Modular Auditory, Tactile clock

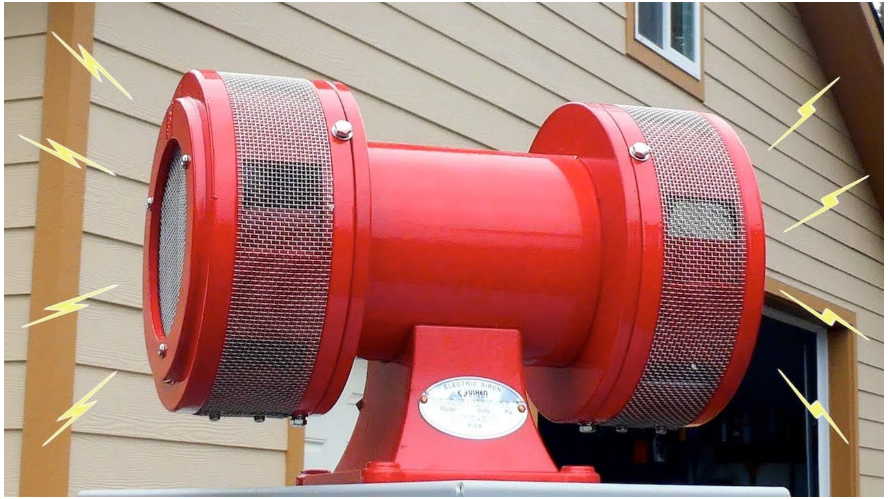
Tahseen Arefeen, Jacob Choi

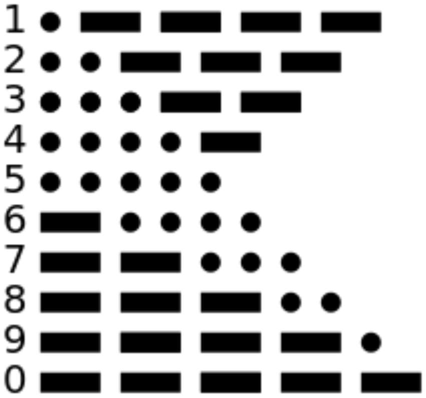
Creativity & Innovation

Two solution approach

- Auditory: air raid siren
- Tactile: morse code

Design must accommodate both designs

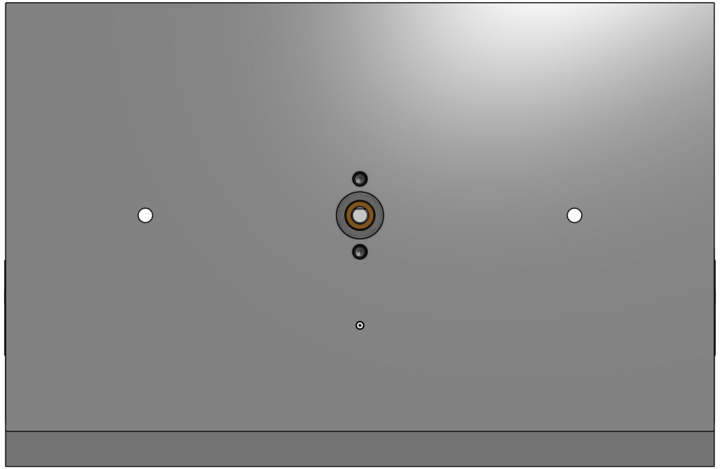


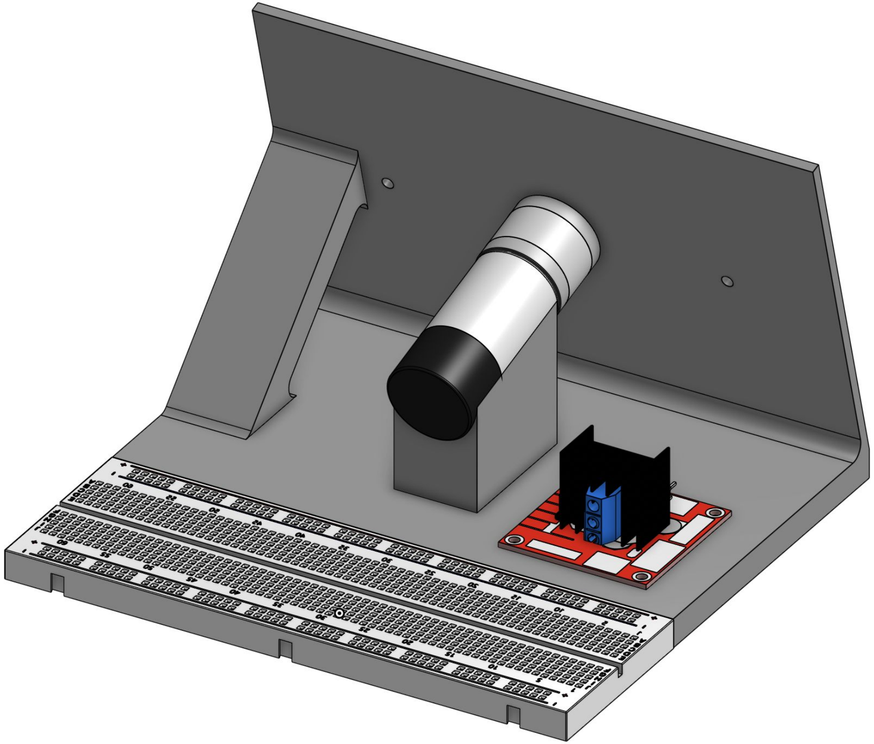


1	● ▬ ▬ ▬ ▬
2	● ● ▬ ▬ ▬
3	● ● ● ▬ ▬
4	● ● ● ● ▬
5	● ● ● ● ●
6	▬ ● ● ● ●
7	▬ ▬ ● ● ●
8	▬ ▬ ▬ ● ●
9	▬ ▬ ▬ ▬ ●
0	▬ ▬ ▬ ▬ ▬

Tahseen Arefeen

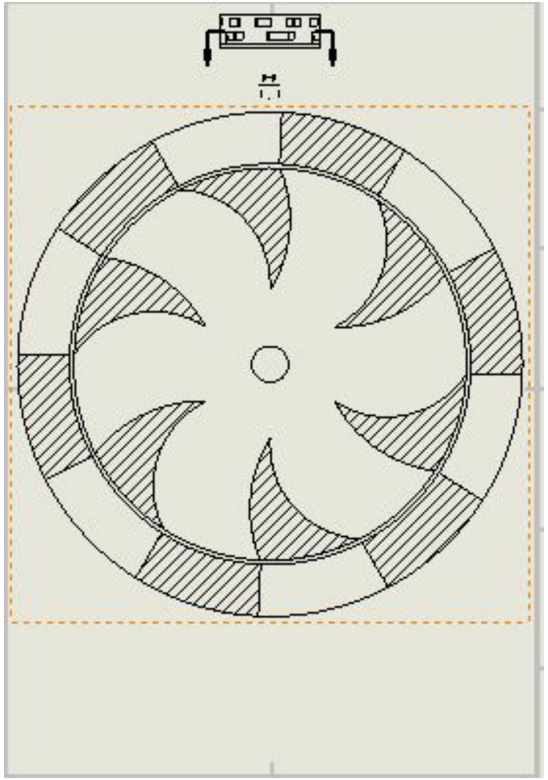
Fixture Design

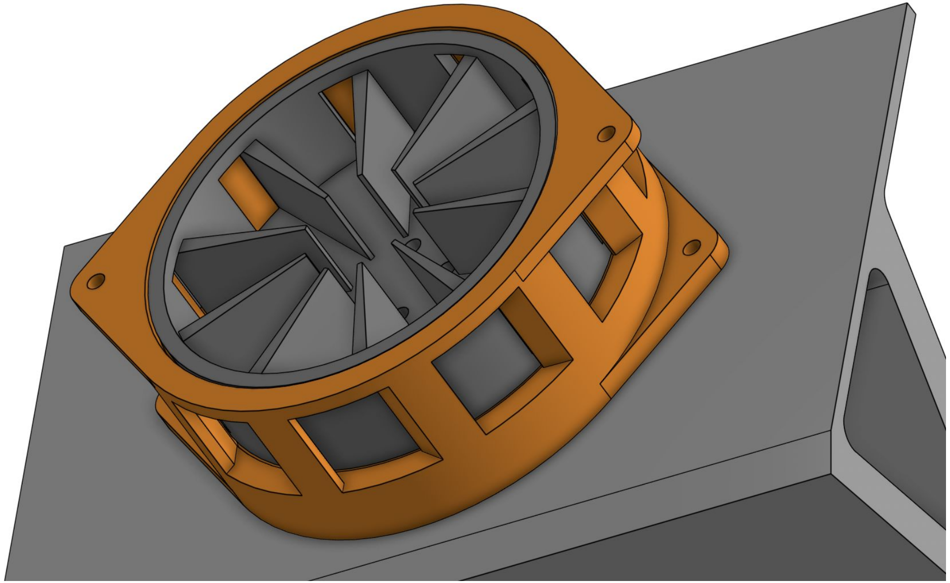




Tahseen Arefeen

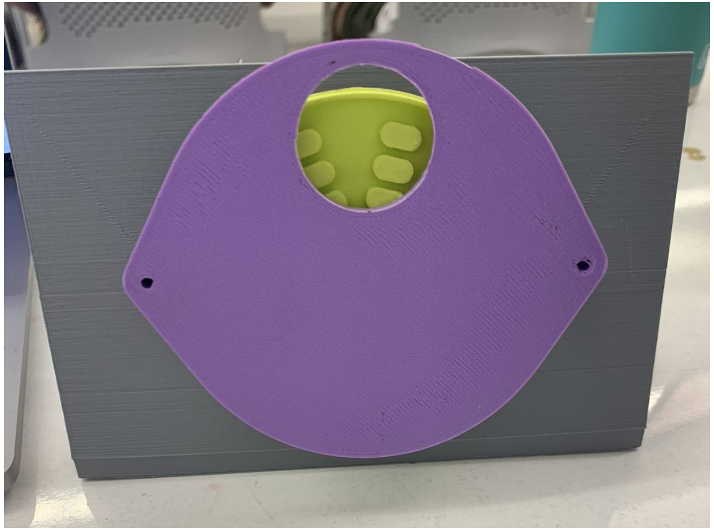
Air Raid Siren Design

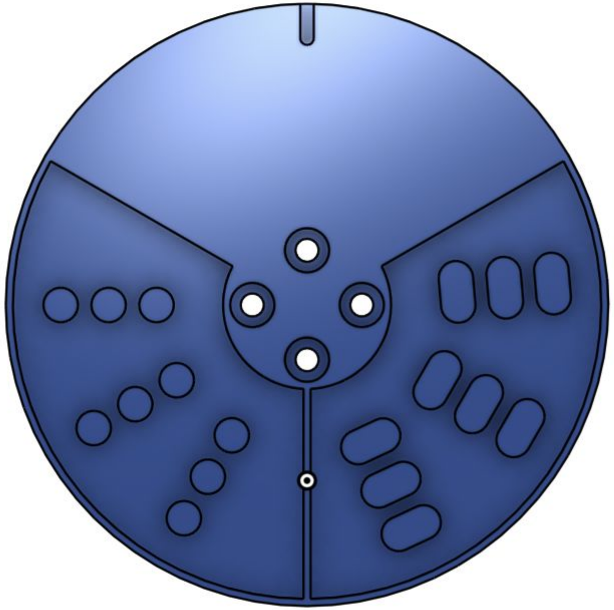




Tahseen Arefeen

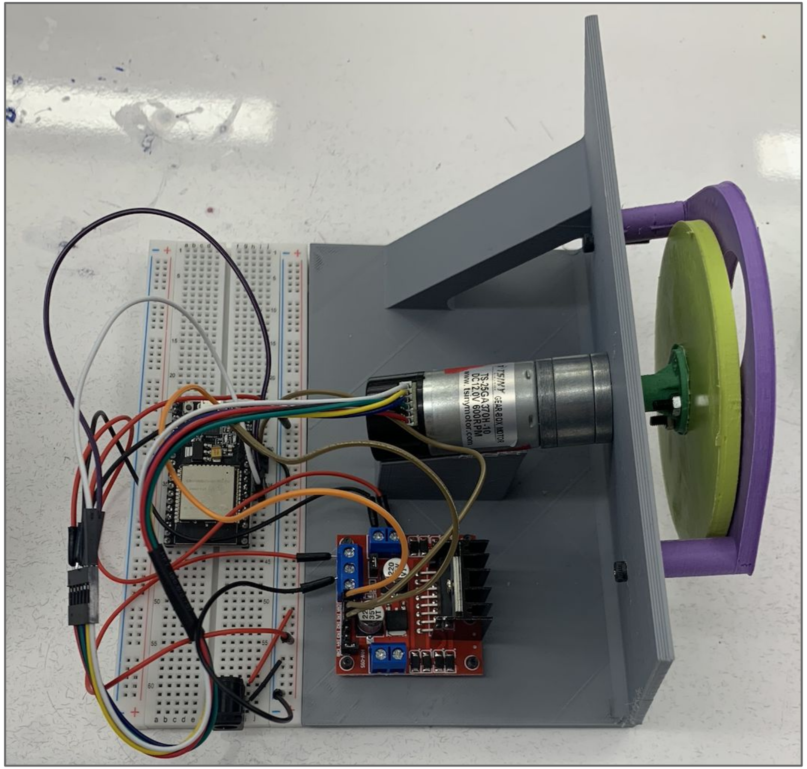
Morse Window Design





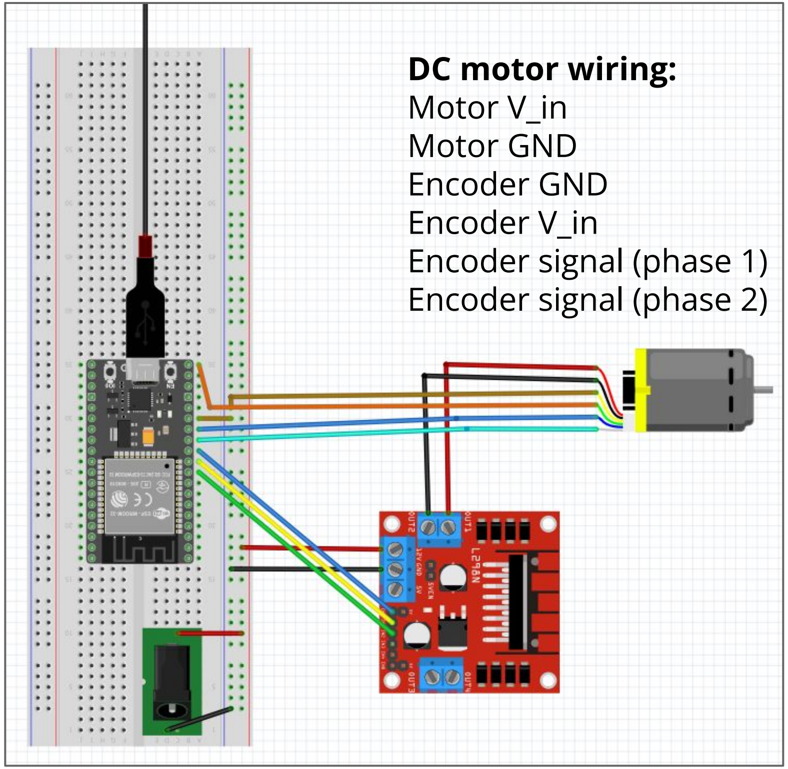
Tahseen Arefeen

Electrical Design | Engineering Reasoning



ESP32 PWM → H-bridge → DC motor speed control

Power line #1: 5V for ESP32, DC encoder



DC motor wiring:
Motor V_in
Motor GND
Encoder GND
Encoder V_in
Encoder signal (phase 1)
Encoder signal (phase 2)

Power line #2: 12V for DC motor, H-bridge (separate power supplies for microcontrollers and peripherals vs. noisy motors)

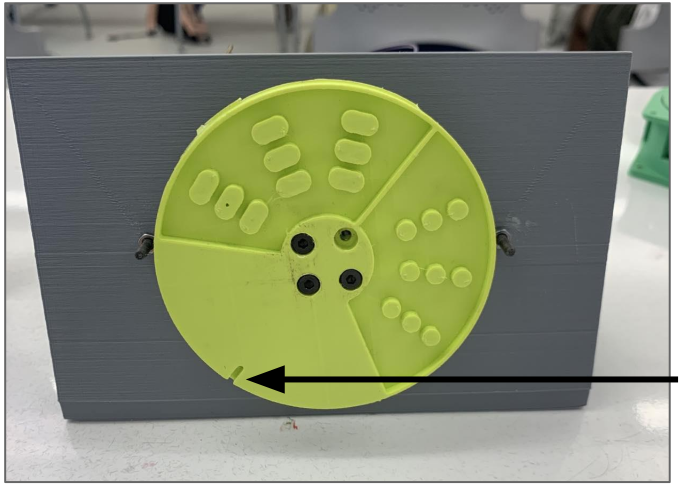
Jacob Choi

Software Design | Engineering Reasoning

Overview:

1. Initialize current time
2. Update time based on desired interval
3. Translate current time to Morse code
4. Actuate based on dot vs. dash
5. Obtain encoder positional feedback
6. Adjust for error based on threshold

ENCODER_THRESHOLD: max. allowable encoder delta before more than 1 section appears in window view



Divot to indicate "origin",
i.e, encoder value = 0

Math:

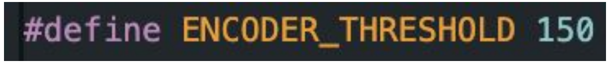
360 rotation° = ~1100 delta encoder reading

Each section is ⅓ = ~367 delta

Origin is halfway ∴ 367/2 = 183.5 delta

ENCODER_THRESHOLD = 183.5

In reality, threshold must be more controlled due to motor overshoot (100 ~ 150)



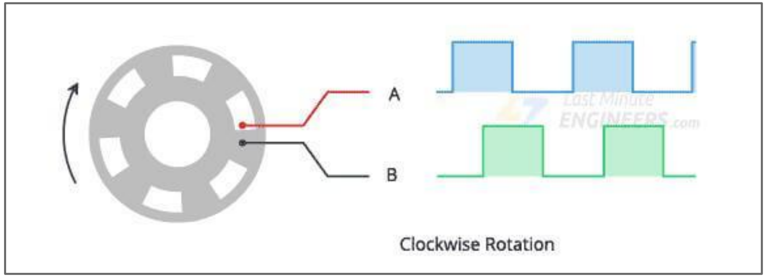
```
#define ENCODER_THRESHOLD 150
```

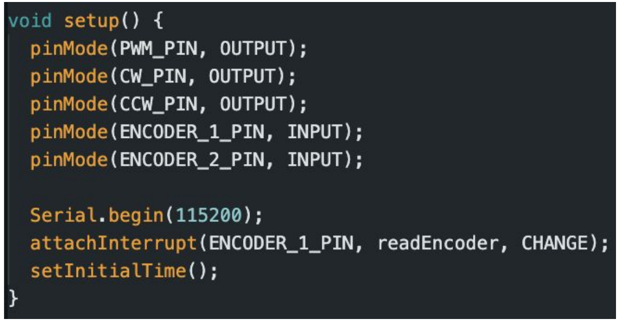
Jacob Choi

Software Design | Engineering Reasoning

2 phase encoder (clockwise): A goes HIGH, B is LOW. A goes LOW, B is HIGH.

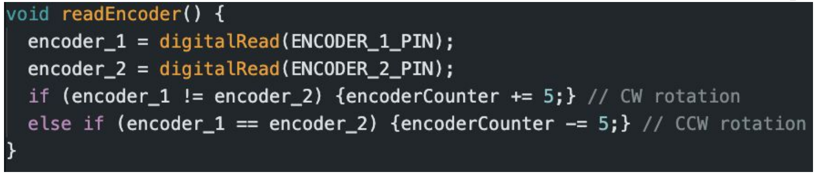
Source





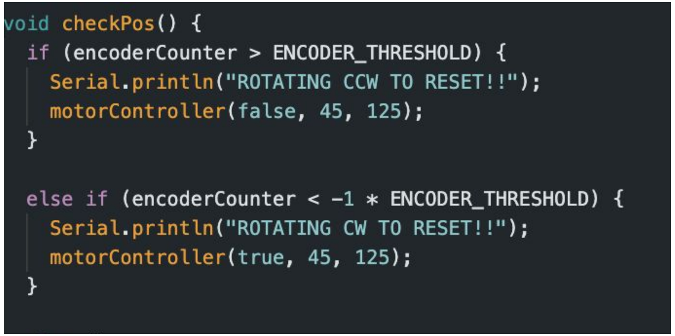
```
void setup() {
  pinMode(PWM_PIN, OUTPUT);
  pinMode(CW_PIN, OUTPUT);
  pinMode(CCW_PIN, OUTPUT);
  pinMode(ENCODER_1_PIN, INPUT);
  pinMode(ENCODER_2_PIN, INPUT);

  Serial.begin(115200);
  attachInterrupt(ENCODER_1_PIN, readEncoder, CHANGE);
  setInitialTime();
}
```

When encoder A changes HIGH/LOW, call the interrupt function readEncoder()



```
void readEncoder() {
  encoder_1 = digitalRead(ENCODER_1_PIN);
  encoder_2 = digitalRead(ENCODER_2_PIN);
  if (encoder_1 != encoder_2) {encoderCounter += 5;} // CW rotation
  else if (encoder_1 == encoder_2) {encoderCounter -= 5;} // CCW rotation
}
```

Positional feedback: “balances” encoder error by spinning oppositely

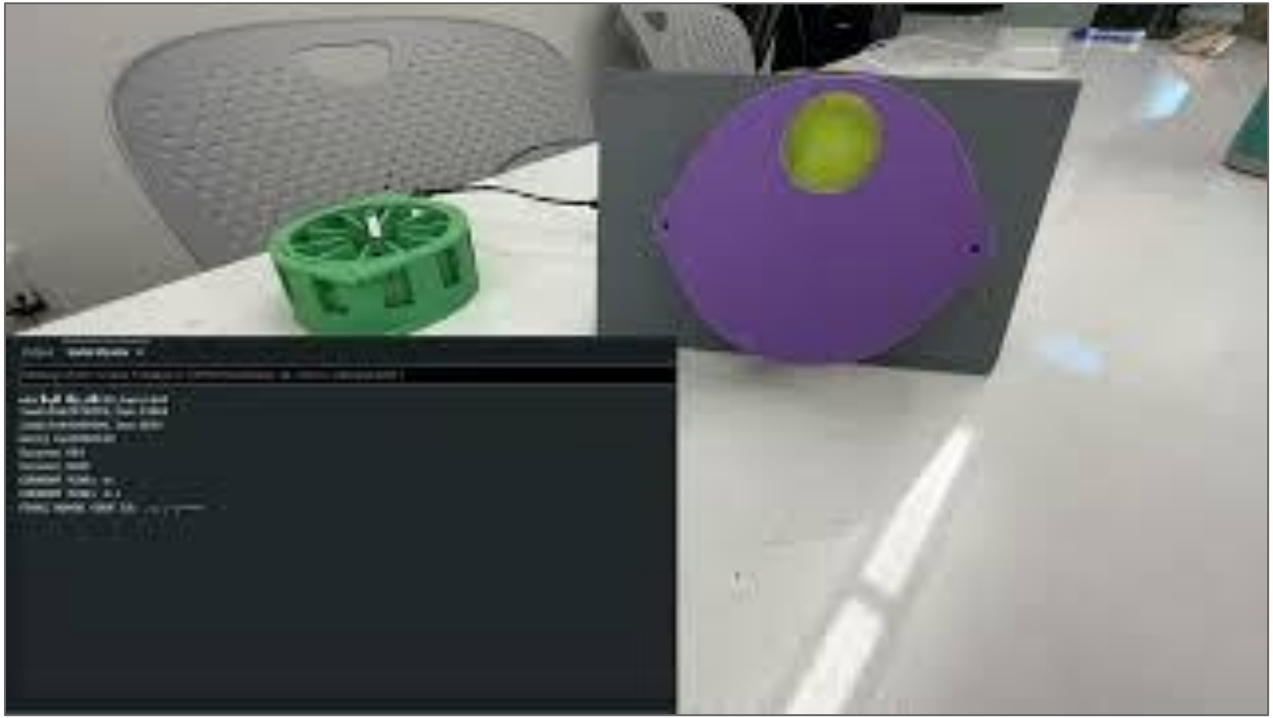


```
void checkPos() {
  if (encoderCounter > ENCODER_THRESHOLD) {
    Serial.println("ROTATING CCW TO RESET!!");
    motorController(false, 45, 125);
  }

  else if (encoderCounter < -1 * ENCODER_THRESHOLD) {
    Serial.println("ROTATING CW TO RESET!!");
    motorController(true, 45, 125);
  }
```

Limitations: arbitrary and uncontrolled spin and only checks encoder error *once*

Jacob Choi

Demo Video



Notes:

- Correct morse code (....- .----) is shown!
- The positional feedback system can be seen working at 0:19 and 0:22 (see serial output and the motor attempting to restabilize to origin)

Thank You!