MBot
Motors & Encoders

Lecture 13 – ROB 550 Winter ‘23

Thursday February 23, 2023

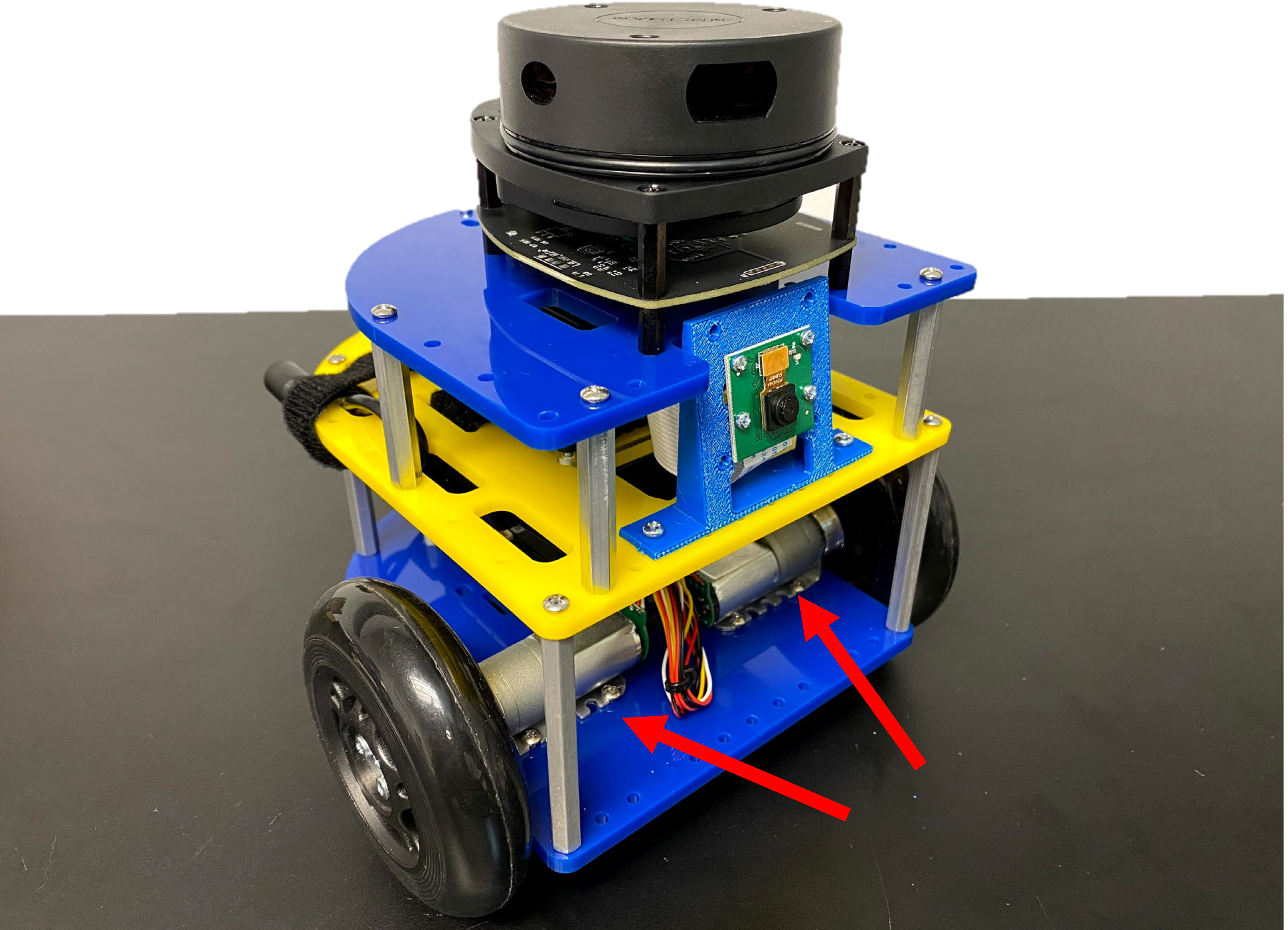
Logistics

- Armlab Reports – due last night
- Armlab Quiz – due tonight!

- Botlab - new teams!

- Questions?

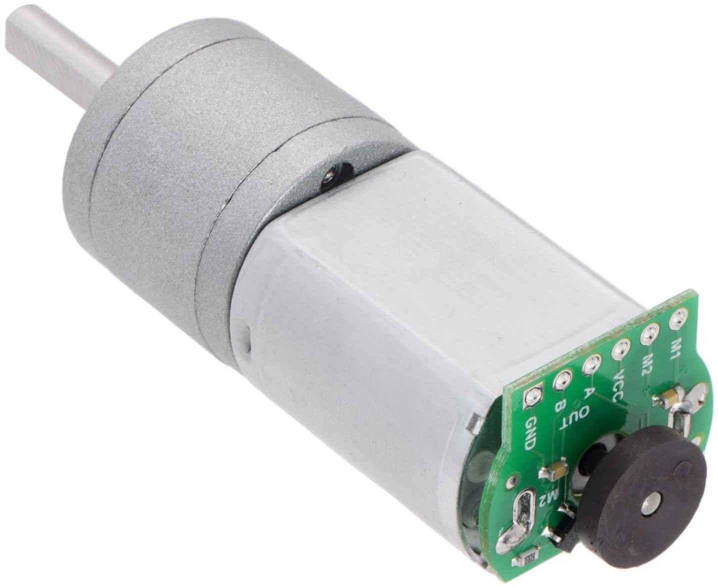
Mbot Mobile Robot Kit

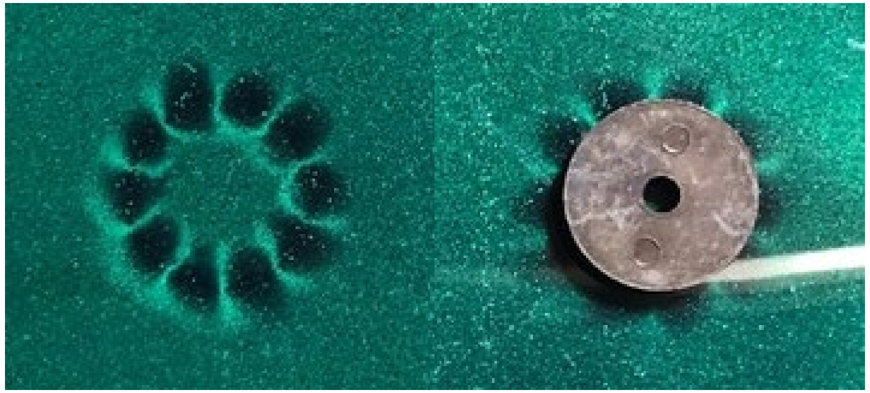


- Bottom
  - Robotics Control Board
  - Brushed DC Motors
  - Magnetic Encoders
  - IMU (Accel, Rate Gyro, Magnetometer)
- Top
  - Raspberry Pi 4B
  - 5MP Camera
  - 2D Scanning LIDAR

Motors + Encoders

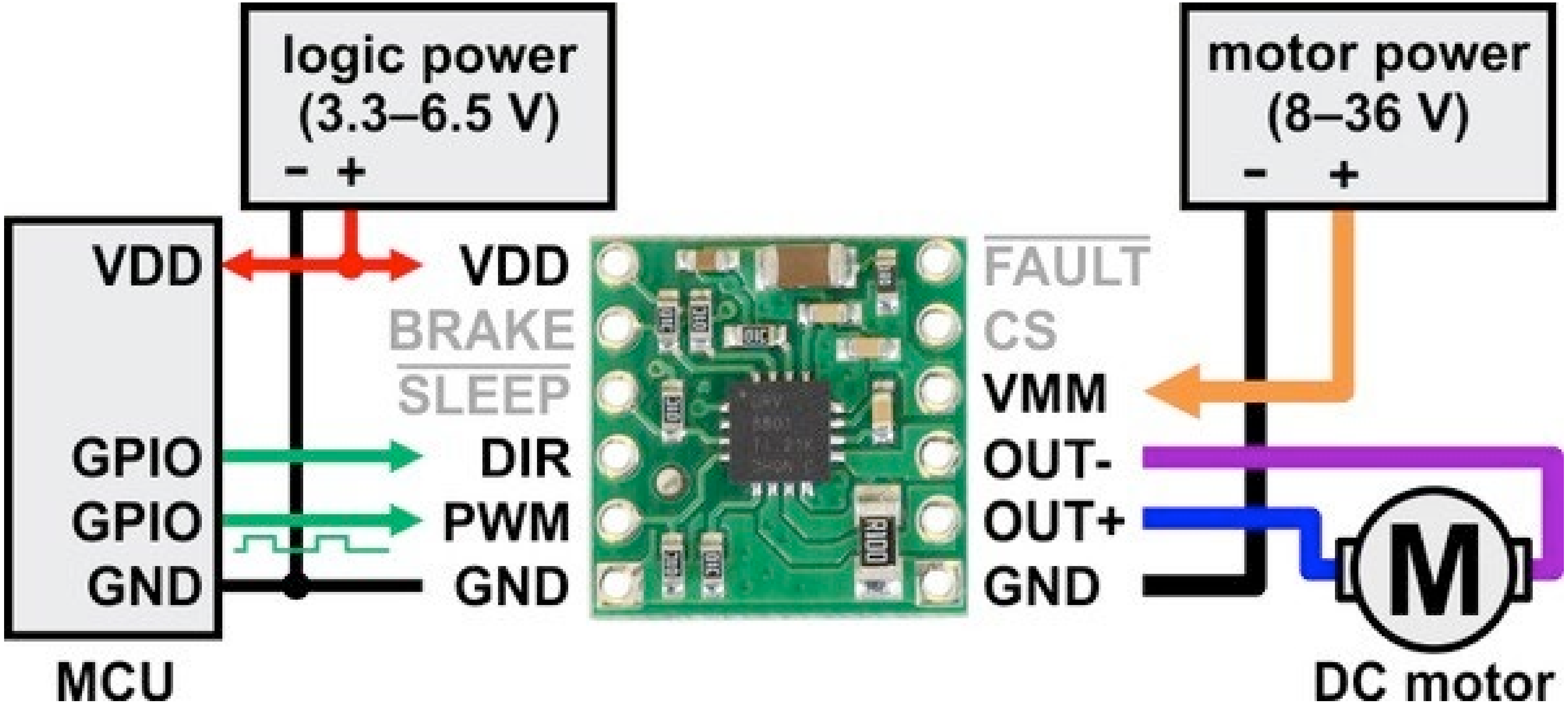
- Pololu 6V Brushed DC gear motors
- 20 count per revolution magnetic encoder
- Almost all are 78:1 Gear Ratio
- You may have 63:1. how to tell?
- Polarity of motors not guaranteed





Jiawei Chen

MBot Motor Driver



Pico-RCLib Motor Functions

- Test motors with program `pico_motor_test`
- Available functions:

```
int rc_motor_init();
int rc_motor_init_freq(uint f);
int rc_motor_cleanup();
int rc_motor_set(uint ch, int32_t duty);
int rc_motor_free_spin(uint ch);
int rc_motor_brake(uint ch);
```

Pico-RCLib Encoder Functions

- Pico-RCLib has 4 encoders, 3 broken out to MRCB
- Test program pico_encoder_test
- Available functions:

```
int rc_encoder_init();
int rc_encoder_cleanup();
int rc_encoder_read_delta(uint ch);
int rc_encoder_read_count(uint ch);
int rc_encoder_write(uint ch, int pos);
```

Pico Time Functions

- RPi Pico-SDK has many functions for timing and sleeping
- Some useful ones:

```
uint64_t to_us_since_boot (absolute_time_t t);
absolute_time_t get_absolute_time (void);
void sleep_us (uint64_t us);
```

Objectives

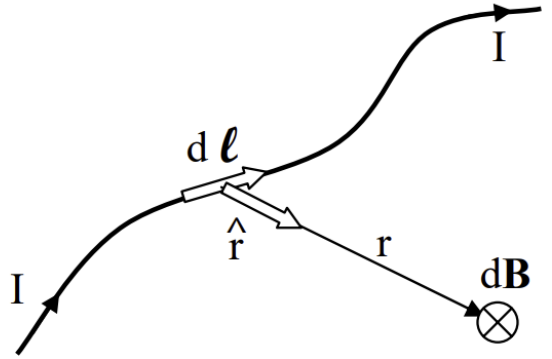
- Understand how DC motors work
- Understand how various encoder technologies work
- Learn how to calculate and calibrate wheel speeds

- Utilize all of this for MBot
  - Understand how our hardware functions
  - Calibrate PWM signal vs. wheel speed
  - Give speed commands, let software control motors

E & M Review

1. Ampere's Law: Moving charges (currents, I) create magnetic fields (**B**)

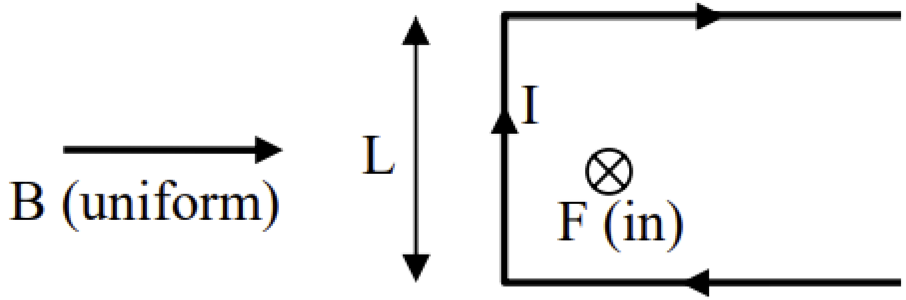
Biot-Savart Law: $\boxed{d\vec{B} = \frac{\mu_0 I}{4\pi} \frac{d\vec{\ell} \times \hat{r}}{r^2}}$



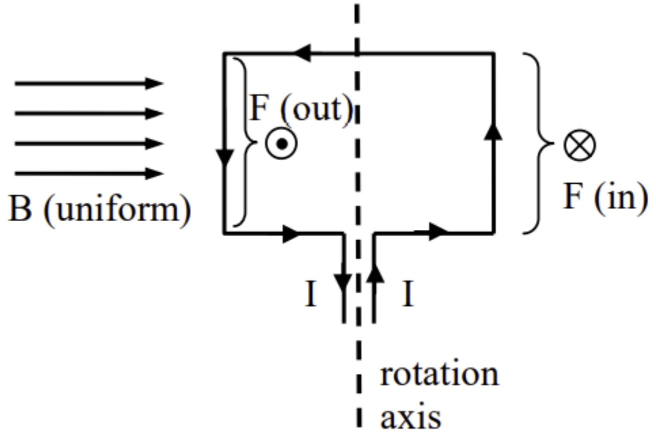
2. Magnetic fields exert forces on moving charges (currents)

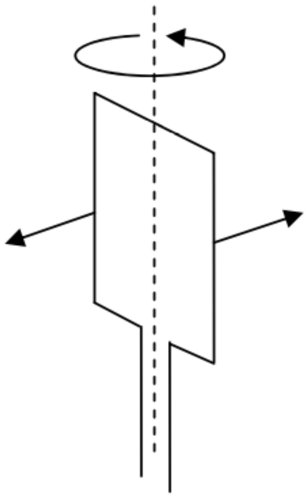
$\boxed{\vec{F}_B = q\vec{v}\times\vec{B}}$

$\boxed{\vec{F} = I\,\vec{L}\times\vec{B}}$



E & M Review

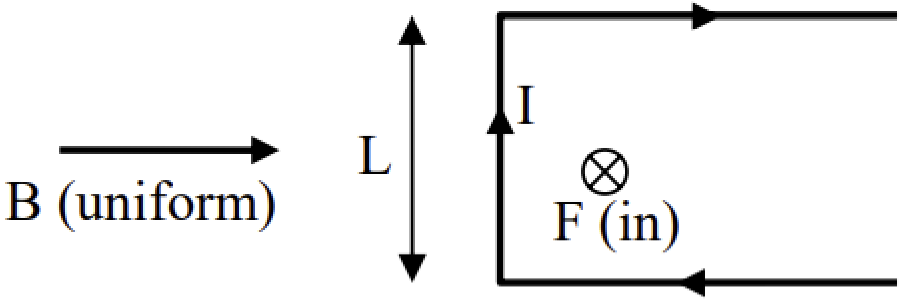




2. Magnetic fields exert forces on moving charges (currents)

$$\boxed{\vec{F}_B = q\,\vec{v}\times\vec{B}}$$

$$\boxed{\vec{F} = I\,\vec{L}\times\vec{B}}$$



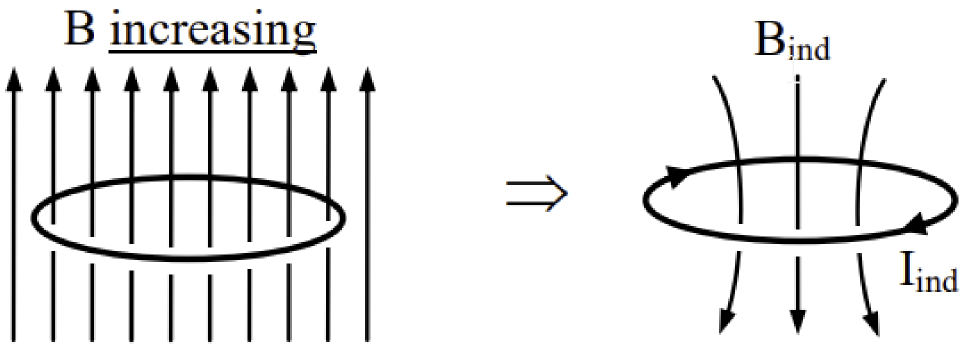
E & M Review

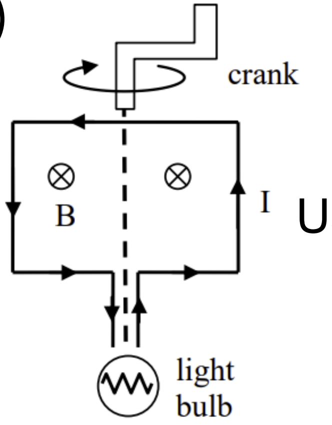
3. Faraday's Law: A changing magnetic field/flux creates an electric field (i.e., induces an electromotive force [e.m.f.])

$$\Phi_B = \int_S \vec{B} \cdot d\vec{A}$$

$$\boxed{\mathcal{E}_{(1\ \text{loop})} = -\frac{d\Phi_B}{dt}}$$

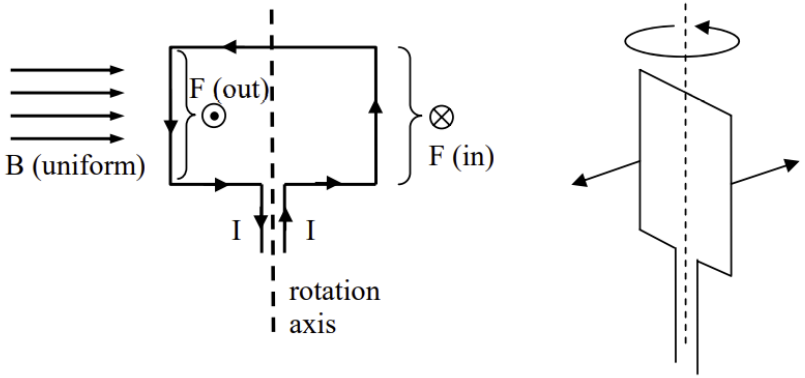
4. Lenz's Law: The induced e.m.f. induces a current in the direction which creates an induced B-field that **opposes** the change in flux (i.e., the induced current creates its own B-field that fights the original)





Useful for generators!

E & M Review

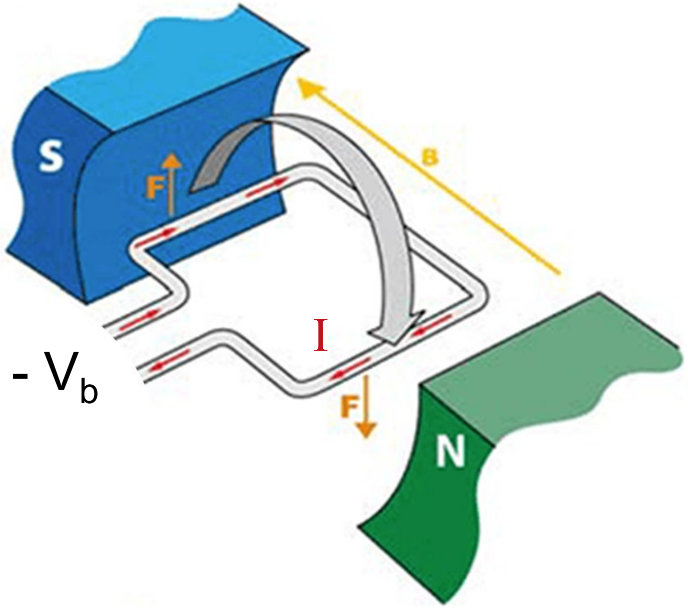


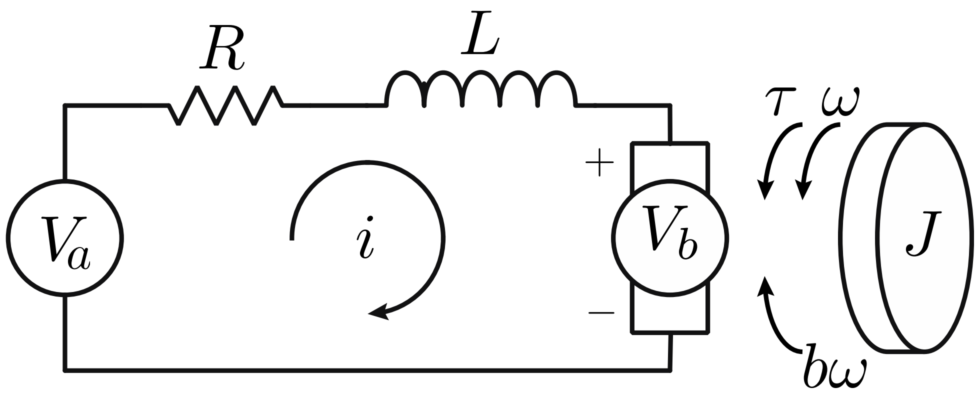
In summary:

- A current-carrying wire loop will have forces induced on it while in a fixed magnetic field. If rotated on an axis, the opposing current directions will create opposing forces, i.e., a **torque** (τ) about that axis.
- Once spinning due to the torque, the magnetic field (relative to the loop) is now changing, and thus will induce an opposing **“back” e.m.f.** ~ voltage (V_b), proportional to the speed

See any issues?

V_a - V_b



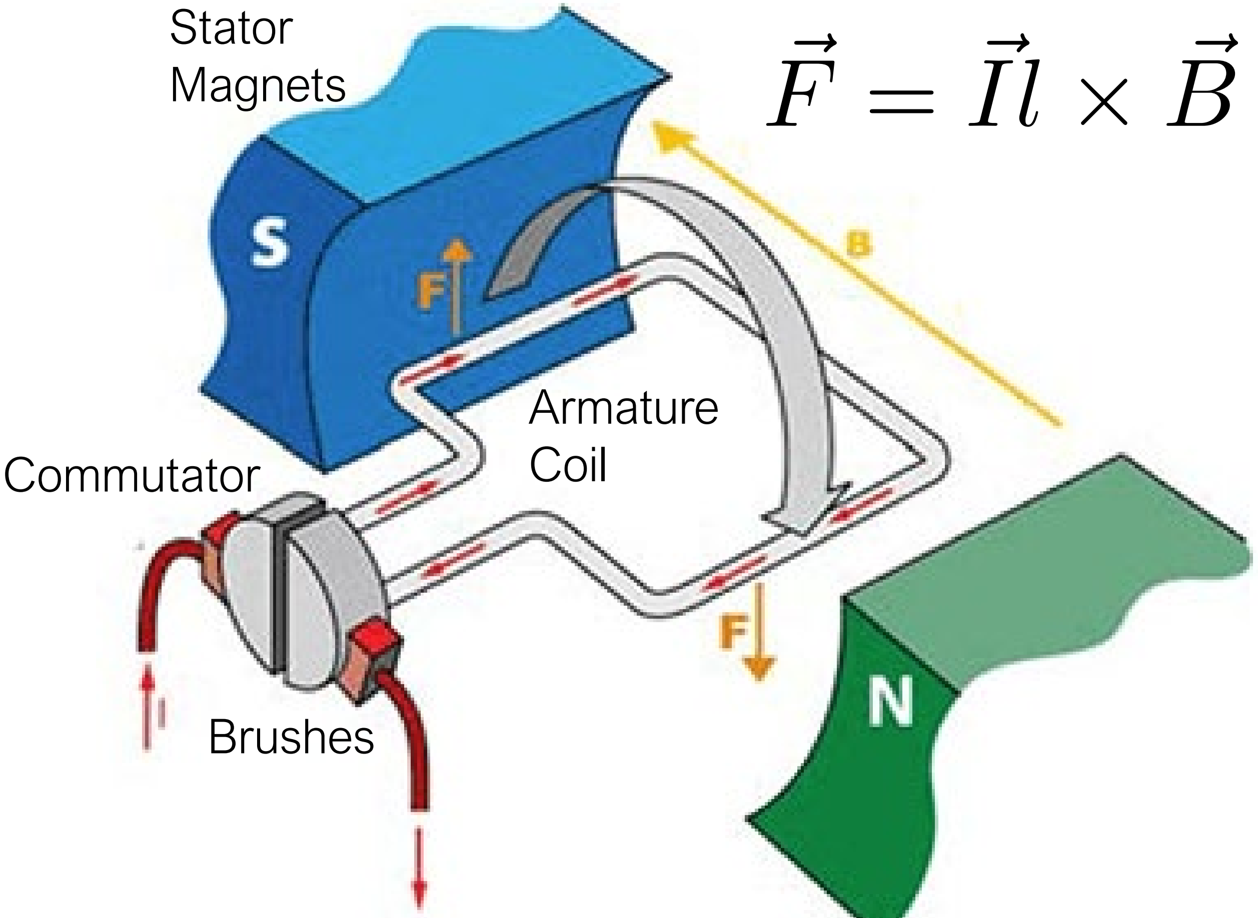


Brushed DC Motor Basics

- A Force is exerted on a current carrying wire in a fixed magnetic field
- Commutator composed of metal or carbon brushes switches the direction of current to continue rotation
- Force proportional to field strength (fixed) and current (input)
- Motion of current in the field produces an opposing magnetic field (back e.m.f., generator), reducing max current/torque possible

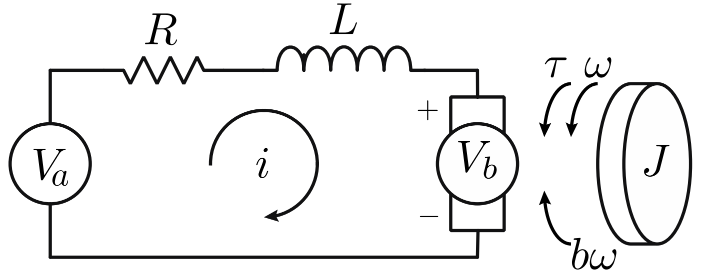
- Q: How does this differ from brushless motors?

$$\vec{F} = \vec{I}l \times \vec{B}$$



Motor Model

- R – armature resistance (Ω)
- L – armature inductance (H)
- V_a – applied voltage (V)
- V_b – back EMF (V)
- ω – motor speed (rad/s)
- τ – motor torque (N.m)
- θ – position (rad)
- b – friction coefficient (N.m)



$$V_a = Ri + L\frac{di}{dt} - V_b$$

$$\vec{F} = \vec{\imath}l \times \vec{B}$$

$$\tau = 2F \cdot \frac{d_{arm}}{2}$$

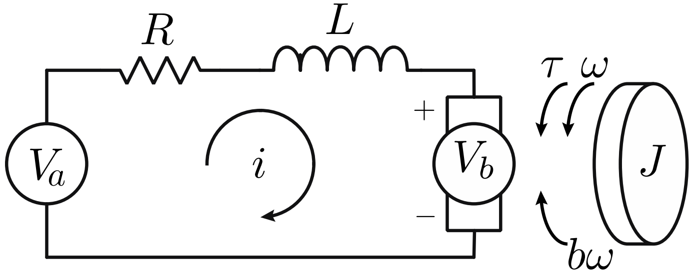
simplified at steady-state conditions:

$$\tau = K_t i$$

$$V_b = K_e \omega$$

$$i = (V_a - V_b)/R \text{ (in steady state)}$$

Motor Constants



- Power in: $P_{in} = IV_a$

$$V_a = IR + V_b$$

$$V_a = IR + K_e\omega$$

$$P_{in} = I^2R + K_eI\omega$$

- Power out: $P_{out} = \text{mechanical power} + \text{electrical losses}$

$$P_{out} = \tau\omega + I^2R$$

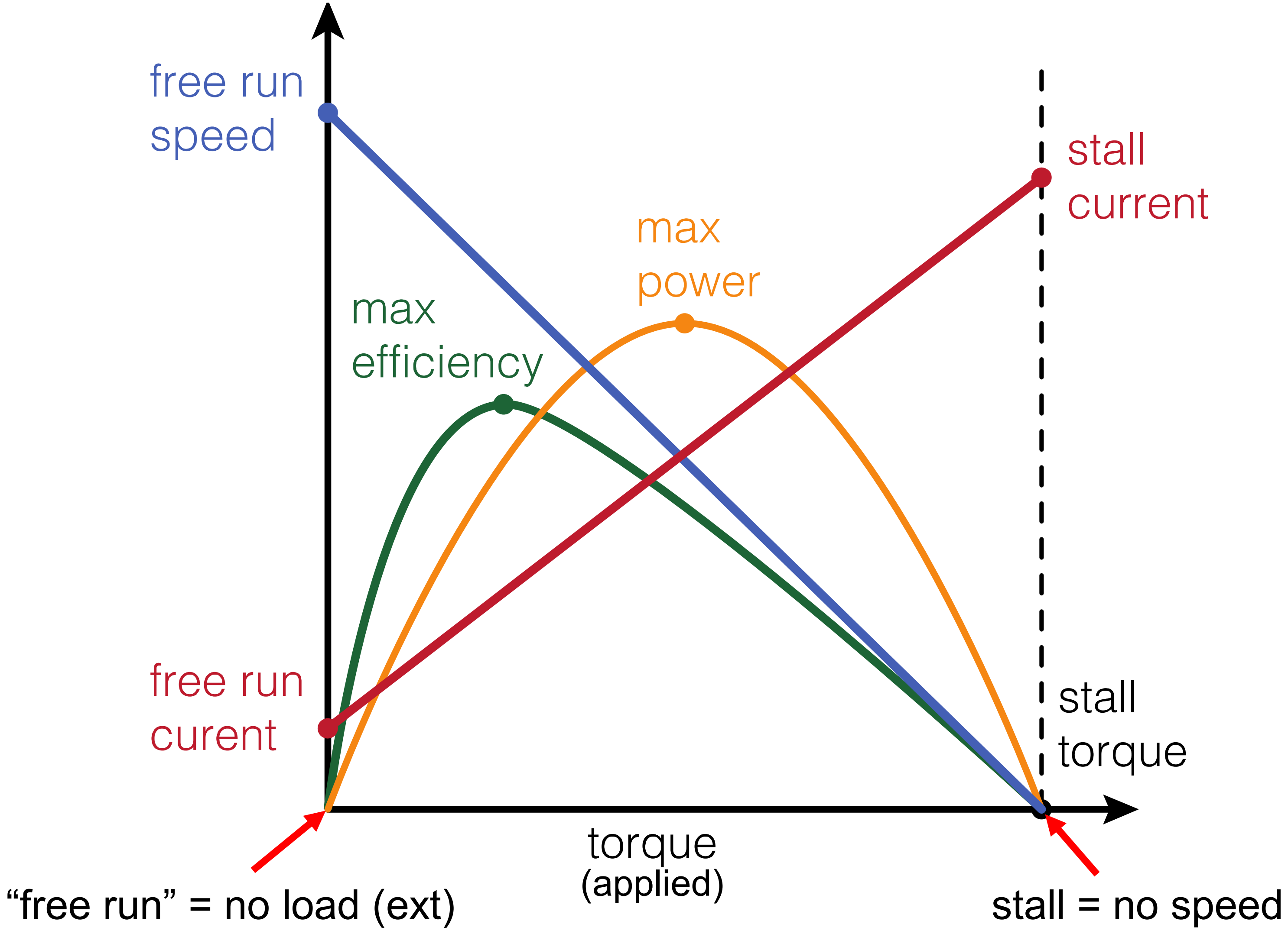
$$P_{out} = K_tI\omega + I^2R$$

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

$$I^2R + K_eI\omega = I^2R + K_tI\omega$$

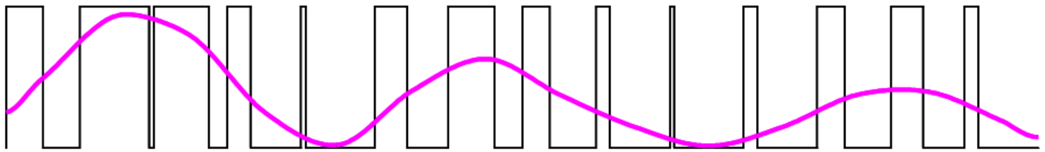
$$K_e = K_t$$

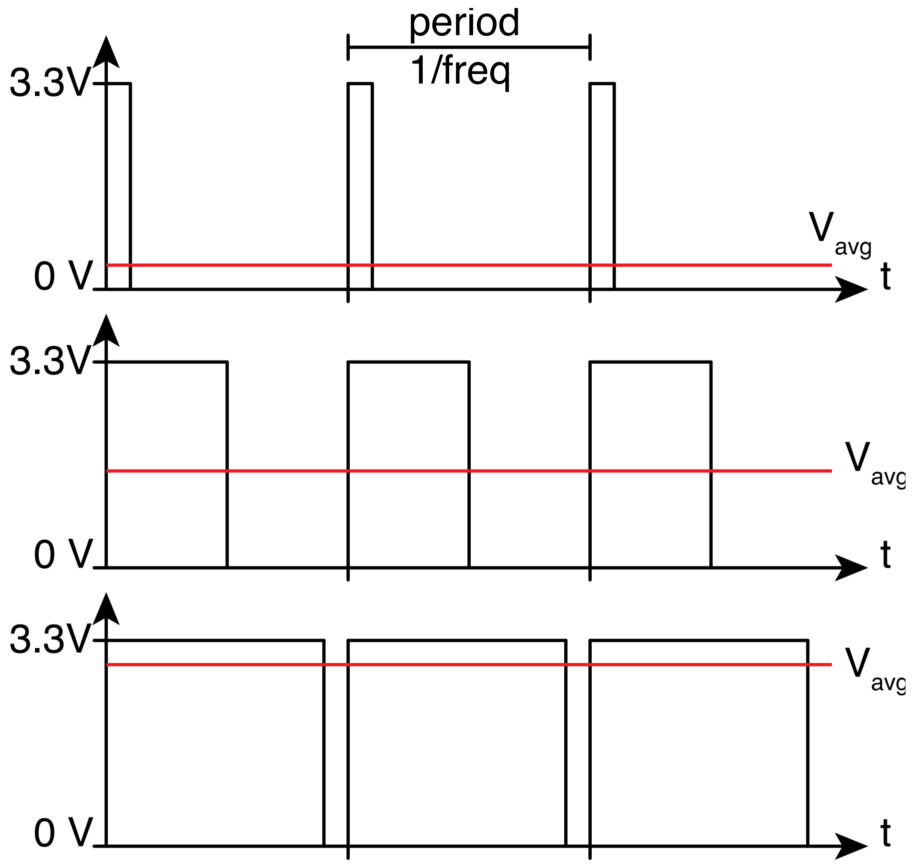
Motor Performance Diagram



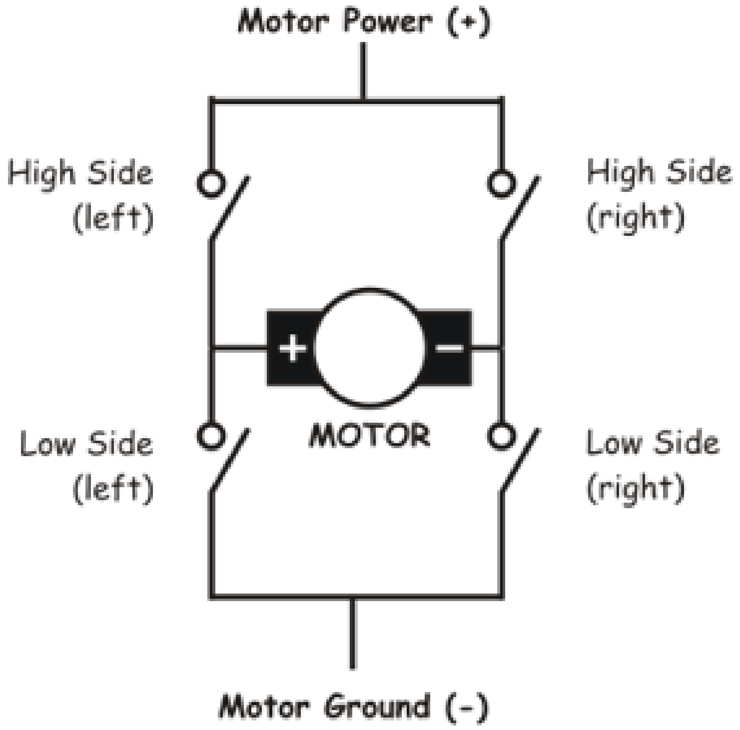
Pulse Width Modulation (PWM)

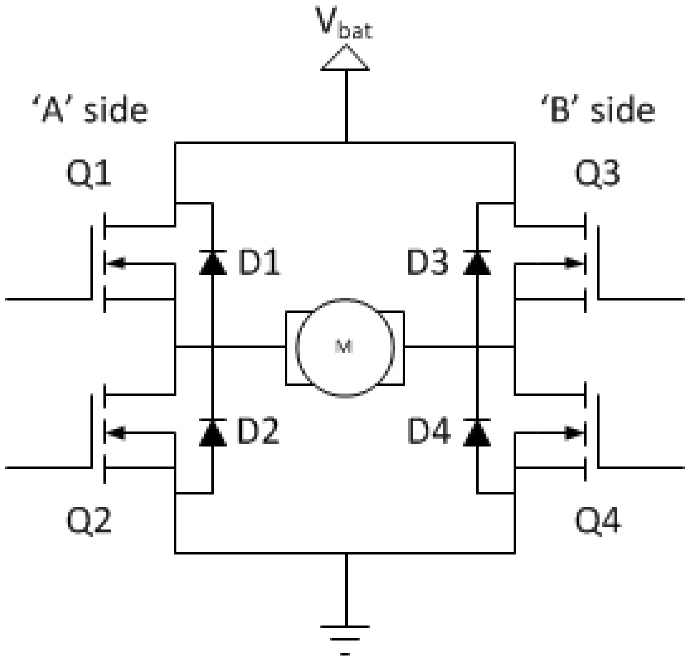
- Vary average voltage by modulating pulse width on a fixed frequency square wave
- PWM frequency typically 10k-40kHz
- PWM can be followed by low pass filter to remove high frequency switching artifacts.
- Motor coil acts as a low pass filter.



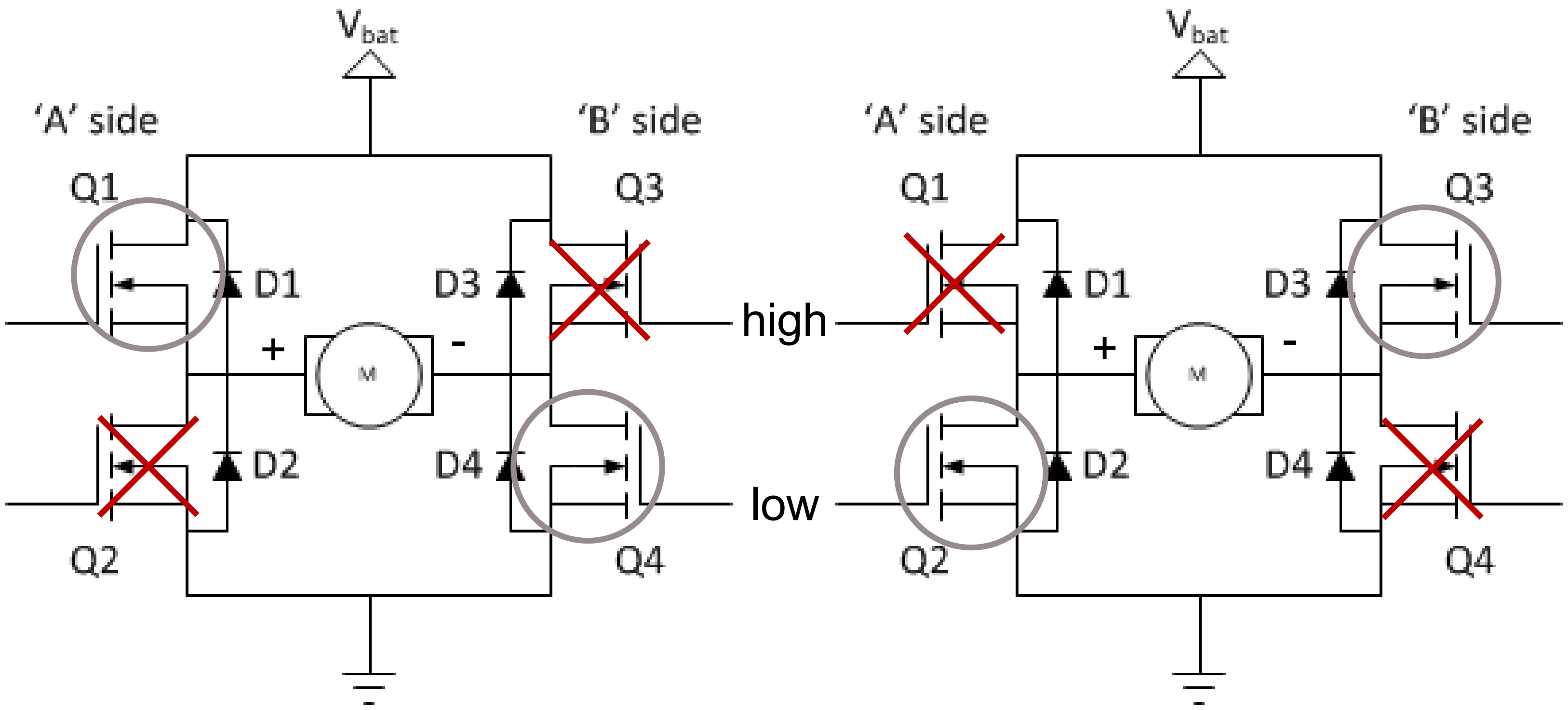


Driving Motors with an H-Bridge





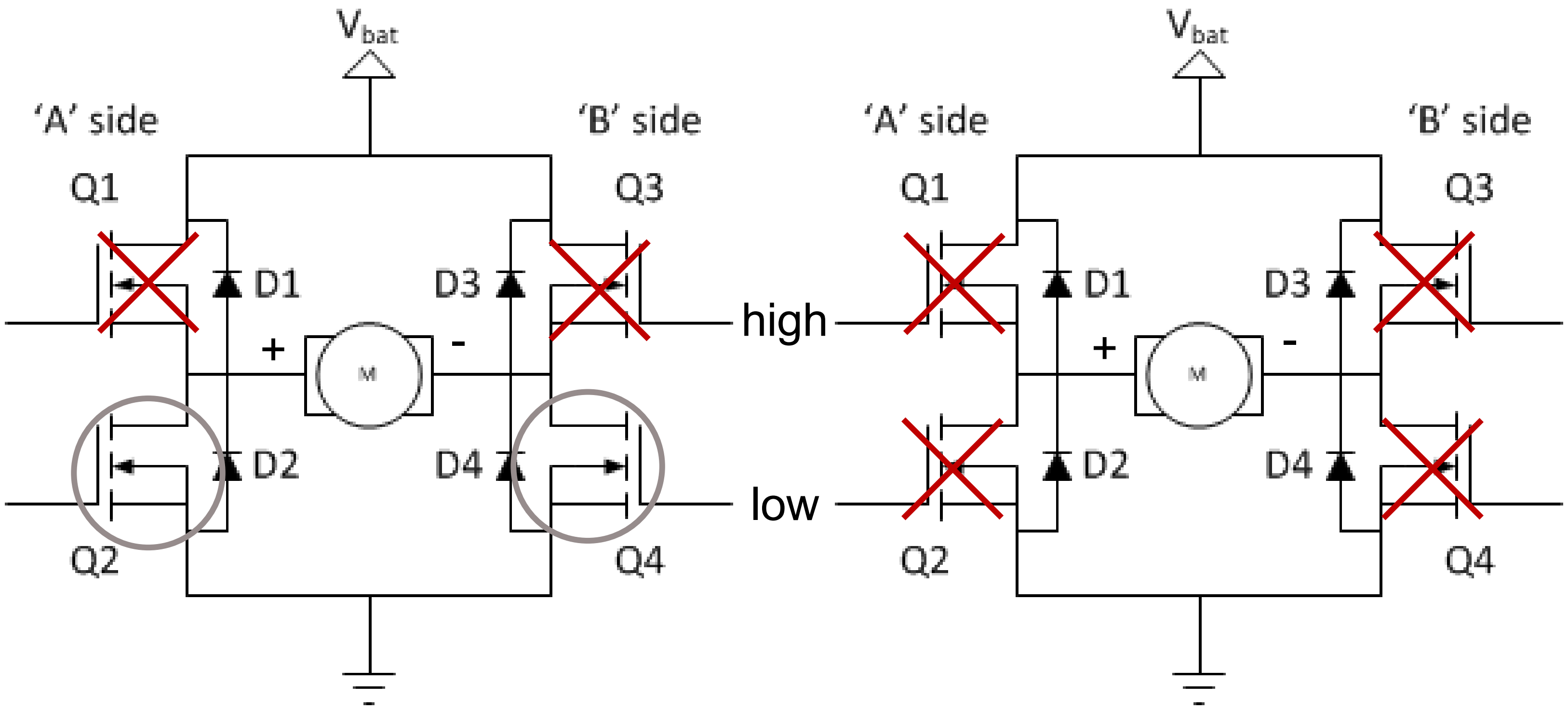
Normal Operation



Forward

Reverse

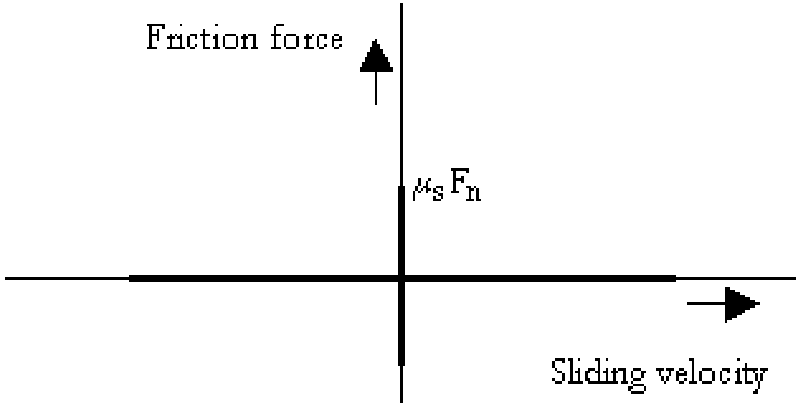
Normal Operation



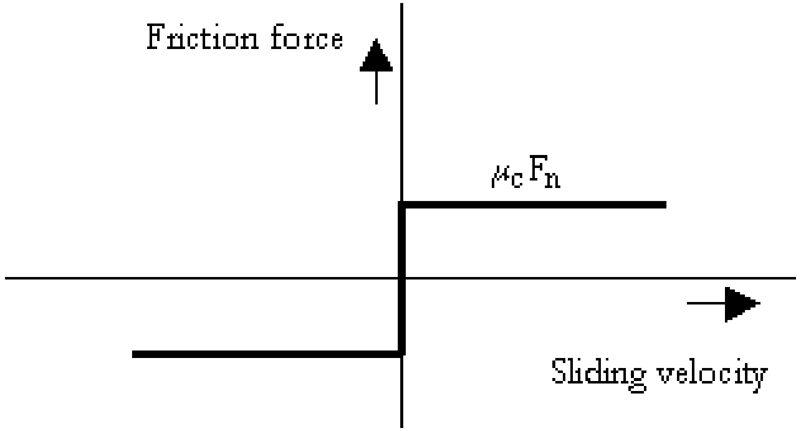
Brake

Coast

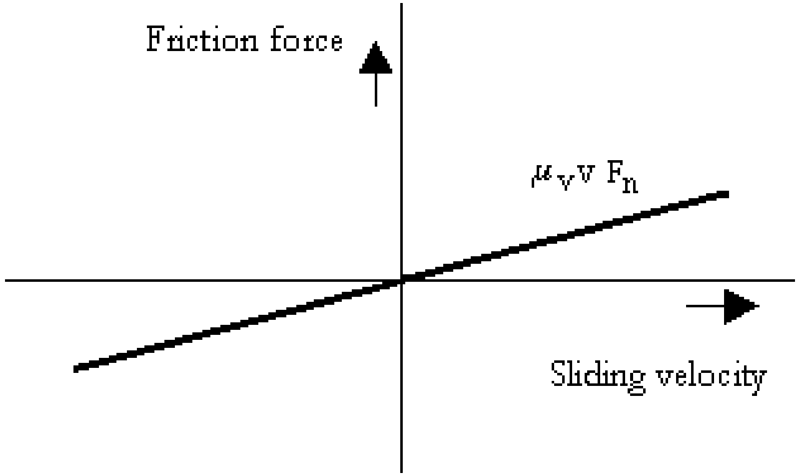
Motor Nonlinear Friction



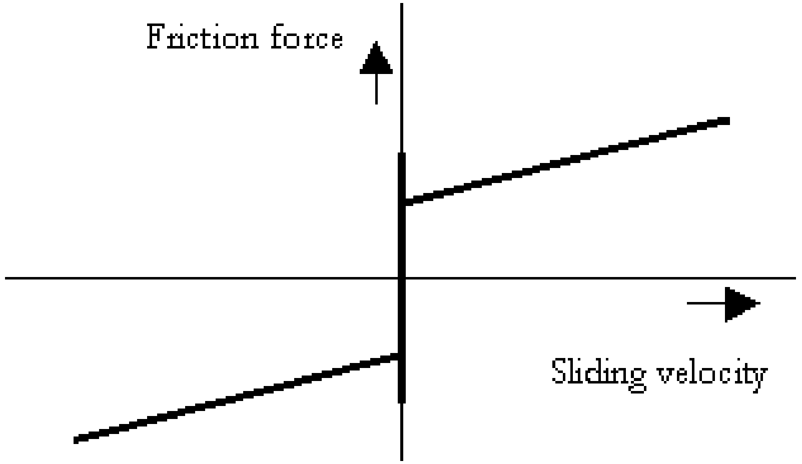
- Static: Motor will not move until torque overcomes static friction



- Coulomb: dependent only on direction of motion (standard sliding friction)



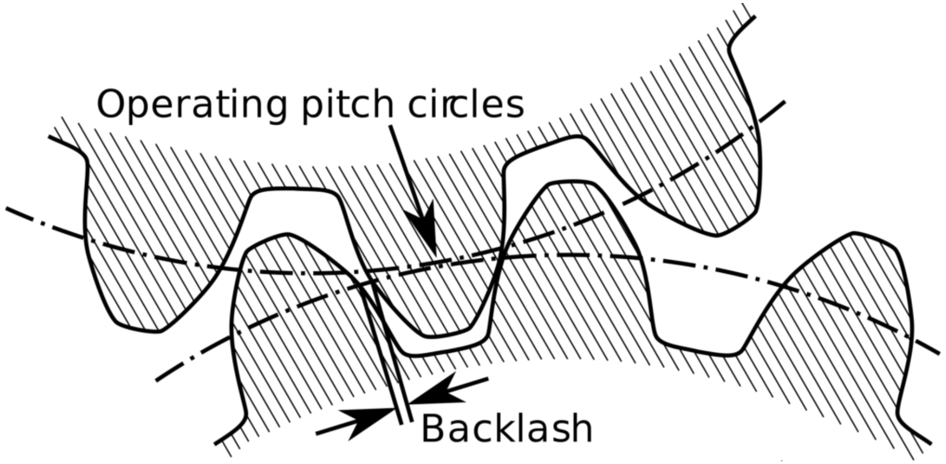
- Viscous: dependent on velocity of motion

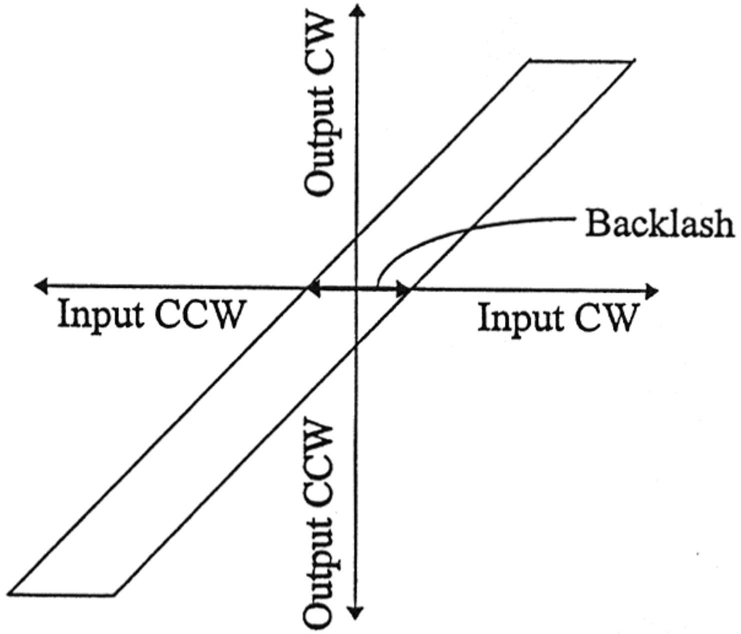


- All frictions combined

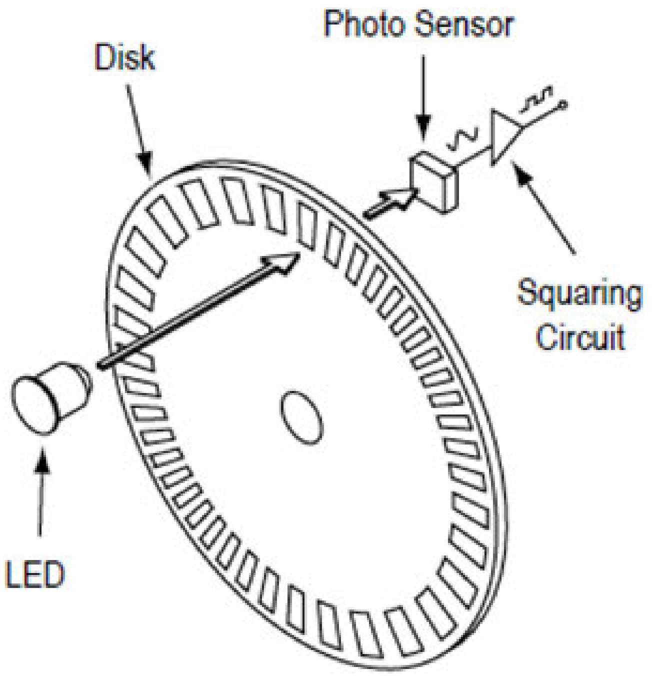
Gear Backlash

- Slack in the gear meshing means when changing motor direction, there will be hysteresis between the motor shaft and the output shaft when changing directions
- Affects encoder resolution/accuracy, especially at directional changes

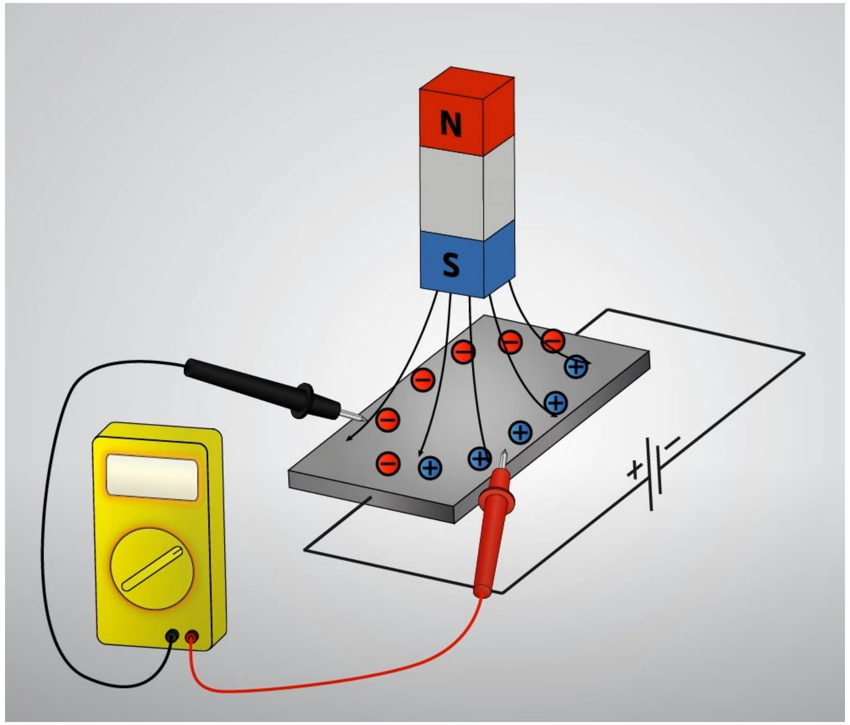




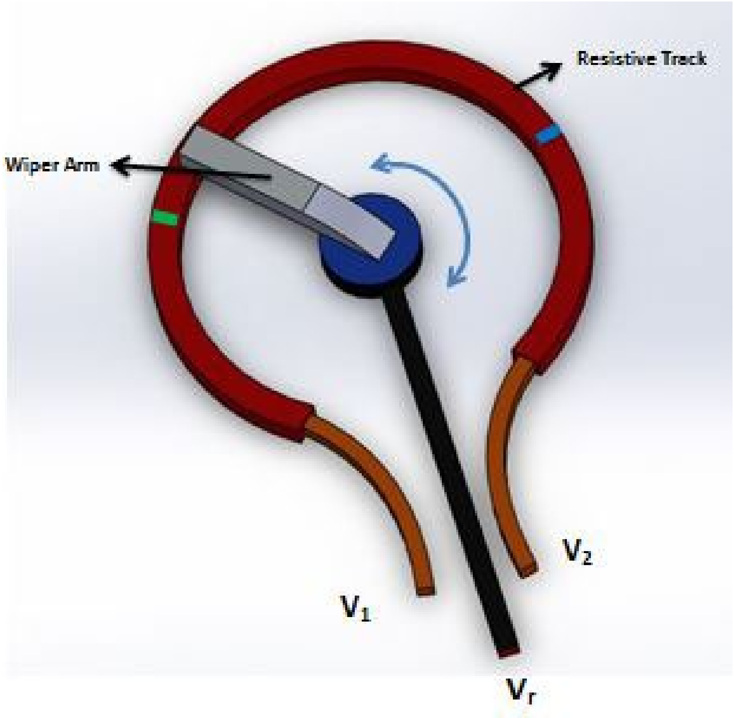
Encoders



Optical Encoder



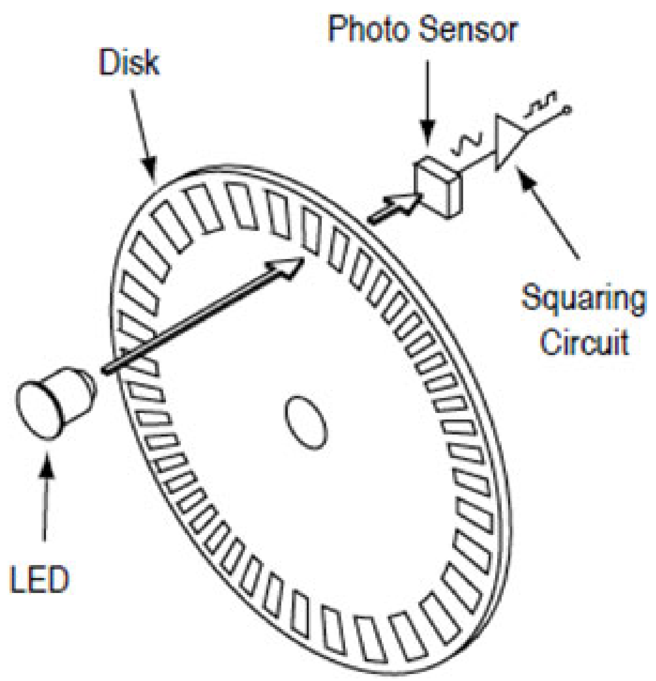
Magnetic Encoder
(Hall Effect)



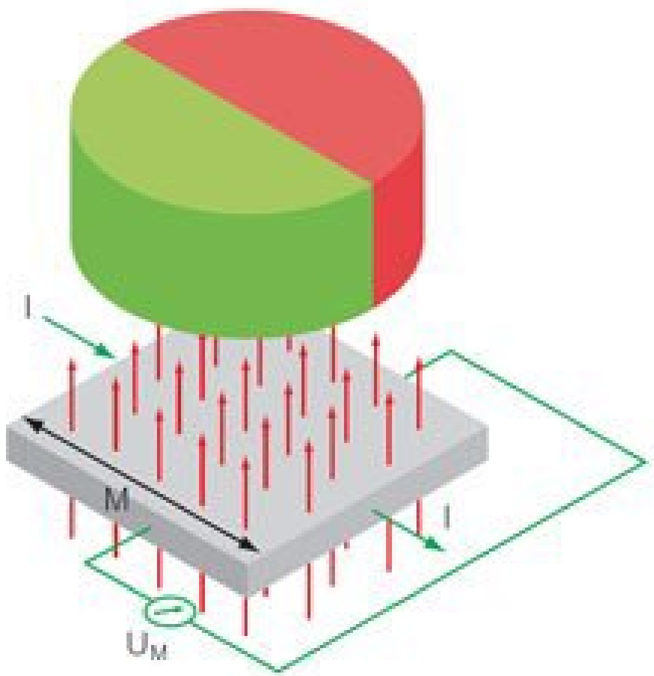
Resistive Encoder

Other types: Inductive (Resolvers), Capacitive, Laser

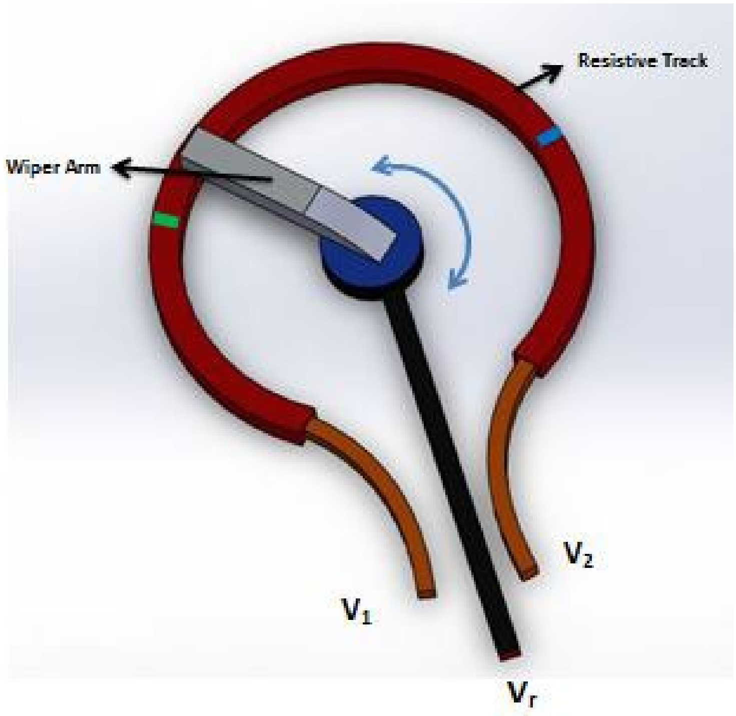
Encoders



Optical Encoder



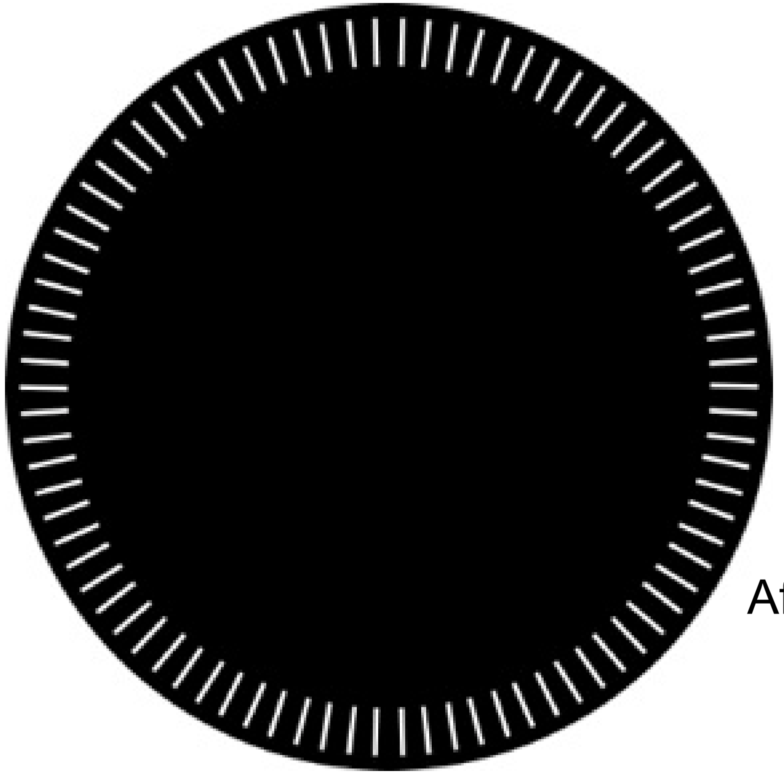
Magnetic Encoder
(Hall Effect)



Resistive Encoder

Other types: Inductive (Resolvers), Capacitive, Laser

Optical Incremental & Absolute Encoders

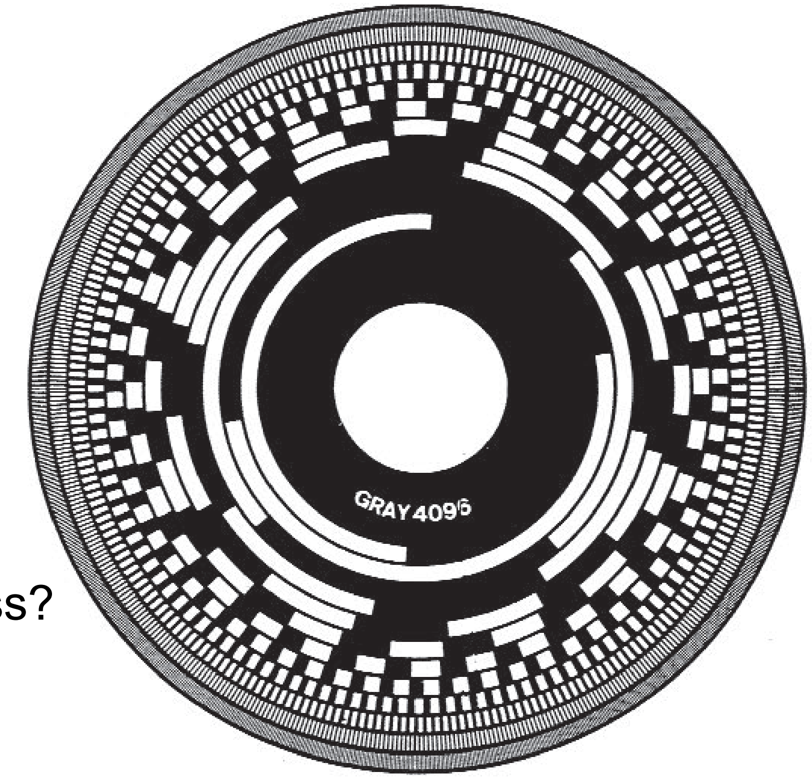


Position?

Speed?

Direction?

After power loss?

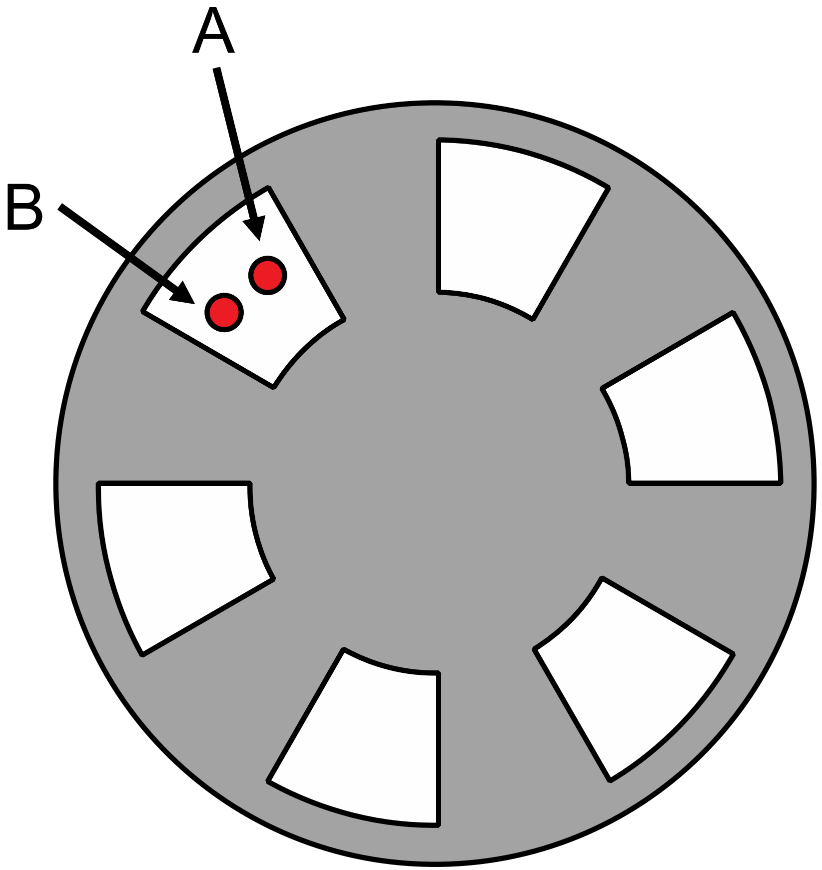


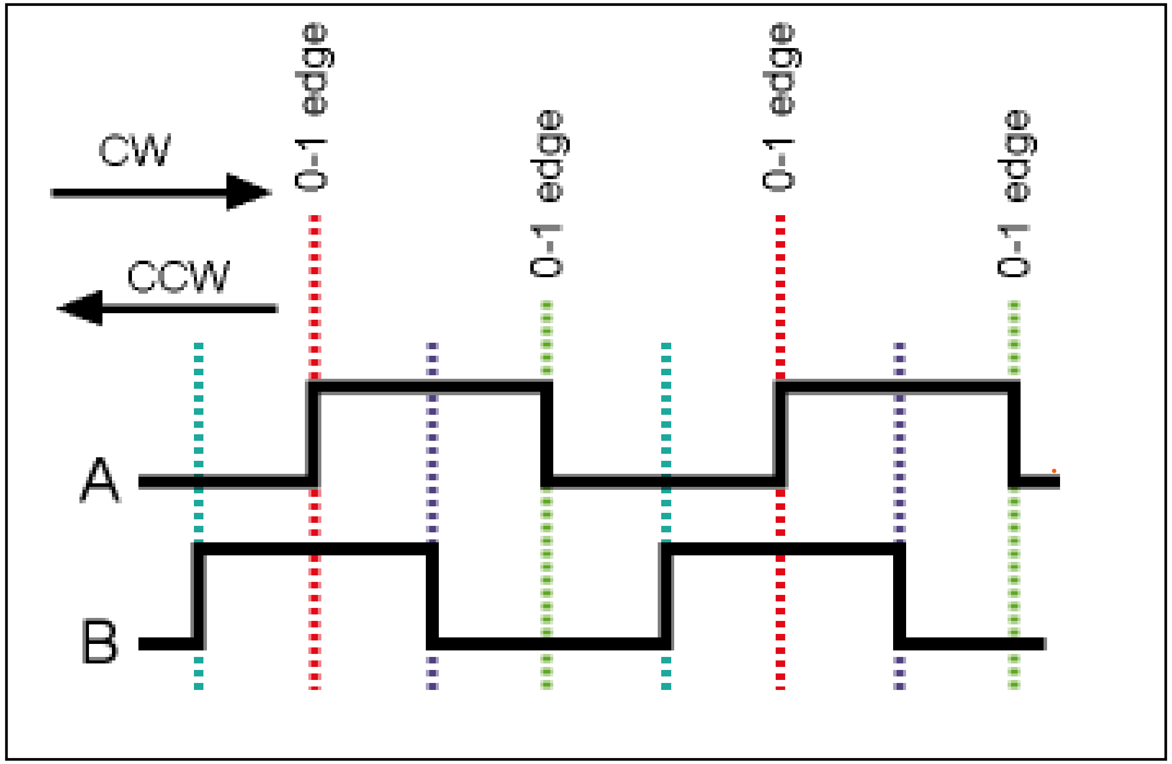
4° Incremental Encoder Wheel

12bit Absolute Encoder Wheel

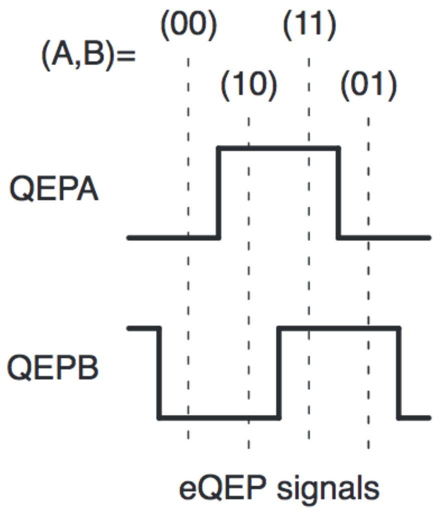
Measuring Direction: Quadrature Encoder

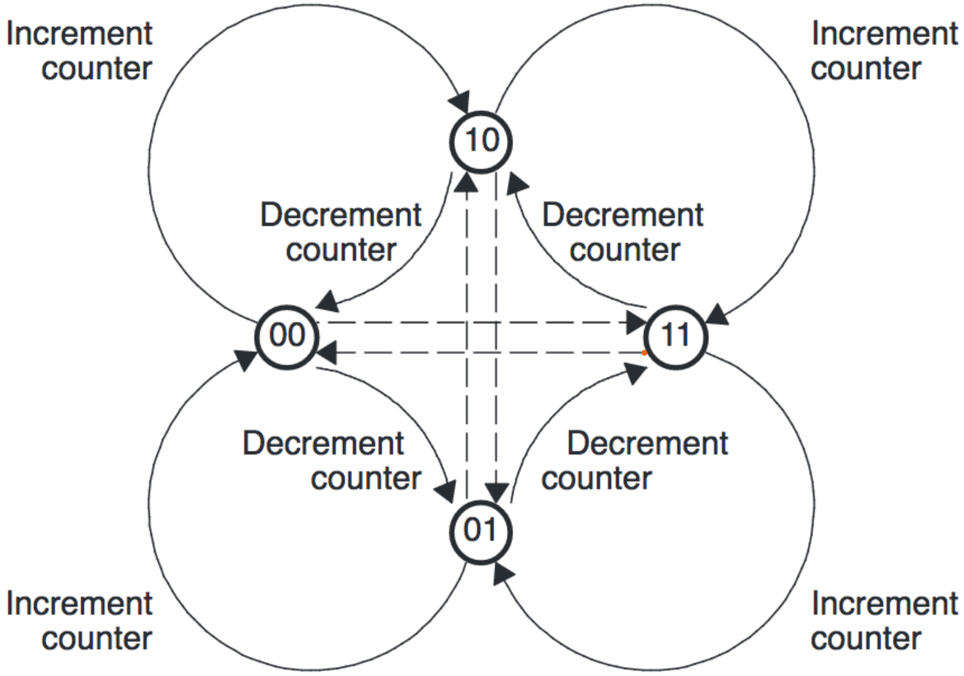
Can be various types…
Optical, magnetic, etc.





Quadrature Decoder State Machine

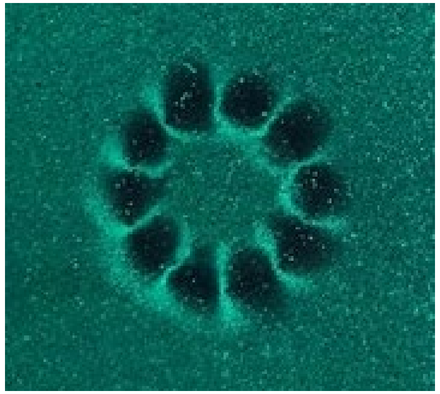




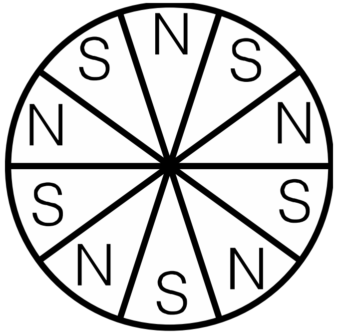
Previous Edge	Present Edge	QDIR	QPOSCNT
QA↑	QB↑	UP	Increment
	QB↓	DOWN	Decrement
	QA↓	TOGGLE	Increment or Decrement

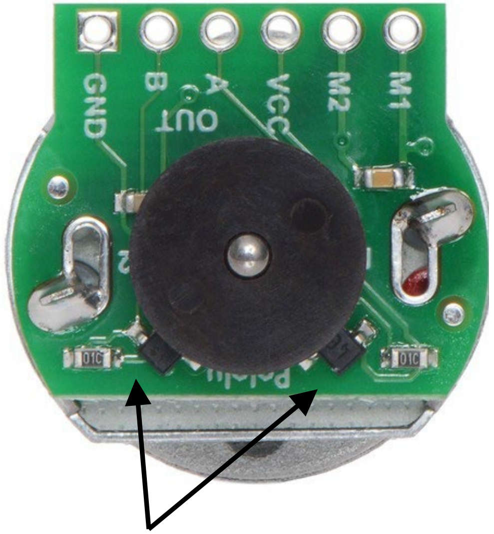
MBot encoders

- 10-pole magnet
- 20 counts per motor revolution

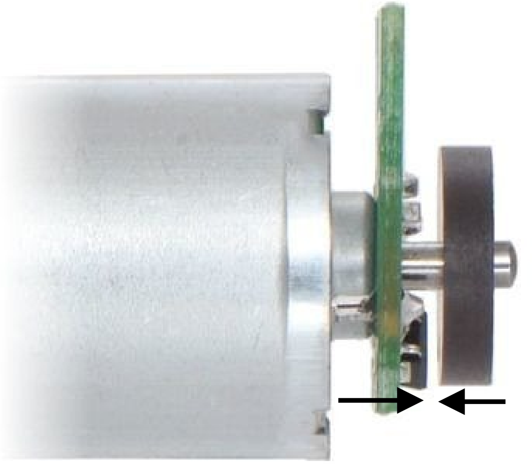


Jiawei Chen





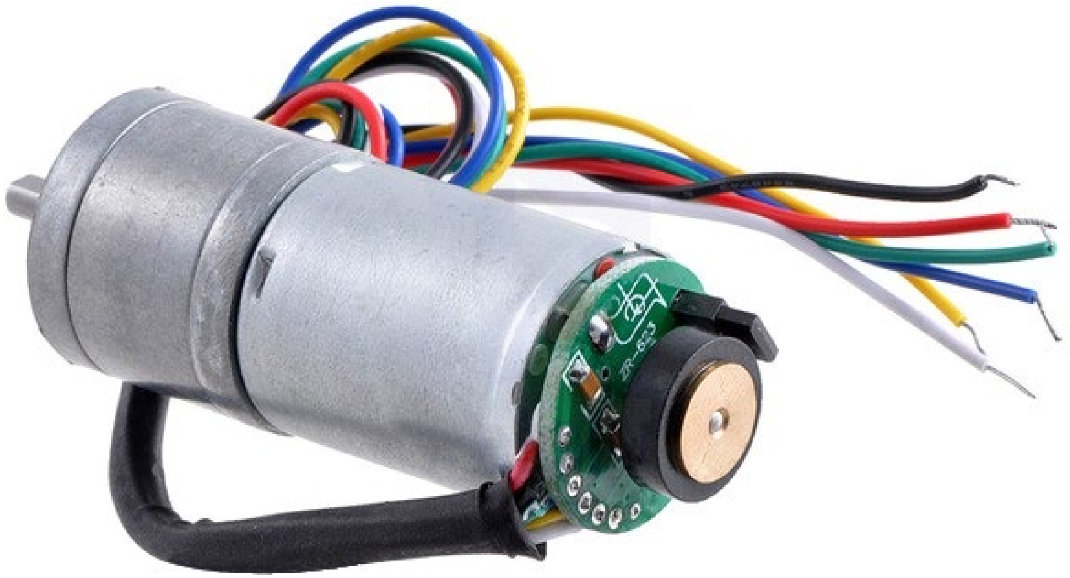
TLE4946-2K
Hall effect latch



~0.5mm

Estimating Velocity from an Encoder

- Typically, we measure the motor shaft before the gearbox
  - Why?
- If we know encoder resolution and gearbox ratio, we can calculate output shaft speed:



$$n_{gearbox} = \frac{n \text{ motor revs}}{1 \text{ wheel rev}}$$ (78:1 or 63:1)

$$n_{encoder} = \frac{n \text{ encoder ticks}}{1 \text{ motor rev}}$$ (20 counts per rev)

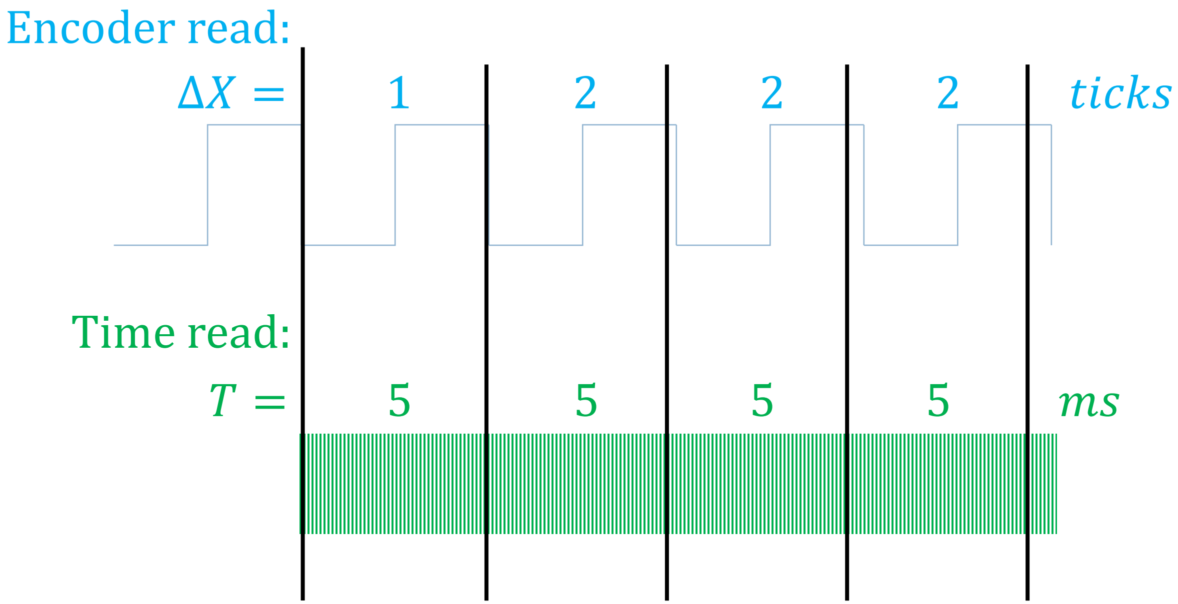
$$\frac{X \text{ encoder ticks}}{\text{time}} \times \frac{1 \text{ motor rev}}{n \text{ encoder ticks}} \times \frac{1 \text{ wheel rev}}{n \text{ motor revs}} = \frac{\text{wheel revs}}{\text{time}}$$

$$\omega_{encoder} \times \frac{1}{n_{encoder}} \times \frac{1}{n_{gearbox}} = \omega_{out}$$

Estimating Velocity from an Encoder

- Read encoder counter at time k and time k-1 and divide by the time between measurements T
- This is the typical method but has inherent accuracy limit at *low* speed
  - Why?
- A 2000 tick/rev encoder has an inherent position resolution of .0005 revolutions
- Sampled at 200Hz gives a velocity resolution of 6 rpm

$$v(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T}$$

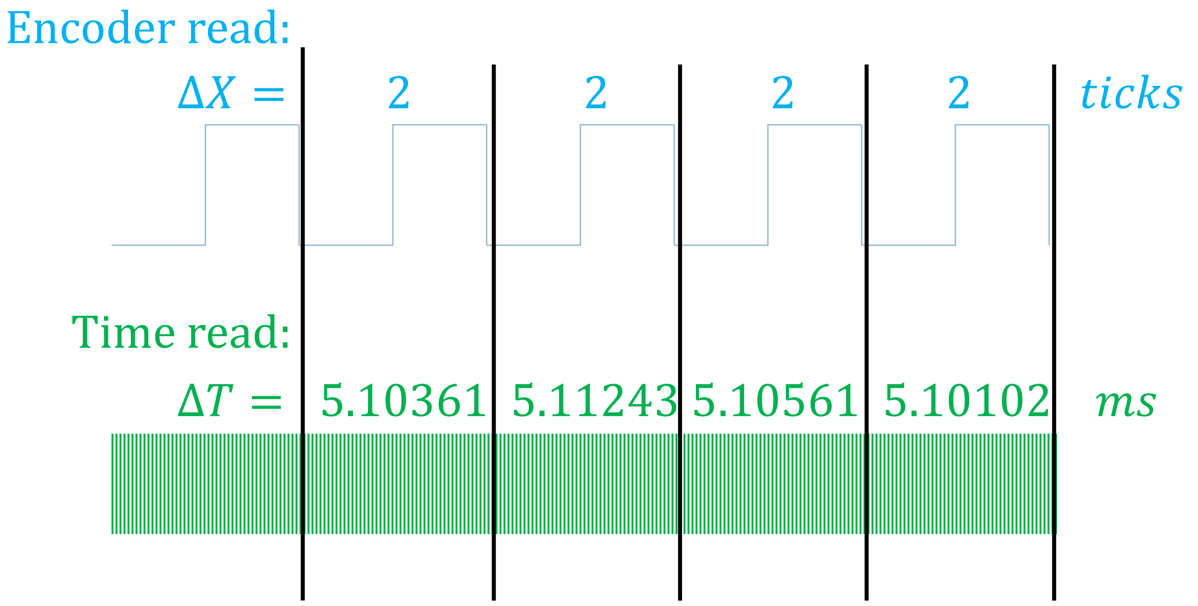


- Q: What's a better way?

Measuring Low speed

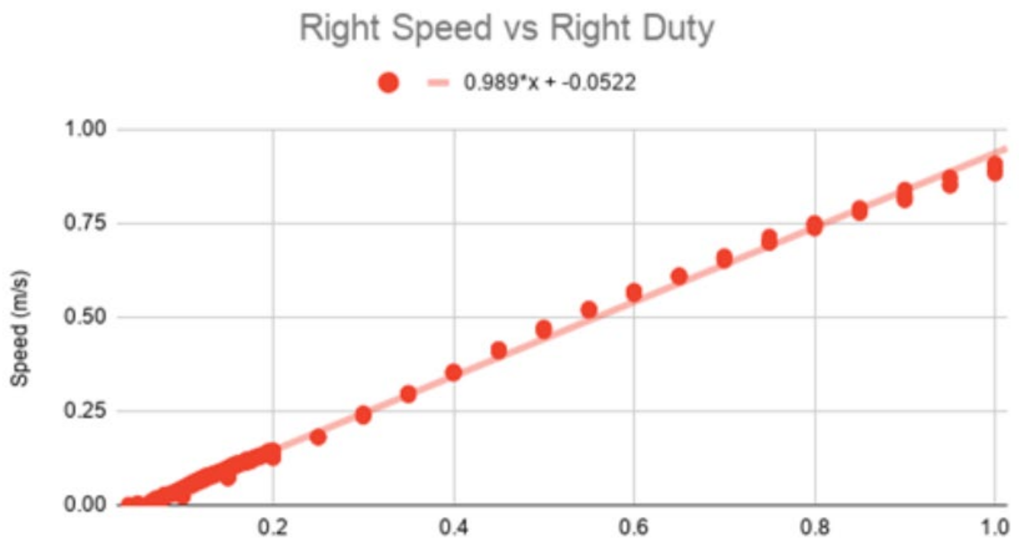
- Trigger a timer based on the rising/falling edge of encoder ticks
- Measure the time difference between ticks with a high-resolution timer
- At high speed, the accuracy becomes influenced by timer resolution as ΔT becomes small
- Not supported in RCLib software (but possible to implement!)

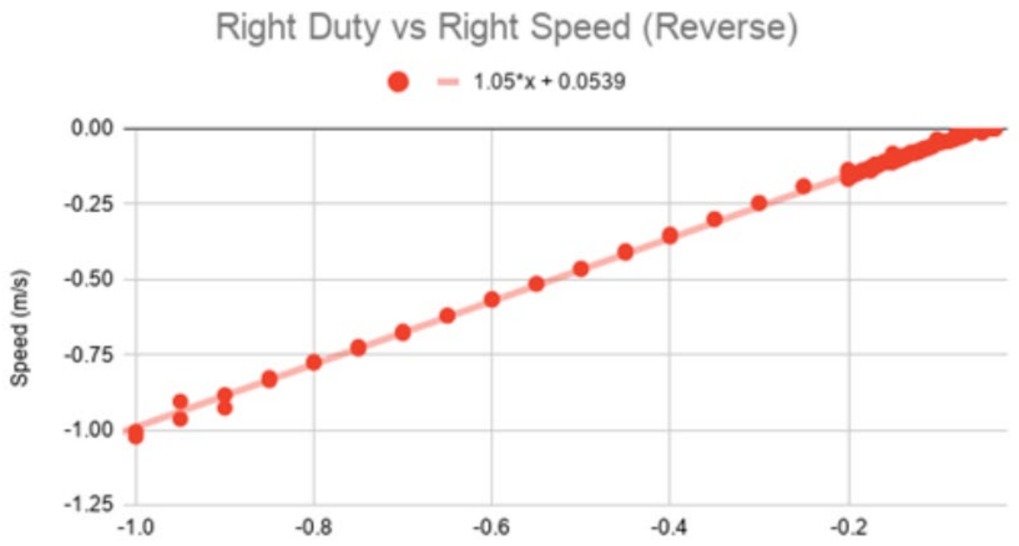
$$v(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T}$$



Calibrating Motors

- Want to know motor speed vs. PWM function - $\omega_{motor}(PWM)$
- Curve is piecewise linear (deadzone, linear, saturation)
- Can invert to fit $PWM(\omega_{motor})$
  - Remove points associated with deadzone
  - Fit remaining points to line with linear regression
  - Slope is calibration, Punch is intercept
- Calibration will depend on motor load – how do we account for this?





Sample Motor Measuring Code

```
rc_encoder_eqep_init();                  // initialize subsystems
rc_motor_init();
for(float pwm = 0.0; pwm <= 1.0 ;pwm += 0.05)
{
  rc_motor_set(1, pwm);                  // set motor command
  rc_nanosleep(2E8);                     // give time to reach steady state
  rc_encoder_eqep_write(1,0);            // reset encoder
  rc_nanosleep(1E9);                     // measure for 1s
  int ticks = rc_encoder_eqep_read(1);   // read encoder
  float speed = ticks_to_speed(ticks);   // convert to speed in rpm or rad/sec
  printf("%f, %f\n", pwm, speed);        // print to terminal
}
rc_motor_set(1,0.0);                     // cleanup
rc_encoder_eqep_cleanup();
rc_motor_cleanup();
```

Today in Lab

- New MBot Teams!
  - Introduce yourselves
  - Plan for working days/hours
  - Plan for team issues
- Get your own MBots!
  - 1 per person, not per team!

- Assemble MBots
- Flash SD Card
- Connect to MBot
  - Claim and log your bots
- Test connectivity and your own laptops
- Test hardware
- Start toward Checkpoint 1 (Motor Calibration)