



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:**

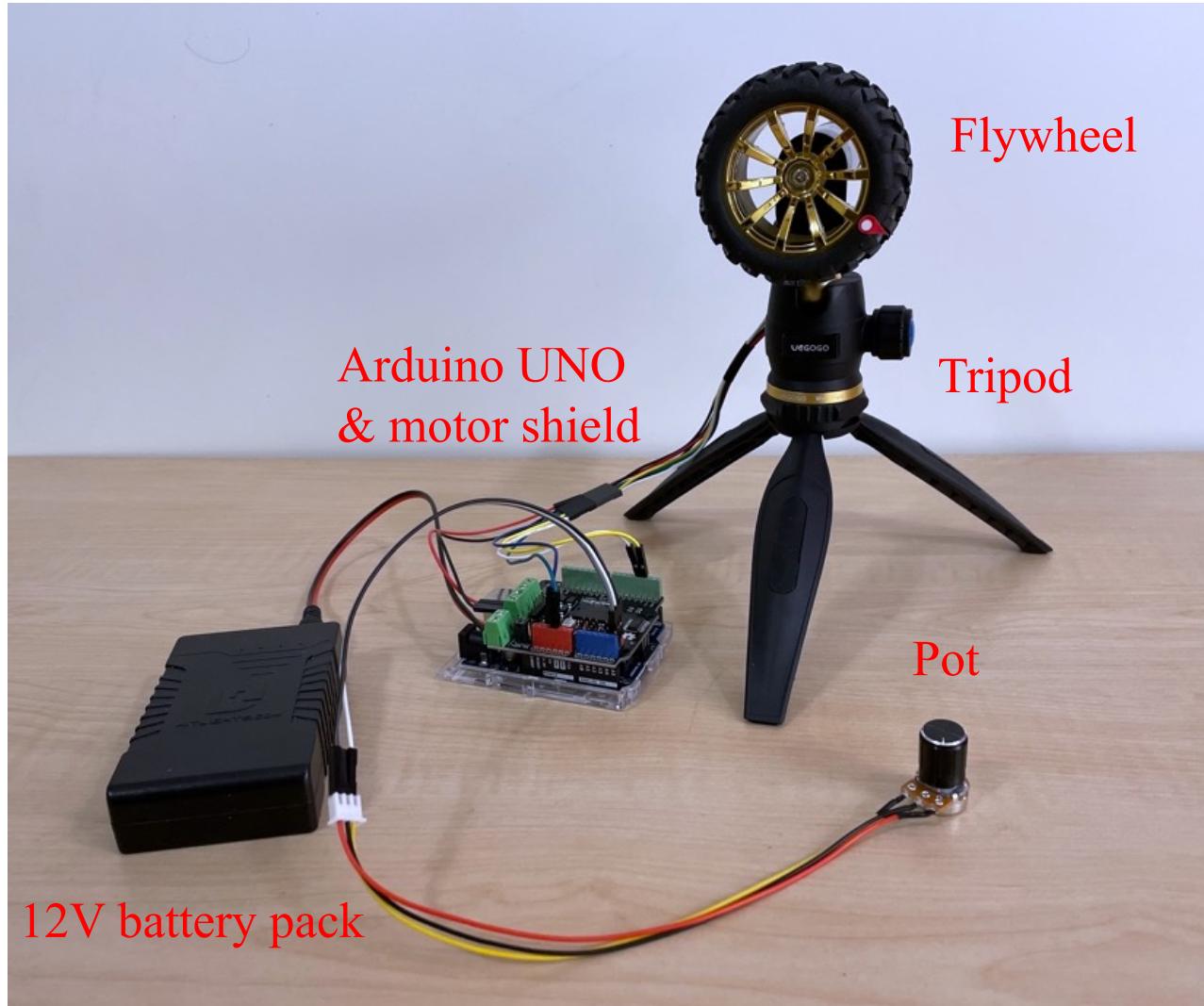
0 – No show / No report received

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

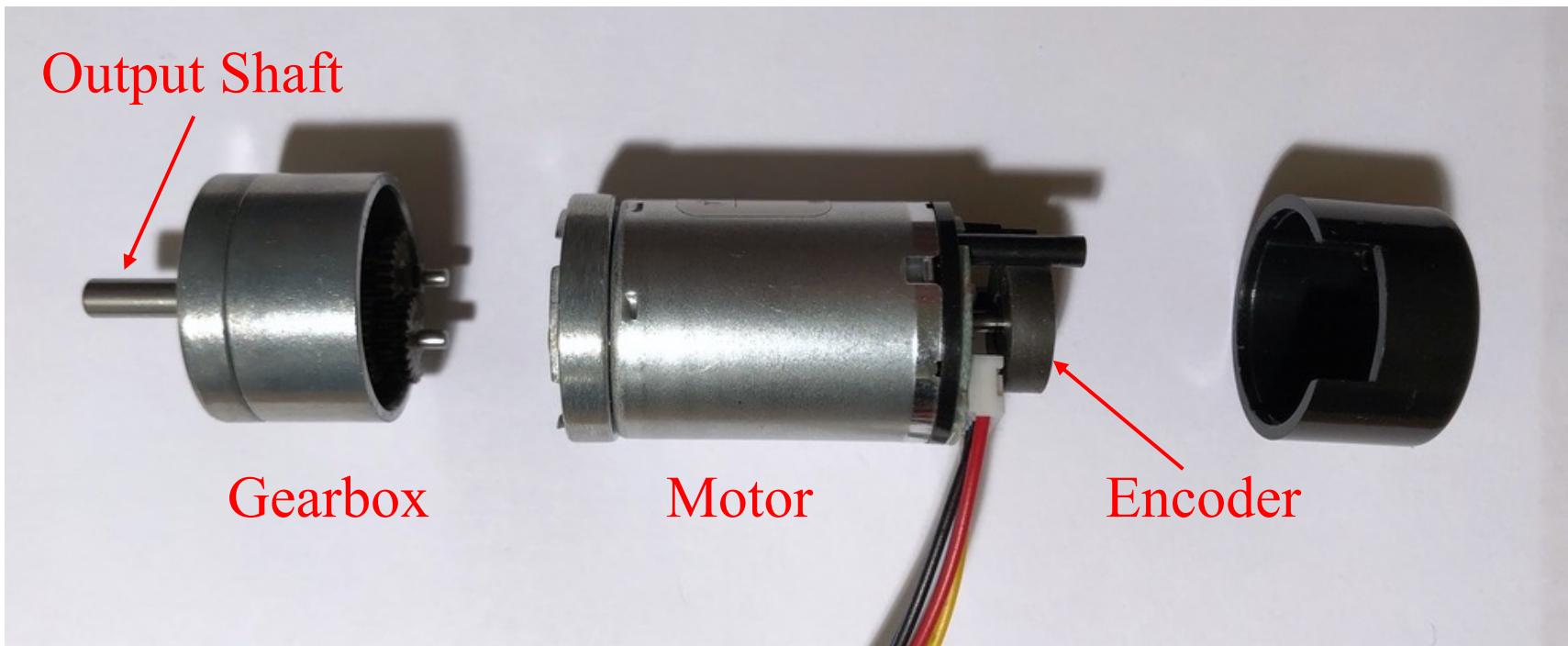


12V DC Gear Motor with Encoder

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

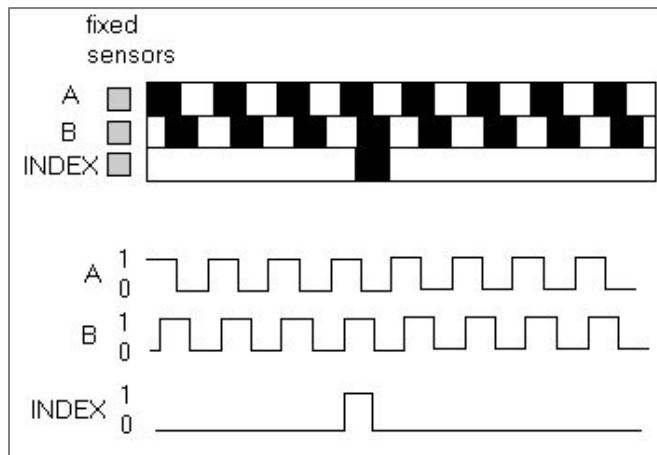


Position Sensing Using Encoder

- 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:

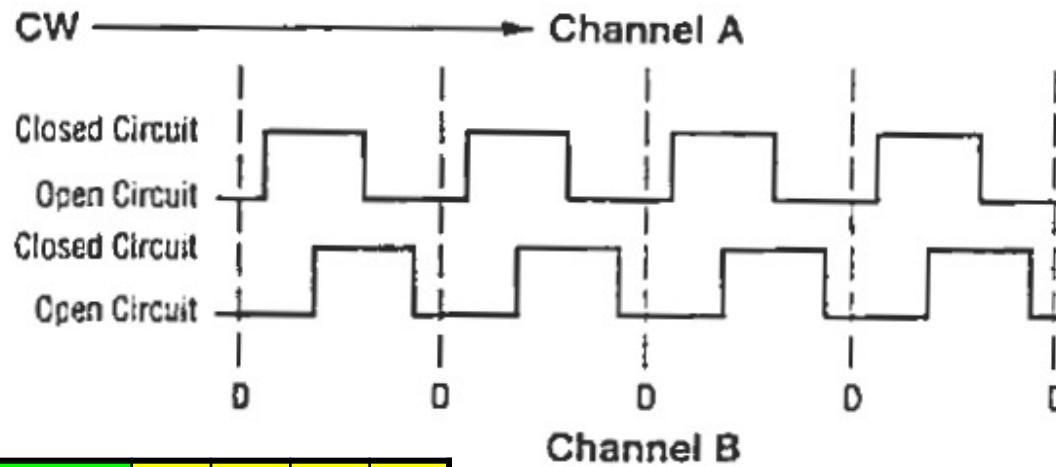
- Absolute:** Unique code for each shaft position.
- Incremental (Relative):** Only the relative position of the shaft is known. No absolute “0” position. An “INDEX” signal can be added to provide “0” position of a shaft.



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

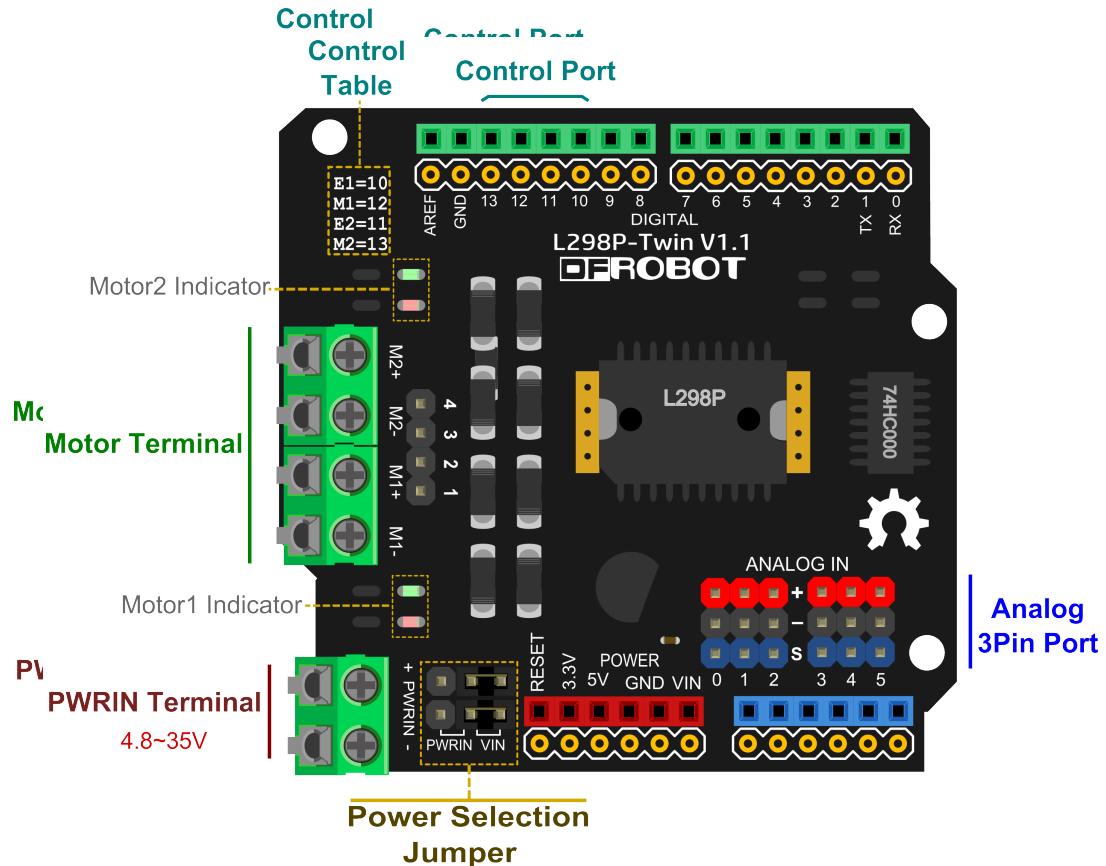


- For this implementation two Interrupt Service Routines (ISR) are used to detect and process the encoder signals. An interrupt service routine is a software routine that hardware invokes in response to an interrupt.
- The Arduino UNO has two hardware interrupt pins: D2 and D3. We connect encoder channel 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 delivers 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 ($T=75^{\circ}\text{C}$)
 - Driven Structure: Dual full-bridge driver



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

Driving The DC Motor

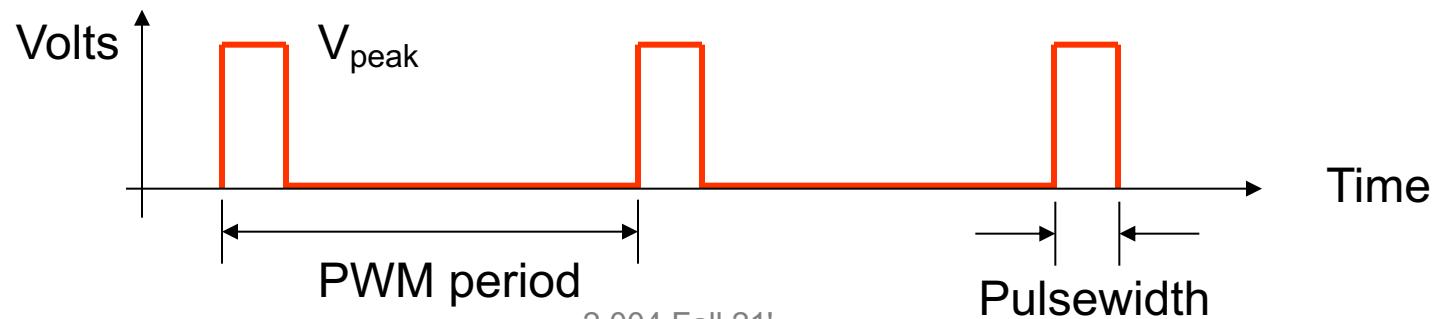


- Low cost microcontrollers such as the Arduino Uno do not include a Digital-to-Analog Converter (DAC), instead they use digital pins to generate a Pulse Width Modulation (PWM) signal to mimic an analog signal. The Uno's default PWM frequency is 490.2 Hz (period = 0.002 seconds) for the pins we use. Note that the PWM frequency can be changed from about 30 Hz to 30 kHz (<https://www.technophiles.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

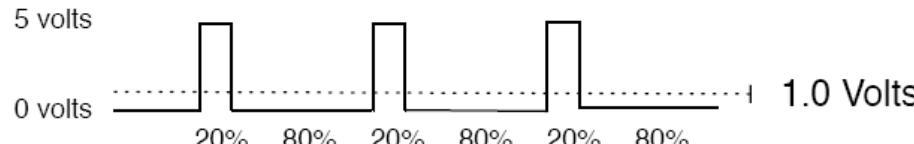
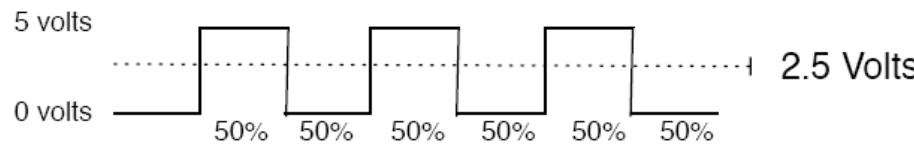
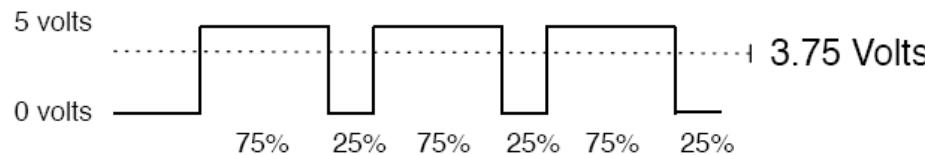


Pulse Width Modulation (PWM)

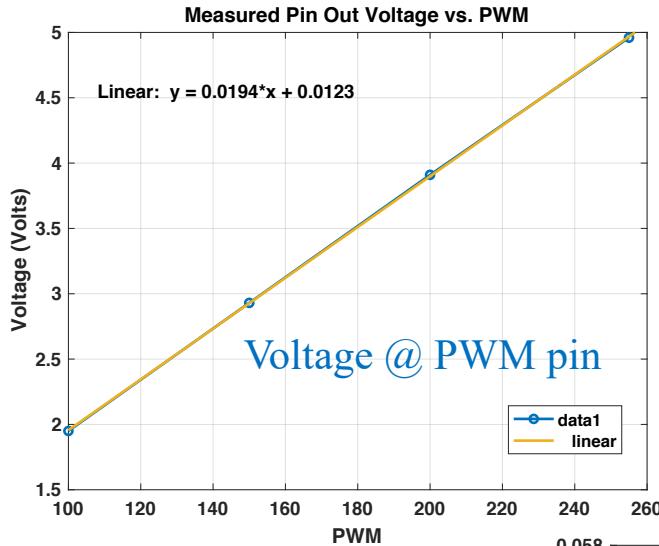
- Can be used as a substitute for analog output (high frequency switching is filtered out by the physical systems and what is left is the mean voltage).
- Applications include: lamp dimmers, motor speed control, power supplies,...

Output voltage is averaged from on vs. off time

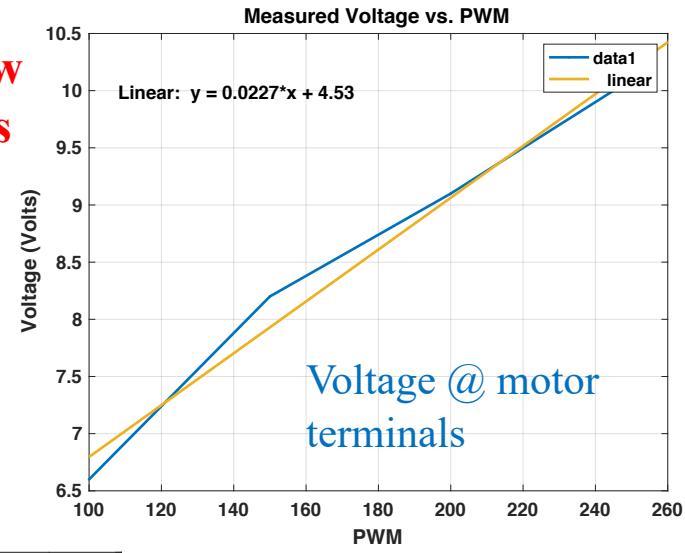
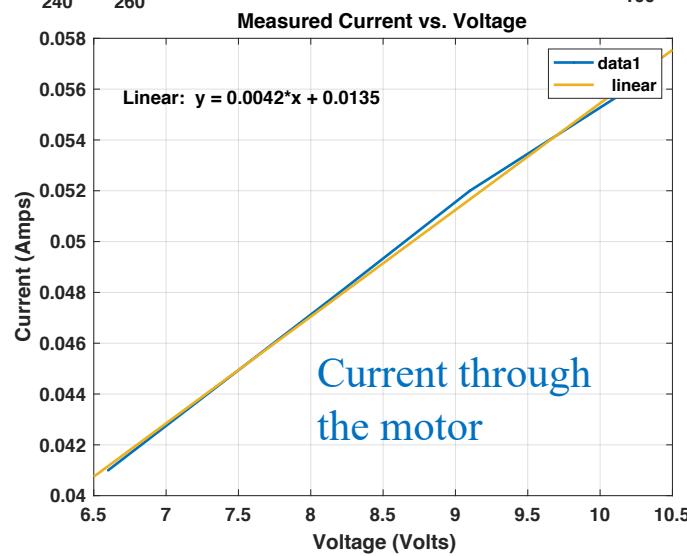
$$\text{output_voltage} = (\text{on_time} / \text{off_time}) * \text{max_voltage}$$



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:

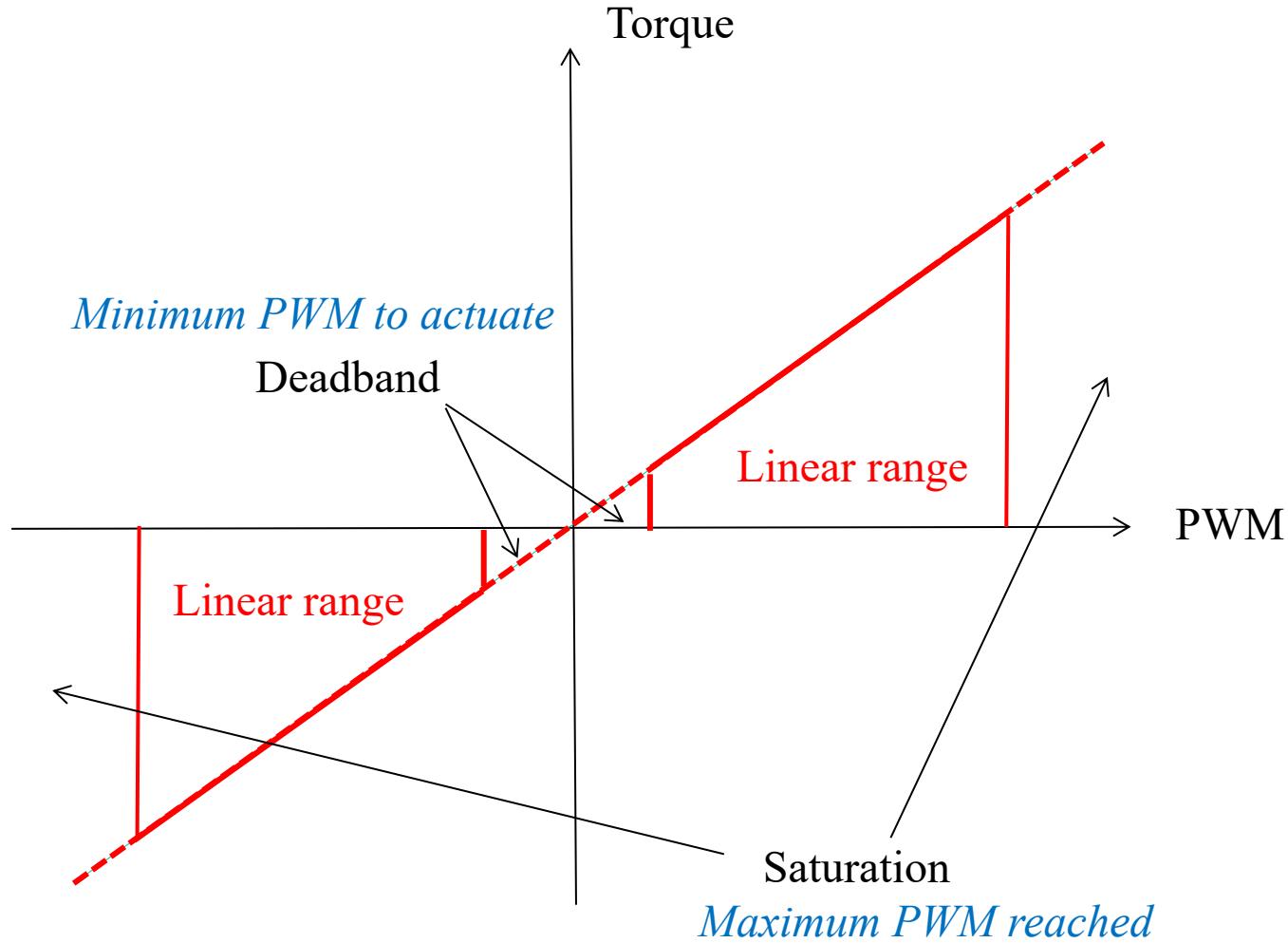
$$\hat{y}(t) = \alpha \cdot y(t) + (1 - \alpha) \cdot \hat{y}(t - \Delta T), \quad 0 \leq \alpha \leq 1$$

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

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

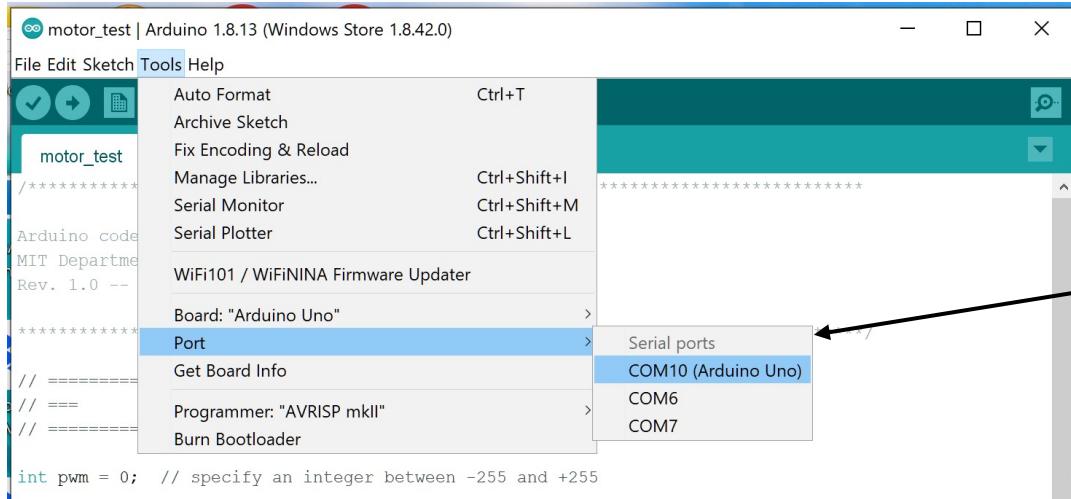


Arduino Software

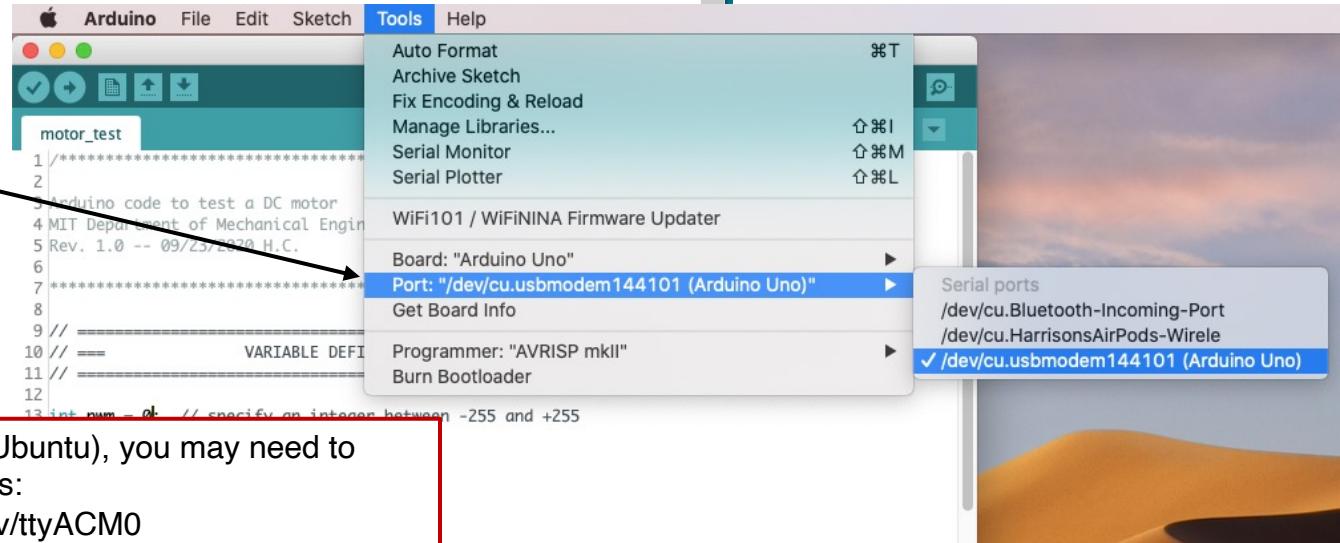


- Download and install the Arduino IDE (<https://www.arduino.cc/en/Main/Software>) if you have not done so already. The latest version is 1.8.16.
- Connect the Arduino UNO board to your computer with the USB cable.
- Open the “motor_encoder_template.ino” sketch, and select the proper Arduino board and 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

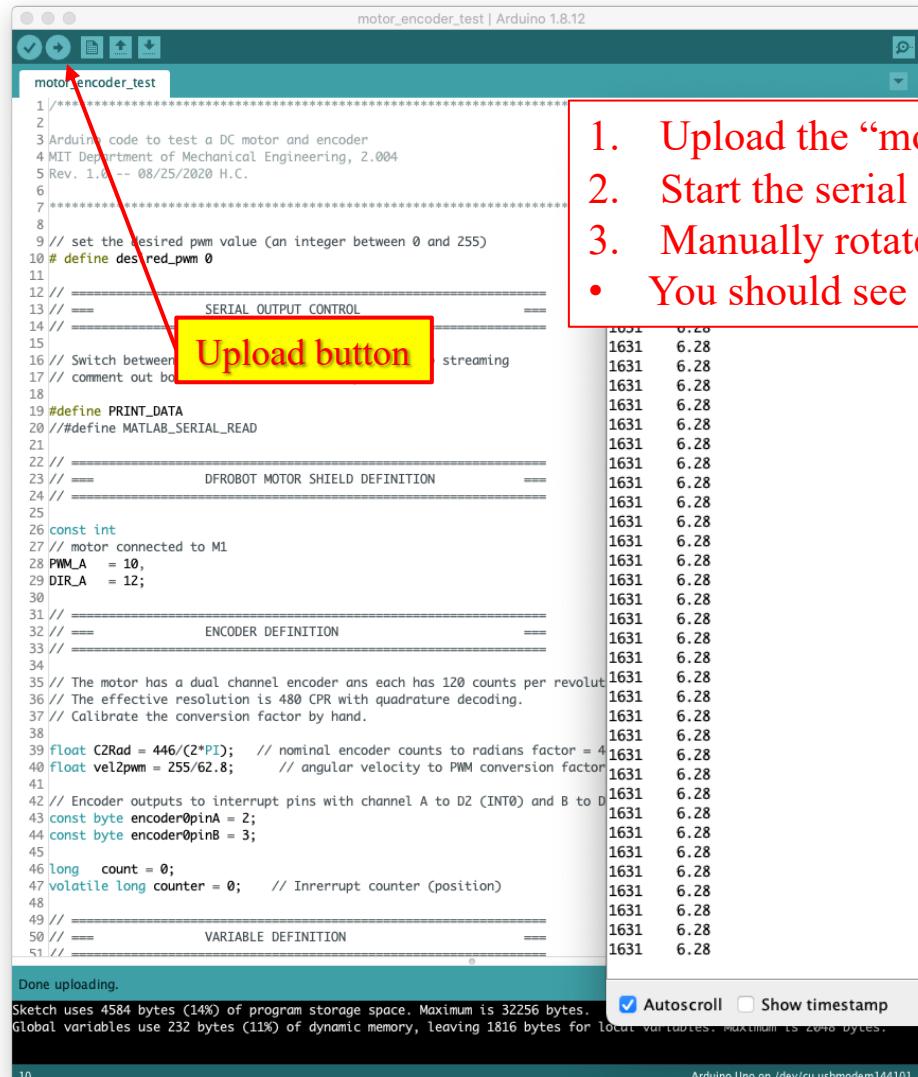
For Linux OS (such as Ubuntu), you may need to enable serial port access:
\$ sudo chmod a+r /dev/ttyACM0

The image shows two screenshots of the Arduino IDE interface. The top screenshot is for MS Windows, and the bottom one is for MacOS. Both show the 'Tools' menu open with the 'Port' option selected, revealing a dropdown menu of available serial ports. In the Windows screenshot, 'COM10 (Arduino Uno)' is selected. In the MacOS screenshot, '/dev/cu.usbmodem144101 (Arduino Uno)' is selected. A red box highlights the terminal window in the bottom screenshot, which contains the code for a DC motor test sketch.

```
motor_test | Arduino 1.8.13 (Windows Store 1.8.42.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries...
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
int pwm = 0; // specify an integer between -255 and +255
```

```
Arduino File Edit Sketch Tools Help
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Manage Libraries...
Serial Monitor ⌘I
Serial Plotter ⌘M
WiFi101 / WiFiINA Firmware Updater
Board: "Arduino Uno"
Port: "/dev/cu.usbmodem144101 (Arduino Uno)" ▶
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
1 // ****
2
3 Arduino code to test a DC motor
4 MIT Department of Mechanical Engineering
5 Rev. 1.0 -- 09/25/2009 H.C.
6
7 ****
8
9 // === VARIABLE DEFINITION
10 // ===
11 // ===
12 int num = 0; // specify an integer between -255 and +255
```

Encoder Check



motor_encoder_test | Arduino 1.8.12

```
1 // ****
2 //
3 Arduino code to test a DC motor and encoder
4 MIT Department of Mechanical Engineering, 2.004
5 Rev. 1.0 -- 08/25/2020 H.C.
6
7 ****
8
9 // set the desired pwm value (an integer between 0 and 255)
10 #define desired_pwm 0
11
12 // ===== SERIAL OUTPUT CONTROL =====
13 // === comment out below if not using ===
14 // ===
15 // Switch between
16 // PRINT_DATA
17 // comment out below if not using
18
19 #define PRINT_DATA
20 // #define MATLAB_SERIAL_READ
21
22 // ===== DFROBOT MOTOR SHIELD DEFINITION =====
23 // ===
24 // ===
25
26 const int
27 // motor connected to M1
28 PWM_A = 10,
29 DIR_A = 12;
30
31 // ===== ENCODER DEFINITION =====
32 // ===
33 // ===
34
35 // The motor has a dual channel encoder and each has 120 counts per revolution
36 // The effective resolution is 480 CPR with quadrature decoding.
37 // Calibrate the conversion factor by hand.
38
39 float C2Rad = 446/(2*PI); // nominal encoder counts to radians factor = 446/6480 = 0.068
40 float vel2pwm = 255/62.8; // angular velocity to PWM conversion factor
41
42 // Encoder outputs to interrupt pins with channel A to D2 (INT0) and B to D3 (INT1)
43 const byte encoderPinA = 2;
44 const byte encoderPinB = 3;
45
46 long count = 0;
47 volatile long counter = 0; // Interrupt counter (position)
48
49 // ===== VARIABLE DEFINITION =====
50 // ===
51 // ===
```

Done uploading.

Sketch uses 4584 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 232 bytes (11%) of dynamic memory, leaving 1816 bytes for local variables. Maximum is 2040 bytes.

Autoscroll Show timestamp

Newline Clear output

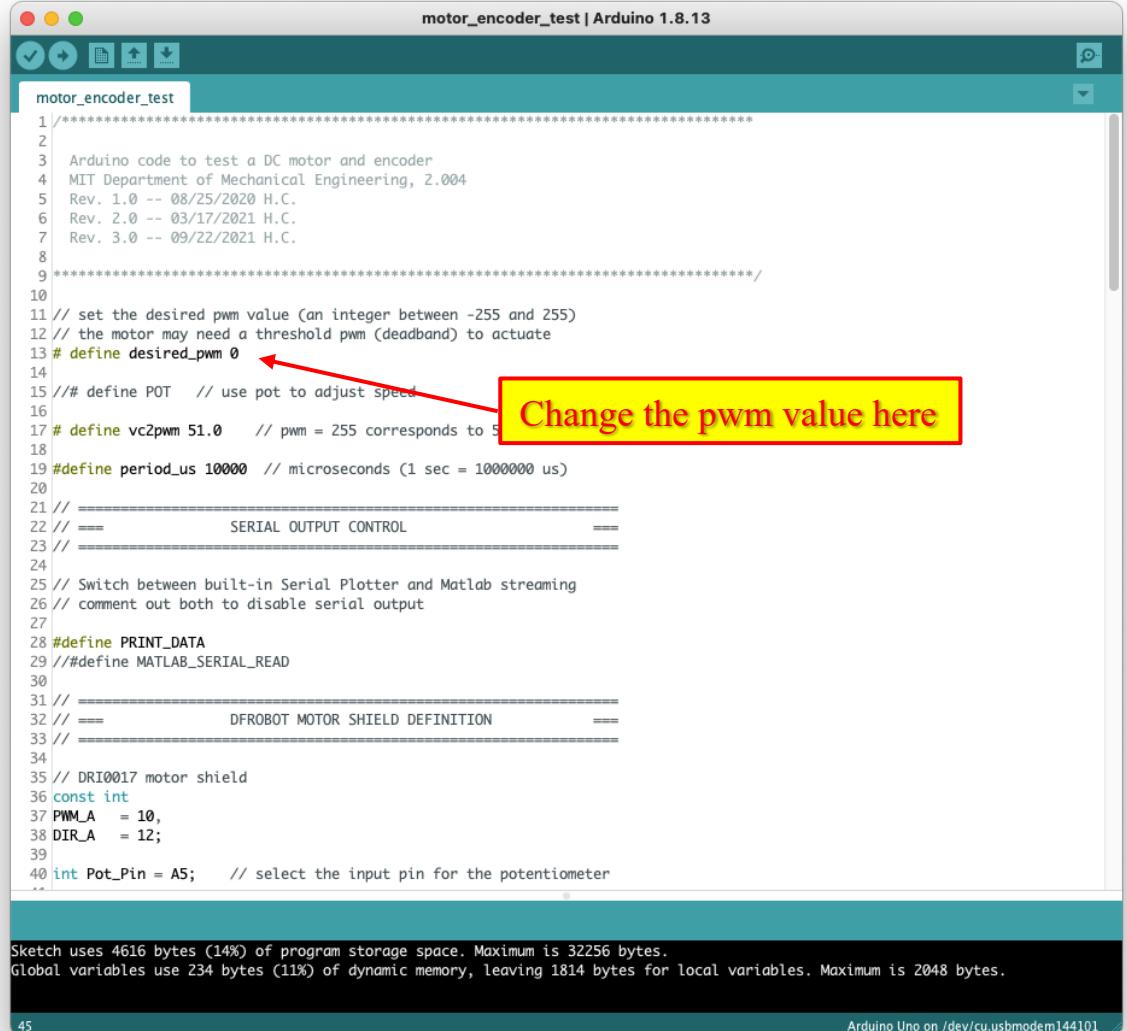
A red arrow points from the 'Serial monitor' icon in the Arduino IDE to a yellow box labeled 'Serial monitor'. Another red arrow points from the 'Upload button' in the IDE to a yellow box labeled 'Upload button'.

Once per revolution index



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.



The screenshot shows the Arduino IDE interface with the sketch 'motor_encoder_test' open. The code is written in C++ and defines a DC motor and encoder setup. It includes comments for serial output control and DFRobot Motor Shield definition. A red arrow points from the text 'Change the pwm value here' to the line '#define desired_pwm 0'. The status bar at the bottom indicates the sketch uses 4616 bytes of program storage space and 234 bytes of dynamic memory.

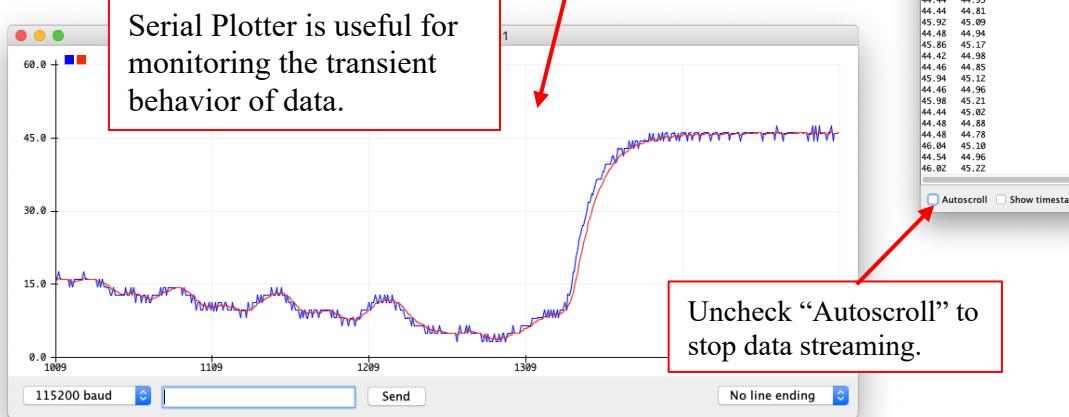
```
1 //*****  
2  
3 Arduino code to test a DC motor and encoder  
4 MIT Department of Mechanical Engineering, Z.004  
5 Rev. 1.0 -- 08/25/2020 H.C.  
6 Rev. 2.0 -- 03/17/2021 H.C.  
7 Rev. 3.0 -- 09/22/2021 H.C.  
8  
9 *****/  
10  
11 // set the desired pwm value (an integer between -255 and 255)  
12 // the motor may need a threshold pwm (deadband) to actuate  
13 #define desired_pwm 0  
14  
15 // define POT // use pot to adjust speed  
16 #define vc2pwm 51.0 // pwm = 255 corresponds to 5V  
17  
18 #define period_us 10000 // microseconds (1 sec = 1000000 us)  
19  
20  
21 // =====  
22 // == SERIAL OUTPUT CONTROL ==  
23 // =====  
24  
25 // Switch between built-in Serial Plotter and Matlab streaming  
26 // comment out both to disable serial output  
27  
28 #define PRINT_DATA  
29 // #define MATLAB_SERIAL_READ  
30  
31 // =====  
32 // == DFRobot MOTOR SHIELD DEFINITION ==  
33 // =====  
34  
35 // DRI0017 motor shield  
36 const int  
37 PWM_A = 10,  
38 DIR_A = 12;  
39  
40 int Pot_Pin = A5; // select the input pin for the potentiometer  
41
```

Sketch uses 4616 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 234 bytes (11%) of dynamic memory, leaving 1814 bytes for local variables. Maximum is 2048 bytes.

Monitoring/Capturing Serial Data



```
Arduino File Edit Sketch Tools Help
  Auto Format
  Archive Sketch
  Fix Encoding & Reload
  Manage Libraries...
  Serial Monitor
  Serial Plotter
  Serial Line Monitor
  Serial Line Listener
  WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "/dev/cu.usbmodem144101 (Arduino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
1
2
3 Arduino code to test a DC motor
4 MIT Department of Mechanical Engrg.
5 Rev. 1.0 -- 08/25/2020 H.C.
6
7 **** set the desired pwm value (0-255)
8 // the motor may need a threshold
9 // define desired_pwm 150
10
11 #define vc2pwm 51.0 // pwm = 255 corresponds to 5v from pwm pin (255/5)
12
13 // ===== SERIAL OUTPUT CONTROL =====
14
15 //=====
16 //=====
17 //=====
```

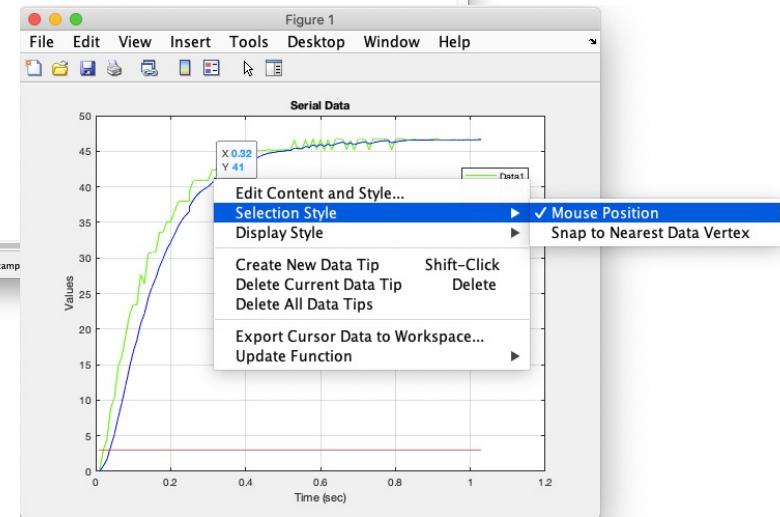


Neither “Serial Monitor” nor “Serial Plotter” is capable of saving data to a file.

```
44.48 44.96
45.92 45.20
44.56 45.02
44.54 44.90
44.48 44.89
46.01 45.10
44.48 44.94
44.46 44.82
44.38 45.11
44.46 44.95
45.92 45.10
44.38 44.99
44.46 44.86
46.00 45.14
44.50 44.98
44.50 44.86
44.48 44.76
44.46 44.68
45.82 44.96
44.46 44.84
44.46 44.74
45.98 45.05
44.52 44.92
44.46 44.89
44.46 44.72
45.92 45.02
44.48 44.88
46.00 45.10
44.52 45.01
44.44 44.87
44.46 44.77
44.92 45.06
44.46 44.91
44.46 44.89
45.99 45.10
44.44 44.93
44.44 44.81
44.52 44.99
44.48 44.94
45.86 45.17
44.42 44.98
44.46 44.85
45.94 45.12
44.46 44.96
45.98 45.21
44.44 45.02
44.46 44.88
44.46 44.78
46.04 45.10
44.54 44.96
46.02 45.22
```

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];



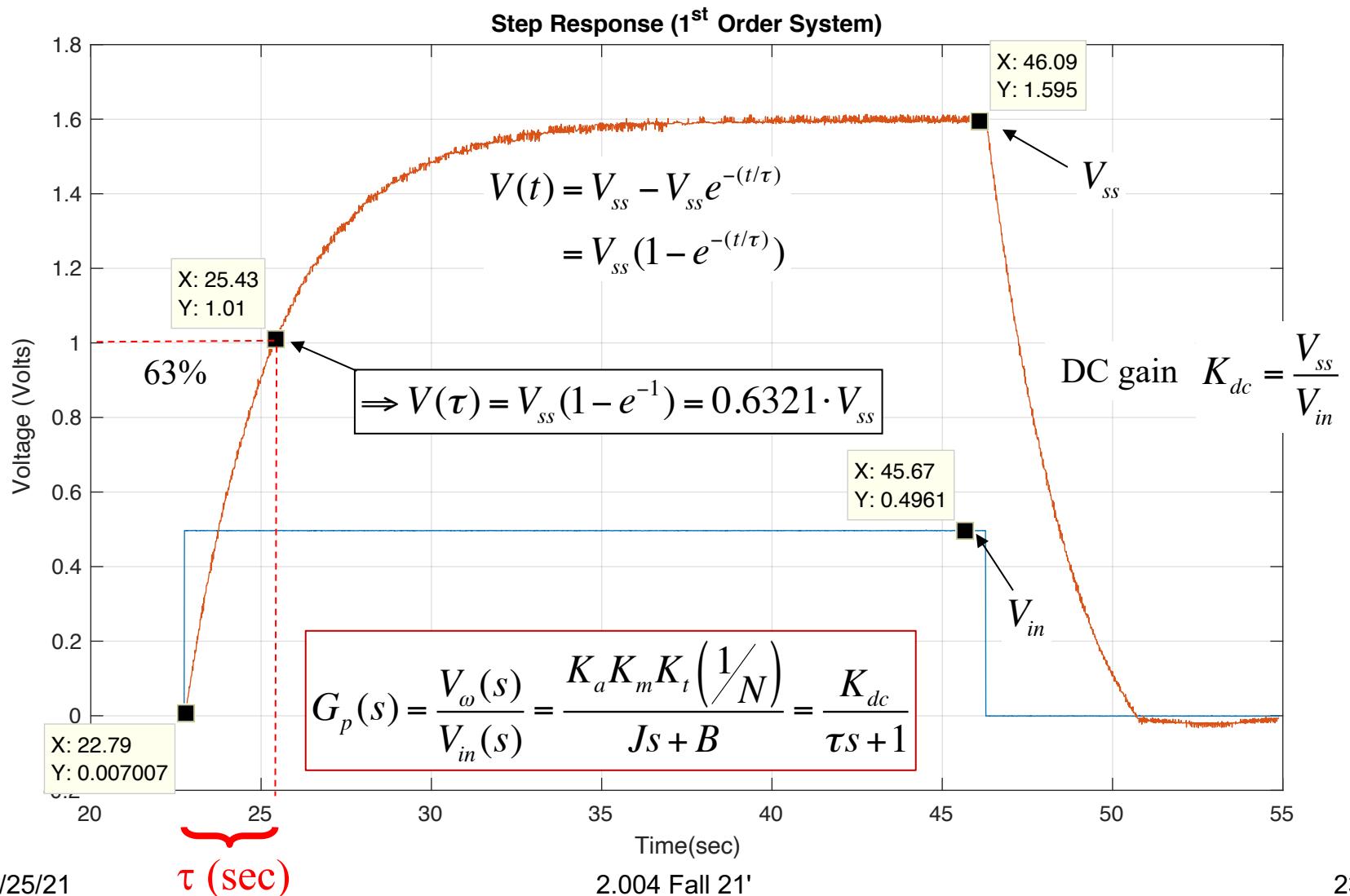
The Matlab script “SerialRead.m” can be used to stream serial data directly into Matlab workspace for plotting and for post processing.

Create Transfer Function

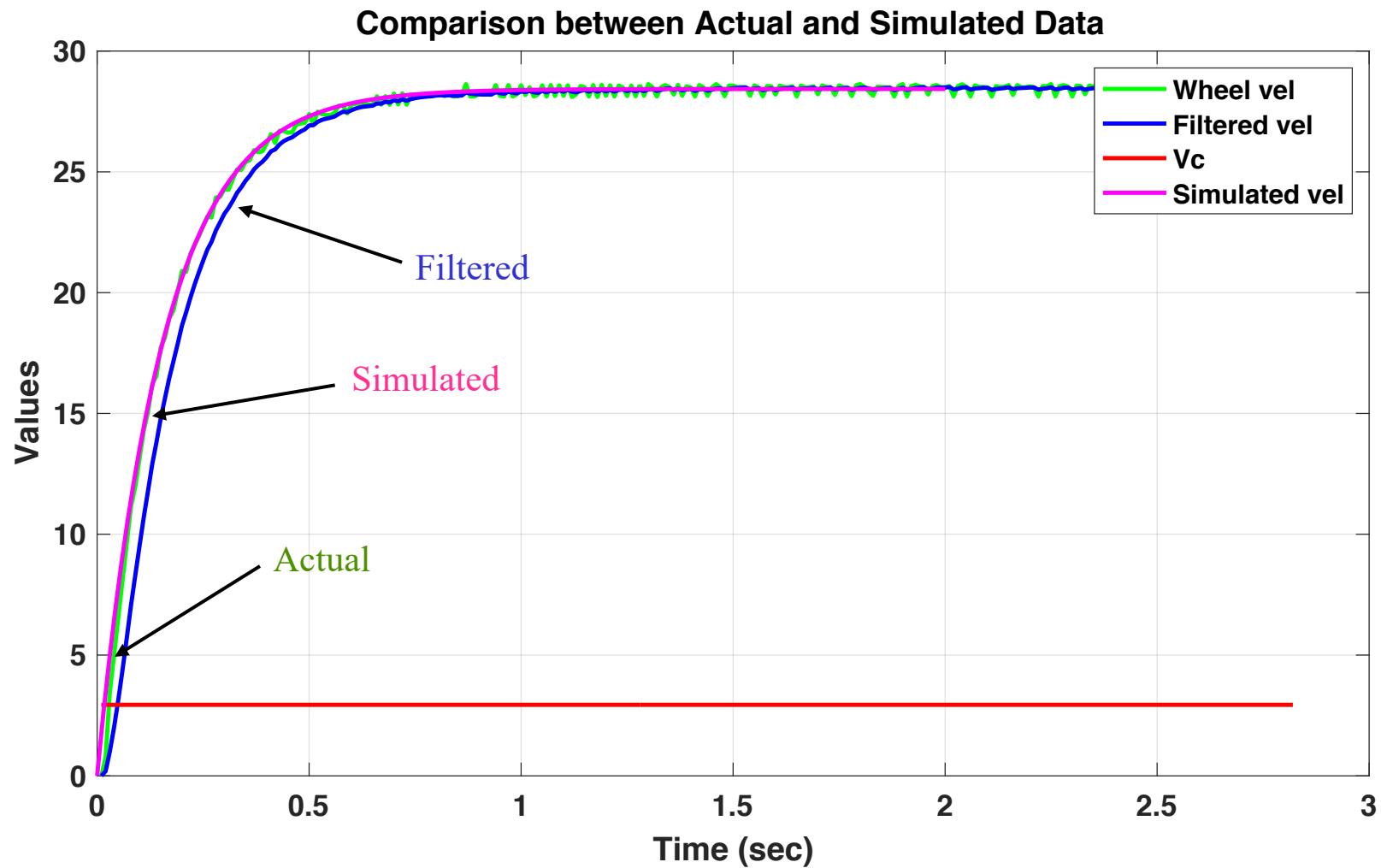


- Set the desired PWM to 150
- Comment out `#define POT`
- Comment out `#define PRINT_DATA` and uncomment `#define MATLAB_SERIAL_READ`
- Upload the sketch to Arduino
- Open the Matlab script “SerialRead.m” and change the serial port name to the Arduino port name.
- Run the Matlab script and capture a 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



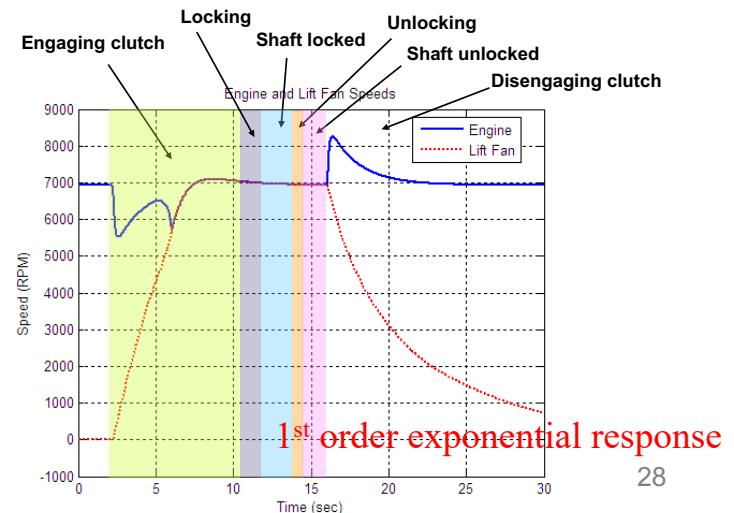
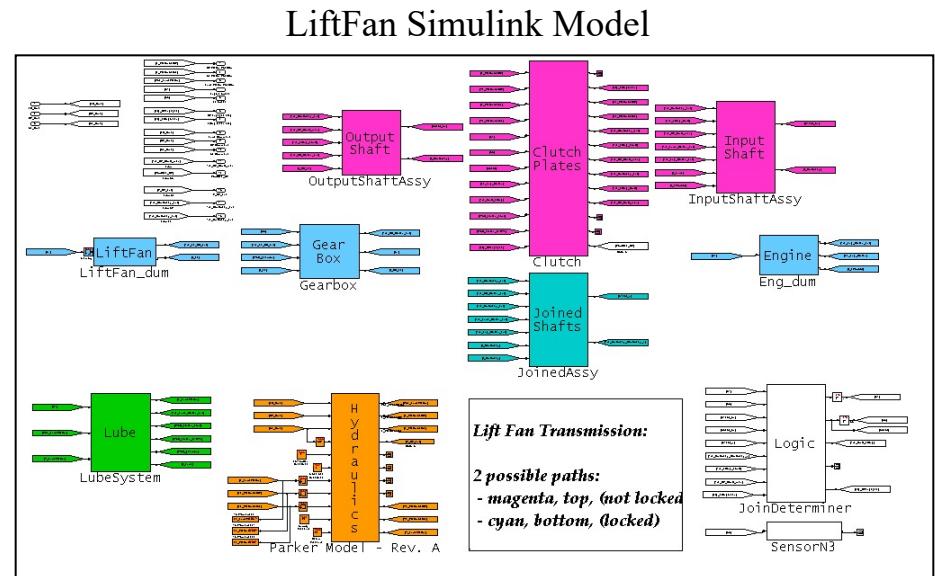
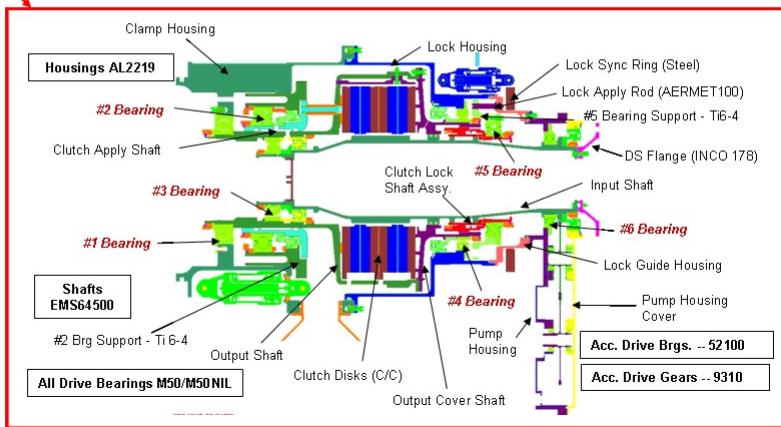
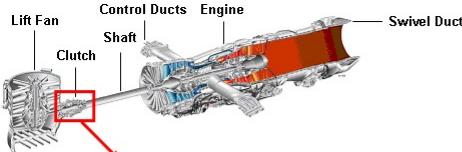
- Determine the appropriate physical parameter values for the given Simscape model.
- Understand each individual block in the model.
- Run simulation and generate a step response.
- Fine tune some parameter values to match the actual response.

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

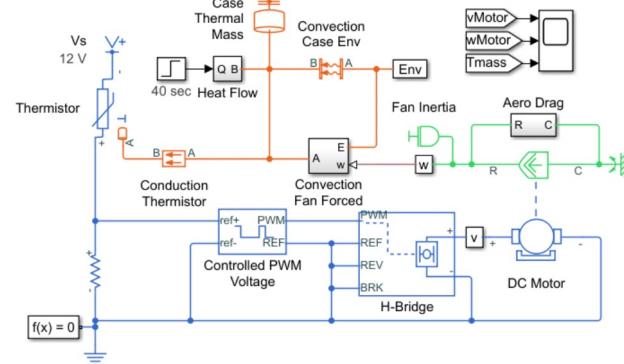
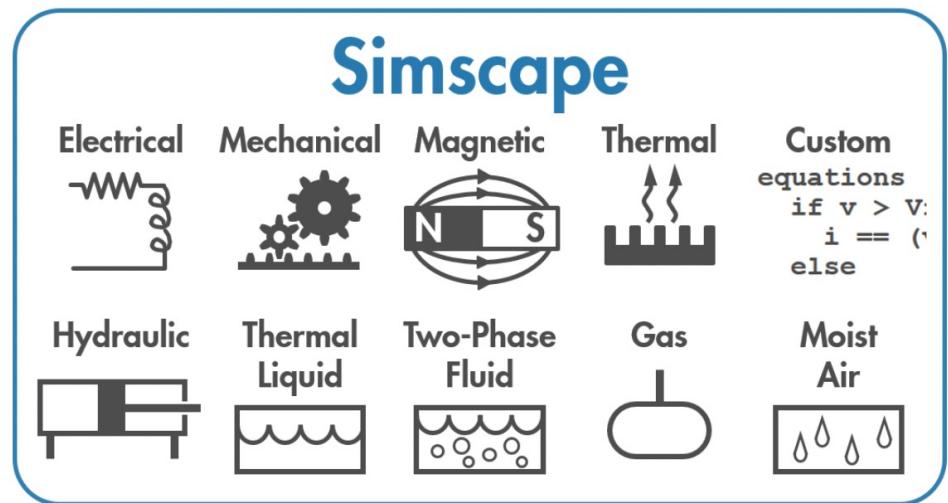
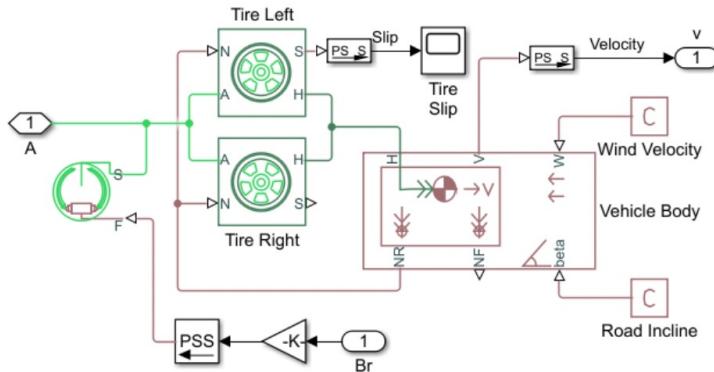


What is Simscape?

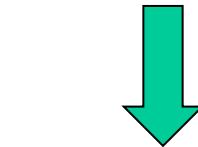
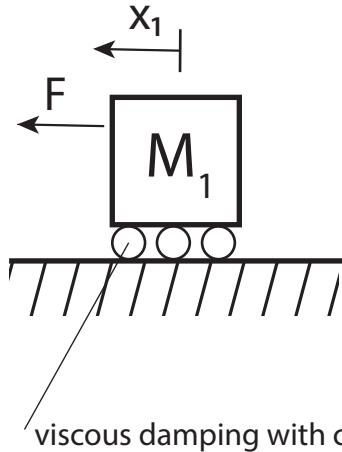
Simscape is a software package built on Simulink to directly model physical systems without the need for underlying equations.

Platform for physical Design

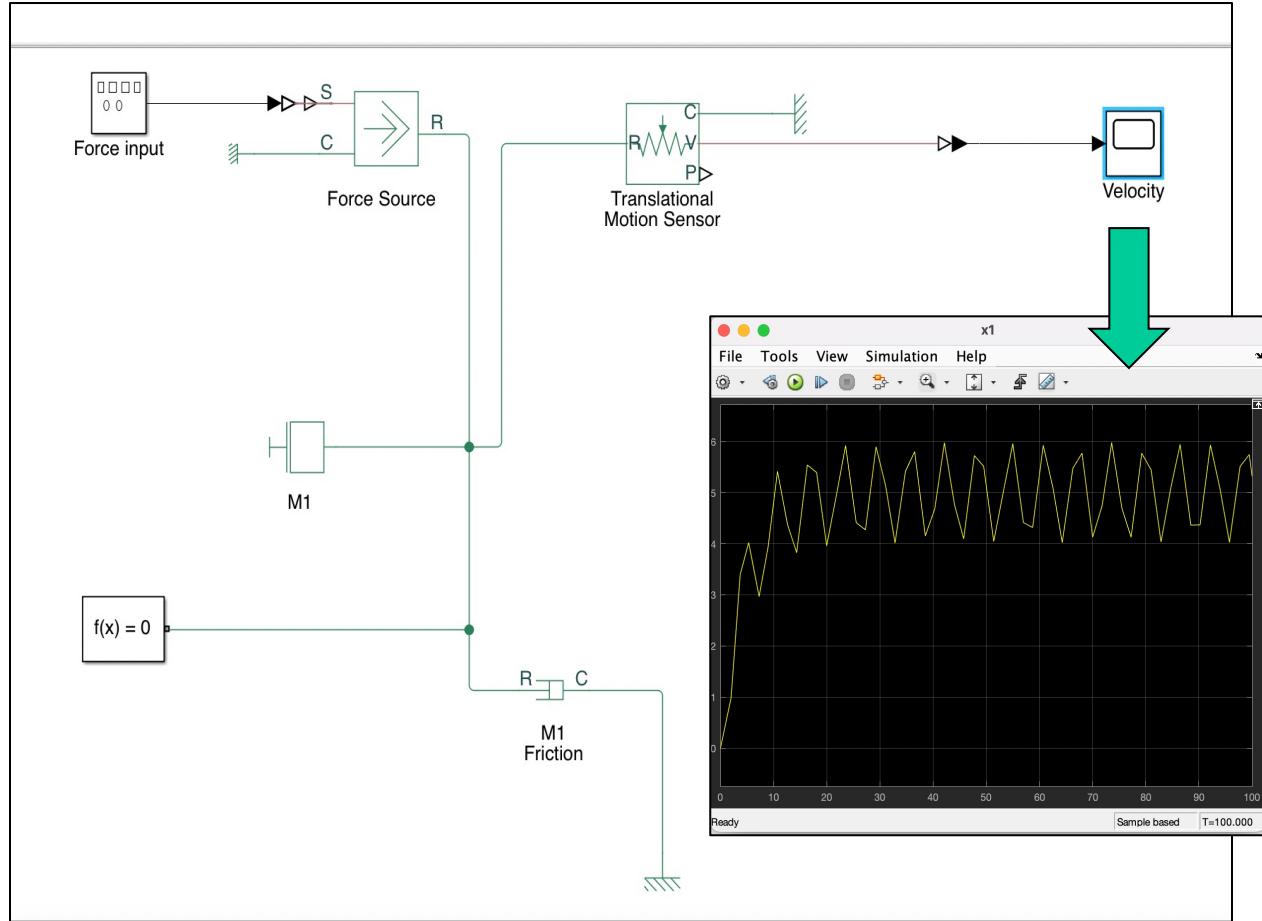
- Block diagram modeling
 - Simulation of physical systems like mechanical, electrical, fluidic, etc.
 - Automatically derives equations from physical block connections



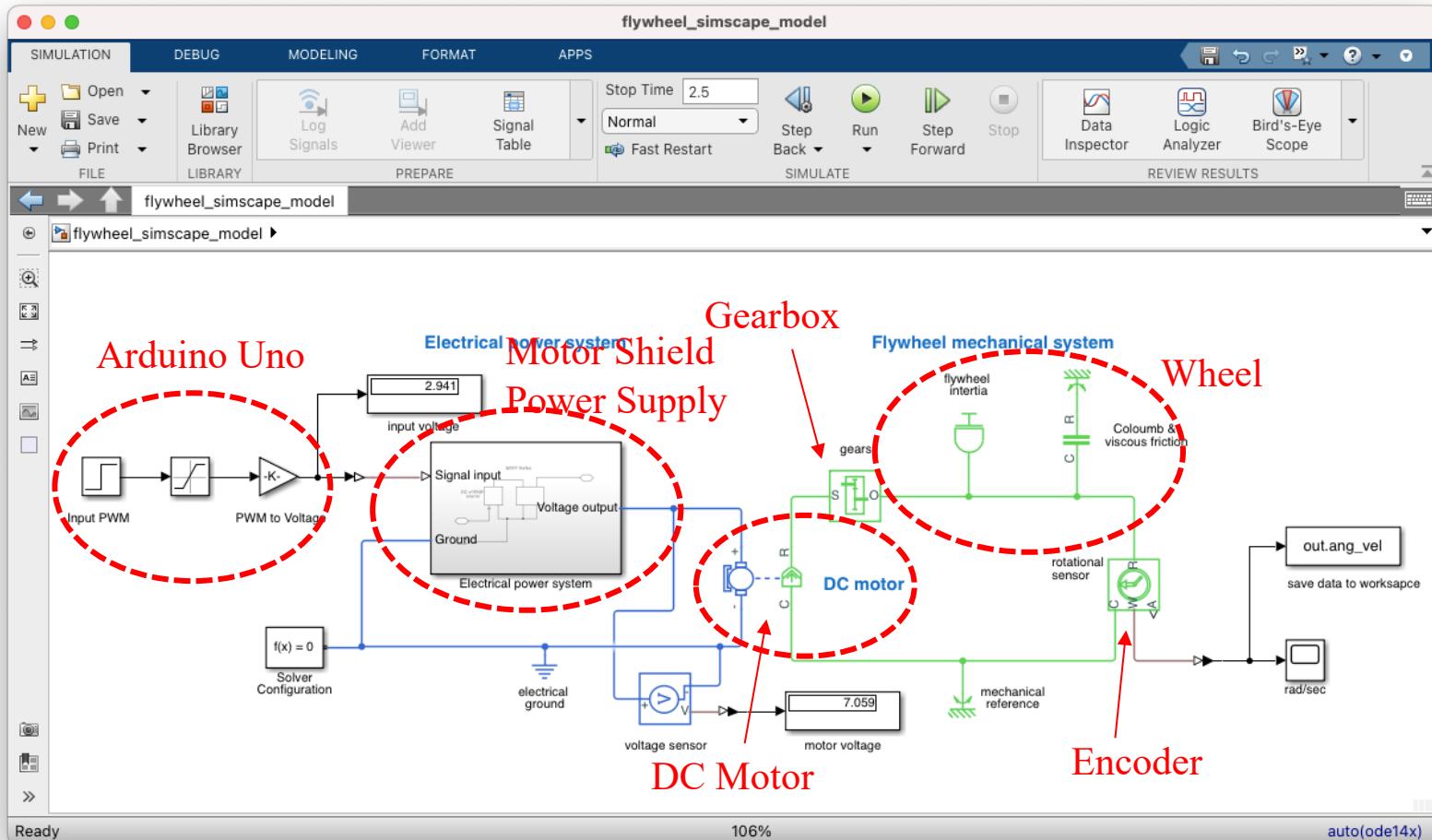
Example: a Single Mass on Wheels with Simscape



Components:
1 Mass
1 damper



Flywheel System Simscape Model



Simplified DC Motor Model



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

$K_m = K_v$ when both are in SI units

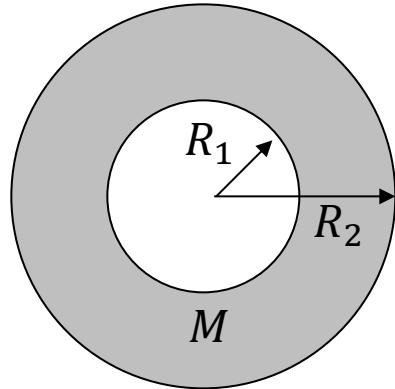
$$\frac{\Omega(s)}{V_s(s)} = \frac{\left(\frac{K_v}{R} \right)}{J_s + \left(b + \frac{K_v^2}{R} \right)} = \frac{K}{\tau s + 1}$$

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

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

Model Parameters

Moment of inertia:



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

$$J = \frac{M}{2} (R_1^2 + R_2^2) \approx 1.47 \times 10^{-4} \text{ [kg.m}^2\text{]}$$

DC motor armature resistance (R) measurement:

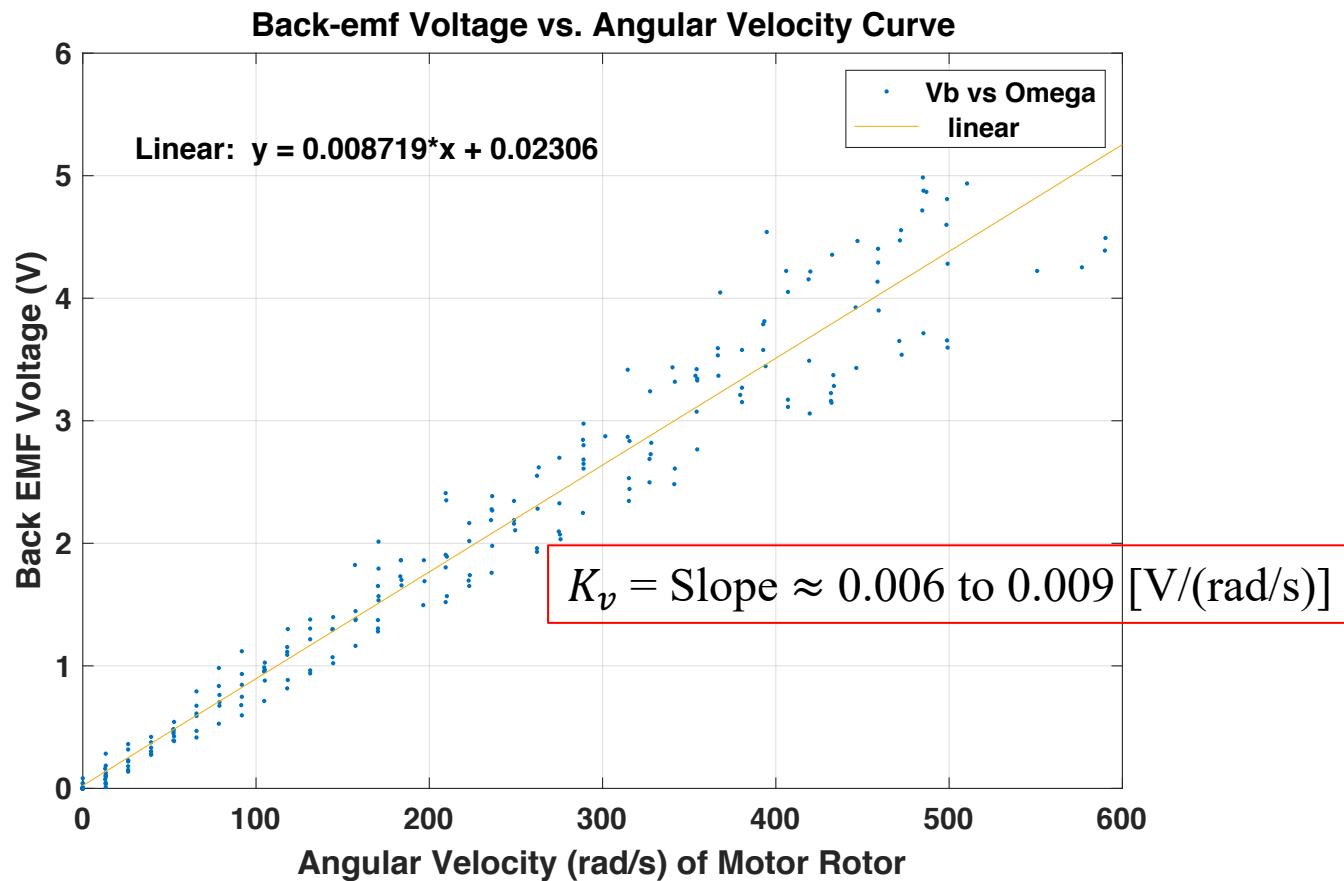
- Method 1: apply a small DC voltage (0.32 V) to the motor without actuating the motor shaft (to prevent back emf voltage generation) and measure the current (0.09 A). The resistance R can then be calculated as:

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

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

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.

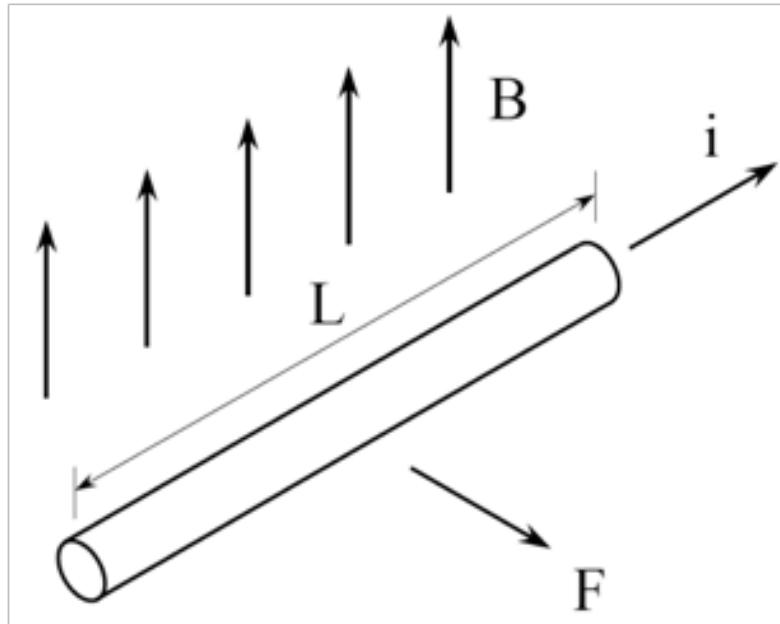


DC Motor

DC Motor Principle

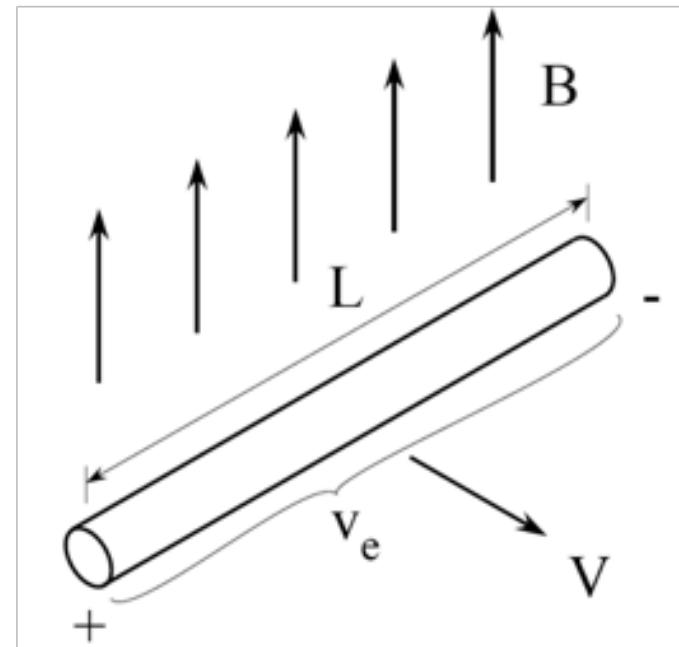
Lorentz law:

magnetic field applies force to a current
(Lorentz force)



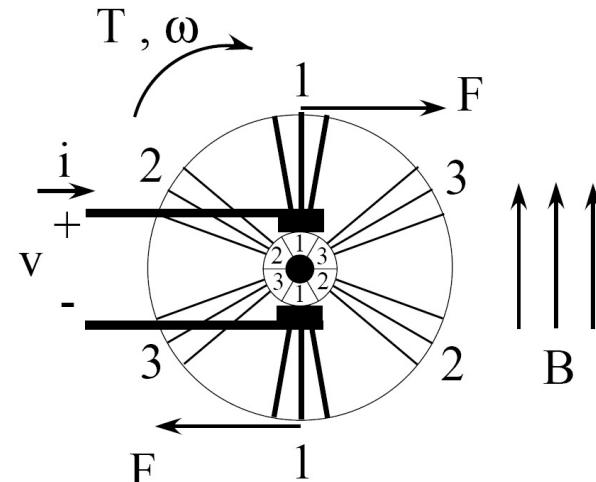
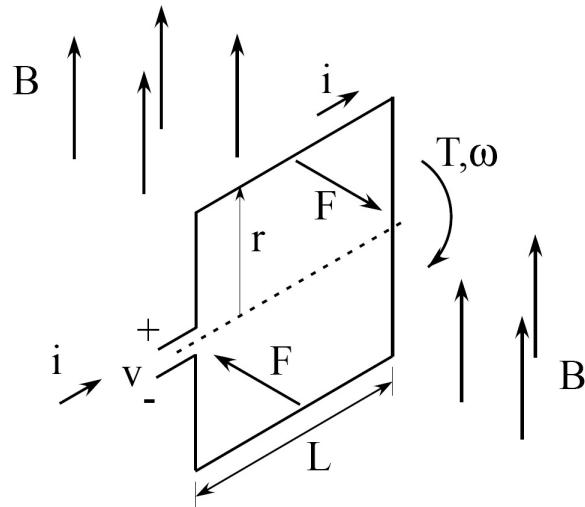
Faraday law:

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

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

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

or

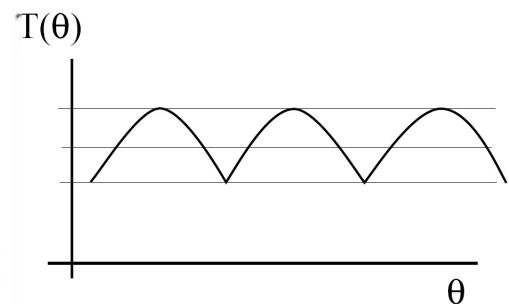
where

$$T = K_m i$$

$$v_e = K_v \omega$$

- $K_m \equiv 2BNlr$ torque constant

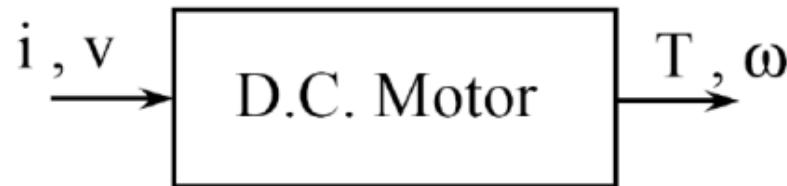
- $K_v \equiv 2BNlr$ back-emf constant



DC Motor as A Transducer

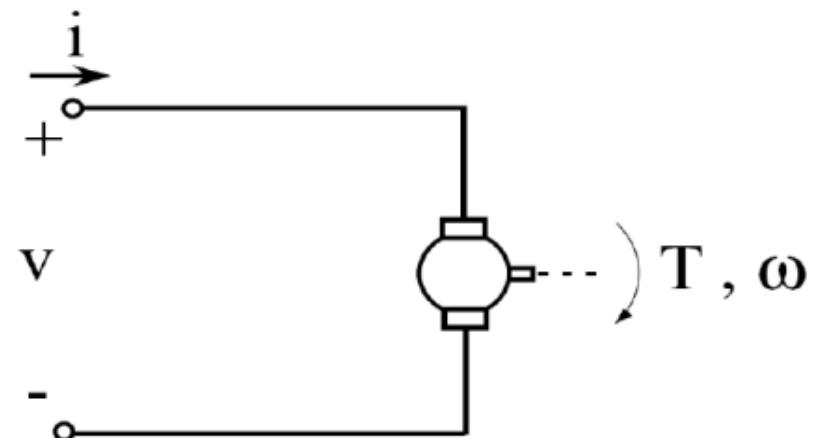
DC motor is an electromechanical system that may have a time constant close to the plant.

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

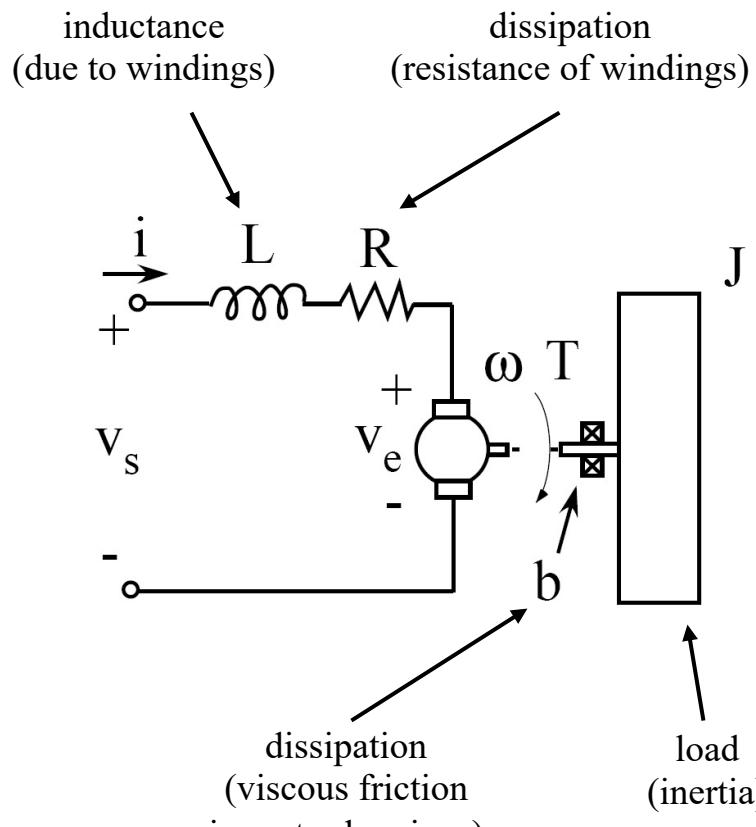


$$P_{in} = P_{out}$$

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



DC Motor with Mechanical Load



K_m : motor torque constant

K_v : motor speed (back EMF) constant

Equation of motion – Electrical

$$\text{KVL: } v_s - v_L - v_R - v_e = 0$$

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

Equation of motion – Mechanical

$$\text{Torque Balance: } T = T_b + T_J$$

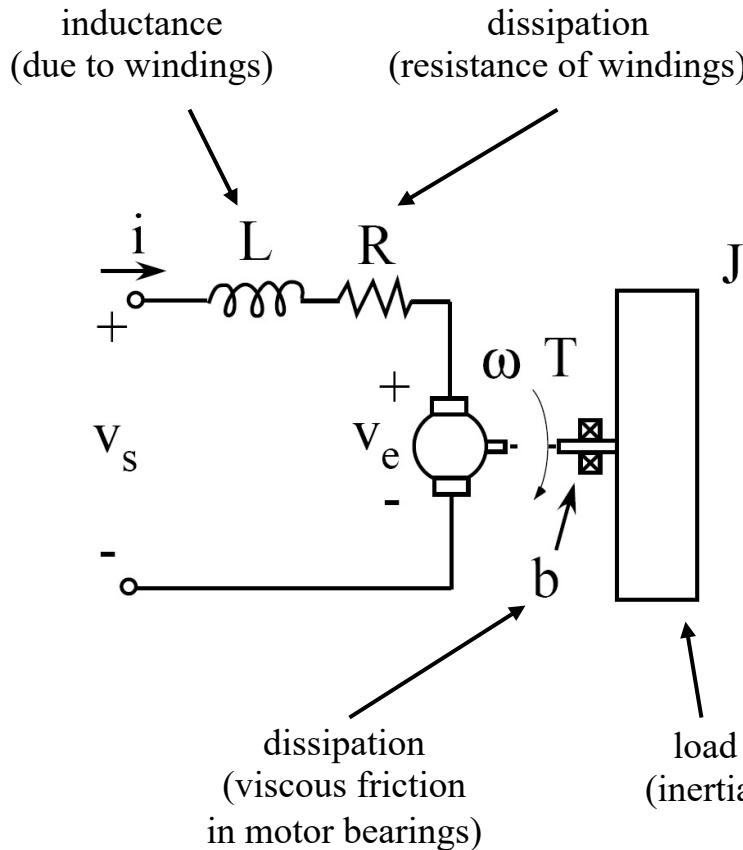
$$\Rightarrow K_m i - b\omega = J \frac{d\omega}{dt}$$

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

$$\text{KVL: } V_s(s) - V_L(s) - V_R(s) - V_e(s) = 0$$

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

Equation of motion – Mechanical

$$\text{Torque Balance: } T(s) = T_b(s) + T_J(s)$$

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

Combined equations of motion

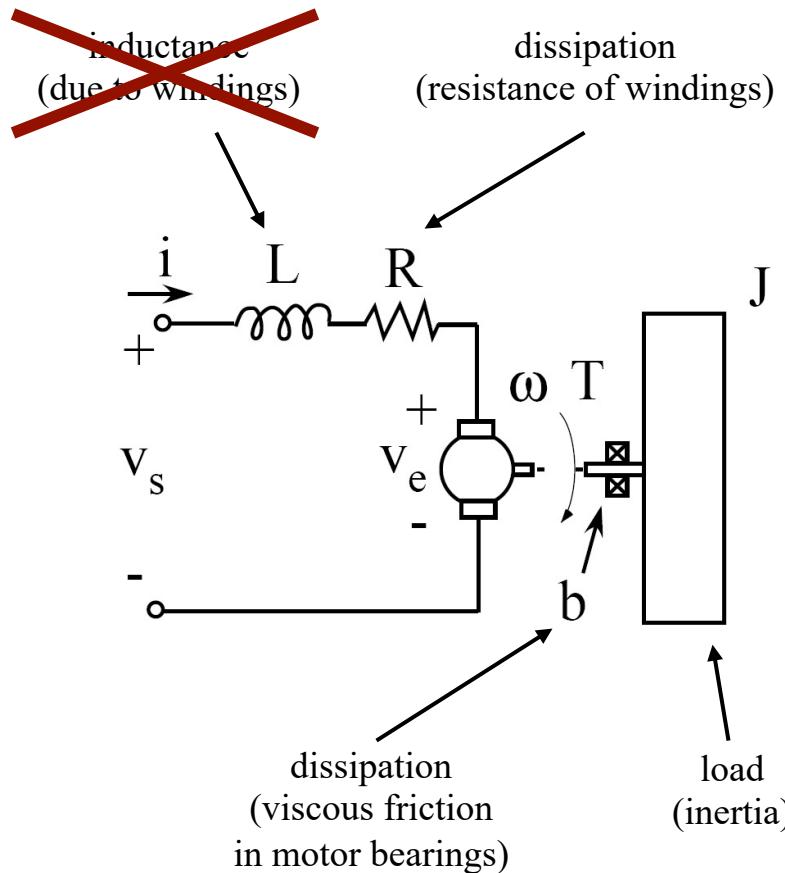
$$LsI(s) + RI(s) + K_v\Omega(s) = V_s(s)$$

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

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

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

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!

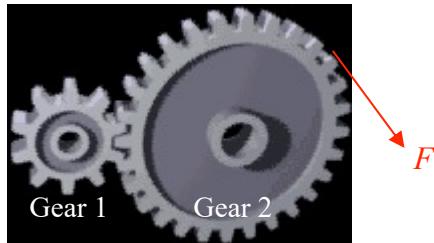
If we are given step input $v_s(t) = V_0 u(t)$
 \Rightarrow we already know the step response

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

where now the time constant is

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

Gear Ratio and Power Conservation



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

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

Unit Conversion:

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

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

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

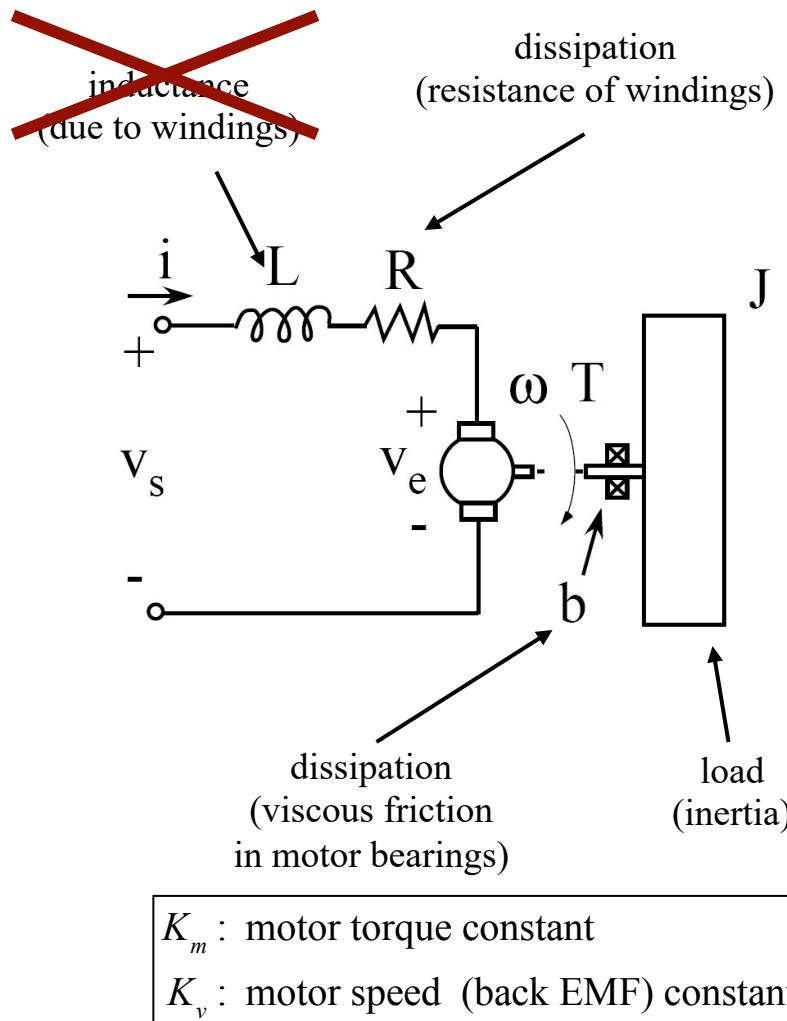
Power Conservation:

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

$$\begin{aligned} P_{mechanical}(t) &= T(t) \times \Omega(t) \\ &= K_m \times i(t) \times \Omega(t) \end{aligned}$$

$$\begin{aligned} P_{electrical}(t) &= v_b(t) \cdot i(t) \\ &= K_v \cdot \Omega(t) \cdot i(t) \end{aligned}$$

DC Motor with Mechanical Load



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

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

where now the time constant is

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