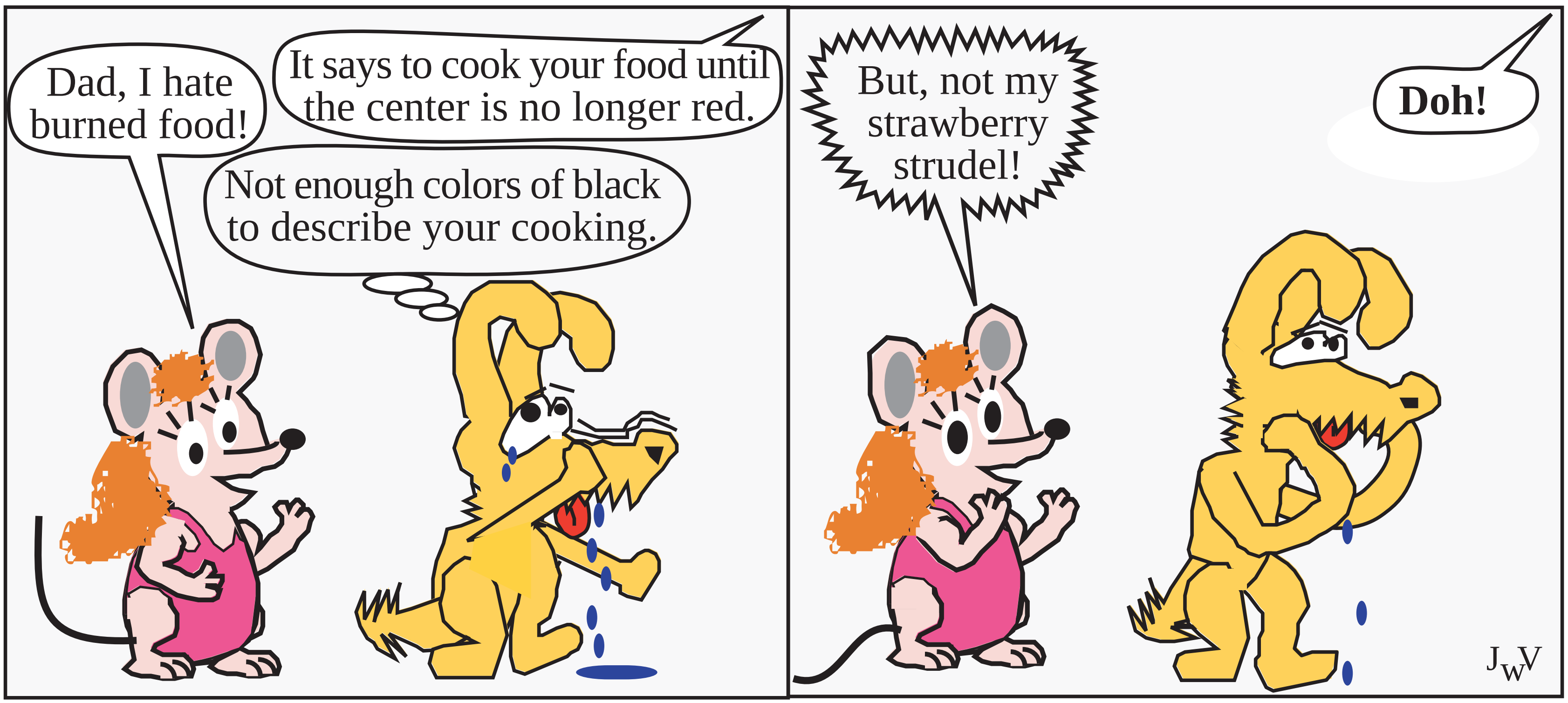
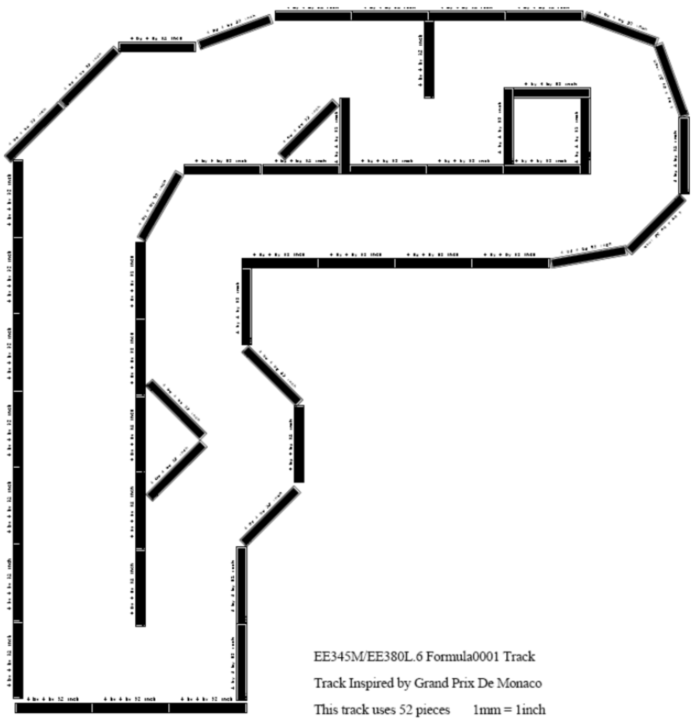
21. Control Systems

- **Easy: Incremental**
- **Linear: P PI PID**
- **Intuitive: Fuzzy Logic**

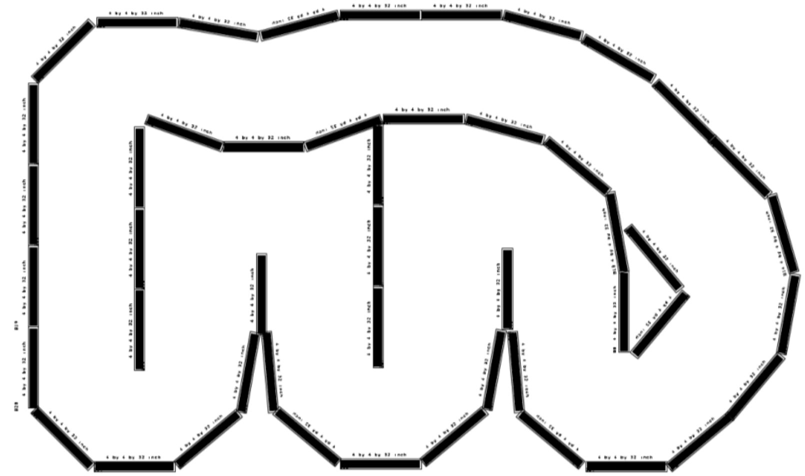


Monaco



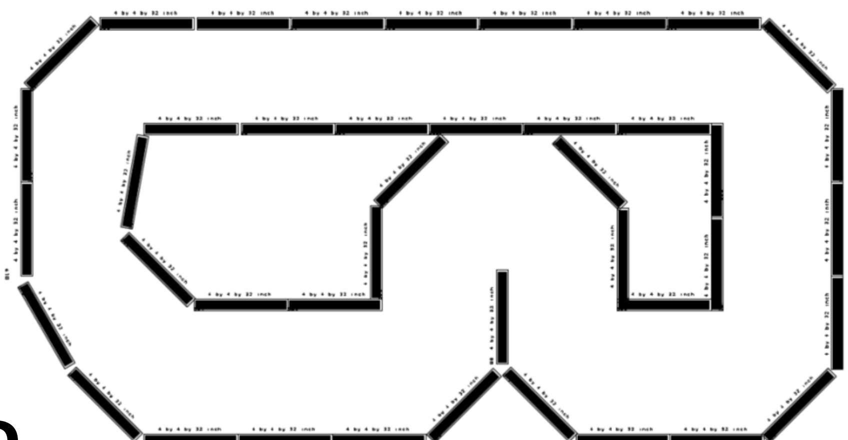
EE345M/EE380L.6 Formula0001 Track
Track Inspired by Grand Prix De Monaco
This track uses 52 pieces   1mm = 1inch

Von Deutschland



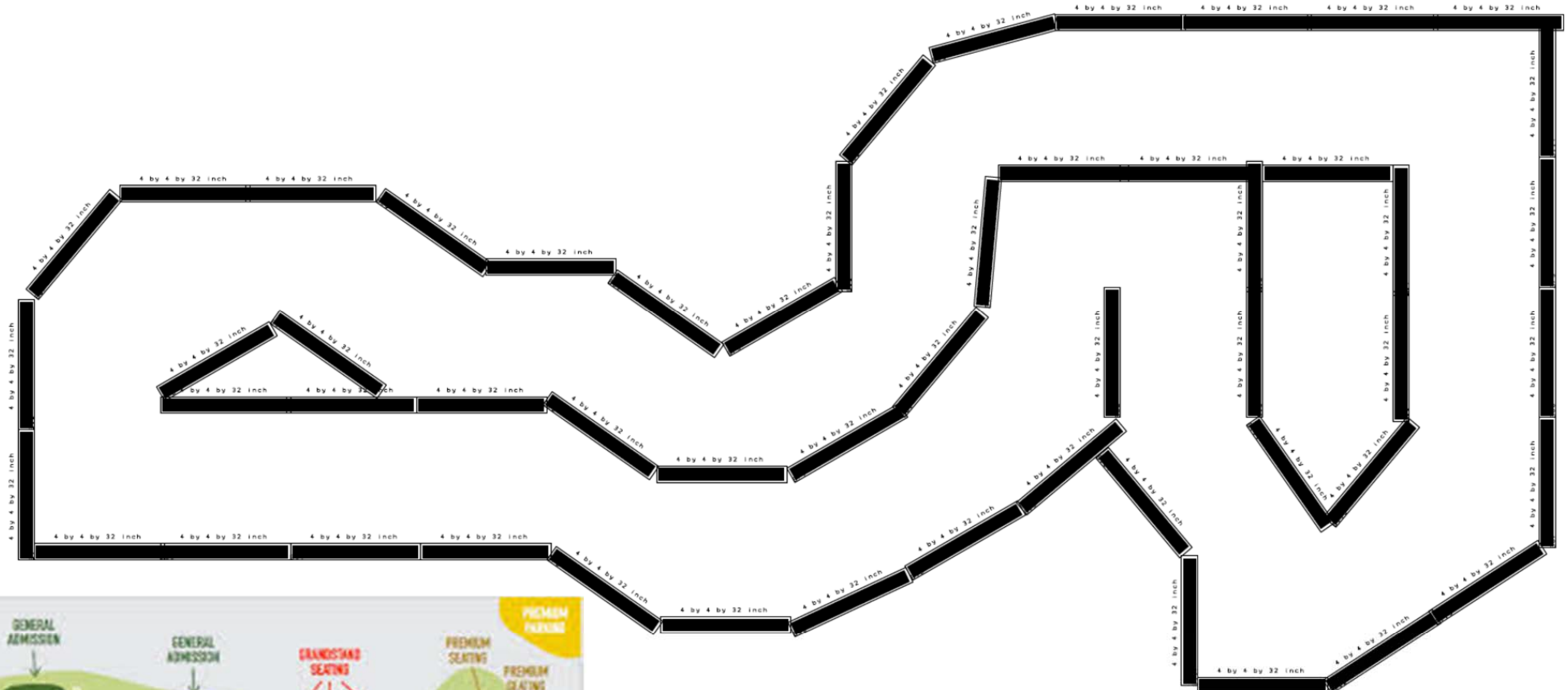
EE345M/EE380L.6 Formula0001 Track
Track Inspired by FORMULA 1 GROSSER PREIS SANTANDER VON DEUTSCHLAND

Espana



EE345M/EE380L.6 Formula0001 Track
Track Inspired by FORMULA 1 GRAN PREMIO DE ESPANA TELEFÓNICA

Circuit of the Americas

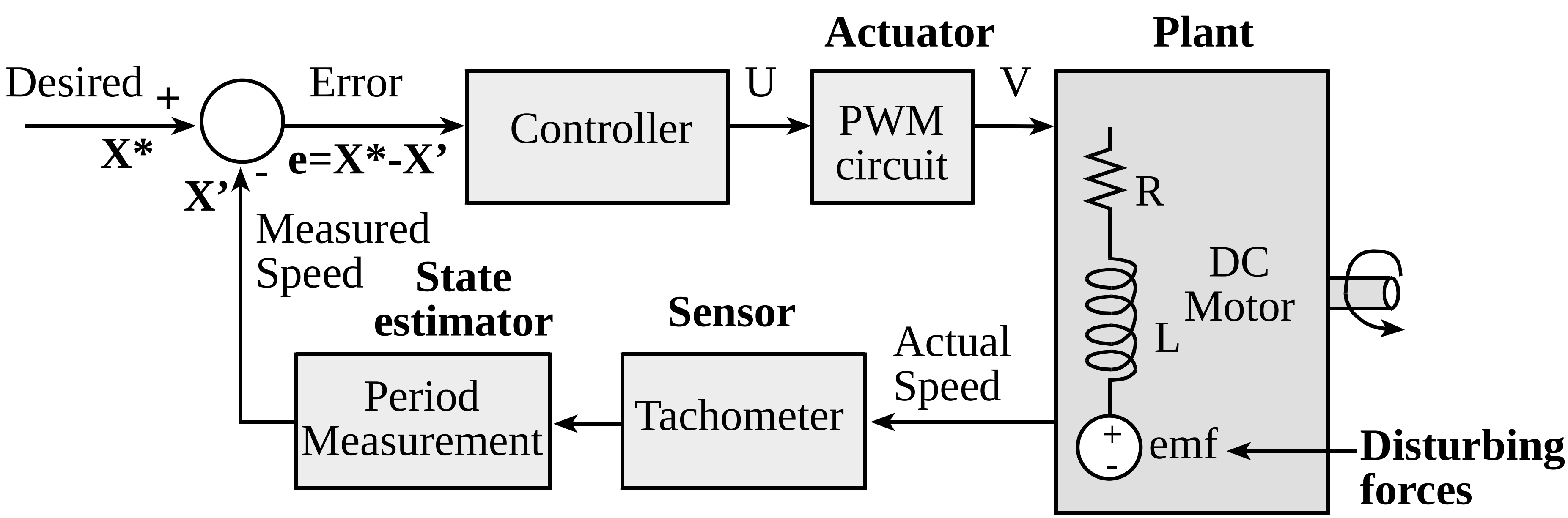


Performance measures

- Accuracy
  - Magnitude of the Error = Desired– Actual
- Stability
  - No oscillations
- Overshoot (underdamped, overdamped)
  - Ringing, slow
- Response Time to new steady state after
  - Change in desired setpoint
  - Change in load

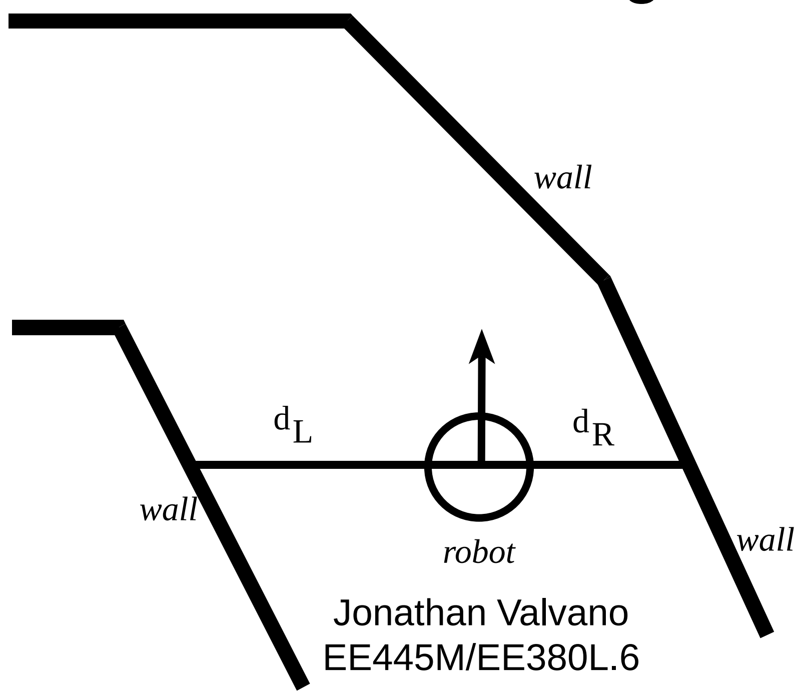
What to control?

- Motor speed
  – Error = Desired speed – Actual speed
  – Error = Desired period – Actual period



What to control?

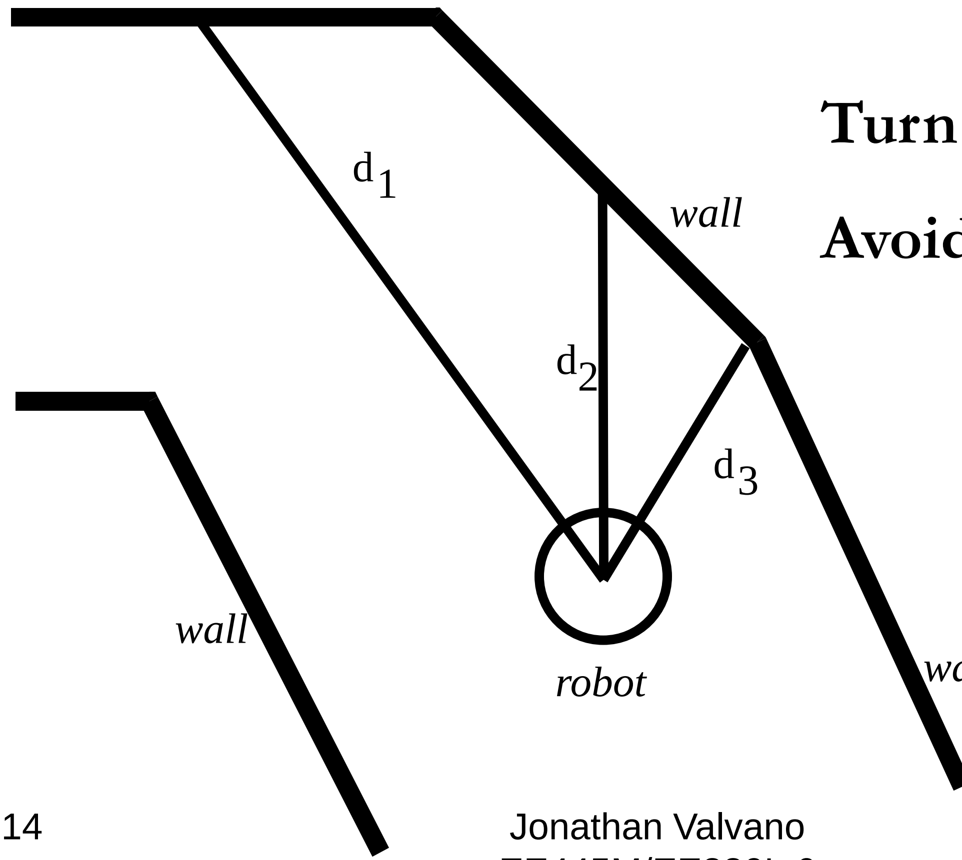
- Distance to one wall
  – Error = Desired distance – Actual distance
- Equal distances to both walls
  – Error = Left distance – Right distance



Sensors
- **Time constant -> f_s**
- **Filters**
  Analog, digital
- **Signal/noise ratio**
- **Calibration**
  accuracy, drift

What to control?

- Angle to most open track
  - Error = Desired – Actual angle



Turn so most open in front

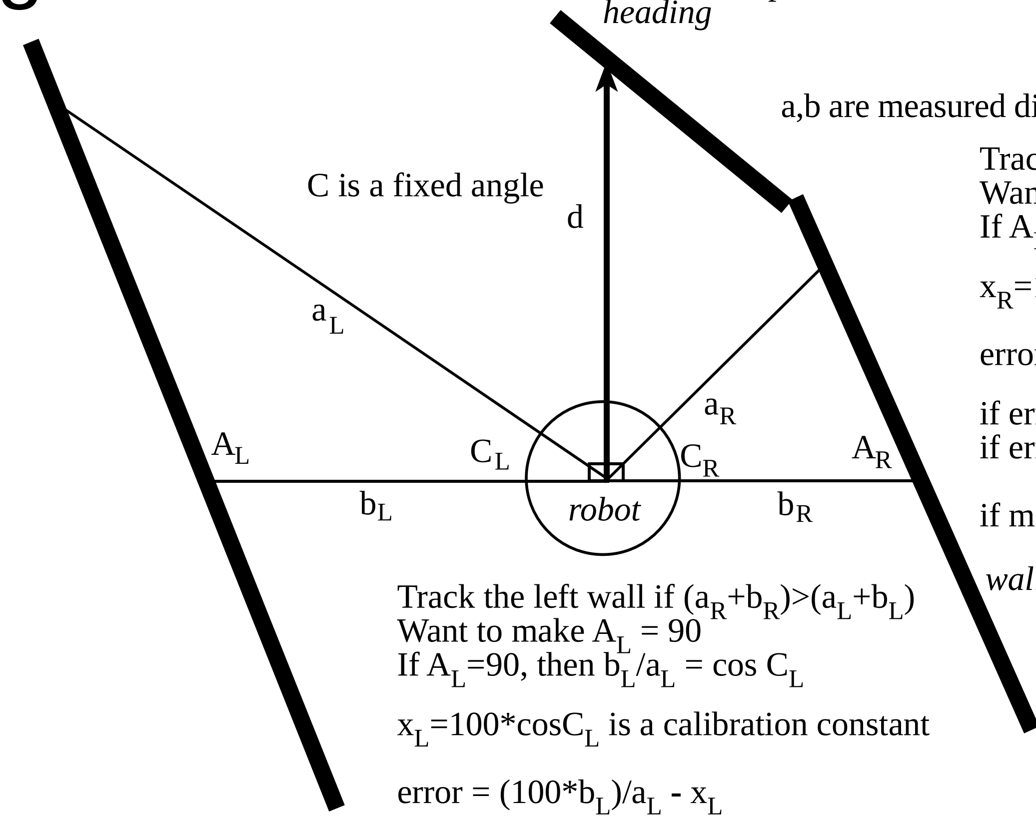
Avoid making U-turns

What to control?

- Angle to wall

speed is determined by distance,d, to object in front

a,b are measured distances from center of robot to the wall



Track the right wall if $(a_R+b_R)<(a_L+b_L)$
Want to make $A_R = 90$
If $A_R=90$, then $b_R/a_R = \cos C_R$

$x_R=100*\cos C_R$ is a calibration constant

error = $(100*b_R)/a_R - x_R$

if error > 0, $A_R < 90$, turn left
if error < 0, $A_R > 90$, turn right

if $\min(a_R,b_R)<20$, move left-right

Track the left wall if $(a_R+b_R)>(a_L+b_L)$
Want to make $A_L = 90$
If $A_L=90$, then $b_L/a_L = \cos C_L$

$x_L=100*\cos C_L$ is a calibration constant

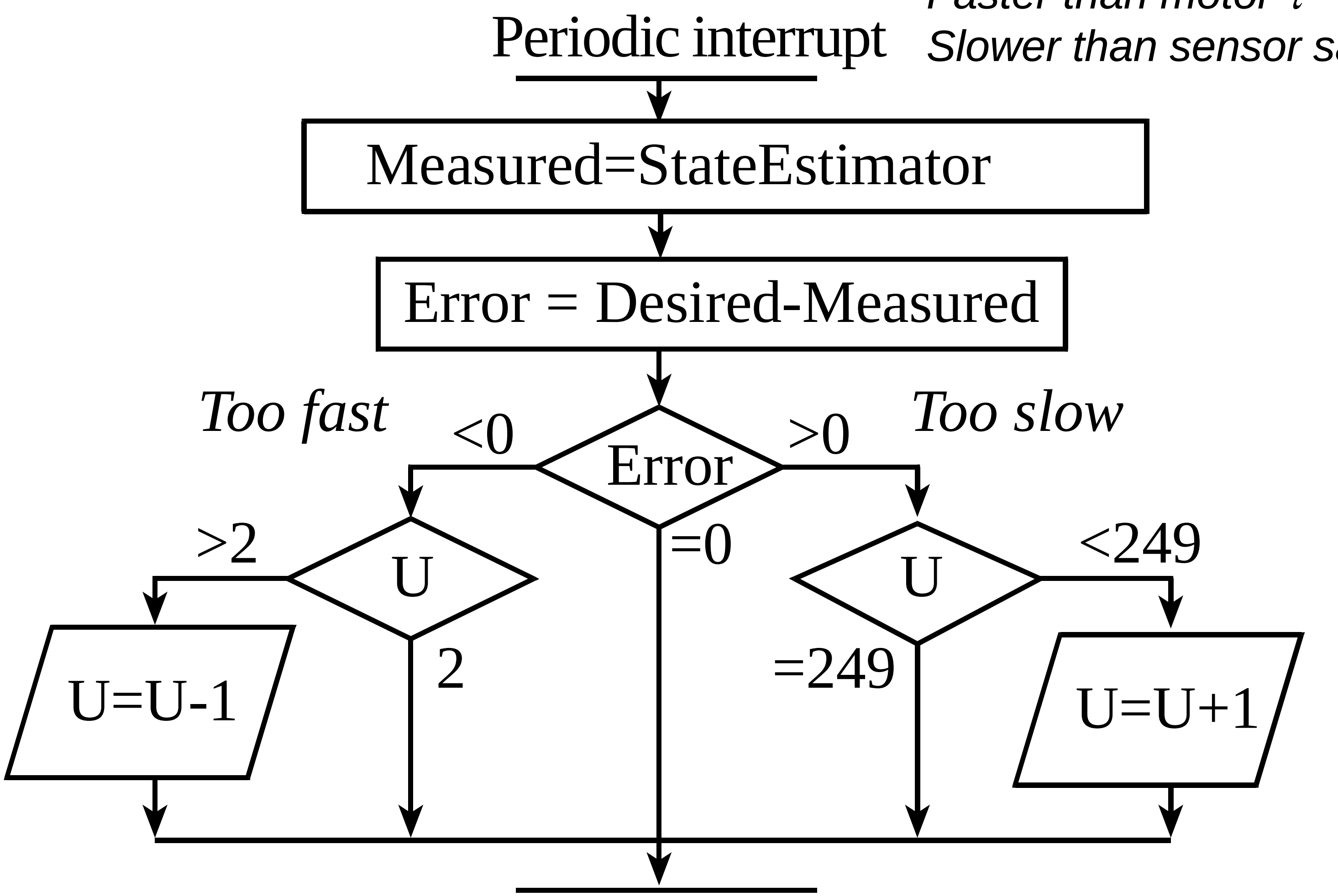
error = $(100*b_L)/a_L - x_L$

if error > 0, $A_L < 90$, turn right
if error < 0, $A_L > 90$, turn left

if $\min(a_L,b_L)<20$, move right-left

Incremental Controller

Faster than motor τ
Slower than sensor sampling rate



Incremental Controller

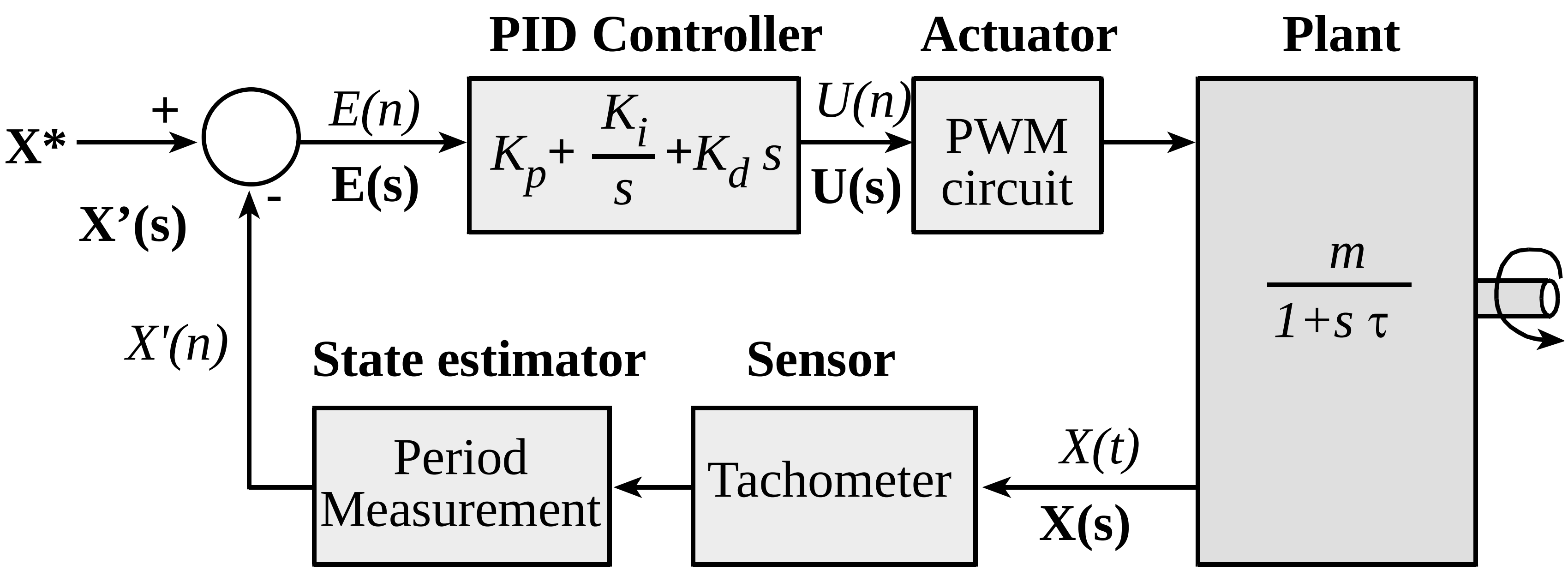
```
long Xstar; // desired
long X;     // actual, measured input
long U;     // PWM actuator output
void Timer0A_Handler(void){ long E;
  E = Xstar-X;              // error
  if(E < 0)      U--;  // too fast
  else if(E > 1) U++;  // too slow
                       // close enough
  if(U <   2) U=2;    // Constrain output
  if(U > 249) U=249;
  PWM0_Duty(U);              // output
  TIMER0_ICR_R = TIMER_ICR_CAECINT;
}
```

Laplace Domain

$$G(s) = K_p + K_d s + \frac{K_i}{s}$$

$$H(s) = \frac{m}{1+\tau\, s}$$

$$\frac{X(s)}{X^*(s)} = \frac{G(s)H(s)}{1+G(s)H(s)}$$



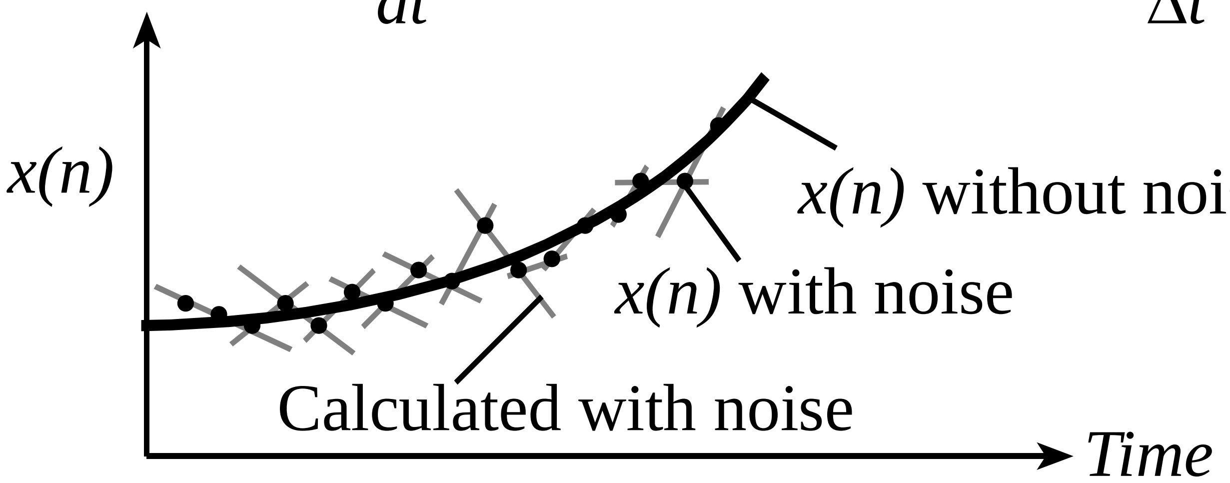
General Approach to PID

$$U(t) = K_p E(t) + \int_0^t K_i E(\tau) d\tau + K_d \frac{dE(t)}{dt}$$

- Proportional $U_p = K_p E$
- Integral $U_i = U_i + K_i E \Delta t$
- Derivative $U_d = K_d(E(n)-E(n-1))/\Delta t$
- PID $U = U_p + U_i + U_d$
- Run ten times faster than motor τ
- Run slower or equal to sensor sampling rate

Derivative Term

$$D(t) = K_d \frac{dE(t)}{dt} \quad \Rightarrow \quad D(n) = K_d \frac{E(n) - E(n-1)}{\Delta t}$$



$$D(n) = K_d \left(\frac{1}{2} \frac{E(n) - E(n-3)}{3\Delta t} + \frac{1}{2} \frac{E(n-1) - E(n-2)}{\Delta t} \right)$$

$$D(n) = K_d \left(\frac{E(n) + 3E(n-1) - 3E(n-2) - E(n-3)}{6\Delta t} \right)$$

- **Analog filter**
- **Digital filter**
- **Linear regression**

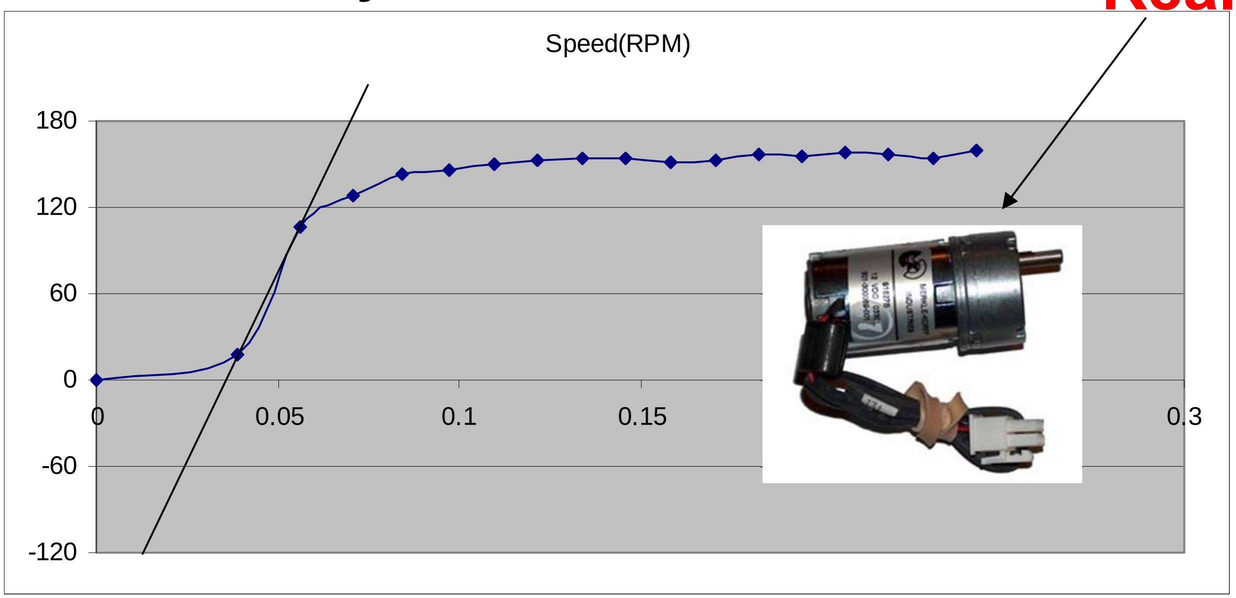
PI Controller

```
void Timer0A_Handler(void){
  E = Xstar-X;
  P = (105*E)/20;       // Kp = 105/20
  I = I+(101*E)/640;    // Ki∆t = 101/640
  if(I < -500) I=-500; // anti-reset windup
  if(I > 4000) I=4000;
  U = P+I;              // PI controller
  if(U < 100) U=100;    // Constrain output
  if(U>19900) U=19900;
  PWM0_Duty(U);         // output
  TIMER0_ICR_R = TIMER_ICR_CAECINT;
}
```

Motor Parameters

- Invoke a step, measure response
  - Time lag, time constant, τ
- Plot speed in RPM versus duty cycle
  - Sensitivity m

Real ≠ Theory



$$H(s) = \frac{m}{1 + \tau\, s}$$

Controller tuning

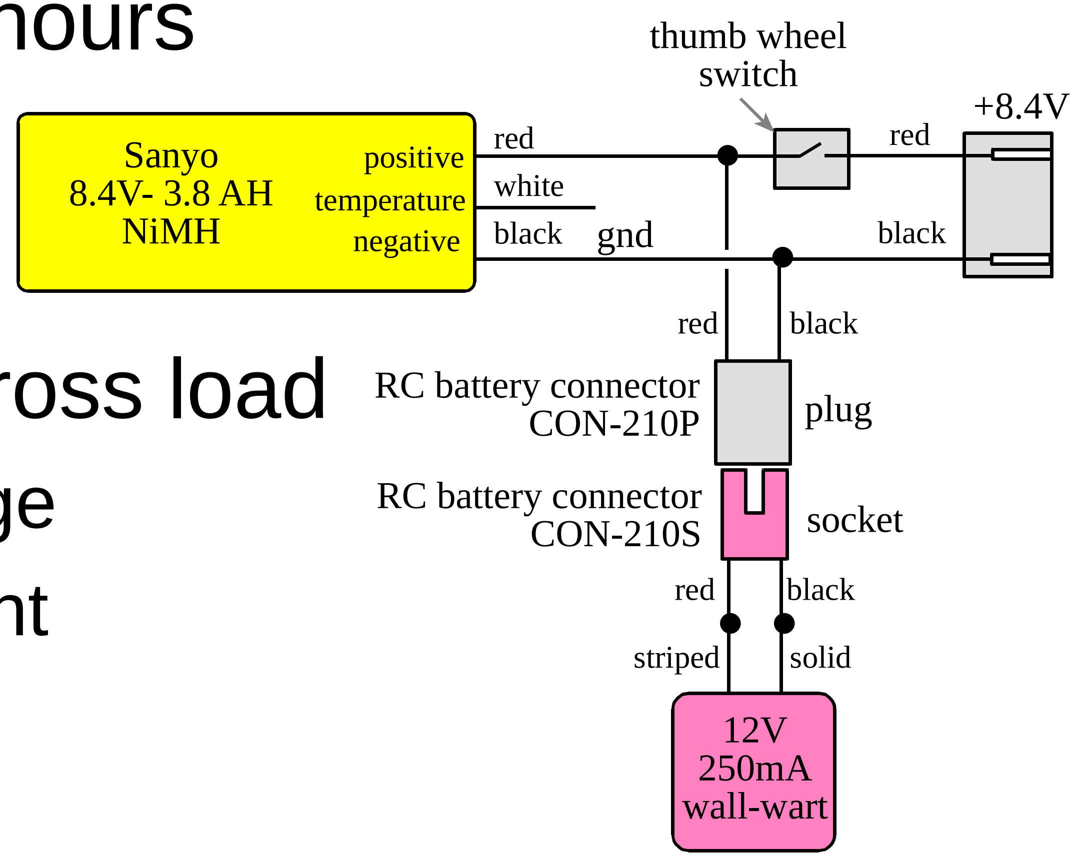
- Start with just a proportional term (**Kp**).
  - proportional controller will generate a smooth motor speed
  - choose the sign of the term **Kp** so the system is stable.
  - try different **Kp** constants until the response times are fast enough
- The next step is to add some integral term (**Ki**)
  - a little at a time
  - to improve the steady state controller accuracy
  - without adversely affecting the response time.
  - choose the sign of the term **Ki** so the system is stable.
  - Don’t change both **Kp** and **Ki** at once.

The last step is the derivative term (**Kd**)
  - a little at a time
  - reduce the overshoots/undershoots in the step response
  - choose the sign of the term **Kd** so the overshoots/undershoots are reduced.

Highly nonlinear -> empirical approach

Charging battery

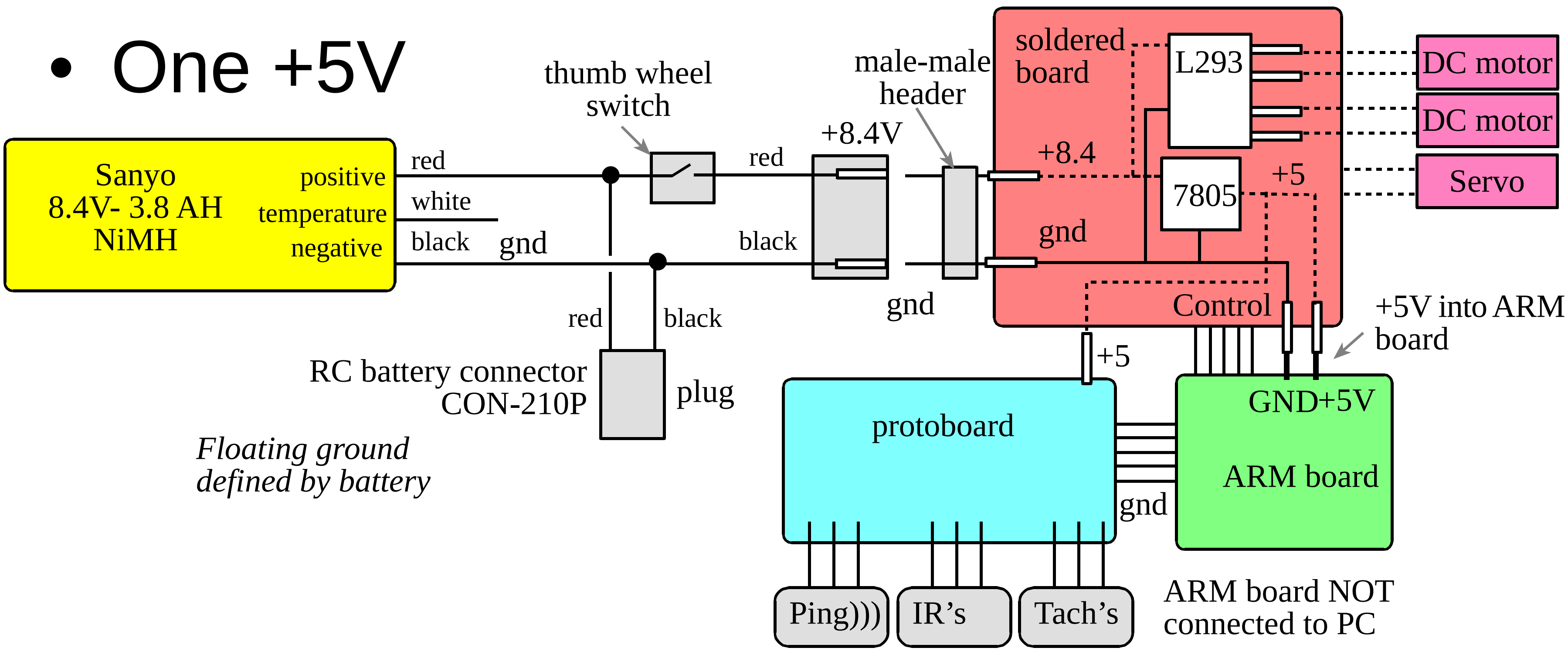
- Drain battery 10Ω 25W resistor (or motor)
- Charge than 8 hours



- Test battery across load
  - Measure voltage
  - Measure current
  - 3.8 Amp-hour

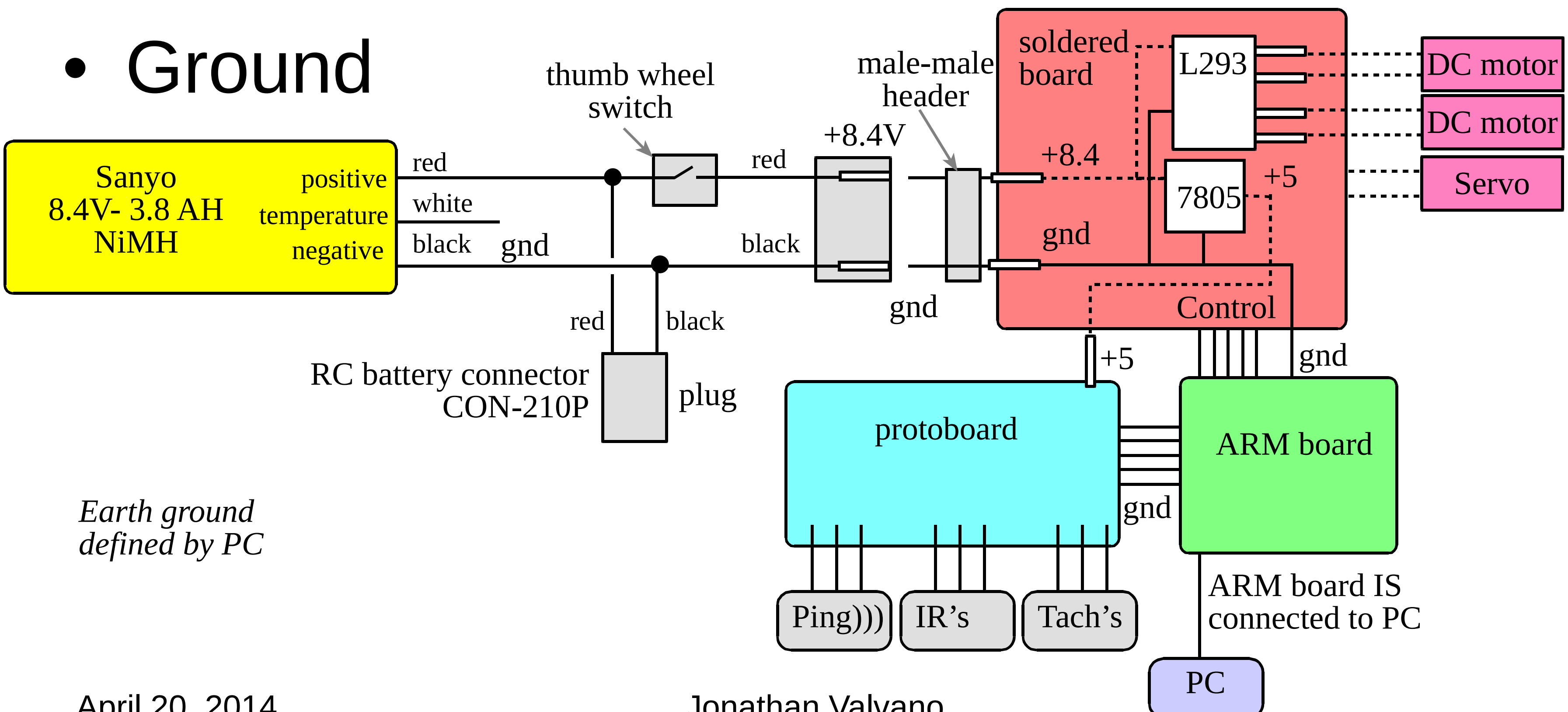
Under Battery Power

- Current path battery-motor-battery
- Ground
- One +5V



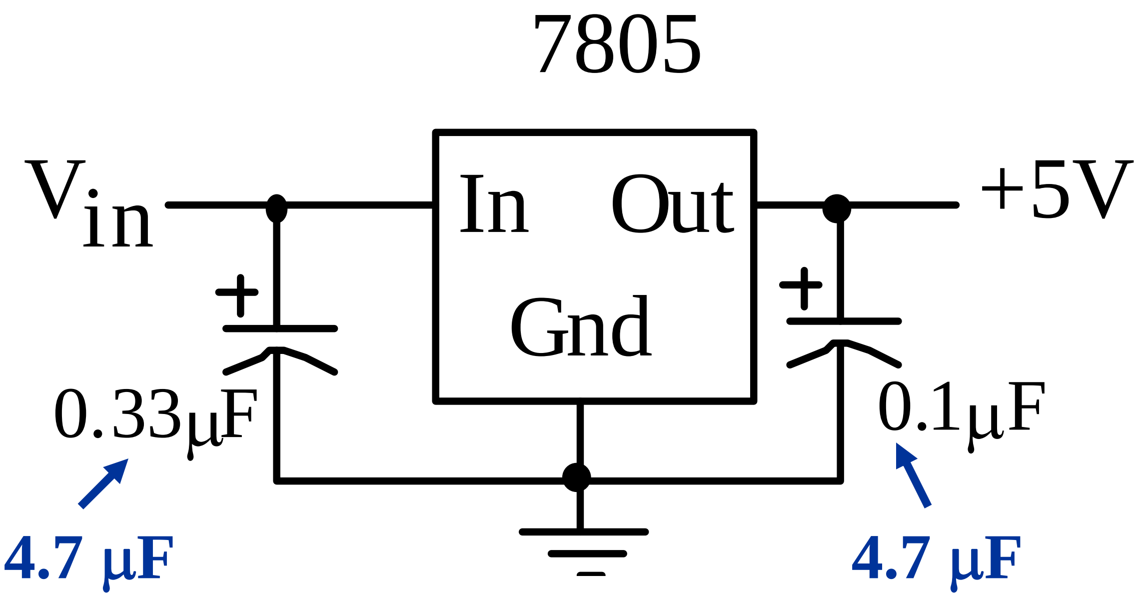
While debugging with PC

- Prevent motor current from going to board
- Two +5V, not connected
- Ground



Regulator

- Drop out, V_d
- Calculate power loss, $P=(8.4-5)*I$



$V_d = 2.5V$

$V_{in} > 7.5V$

$I_{out} < 1A$