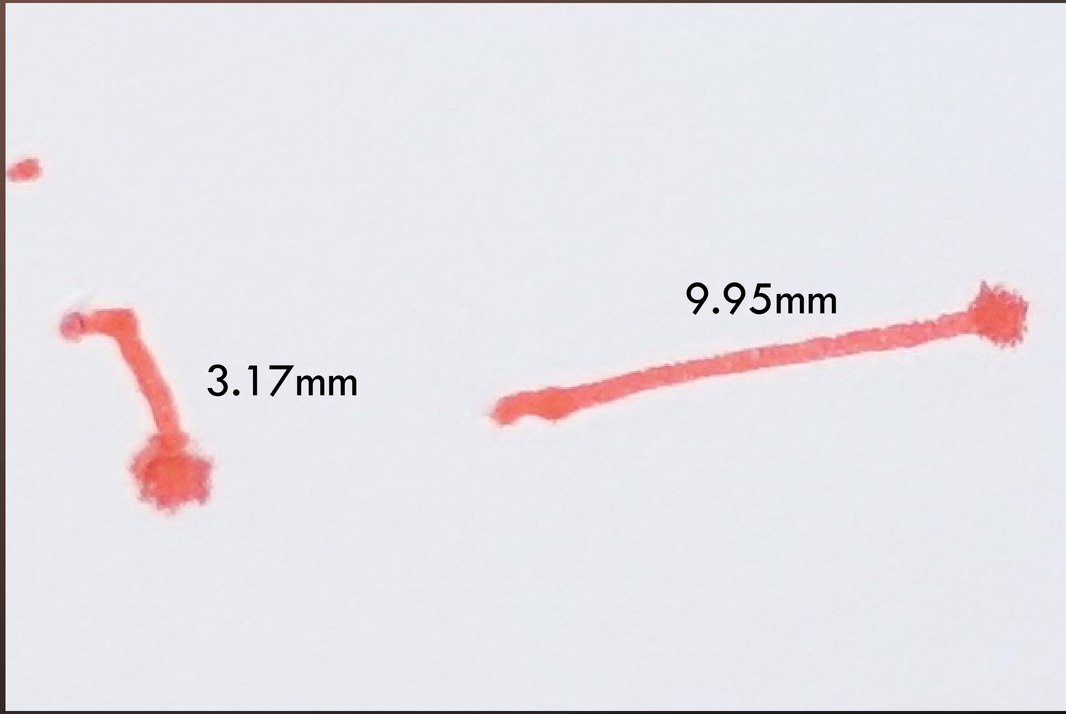
FIRMWARE –BETTER SOLUTION

- Interleaving worked!
- We had to send the HI/LO signals ourselves
  - Jerky movements
  - Single speed PWM wave
  - Non-adjustable speed
- External library (better solution)
  - 30% speed increase
  - Less jerkiness
  - Speed adjustable by G-Code
  - Cleaner code (no need to calculate interleaving)

FIRMWARE – TOLERANCE ANALYSIS 5MM



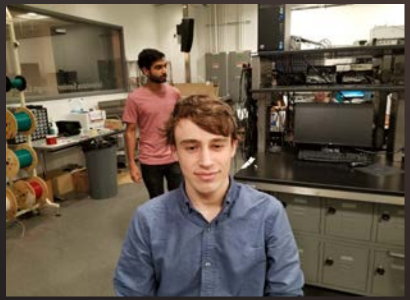
good region

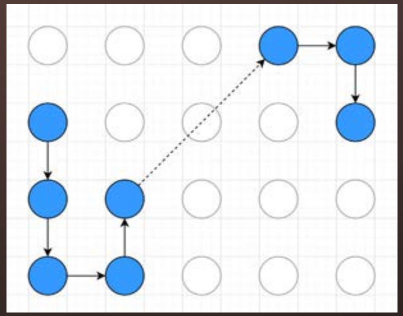


bad region

SOFTWARE OVERVIEW

Image capture → Segmentation → Filtering & morphological operations → Tool path optimization → G-Code

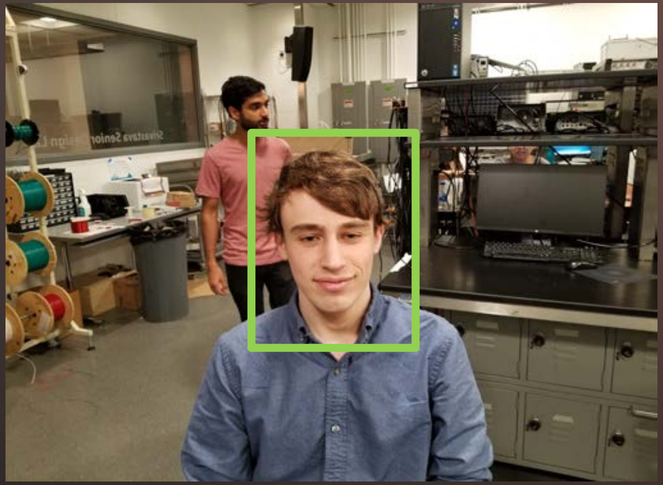




```
G0 X0 Y0 Z1 F25
G0 X-40.8 Y0.0 Z1
G0 X-40.8 Y0.0 Z0
G0 X-39.6 Y0.0 Z0
G0 X-38.4 Y0.0 Z0
G0 X-37.2 Y0.0 Z0
G0 X-36.0 Y0.0 Z0
```

SEGMENTATION

Face detection with Haar Cascades [1]



Outside rectangle = “probable background”
Inside rectangle = “probable foreground”

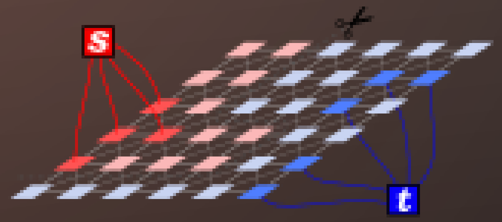
After ~5 iterations of GrabCut

[1] https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html

SEGMENTATION

- "GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts [2]
- Gaussian Mixture Models represent foreground and background
- Algorithm: alternate between parameter estimation and segmentation

Estimate GMM parameters θ



Minimize $\sum_{i=0}^{N} \underbrace{U(\theta)}_{\text{Likelihood}} + \underbrace{V(\theta)}_{\text{Smoothing}}$

[2] https://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

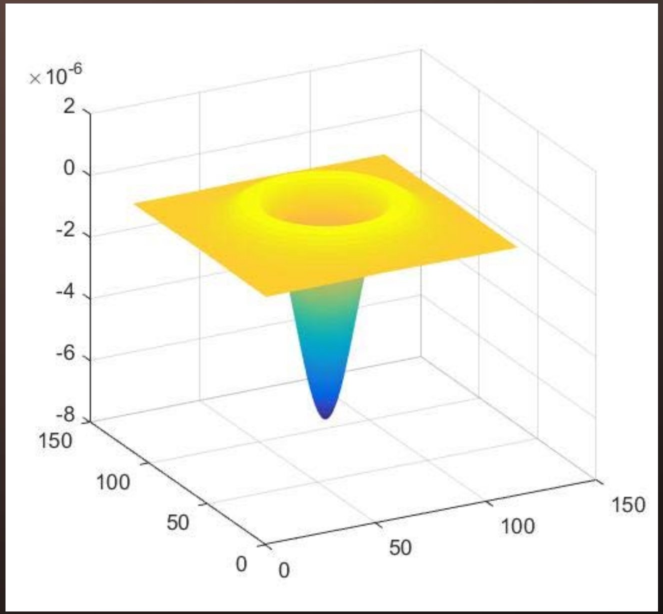
FILTERING & MORPHOLOGICAL OPERATIONS

- Edge detection: Laplacian of Gaussian filter

$$\mathrm{LoG}(x,y) = -\frac{1}{\pi\sigma^4}\left[1-\frac{x^2+y^2}{2\,\sigma^2}\right]e^{-\frac{x^2+y^2}{2\,\sigma^2}}$$

- Tune sigma using our dataset

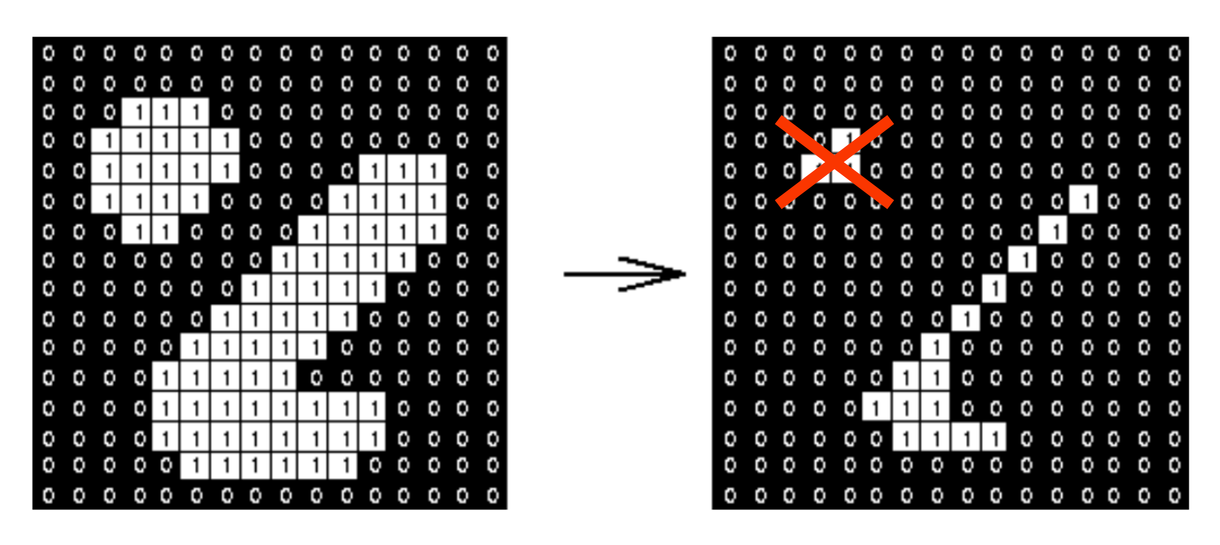
- Apply threshold to get binary image



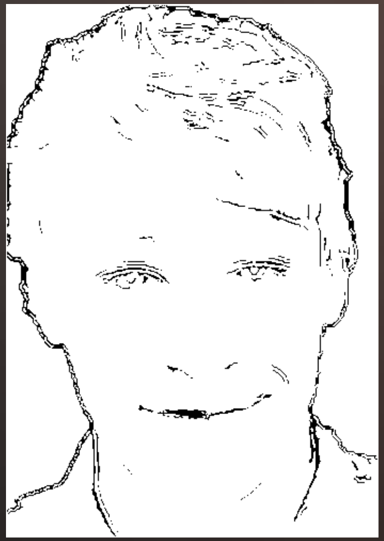
FILTERING & MORPHOLOGICAL OPERATIONS

- Binary erosion – reduce object thickness

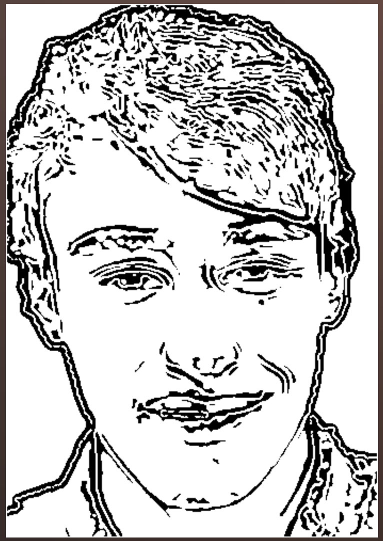
- Connected component labeling – identify and remove small components



FILTERING & MORPHOLOGICAL OPERATIONS



$\sigma=1$
$N=897$



$\sigma=4$
$N=10{,}327$

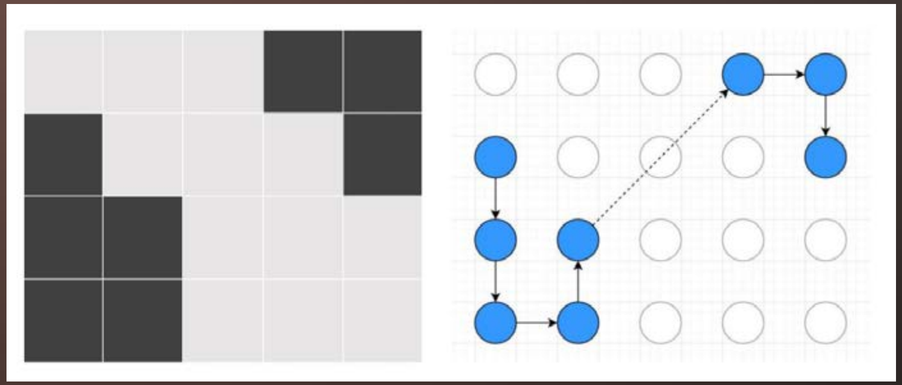
$\sigma=6$
$N=15{,}325$

$\sigma=8$
$N=20{,}187$

$\sigma=10$
$N=24{,}263$

TOOL PATH OPTIMIZATION

- Pen needs to travel to every filled pixel
- Minimize draw time = find Hamiltonian cycle of minimum length
- Greedy / nearest insertion heuristics
  - computationally inexpensive – O(n log n)
  - future improvements possible via software update



Binary image and its corresponding TSP instance

TOOL PATH OPTIMIZATION

TOOL PATH OPTIMIZATION – RESULTS

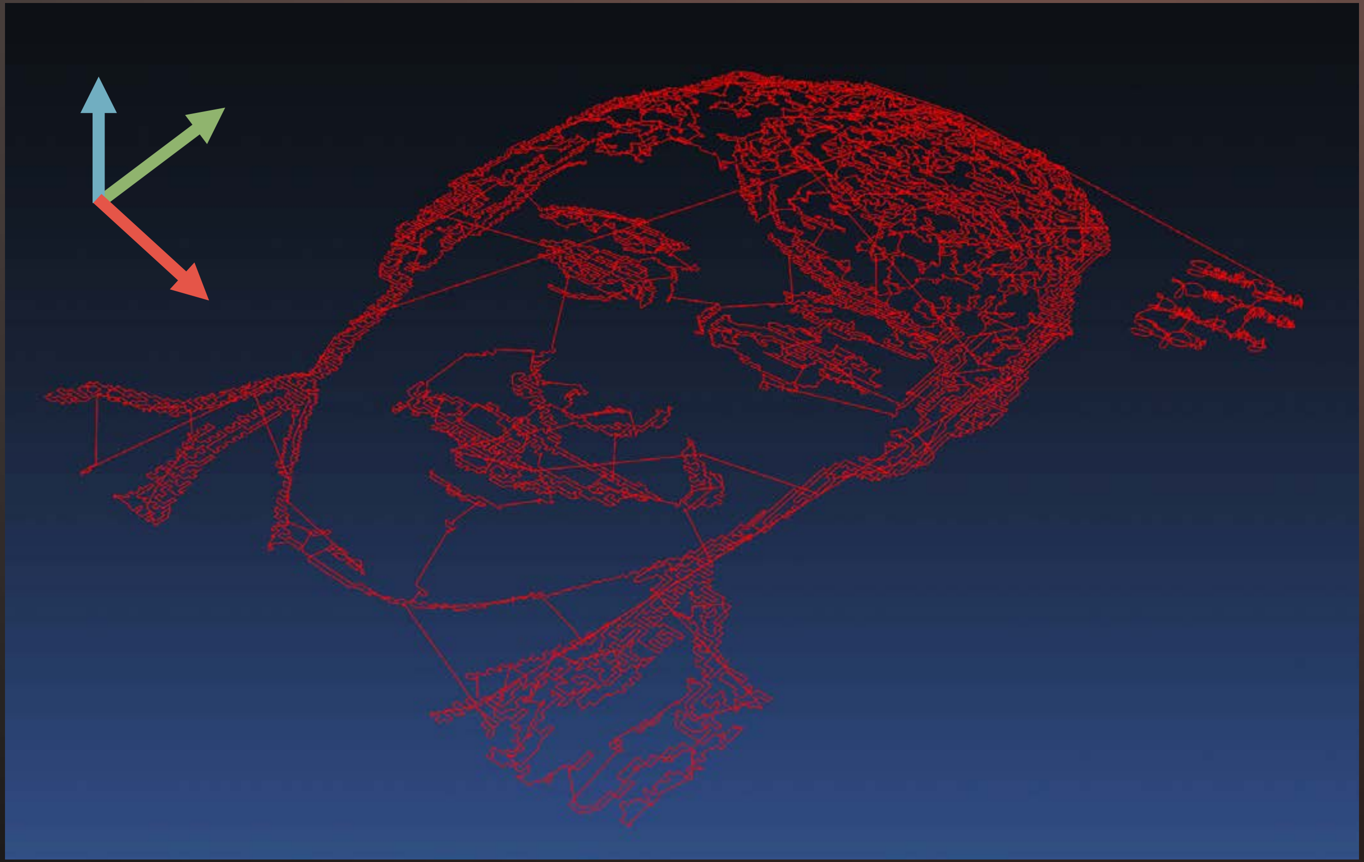
- Optimization with greedy TSP solver gives 58% average reduction in draw time
- Leverage sparse structure of images
- Runs in < 10 sec on laptop CPU for all images tested

	Zig-Zag pattern	Optimized
Travel distance (mm)	36,900 (± 1,780)	14,300 (± 2,210)
Pen lifts	237 (± 7.10)	309 (± 42.1)
Travel time (s) *	1,480 (± 71.3)	573 (± 88.5)
Pen lift time (s) **	59.3 (± 1.77)	77.3 (± 10.5)
Total draw time (min)	**25.6 (± 1.21)**	**10.8 (± 1.63)**

Table 1: Results of tool path optimization

* Running at 25mm / s
** Assuming 0.250s / pen lift

G-CODE VALIDATION – SIMULATION



isomorphic view

side view

ETHICS

- Overfitting issues need to be examined closely
- Our project isn't art; it's just for fun (entertainment?)

CONCLUSION

- Works from start to finish
- Single command to complete entire process
- Drawings take ~15 minutes to complete
- Future work?