Motor Servoing

John Rushing
G00565091

ECE 370
20 March 2019

Video

https://www.youtube.com/watch?v=02Y5Pd1RX4Y

Github

http://github.com/johnmrushing/ece370servo

Writeup

I did not meet the project specifications.

The H-Bridge and Encoder from previous successful assignments were used, along with the printed mounts. As with the velocity calculations, the motor has much more resistance going backward than forward.

A new dial was printed and attached to the motor mount. A flag for the output shaft was printed, but I could not get the flag onto the shaft without risking damage to either the flag or motor. I still have my debugging flag of masking tape on the shaft.

I initially attempted servoing with the moderated process we discussed in class. I was unable to get this to work after multiple independent attempts. Initially, the kp value was self-adjusting but ramped to NaN or Inf too quickly and for unknown reasons. Then, after constraining Kp, I was unable to correctly interpret the necessary kp values and at what time.

So, I rely on dead reckoning. Serial input from USB was used to read the desired relative angle to rotate to. The rotations of the motor shaft (pre-gearbox) are counted by a single IR sensor and these values are compared against the desired angle after conversion.

Because of the inertia of the encoder and motor, I have to stop short. The values for stopping short were determined experimentally.

The Raspberry Pi was not used.

Test Setup

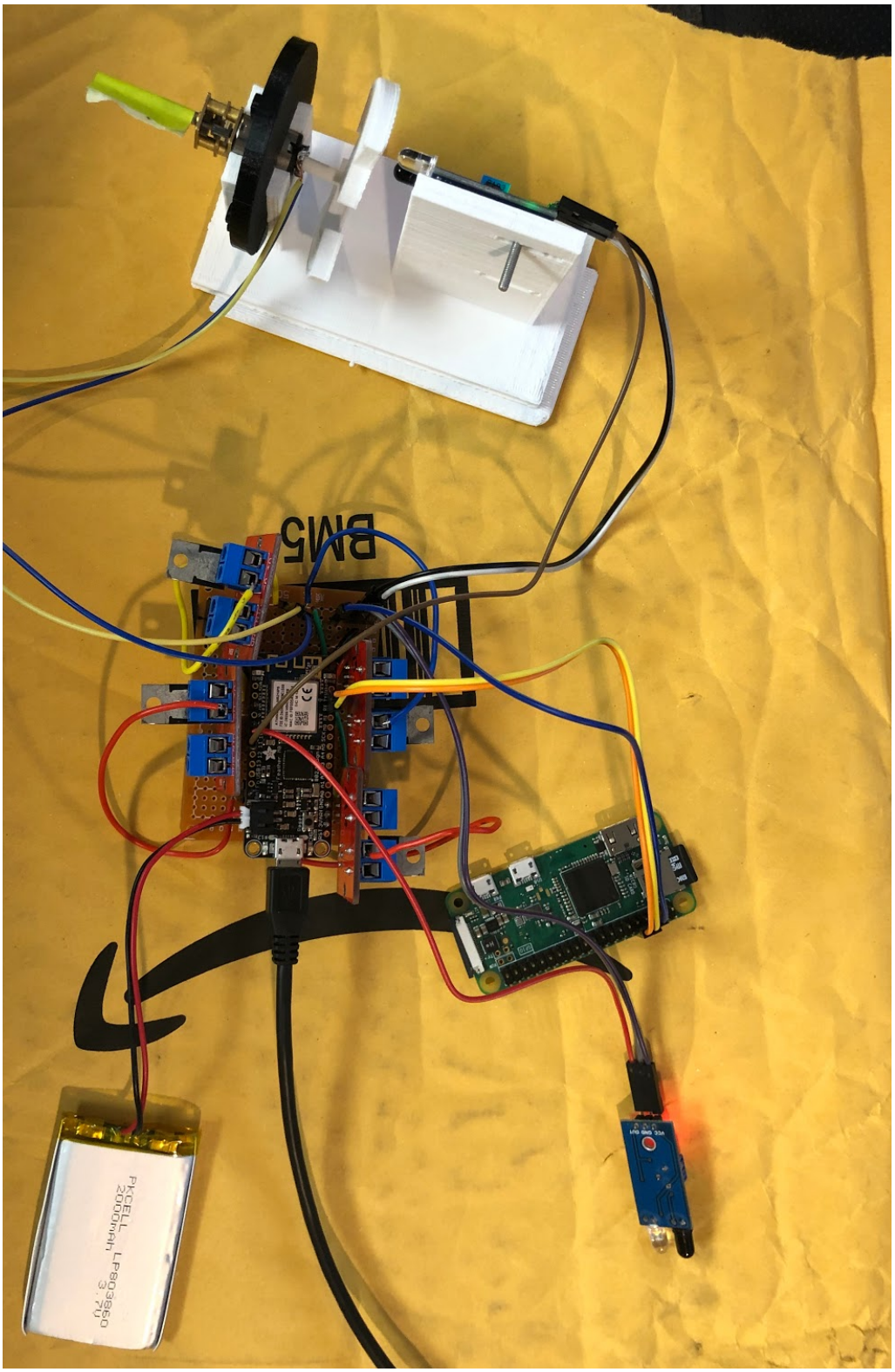

Figure 1 - Test setup

Schematic

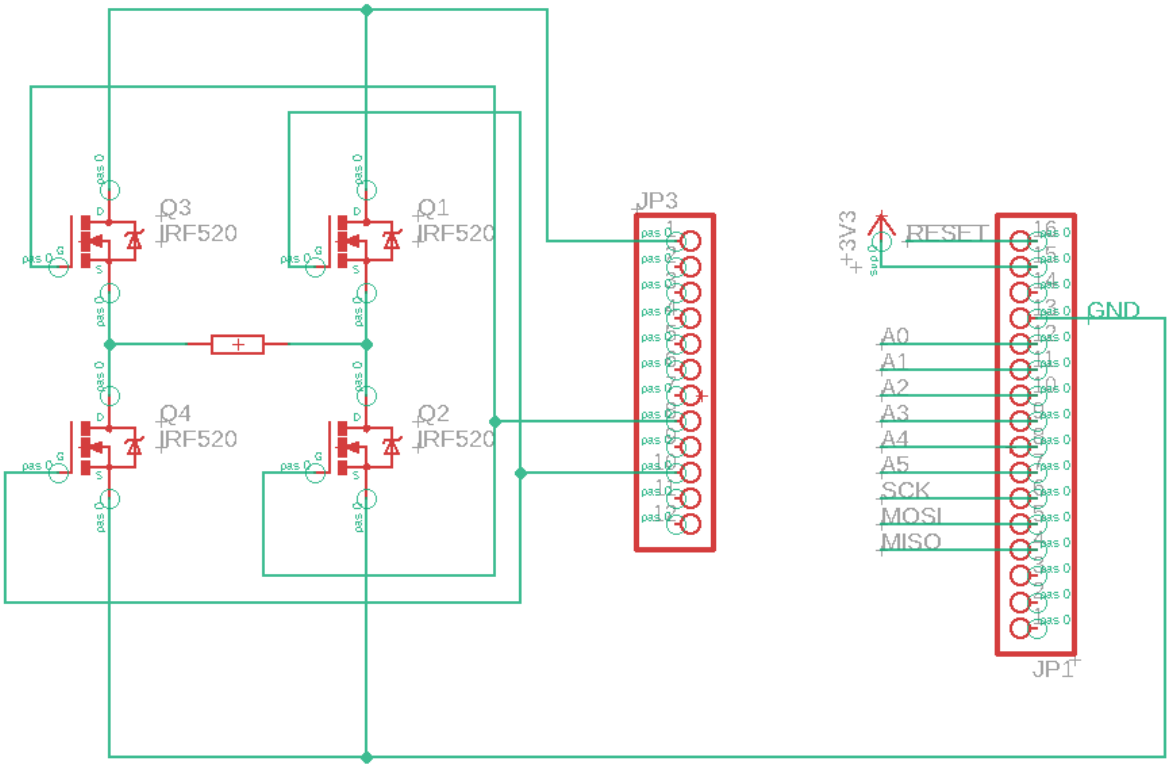

Figure 2 - Test setup schematic to H Bridge and Motor

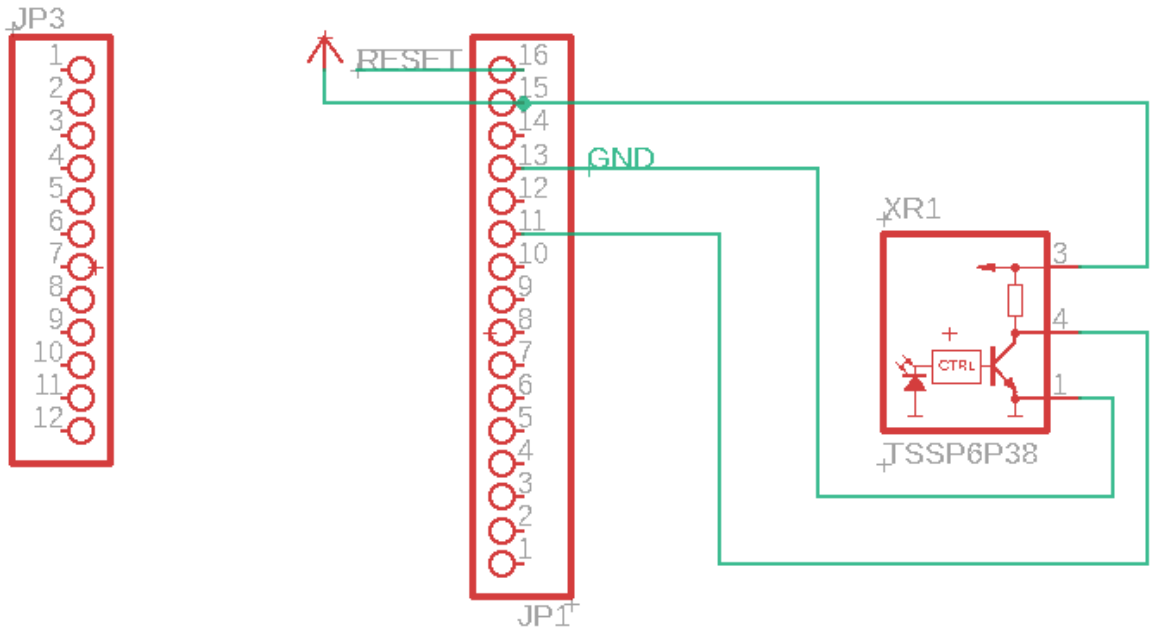

Figure 3 - Test setup schematic to IR Sensor

Pseudocode

```
begin Serial @9600bd

__IR_interrupt {
      increment number of ticks

      if number of ticks is above a threshold
            readjust velocity
            or
            brake and stop

}

loop forever {
      if serial.available
            read serial data
            parse as an float
            pass to servo function
}

servo function {
      determine necessary direction of travel
      set velocity to go in that direction
}

velocity function {
      interpret direction of travel
      interpret pwm dutycycle
      deactivate undesired direction
      activate desired direction at dutycycle
}

braking function {
      set all PWM signals to 0%
}
```