2.004 Lab 2 Intro
Characterization of The DC Motor / Flywheel Plant

September 27, 2021

Today's Tasks

- **Assemble DC motor / flywheel plant**
- **Verify encoder signal and conversion factor**
- **Derive angular velocity from the encoder signal**
- **Code a first order low-pass filter to smooth the velocity signal**
- **Create an open-loop transfer function between PWM pin voltage and output angular velocity**
- **Extra Credit Task: Simscape simulation**
- **Deliverable:**
  - Lab 2 report (use the report template) to Gradescope by midnight

Lab Report Grading

- **Lab Report Grading:**

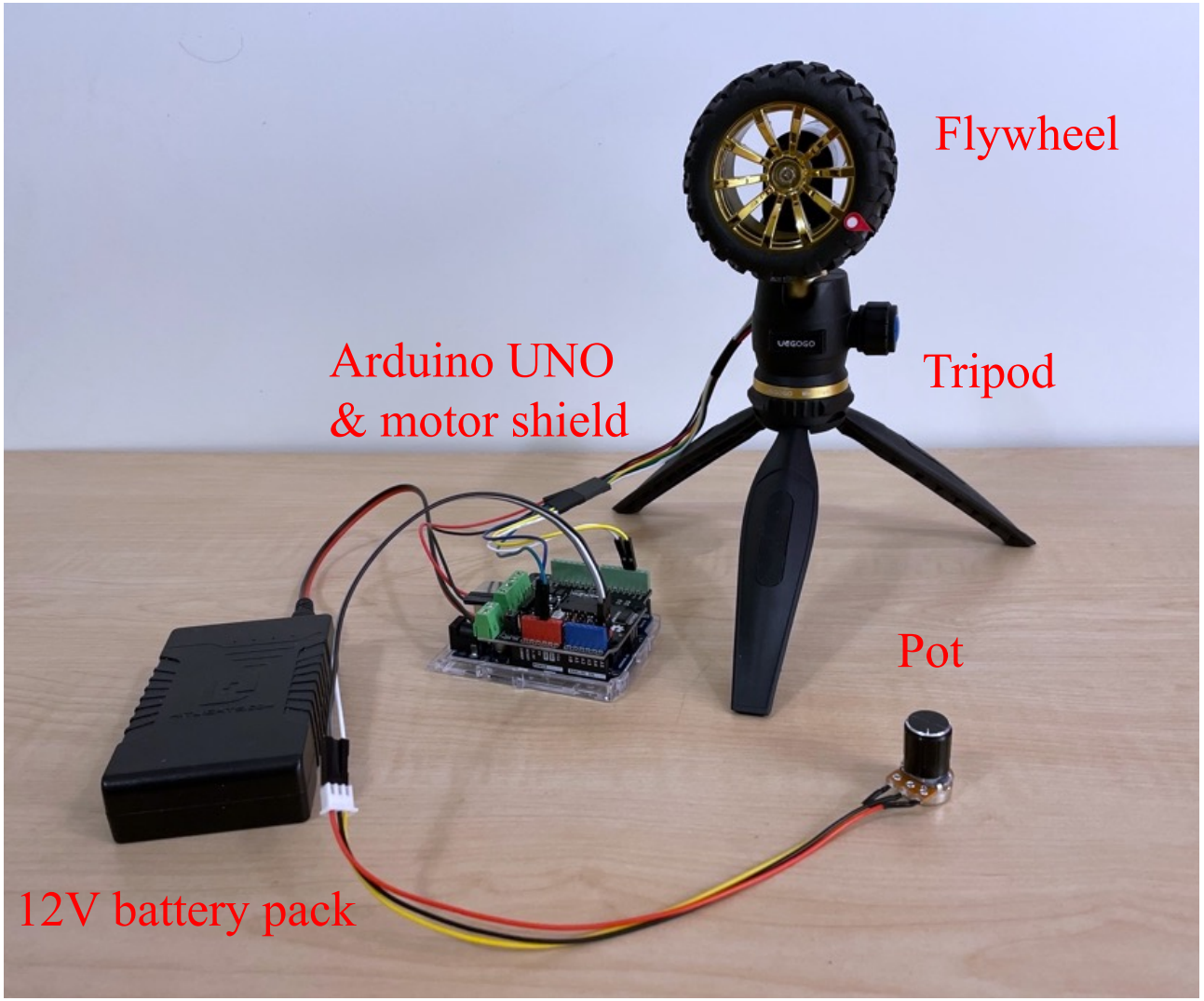
  0 – No show / No report received

  20 – Completed all the required parts with reasonable results

  Deduction for mistakes such as missing units, unreasonable answers, missing figures, etc.

  ★ Earn additional 3 points if completed all the required parts and the extra credit task(s)

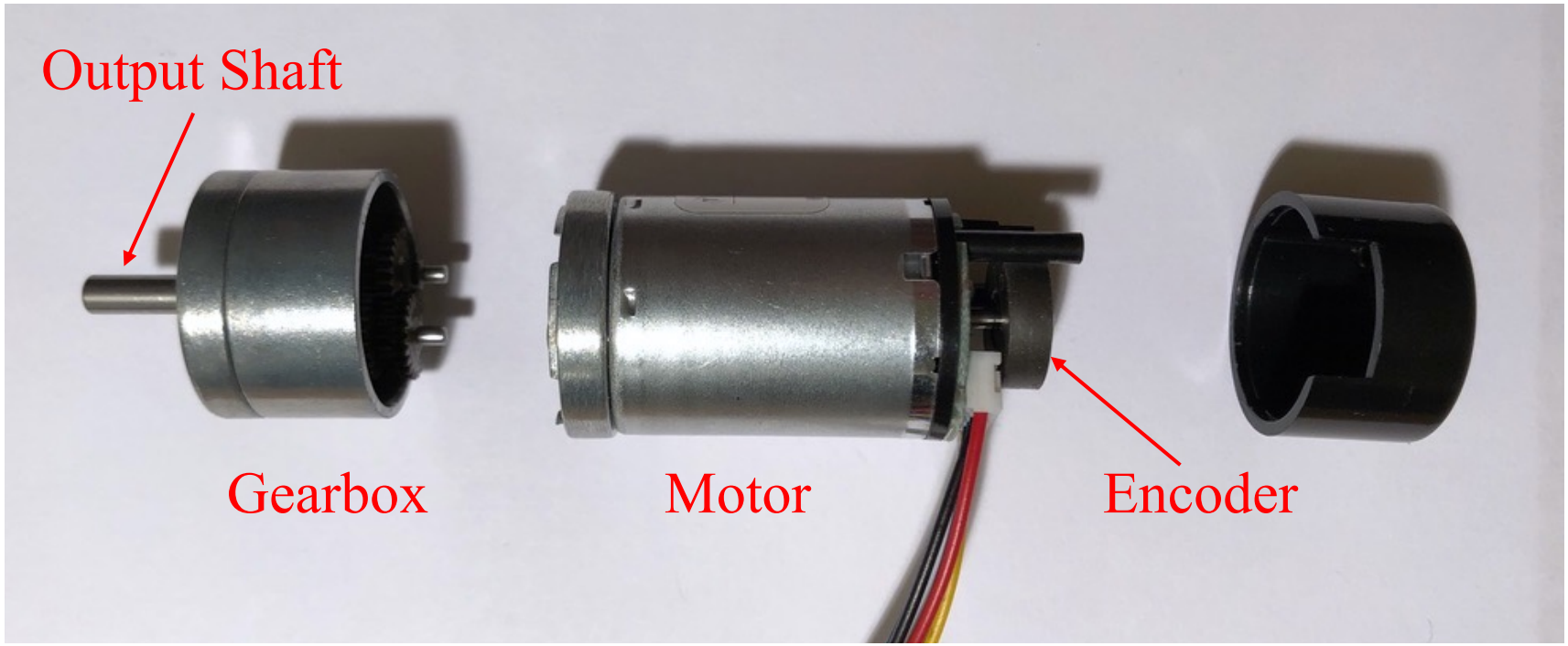
The DC Motor / Flywheel Plant



12V DC Gear Motor with Encoder

Motor Characteristics	
Input Voltage	3 – 12 VDC
Rated Current	350 mA
Stall Torque	11 kg.cm
No-load Speed	294 rpm (rev. per minute)
Encoder Resolution	12 cpr (counts per rev.) per channel
Gear Ratio	1:34
Encoder Voltage	3.3 – 5 V

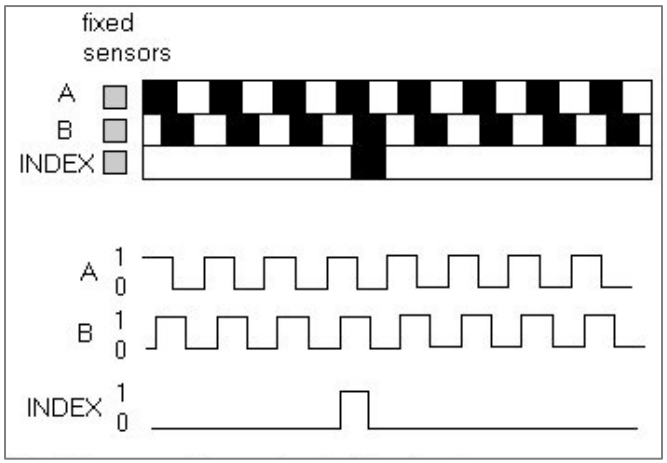
Gear Motor with Encoder



Position Sensing Using Encoder

- The back of the motor is equipped with a dual channel Hall effect incremental encoder that is used to sense the rotation of a magnetic disk on the rear of the motor shaft. Each channel of the encoder provides a resolution of 12 counts per revolution (CPR) of the motor shaft. With quadrature decoding, the effective encoder resolution becomes 48 CPR.
- Since there is a 1:34 reduction gearbox at the front of the motor the output shaft encoder resolution becomes 1632 CPR.

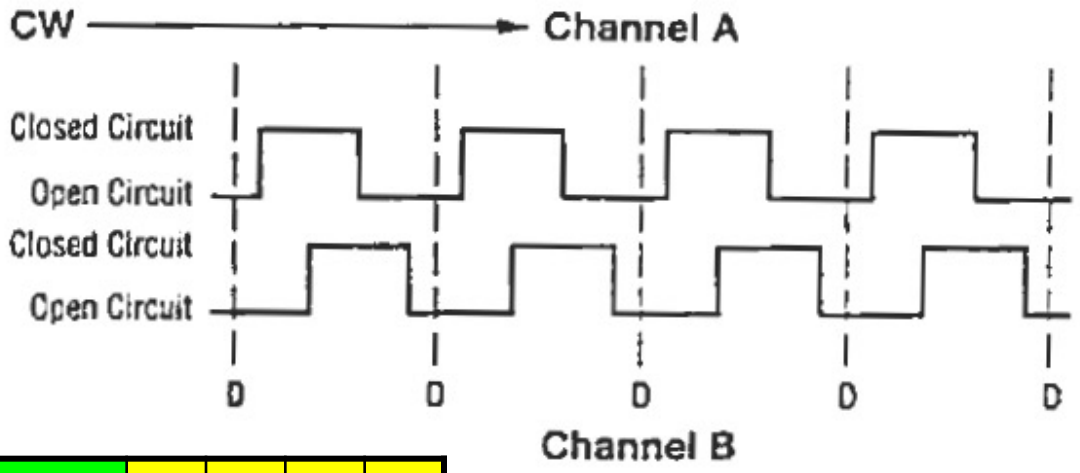
- **Encoder Types:**
  - **Absolute:** Unique code for each shaft position.
  - **Incremental (Relative):** Only the relative position of the shaft is known. No absolute "0" position. An "INDEX" signal can be added to provide "0" position of a shaft.



Example of absolute encoder

Quadrature Decoding

Increasing the encoder resolution by 4x as well as detecting the direction of rotation.



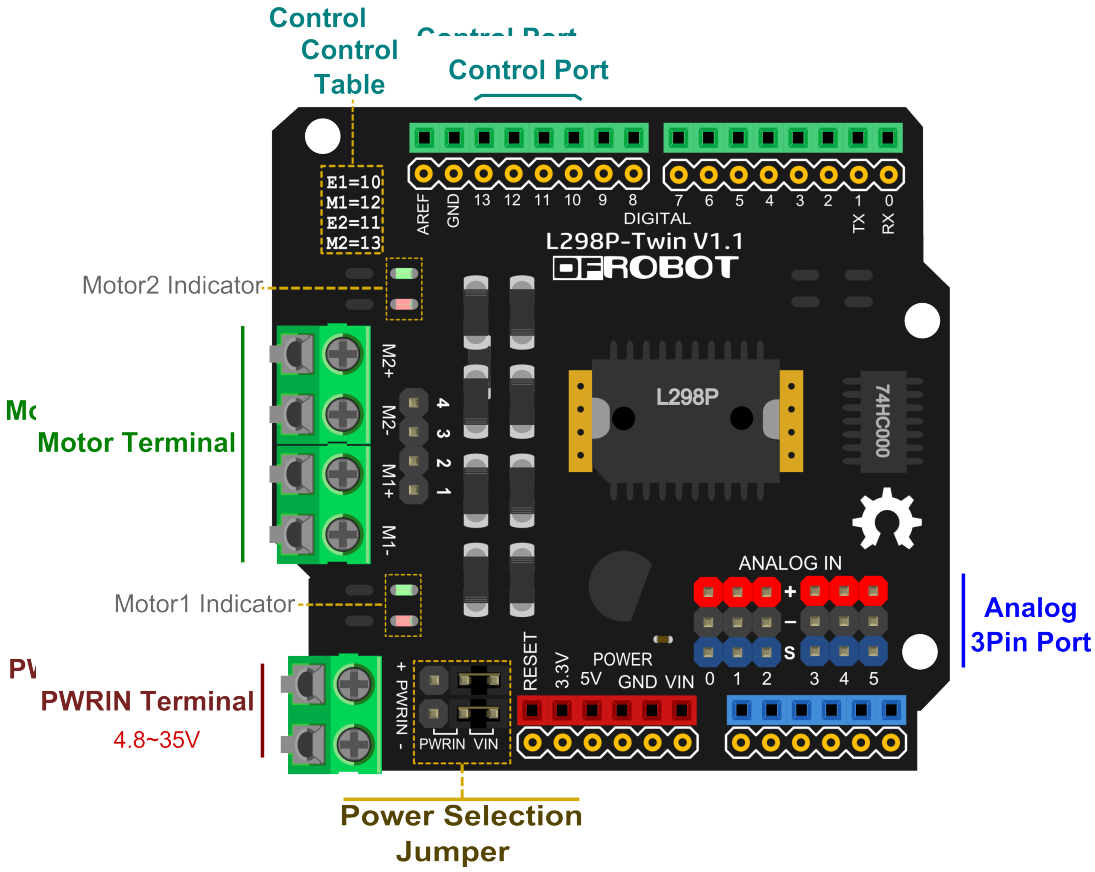
X (Ch A)	Y (Ch B)	F0	F1	F2	F3
0	0	1	0	0	0
1	0	0	1	0	0
1	1	0	0	1	0
0	1	0	0	0	1

Quadrature Decoding Using Interrupts

- For this implementation two Interrupt Service Routines (ISR) are used to detect and process the encoder signals. An interrupt service routine is a software routine that hardware invokes in response to an interrupt.
- The Arduino UNO has two hardware interrupt pins: D2 and D3. We connect encoder channel A to D2, and channel B to D3, respectively.
- When a change in any of the two encoder signals is detected, the corresponding software routine would then read the states of both encoder signals and execute a logic statement to update the encoder counts.

Dual Channel 2A Motor Shield

- **The motor shield allows Arduino to drive two DC motors, using an L298N chip which deliveries output current up to 2A for each motor.**
  - Motor Driven Voltage: 4.8V to 35V
  - Output Current: up to 2A/channel
  - Total Power Dissipation: 25W (T=75°C)
  - Driven Structure: Dual full-bridge driver



https://www.dfrobot.com/product-1180.html

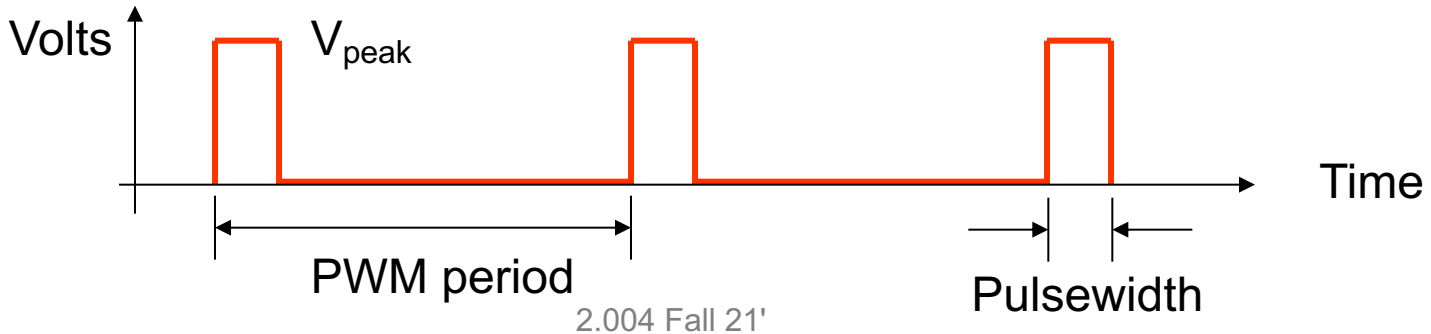
Driving The DC Motor

- Low cost microcontrollers such as the Arduino Uno do not include a Digital-to-Analog Converter (DAC), instead they use digital pins to generate a Pulse Width Modulation (PWM) signal to mimic an analog signal. The Uno's default PWM frequency is 490.2 Hz (period = 0.002 seconds) for the pins we use. Note that the PWM frequency can be changed from about 30 Hz to 30 kHz (https://www.etechnophiles.com/change-frequency-pwm-pins-arduino-uno/).
- Change the value of 'desired_pwm' to a non zero integer value between -255 and 255 (e.g., 150) and upload the code.
- Turn on the power switch in order to deliver the required amount of power to the motor.

Pulse Width Modulation (PWM)

- PWM frequency (Hz) = 1 / PWM period
- Duty cycle = Pulsewidth / PWM period
- PWM frequencies typically range from 100Hz into MHz
- Duty cycles can be used from 0 – 100%, although some systems use much smaller ranges, e.g. 5-10% for hobby remote servos.
- The waveform has two pieces of information: Period and Pulsewidth, although they are usually not changed simultaneously.

Use a scope to look at the PWM signal if you can

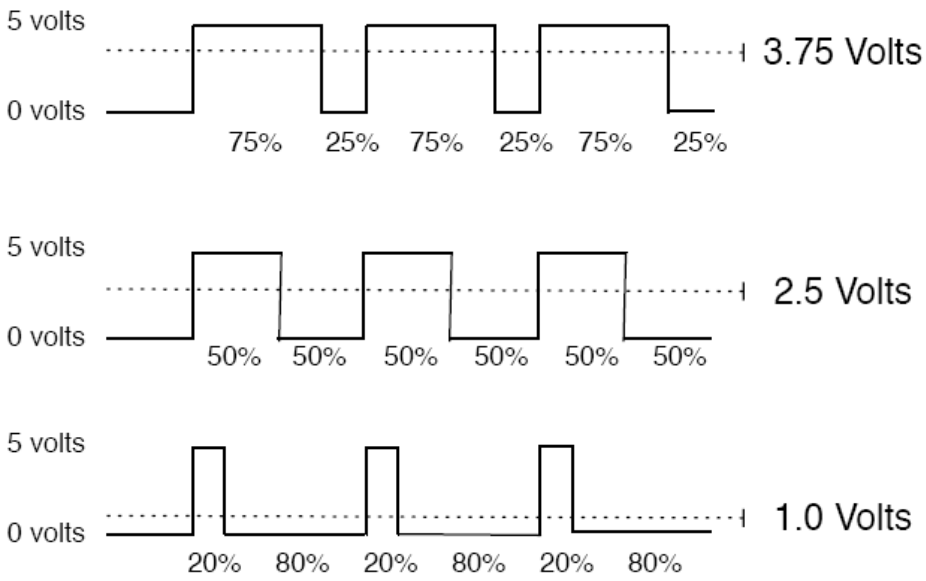


Pulse Width Modulation (PWM)

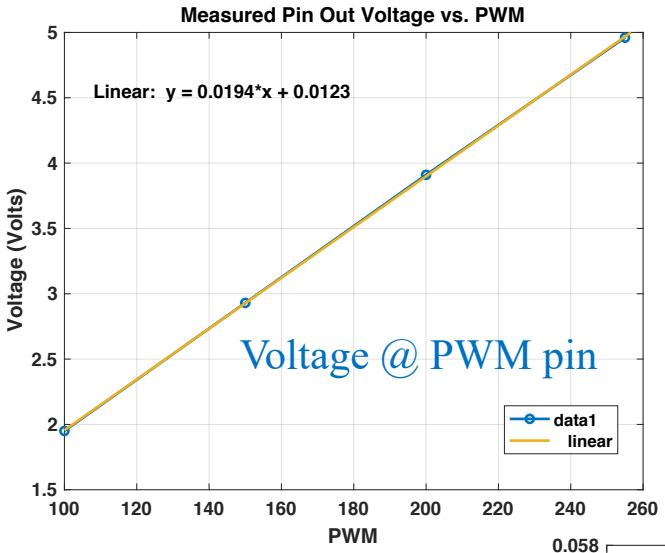
- **Can be used as a substitute for analog output (high frequency switching is filtered out by the physical systems and what is left is the mean voltage).**
- **Applications include: lamp dimmers, motor speed control, power supplies,…**

Output voltage is averaged from on *vs.* off time

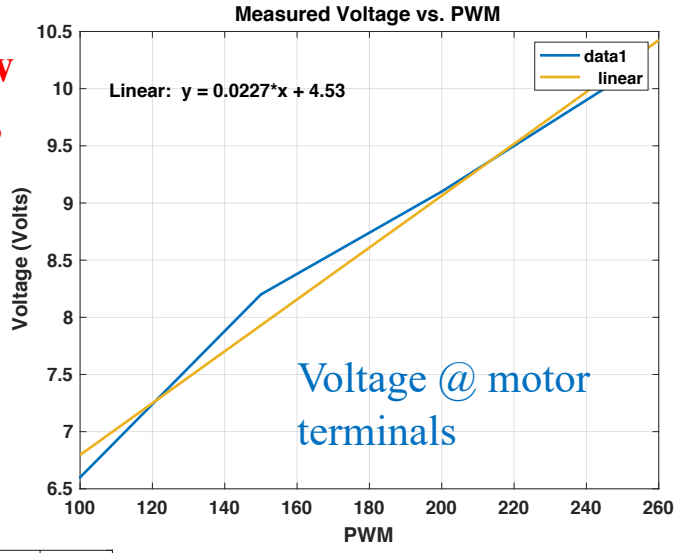
output_voltage = (on_time / off_time) * max_voltage

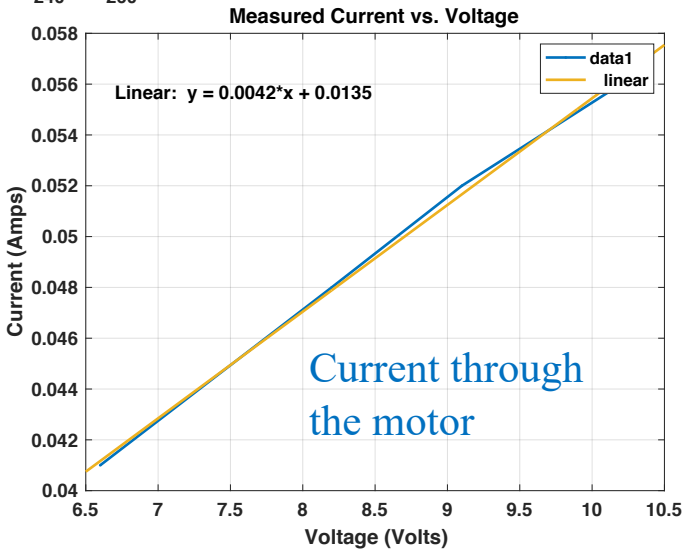


Motor Shield Measurements (with a Multimeter)



All three plots show linear relationships





Velocity Calculation and Smoothing Filter

Calculate velocity from displacement:

$$\omega(t) = \frac{d}{dt}\theta(t) \approx \frac{\theta(t) - \theta(t - \Delta T)}{\Delta T}$$

Simple low-pass filter to reduce noise:

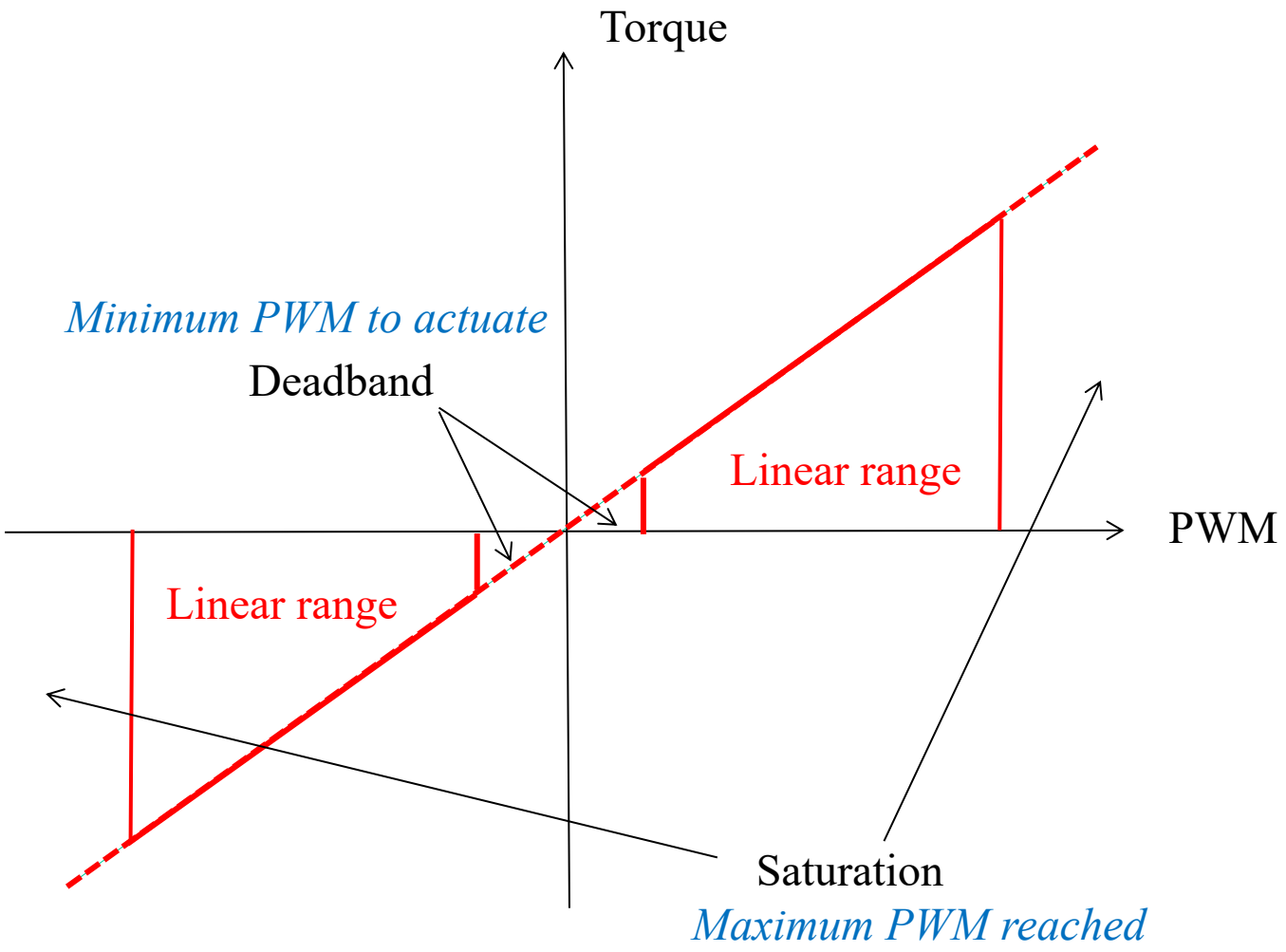
$$\hat{y}(t) = \alpha \cdot y(t) + (1 - \alpha) \cdot \hat{y}(t - \Delta T), \qquad 0 \le \alpha \le 1$$

$$\alpha = \frac{\Delta T}{(\Delta T + \tau)}$$

Example: sample period $\Delta T = 0.015s$ and desired filter time constant $\tau = 0.1s$

$$\alpha = \frac{0.015}{(0.015 + 0.1)} = 0.13$$

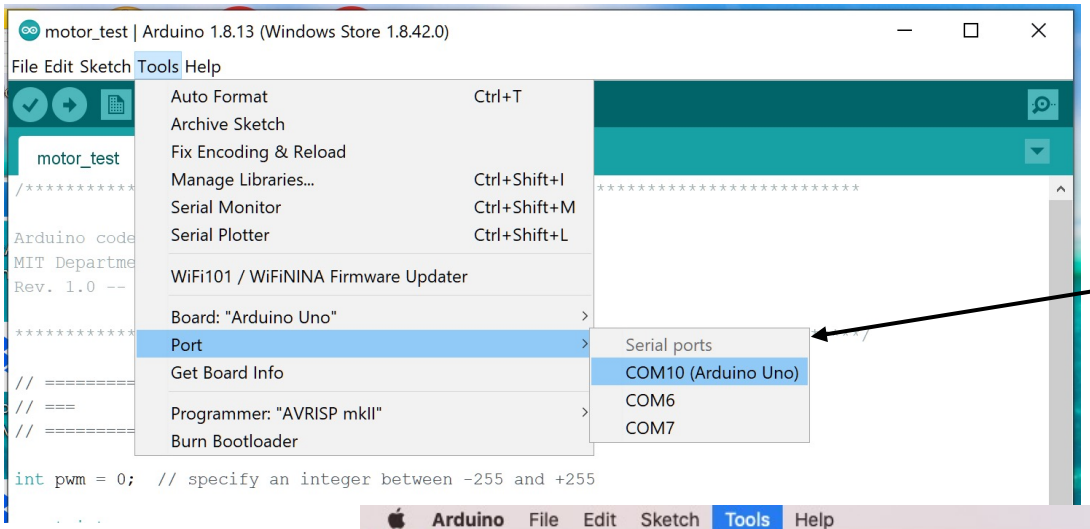
Saturation and Deadband



Arduino Software

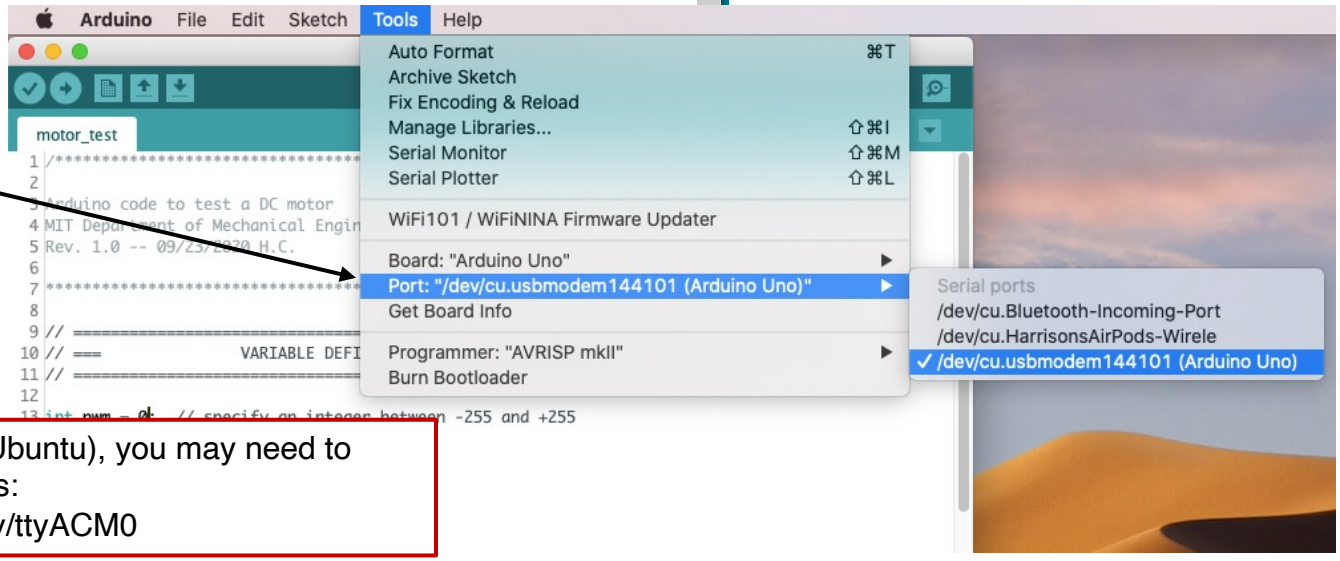
- **Download and install the Arduino IDE (https://www.arduino.cc/en/Main/Software) if you have not done so already. The latest version is 1.8.16.**
- **Connect the Arduino UNO board to your computer with the USB cable.**
- **Open the "motor_encoder_template.ino" sketch, and select the proper Arduino board and serial port from the top menu bar:**
  - Tools → Board → Arduino UNO
  - Tools → Port → (The serial port used by the Arduino)
- **Upload the sketch to the board by pressing the "upload" button.**

Serial Port Selection Example



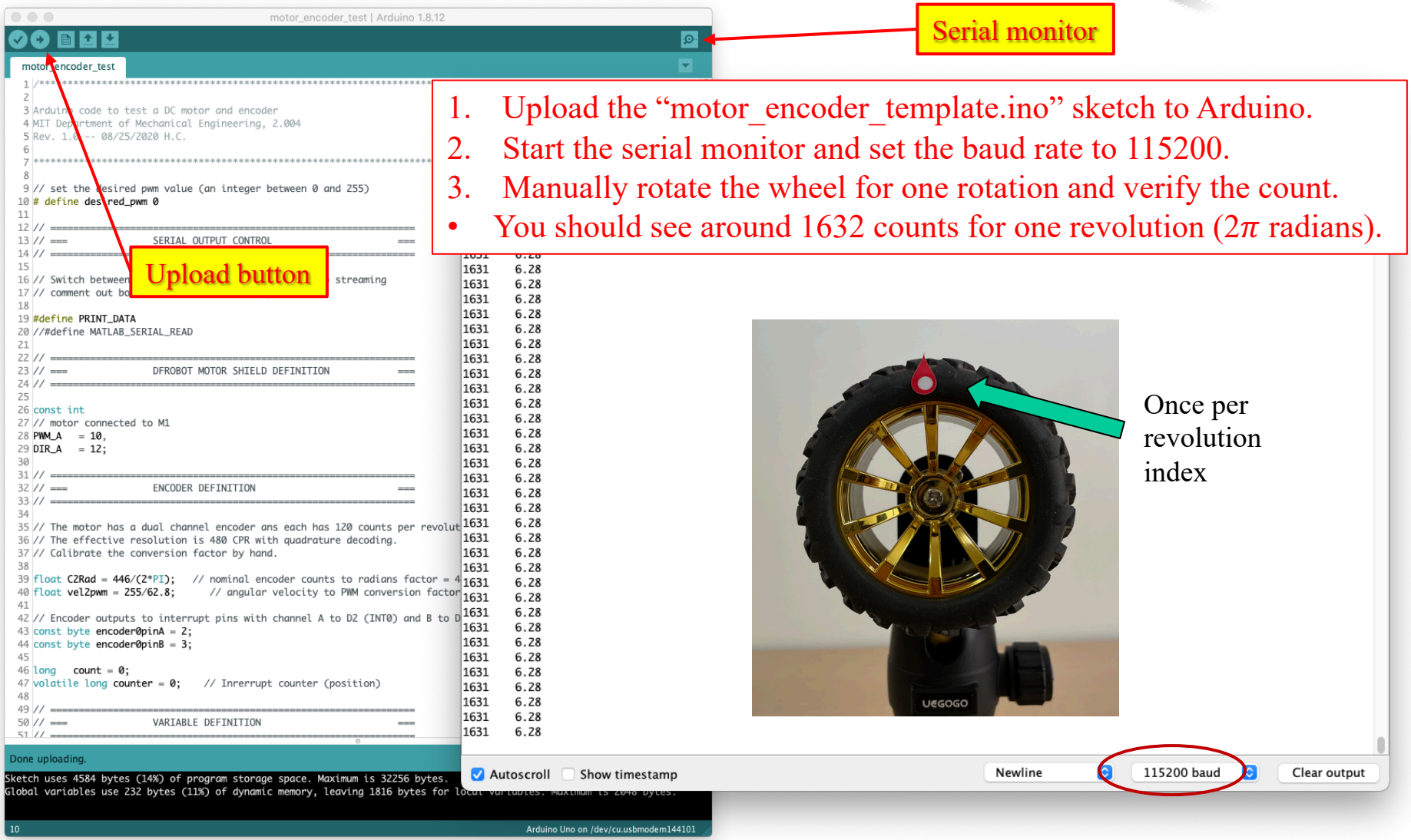
MS Windows

MacOS



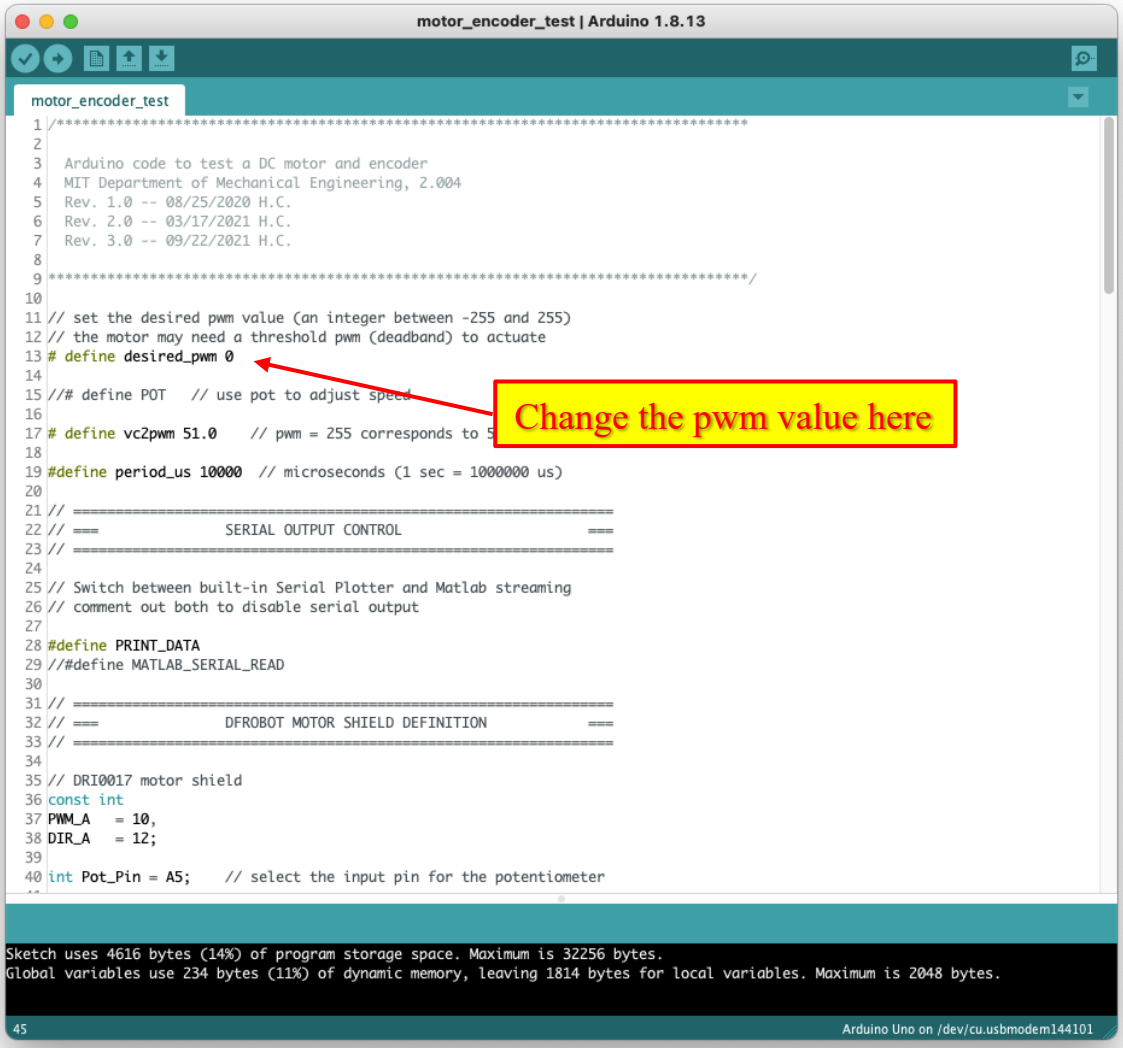
For Linux OS (such as Ubuntu), you may need to enable serial port access:
$ sudo chmod a+rw /dev/ttyACM0

Encoder Check

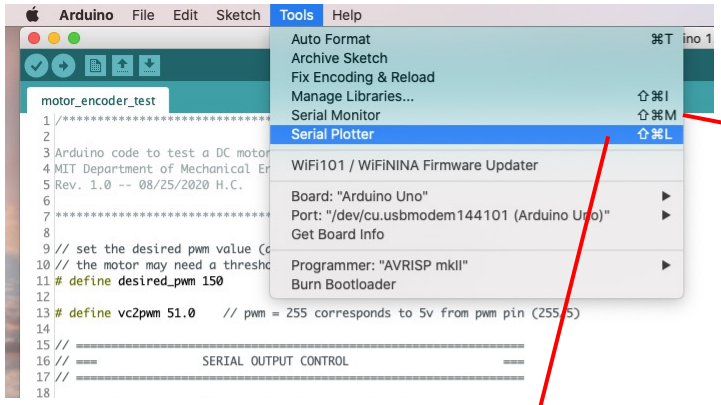


Motor Check

- Specify a pwm value (an integer between -255 and 255), e.g., 150, on line 13 and upload the sketch.
- Turn on the battery switch and the motor should spin.
- Uncomment line 15 so you can use the potentiometer to change the velocity and direction of the motor. Upload the sketch.
- Turn the potentiometer knob to make the wheel spin. If not spinning, check the code and/or the wire connections.



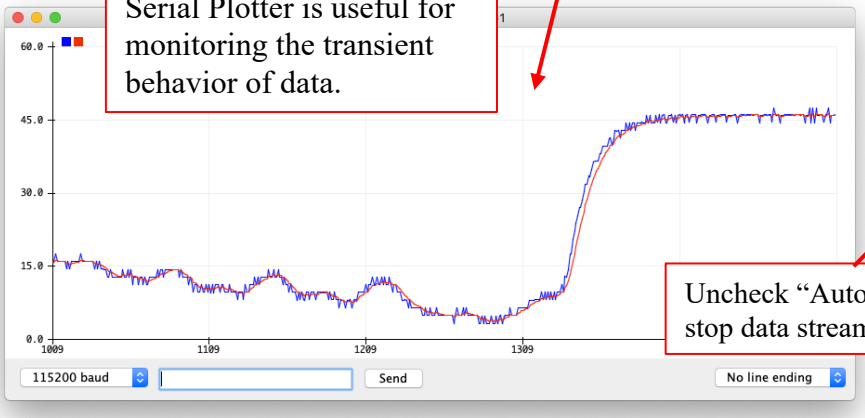
Monitoring/Capturing Serial Data



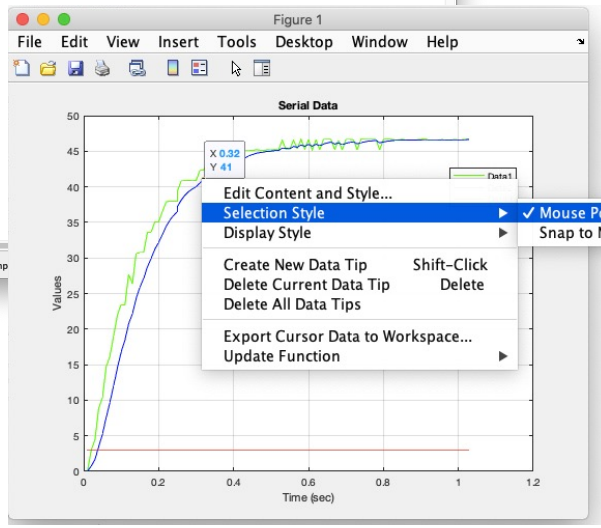
Serial Monitor is good for reading data values and to copy a portion of data record to clipboard.

Highlight a portion of data and copy and paste (may need to use keyboard shortcuts) to a variable in Matlab workspace, for example:
>> data =[*paste data here*];

Serial Plotter is useful for monitoring the transient behavior of data.



Uncheck "Autoscroll" to stop data streaming.



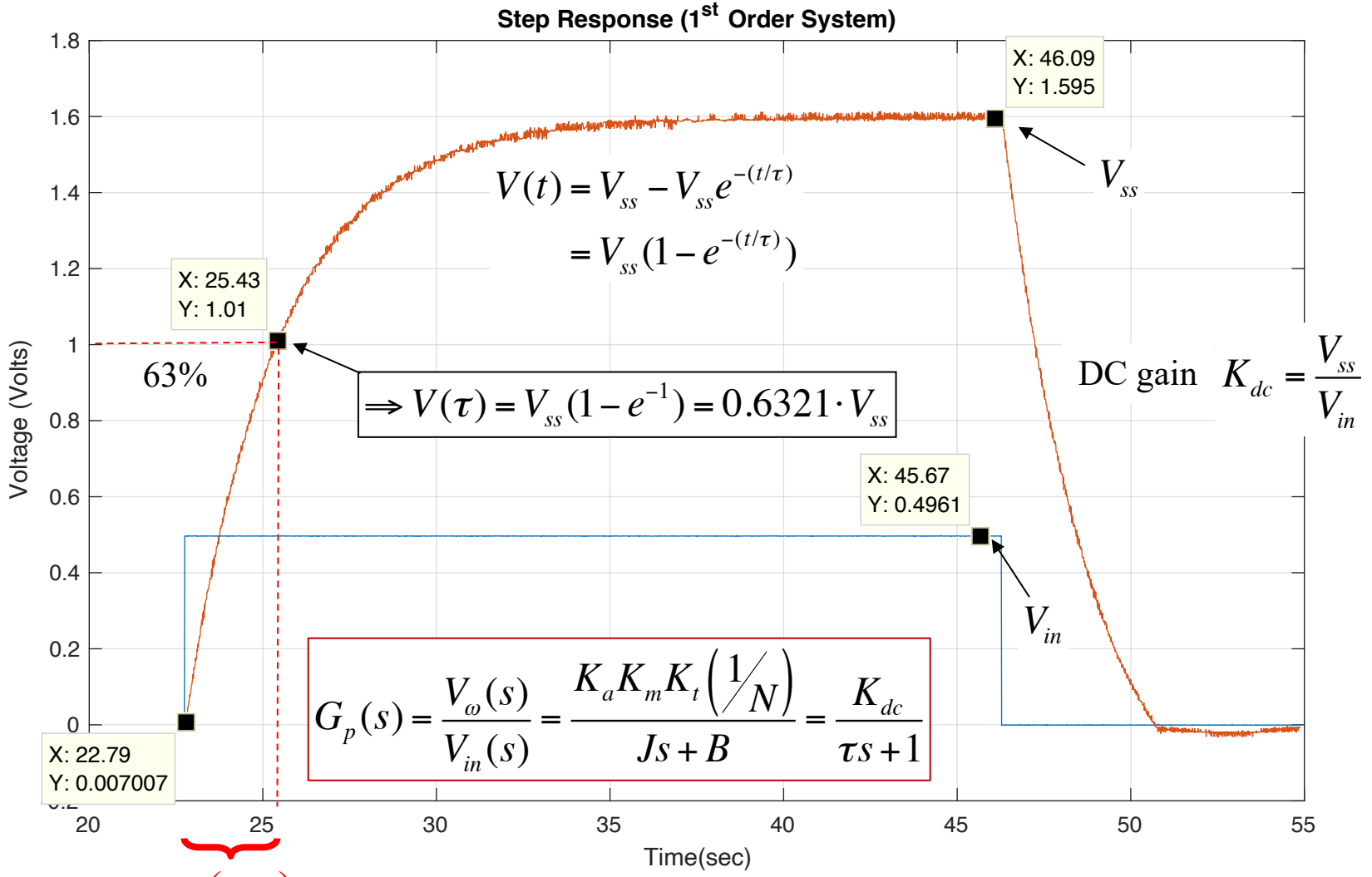
Neither "Serial Monitor" nor "Serial Plotter" is capable of saving data to a file.

The Matlab script "SerialRead.m" can be used to stream serial data directly into Matlab workspace for plotting and for post processing.

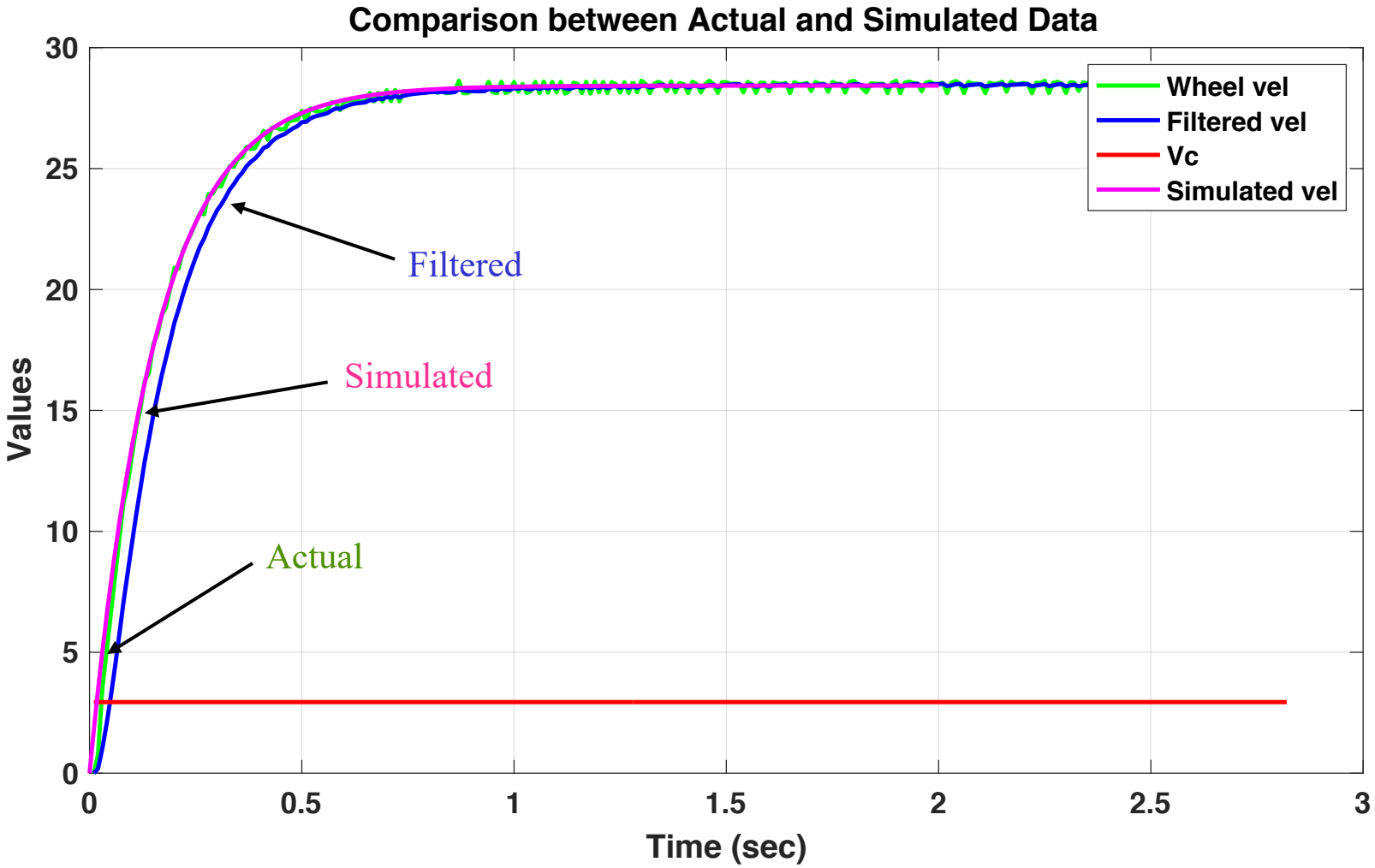
Create Transfer Function

- Set the desired PWM to 150
- Comment out #define POT
- Comment out #define PRINT_DATA and uncomment #define MATLAB_SERIAL_READ
- Upload the sketch to Arduino
- Open the Matlab script “SerialRead.m” and change the serial port name to the Arduino port name.
- Run the Matlab script and capture a step response.
- Estimate the transfer function from the response curve to relate the PWM pin voltage to the output velocity.

1st Order Step Response



Actual and Simulated Step Responses



Extra Credit Task

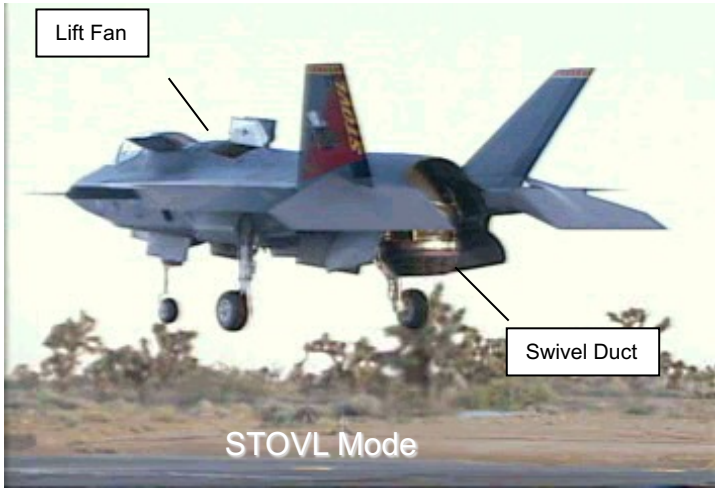
Simscape Simulation

- Determine the appropriate physical parameter values for the given Simscape model.
- Understand each individual block in the model.
- Run simulation and generate a step response.
- Fine tune some parameter values to match the actual response.

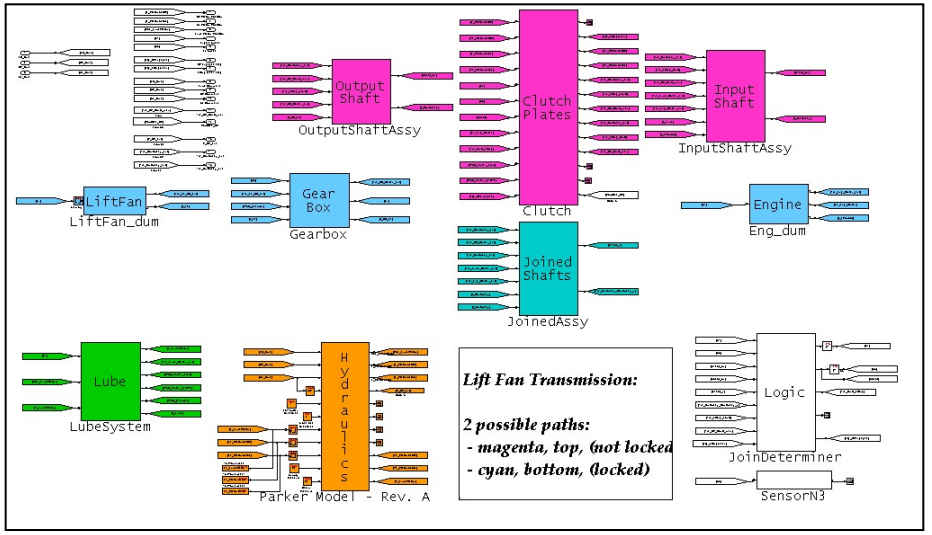
Why Model When One Can Experiment?

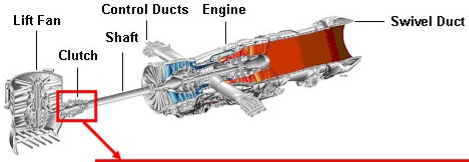
- **Model based design allows rapid iteration of system design to converge on the final product with desired specification:**
  - Significantly cheaper and easier than building actual prototype
  - Automated parameter sweep and optimization
- **In the "real-world", control engineers spend an overwhelming amount of effort on modeling before even beginning to build prototype.**

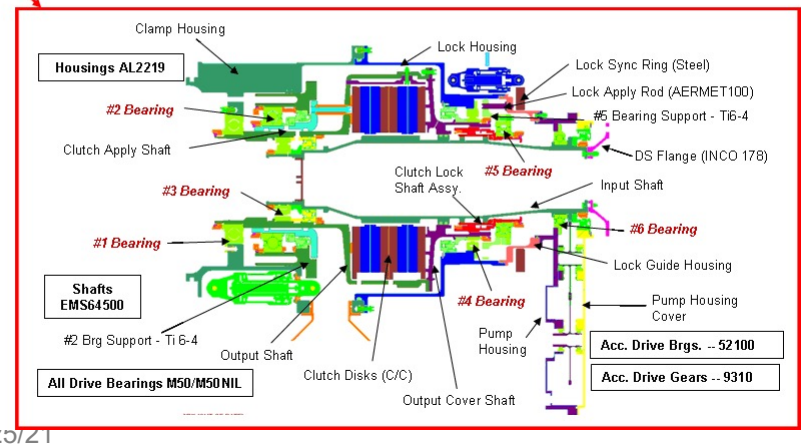
Example: F-35B STOVL Flight Control

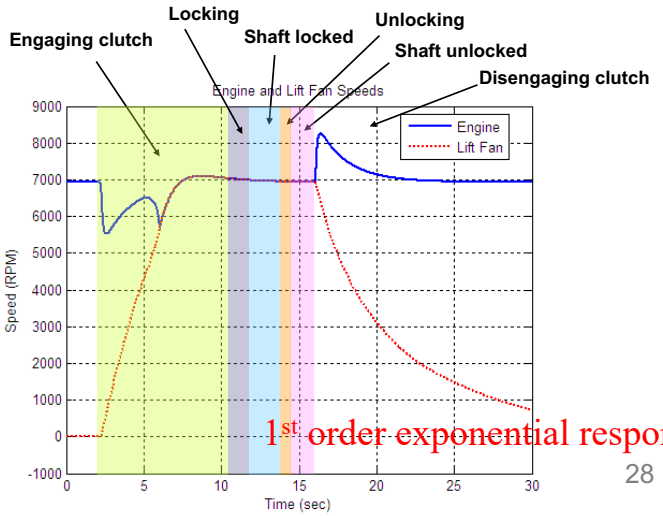


LiftFan Simulink Model









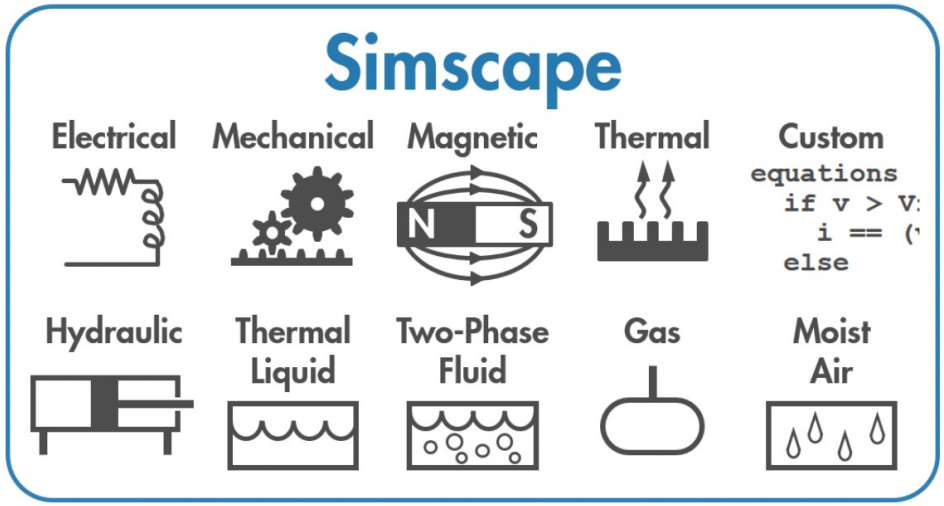
1st order exponential response

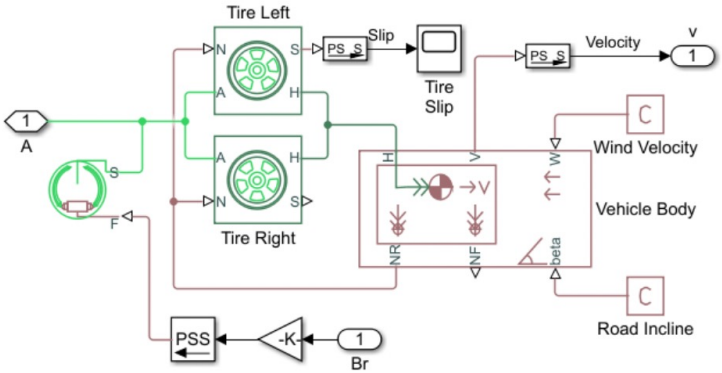
What is Simscape?

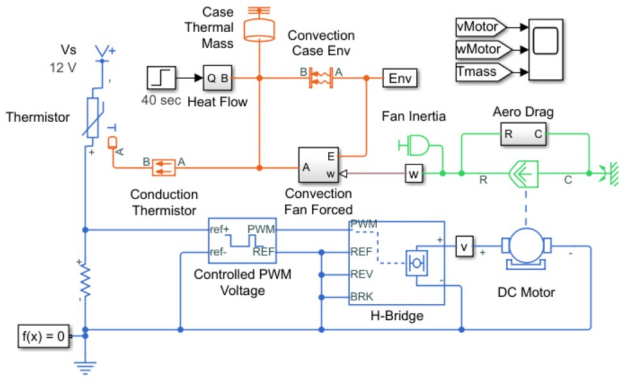
Simscape is a software package built on Simulink to directly model physical systems without the need for underlying equations.

Platform for physical Design

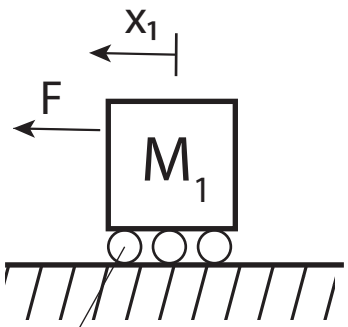
- Block diagram modeling
- Simulation of physical systems like mechanical, electrical, fluidic, etc.
- Automatically derives equations from physical block connections





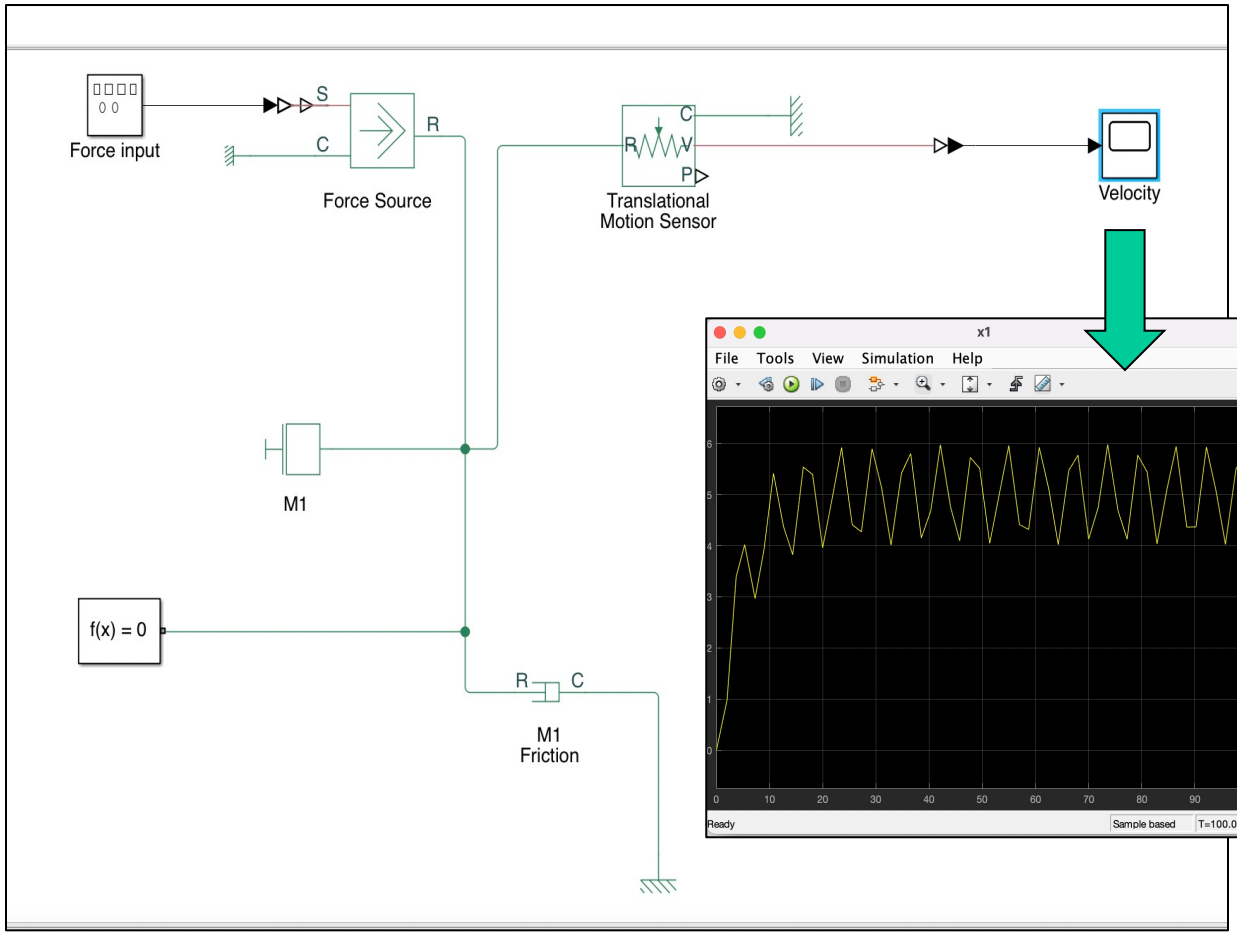


Example: a Single Mass on Wheels with Simscape

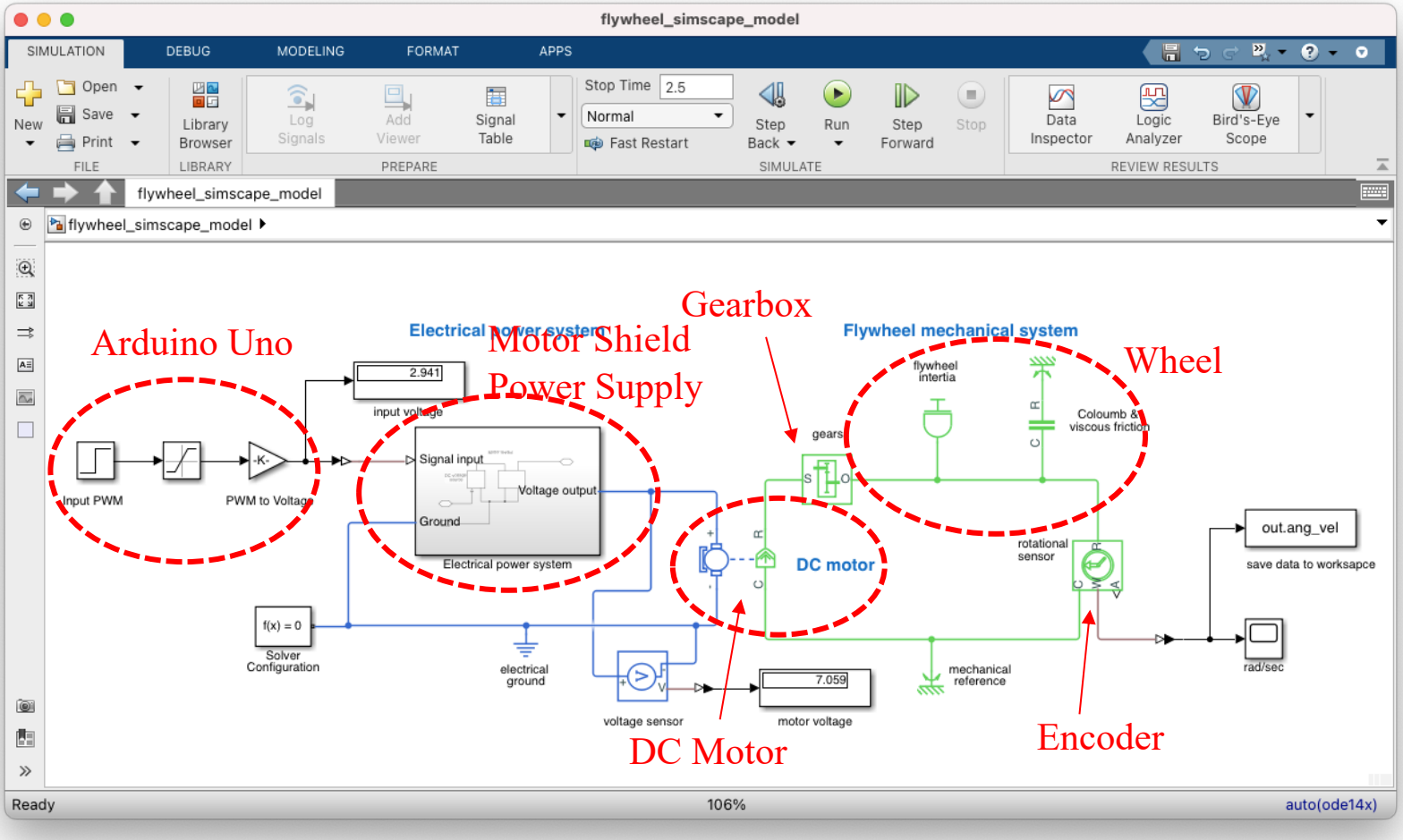


viscous damping with coefficient μ

Components:
1 Mass
1 damper



Flywheel System Simscape Model



Simplified DC Motor Model

$$\left[Js + \left(b + \frac{K_m K_v}{R}\right)\right]\Omega(s) = \frac{K_m}{R} V_s(s)$$

$$\Rightarrow \frac{\Omega(s)}{V_s(s)} = \frac{\left(\frac{K_m}{R}\right)}{Js + \left(b + \frac{K_m K_v}{R}\right)}$$

$K_m = Kv$ when both are in SI units

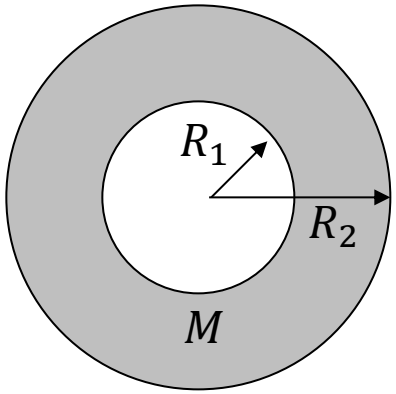
$$\frac{\Omega(s)}{V_s(s)} = \frac{\left(\frac{K_v}{R}\right)}{Js + \left(b + \frac{K_v^2}{R}\right)} = \frac{K}{\tau s + 1}$$

$$\tau = \frac{J}{\left(b + \frac{K_v^2}{R}\right)}$$

$$K = \frac{\left(\frac{K_v}{R}\right)}{\left(b + \frac{K_v^2}{R}\right)} = \frac{K_v}{bR + K_v^2}$$

Model Parameters

Moment of inertia:



$$\begin{cases} M = 0.05\ [kg] \\ R_1 = 0.02\ [m] \\ R_2 = 0.074\ [m] \end{cases}$$

$$J = \frac{M}{2}(R_1^2 + R_2^2) \approx 1.47\times10^{-4}\ [kg.m^2]$$

DC motor armature resistance (R) measurement:

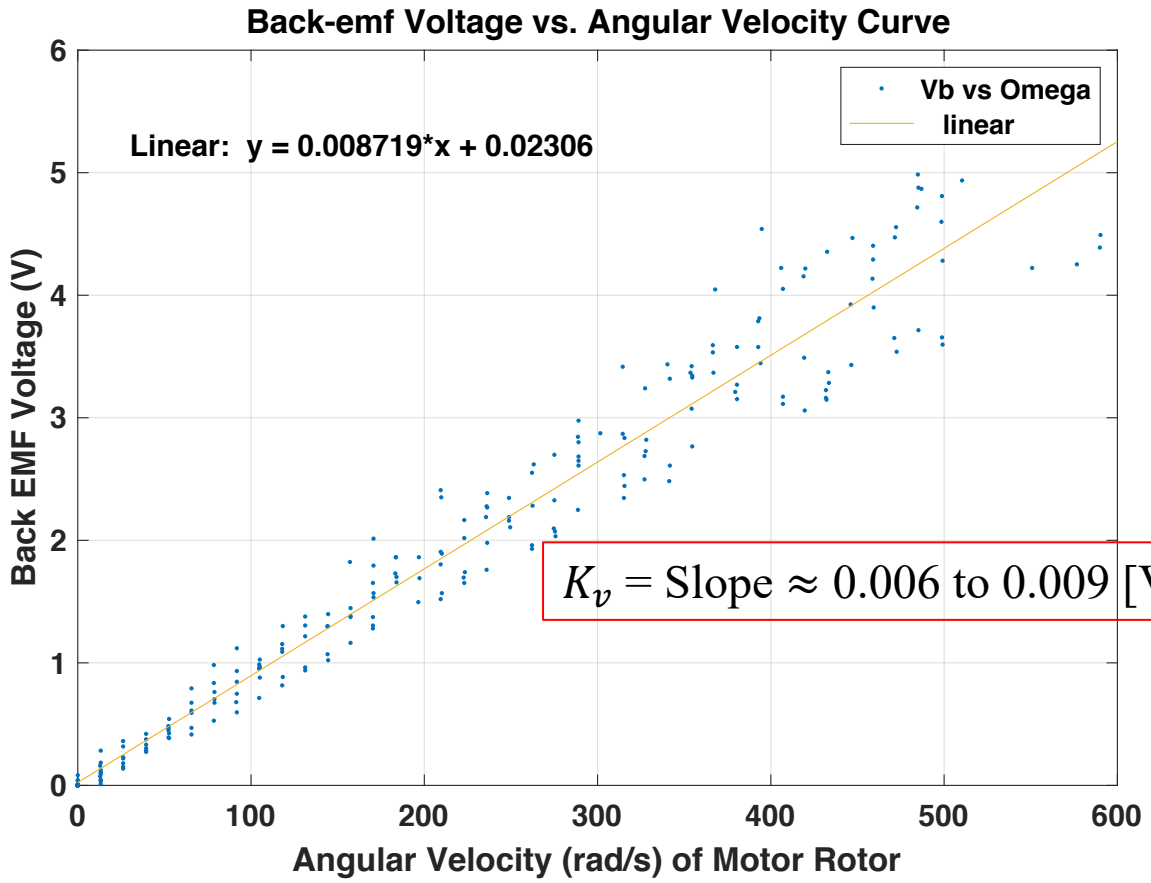
- Method 1: apply a small DC voltage (0.32 V) to the motor without actuating the motor shaft (to prevent back emf voltage generation) and measure the current (0.09 A). The resistance R can then be calculated as:

$$R = \frac{V}{I} = \frac{0.32}{0.09} \approx 3.5 \text{ ohms}$$

- Method 2: direct multi-meter measurement: $R \approx 3.2$ ohms

Motor Back-EMF Constant (K_v)

Acquire data by spinning flywheel by hand, and record both angular velocities and back-emf voltages from the motor terminals.

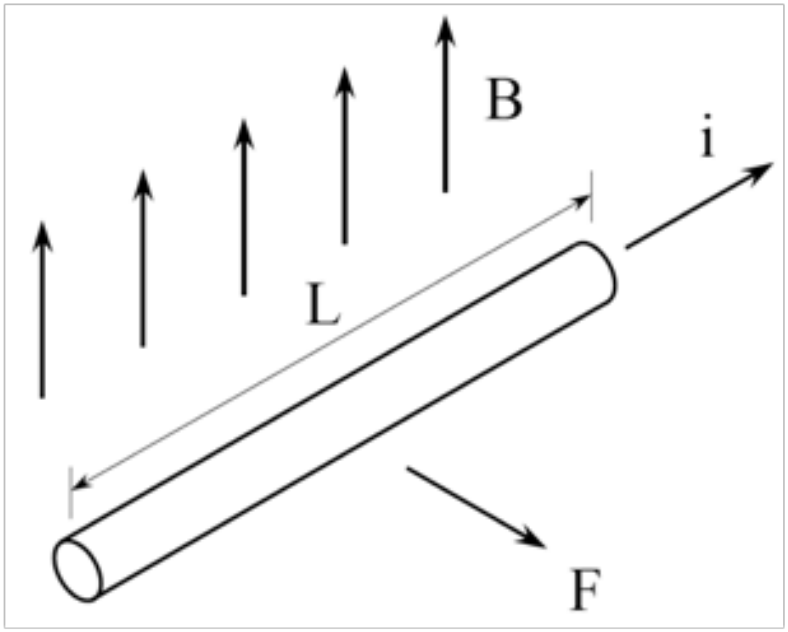


K_v = Slope ≈ 0.006 to 0.009 [V/(rad/s)]

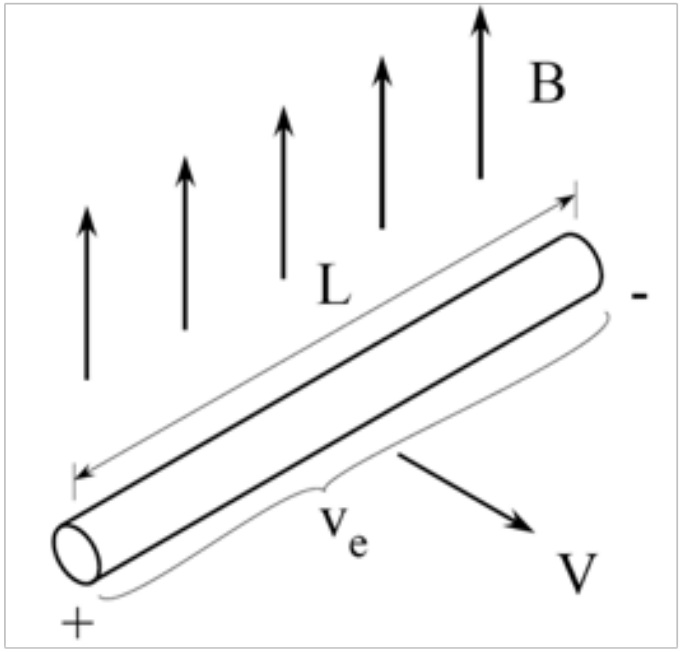
DC Motor

DC Motor Principle

Lorentz law:
magnetic field applies force to a current
(Lorentz force)

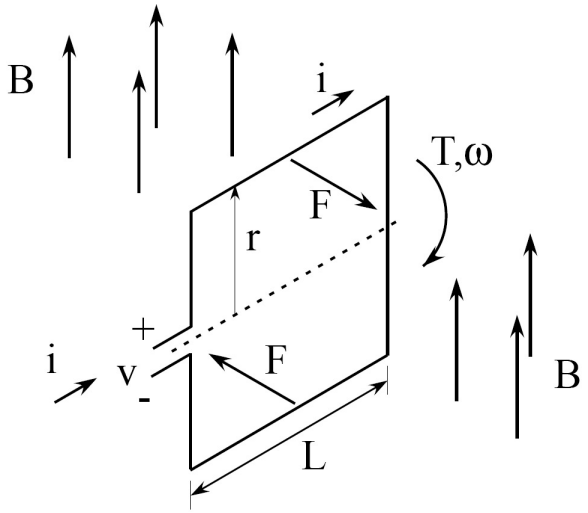


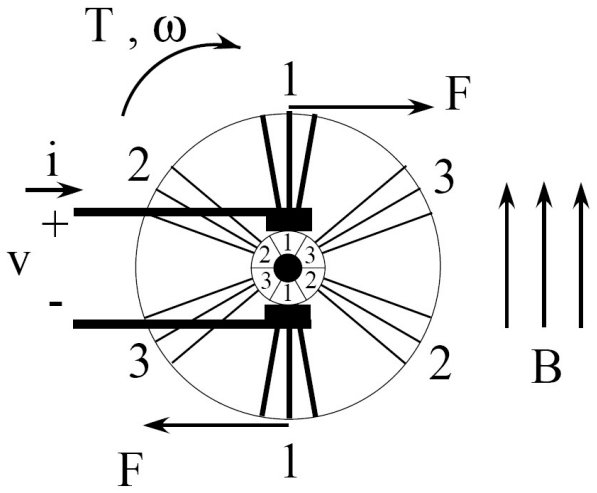
Faraday law:
moving in a magnetic field results
in potential (back EMF)



$$F = (\mathbf{i} \times \mathbf{B}) \cdot l = iBl \quad (\mathbf{i} \perp \mathbf{B}) \qquad v_e = \mathbf{V} \times \mathbf{B} \cdot l = VBl \quad (\mathbf{V} \perp \mathbf{B})$$

DC Motor Principle (Cont.)





multiple windings *N*: continuity of torque

$$T = 2Fr = 2(iBNl)r \qquad \text{(Lorentz law)}$$

$$v_e = 2VBNl = 2(\omega r)BNl \qquad \text{(Faraday law)}$$

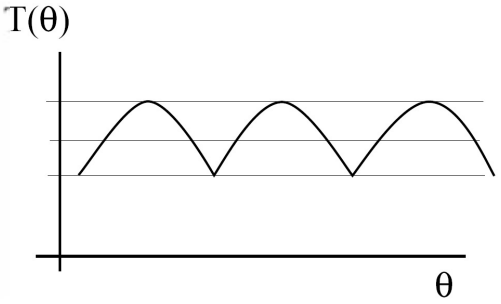
or

$$T = K_m i$$

$$v_e = K_v \omega$$

where

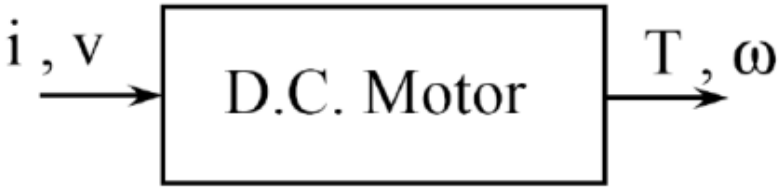
- $K_m \equiv 2BNlr$ torque constant
- $K_v \equiv 2BNlr$ back-emf constant



DC Motor as A Transducer

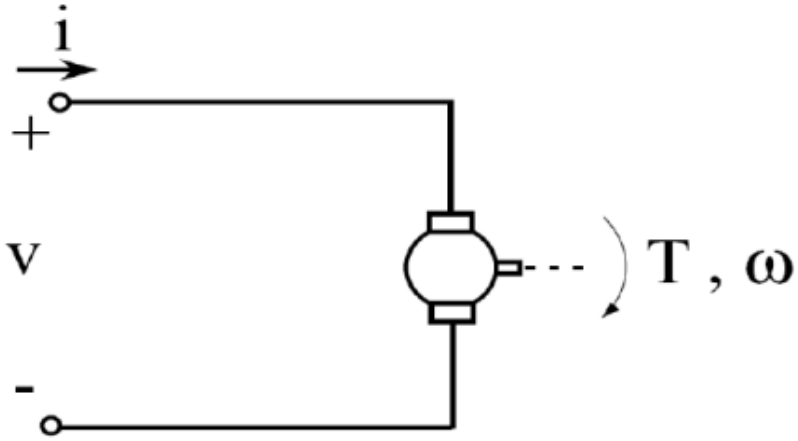
DC motor is an electromechanical system that may have a time constant close to the plant.

A transducer converts energy from one domain (e.g., electrical) to another (e.g., mechanical), or vice versa.

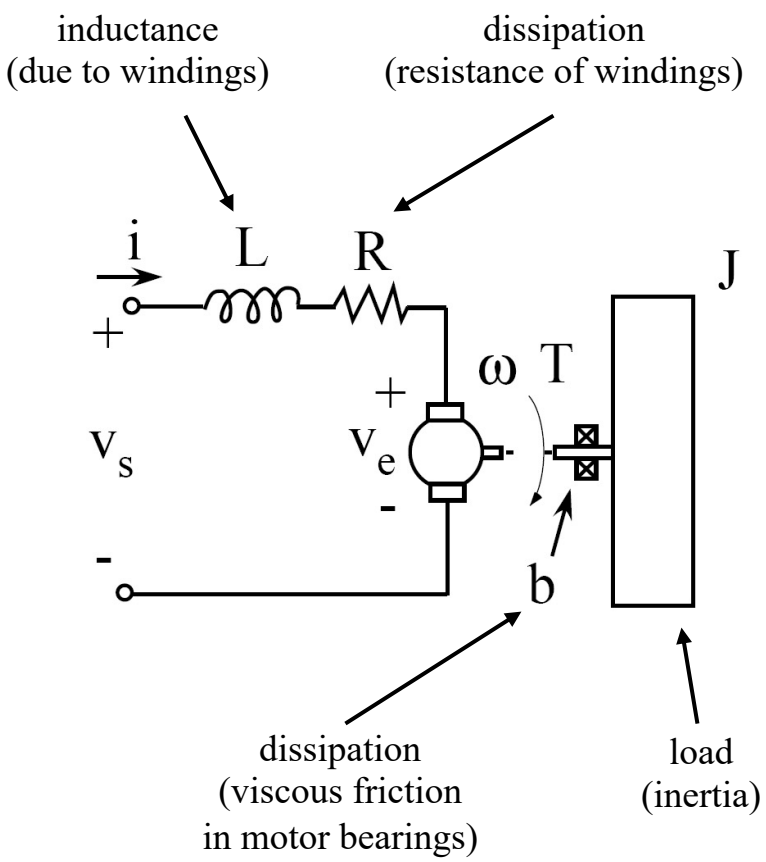


$$P_{in} = P_{out}$$

$$i(t) * v(t) = T(t) * \omega(t)$$



DC Motor with Mechanical Load



K_m : motor torque constant

K_v : motor speed (back EMF) constant

Equation of motion – Electrical

KVL: $v_s - v_L - v_R - v_e = 0$

$$\Rightarrow v_s - L\frac{di}{dt} - Ri - K_v\omega = 0$$

Equation of motion – Mechanical

Torque Balance: $T = T_b + T_J$

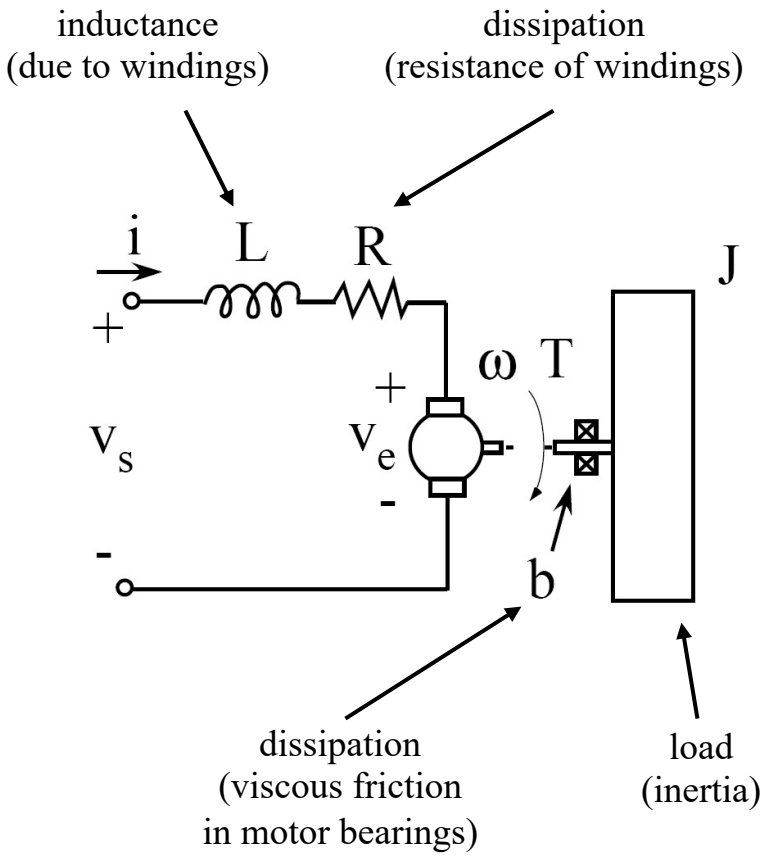
$$\Rightarrow K_m i - b\omega = J\frac{d\omega}{dt}$$

Combined equations of motion

$$L\frac{di}{dt} + Ri + K_v\omega = v_s$$

$$J\frac{d\omega}{dt} + b\omega = K_m i$$

DC Motor with Mechanical Load



Equation of motion – Electrical

KVL: $V_s(s) - V_L(s) - V_R(s) - V_e(s) = 0$

$$V_s(s) - LsI(s) - RI(s) - K_v\Omega(s) = 0$$

Equation of motion – Mechanical

Torque Balance: $T(s) = T_b(s) + T_J(s)$

$$K_m I(s) - b\Omega(s) = Js\Omega(s)$$

Combined equations of motion

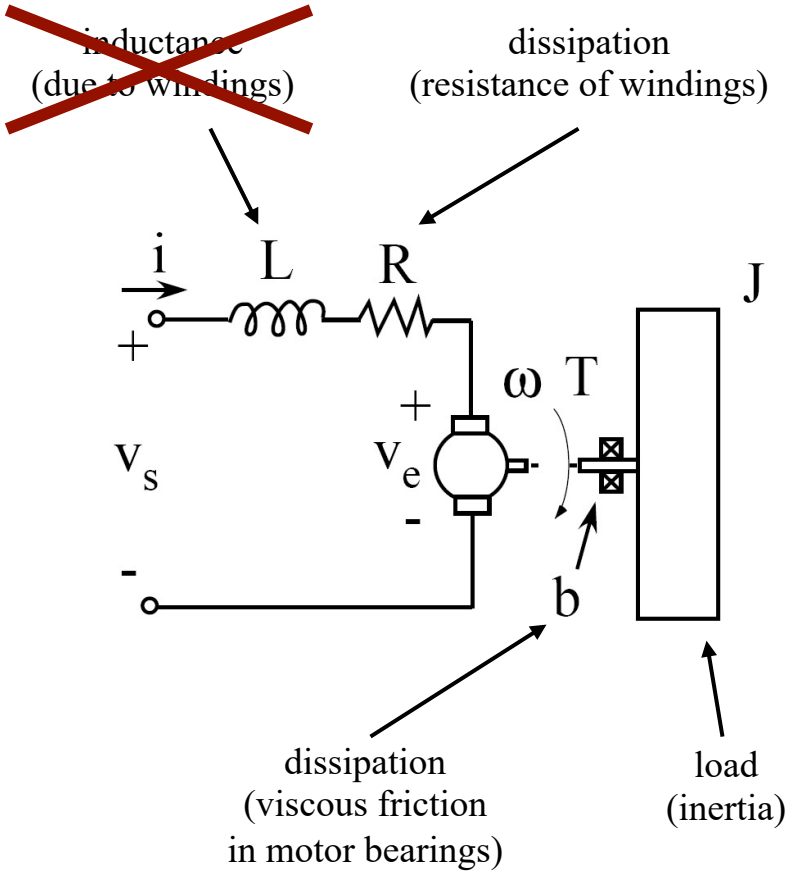
$$LsI(s) + RI(s) + K_v\Omega(s) = V_s(s)$$

$$Js\Omega(s) + b\Omega(s) = K_m I(s)$$

$$\Rightarrow \left[(Ls+R)\left(\frac{Js+b}{K_m}\right) + K_v\right]\Omega(s) = V_s(s)$$

$$\Rightarrow \left[\frac{LJ}{R}s^2 + \left(\frac{Lb}{R} + J\right)s + \left(b + \frac{K_m K_v}{R}\right)\right]\Omega(s) = \frac{K_m}{R}V_s(s)$$

DC Motor with Mechanical Load



Neglecting the inductance (why?)

$$L \approx 0$$

$$\Rightarrow \left[Js + \left(b + \frac{K_m K_v}{R}\right)\right] \Omega(s) = \frac{K_m}{R} V_s(s)$$

This is our familiar 1st–order system!

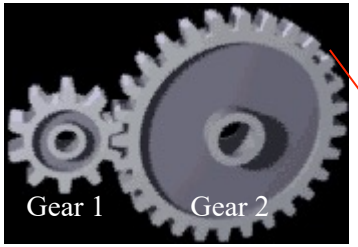
If we are given step input $v_s(t) = V_0 u(t)$
$\Rightarrow$ we already know the step response

$$\omega(t) = \frac{K_m}{R} V_0 \left(1 - \mathrm{e}^{-t/\tau}\right) u(t),$$

where now the time constant is

$$\tau = \frac{J}{\left(b + \frac{K_m K_v}{R}\right)}.$$

Gear Ratio and Power Conservation



Gear Ratio N: $\frac{n_1}{n_2} = \frac{r_1}{r_2} = \frac{\Omega_2}{\Omega_1} = \frac{T_1}{T_2} = \frac{F_1^T r_1}{F_2^T r_2}$

$$\frac{n_1}{n_2} = \frac{44}{180}$$

Unit Conversion:

$$rpm = \frac{2\pi}{60}(rad/s)$$

$$N = \frac{kg \times m}{s^2}$$

$$V(voltage) = \frac{kg \times m^2}{s^3 \times A}$$

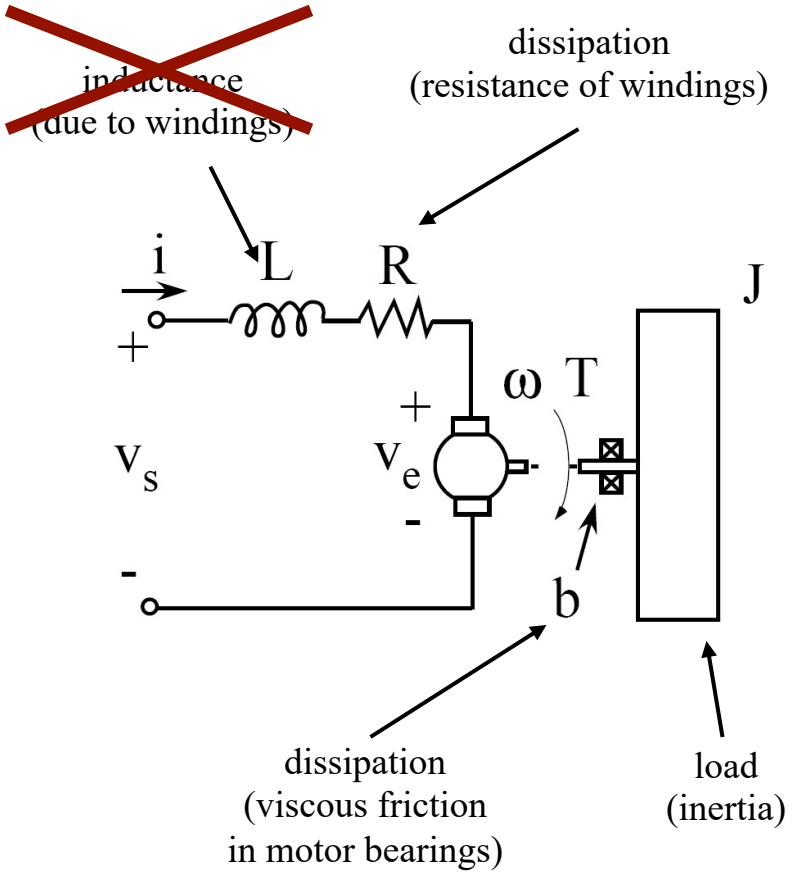
Power Conservation:

$$P_{mechanical} = P_{electrical}$$

$$P_{mechanical}(t) = T(t) \times \Omega(t) = K_m \times i(t) \times \Omega(t)$$

$$P_{electrical}(t) = v_b(t) \cdot i(t) = K_v \cdot \Omega(t) \cdot i(t)$$

DC Motor with Mechanical Load



K_m :	motor torque constant
K_v :	motor speed (back EMF) constant

Neglecting the inductance…

$$L \approx 0$$

$$\Rightarrow \left[Js + \left(b + \frac{K_m K_v}{R}\right)\right] \Omega(s) = \frac{K_m}{R} V_s(s)$$

This is our familiar 1st–order system!

If we are given step input $v_s(t) = V_0 u(t)$
$\Rightarrow$ we already know the step response

$$\omega(t) = \frac{K_m}{R} V_0 \left(1 - \mathrm{e}^{-t/\tau}\right) u(t),$$

where now the time constant is

$$\tau = \frac{J}{\left(b + \frac{K_m K_v}{R}\right)}.$$