SVPWM BLDC ESC

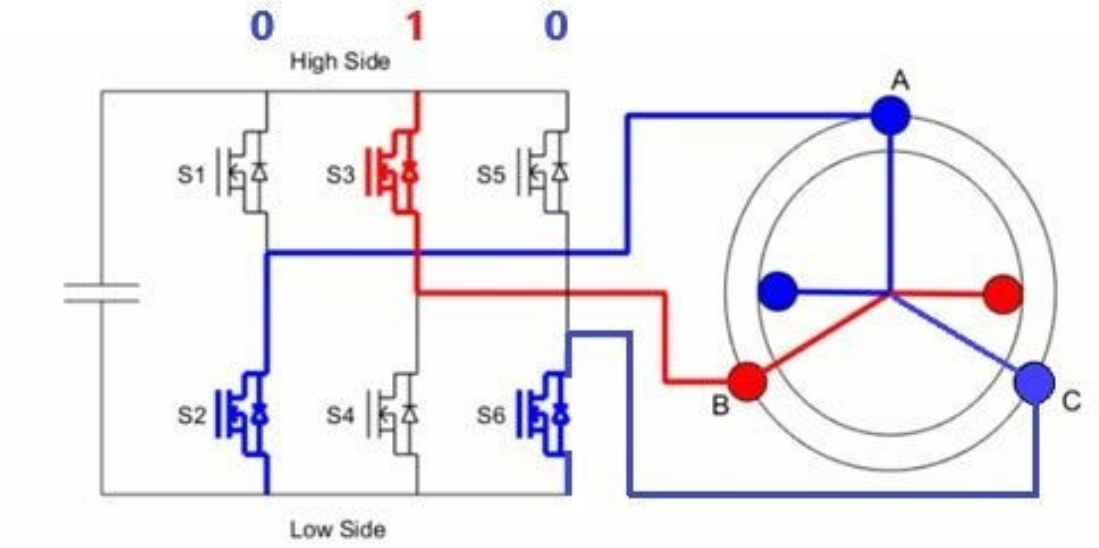
Ryan Cramer & Avery White

SVPWM? What's That?

SVPWM is a way to control three phase motors by modulating the duty cycle. Changing the duty cycle changes the strength.

To drive it clockwise:
FAST: DUTY_A = 100%, DUTY_B = 50%, DUTY_C = 0%
SLOW: DUTY_A = 30%, DUTY_B = 15%, DUTY_C = 0%

To drive it counterclockwise:
FAST: DUTY_A = 0%, DUTY_B = 50%, DUTY_C = 100%
SLOW: DUTY_A = 0%, DUTY_B = 15%, DUTY_C = 30%

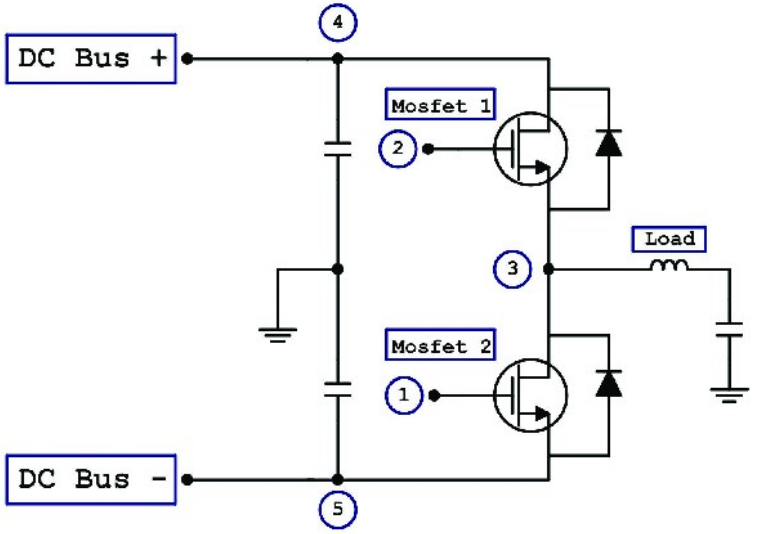


BUT: Motor is constantly rotating, so these duty cycles need to osc
back EMF / torque from the motor

ROTOR angle is really hard to track!!! REQUIRES HALL EFFECT SENSING

SVPWM research compiled by Avery White

STM32 → Motor Block Diagram

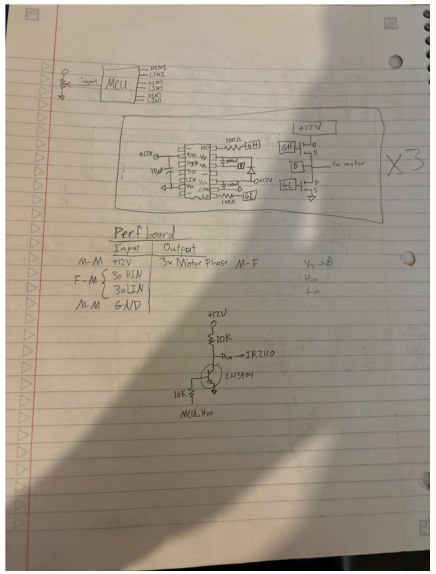


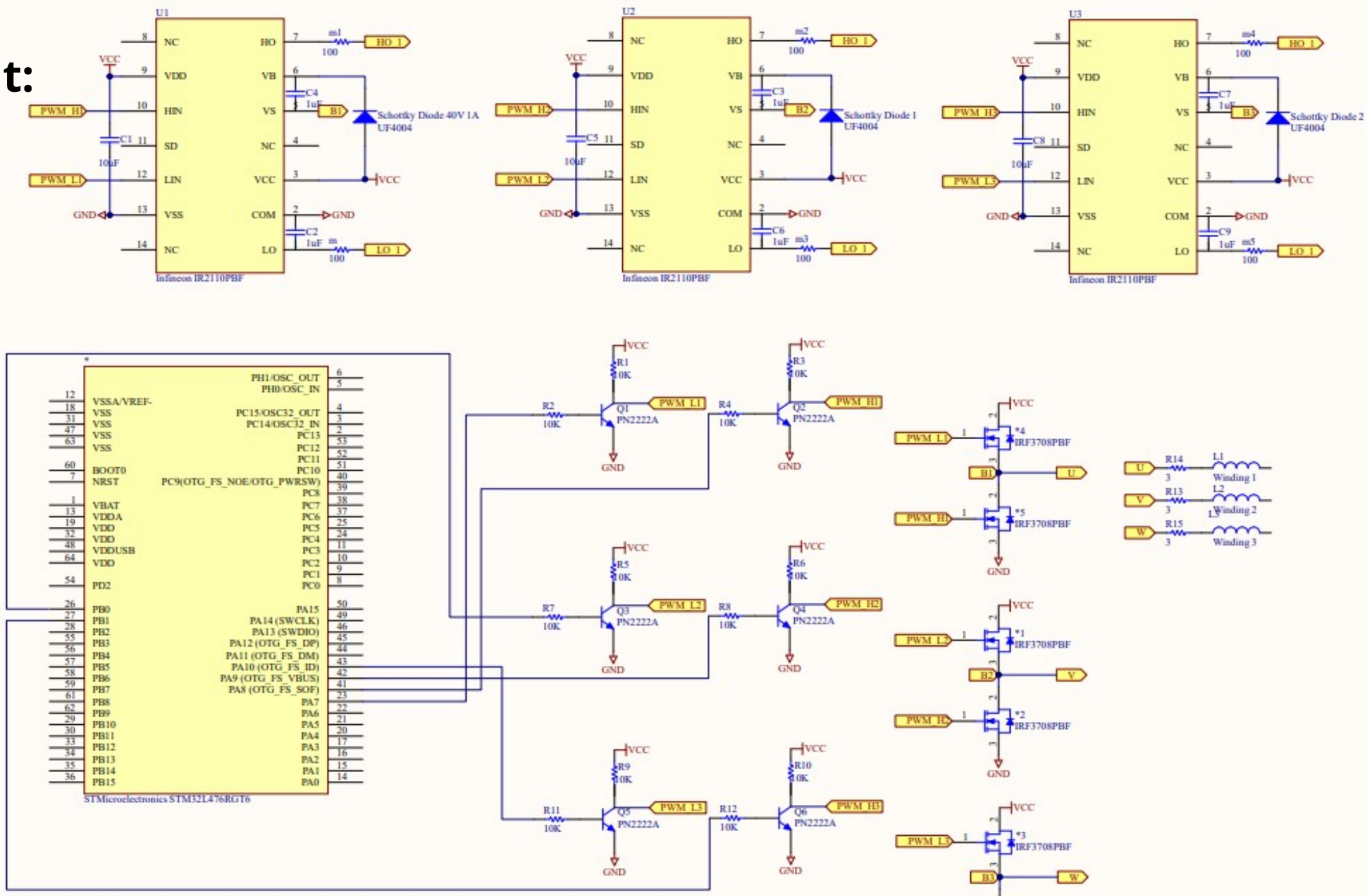
TIM1 PWM_1 → IR2110 IC #1 → Half-Bridge #1 → Motor Winding U

TIM1 PWM_2 → IR2110 IC #2 → Half-Bridge #2 → Motor Winding V

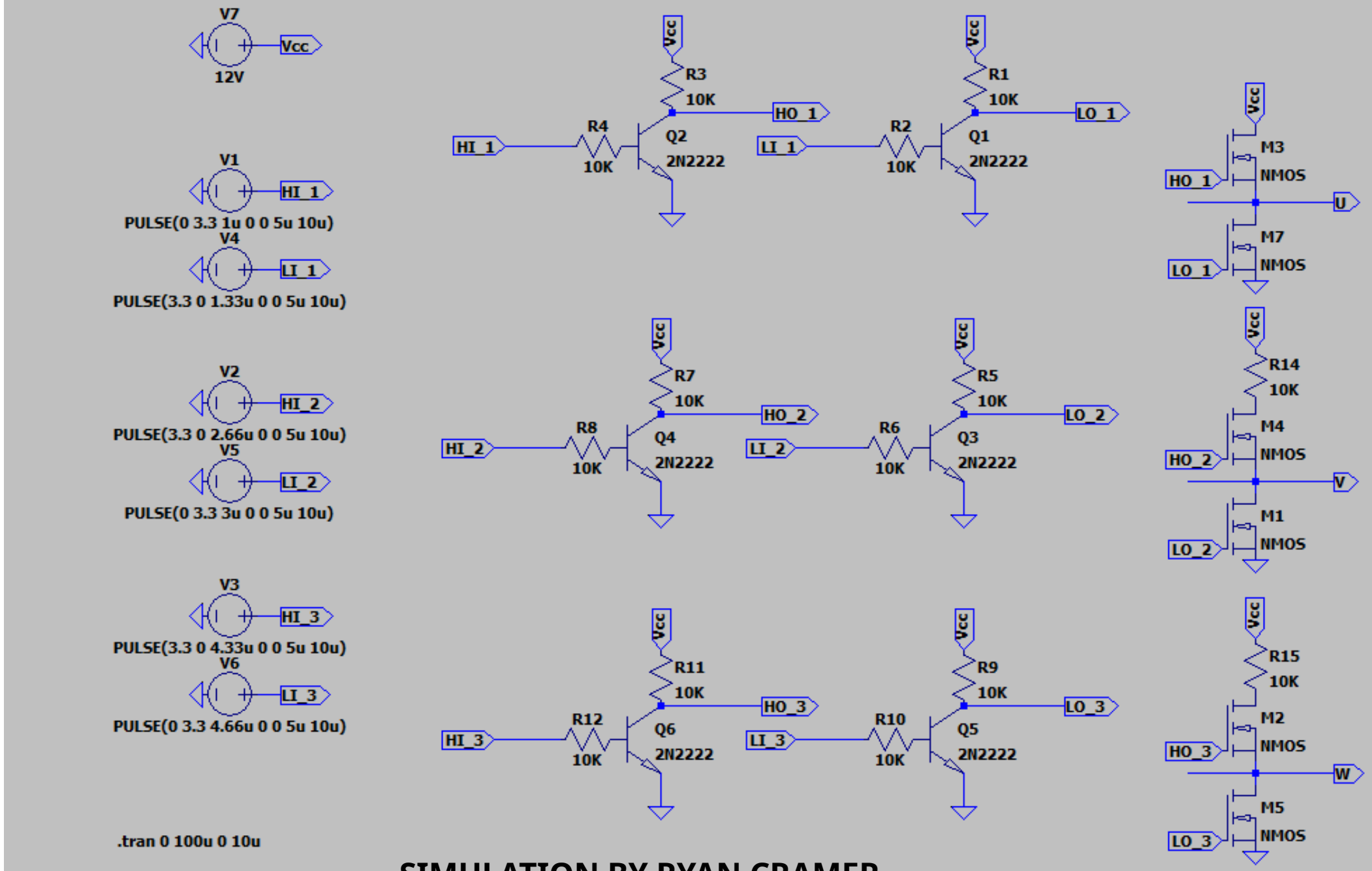
TIM1 PWM_3 → IR2110 IC #3 → Half-Bridge #3 → Motor Winding W

Electrical Layout: Avery White



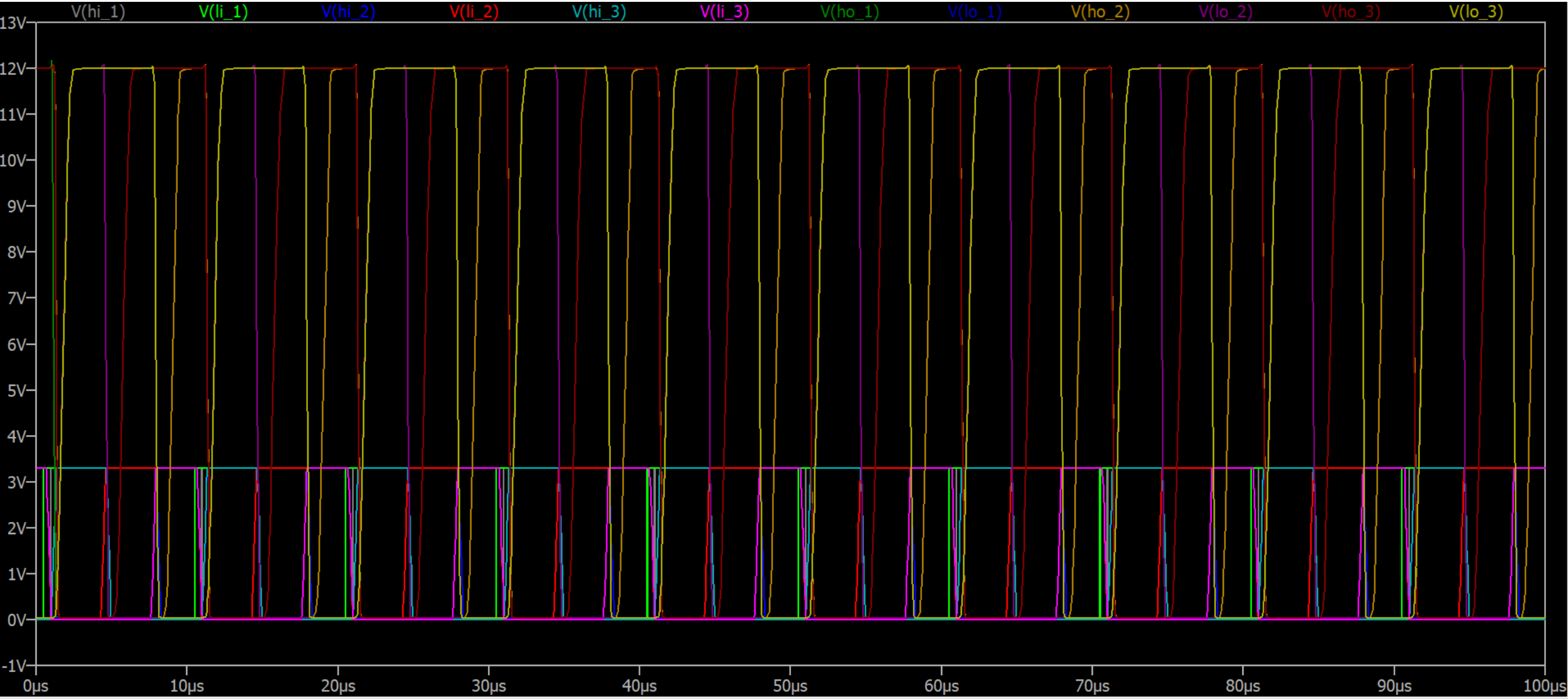


Schematic : Ryan Cramer



SIMULATION BY RYAN CRAMER

Simulation of 3.3V STM32 PWM compared with 12V IR2110 Logic



SIMULATION BY RYAN CRAMER

SVPWM Oscilloscope Demo

https://www.youtube.com/watch?v=36C-fn4Lms8

Captured by Avery White

Software Design

Code written by Avery White, Timer settings configured by Ryan Cramer

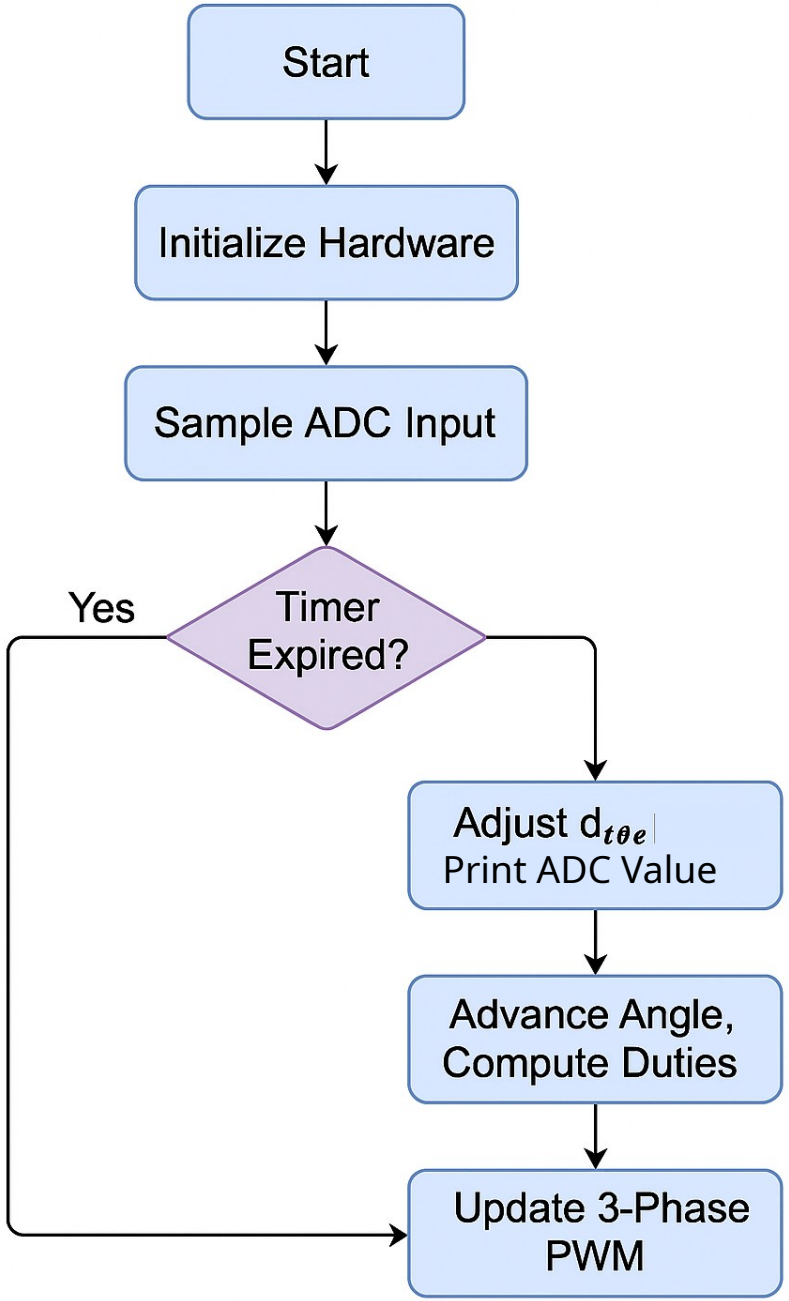
- sample potentiometer input
- Adjusts motor drive characteristics using SVPWM
- Timer1 synchronizes PWM, main loop adjusts PWM and LCD based on the potentiometer input

ComputeSVPWMDuties() → Performs the inverse Park+Clarke Transform

- Converts voltage vector (α, β) into 3-phase voltages (Va, Vb, Vc).
- Normalizes and scales to duty cycles (0–100%).

SetDutyCycle()→

- Reads PWM auto-reload value (ARR)
- Converts duty % into compare values
- Updates compare registers for TIM1 PWM outputs



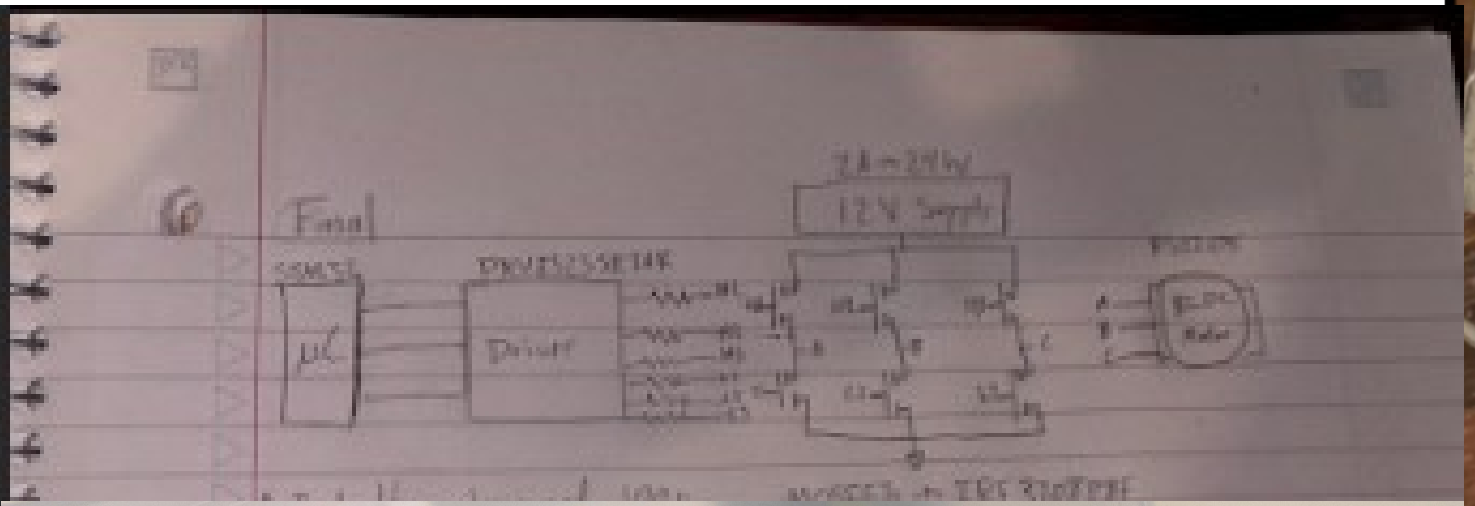
Basic Flow:

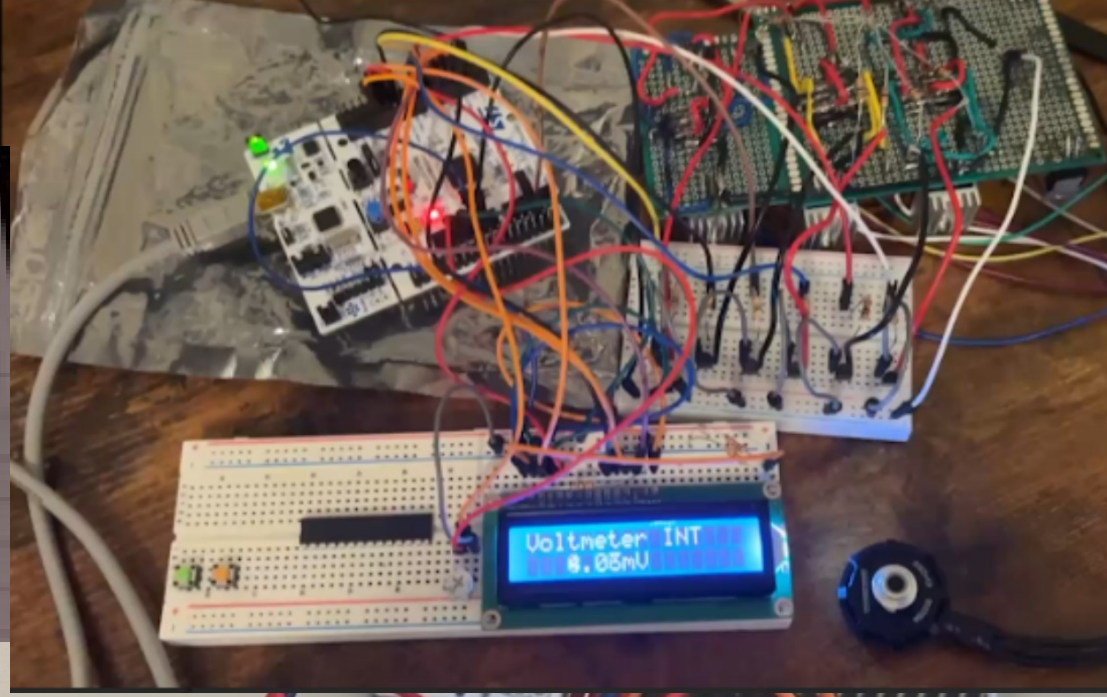
- Initialization before main loop
- In loop, act if TIMER = 0
  - TIMER1 = 1us, synchronizes PWM adjust
  - TIMER2 = 1ms, synchronizes LCD update
- Read ADC Value
- Adjust Angles + Compute Duty Cycle Ratios
- Adjust PWM

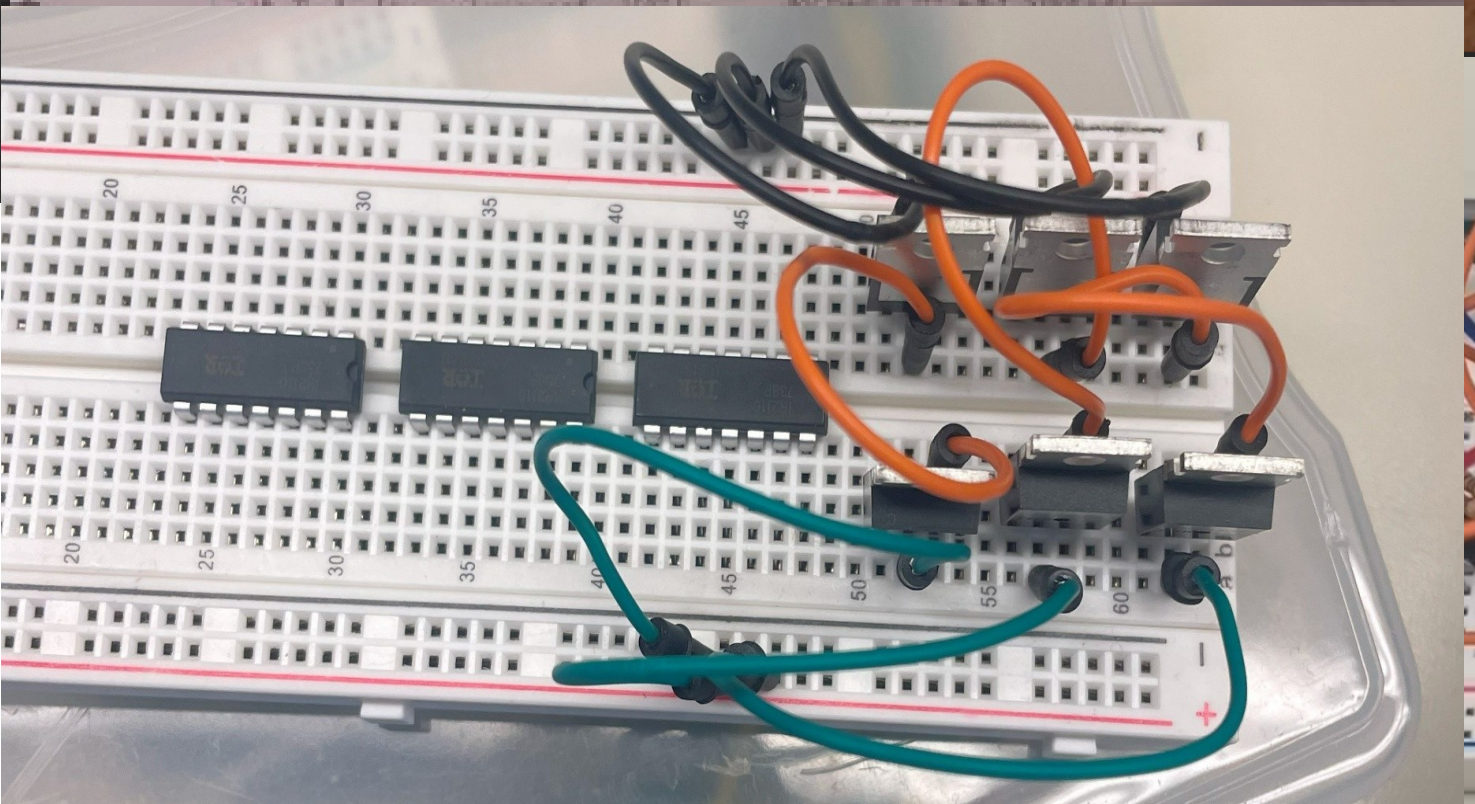
- Angles are based on position of potentiometer, the greater the ADC reading, the stronger the forward (clockwise) rotation vector, but the weaker, the more the PWM duty cycle holds the motor at a specific angle

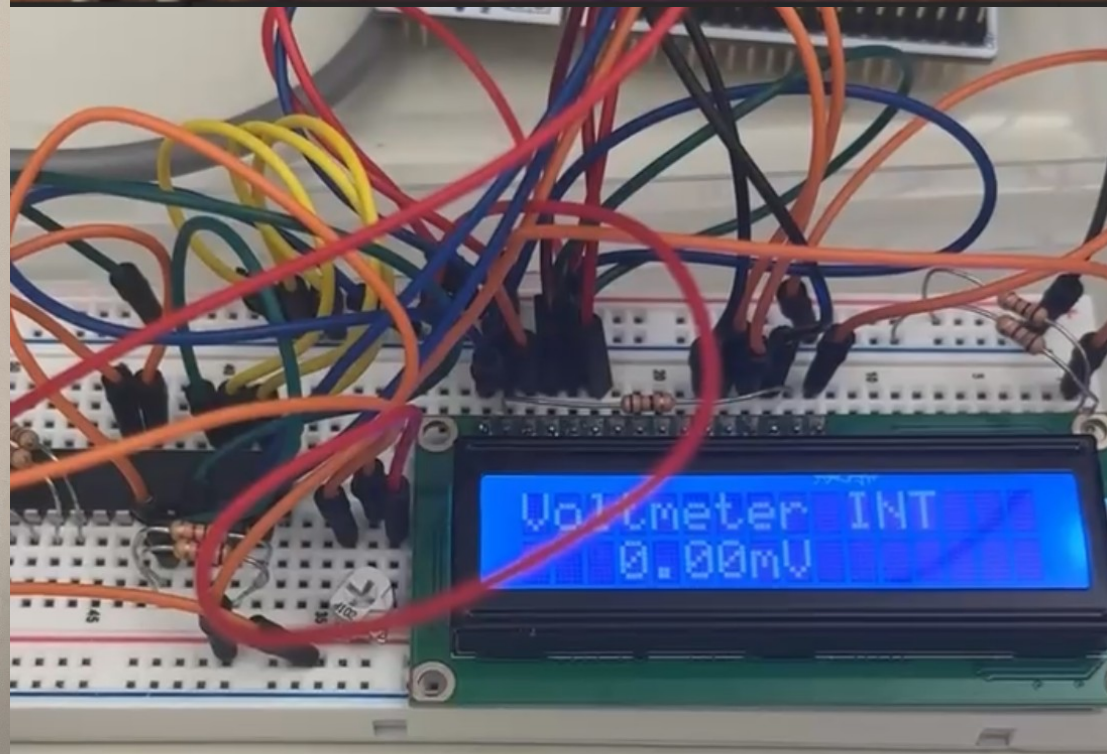
Software written by Avery White

MFG. (Avery)

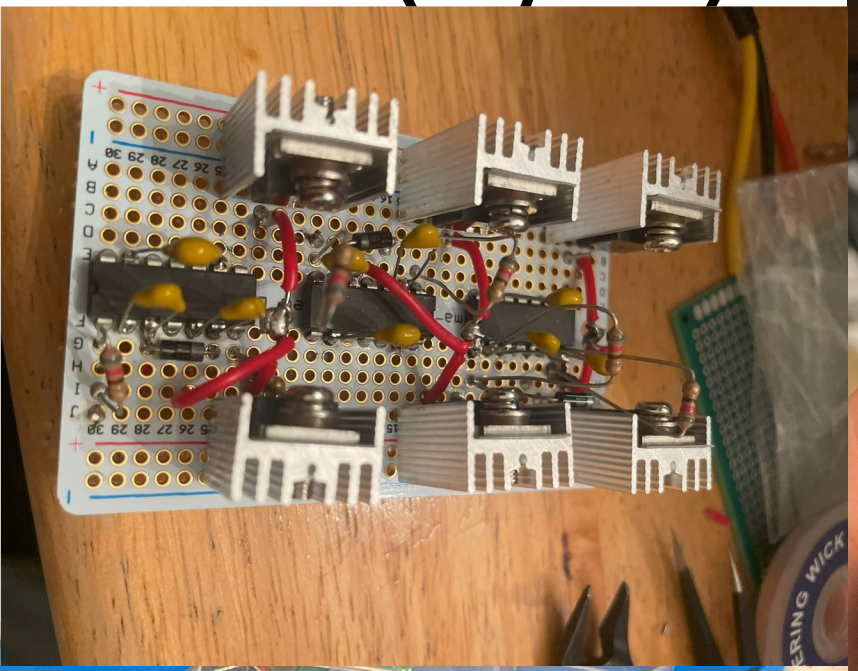


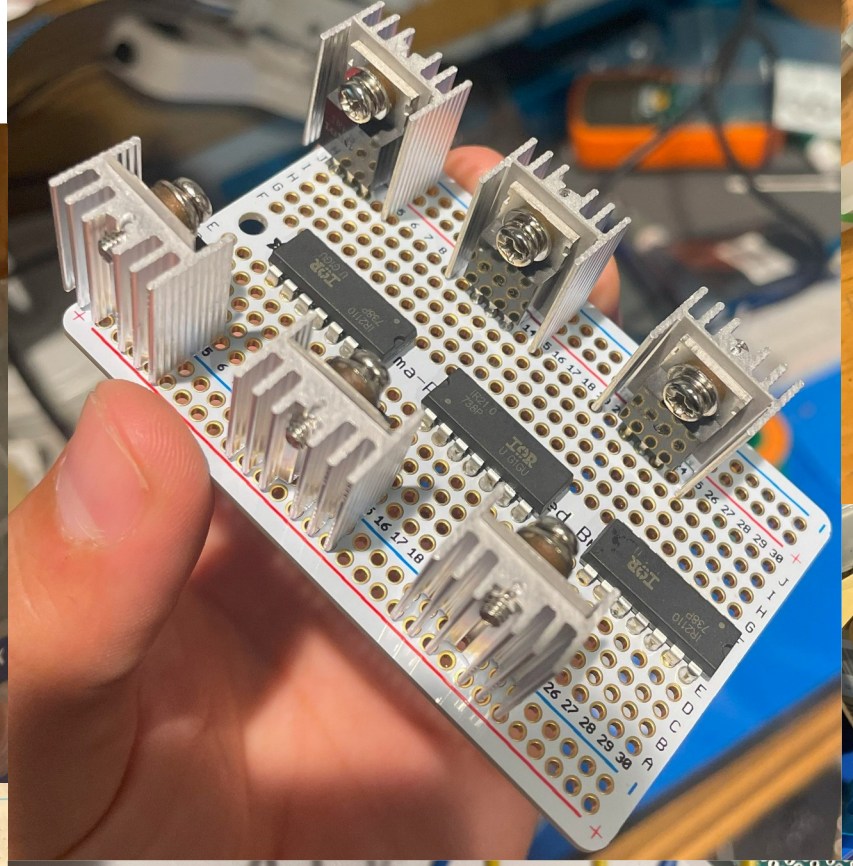


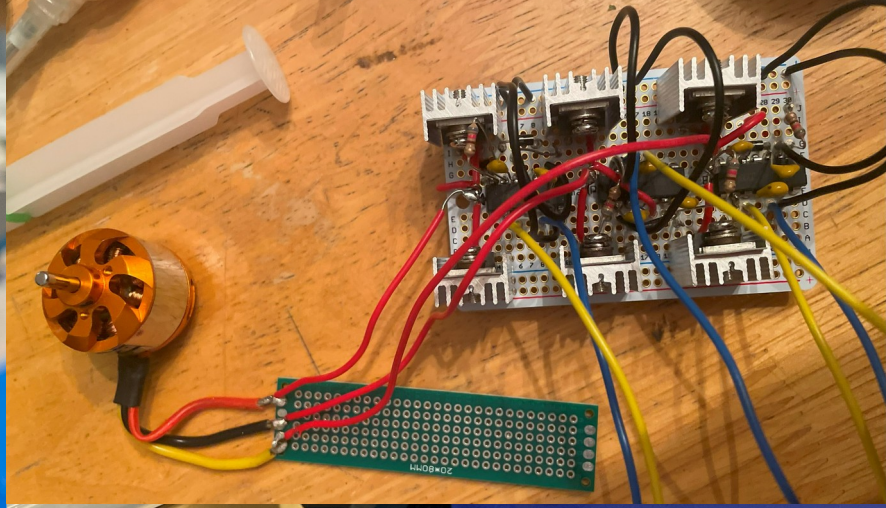


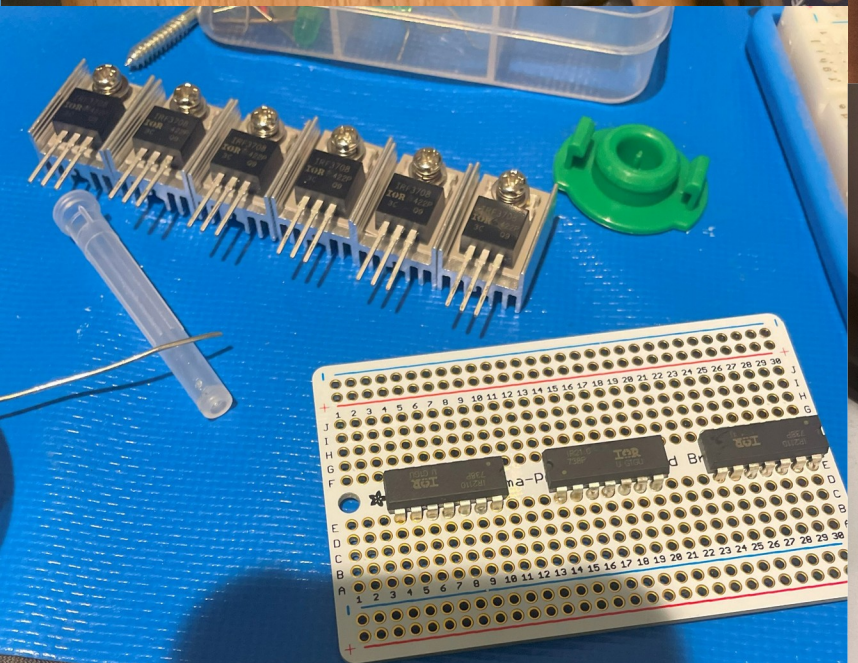


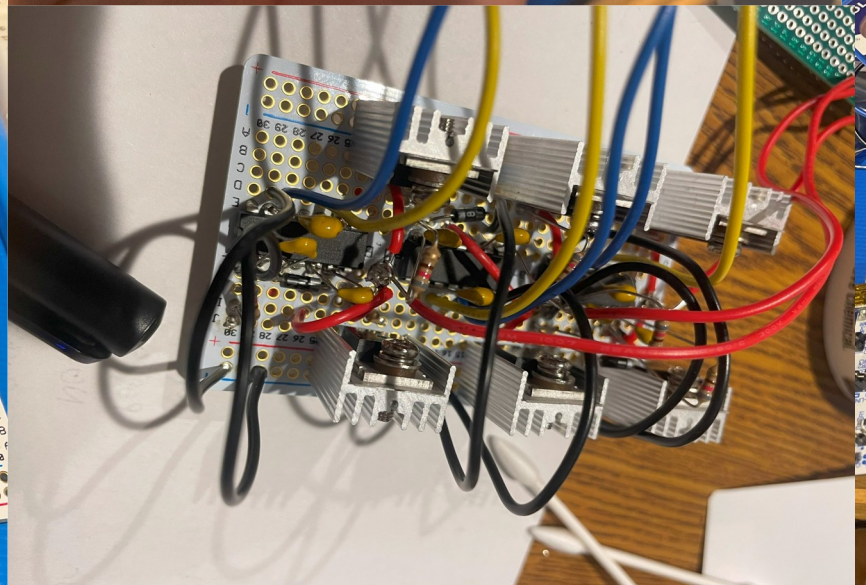
MFG. (Ryan)

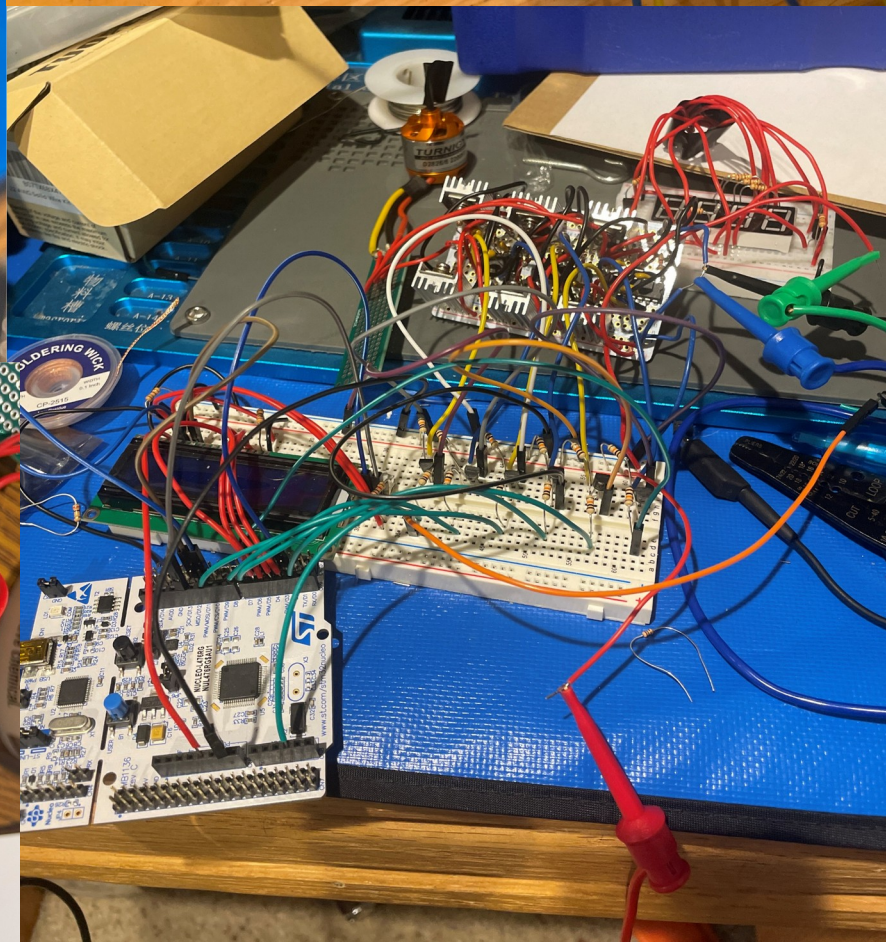












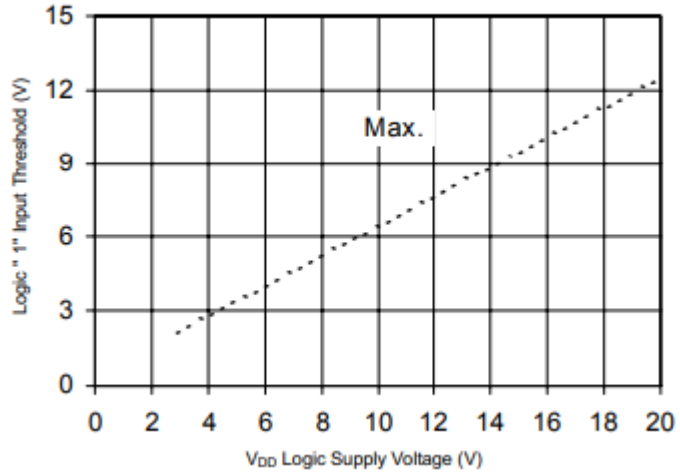
Demo Videos (1.5A, 2.85 A)

https://www.youtube.com/watch?v=5-J6cNFdAwE

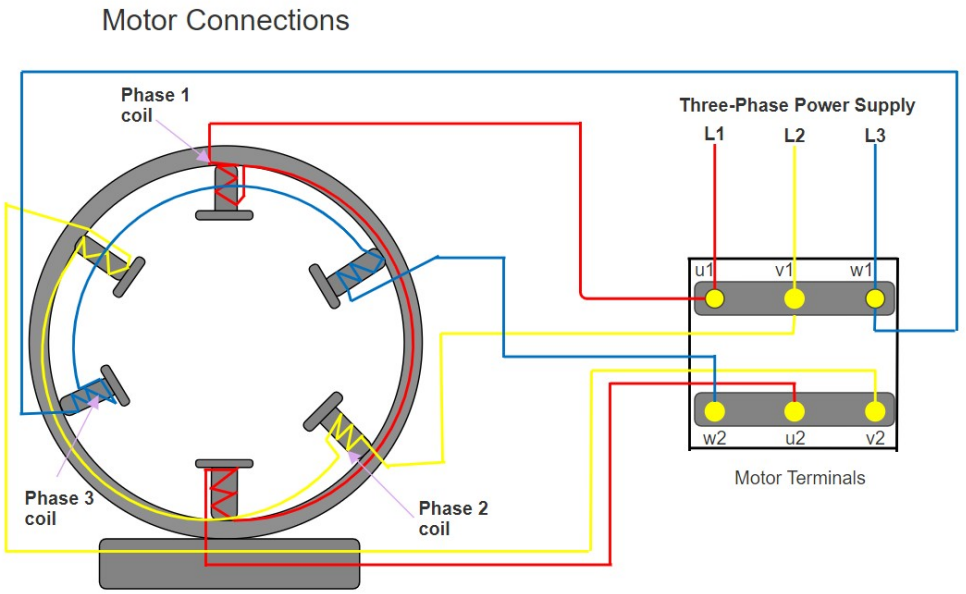
Design Challenges

1. IR2110 Logic Level and MOSFET heat
2. Phase Connections
3. Motor Current
4. Dtheta resolution

IR2110 has varying accepted logic levels based on input VCC. The MOSFETs needed heatsinks due to high frequency



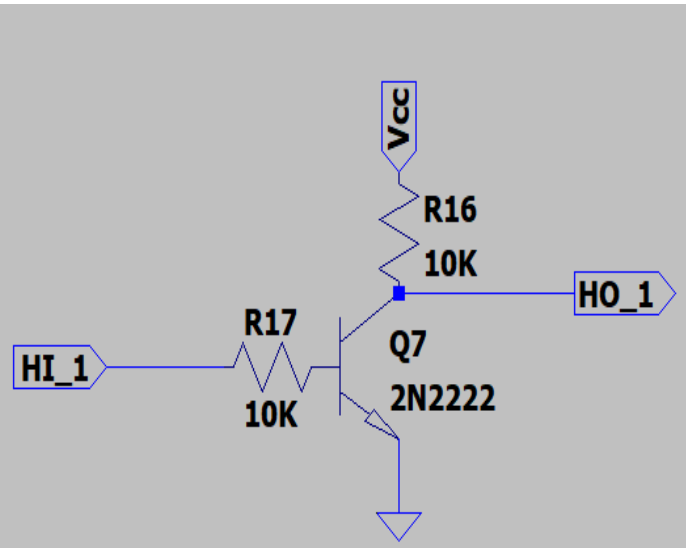
Phase Connections (You CAN hook them up wrong)

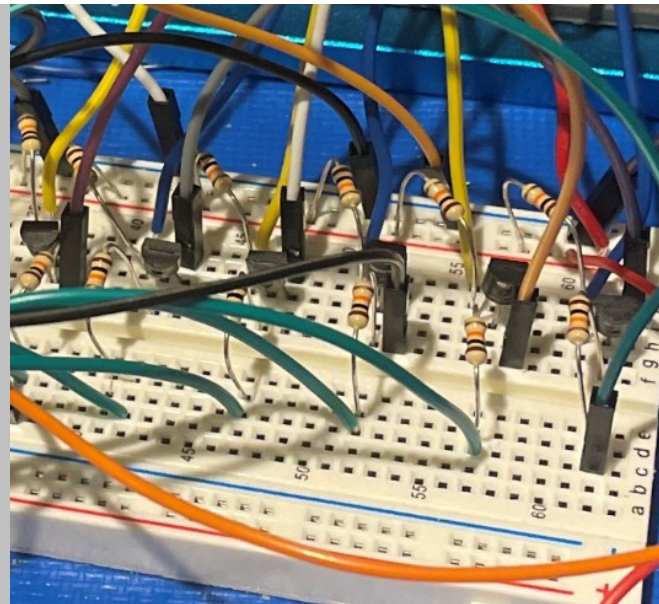


+3.3V is the lowest logic level for IR2110 driver.

Works when the driver has VCC of 7V, not 12V.

FIX = Control Logic w/ NPN





PWM/Floating Point Challenges

The motor originally ran very clunky, would jump to angles, jump backwards, and generally not rotate more than 30 degrees at a time.

This was due to a lack of resolution our PWM. The PWM timer was set to 1ms but we needed much finer control, and changing it to 1us improved it

The dtheta and theta angle were both set using ints, with very low resolution, after coverting them to floating point, we had much finer angle control of the motor

The floating point type caused an issue where it took too long to calculate (multiple floating point calculations), and therefore we precomputed the Inverse Park-Clarke Trasform constants (EX: sqrt(3/2))

Power Challenges

The motor is a 3 Phase BLDC drone motor

- The input is 10A to get 12V
- The average DC power supply is 30V/3A
- We had to test using 3A, which was unsafe due to the 22 AWG wire (max rated 2A). Therefore, the wires got warm during testing, and possibly degraded
- The motor would only take in 1V at 2A, so it barely spun at all
- At 3A, the motor was spinning much faster, with much less jitter

Lessons Learned

- As an engineer, you need to understand the datasheet
  - Things SHOULD NOT be hard to hook up, they are designed to be easy
- Don’t wait last minute to get parts, right now many parts are back ordered- so if its last minute you end up settling for available parts, which may be discontinued / poor quality.
- Know your power requirements before you solder!
- Get CircuitMaker its basically Altium Designer for free
- Try to pick common parts you can easily simulate
- Know your physics! It goes a long way.

Interested in Working on This?

- https://github.com/ryancramuh/SVPWM-STM32-Motor-Project

- We need to incorporate back EMF-Sensing and expand to work with 18 AWG wire that can handle up to 10A.

- Test with through-hole components, verify clean operation under various conditions, and then order a PCB using equivalent SMD components

Q & A