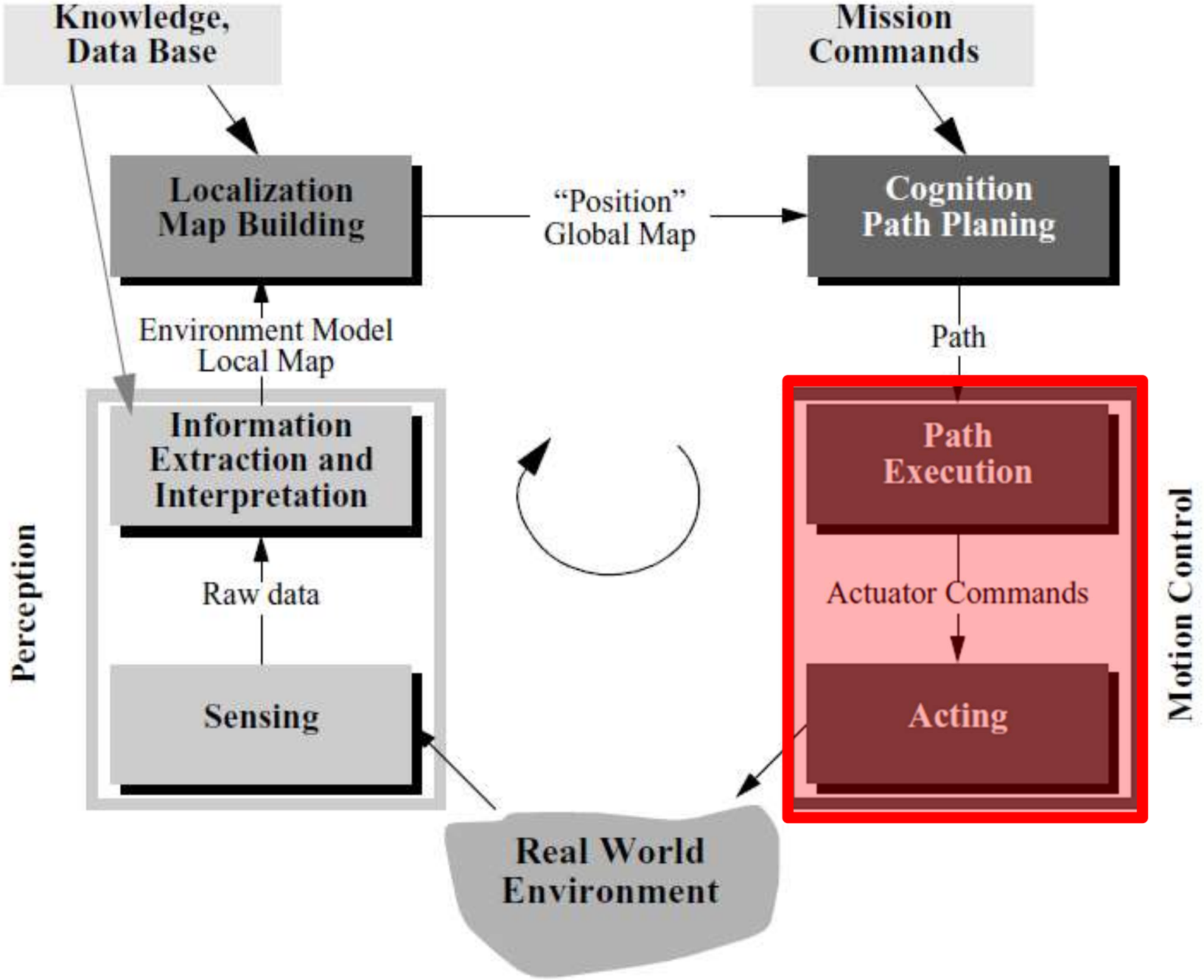
ENPM 809T

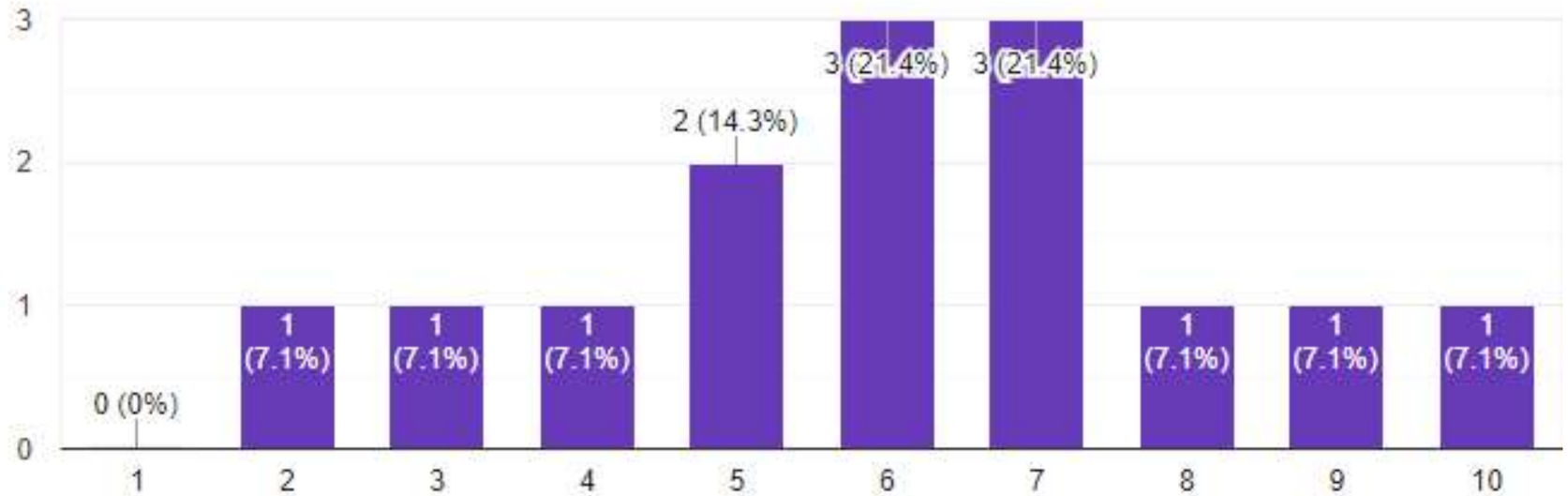
UMCP, Mitchell, Summer 2019


Knowledge,
Data Base
Mission
Commands
Localization
Map Building
"Position"
Global Map
Cognition
Path Planing
Environment Model
Local Map
Path
Information
Extraction and
Interpretation
Path
Execution
Raw data
Actuator Commands
Sensing
Acting
Perception
Motion Control
Real World
Environment

ENPM809T: Autonomous Robotics

With 10 being most comfortable, rate your comfort/experience level with robotics hardware (e.g. motors, cameras, H-bridges, etc.)

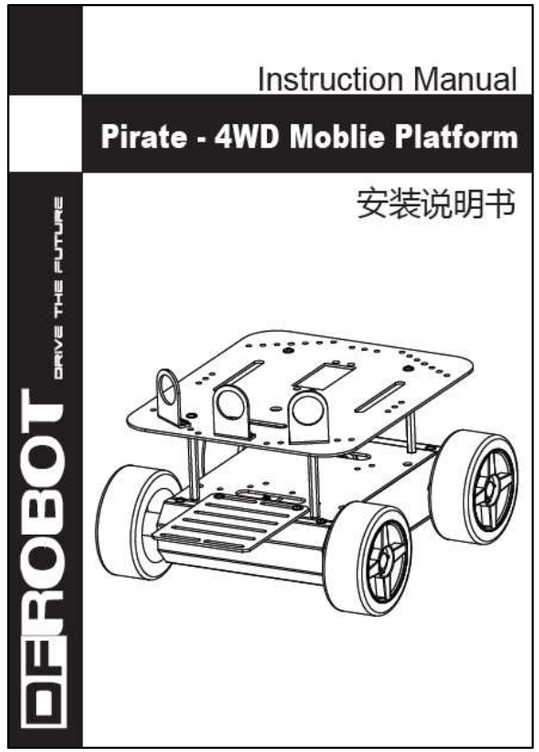
14 responses



Assembly

- Materials required to assemble ground vehicle:

1. Robot kit
2. Zip ties
3. 80-deg 3D printed camera mount
4. Teflon standoffs
5. Miscellaneous wires
6. Tools

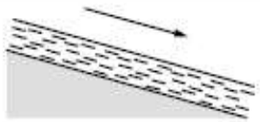
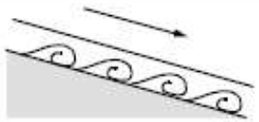
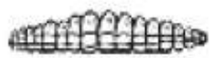
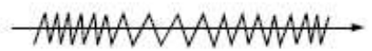
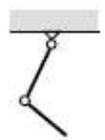
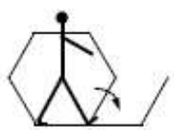


Assembly

- Unpack materials

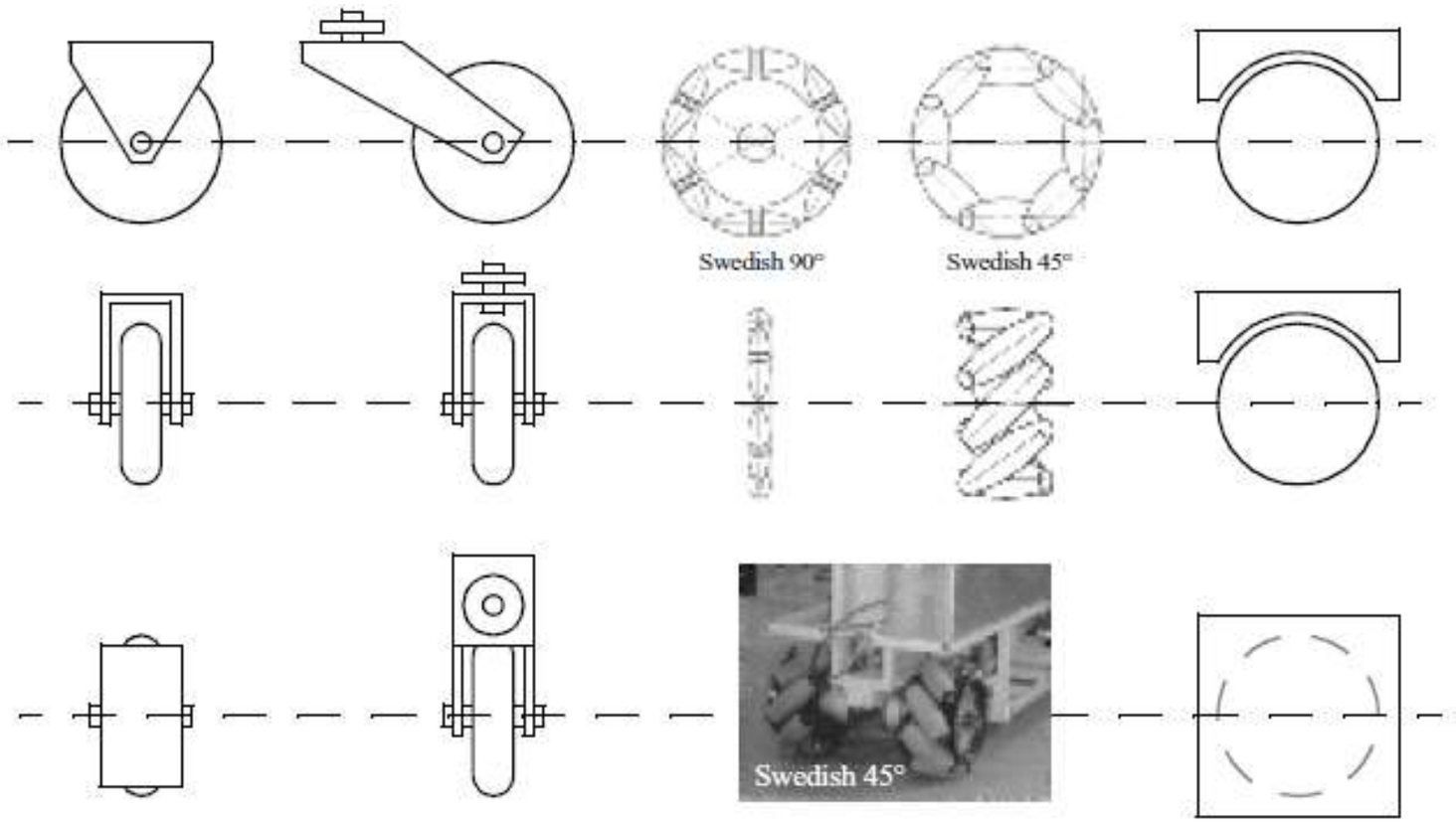
Locomotion

- Mechanisms that enable mobile robots to move unbounded throughout their environment
- Often inspired by biological systems

Type of motion	Resistance to motion	Basic kinematics of motion
Flow in a Channel 	Hydrodynamic forces	Eddies 
Crawl 	Friction forces	Longitudinal vibration 
Sliding 	Friction forces	Transverse vibration 
Running 	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum 
Jumping 	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum 
Walking 	Gravitational forces	Rolling of a polygon (see figure 2.2) 

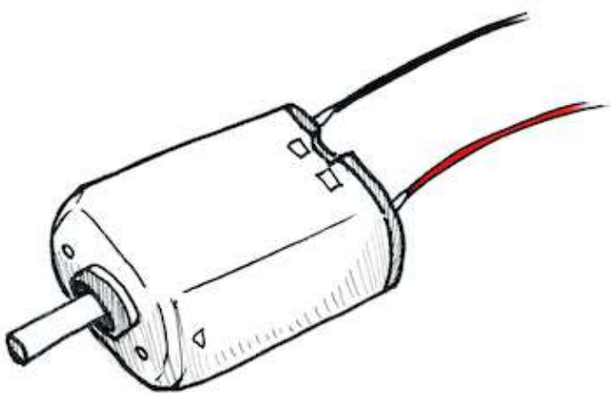
Locomotion

- Actively powered wheel achieves high efficiency on flat ground



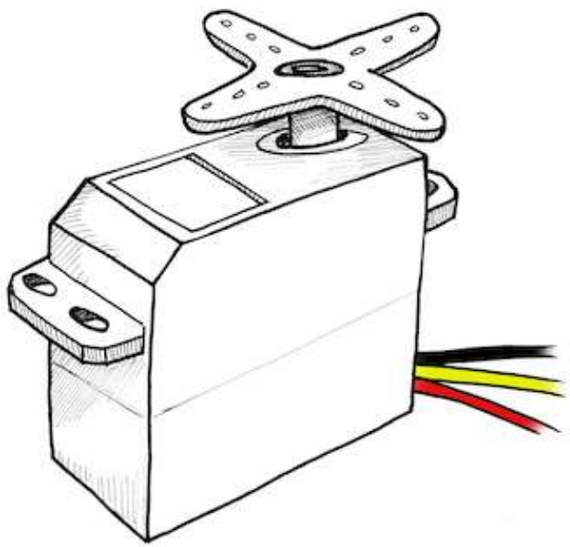
Locomotion

DC Motor



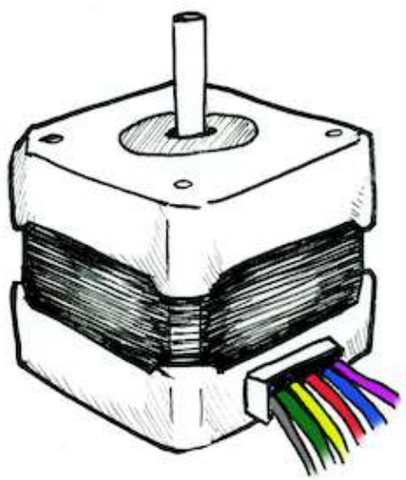
2 Wires

Servo Motor

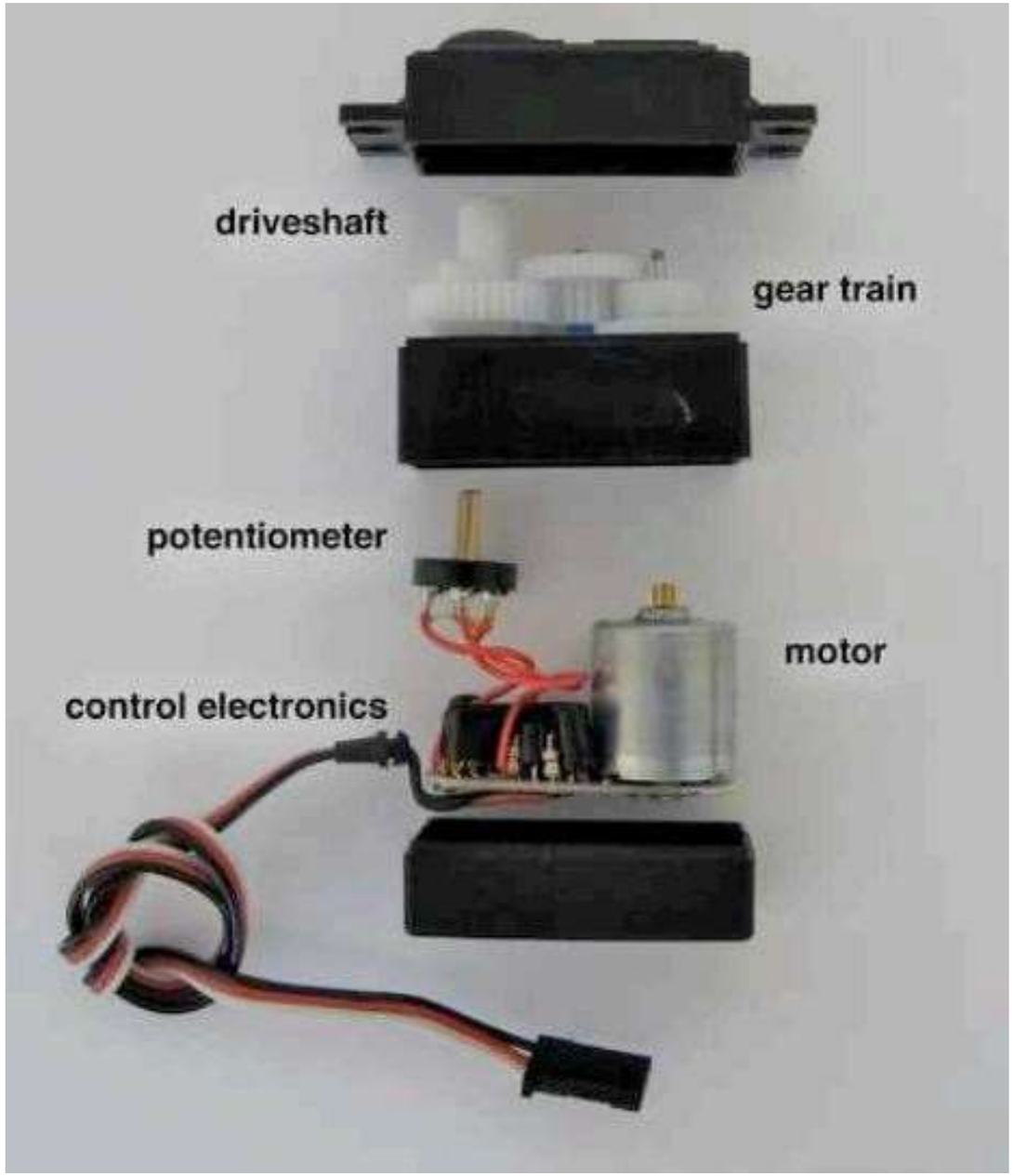


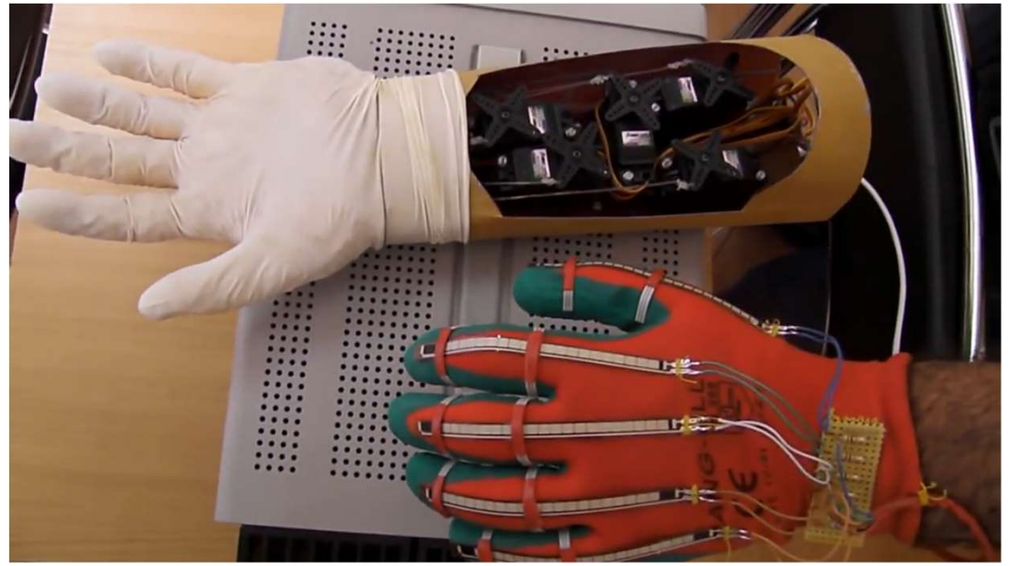
3 Wires

Stepper Motor



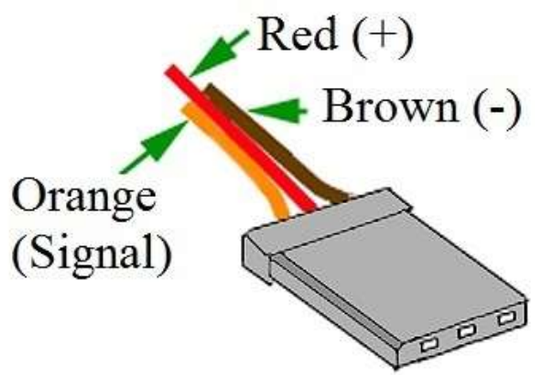
Lots of wires


driveshaft
gear train
potentiometer
motor
control electronics

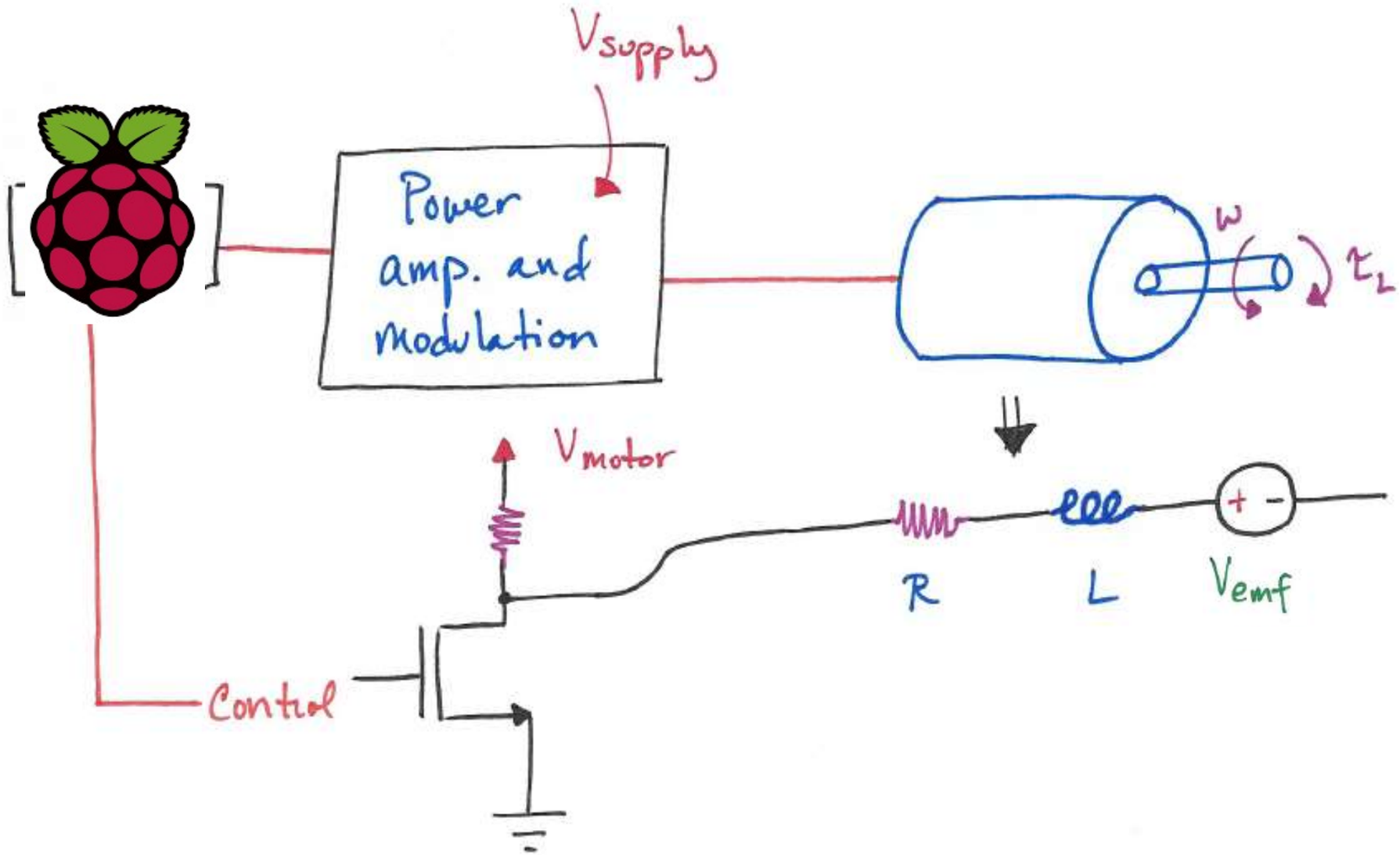



TOWER PRO
MICRO SERVO

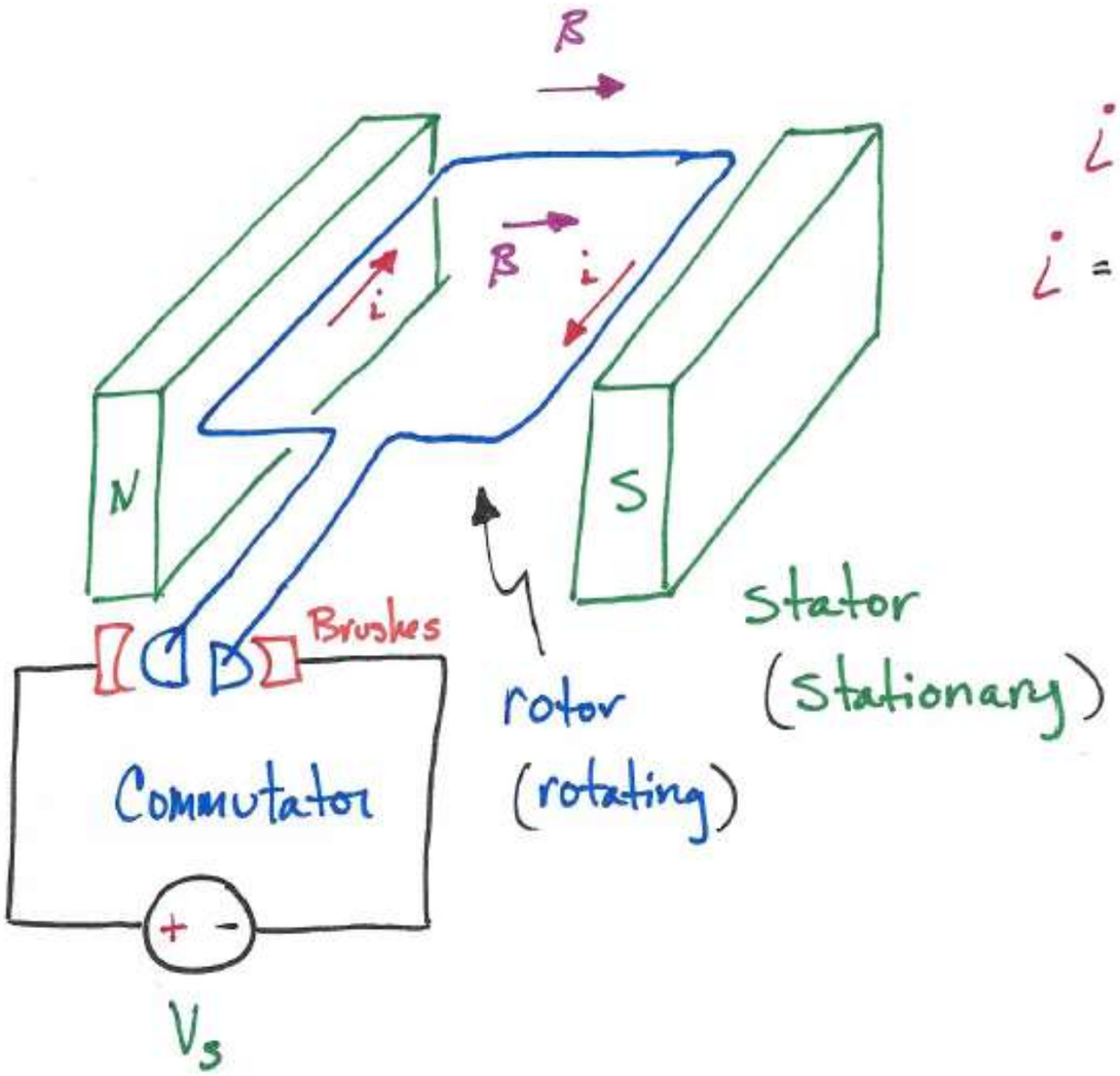

OSOYOO
Micro Servo Motor
SG90


Red (+)
Brown (-)
Orange
(Signal)

Electromagnetic actuation



What's inside a brushed DC motor?



What's inside a brushed DC motor?

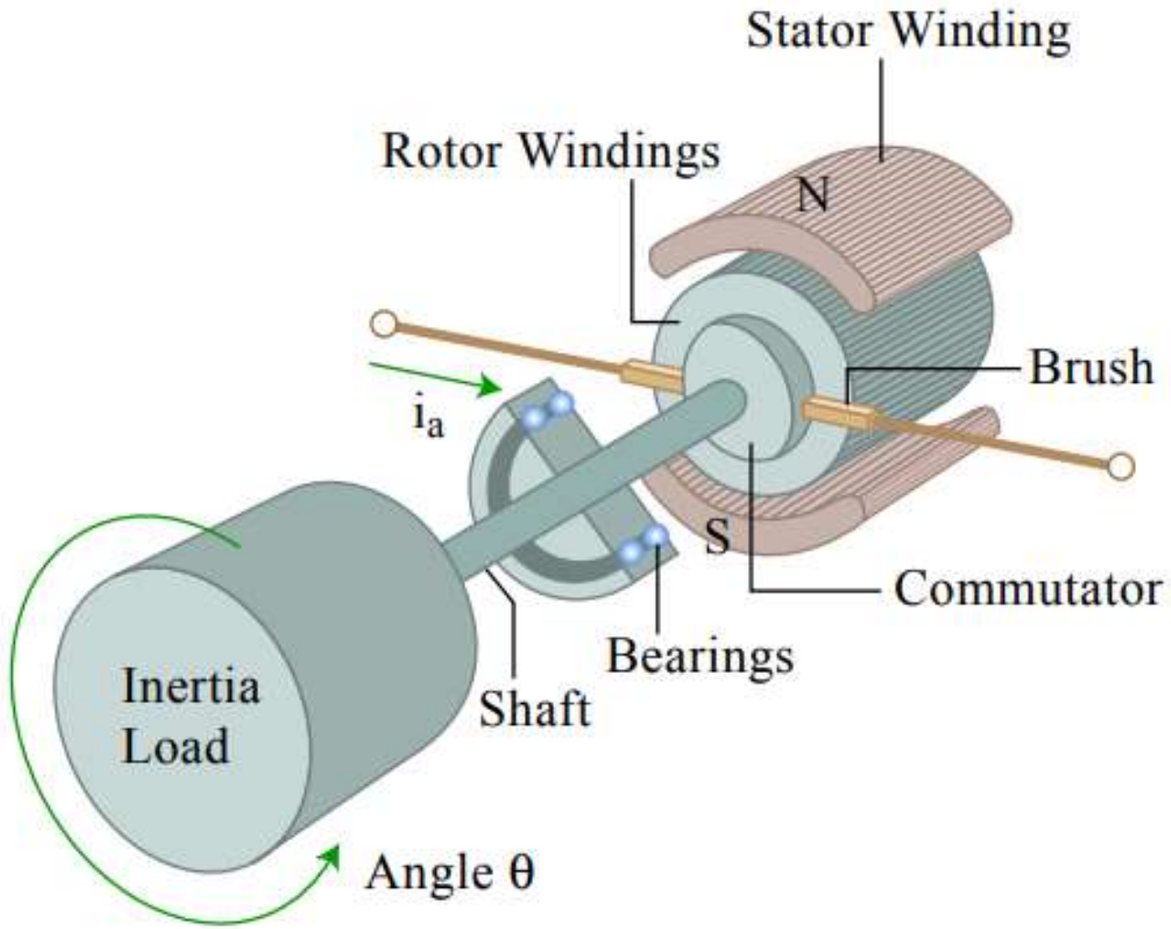


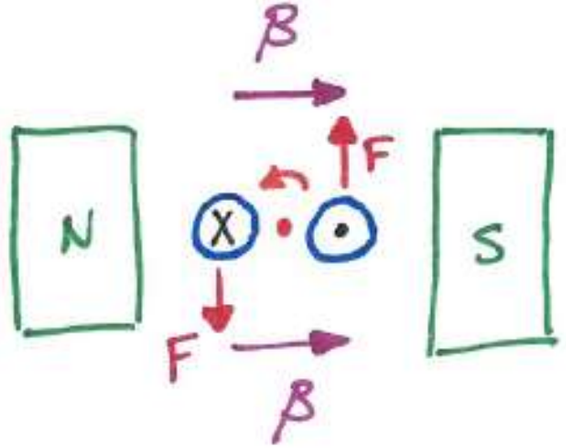
Image: Asada

Motor law

Lorentz Force Law

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

coil current — length of coil — magnetic field



current

$$F = i l B \sin\theta \qquad \sin\theta,\ \theta = 90^\circ$$

$$F = i l B$$

$$T = i\, l_{coil_1} \cdot l_{coil_2} \cdot B$$

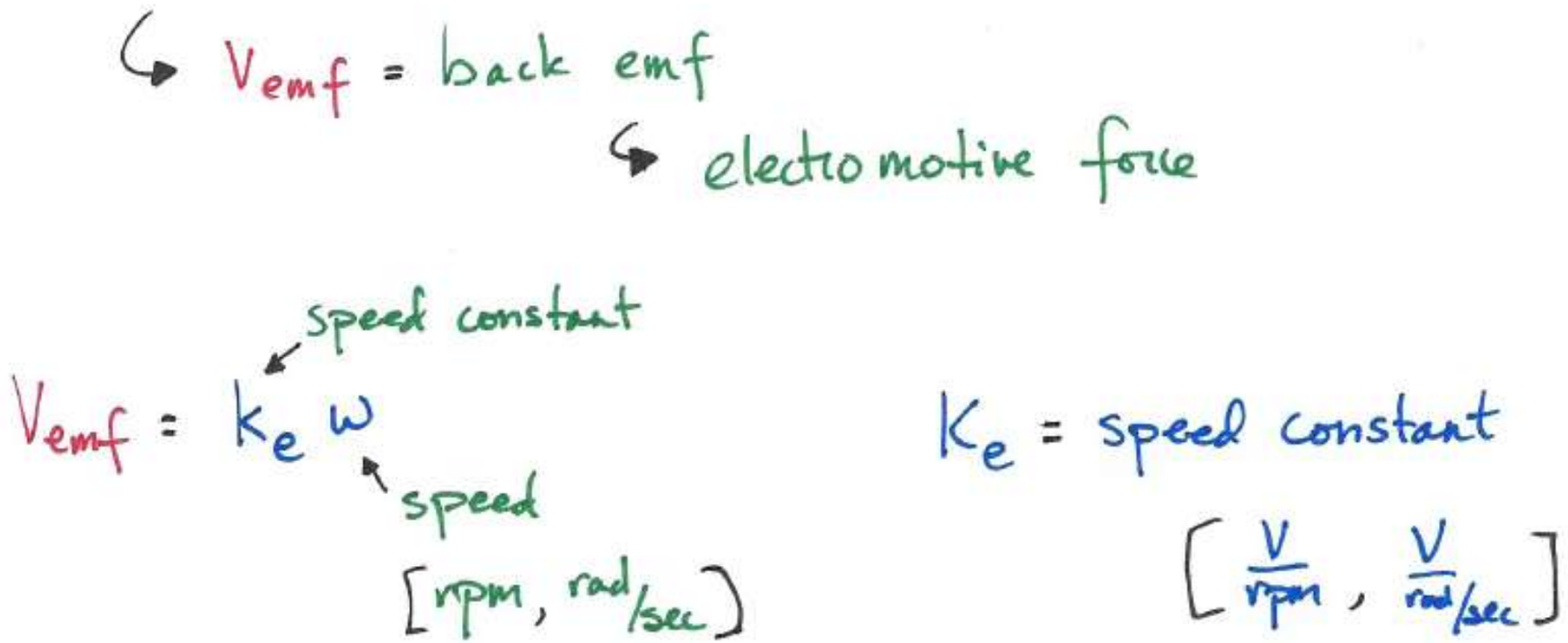
Motor parameters

torque

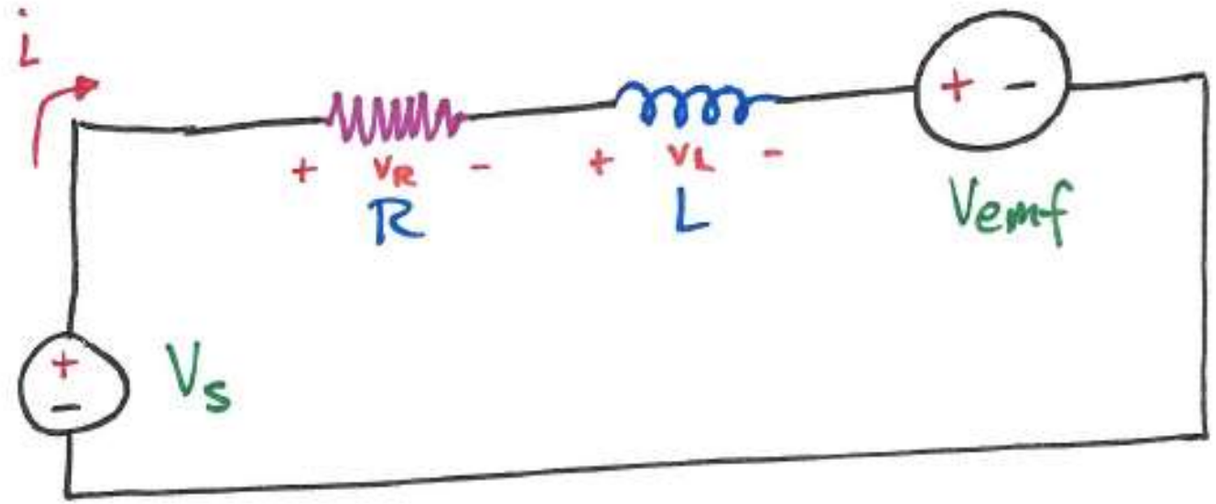
$$T = K_T i \leftarrow \text{Motor Law}$$

Generator law

- Faraday's Law: rotating a coil through a magnetic field will generate a **voltage** to **oppose** rotation



DC motor model



R = coil resistance [Ω]

L = coil inductance [H]

$V_{emf} = K_e \omega$

V_s = supply voltage

i = coil current

KVL $\rightarrow$ $V_s = V_R + V_L + V_{emf}$

$= iR + K_e \omega$

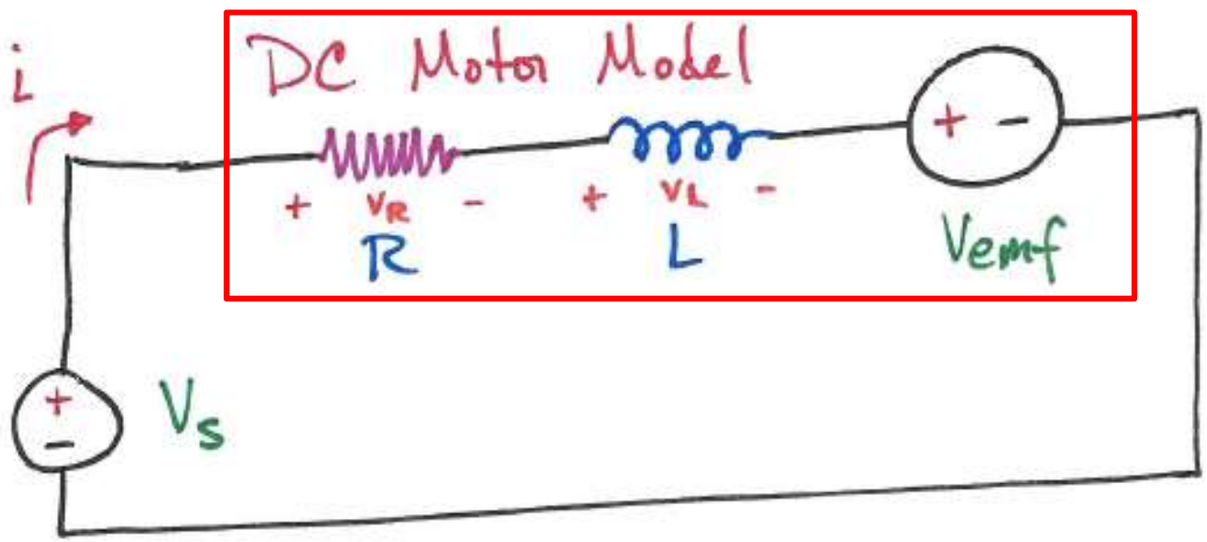
$i = \frac{T}{K_T}$ ← T = torque

ω = speed

$V_L = L \frac{di}{dt}$

$\therefore V_s = \frac{T}{K_T} R + K_e \omega$

DC motor model



$$\omega = \frac{V_s}{K_e} - \frac{R}{K_e K_T} \cdot T$$

$$T = \frac{K_T V_s}{R} - \frac{K_e K_T}{R} \cdot \omega$$

$$V_s = \frac{T}{K_T} R + K_e \omega$$

- T: torque
- K_T: torque constant
- R: resistance in coil
- K_e: speed constant
- ω: speed

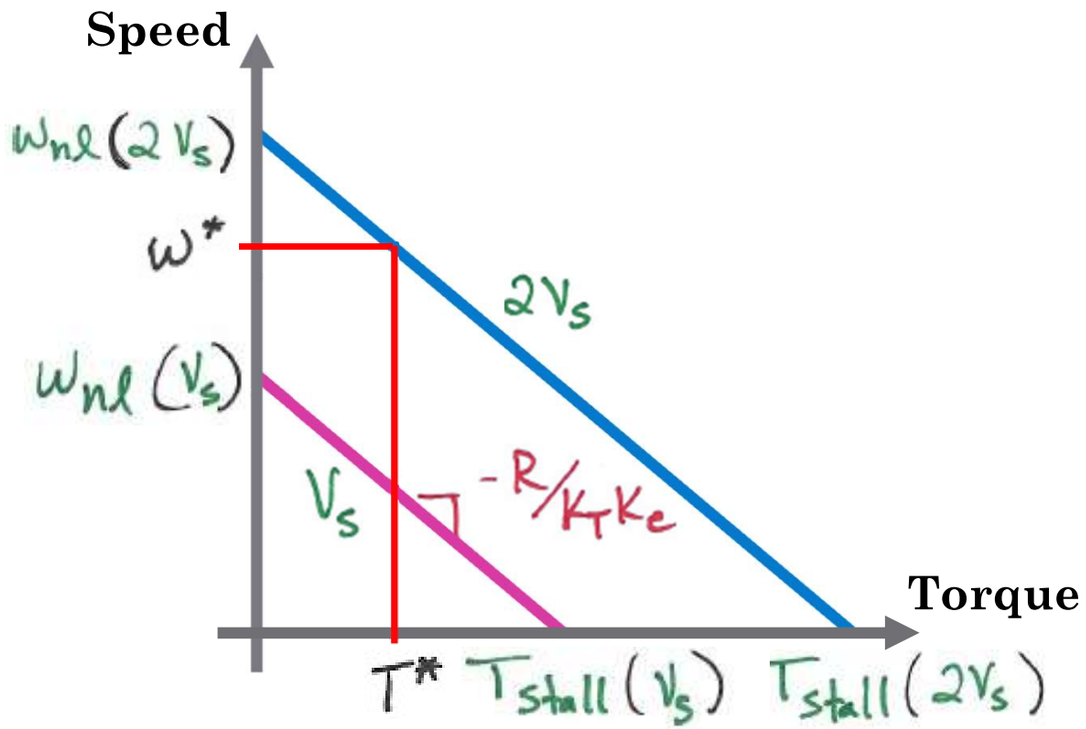
Torque vs. speed

$$\omega = \frac{V_s}{K_e} - \frac{R}{K_e K_T} \cdot T$$

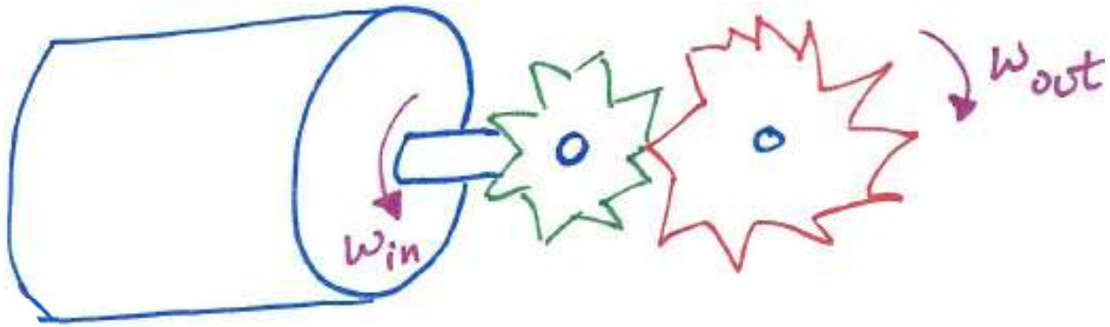
$$\omega_{no\text{-}load} = \frac{V_s}{K_e} = \omega_{max}$$

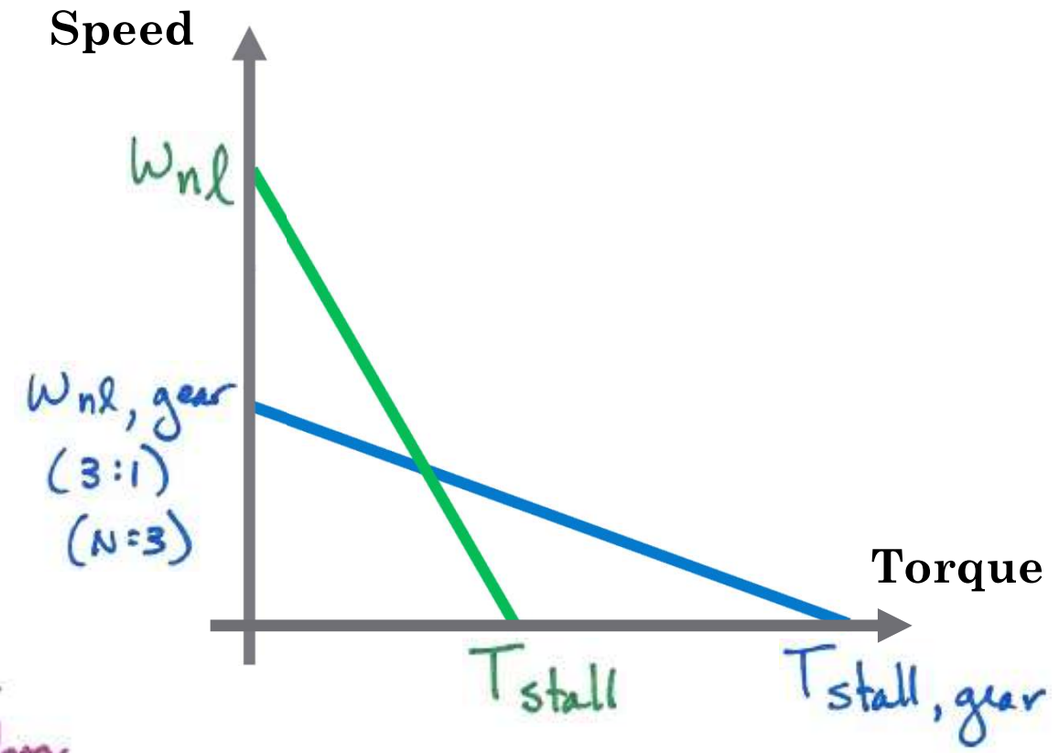
$$T = \frac{K_T V_s}{R} - \frac{K_e K_T}{R} \cdot \omega$$

$$T_{stall} = \frac{K_T V_s}{R} = T_{max}$$



DC motor gearing





Gear Ratio

$$N = \frac{\omega_{in}}{\omega_{out}} = \frac{\text{input speed from motor}}{\text{output speed from gearbox}}$$

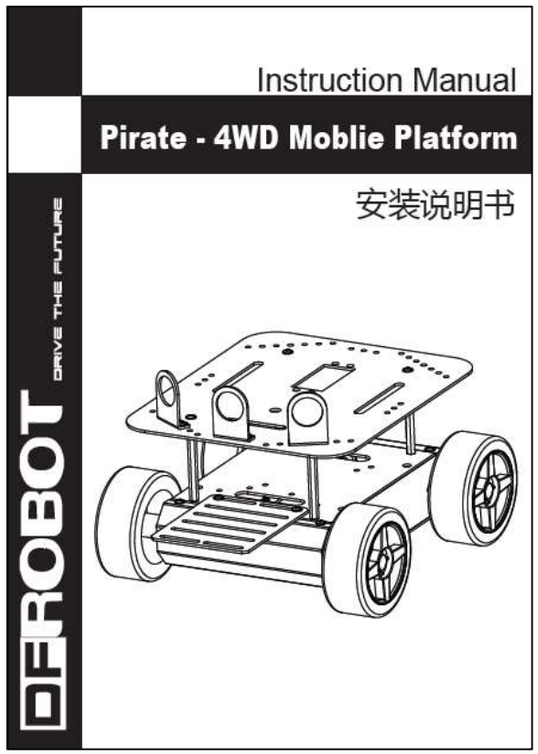
$\omega_{nl} = 1000$ rpm

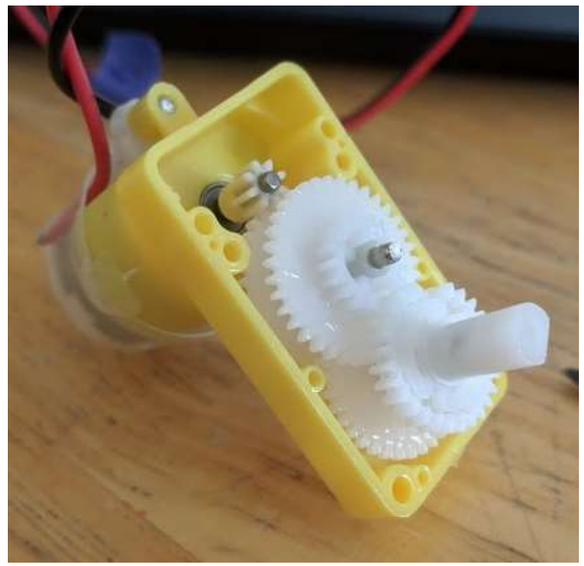
10:1 gear ratio $(N = 10)$

$$\eta_{gearbox} = \frac{T_{out}\,\omega_{out}}{T_{in}\,\omega_{in}}$$

$$\frac{T_{out}}{T_{in}} = N \cdot \eta_{gearbox}$$

DC motor gearing



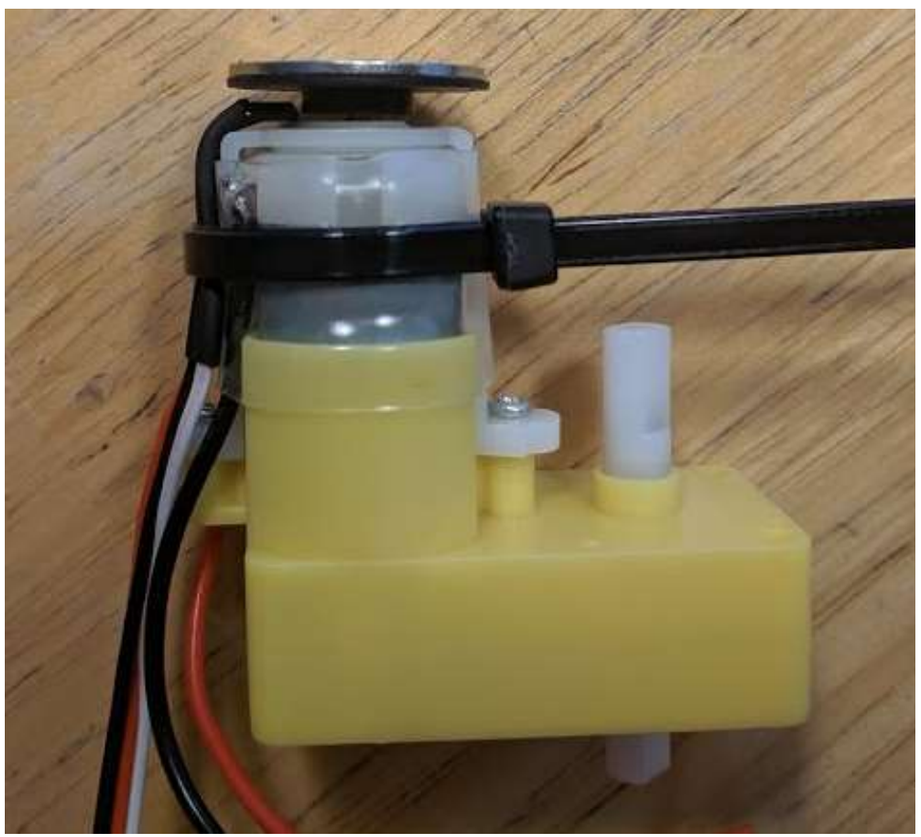


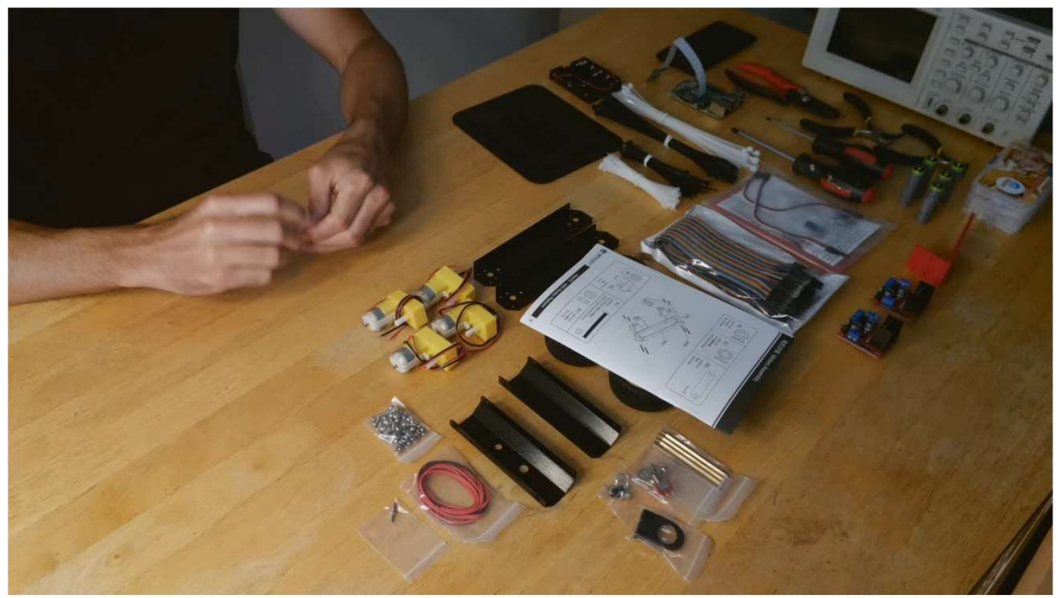
Motor Feature :

- Gear Ratio : 1:120
- No-load speed (3V): 100RPM
- No-load speed (6V): 200RPM
- No-load current (3V): 60mA
- No-load current (6V): 71mA
- Stall current (3V): 260mA
- Stall current (6V): 470mA
- Torgue (3V): 1.2Kgcm
- Torque (6V): 1.92Kgcm
- Weight : 45g

Assembly

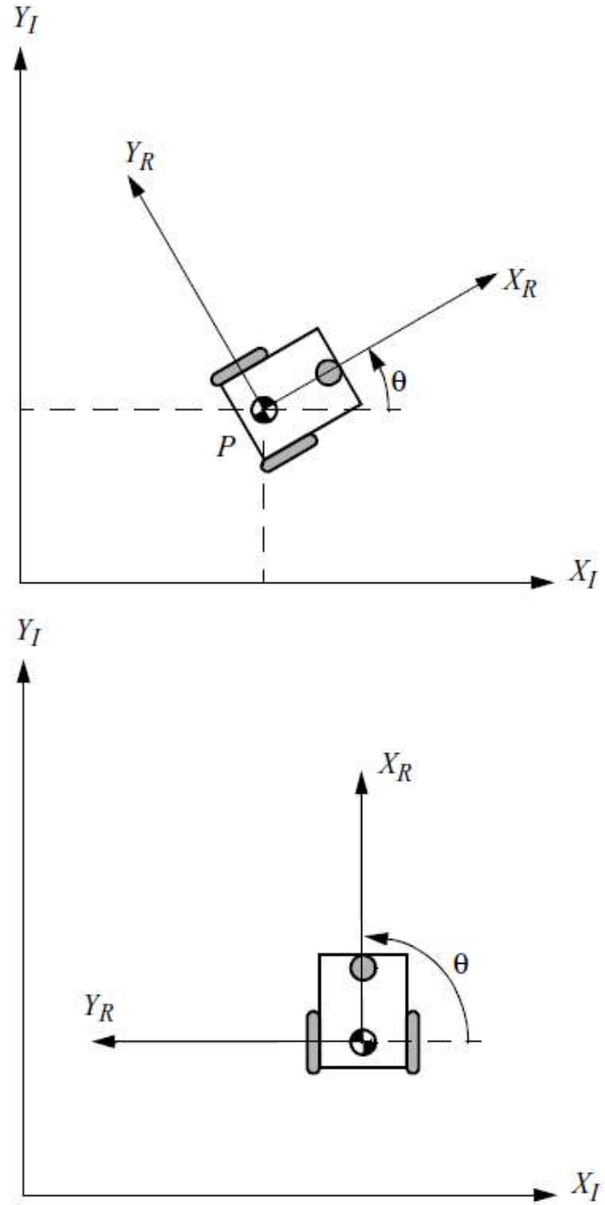
- Assemble encoders onto motors



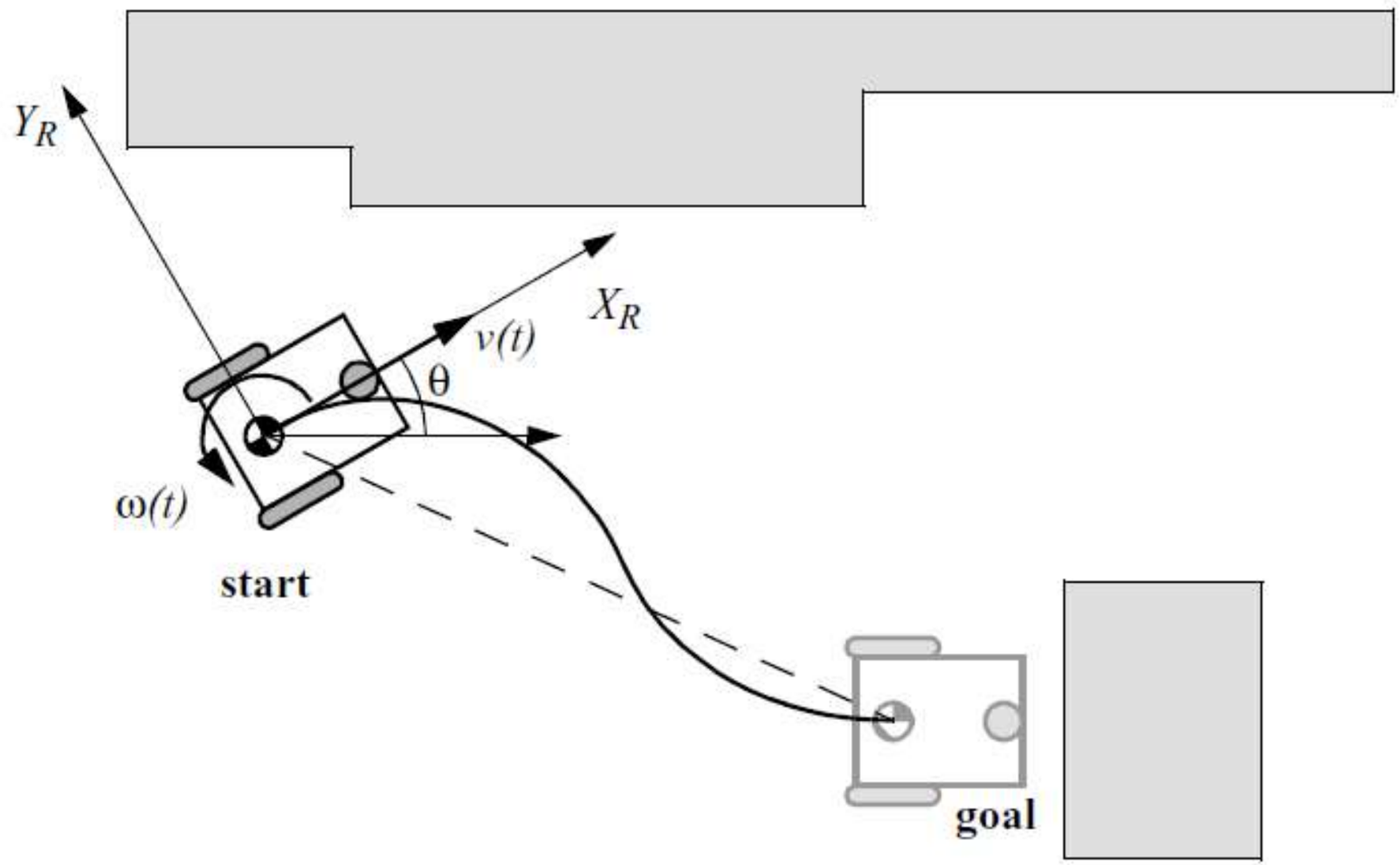


Kinematics

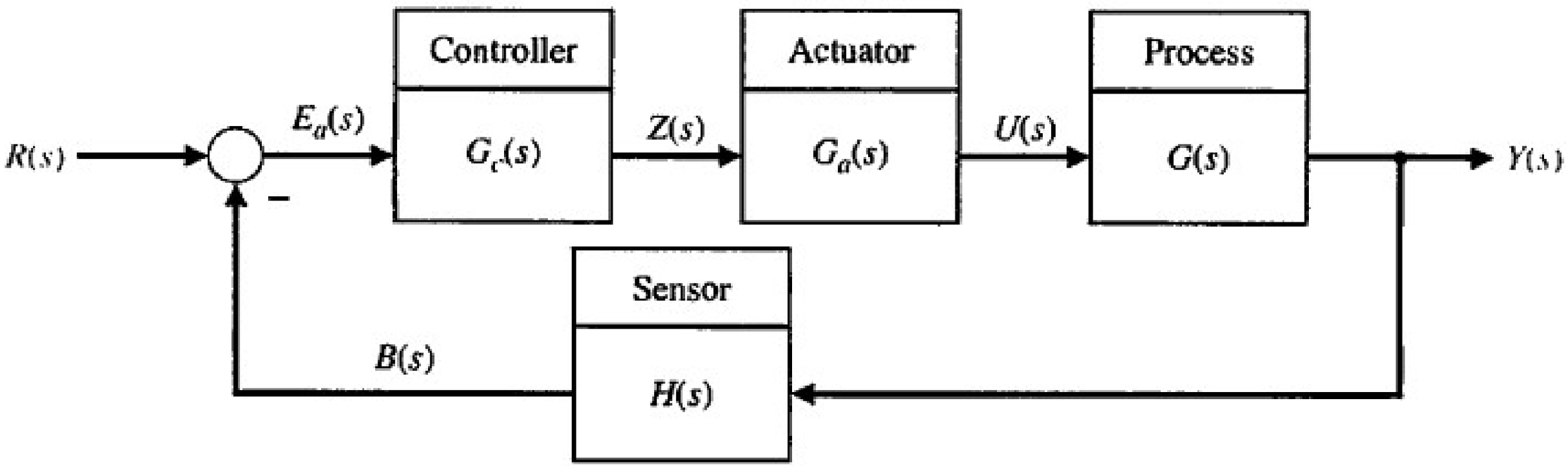
- Study of how mechanical systems behave
- Mobile robotics
  - Design for specific tasks
  - Create control software to drive hardware
- Global reference frame & robot local reference frame



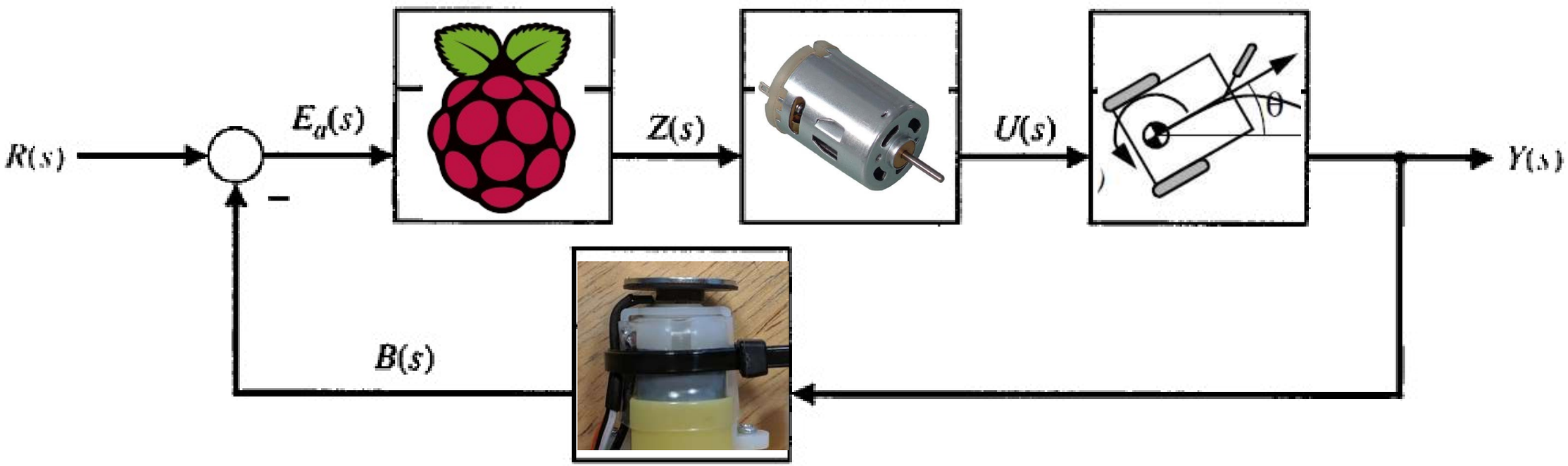
Kinematics



Control Theory



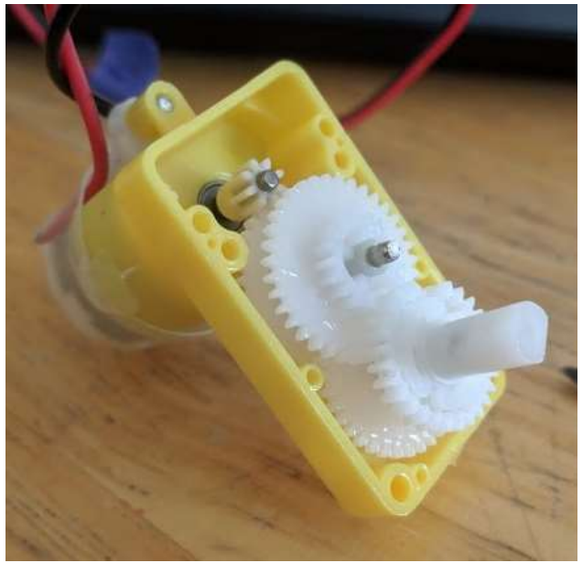
Control Theory

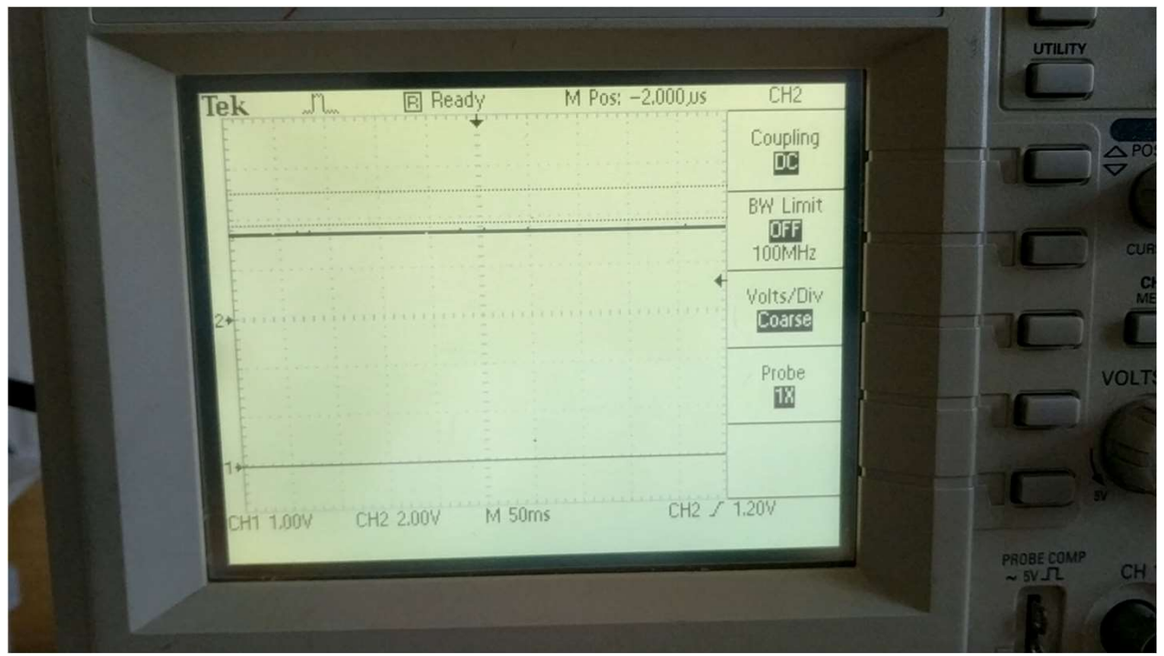


Encoders

- Electro-mechanical device
- Converts motion into sequence of digital pulses
- Pulse train converted to position measurements
  - Robot localization
- Typically magnetic or optical

Proprioceptive Sensor





https://www.pololu.com/product/2590

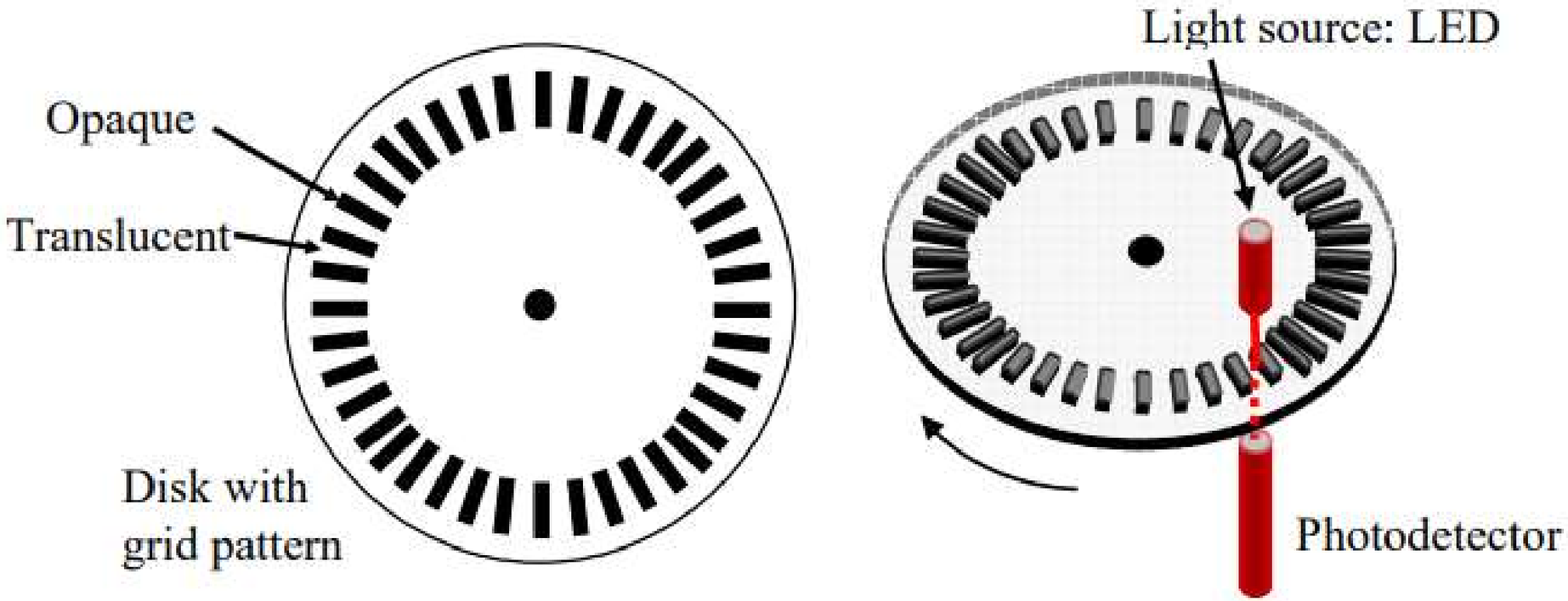


Figure 2.5.1 Basic construction of optical shaft encoder

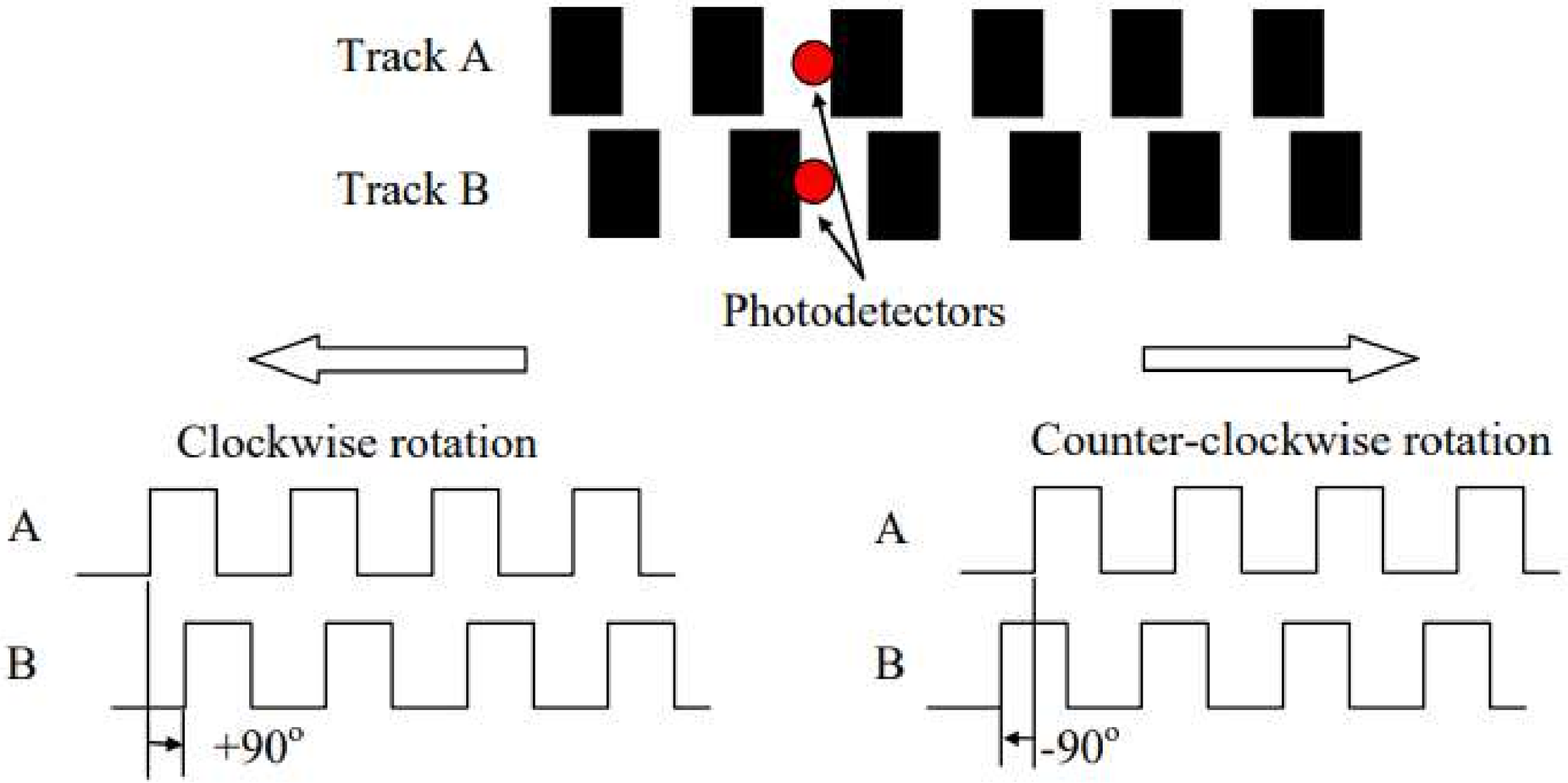
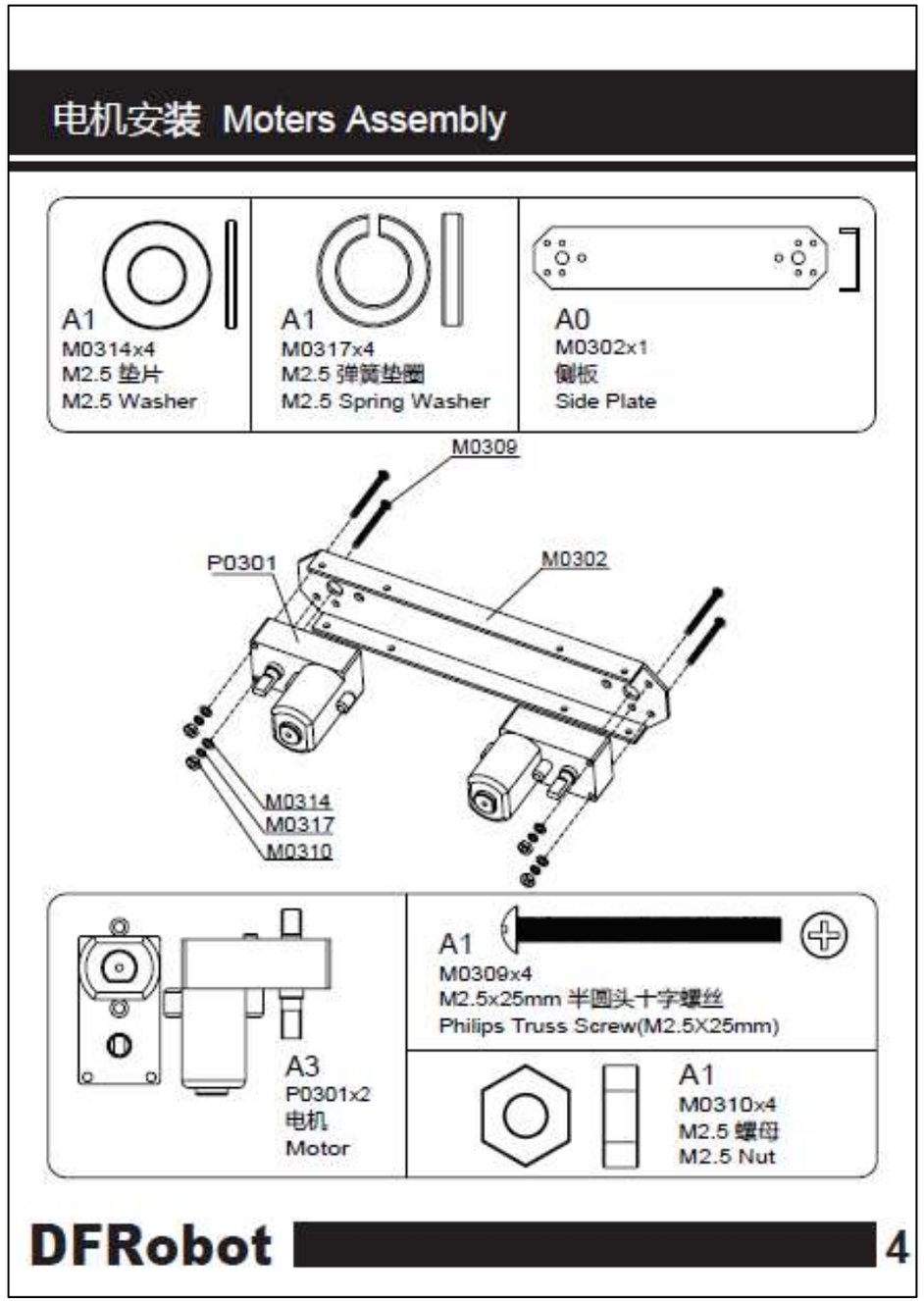


Figure 2.5.2 Double track encode for detection of the direction of rotation

Assembly

- Assemble motors to side plates

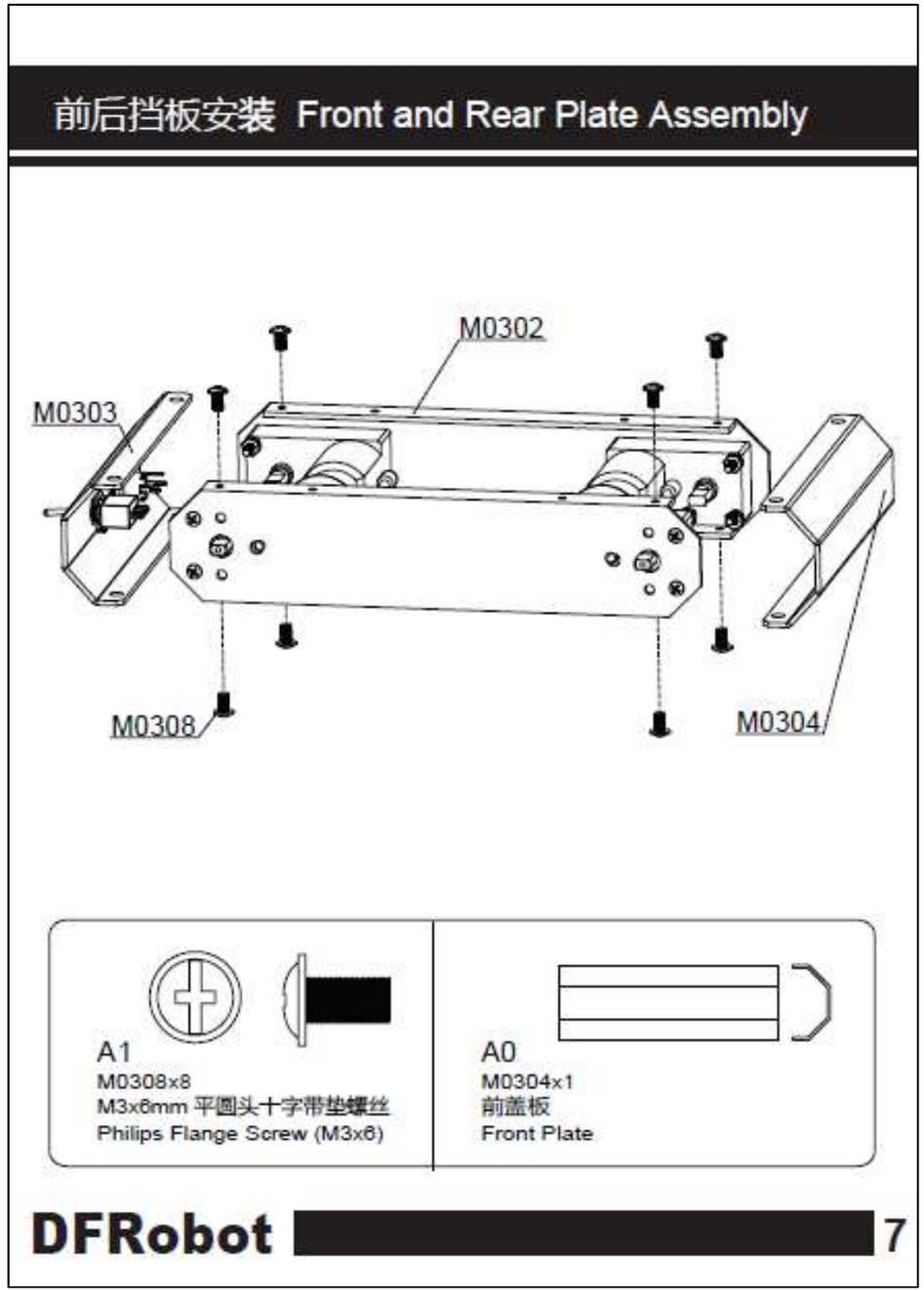


Assembly

- Assemble motors to side plates

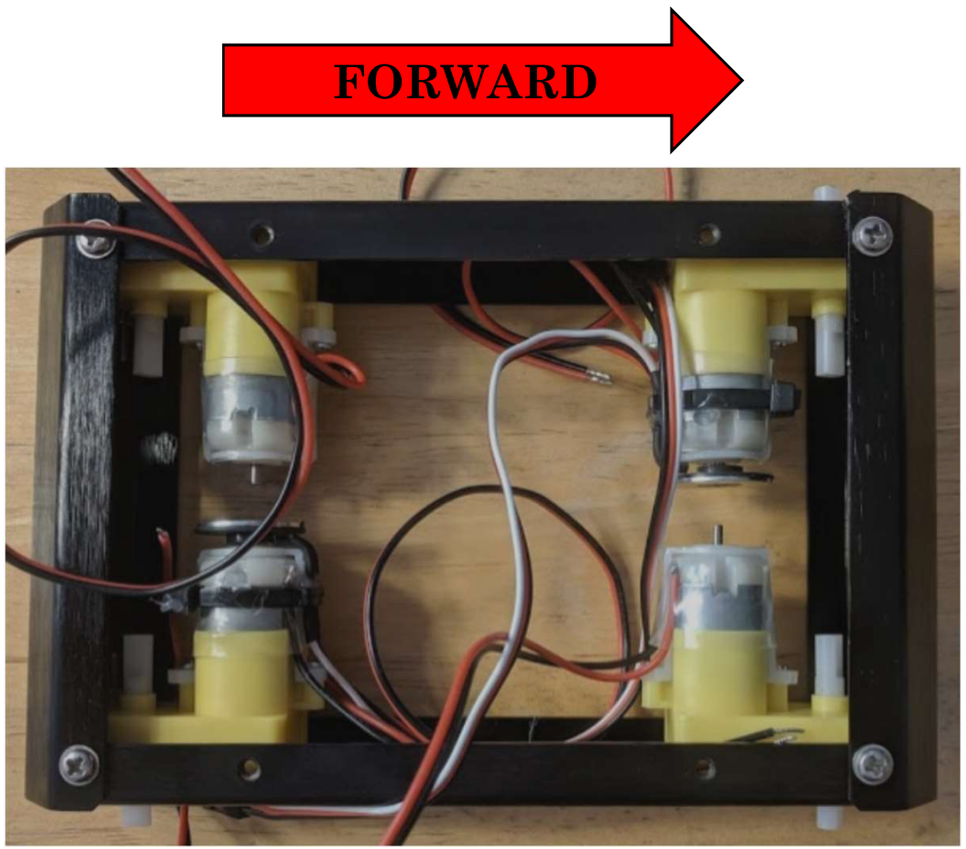
Assembly

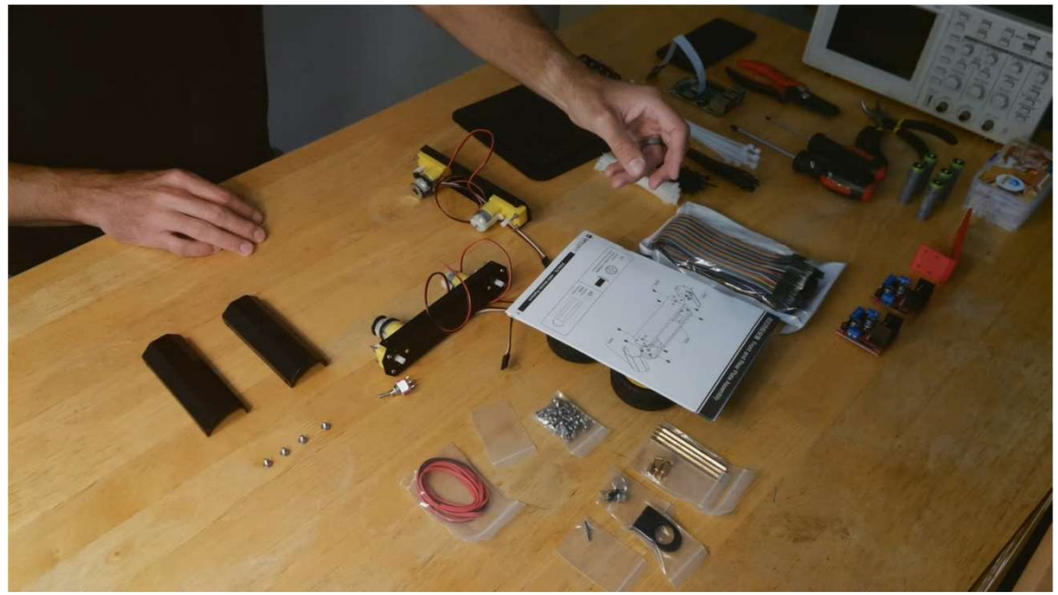
- Assemble front & rear plates



Assembly

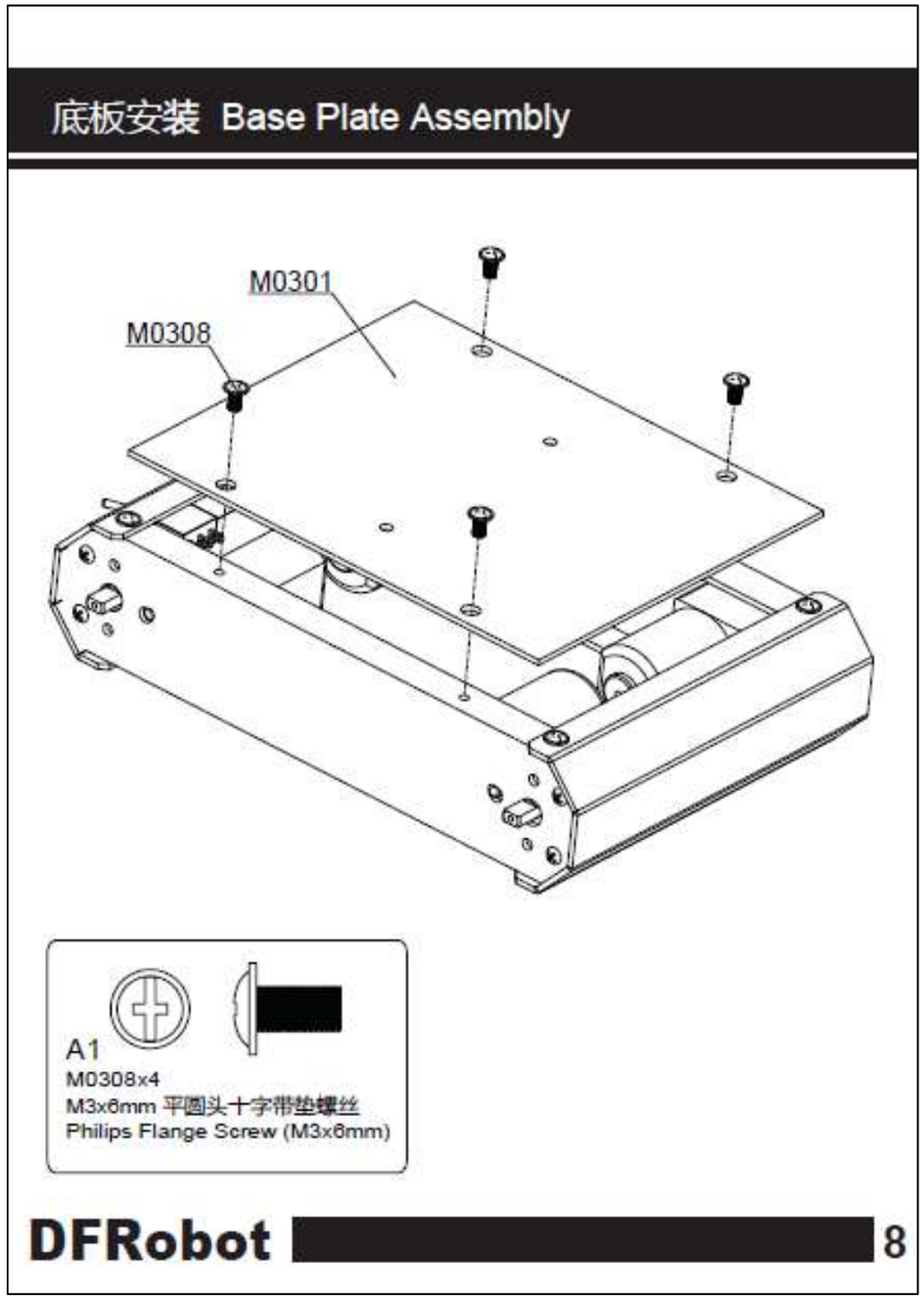
- Assemble front & rear plates





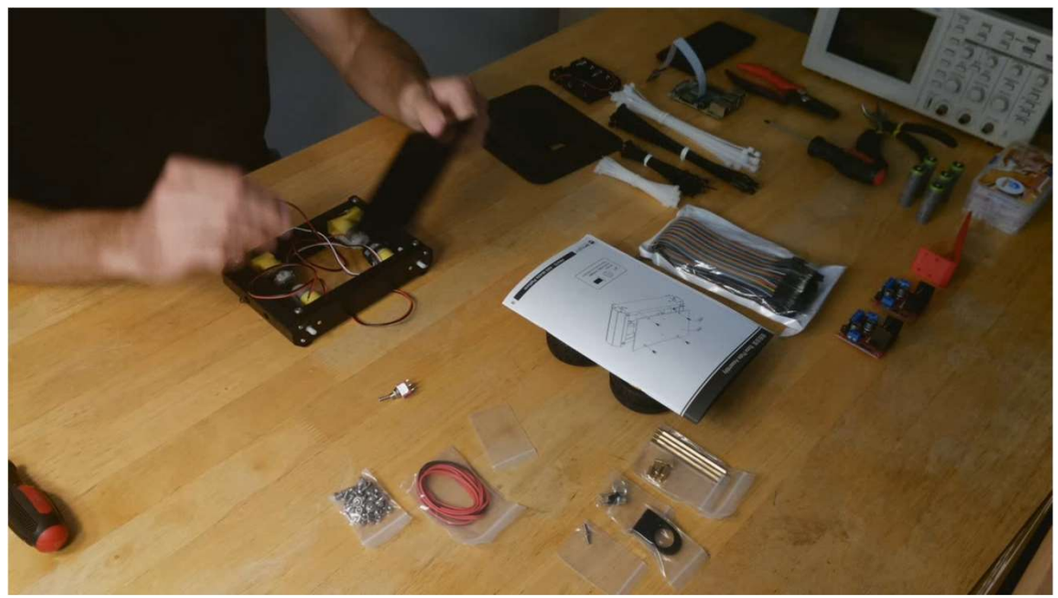
Assembly

- Assemble base plate to frame



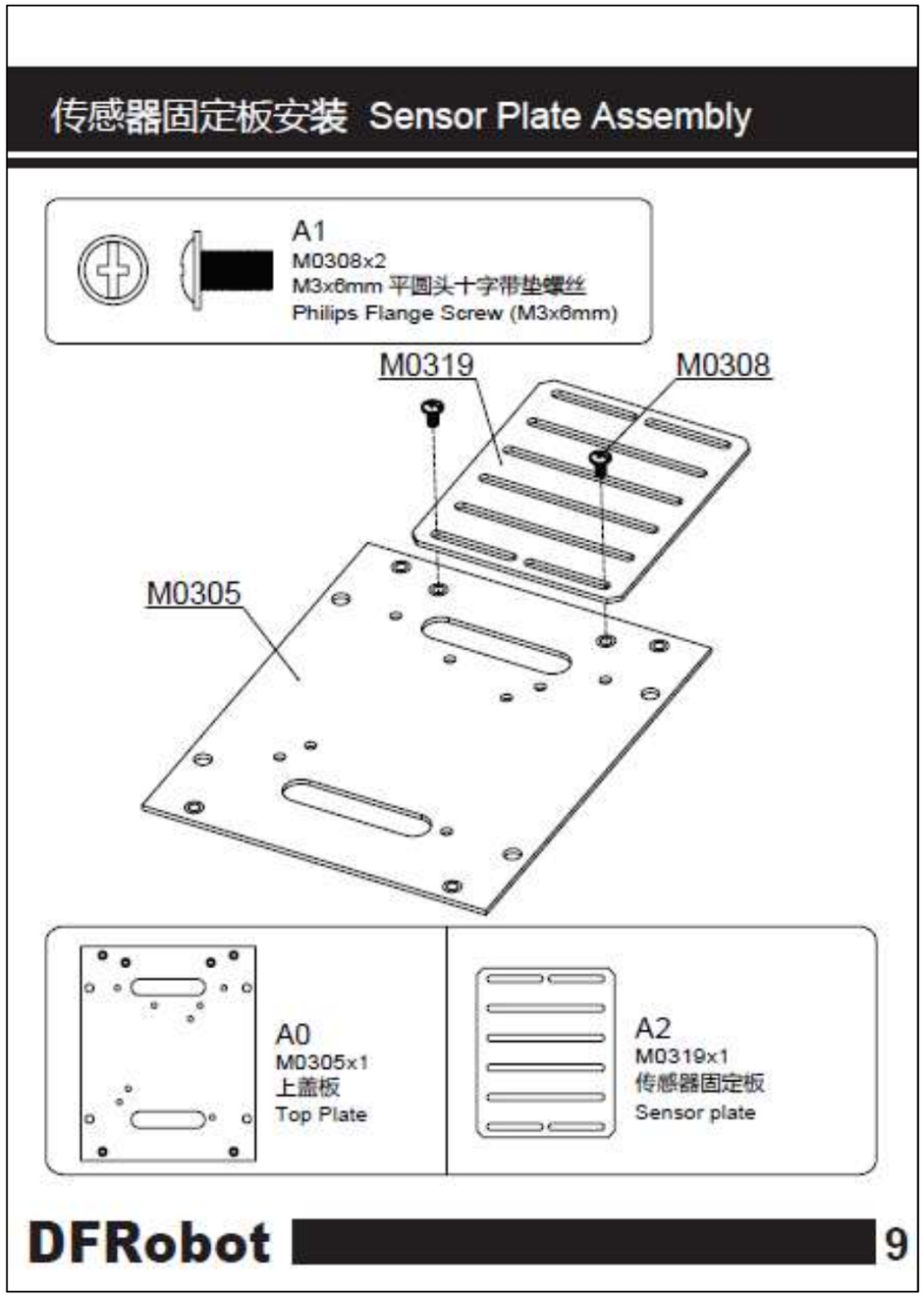
Assembly

- Assemble base plate to frame



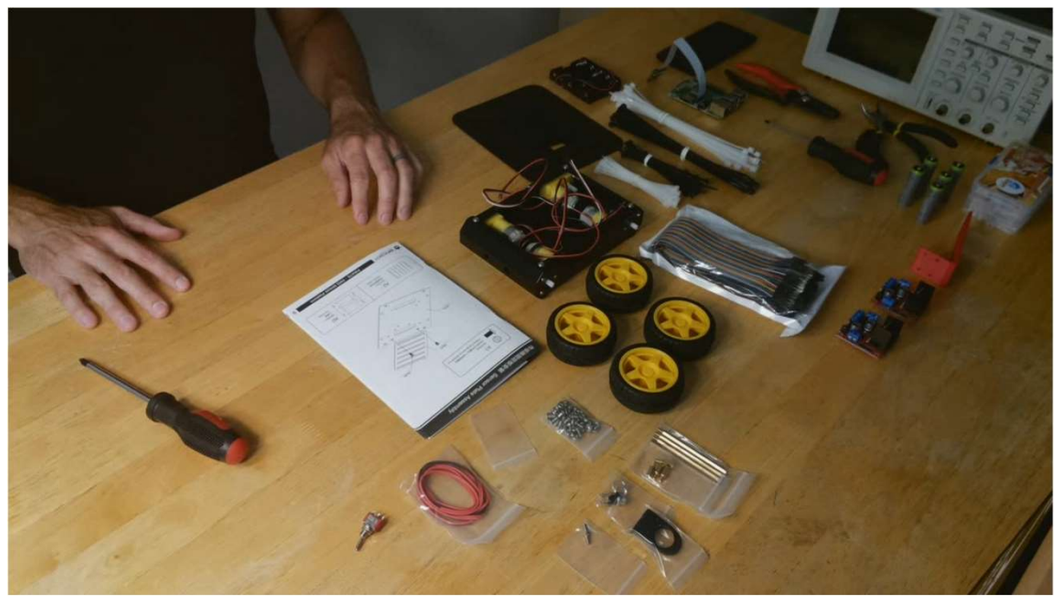
Assembly

- Assemble sensor plate to top plate



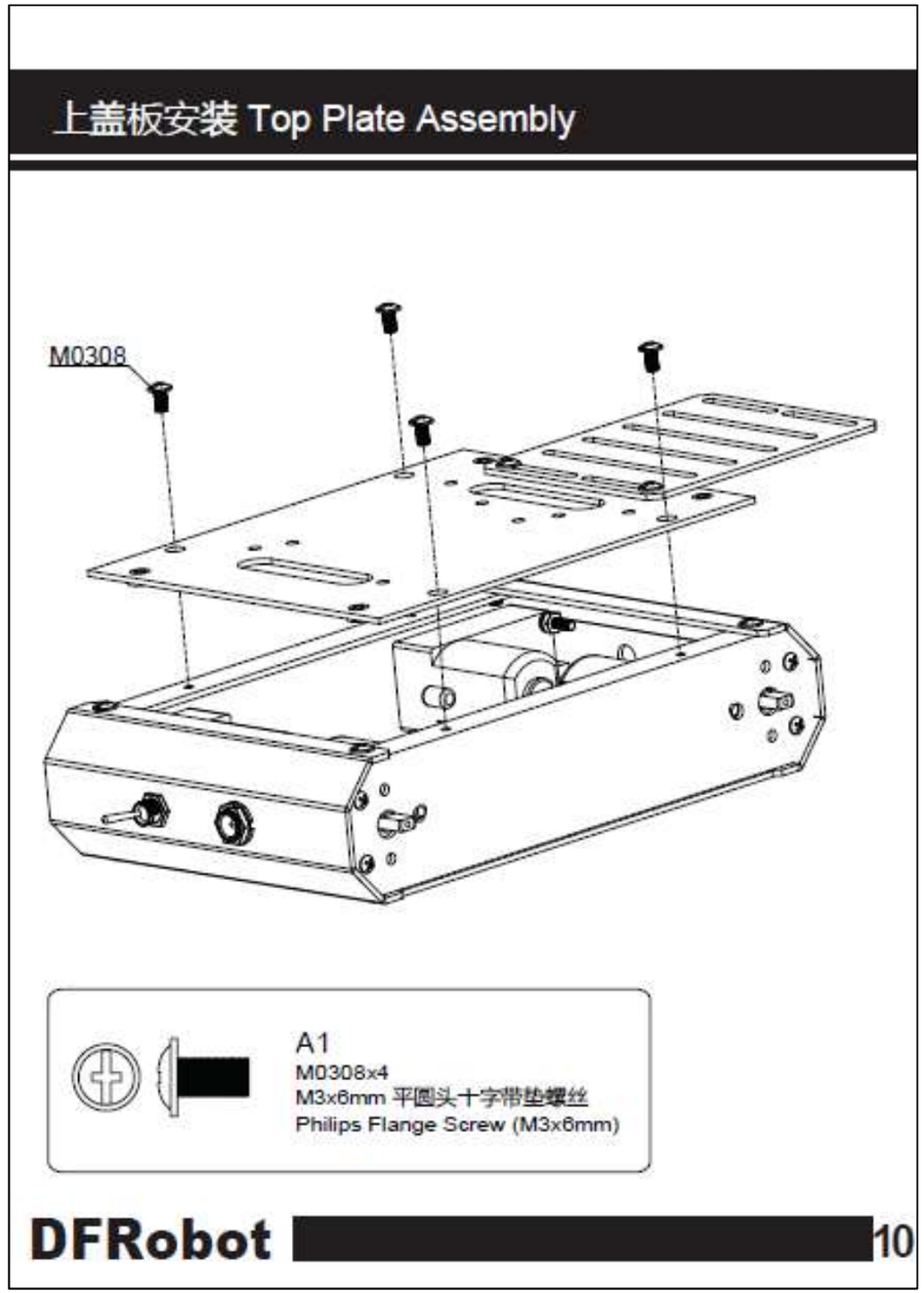
Assembly

- Assemble sensor plate to top plate



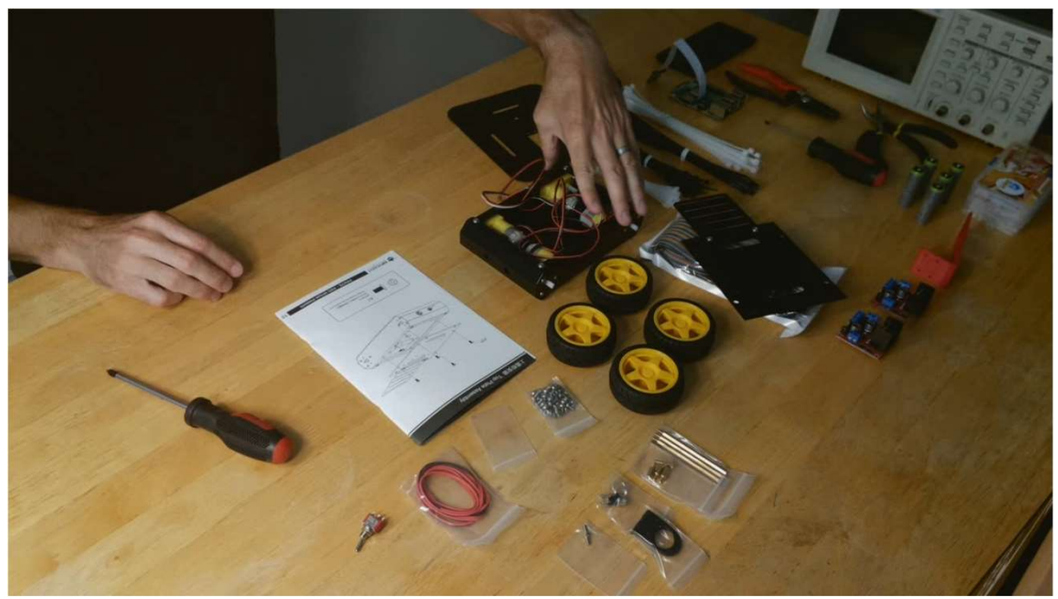
Assembly

- Assemble top plate to frame



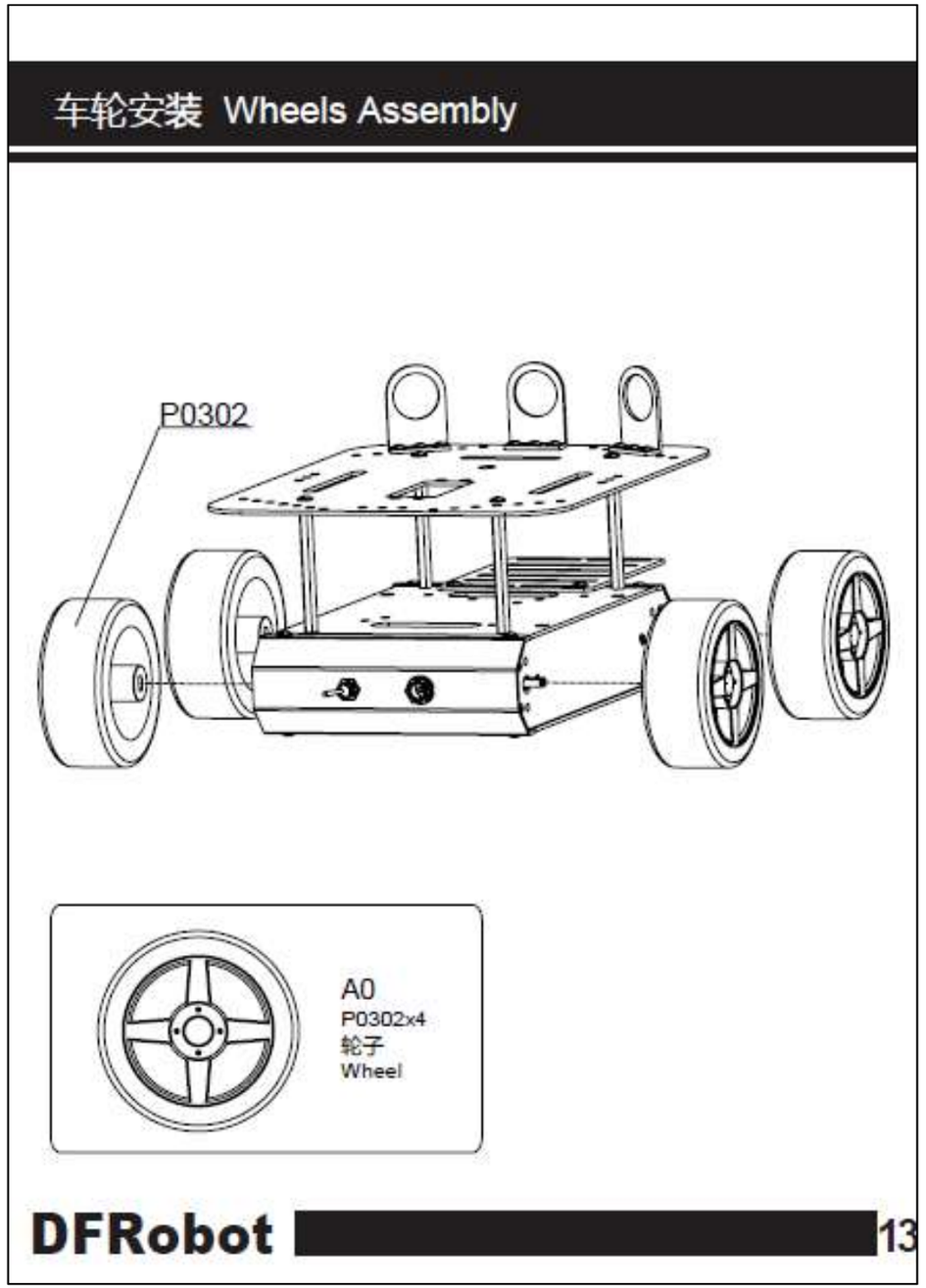
Assembly

- Assemble top plate to frame



Assembly

- Mount wheels onto motors

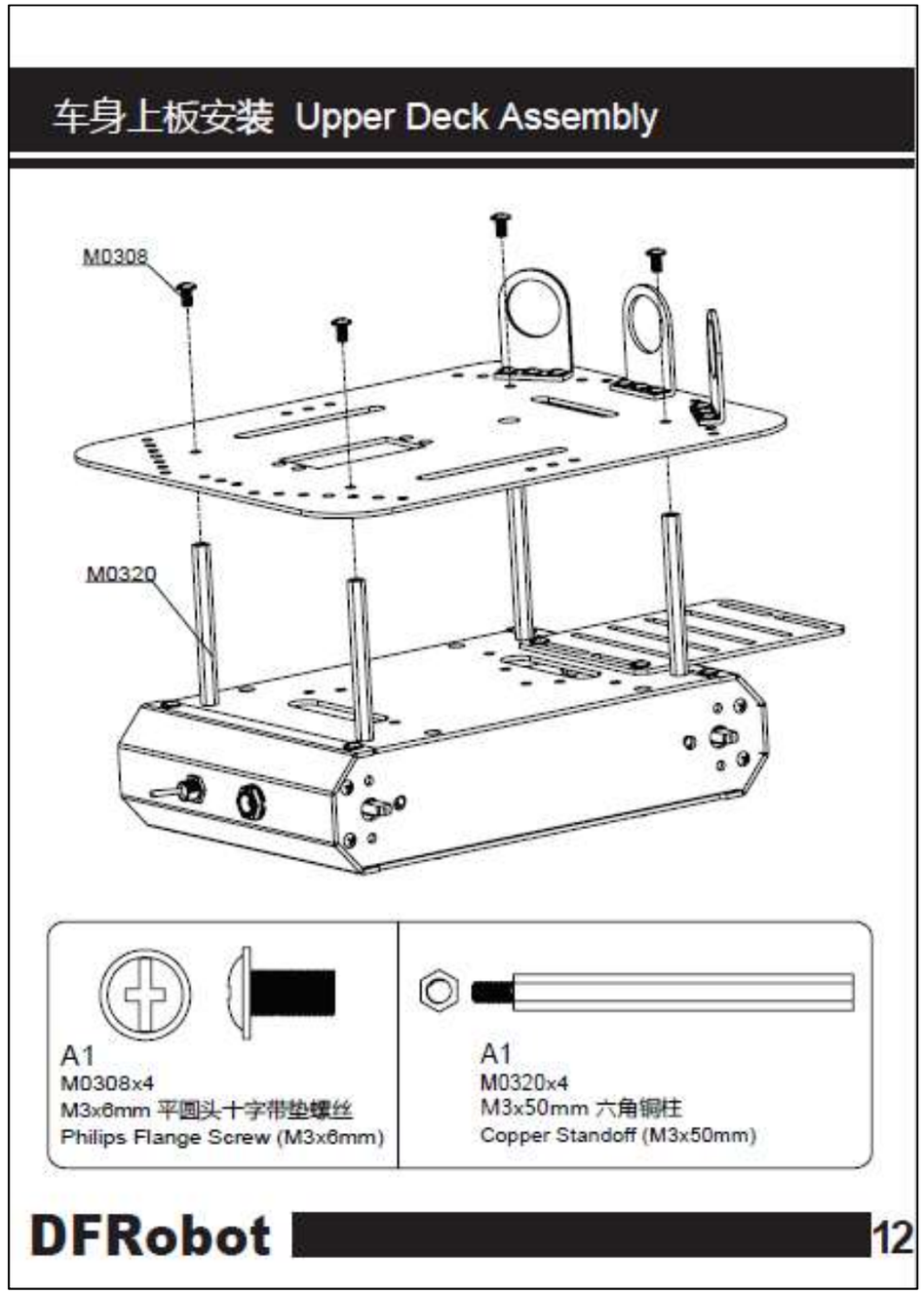


Assembly

- Mount wheels onto motors

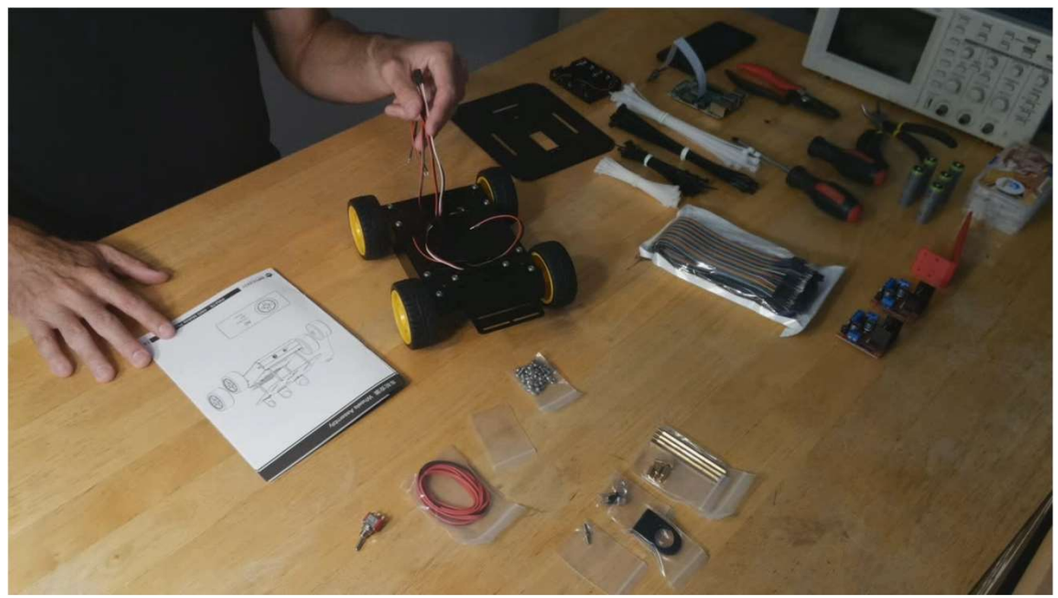
Assembly

- Assemble standoffs for upper deck



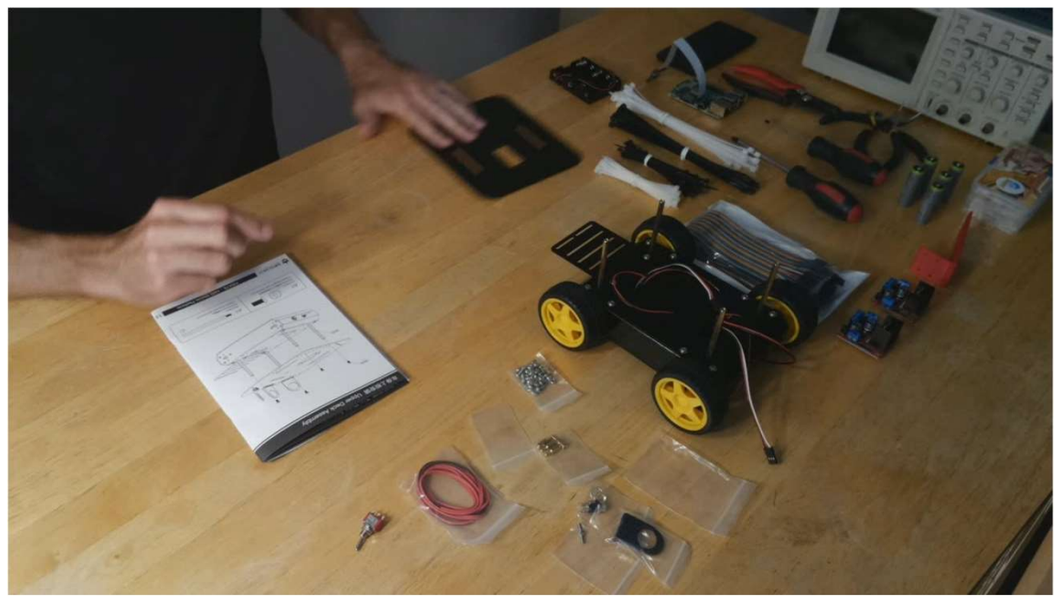
Assembly

- Assemble standoffs for upper deck



Assembly

- Mount upper deck on standoffs



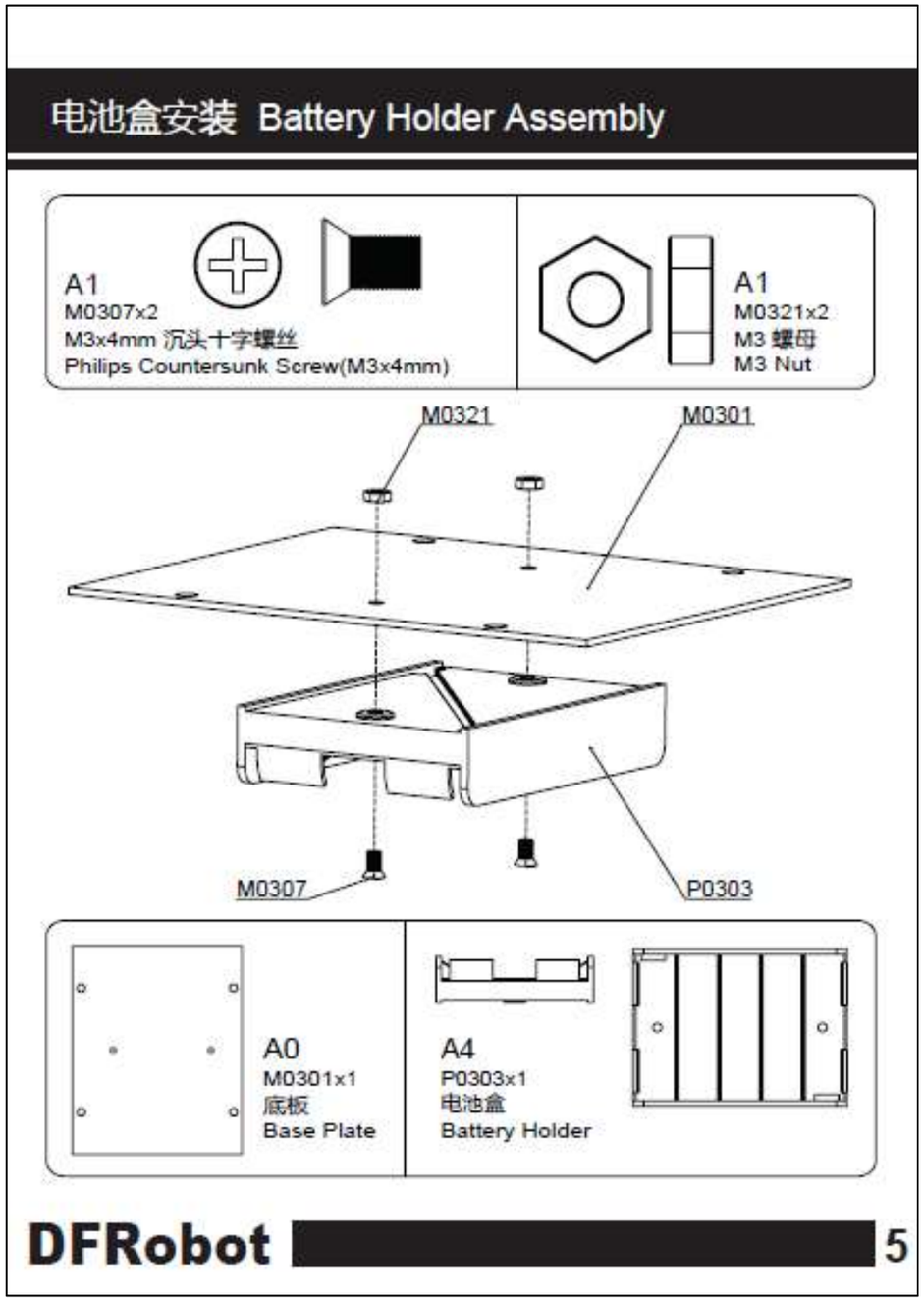
Assembly

- Solder & assemble battery pack

How to solder: *https://www.youtube.com/watch?v=oqV2xU1fee8*

Assembly

- Mount AA battery pack

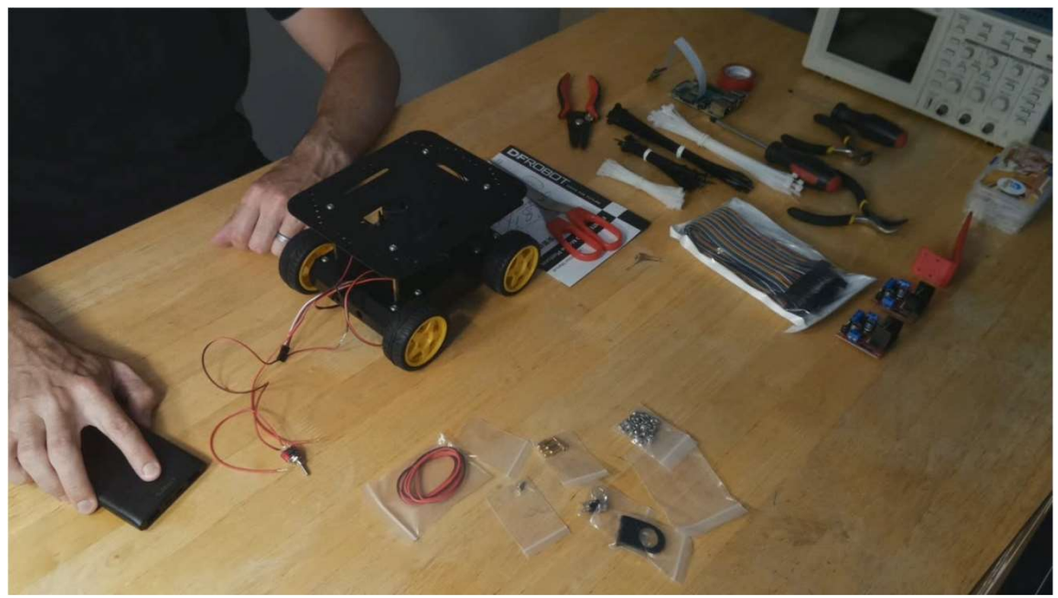


Assembly

- Mount AA battery pack

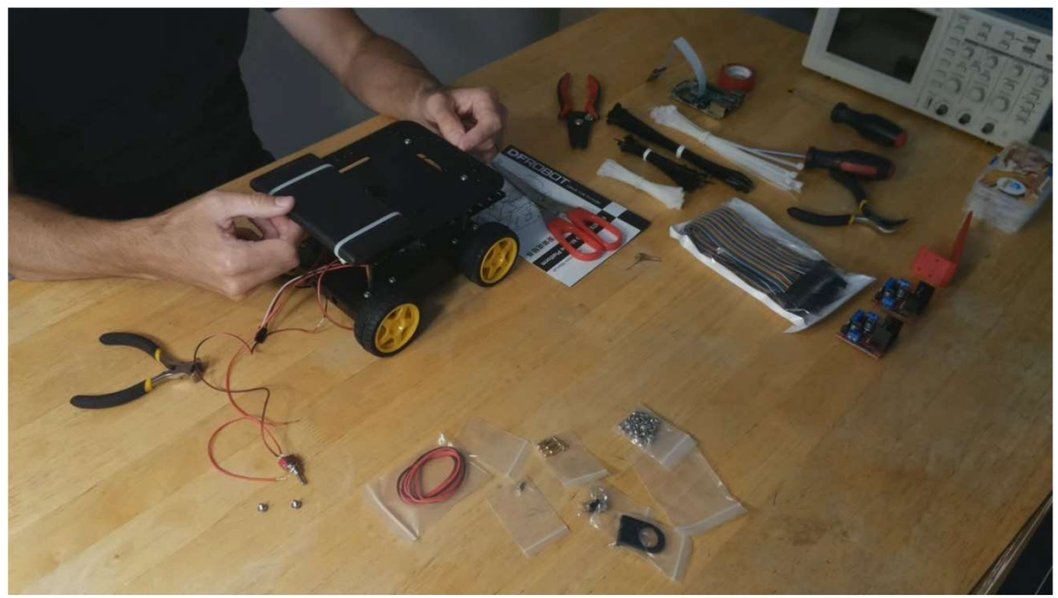
Assembly

- Mount USB battery pack



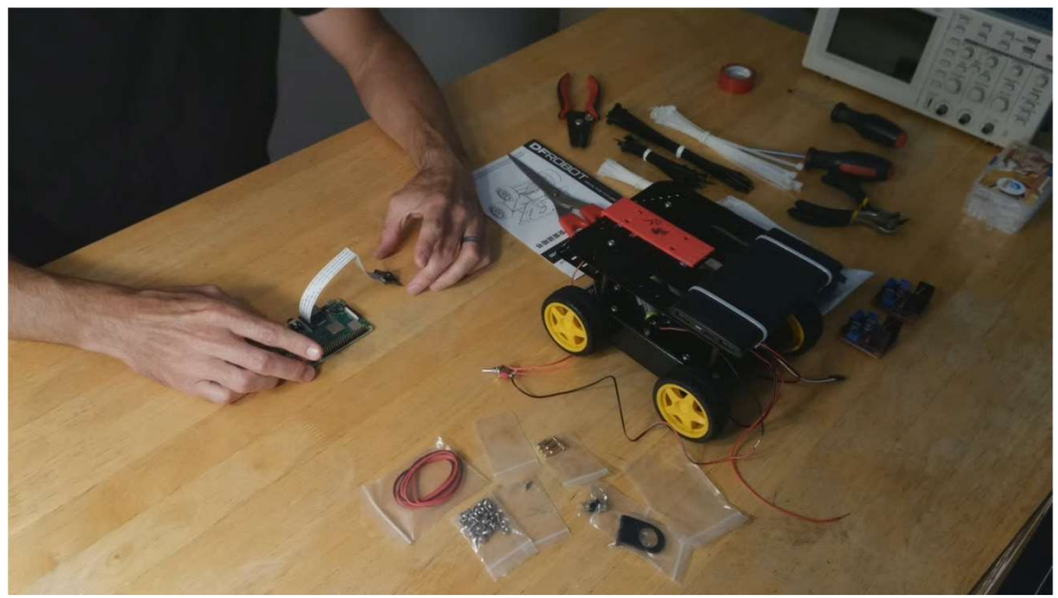
Assembly

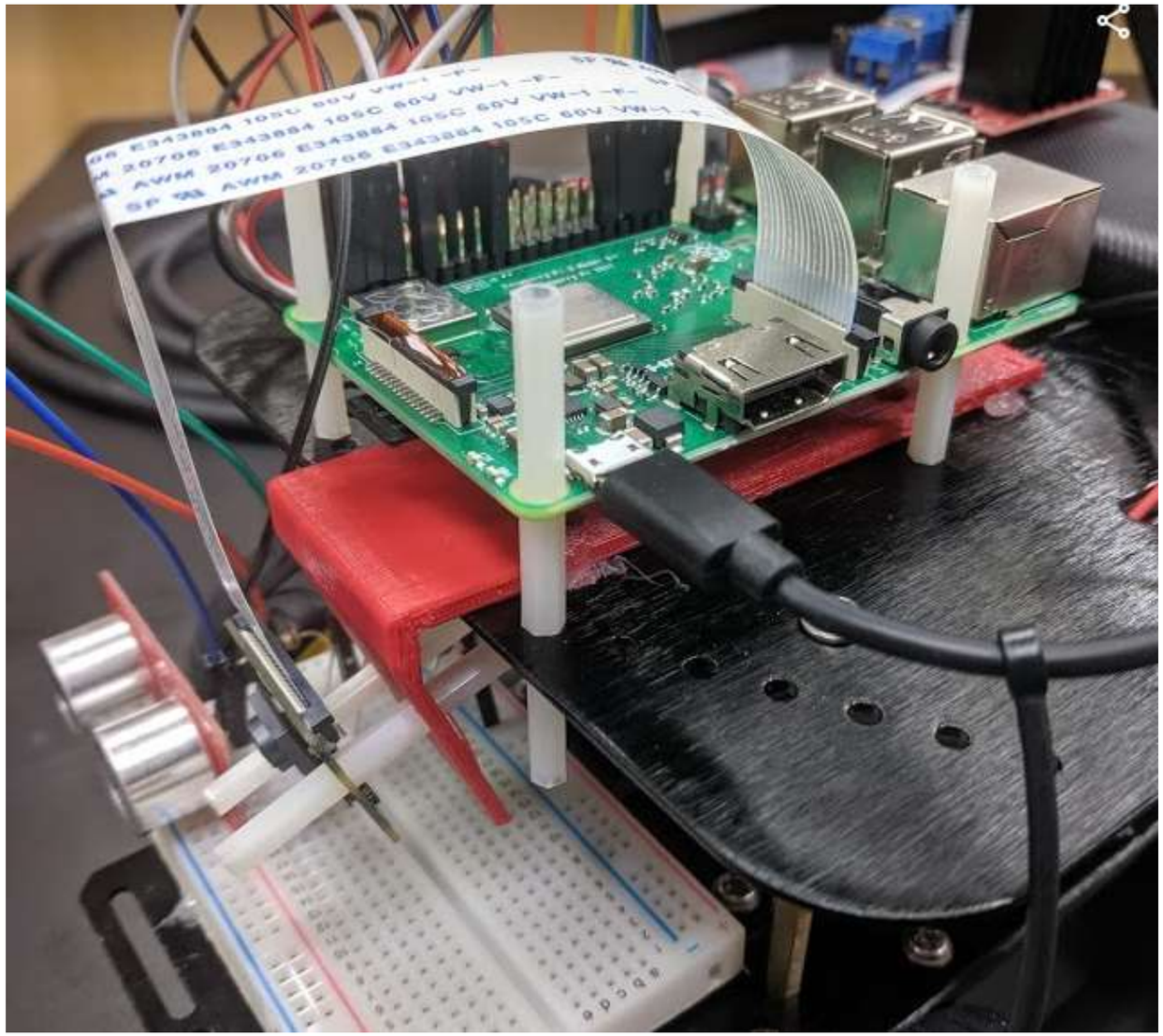
- Assemble 3D printed camera mount

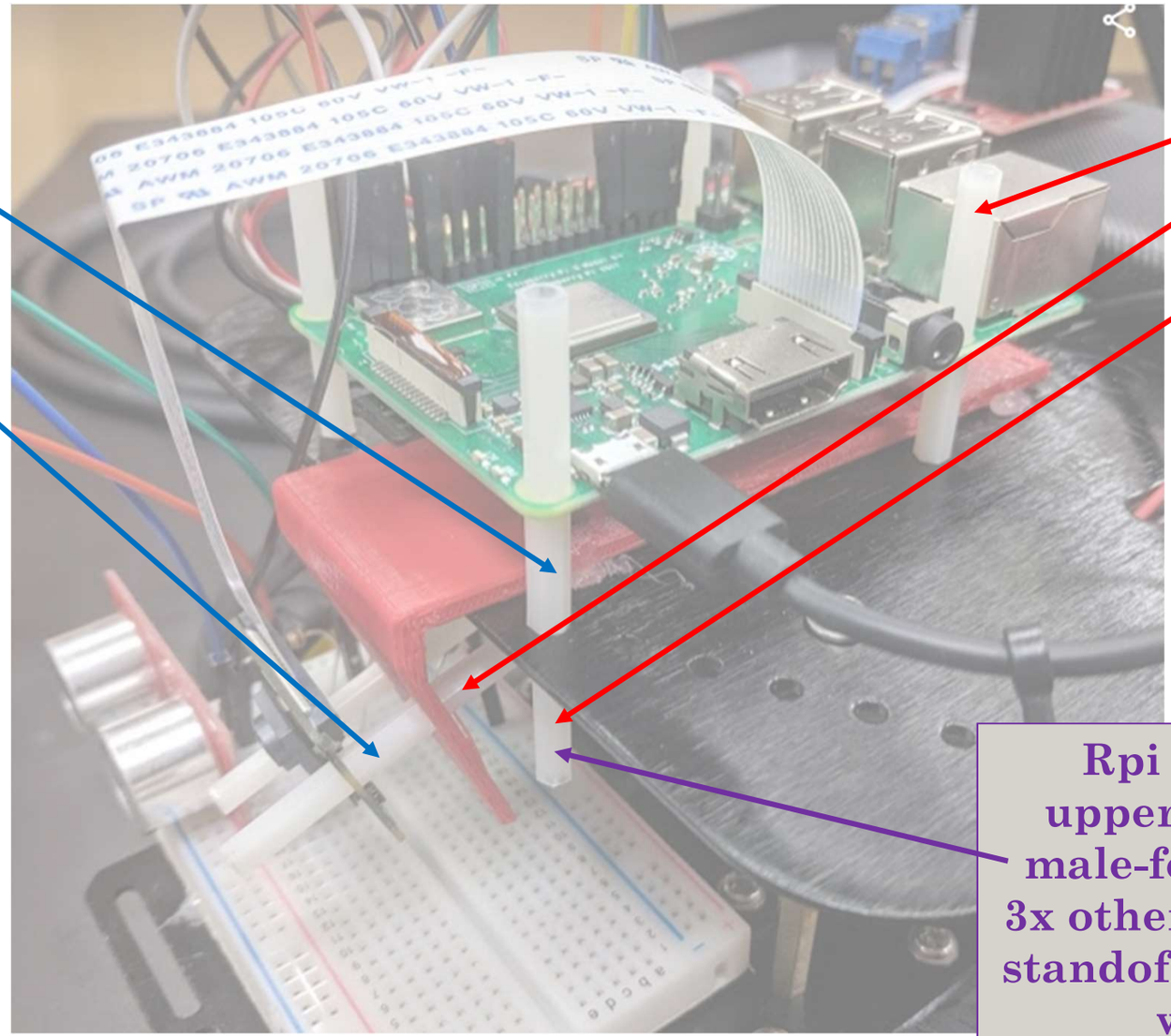


Assembly

- Mount Raspberry Pi to upper deck







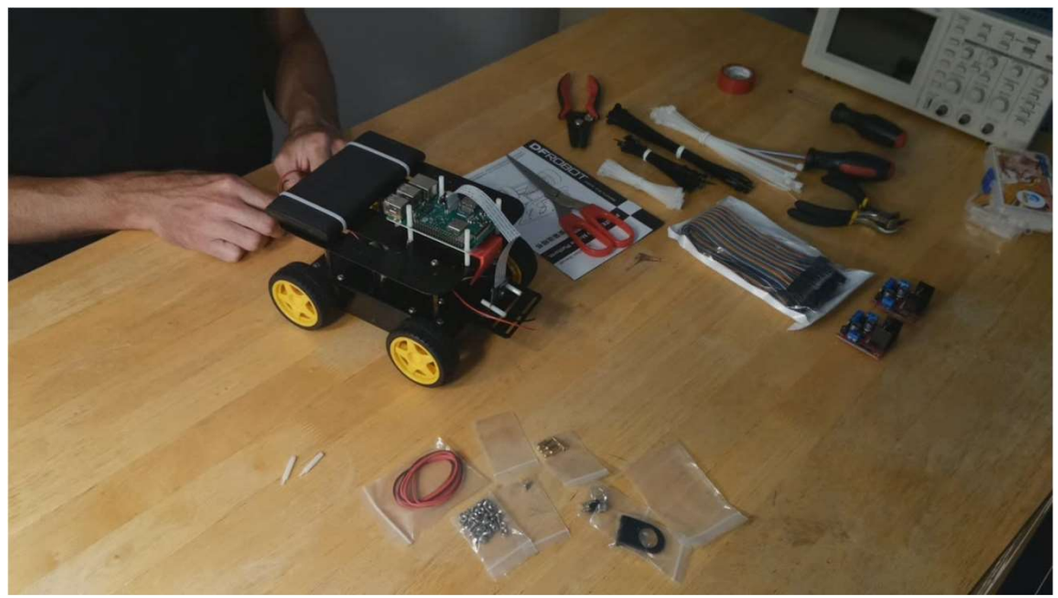
Female-female Teflon standoff

Male-female Teflon standoff

Rpi attached to upper deck with 1x male-female standoff 3x other female-female standoffs tacked down with glue

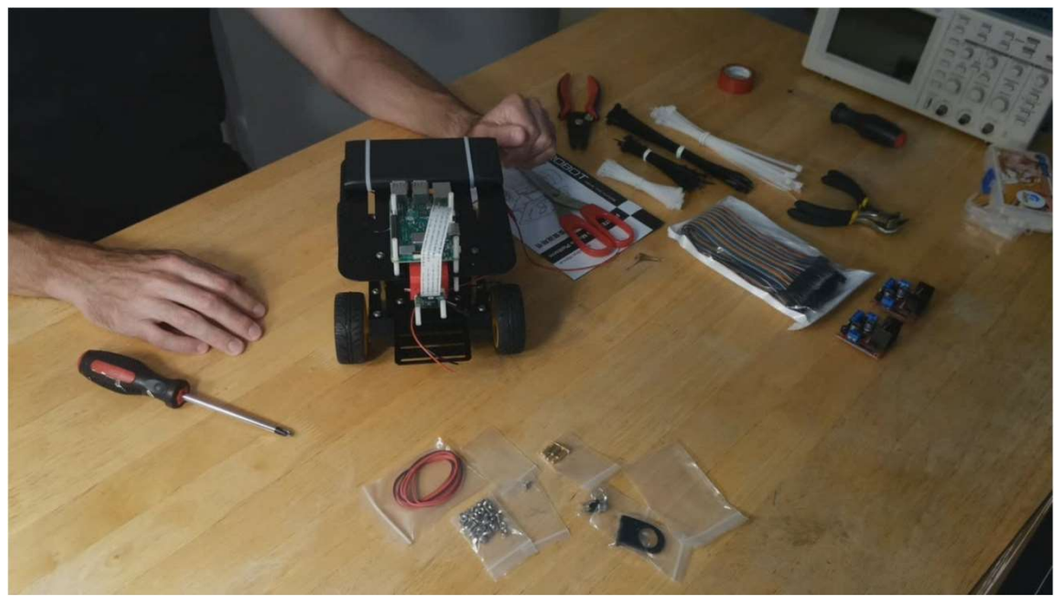
Assembly

- Mount Raspberry Pi camera to 3D printed mount

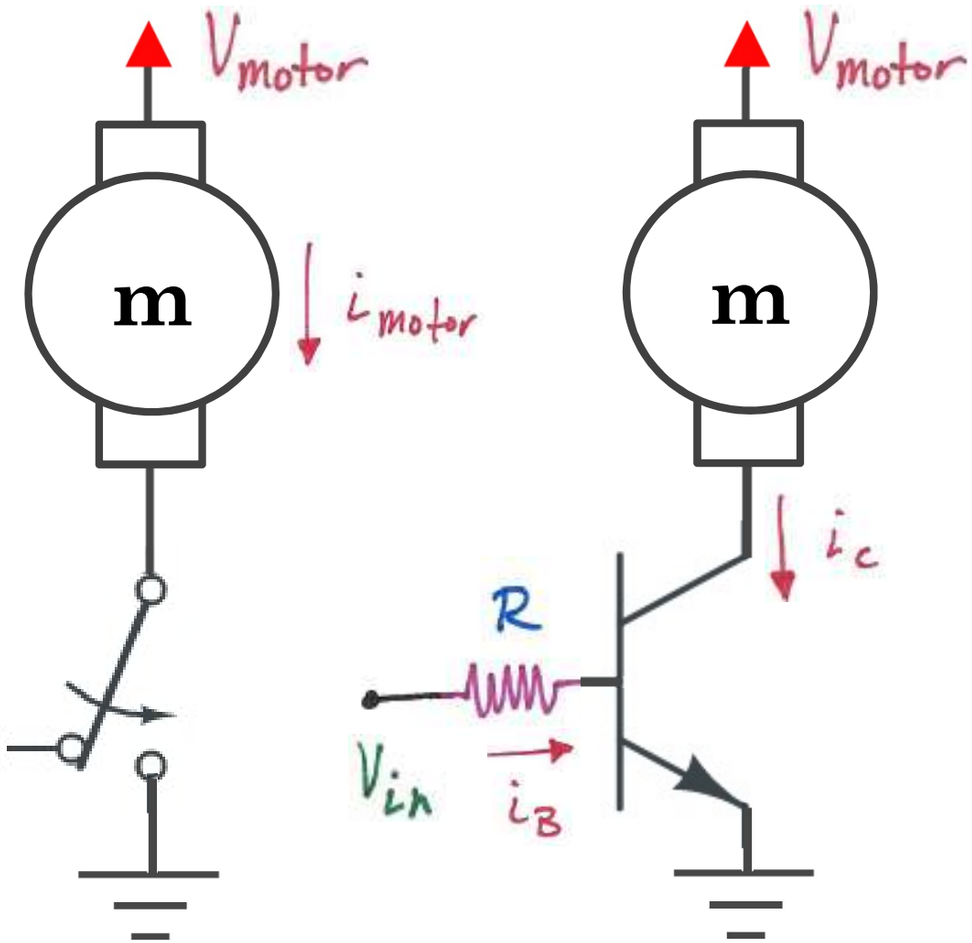


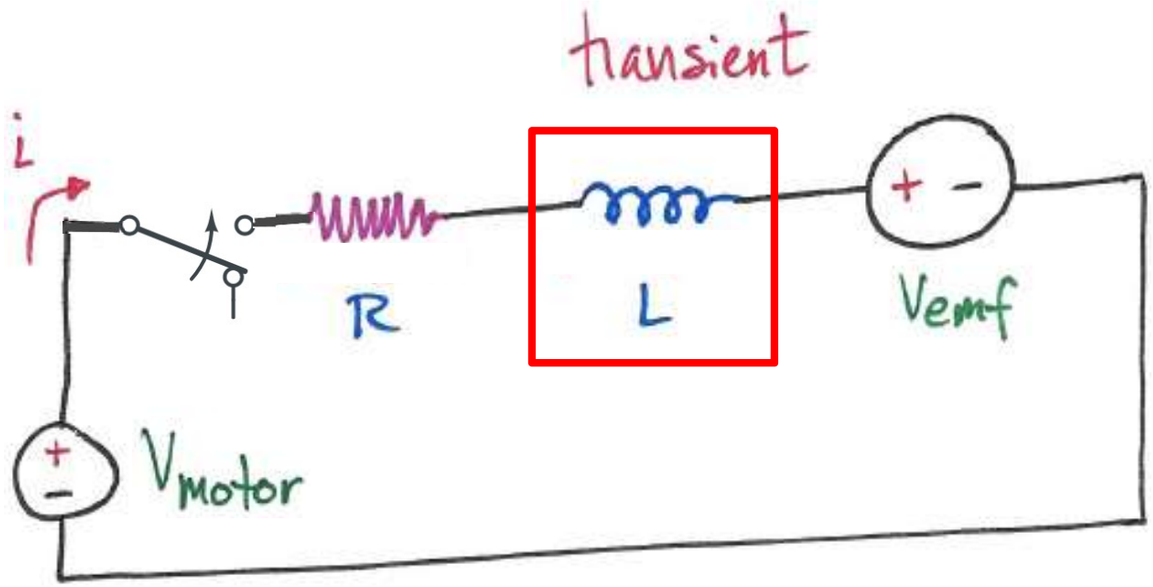
Assembly

- Mount H-bridges & ziptie power switch to upper deck

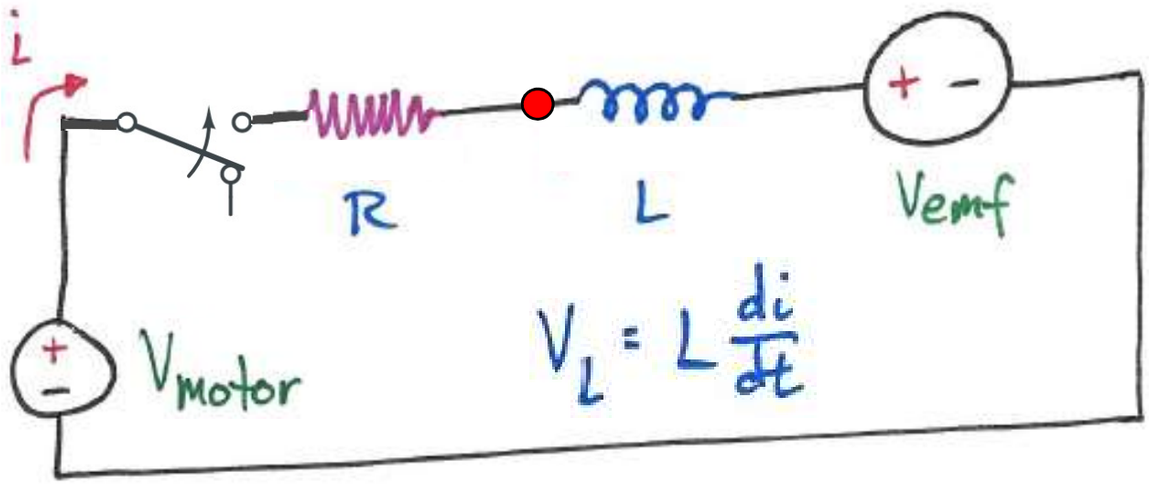


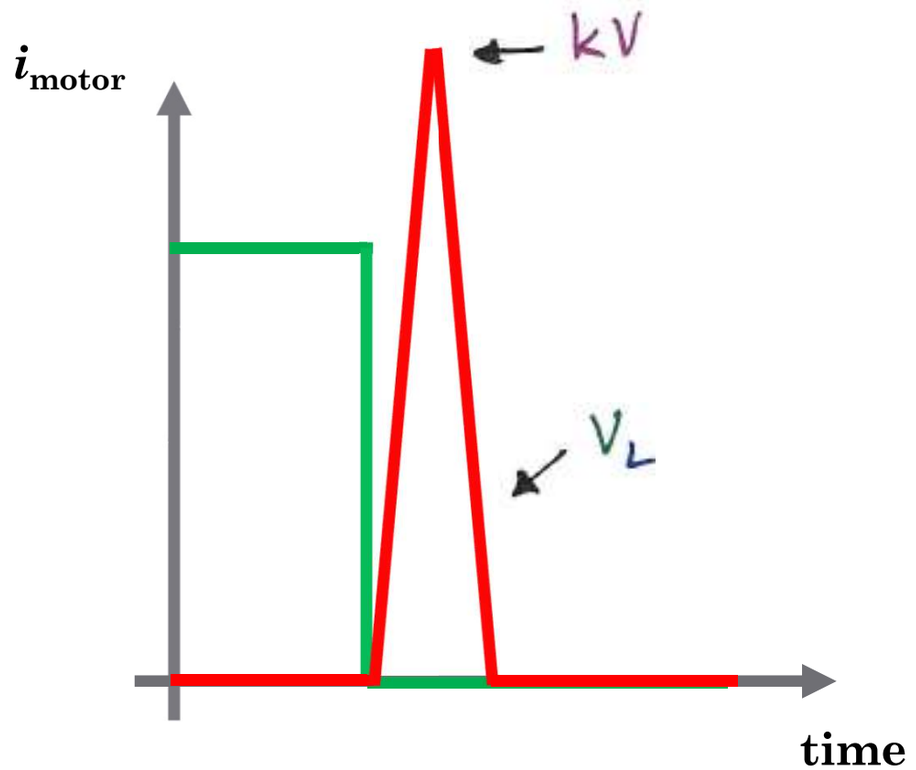
Locomotion


V_{motor}
m
i_{motor}
V_{motor}
m
i_c
R
V_{in}
i_B

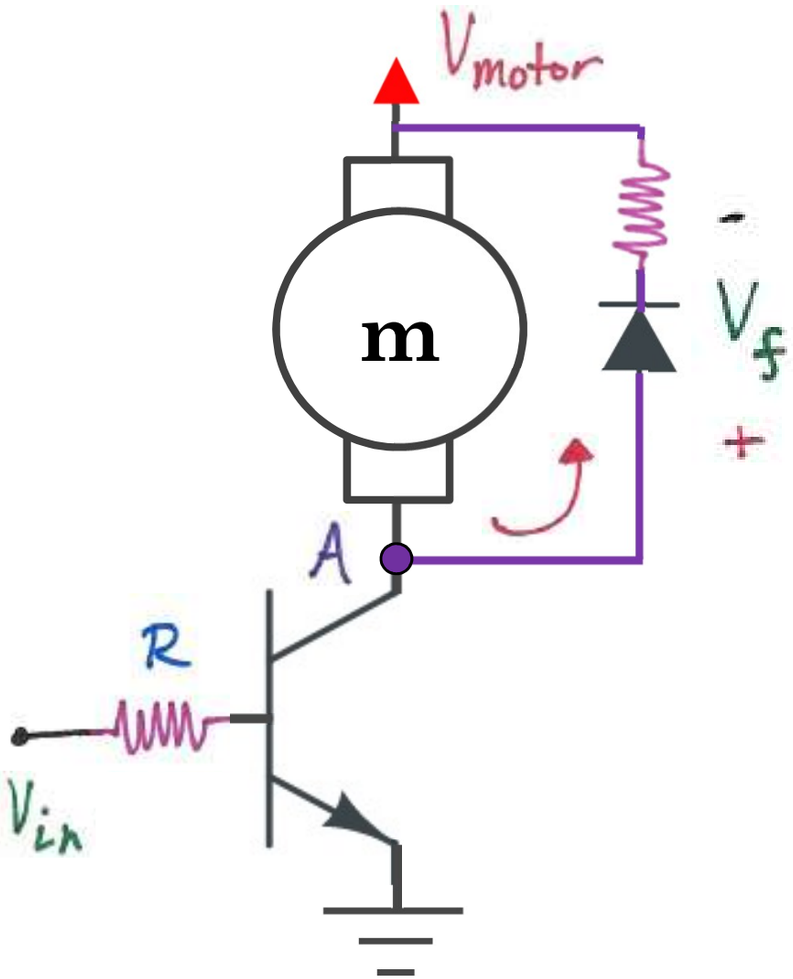

transient
i
R
L
+ −
V_{emf}
+ −
V_{motor}

What about the inductor in the motor?



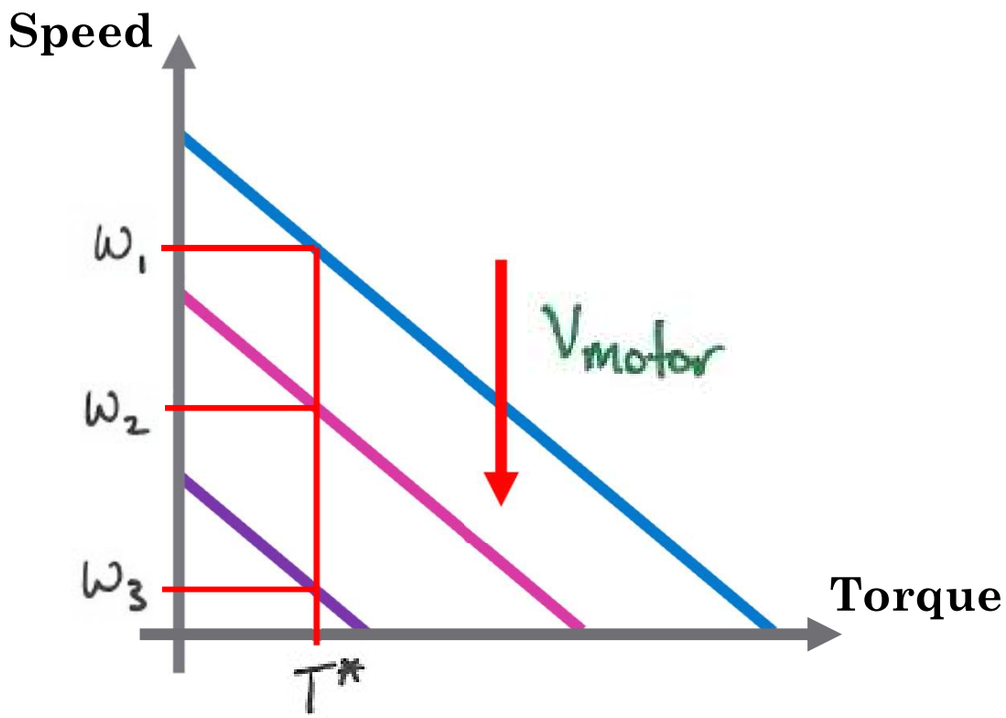


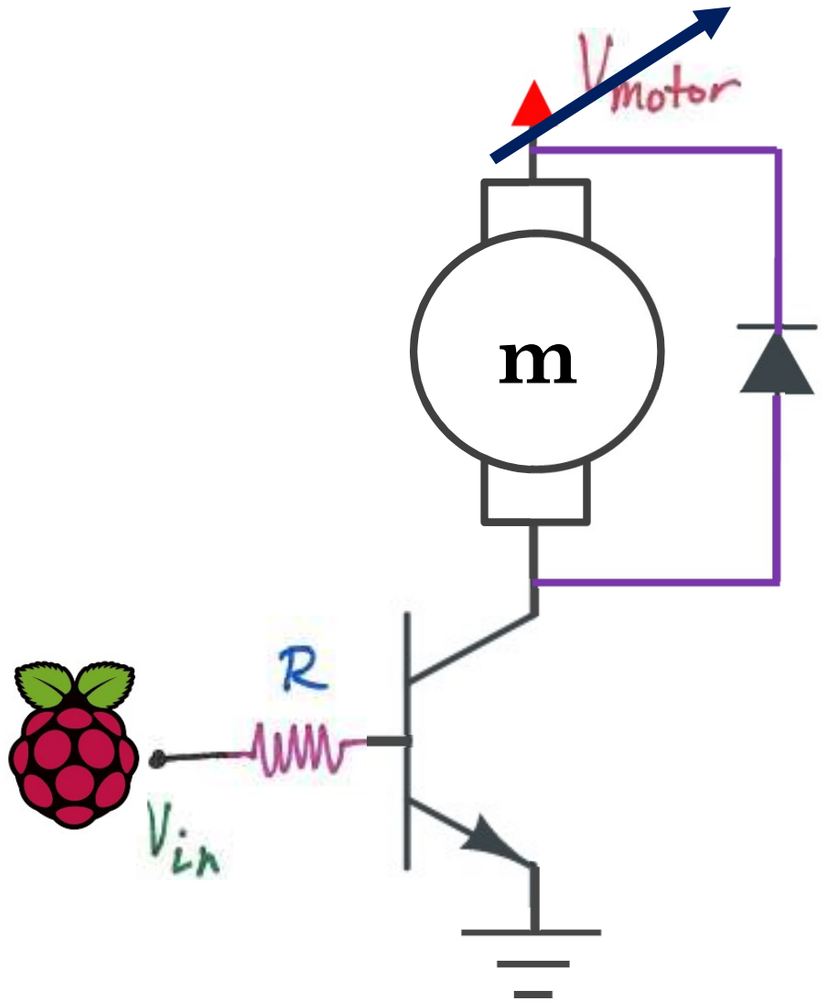
Flyback diode



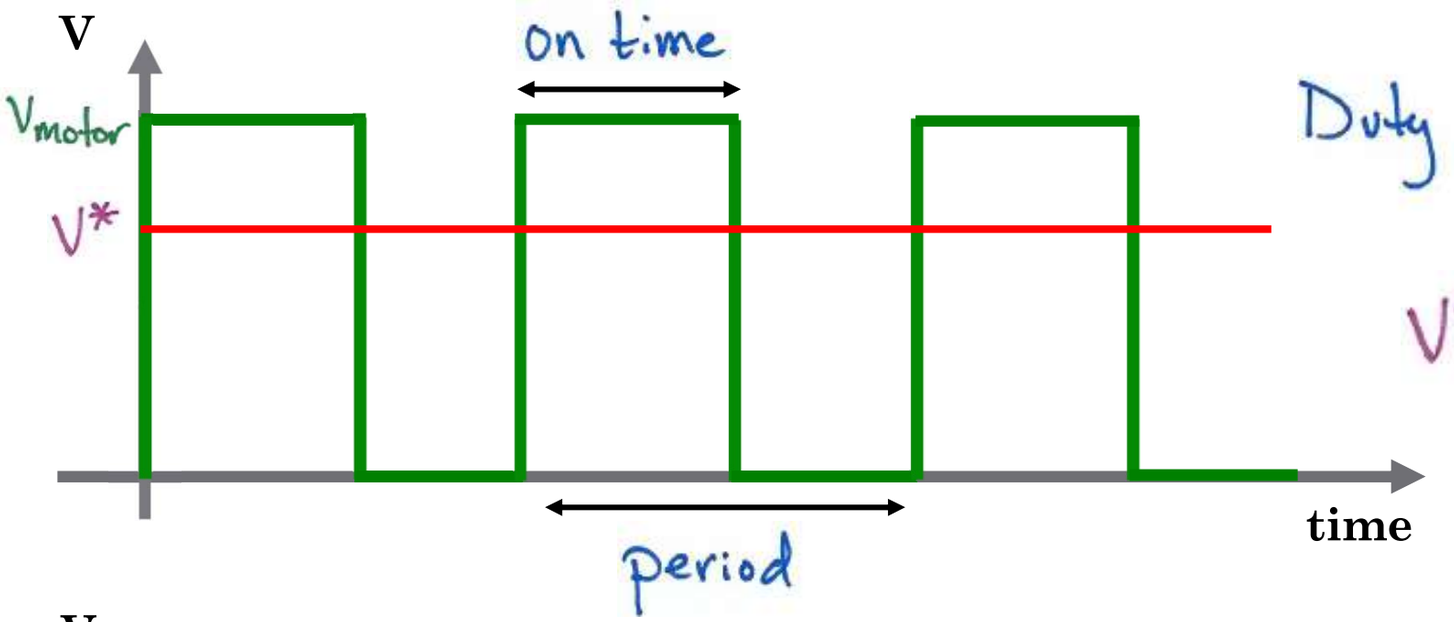
- Diode is a **one-way** current valve
  - → no affect in normal operation
  - → diode allows current to flow when voltage @ A is $V_{motor} + V_f$

Controlling speed via variable voltage



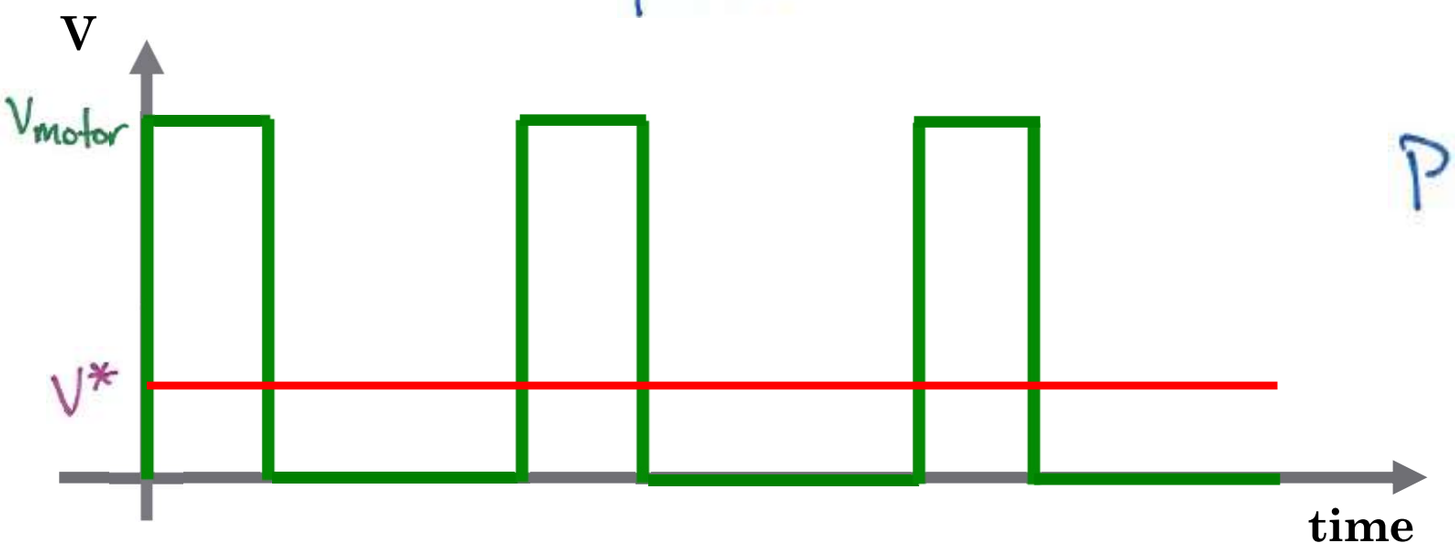


Variable voltage via PWM



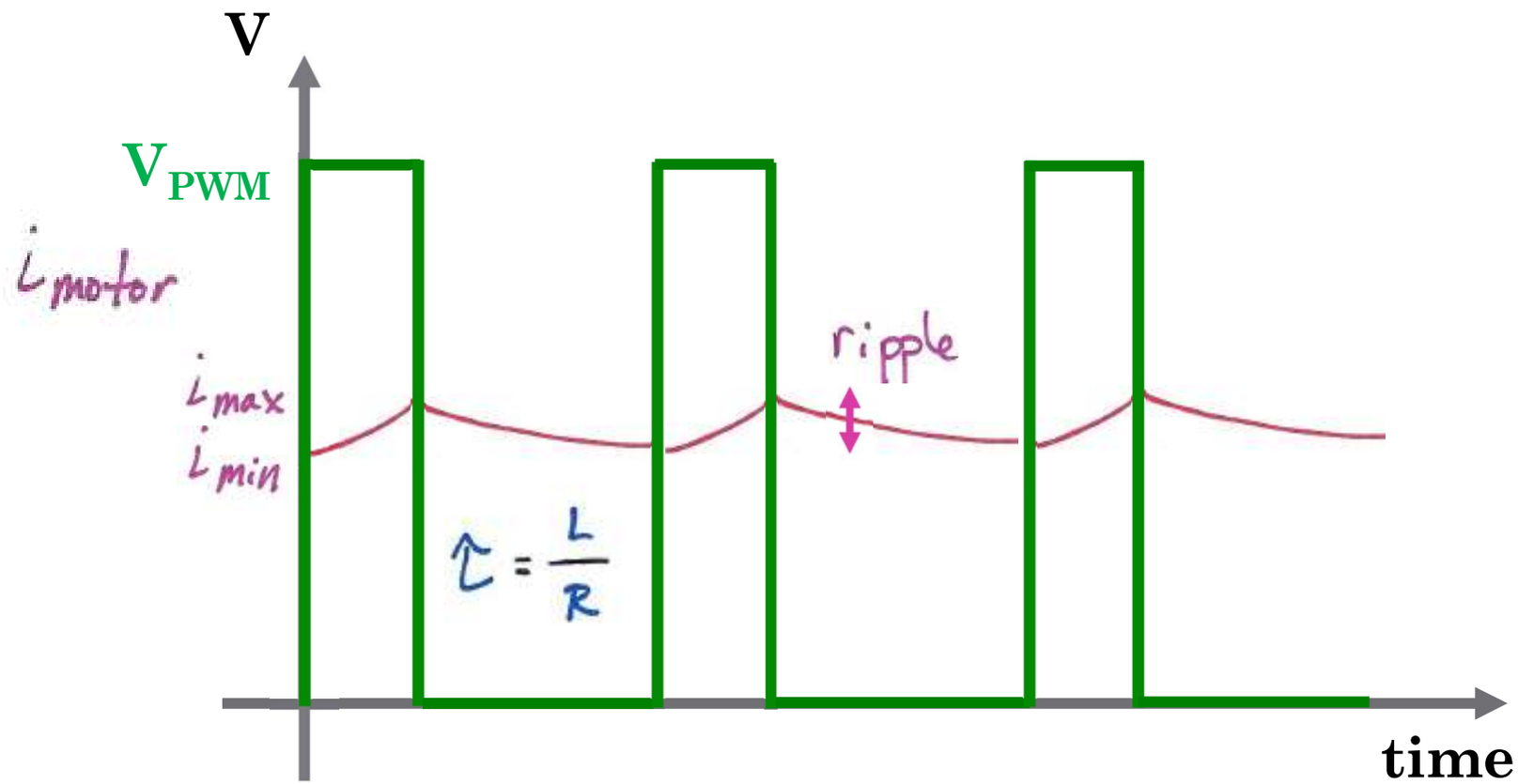
$$\text{Duty cycle}(\%) = \frac{\text{"ON" time}}{\text{period}} \cdot 100$$

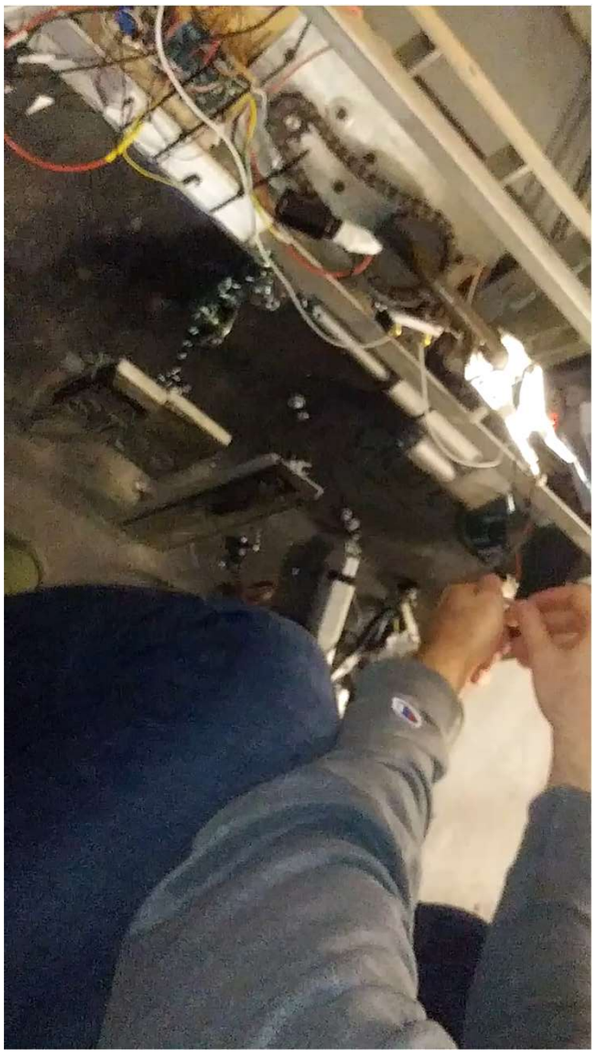
$$V^* = \text{Duty cycle} \cdot V_{motor}$$

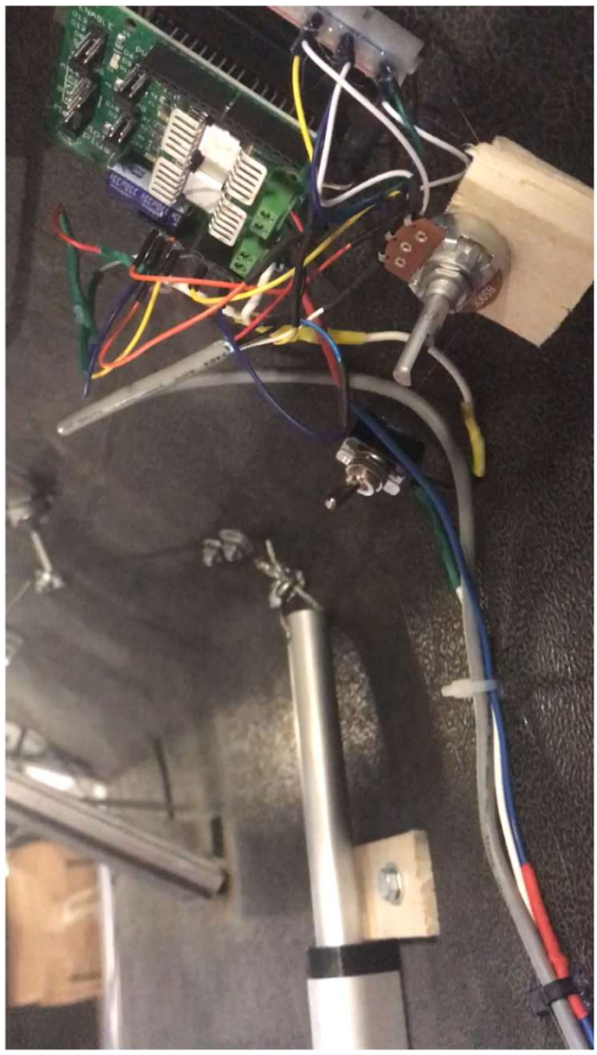


$$\text{PWM frequency} = \frac{1}{\text{period}}$$

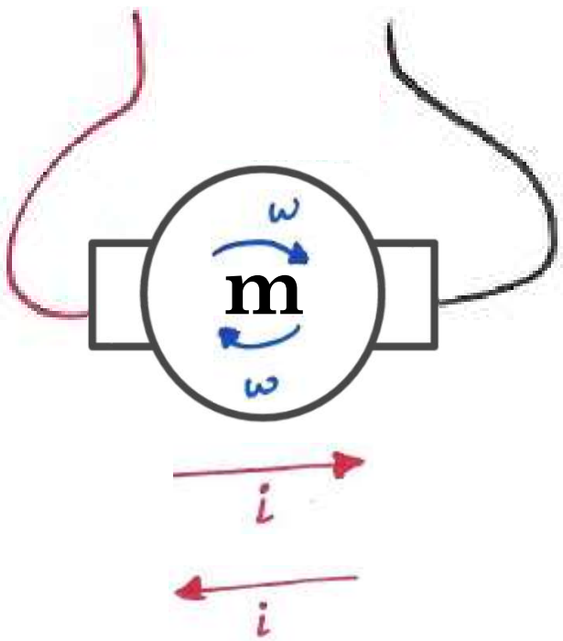
Torque ripple

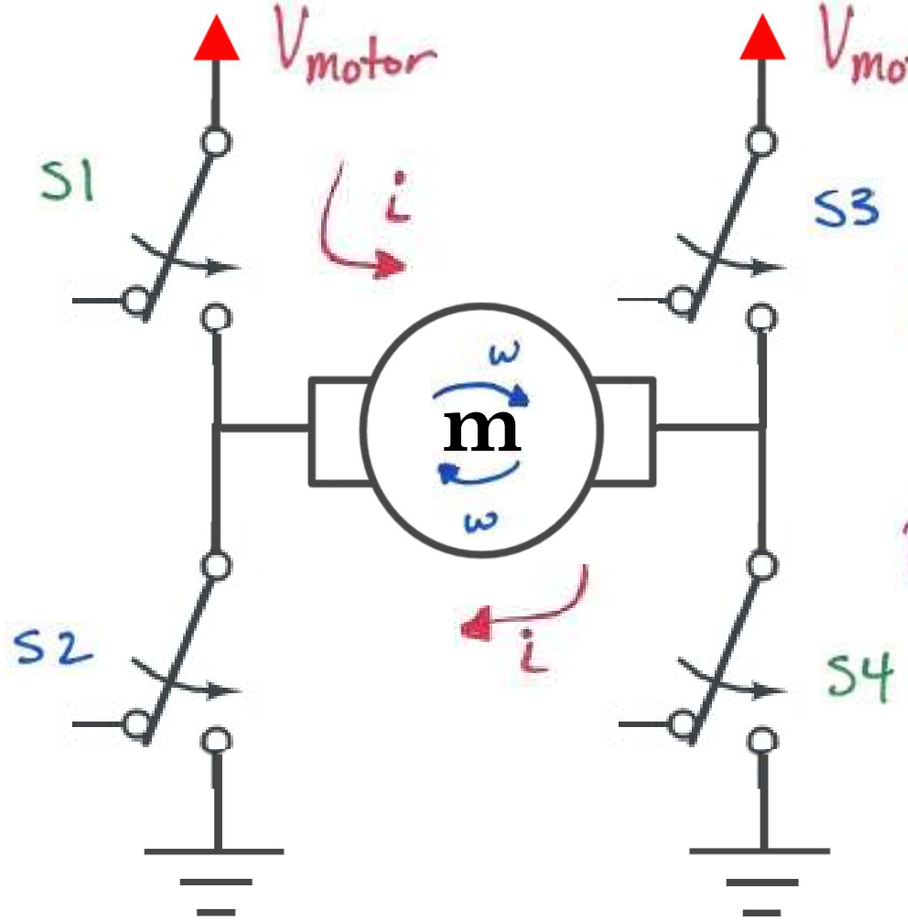






Basic direction control

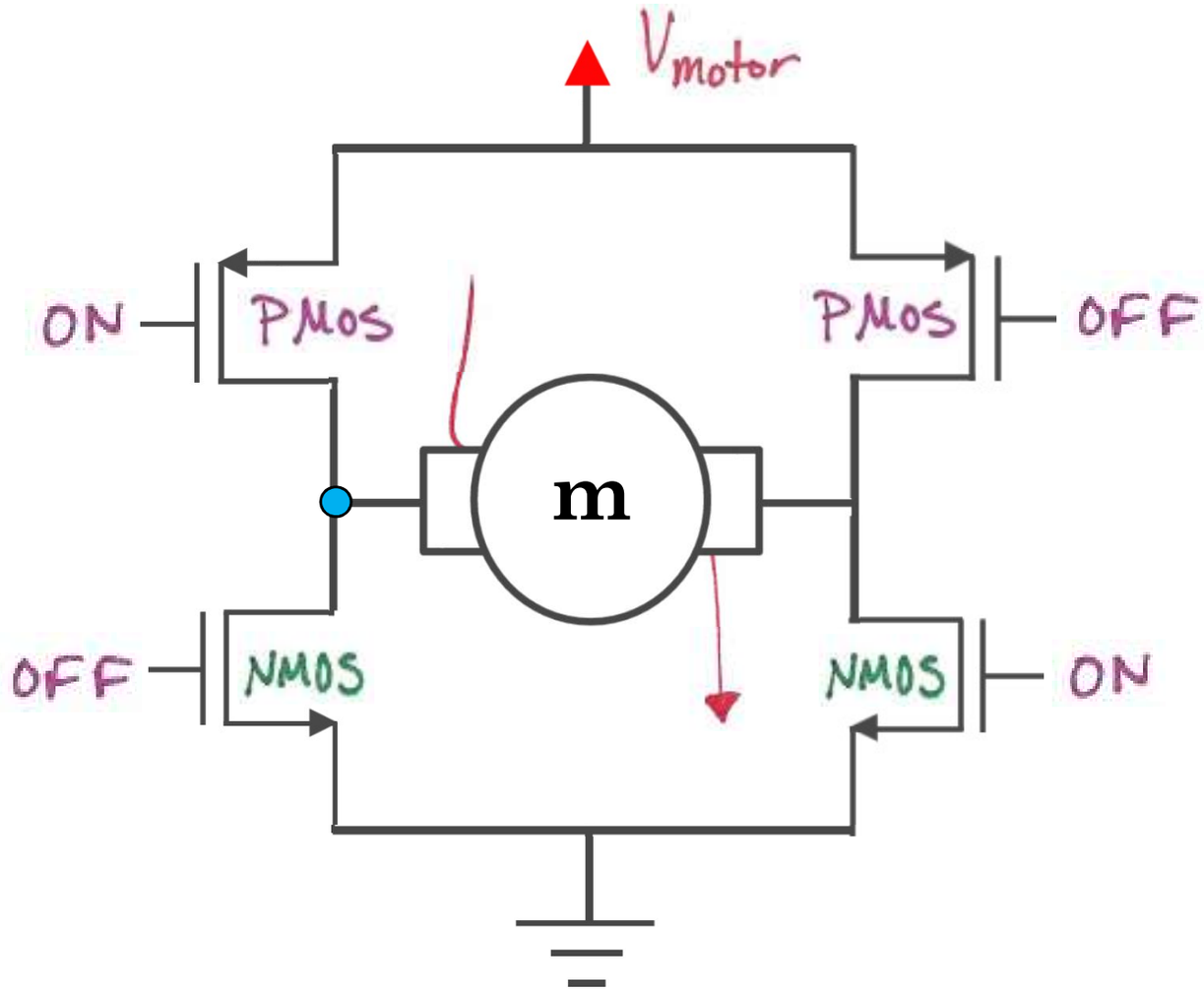




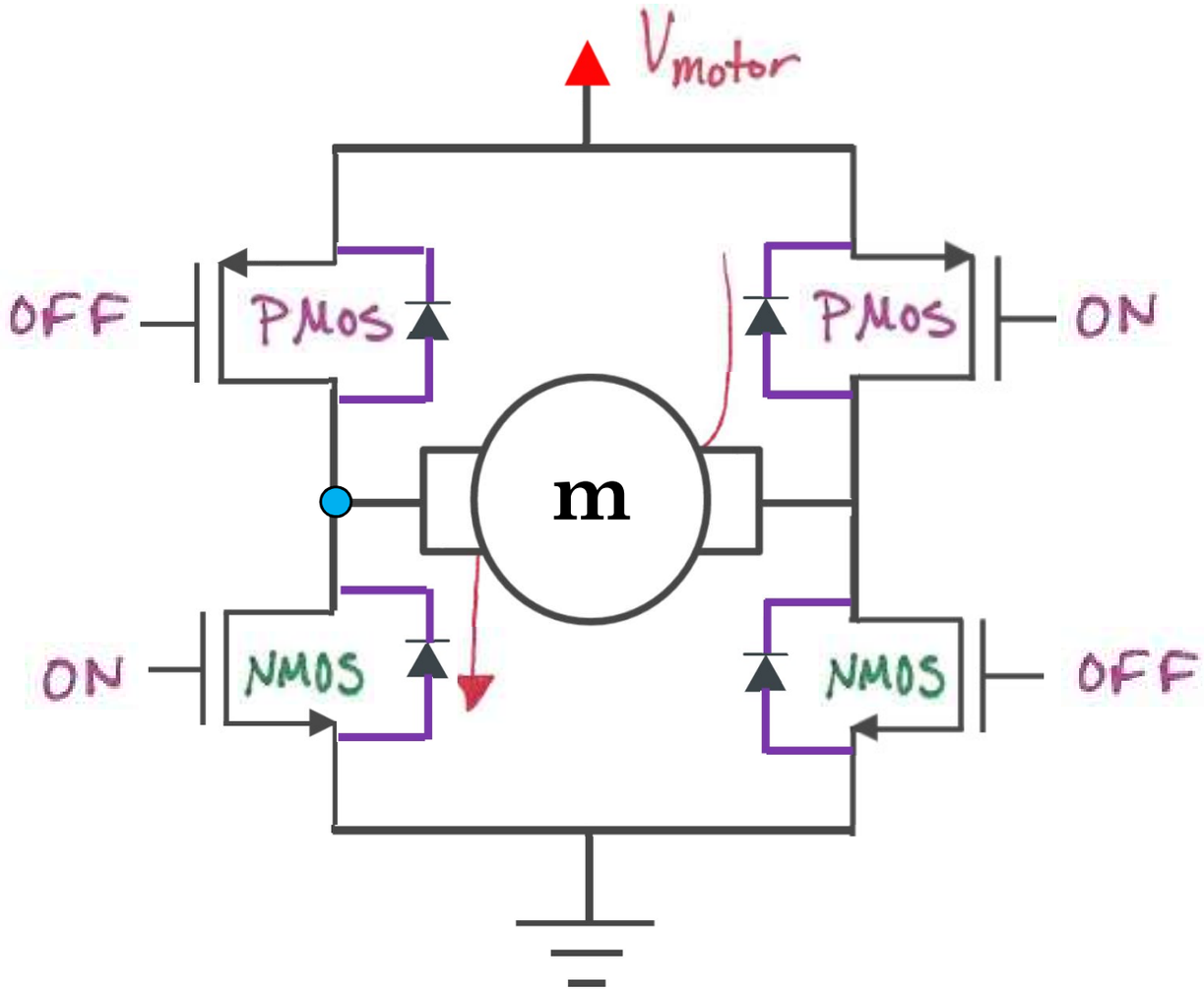
Forward: S1/S4 closed
S2/S3 open

Reverse: S2/S3 closed
S1/S4 open

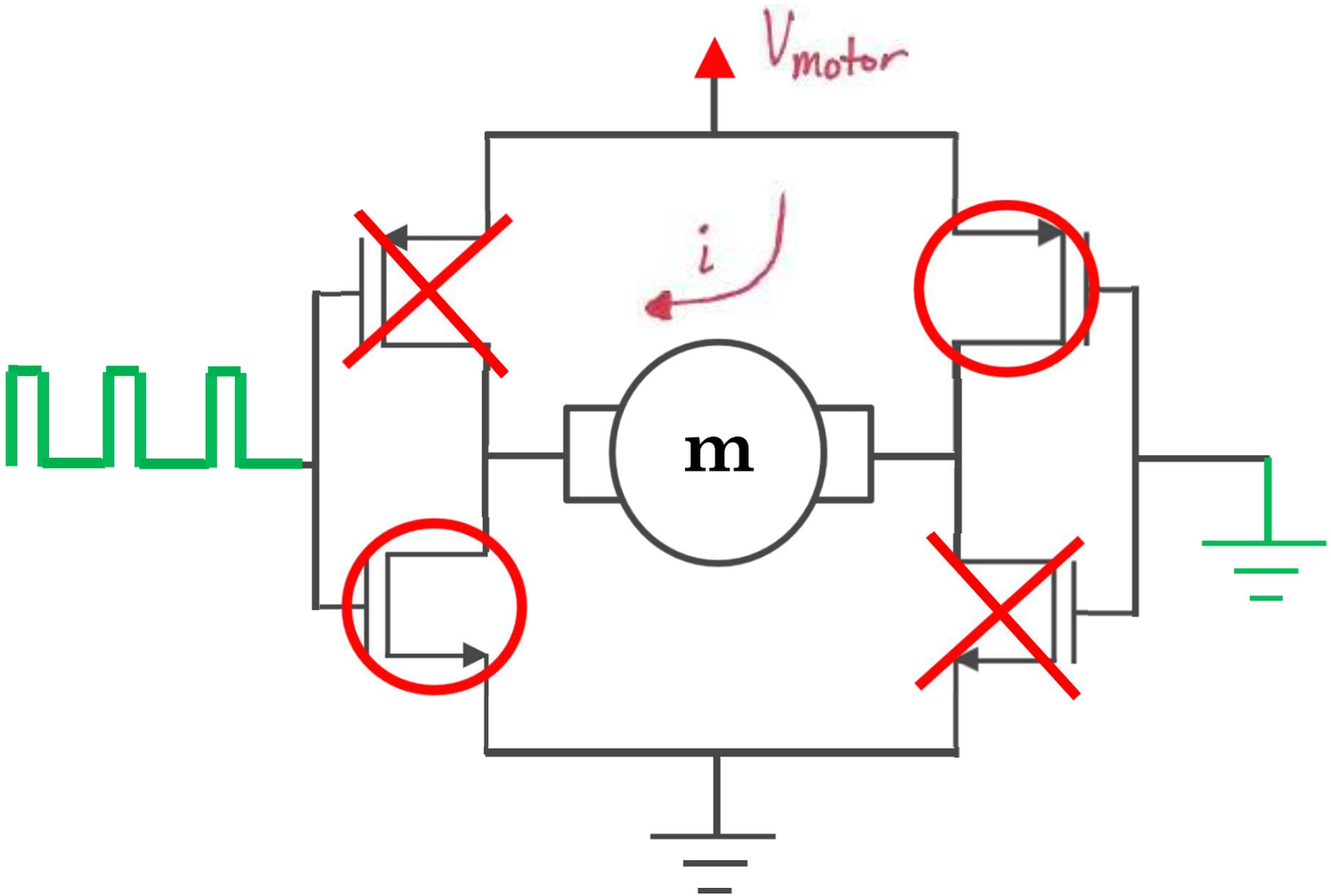
Transistor H-Bridge



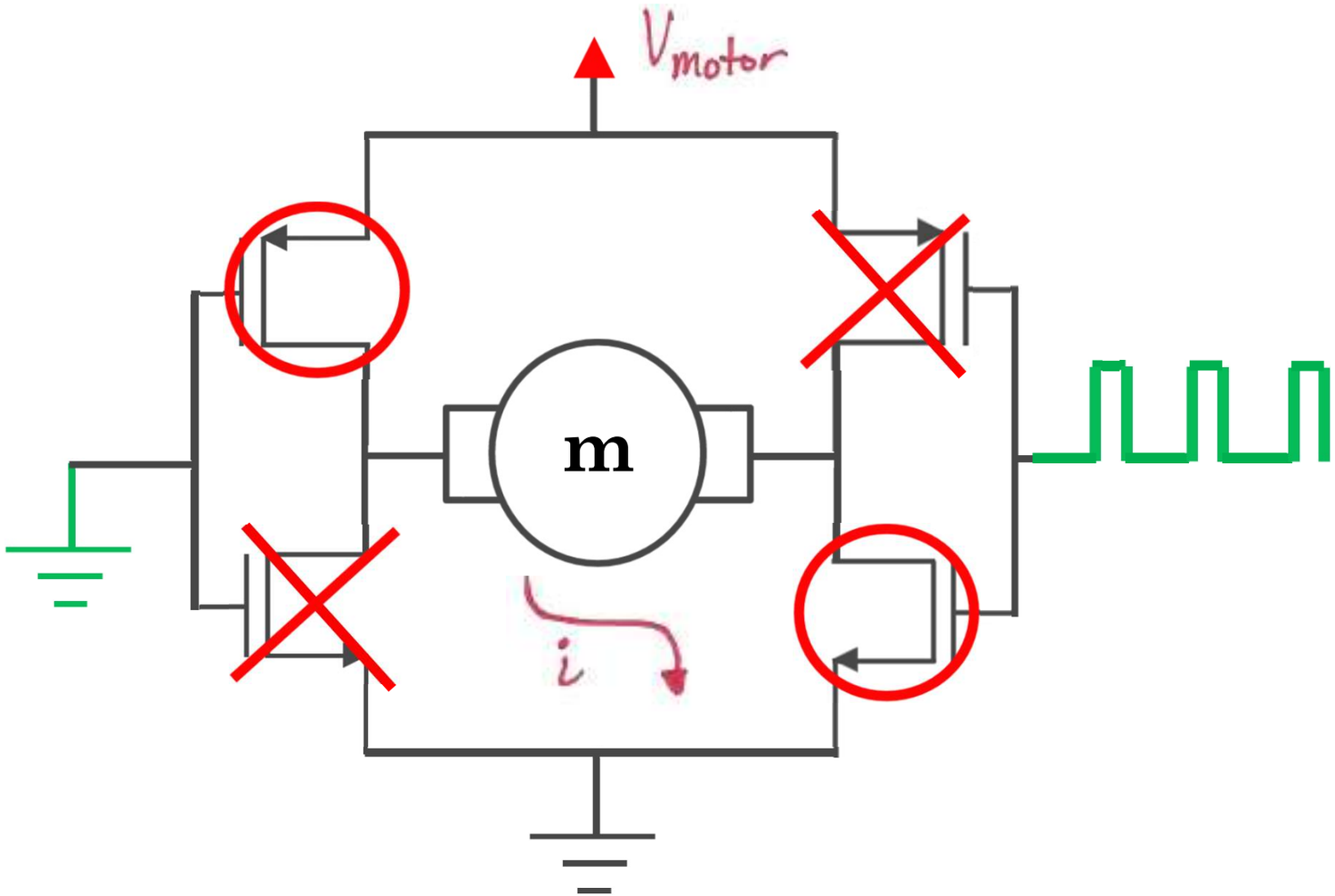
Transistor H-Bridge



Transistor H-Bridge

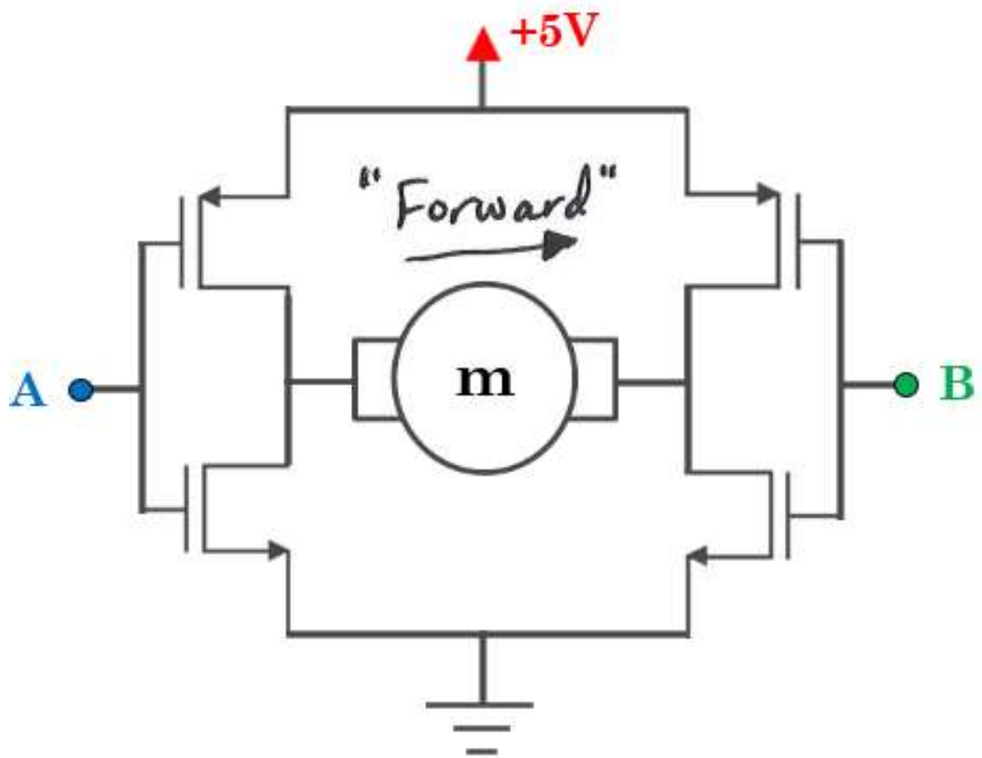


Transistor H-Bridge

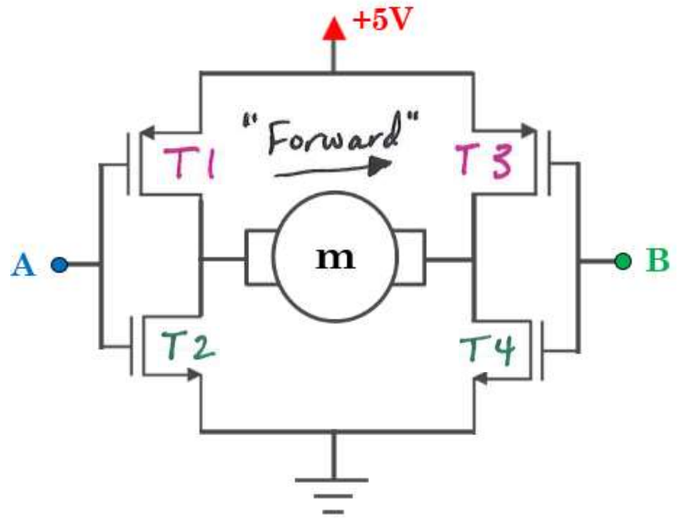


In-Class Exercise

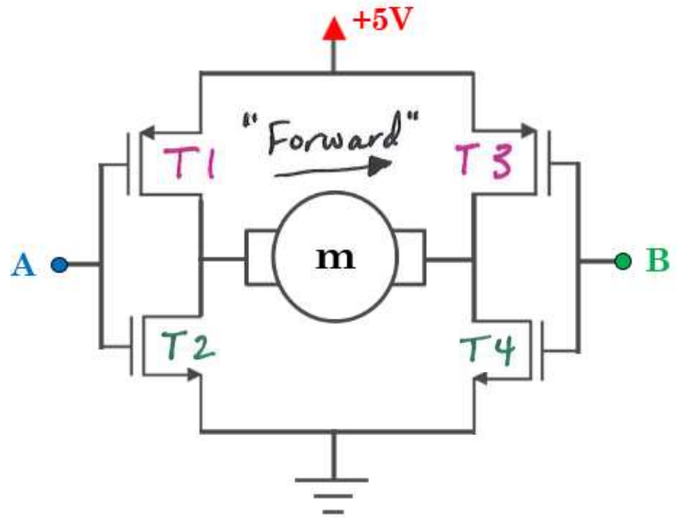
- Using the circuit below and a Raspberry Pi to drive the motor, **sketch the time-domain signals** required at **A** and **B** to turn forward at 30% full speed and reverse at 70% full speed. Assume speed is proportional to voltage, the threshold voltage for all transistors is 3V, and a V_{PWM} period of 1 msec.



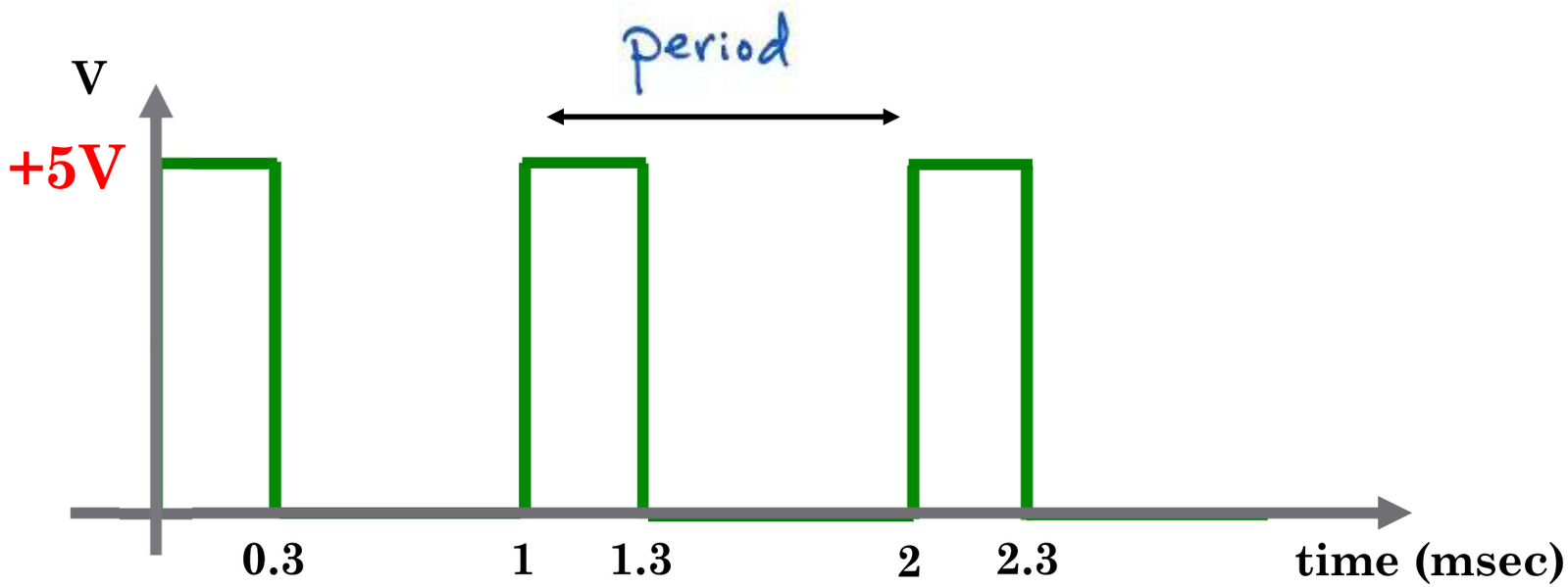
In-Class Exercise



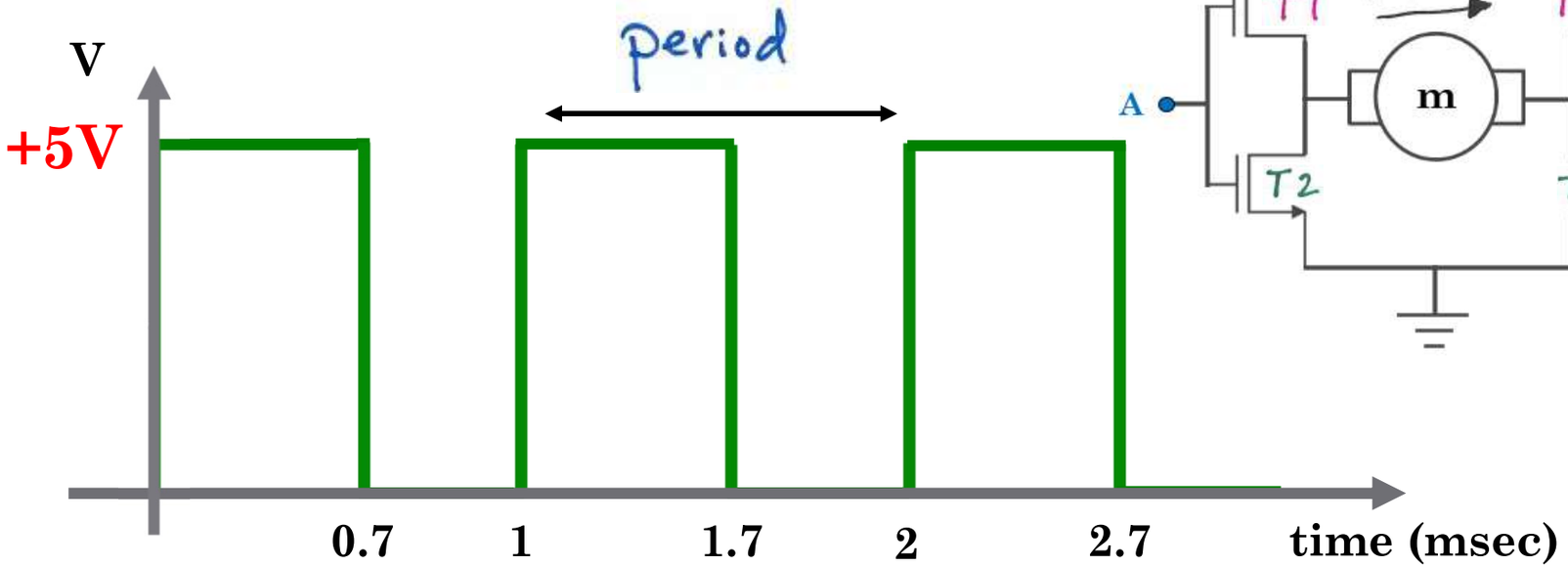
In-Class Exercise

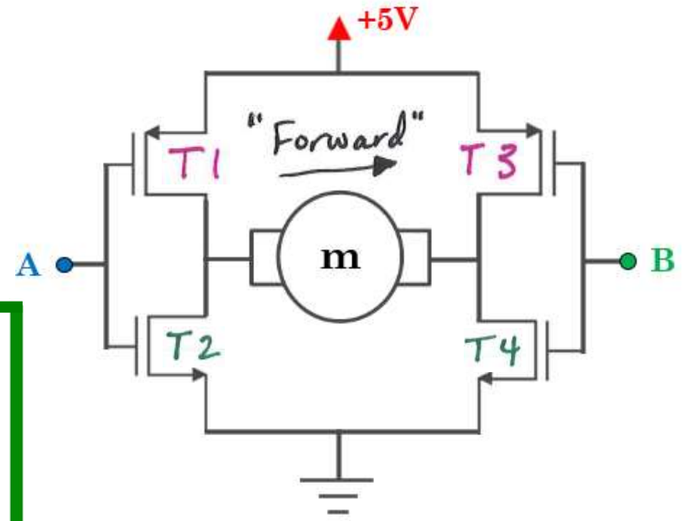


Forward
T1
T4



In-Class Exercise





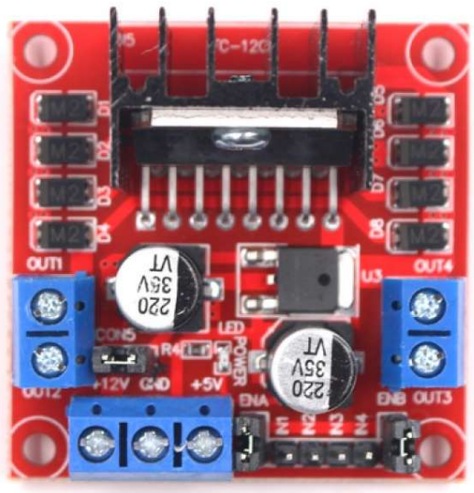
Reverse

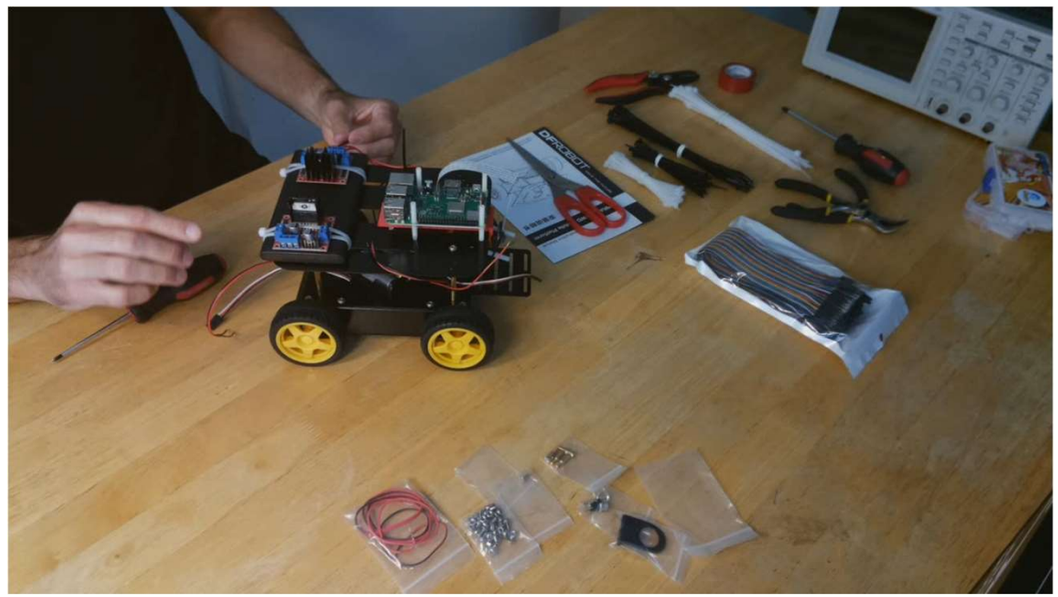
T2

T3

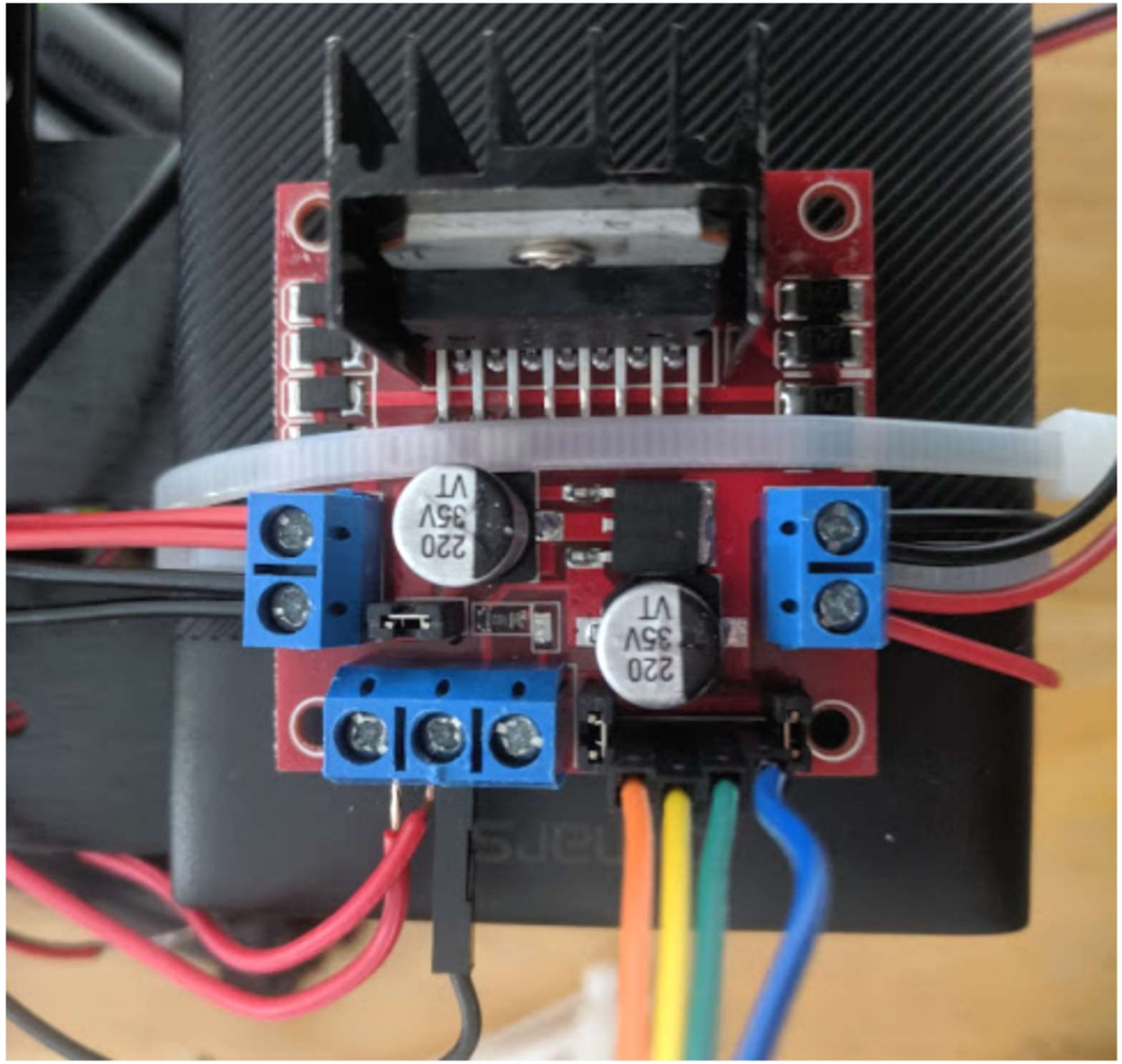
Assembly

- Wire motors to H-bridge

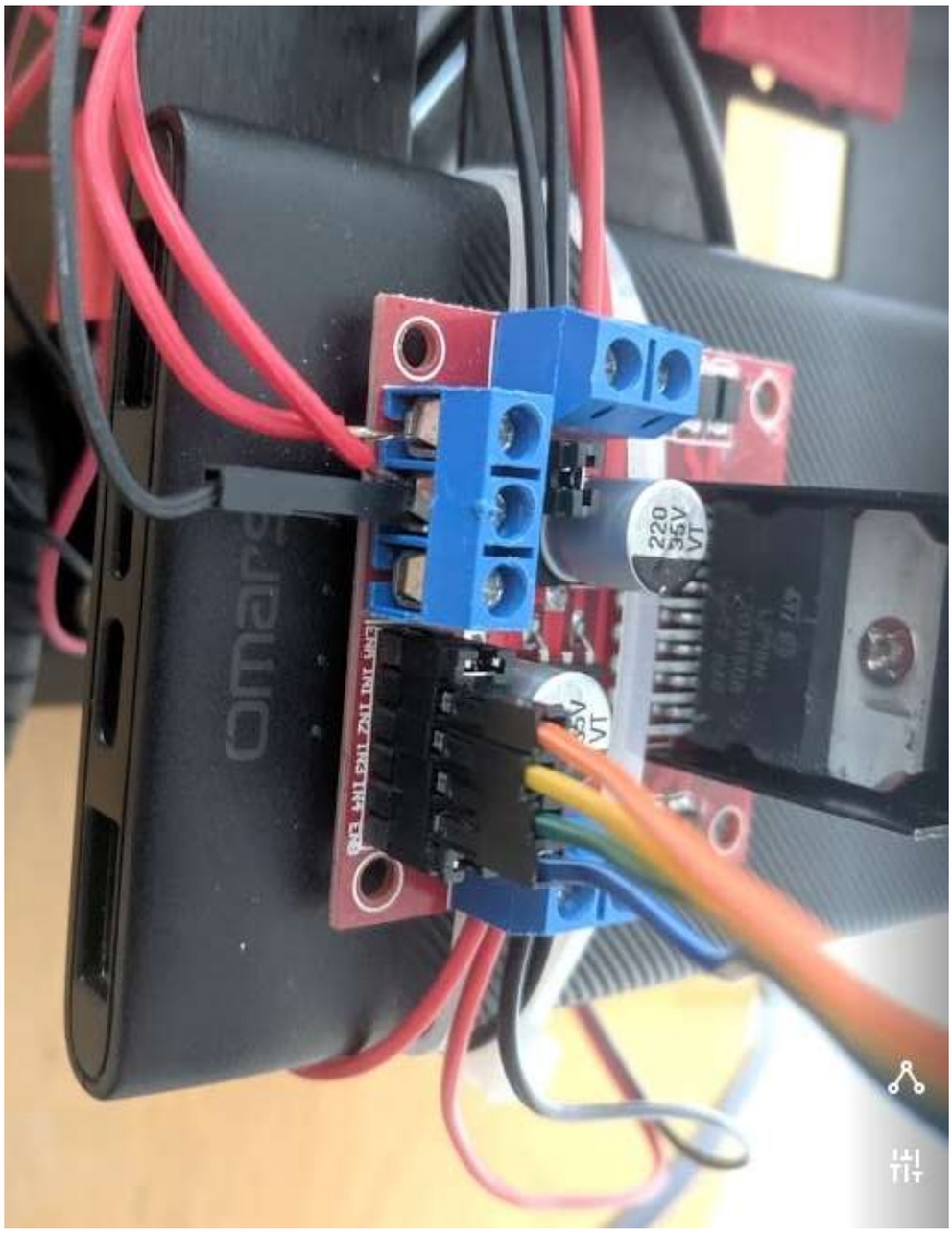


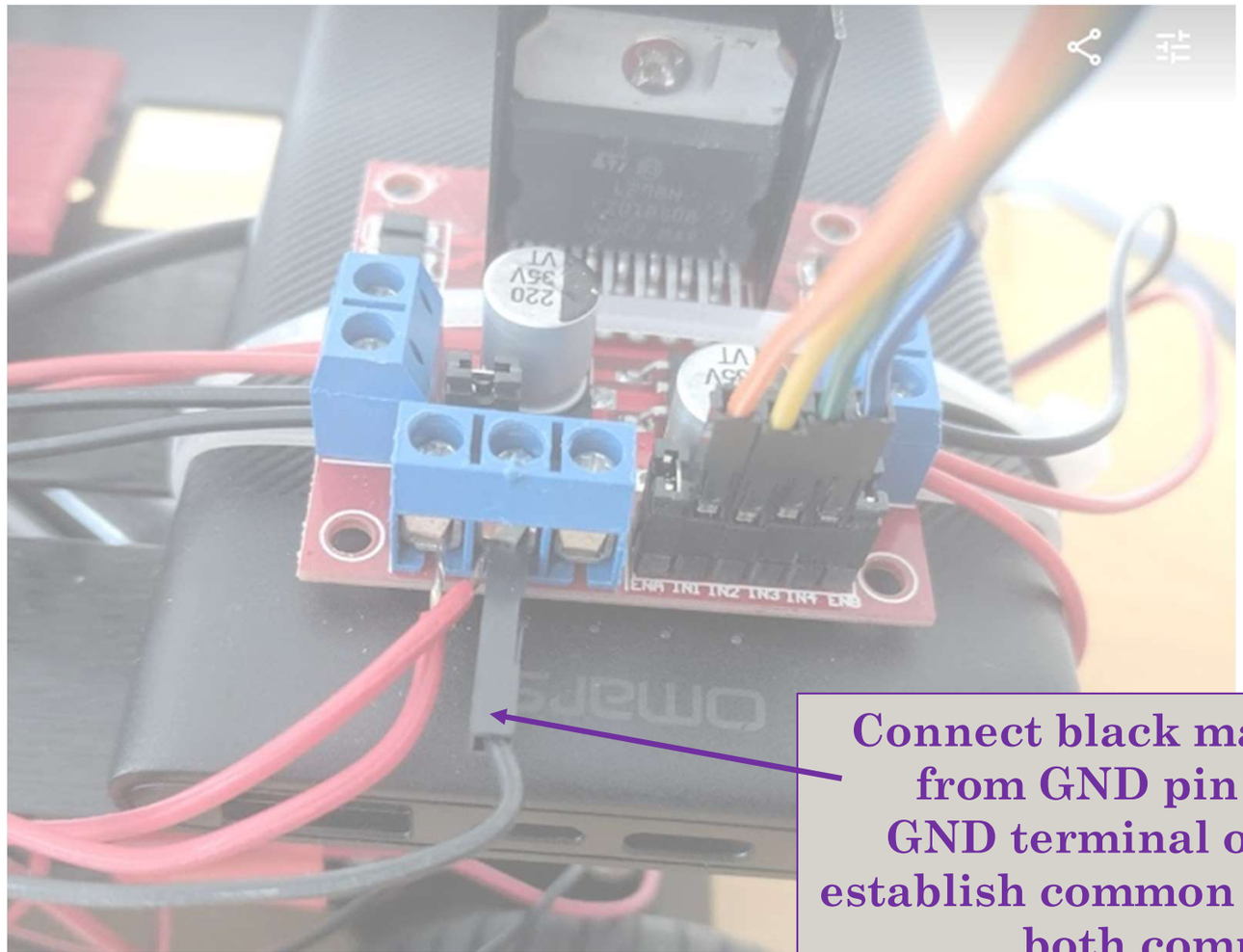


1: “+” voltage, left motors
2: “-” voltage, left motors
3: **power in** from AA battery pack
4: **GND**
5: 5V output
6: “+” voltage, right motors
7: “-” voltage, right motors



1: “+” voltage, left motors
2: “-” voltage, left motors
3: **power in** from AA battery pack
4: **GND** from AA battery pack & Raspberry Pi
5: 5V output
6: “+” voltage, right motors
7: “-” voltage, right motors

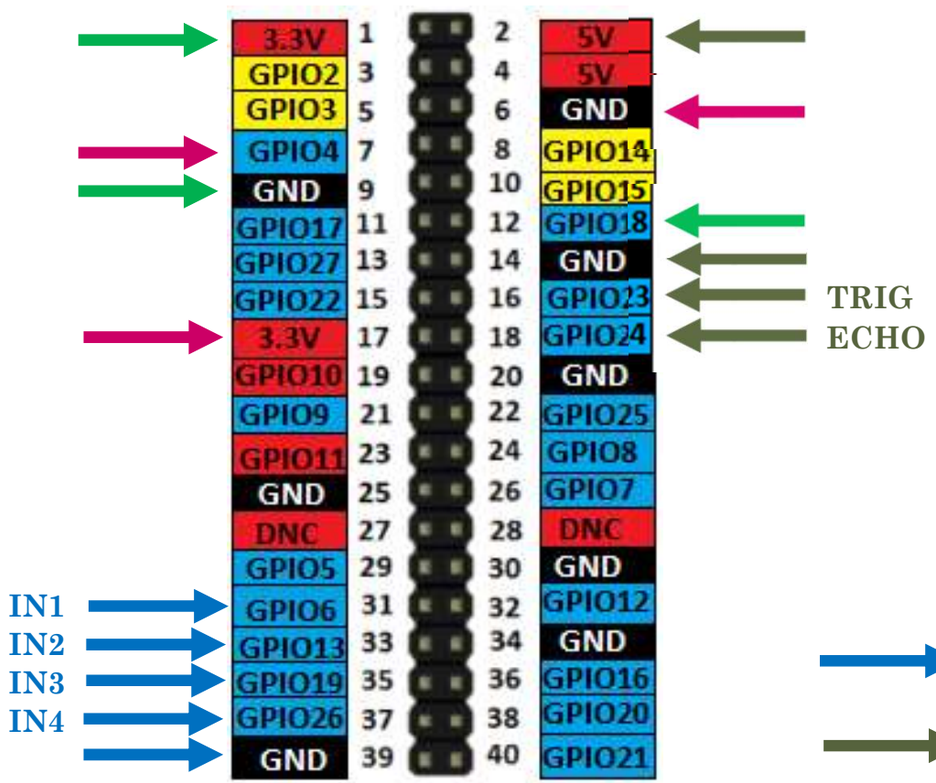

220
35V
VT

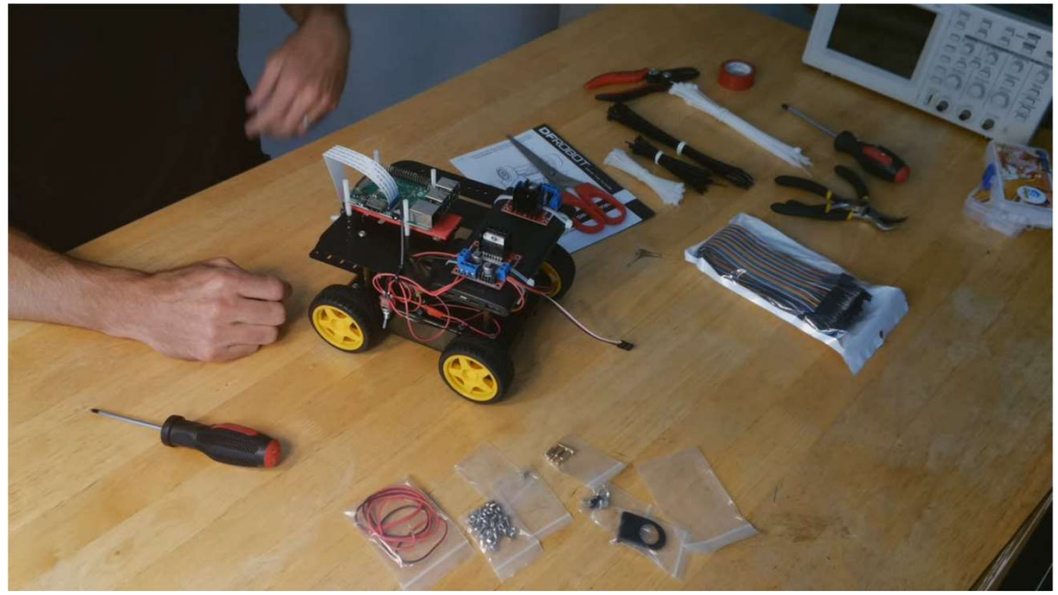


Connect black male-female wire from GND pin 39 on Rpi to GND terminal on H-bridge to establish common ground between both components

Assembly

- Wire H-bridge to Raspberry Pi





RPi to H-bridge

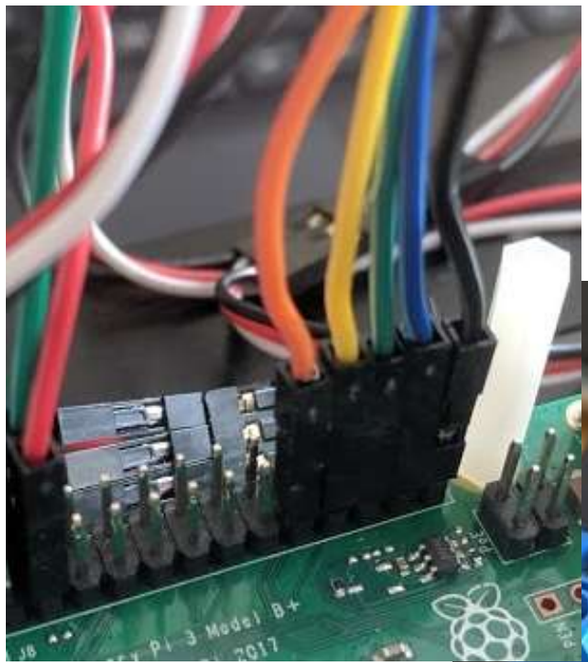
Distance sensor

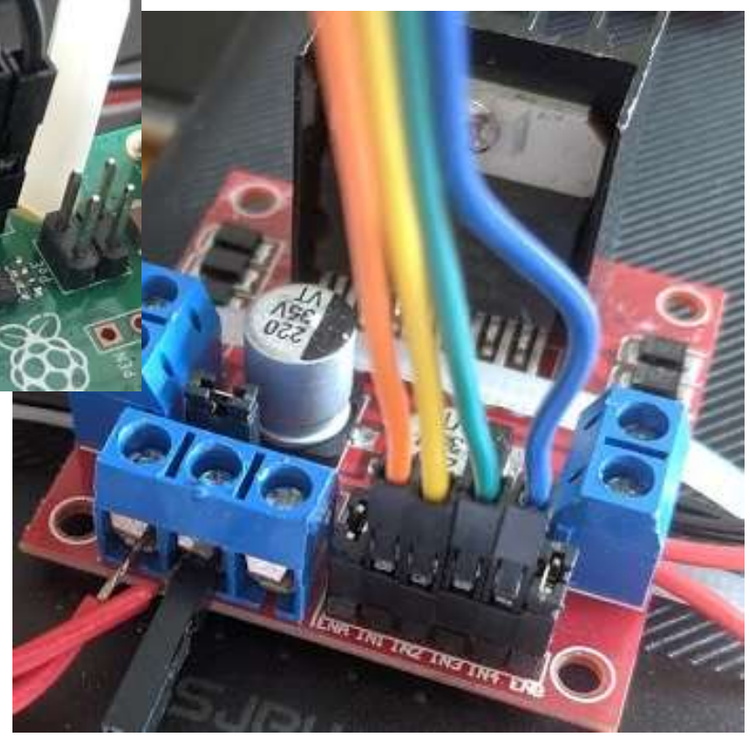
Right encoder

Left encoder

H-Bridge Circuit

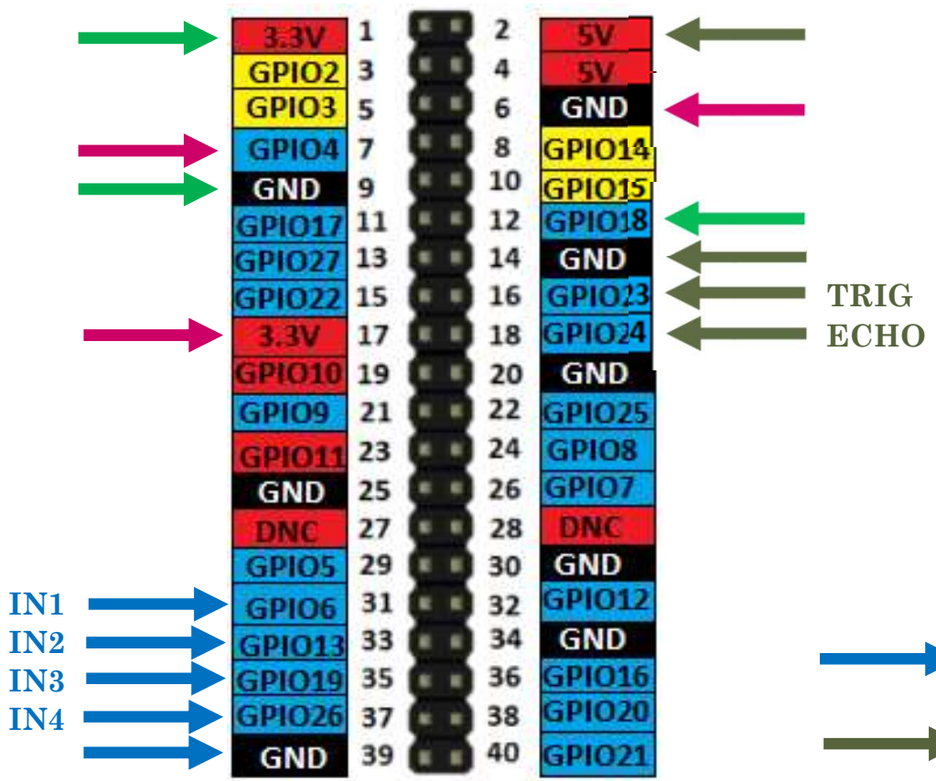
- Plug **orange** female-female wire into **IN1** and pin 31
- Plug **yellow** female-female wire into **IN2** and pin 33
- Plug **green** female-female wire into **IN3** and pin 35
- Plug **blue** female-female wire into **IN4** and pin 37

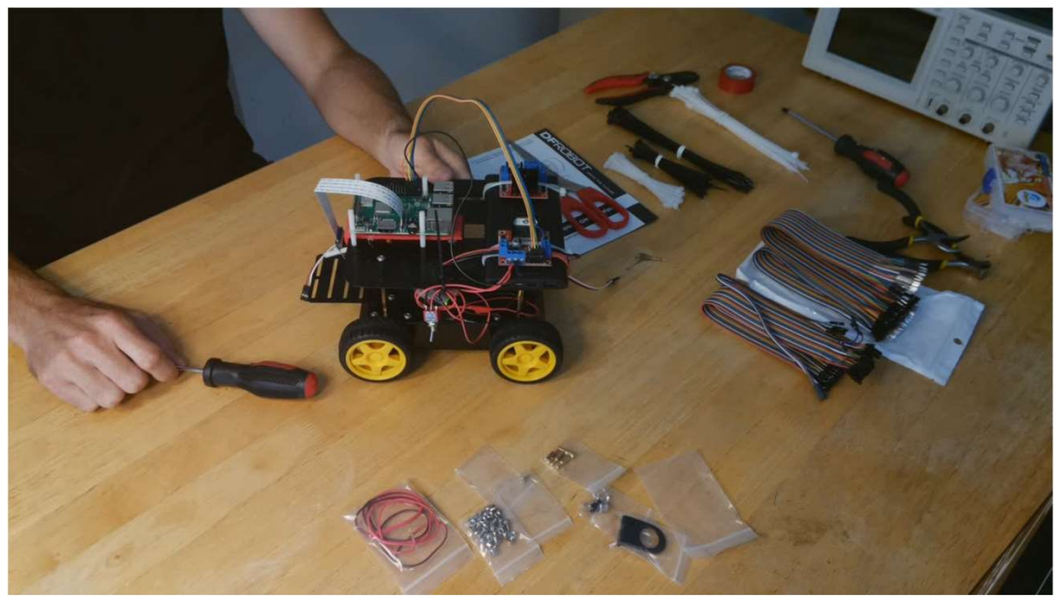




Assembly

- Wire motor encoders to Raspberry Pi





RPi to H-bridge

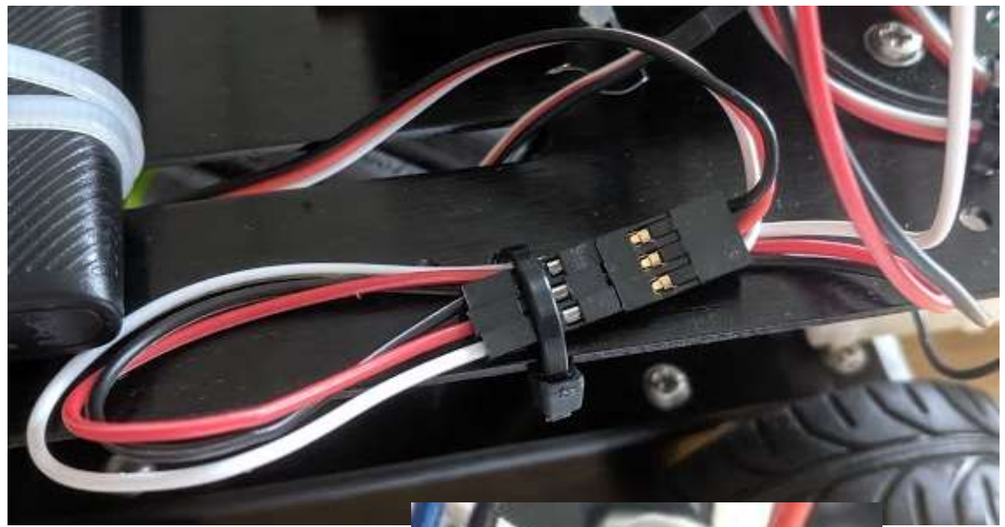
Distance sensor

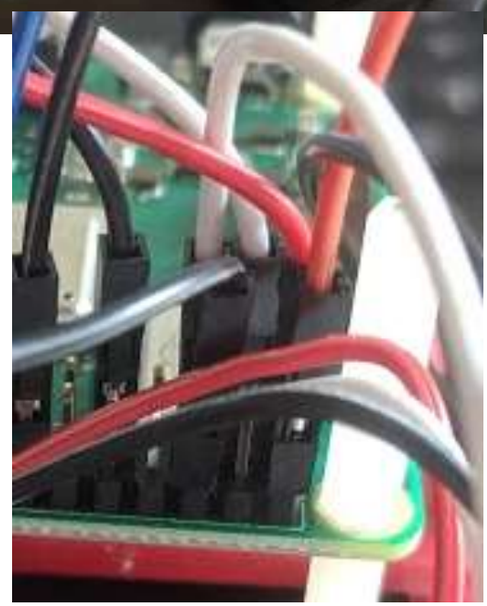
Right encoder

Left encoder

Encoder Circuit

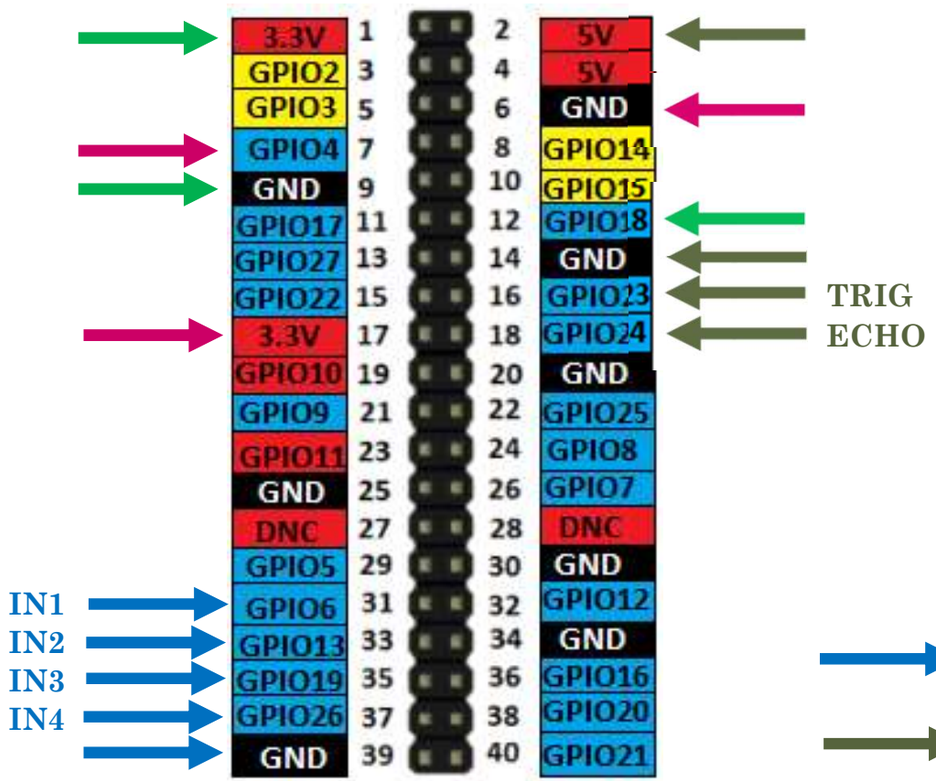
- Plug **red** male-female wire into pin 1 or 17
  - Encoder 3.3V **power**
- Plug **black** male-female wire into pin 9 or 6
  - Encoder **GND**
- Plug white male-female wire into pin 12 or 7
  - Encoder signal

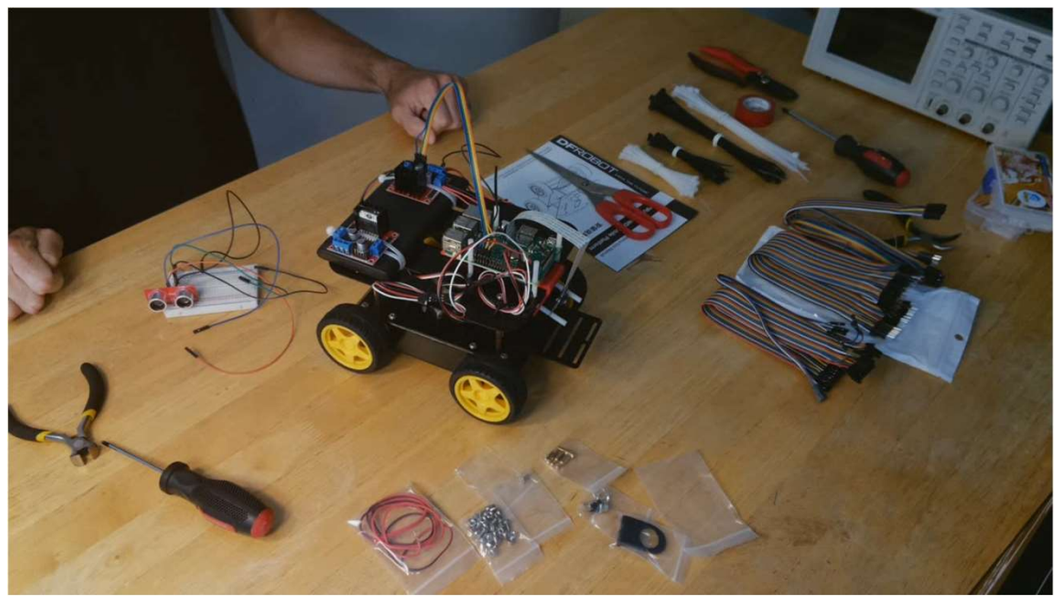




Assembly

- Mount & wire distance sensor





RPi to H-bridge

Distance sensor

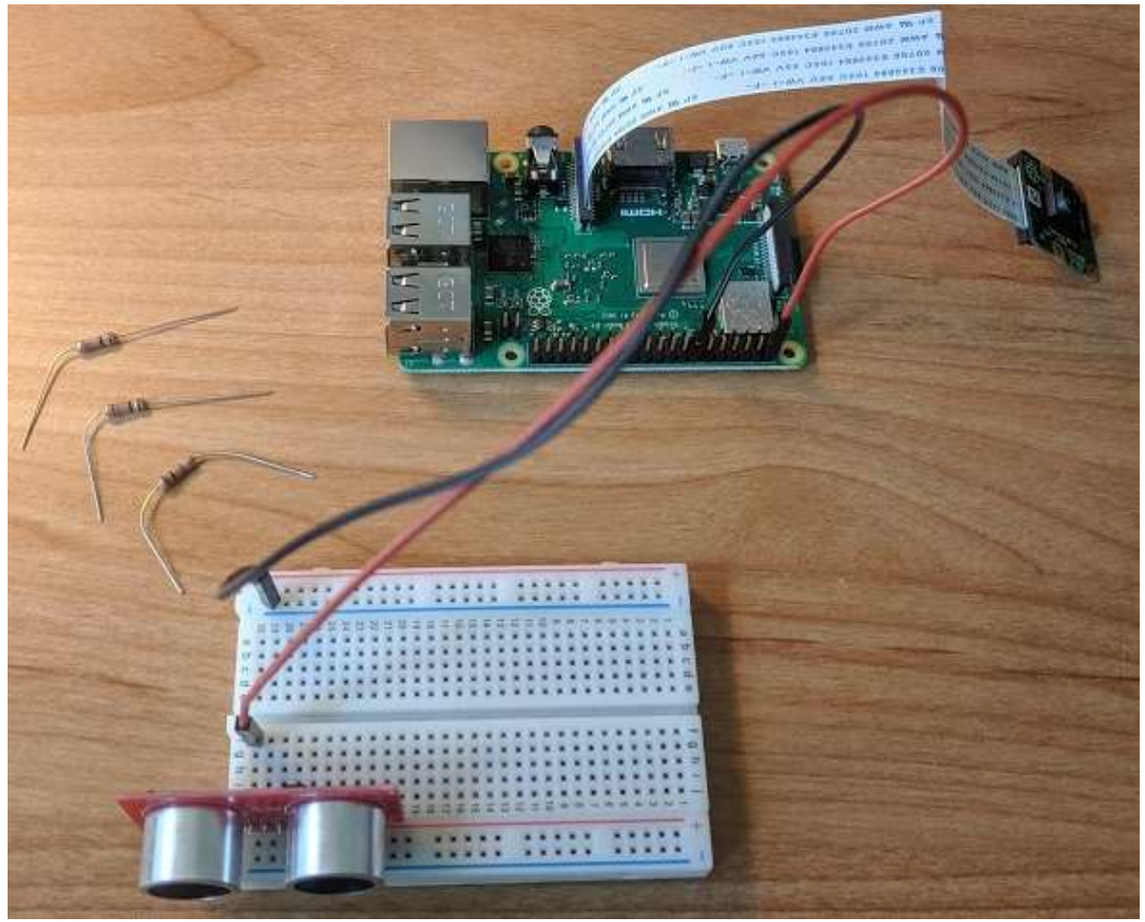
Right encoder

Left encoder

Distance Sensor Circuit

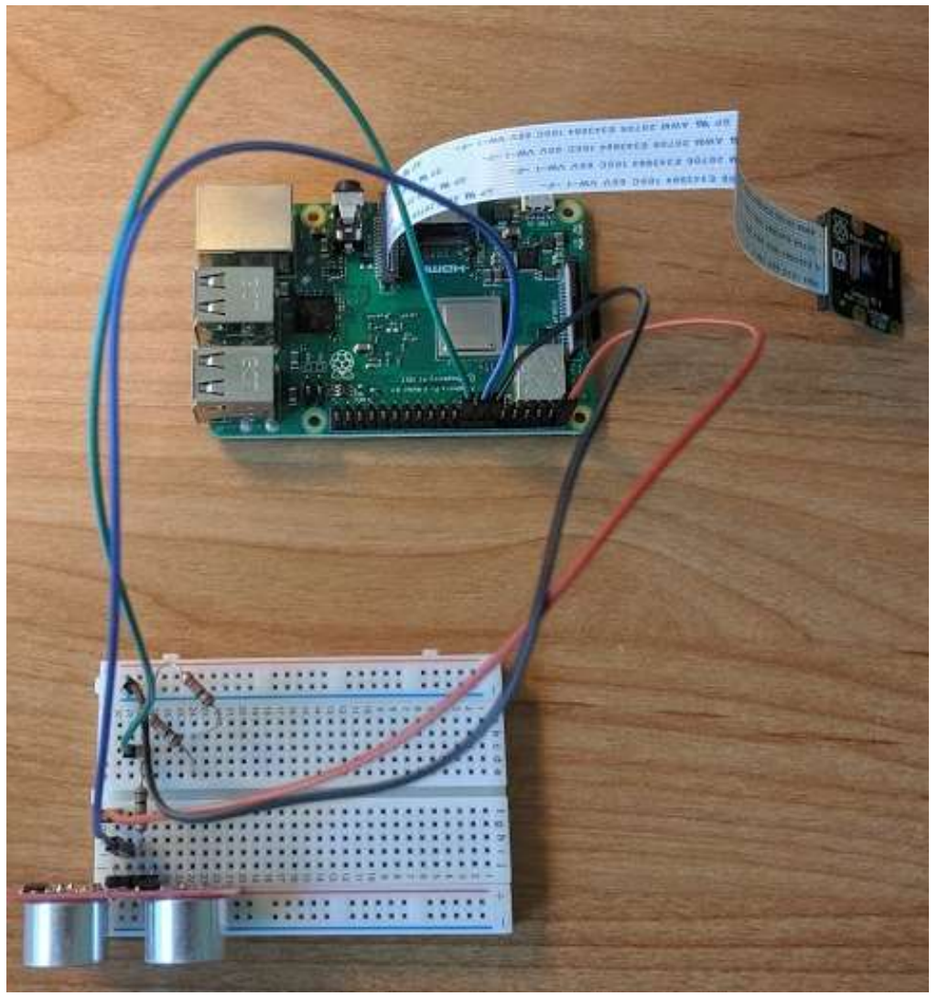
- Plug **red** male-female wire into pin 2
  - **5V** Vcc supply
- Plug **black** male-female wire into pin 14
  - **GND**

- Plug 5V into sensor Vcc
- Plug GND into breadboard ground rail



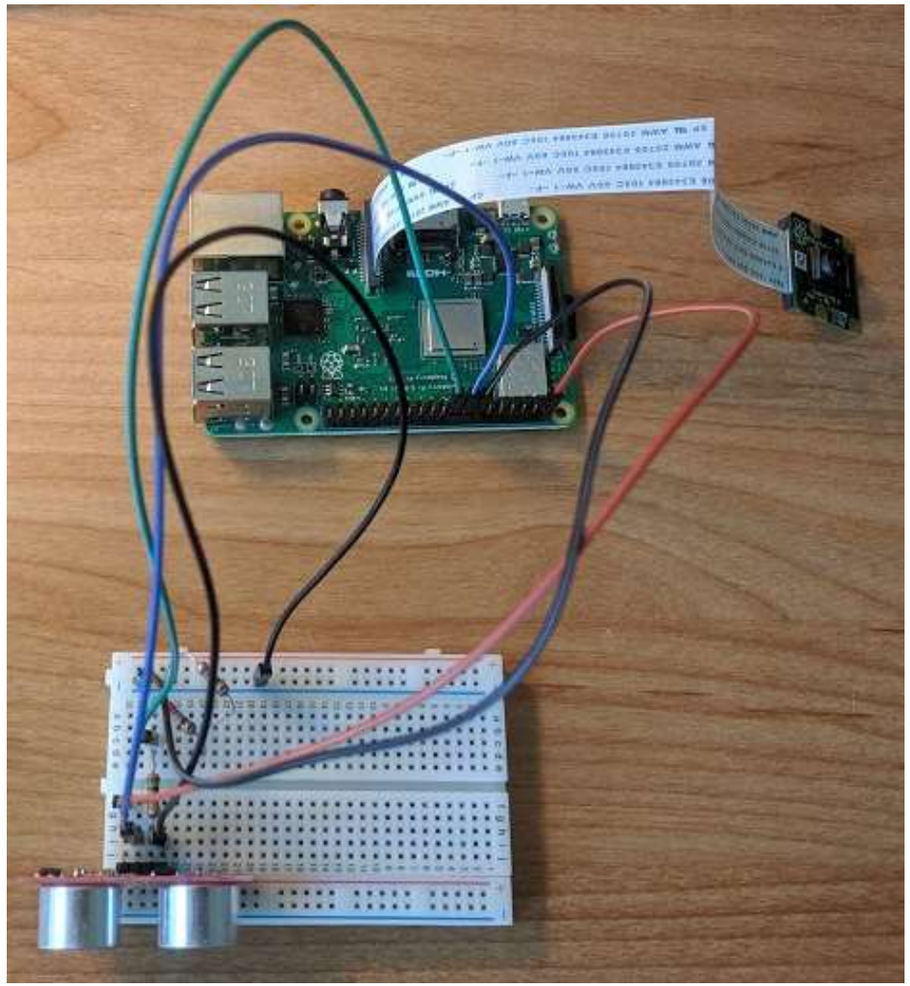
Distance Sensor Circuit

- Plug **blue** male-female wire into pin 16
  - Sensor **Trig**
- Plug **green** male-female wire into pin 18
  - Sensor **Echo**
- Connect each wire on breadboard



Distance Sensor Circuit

- Plug **black** male-male wire between sensor **GND** and breadboard **GND** rail

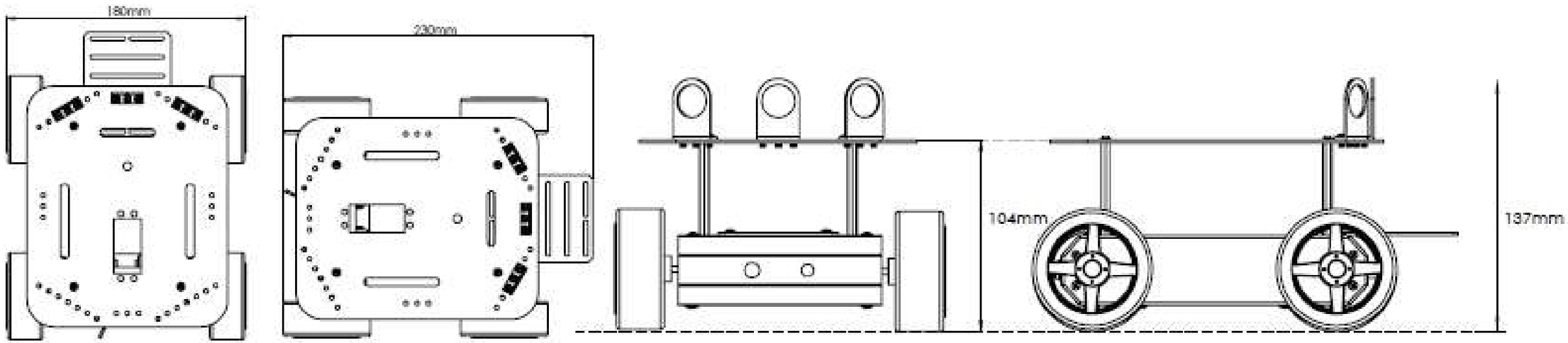


Assembly

- **Double check all electrical connections!**
- Power Raspberry Pi via USB battery pack
- Connect to Raspberry Pi using Putty/Terminal and VNC

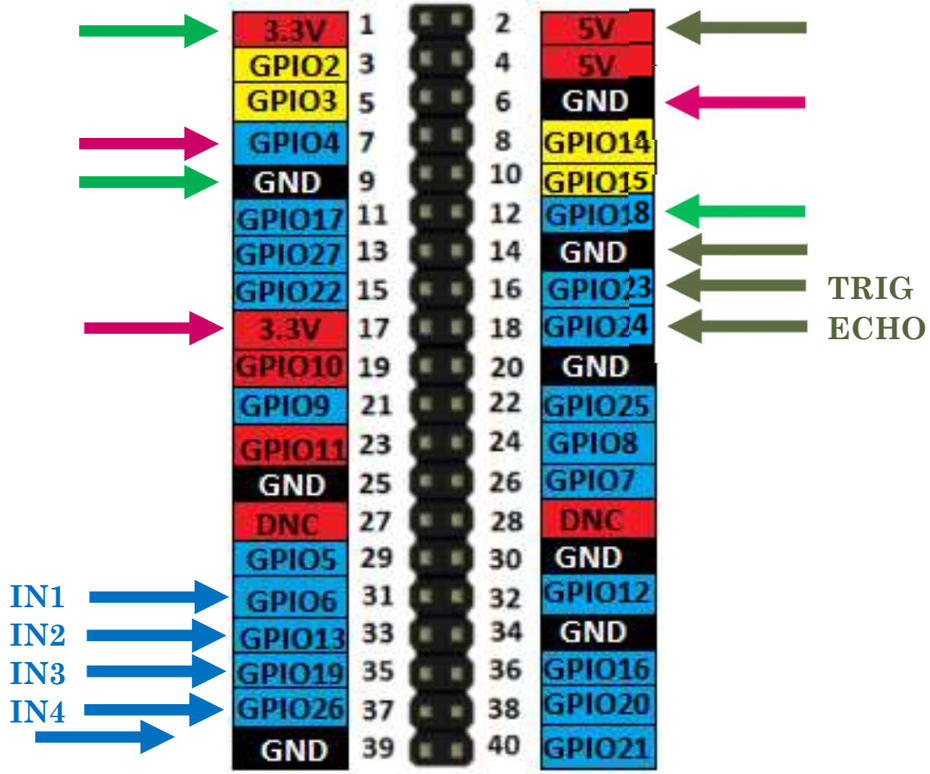
DFROBOT
MOBILE PLATFORM

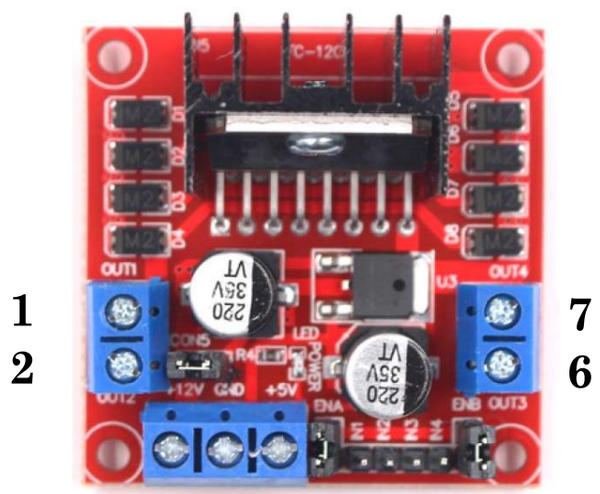
Dimensions :



Teleoperation

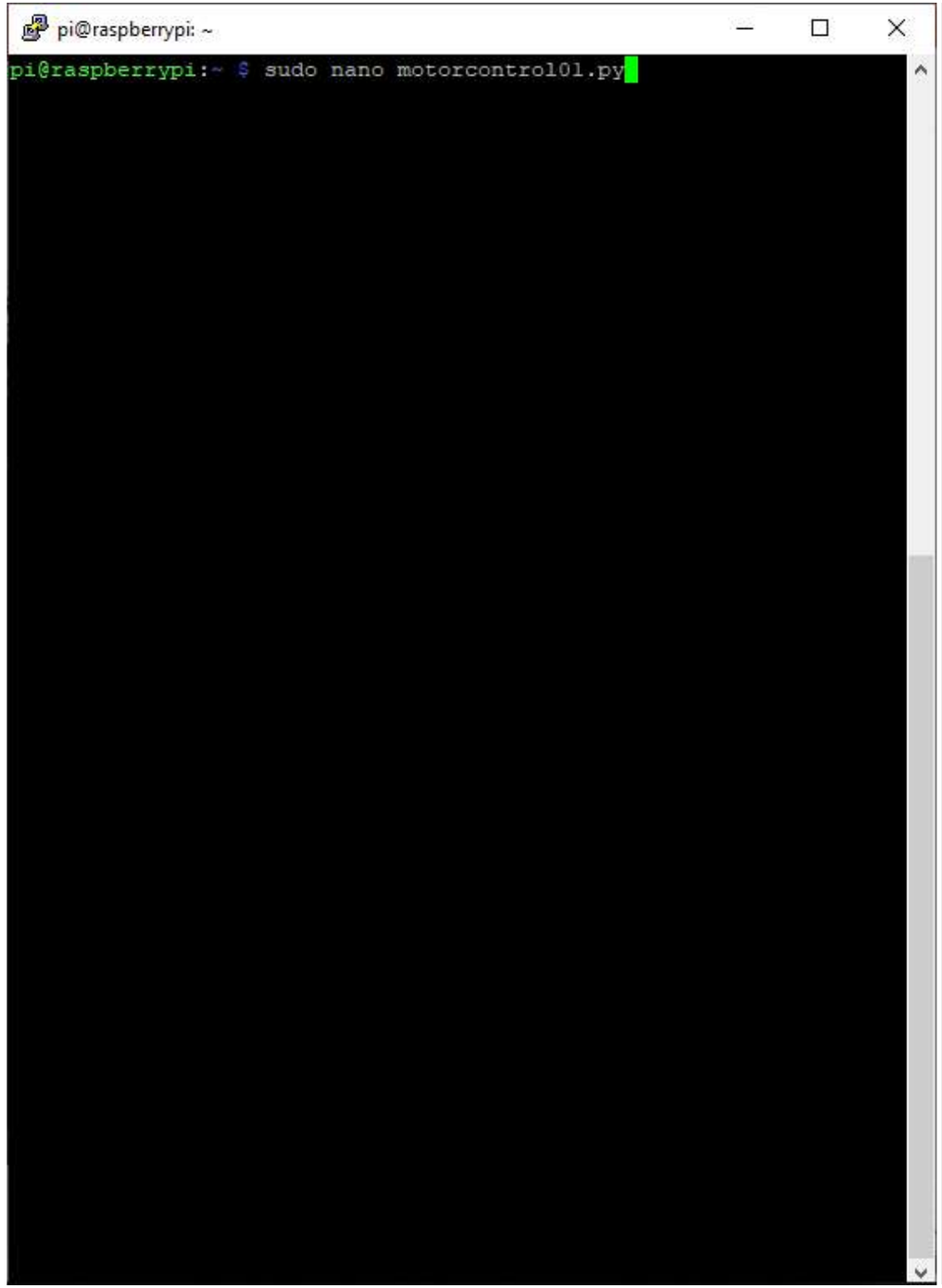
- Motor direction controlled by sending a HIGH or LOW signal to the drive for each motor
- Left motors
  - IN1: HIGH & IN2: LOW
  - IN1: LOW & IN2: HIGH
- Right motors
  - IN3: HIGH & IN4: LOW
  - IN3: LOW & IN4: HIGH





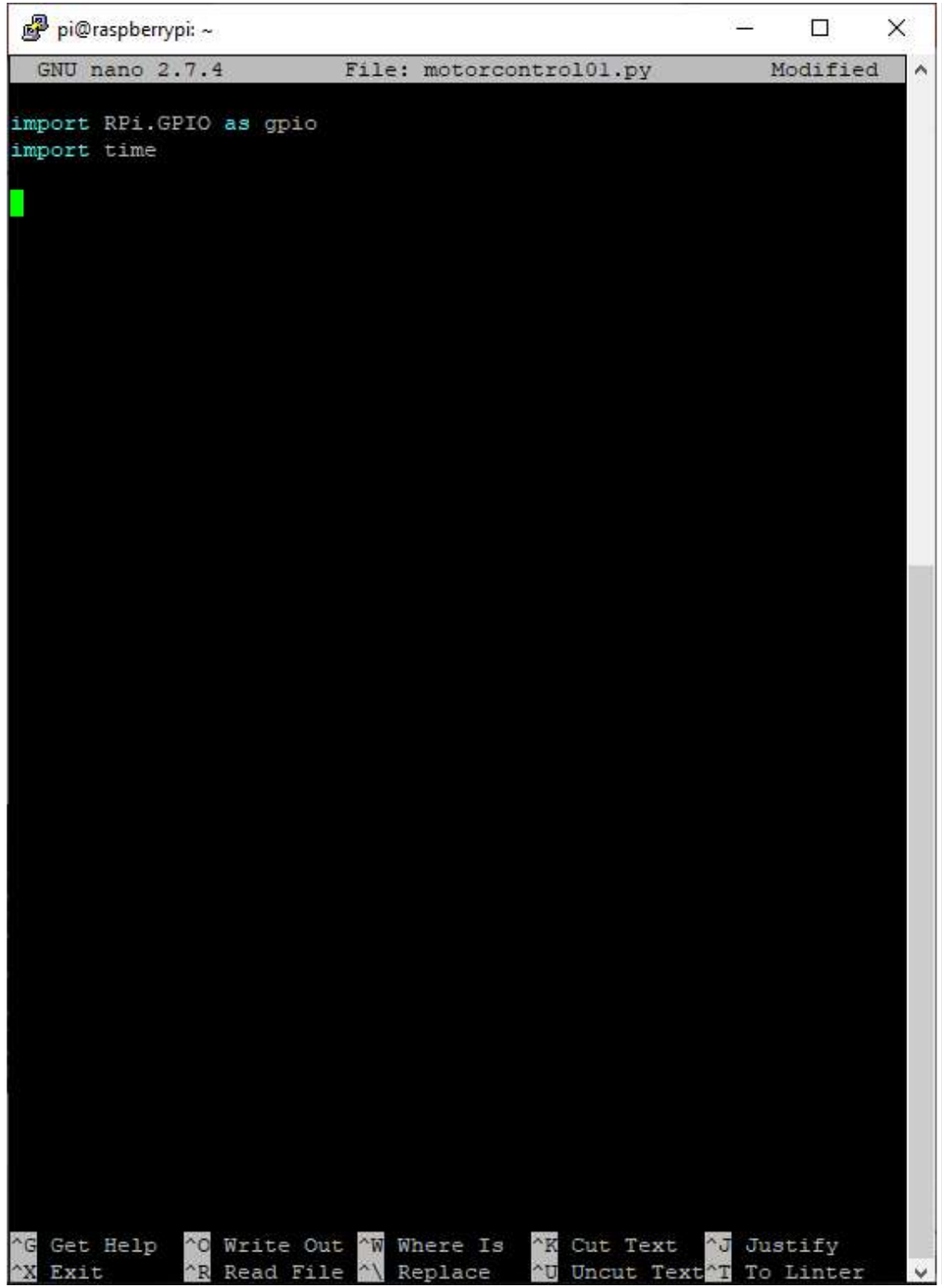
Teleoperation

- Create new Python script: ***motorcontrol01.py***



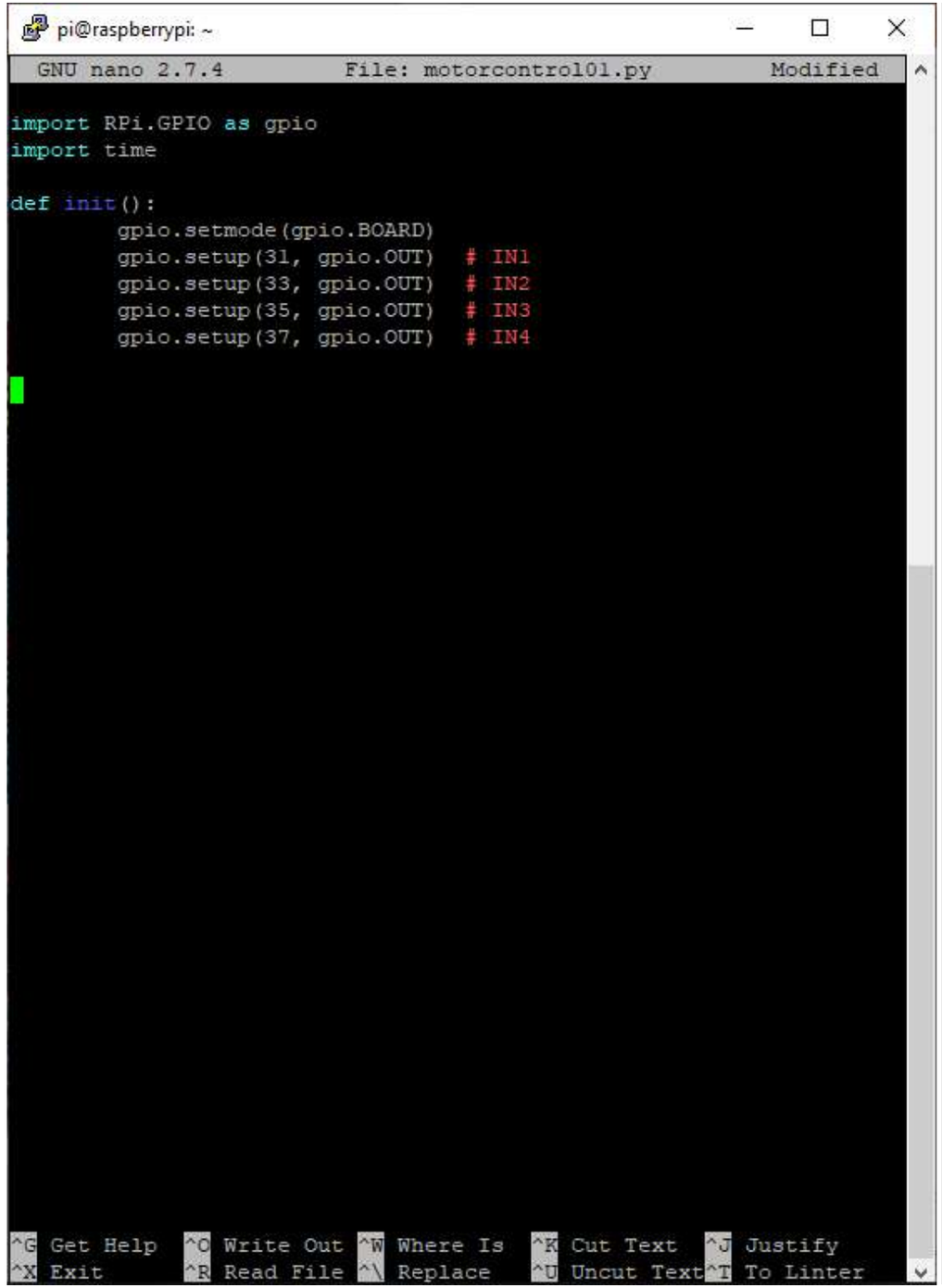
Teleoperation

- Import required packages



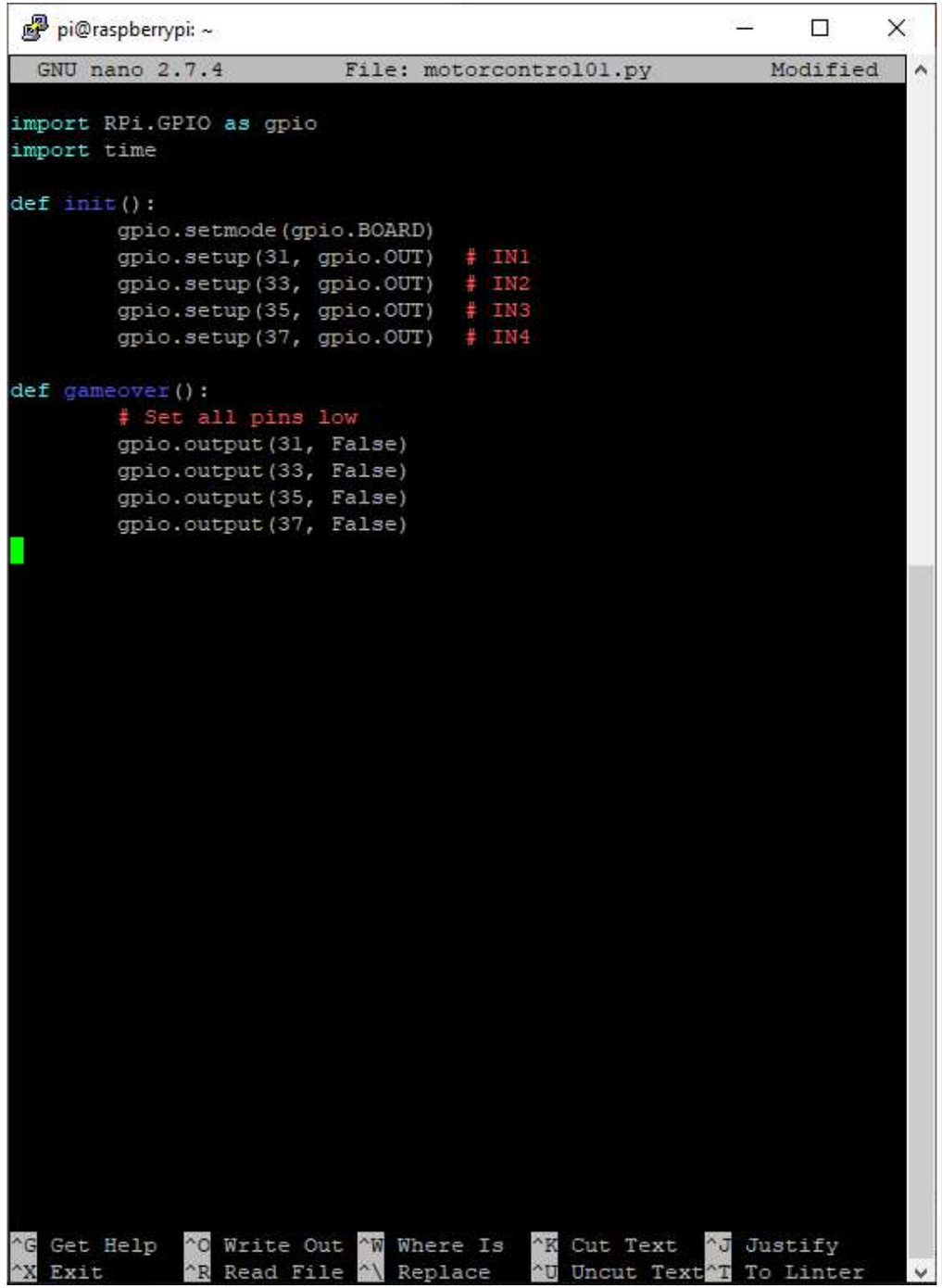
Teleoperation

- Create initialization function *init( )*
- Set board mode
- Set pins 31 through 37 as outputs



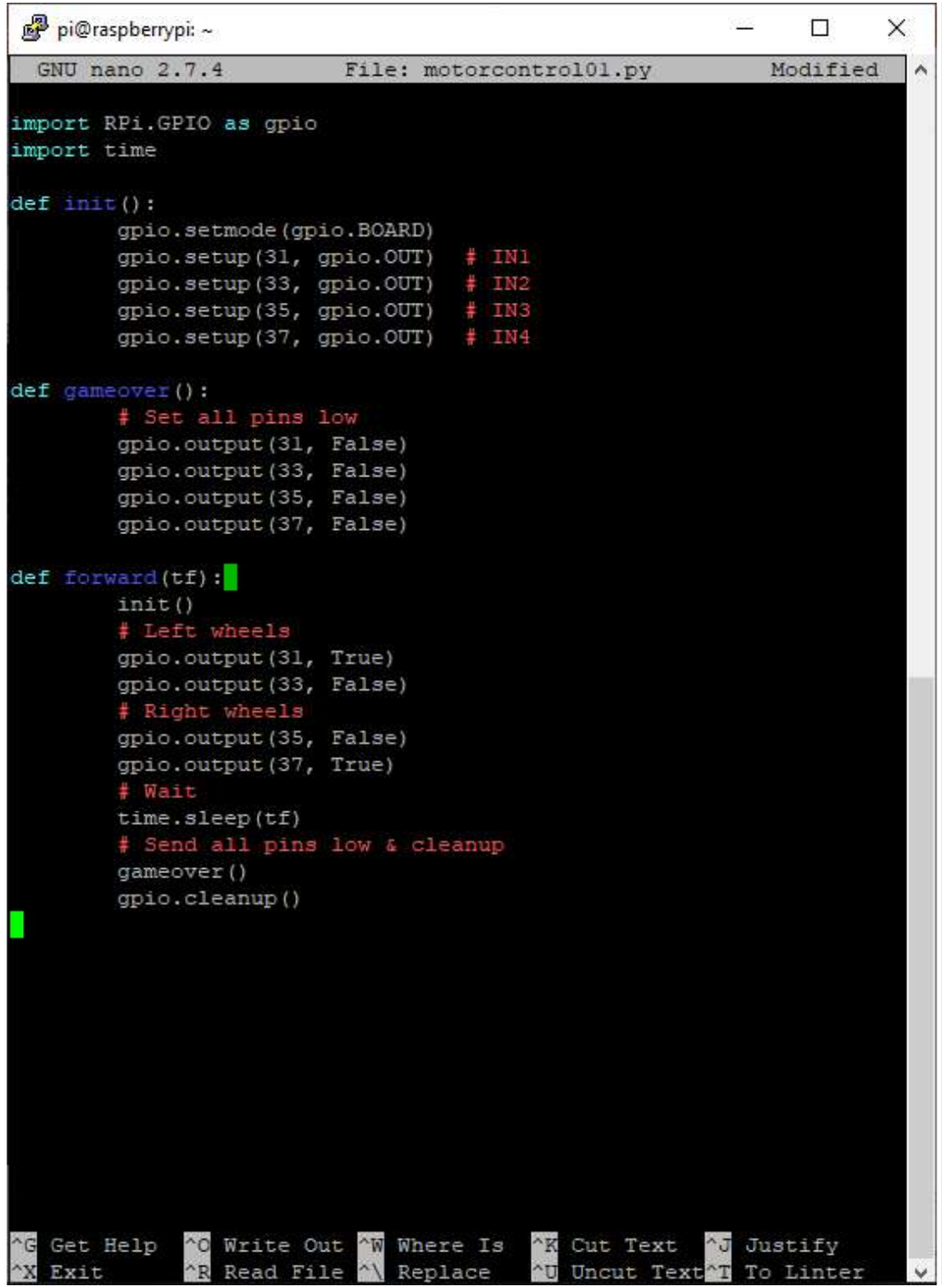
Teleoperation

- Define *gameover( )* function
- Set all pins low



Teleoperation

- Define *forward( )* function
- Function takes as an input the length of time **tf** to run function
- Once complete, set all pins low & clean up gpio



Teleoperation

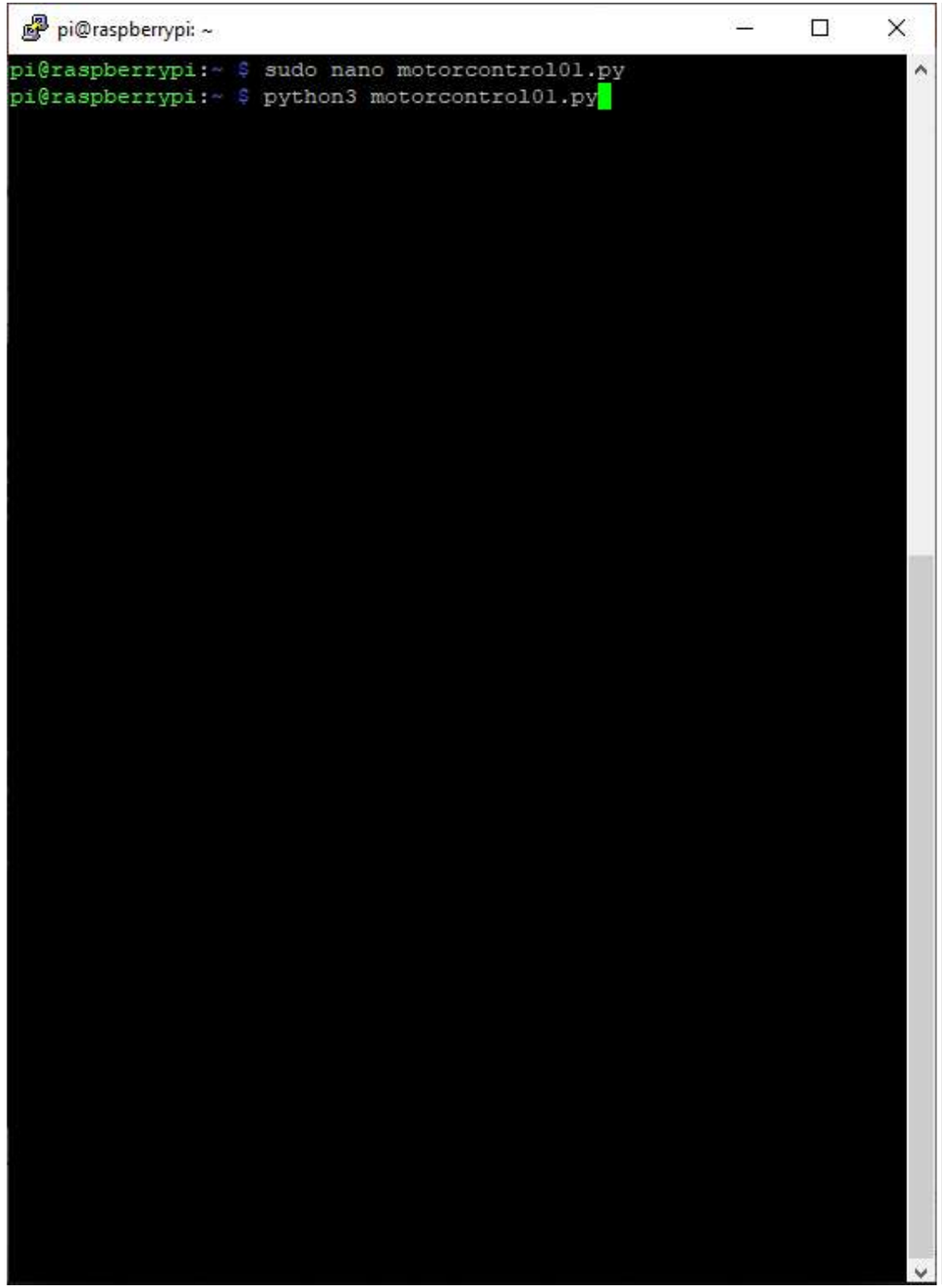
- Call *forward( )* function
- Requires time input in seconds

pi@raspberrypi: ~

```
 GNU nano 2.7.4          File: motorcontrol01.py          Modified

import RPi.GPIO as gpio
import time

def init():
        gpio.setmode(gpio.BOARD)
        gpio.setup(31, gpio.OUT)   # IN1
        gpio.setup(33, gpio.OUT)   # IN2
        gpio.setup(35, gpio.OUT)   # IN3
        gpio.setup(37, gpio.OUT)   # IN4

def gameover():
        # Set all pins low
        gpio.output(31, False)
        gpio.output(33, False)
        gpio.output(35, False)
        gpio.output(37, False)

def forward(tf):
        init()
        # Left wheels
        gpio.output(31, True)
        gpio.output(33, False)
        # Right wheels
        gpio.output(35, False)
        gpio.output(37, True)
        # Wait
        time.sleep(tf)
        # Send all pins low & cleanup
        gameover()
        gpio.cleanup()

forward(2)


^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Linter
```

Teleoperation

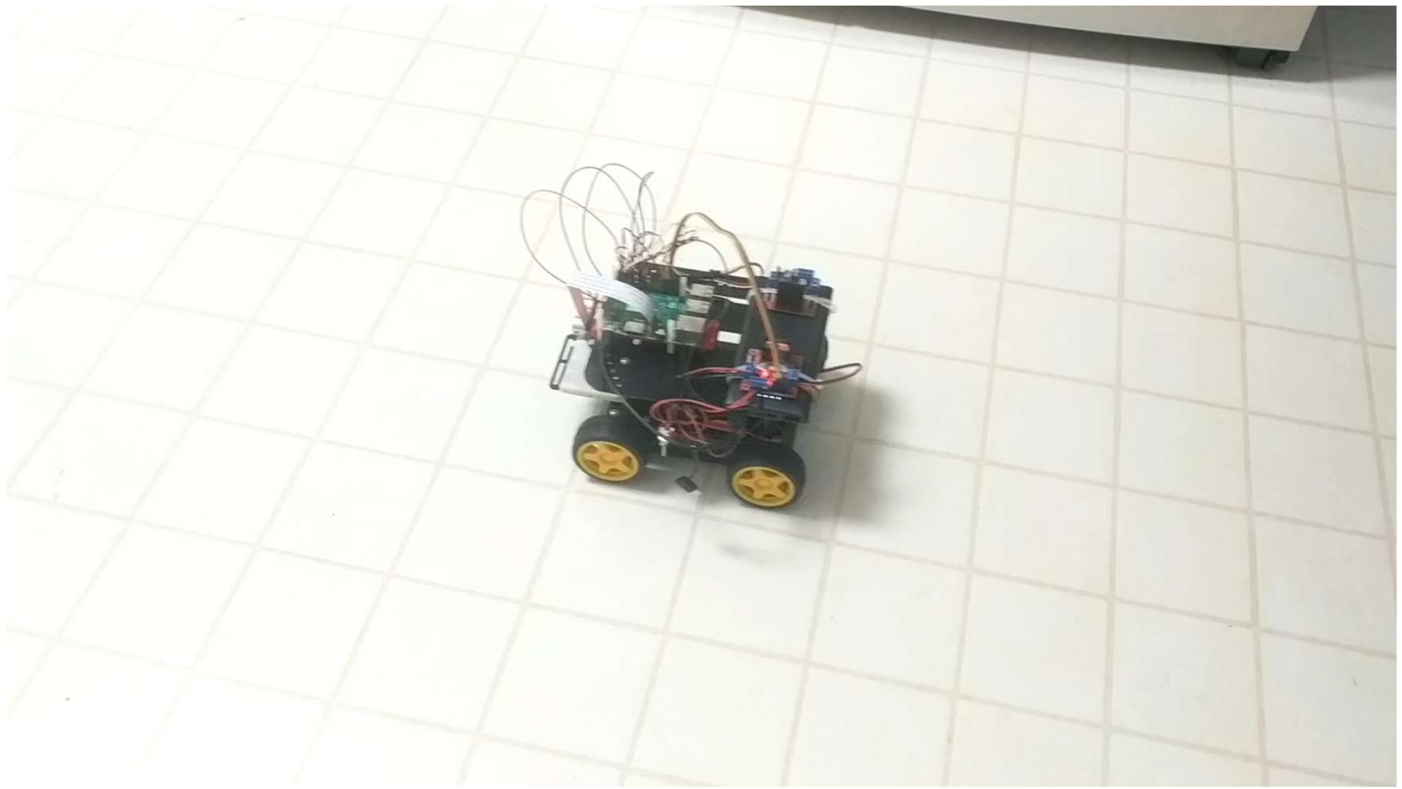
- Run program



In-Class Exercise

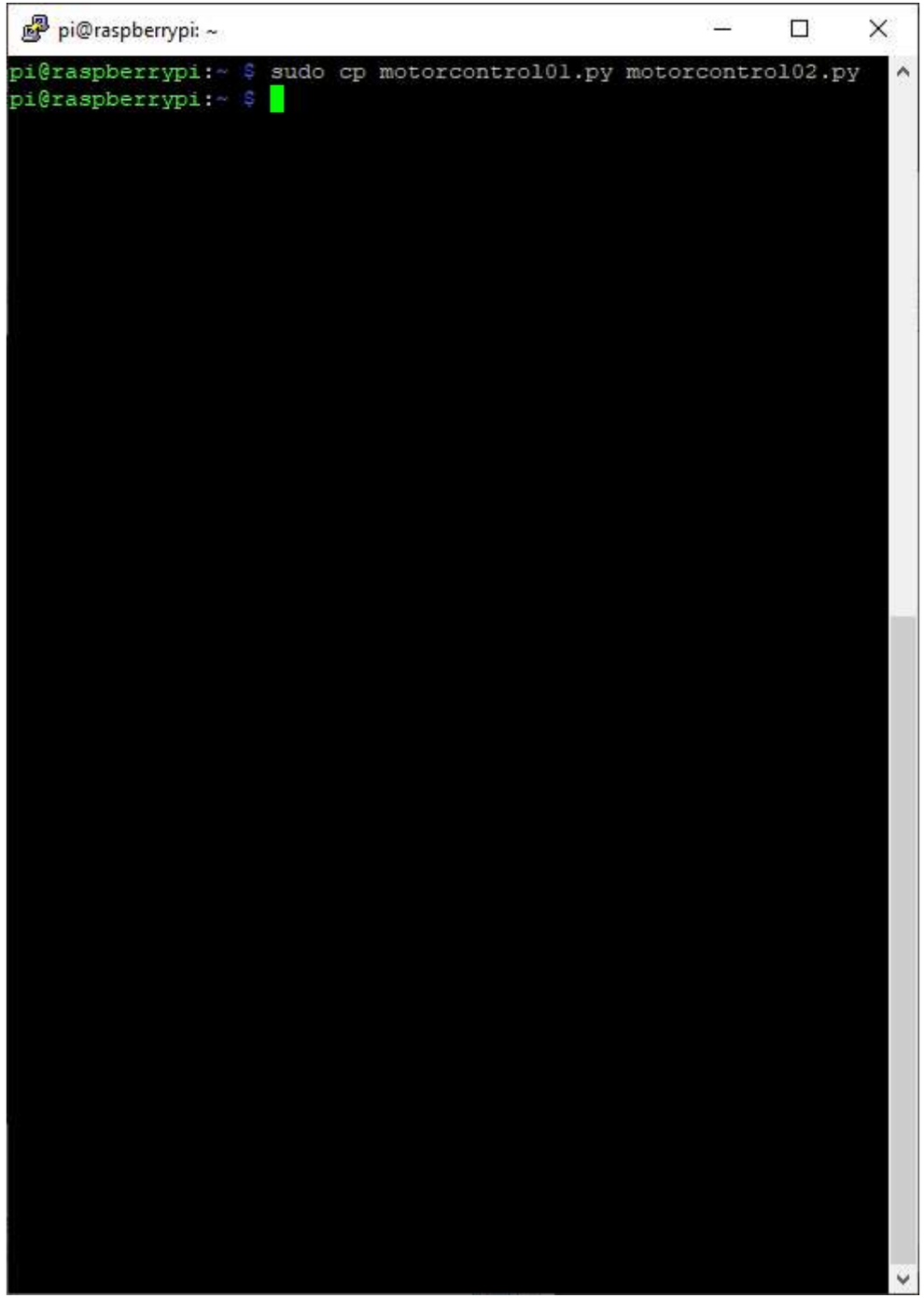
- Update the motorcontrol01.py script to:

1. Drive forward
2. Drive in reverse
3. Pivot left
4. Pivot right



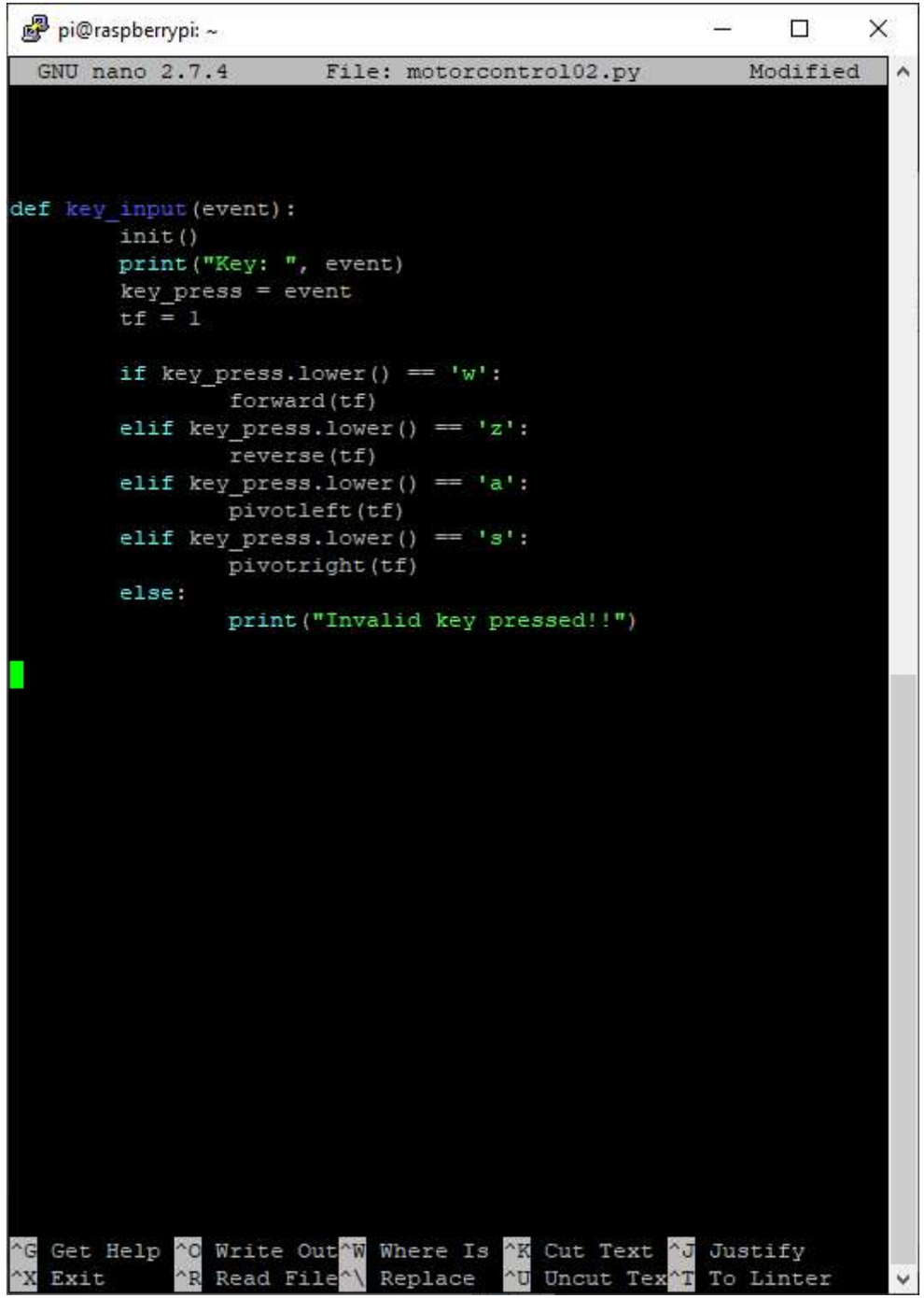
Teleoperation

- Copy Python script over to new file: ***motorcontrol02.py***



Teleoperation

- After forward( ), reverse( ), pivotleft( ), and pivotright( ) functions…
- Define *key_input( )* function
- Takes user-defined input from keyboard
- Assigns driving direction based on user input



Teleoperation

- Setup while True loop

pi@raspberrypi: ~

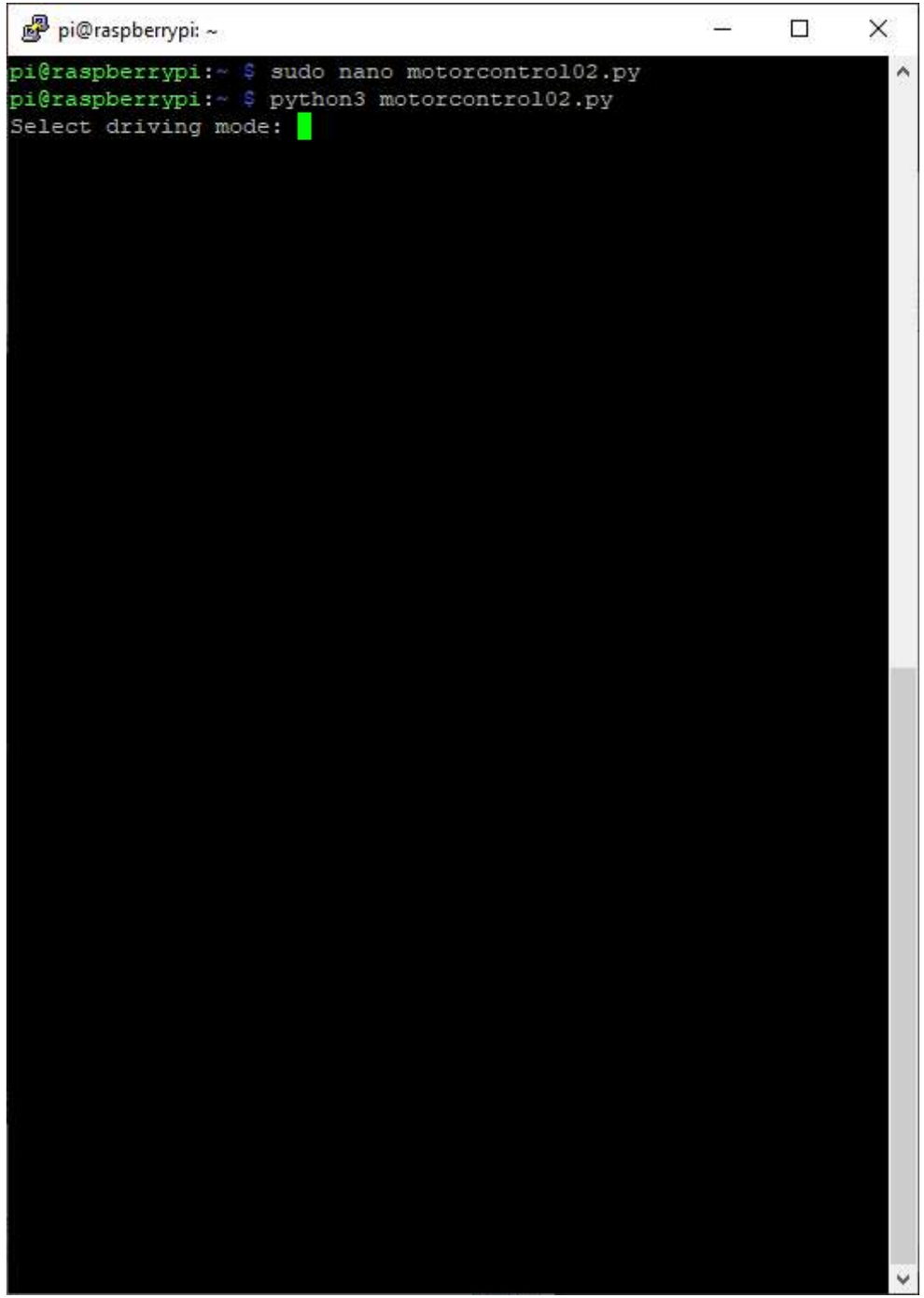
GNU nano 2.7.4 File: motorcontrol02.py Modified

```
def key_input(event):
        init()
        print("Key: ", event)
        key_press = event
        tf = 1

        if key_press.lower() == 'w':
                forward(tf)
        elif key_press.lower() == 'z':
                reverse(tf)
        elif key_press.lower() == 'a':
                pivotleft(tf)
        elif key_press.lower() == 's':
                pivotright(tf)
        else:
                print("Invalid key pressed!!")

while True:
        key_press = input("Select driving mode: ")
        if key_press == 'p':
                break
        key_input(key_press)
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Uncut Tex ^T To Linter

Teleoperation

- **Mount robot on top of box** such that wheels can spin freely
- Run program



References

- *Introduction to Autonomous Mobile Robots*, Siegwart
  - Chapter 2, 3
- *Introduction to Robotics*, Asada
  - Chapter 2
  - https://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/chapter2.pdf