Magnetic Levitation

Michael Everett
Wyatt Ubellacker

Outline

Hardware Setup

Linearization

Controller Design

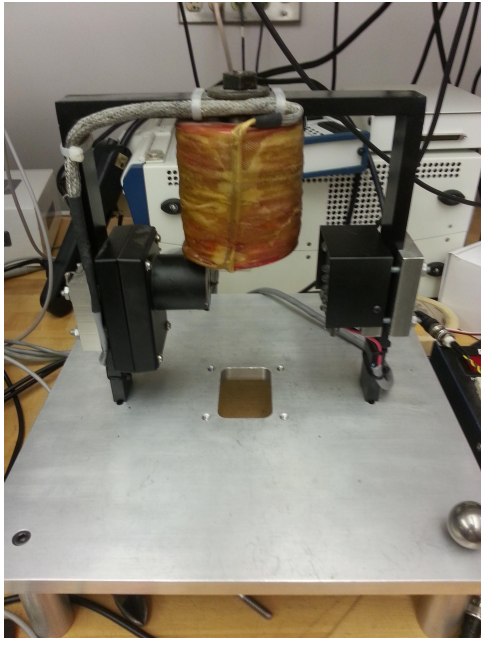
Challenges

Results

Demo

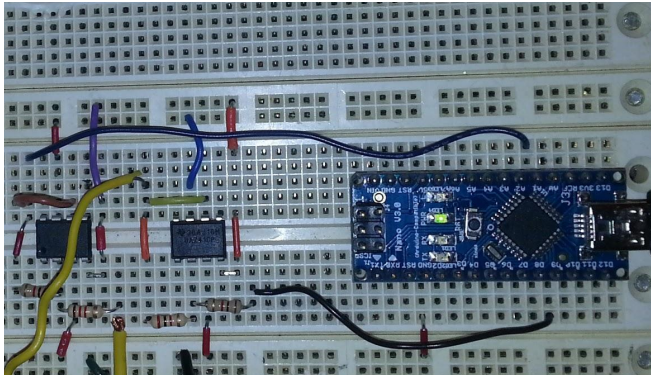
Hardware Setup

Light-based position sensor and coil wound electromagnet



Arduino vs MyRio

- Wanted to implement controller in C-like language with difference equation rather than a transfer function block
- Good documentation, lots of examples



Arduino scaling Op-Amps for sensor and drive

- Arduino analog input is 0-5V
- High frequency PWM output to drive

Linearization

System Dynamics (non-linear):

$$F_{mag} - mg = m\ddot{x}$$

$$F_{mag} = C\left(\frac{i}{x}\right)^2$$

Linearizing about some operating point and some operating current gives:

$$F_{mag} = C\left(\frac{i_0}{x_0}\right)^2 + \frac{\partial F_{mag}}{\partial i} * \tilde{i} + \frac{\partial F_{mag}}{\partial x} * \tilde{x}$$

Linearization (cont.)

Plugging in for system dynamics gives:

$$m\ddot{x} = -mg + C\left(\frac{i_0}{x_0}\right)^2 + \frac{\partial F_{mag}}{\partial i} * \tilde{i} + \frac{\partial F_{mag}}{\partial x} * \tilde{x}$$

Choosing our operating point and corresponding current to balance the force of gravity gives the final linearized system:

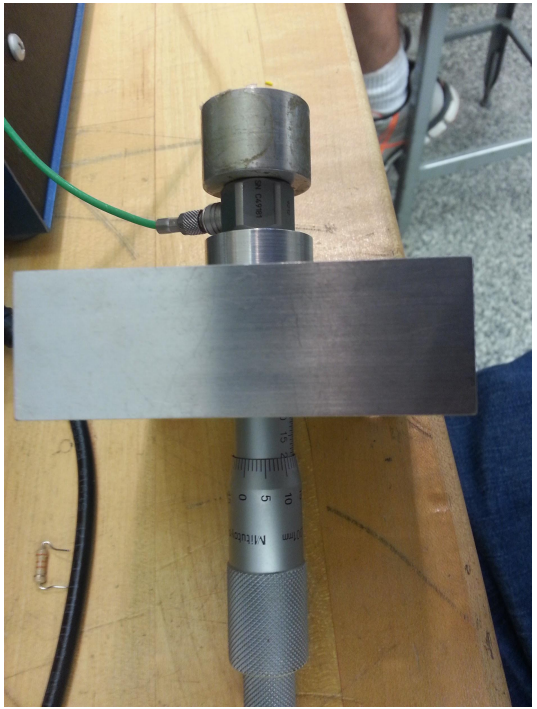
$$m\ddot{x} = \frac{\partial F_{mag}}{\partial i} * \tilde{i} + \frac{\partial F_{mag}}{\partial x} * \tilde{x}$$

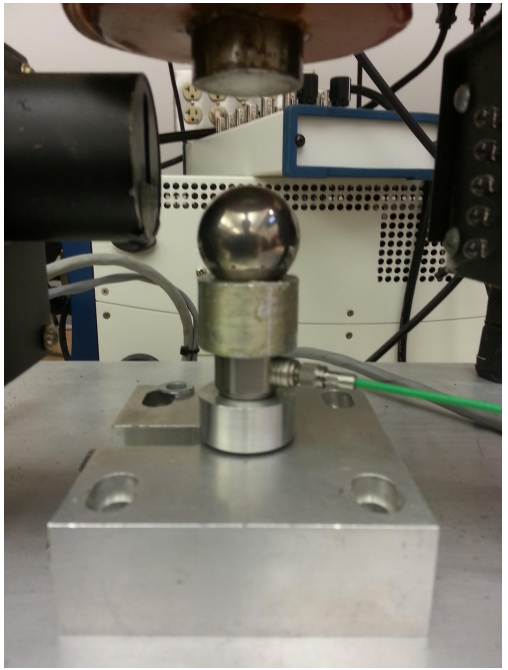
Linearization (cont.)

We need to measure:

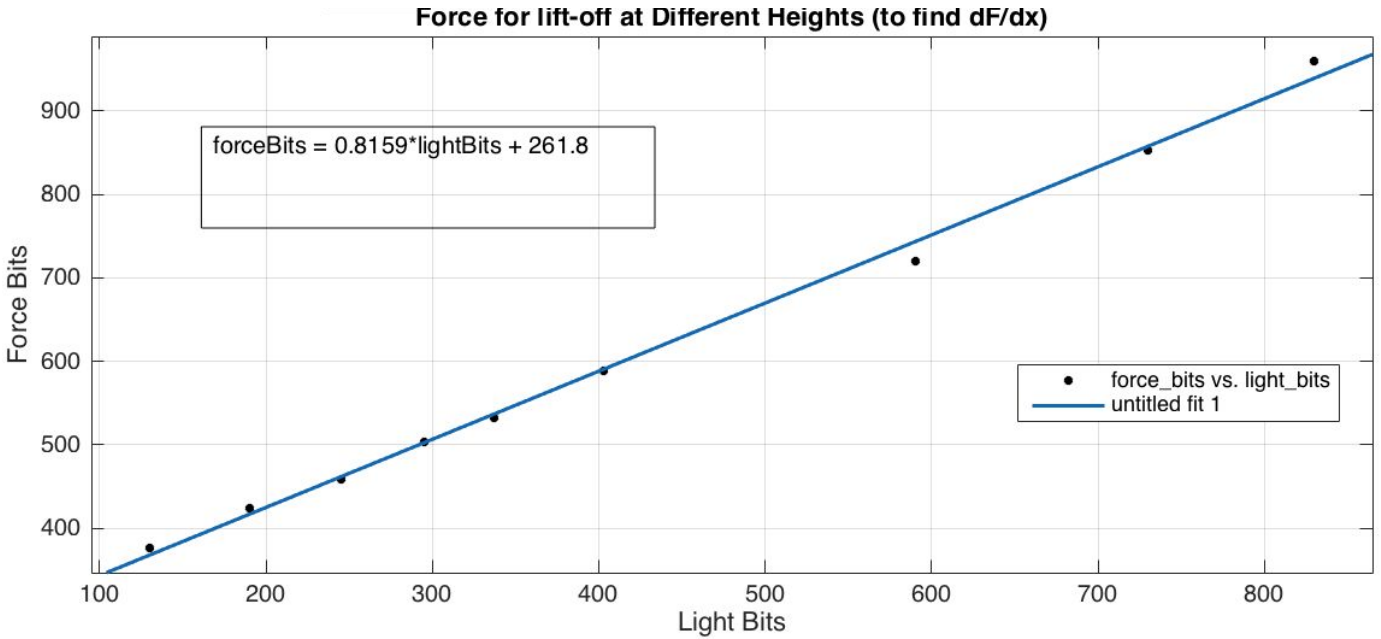
$$\frac{\partial F_{mag}}{\partial x} \qquad \frac{\partial F_{mag}}{\partial i}$$

We glued the ball to a micrometer and measured change in force for varying height and current around the operating points.





Linearization (cont.)



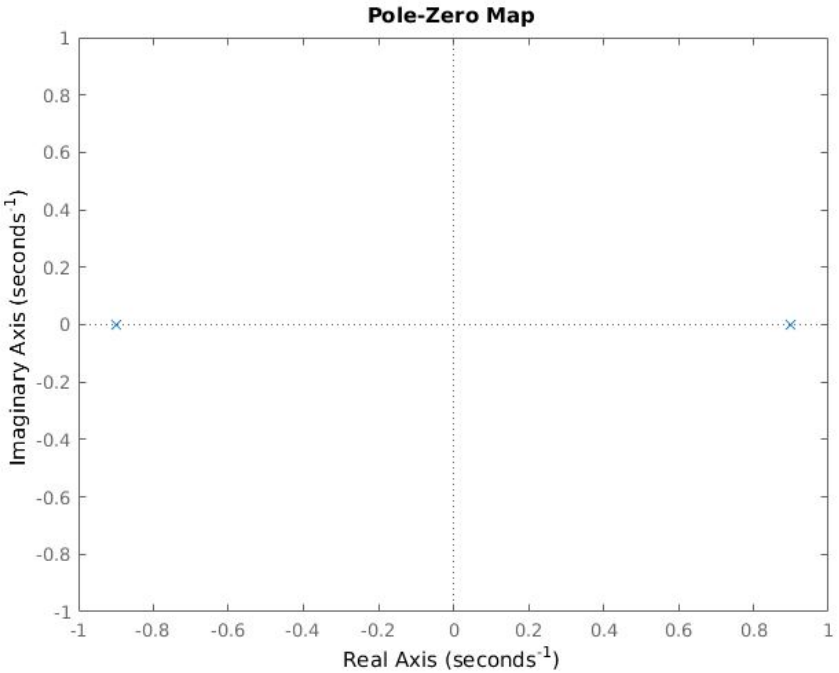
Linearization (cont.)

Fitting the measurements results in:

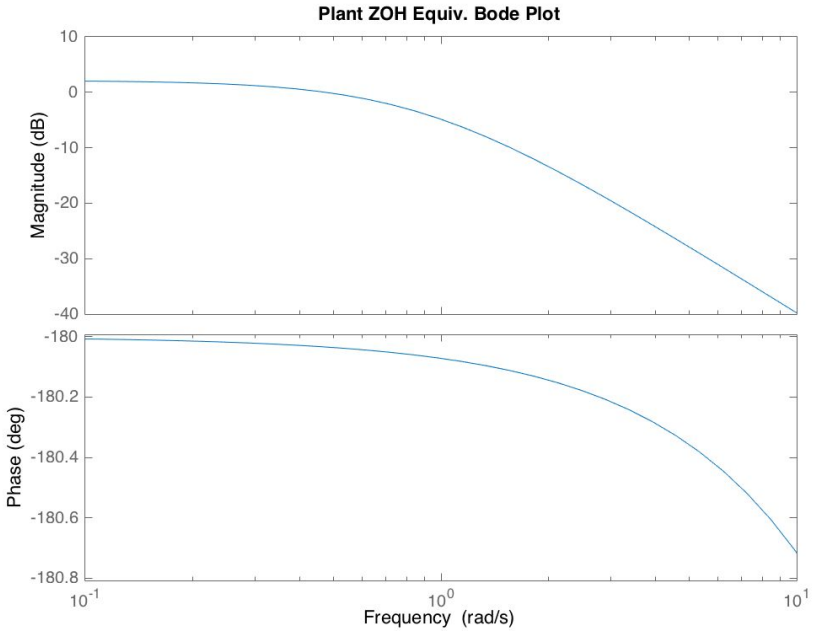
$$\frac{\partial F_{mag}}{\partial i} = 1.034 \qquad \frac{\partial F_{mag}}{\partial x} = -0.81$$

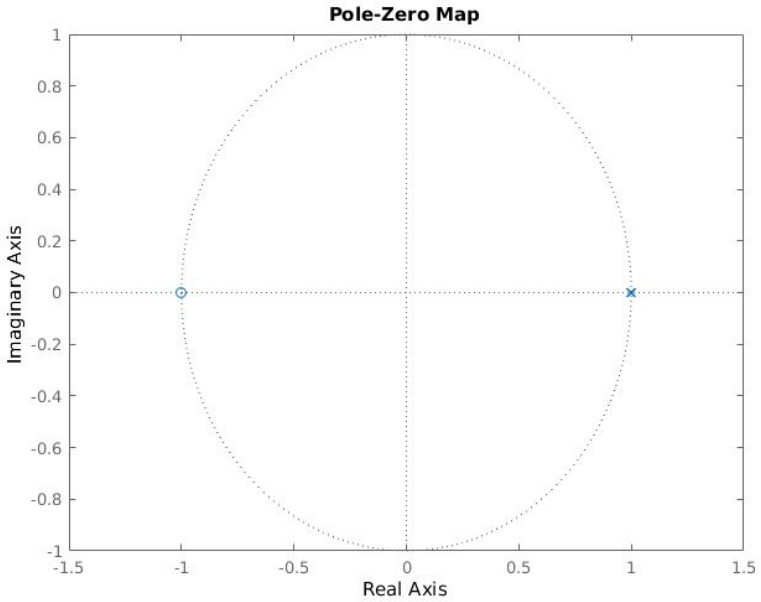
And a linearized transfer function of:

$$\frac{I}{X} = \frac{1.034}{s^2 - .81}$$

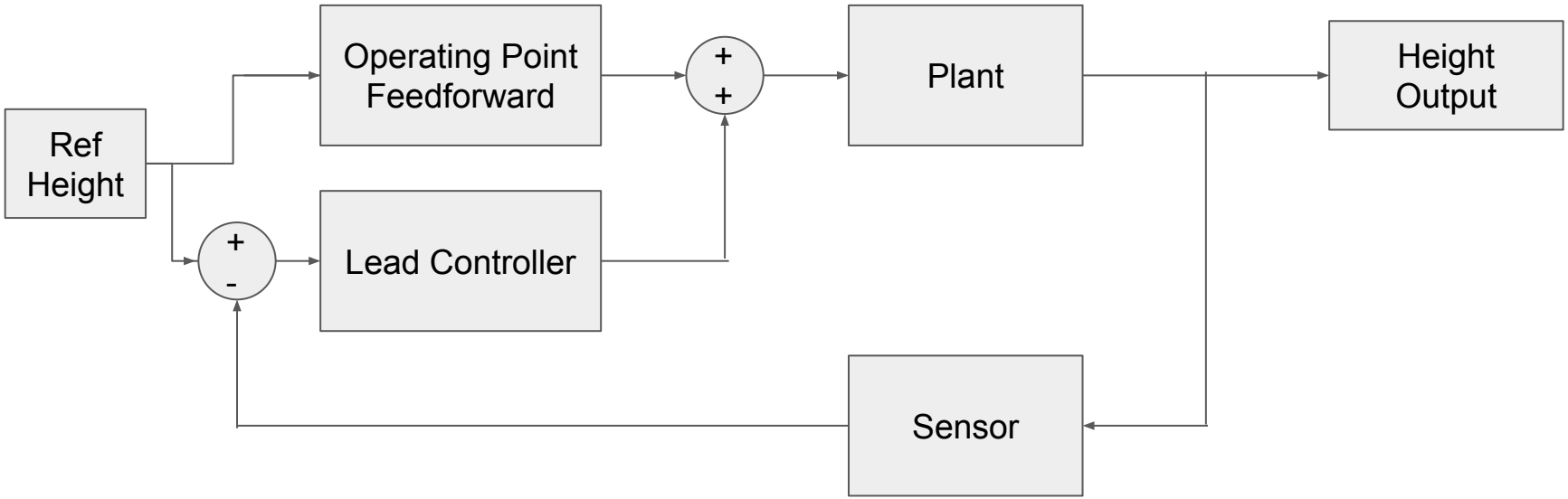


Linearization (cont.)





Controller Design



Controller Design

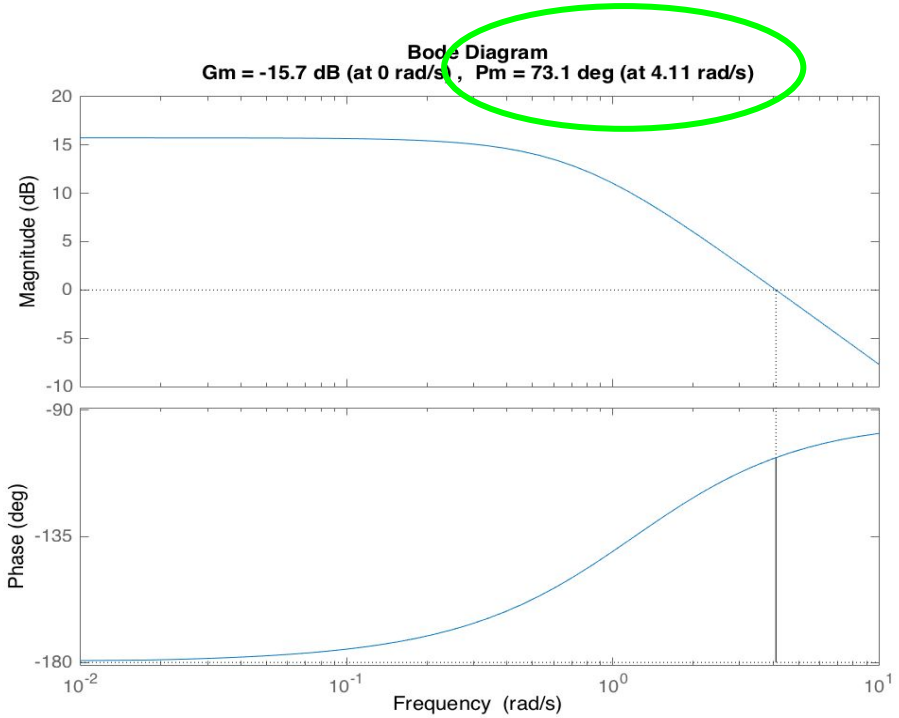
Don’t really care about SS error, so we can just use a Lead Compensator:

$$C(z) = K_p \frac{z - a}{z - b}$$

$$a = .997$$

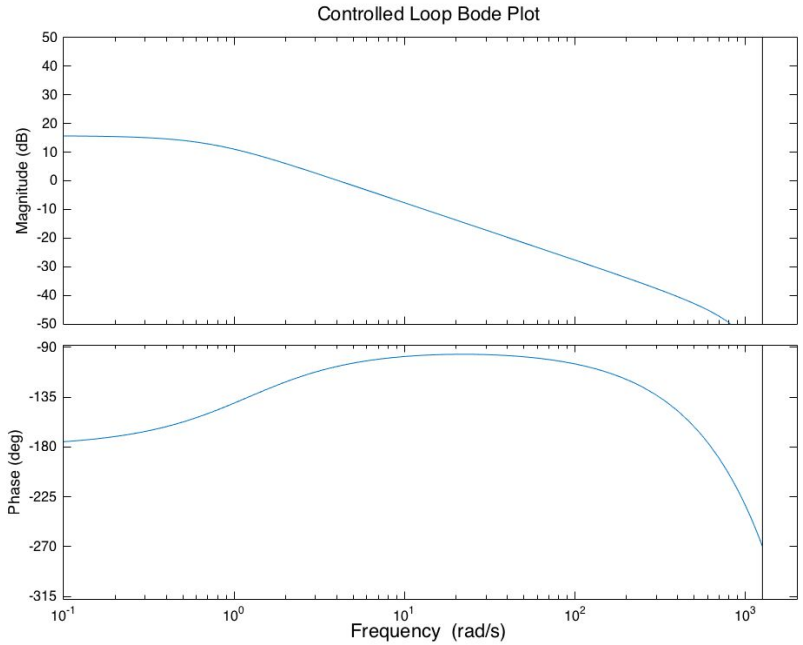
$$b = 0$$

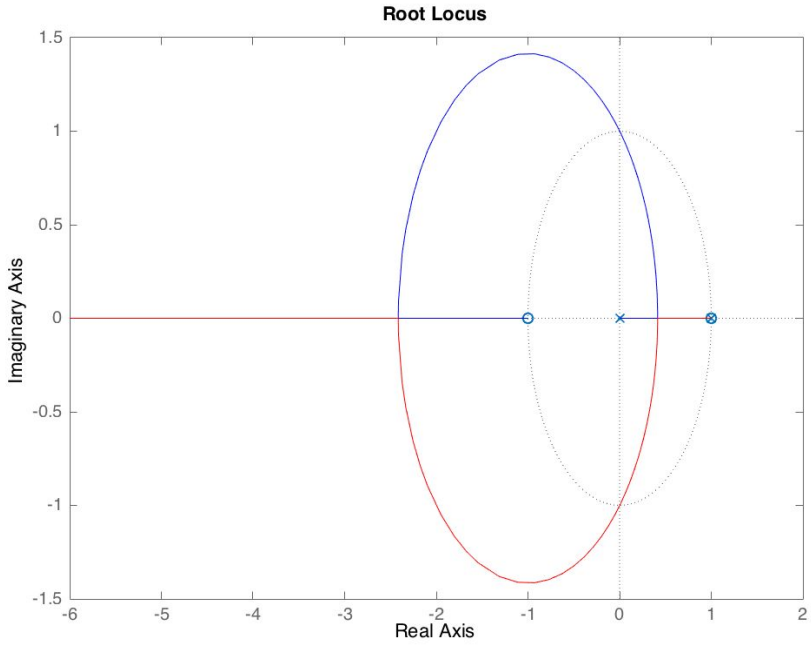
$$K_p = 1600$$



Controller Design (cont.)

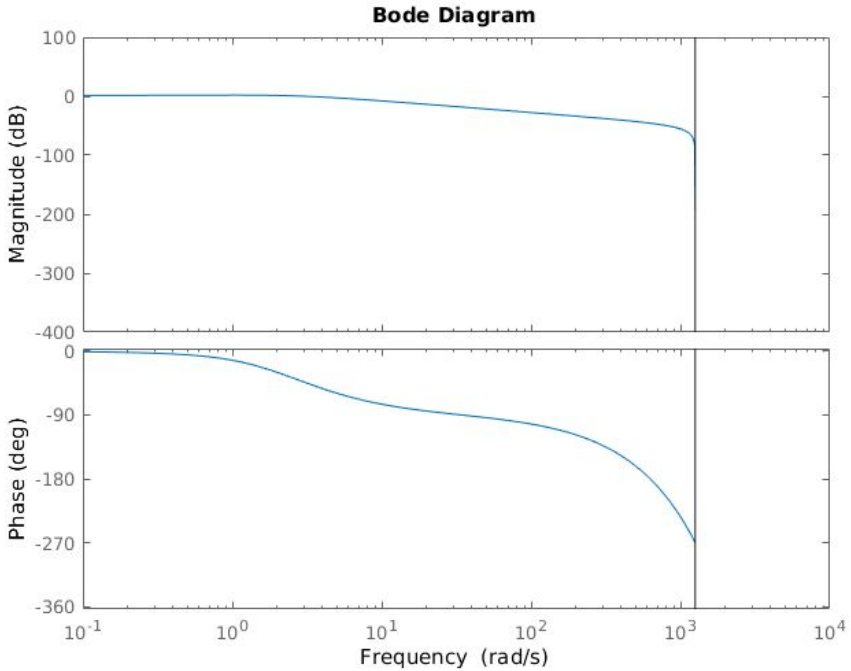
Loop Transmission Bode plot and root locus:

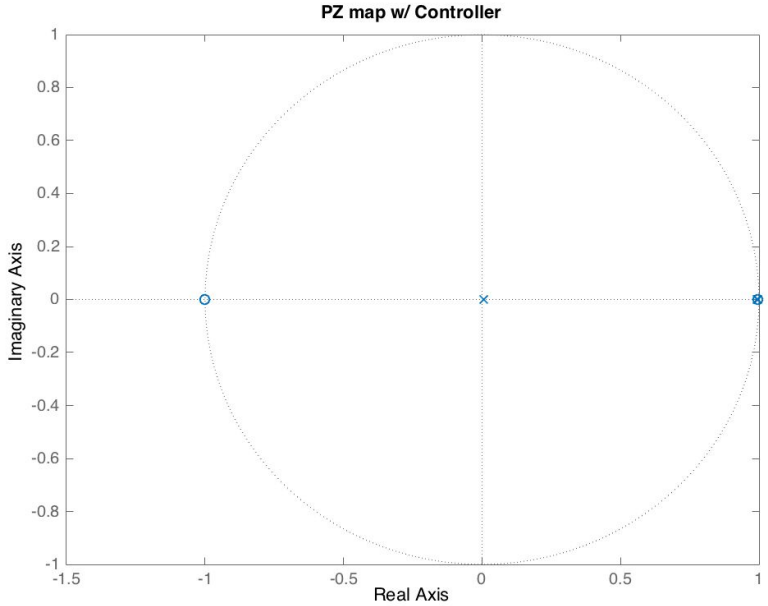




Controller Design (cont.)

Closed Loop transfer function, Bode plot, and pole zero map



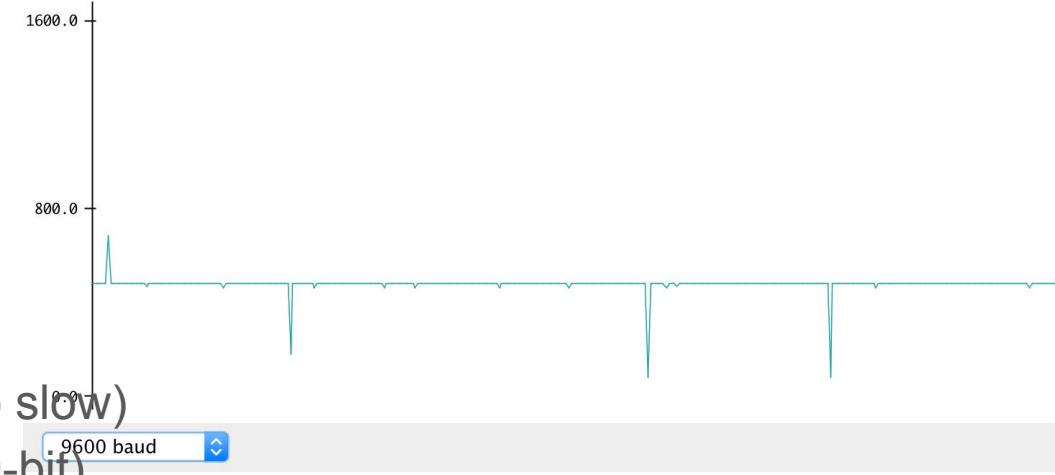


$$\frac{0.00517\,(z+1)\,(z-0.997)}{(z-0.9957)\,(z-0.9939)\,(z-0.005208)}$$

Sample time: 0.0025 seconds

Challenges

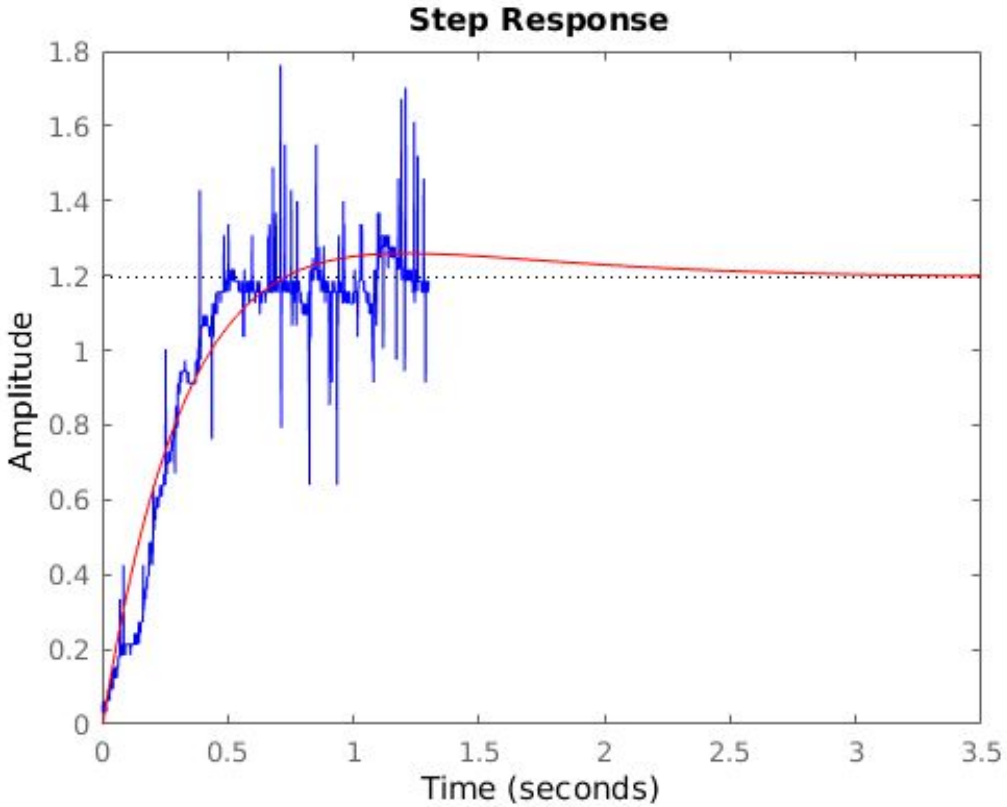
Arduino hardware limitations



- No DAC (only PWM)
- Standard PWM is at 500 Hz (too slow)
- Low output resolution (8- and 10-bit)
  - High Kp -> high crossover, but lowers output resolution
- RC Filtering PWM output - even enhanced PWM too slow
- external DAC chip was only 8-bits - too low resolution
- Hard to measure Bode Plot
  - Sine calculation slow
  - Lookup table also slow
  - Interferes with loop executing @ 4kHz
- 0.1 Hz sensor noise
- Significant sensor noise in last few days

Results

	Simulated	Experimental
Time Constant	260ms	250ms
Overshoot	5%	3%
S.S. error	20%	16%



Demo

https://www.youtube.com/watch?v=Chfn8uVCth8