Questions? Contact jeff@jvon.org or michael@jvon.org

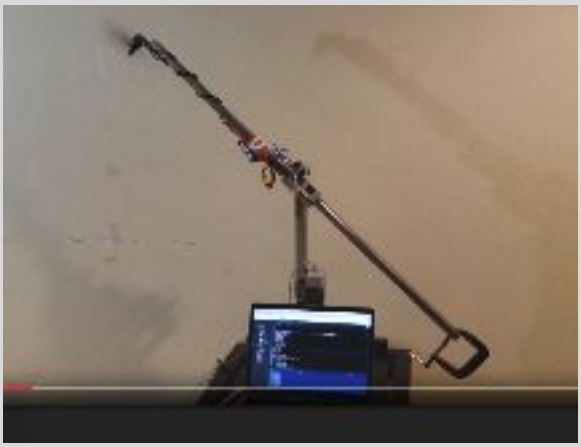
PID Controller Challenge

Challenge: create a PID controller that can dynamically balance a lever using a single quadcopter motor and rotor blade.

The controller should adjust the throttle setting to seek a level attitude, and should self-adjust for small changes in the counterweight (eg, rotating the C-clamp).

The control program should be based on a feedback loop that relies on a proportional-integral-derivative (PID) control algorithm (and some tuning).

https://youtu.be/MPQ2g6QYLtQ



Feedback Loop

- Action == change the throttle setting
- Reaction == lever arm moves
- Modification == calculate a new setting

The modification is based on PID control.

PID Control

A PID control loops measures an amount of error, then adjusts its output by adding three values, in proportion to:

if $error = actual\ angle - target\ angle$ then:

1. **P** == the current magnitude of error
2. **I** == the cumulative error over time
3. **D** == the rate of change of the error

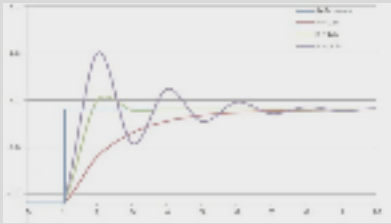
```
14
15  # totally goofy pseudo-code for PID
16  while True:
17      angle = measure_angle_of_lever_arm()
18      error = target_angle - angle
19
20      p = error
21      i += error * timespan
22      d = error / timespan
23
24      throttle = (kp * p) + (ki * i) + (kd * d)
25      time.sleep(for_a_little_while)
26
```

Tuning

The controller is tuned by combining the three values P, I, D in fixed proportions:

$$throttle = (kp * P) + (ki * I) + (kd * D)$$

Try different settings for kp, ki, and kd...



History, and Math

Early PID control loops were developed to govern steam engines and steer ships. Those early efforts were based more on intuition and less on math, but the math is just simple calculus.

$$P(t) = actual\ angle_t - target\ angle_t$$

$$I = \int_0^t P(t)\,dt$$

$$D = \frac{dP(t)}{dt}$$