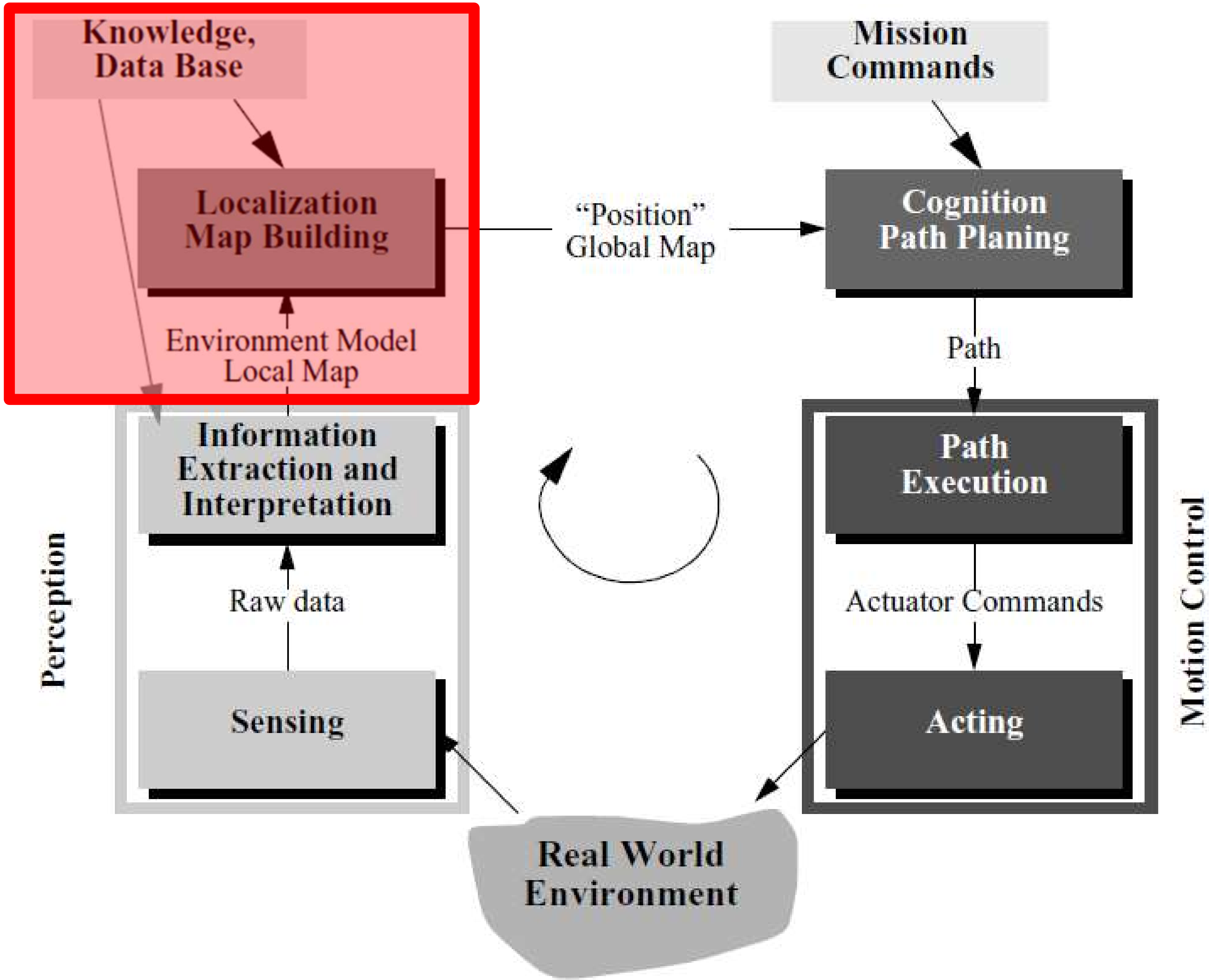
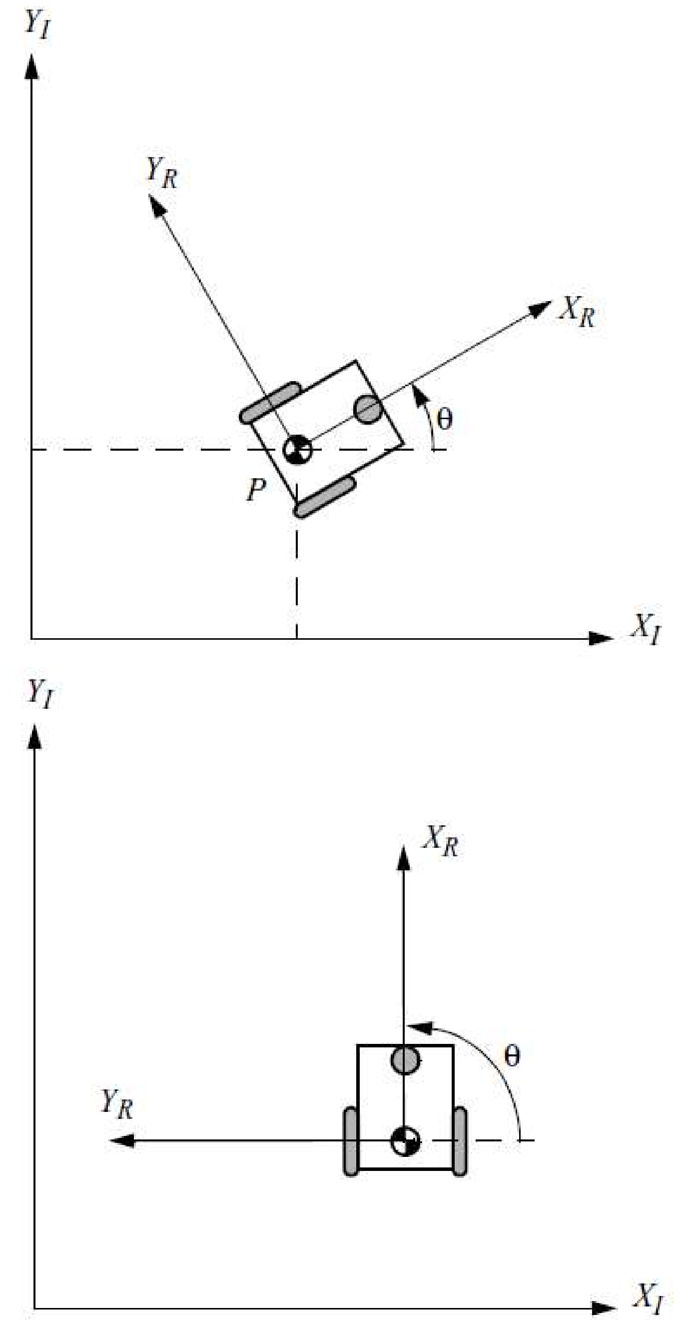
ENPM 809T

UMCP, Mitchell

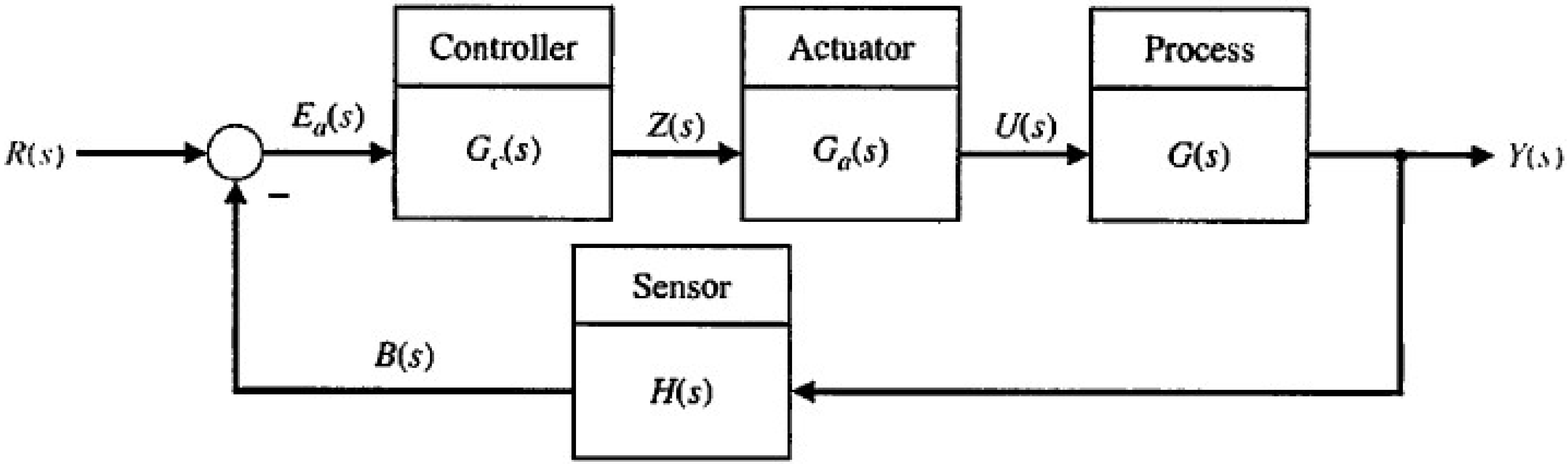

Knowledge,
Data Base
Localization
Map Building
"Position"
Global Map
Mission
Commands
Cognition
Path Planing
Environment Model
Local Map
Path
Information
Extraction and
Interpretation
Path
Execution
Perception
Raw data
Actuator Commands
Motion Control
Sensing
Acting
Real World
Environment

Kinematics

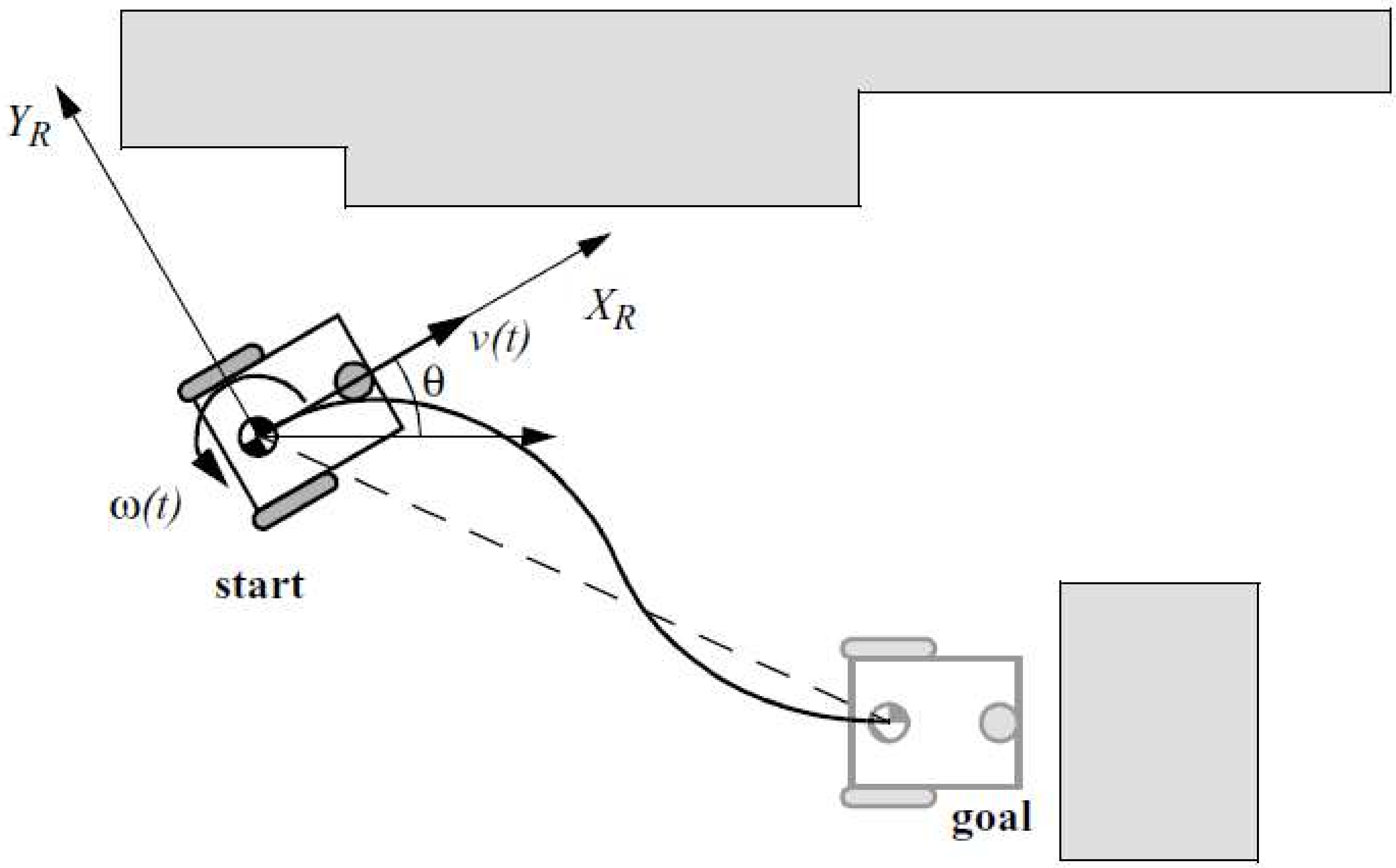
- Study of how mechanical systems behave
- Mobile robotics
  - Design for specific tasks
  - Create control software to drive hardware
- Global reference frame & robot local reference frame



Control Theory

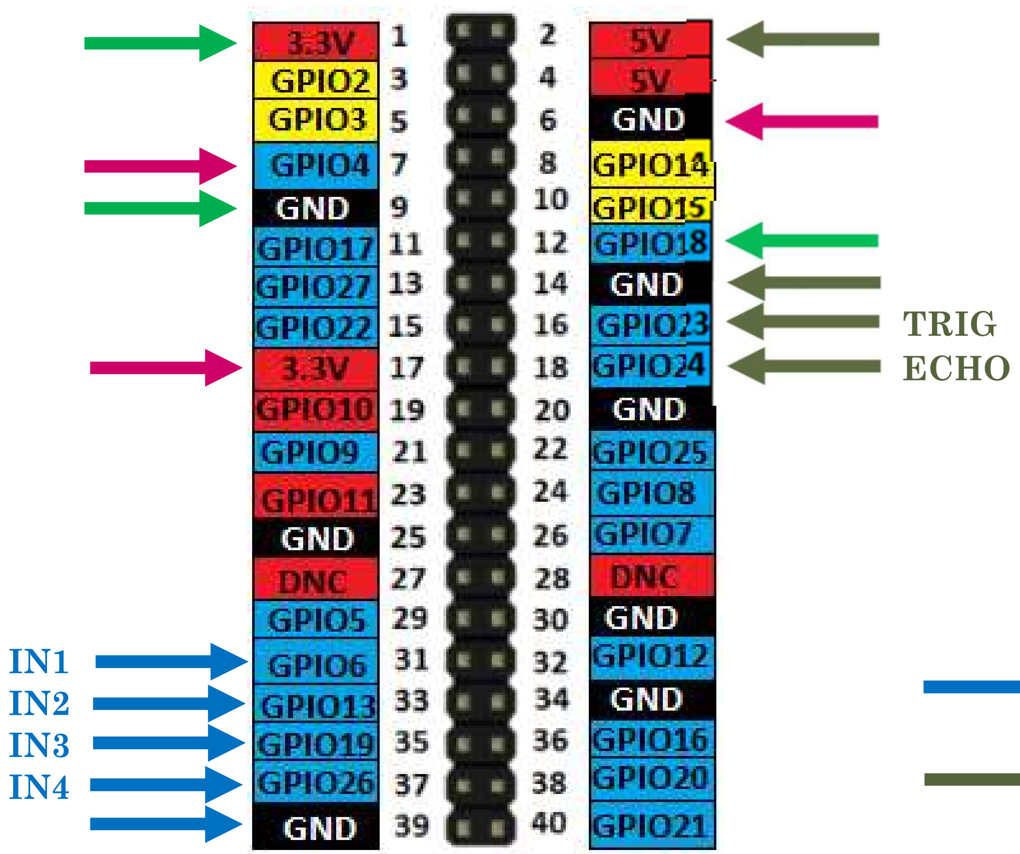


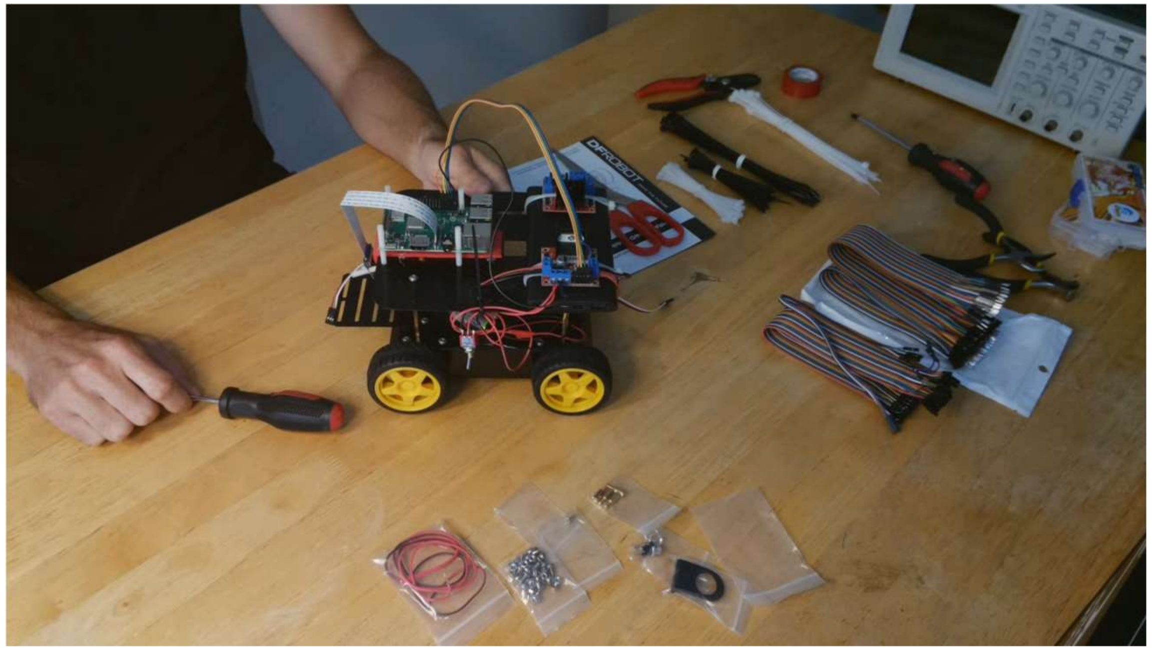
Kinematics & Localization



Assembly

- Confirm proper wiring of motor encoders to Raspberry Pi





RPi to H-bridge

Distance sensor

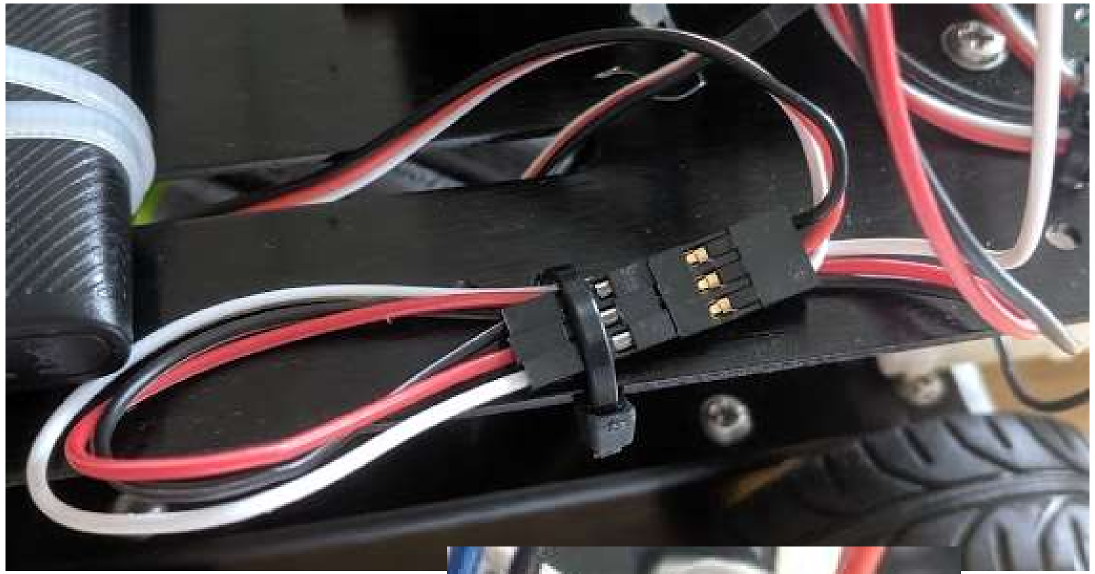
Servo Motor

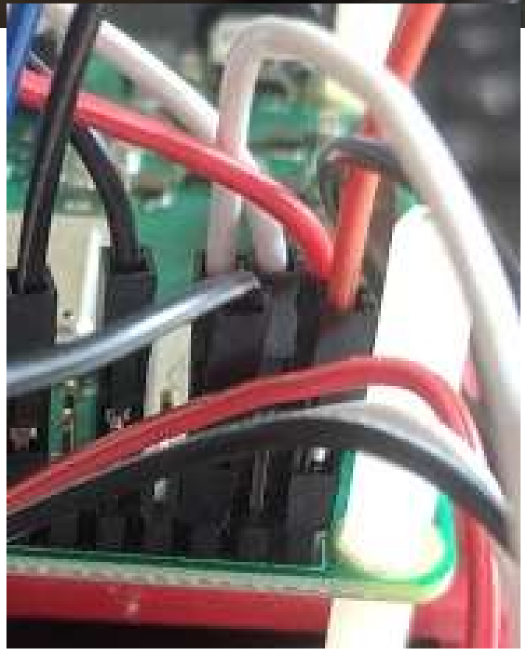
Right encoder

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





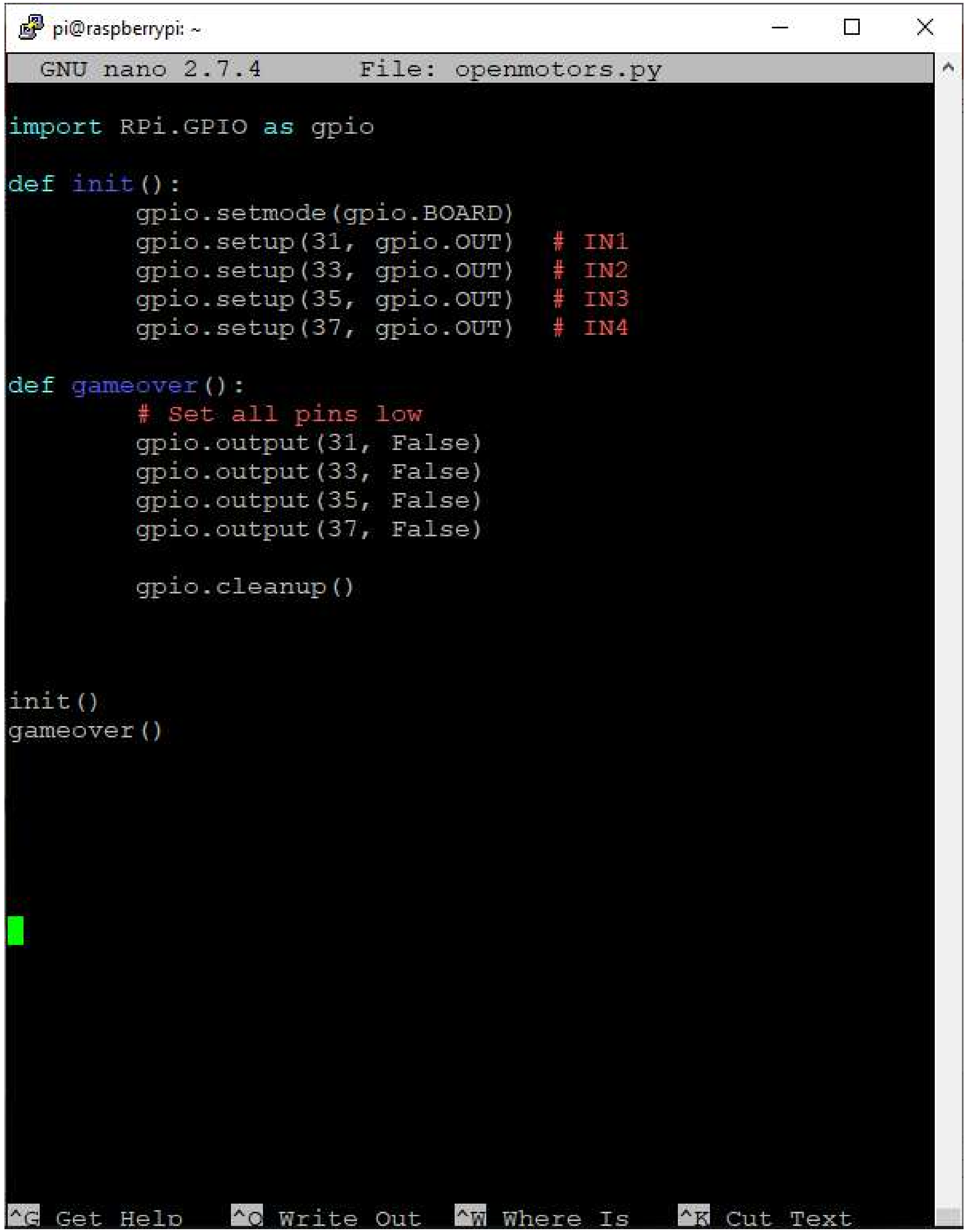
* DISCLAIMER *

- It is **vitally important** to be careful NOT to plug in any 5V pins!
- …sending 5V to the 3.3V GPIO pins will ***permanently damage*** your RPi!!

- In general, recommend transferring/backing up all .py files to your laptop before each lecture

Assembly

- **Double check all electrical connections!**
- Power Raspberry Pi via USB battery pack
- Connect to Raspberry Pi using Putty/Terminal and VNC
- Run ***openmotors.py*** prior to applying power to the H-bridge



pi@raspberrypi: ~

GNU nano 2.7.4 File: openmotors.py

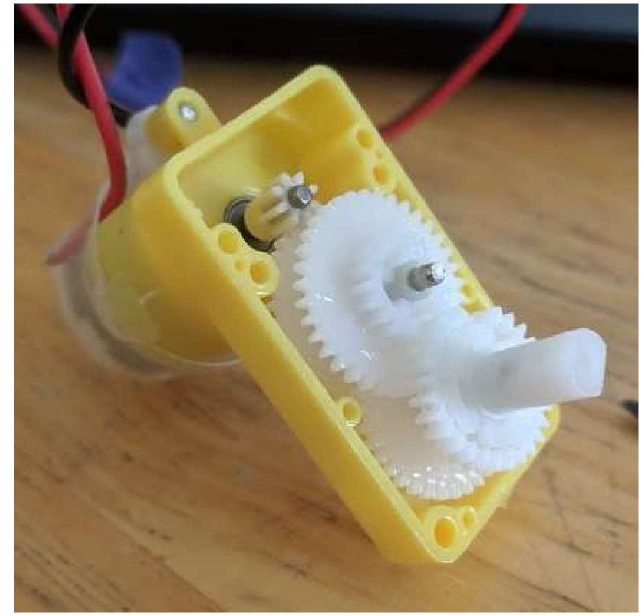
```
import RPi.GPIO as gpio

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)

        gpio.cleanup()


init()
gameover()
```

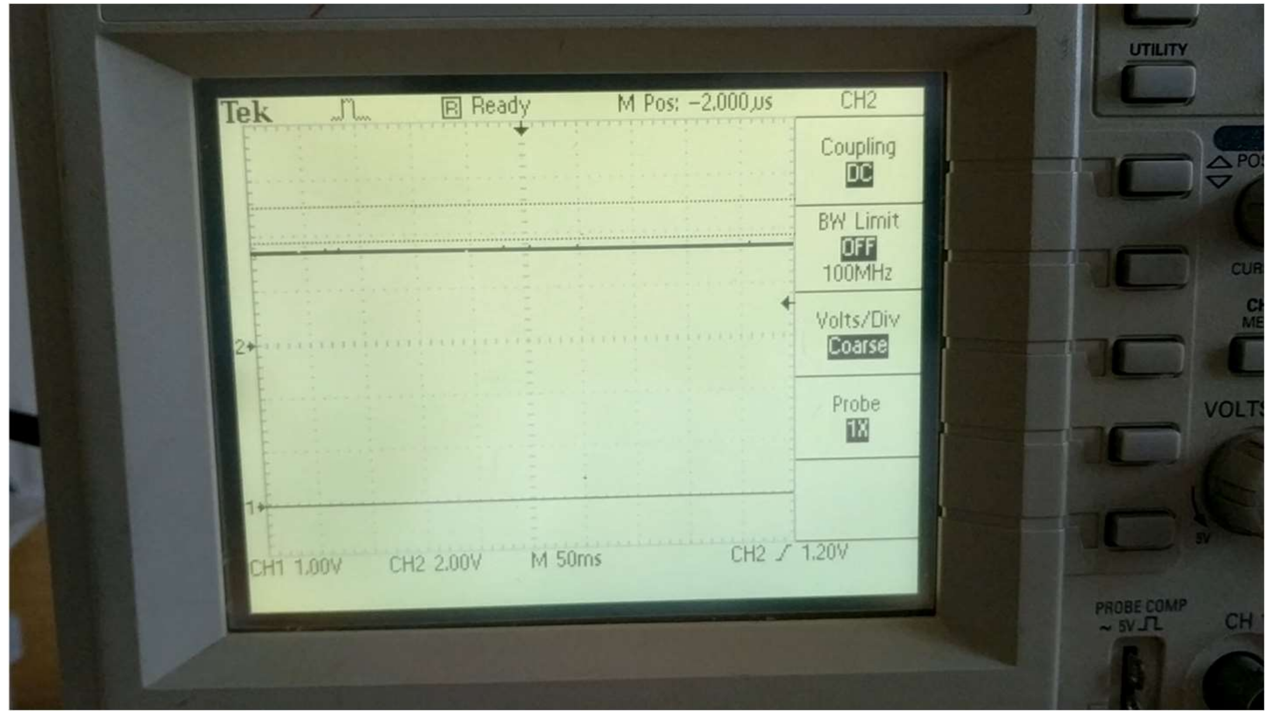
^G Get Help ^O Write Out ^W Where Is ^K Cut Text

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

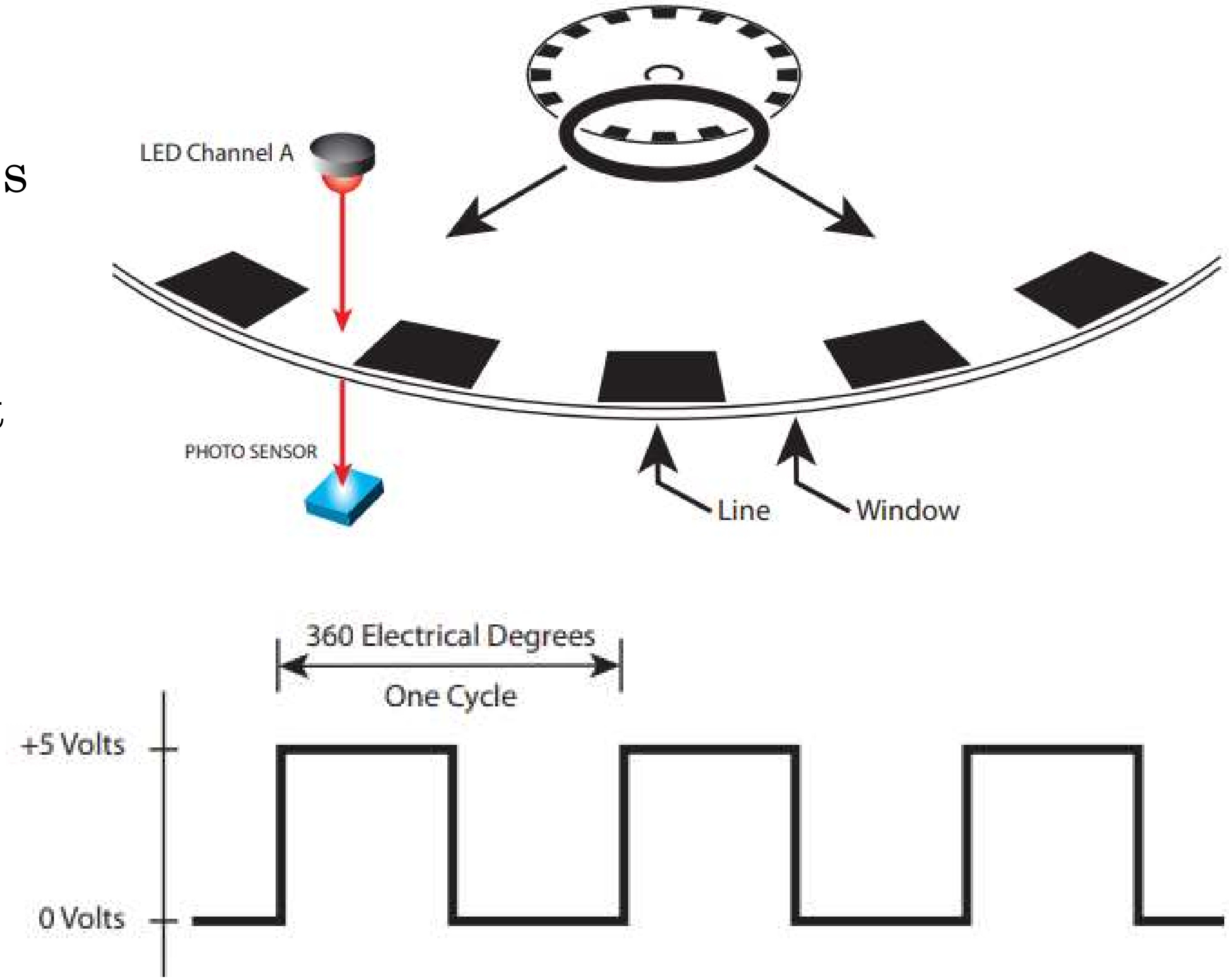




https://www.pololu.com/product/2590

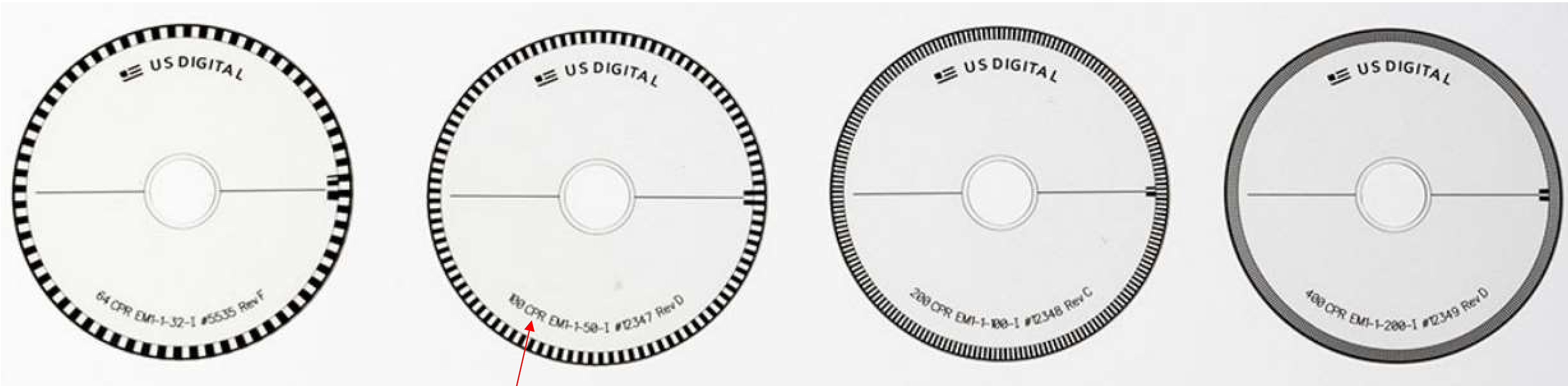
Encoders

- Resolution: number of times the output signal goes high per revolution
- Lines & windows represent *relative* positions on disk



https://cdn.usdigital.com/assets/general/usd-encoder-precision.pdf

Encoders

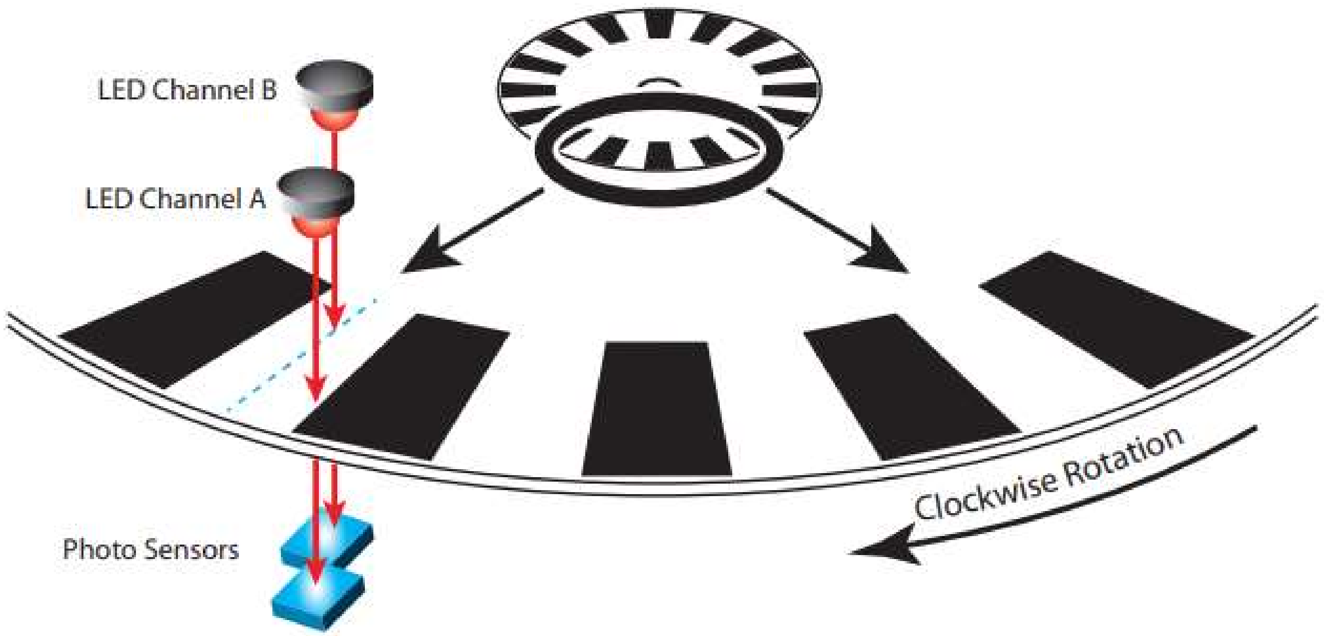


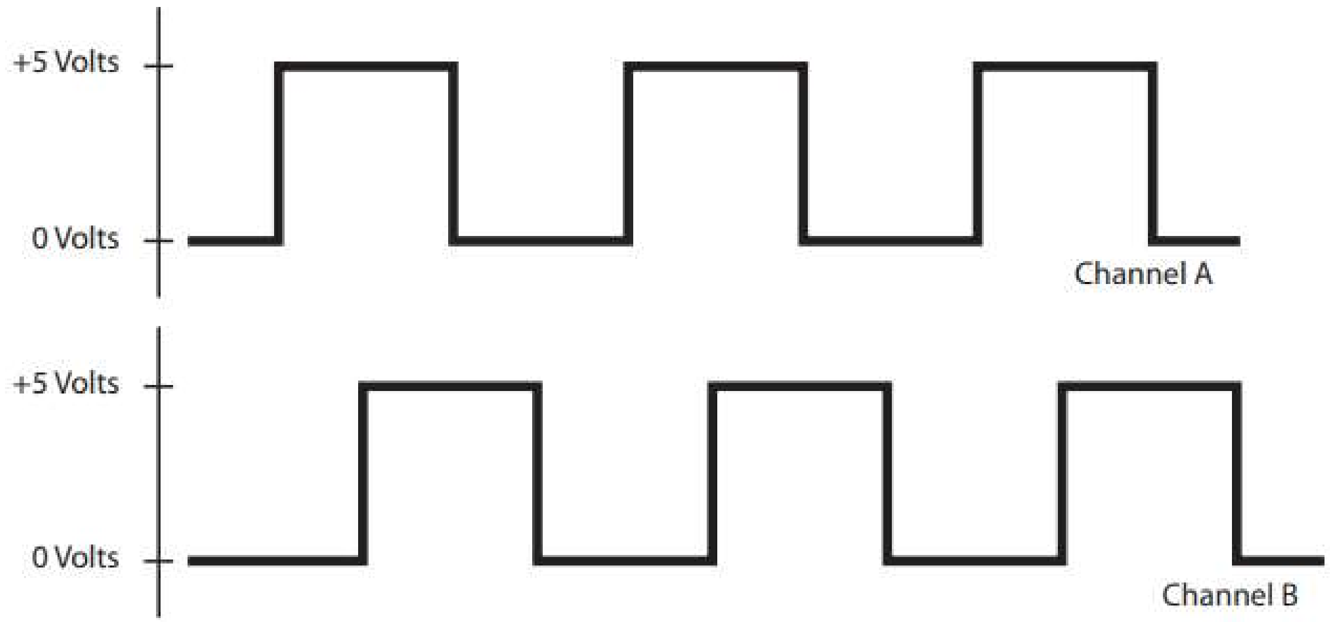
Cycles per Revolution (CPR): number of line/window pairs

Encoders

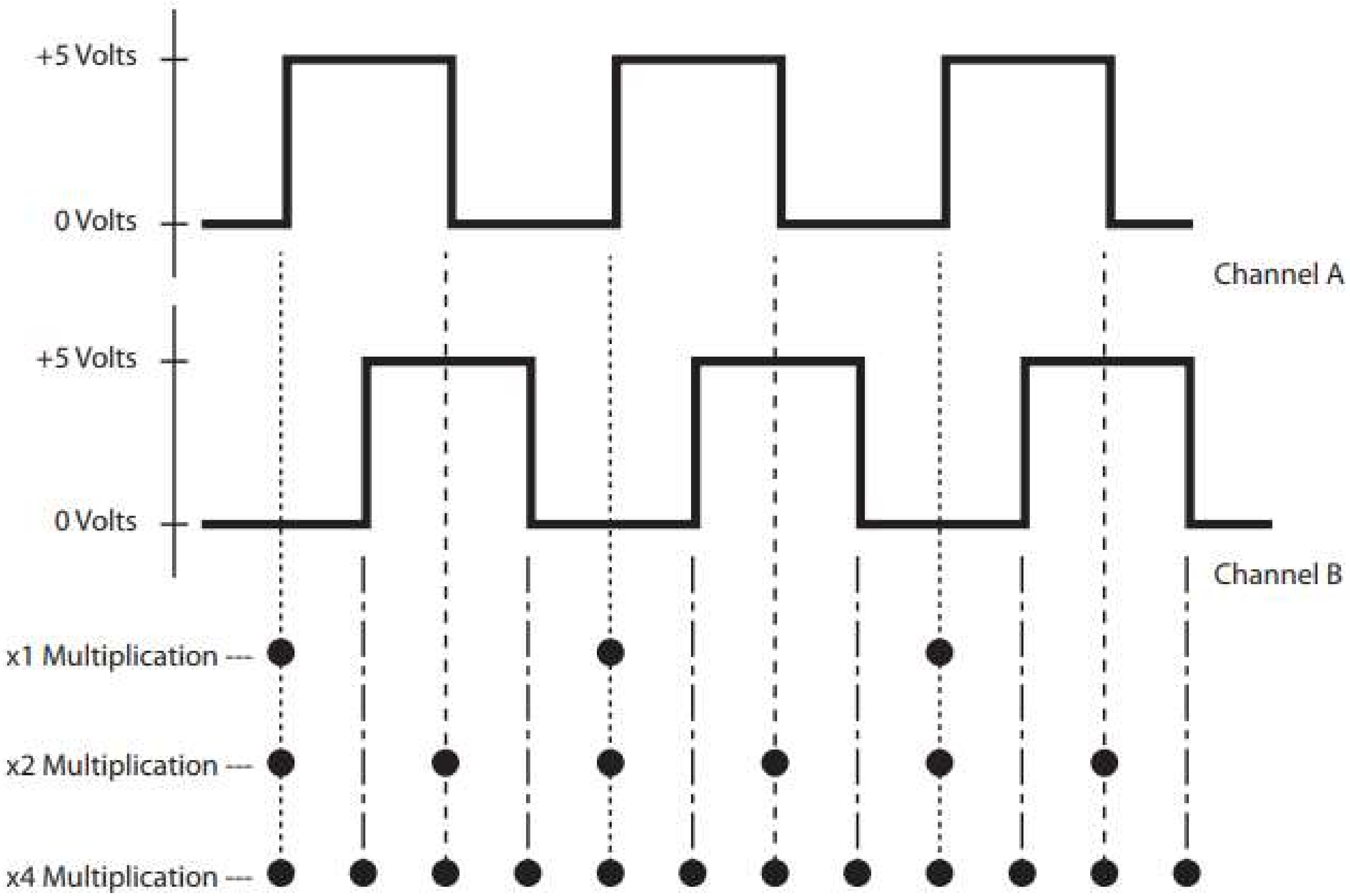
- Quadrature: additional LED & photosensor displaced by 90-deg electrically

1. Direction
2. Resolution increased





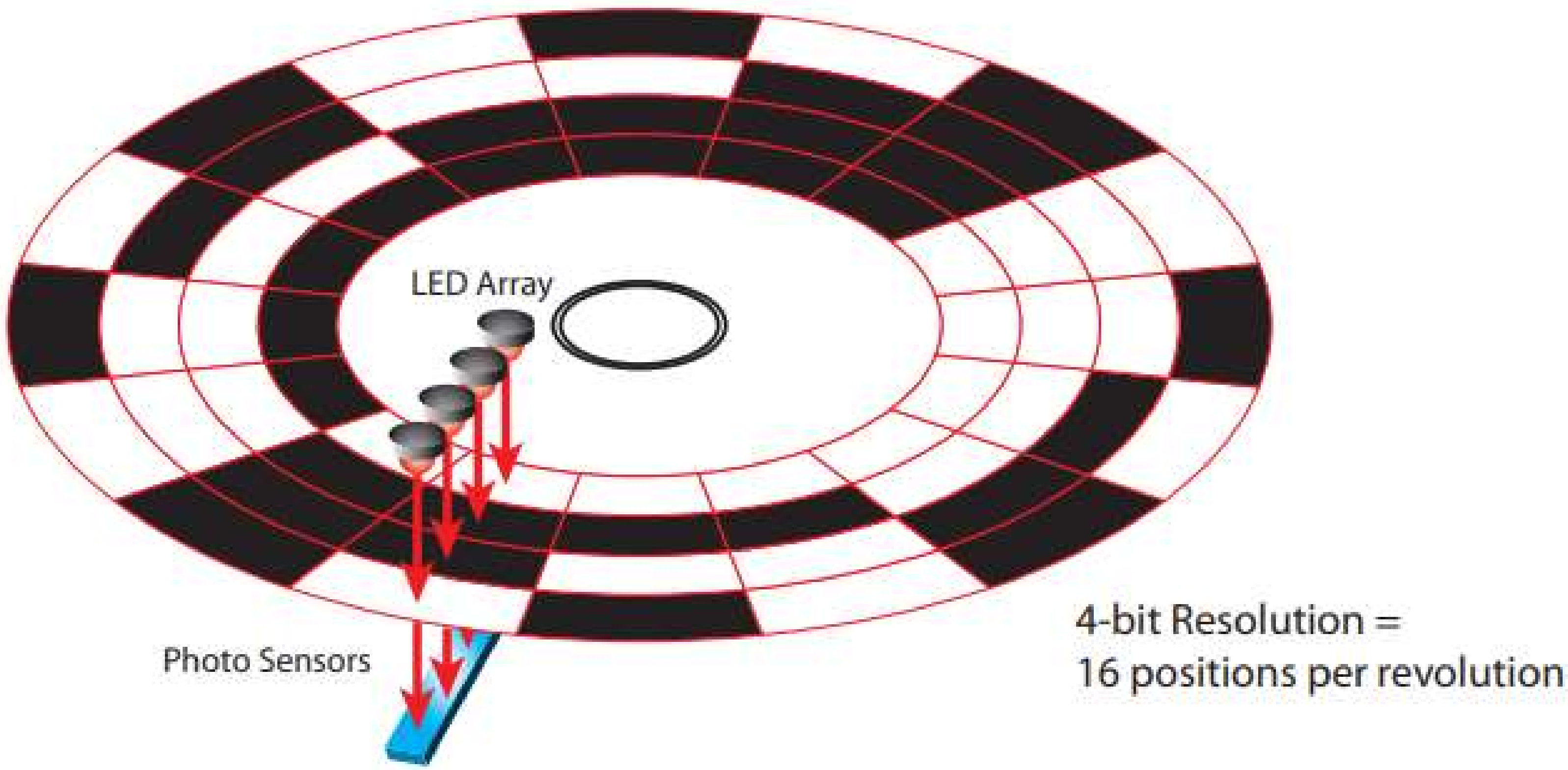
Encoders



+5 Volts
0 Volts
Channel A
+5 Volts
0 Volts
Channel B
x1 Multiplication ---
x2 Multiplication ---
x4 Multiplication ---

Encoders

- Absolute: output unique code for each position on disk
- Resolution defined here in terms of bits



Resolution in Bits	Positions per Revolution	Degrees of Rotation
8-bit resolution	256 positions	1.41° of rotation per position
10-bit resolution	1,024 positions	0.35° of rotation per position
12-bit resolution	4,096 positions	0.09° of rotation per position

Encoders

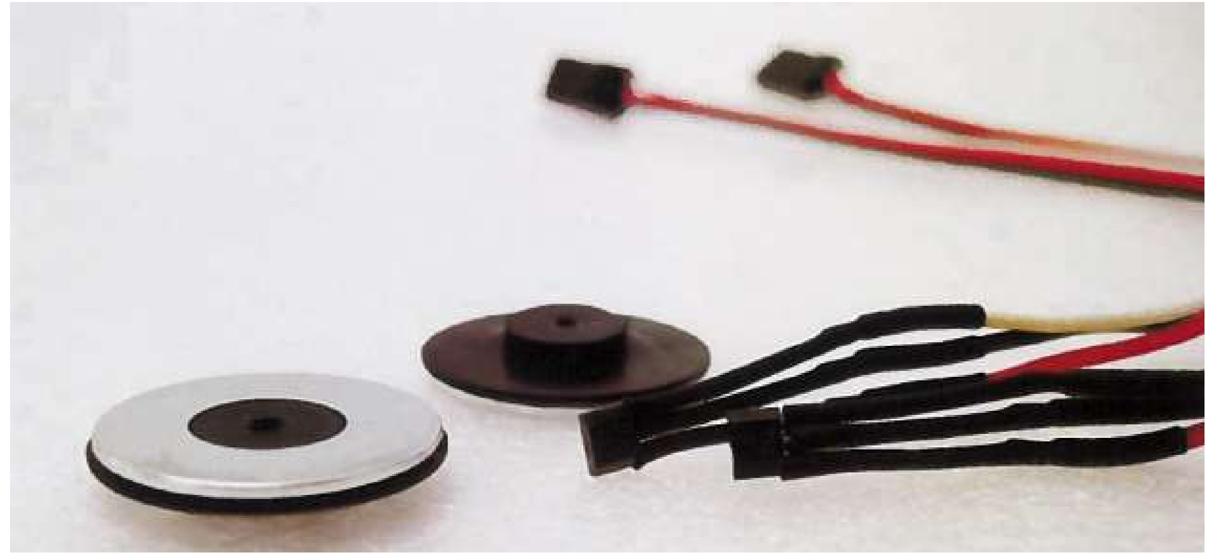
- Neodymium 8-pole magnet
  - 4 north poles
  - 4 south poles
- Hall effect sensor

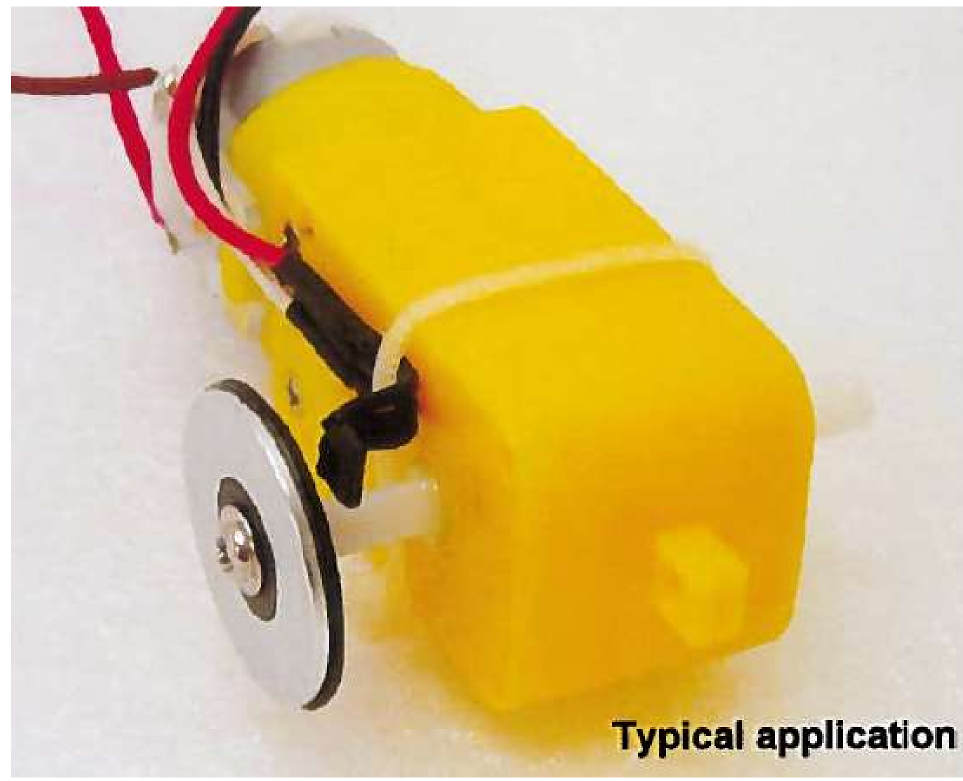
Specifications:

Supply voltage:	3V-24V
Supply current:	4mA per sensor
Output voltage:	26V maximum
Output current:	25mA continuous
Output type:	Open drain

Wiring:

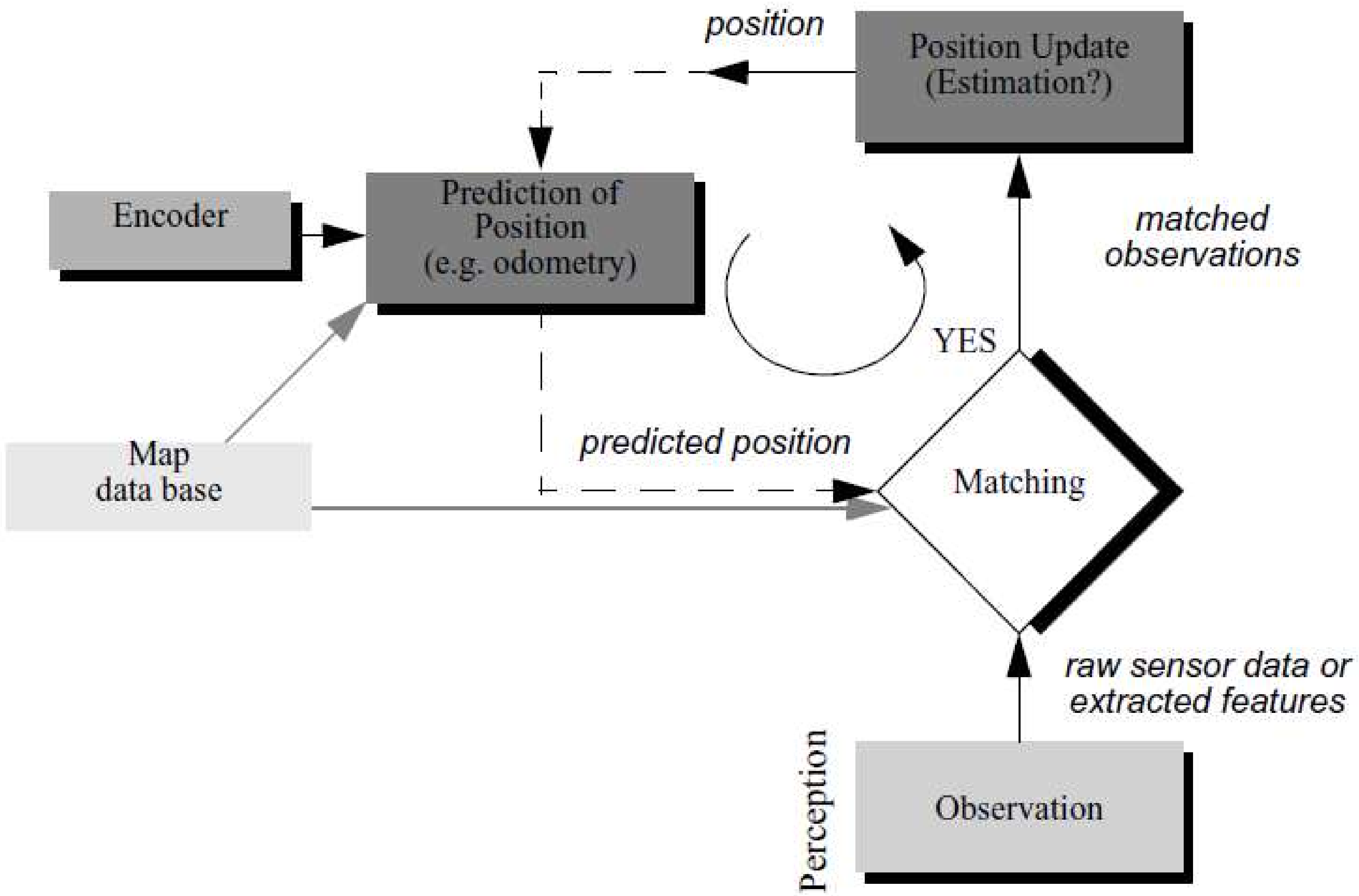
White wire:	Output
Red wire:	+3V to 24V
Black wire:	Ground (0V)



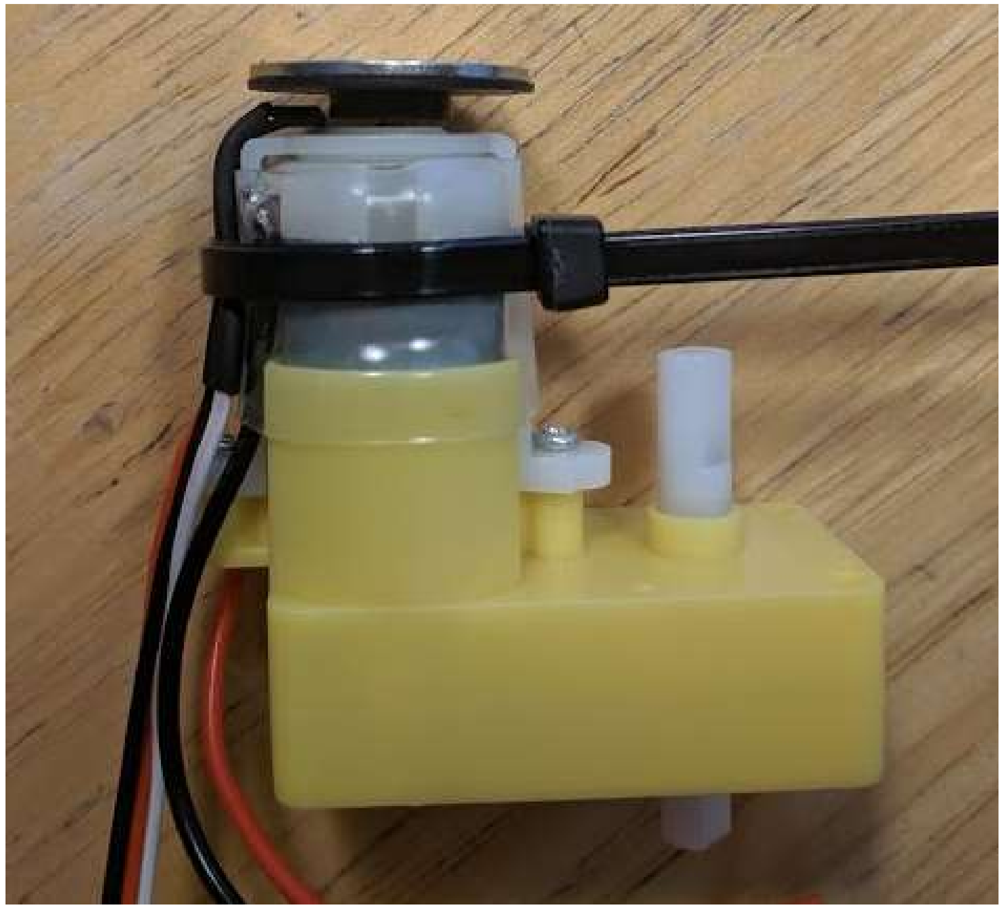


https://cdn.sparkfun.com/datasheets/Robotics/multi-chassis%20encoder001.pdf

Encoders & Localization

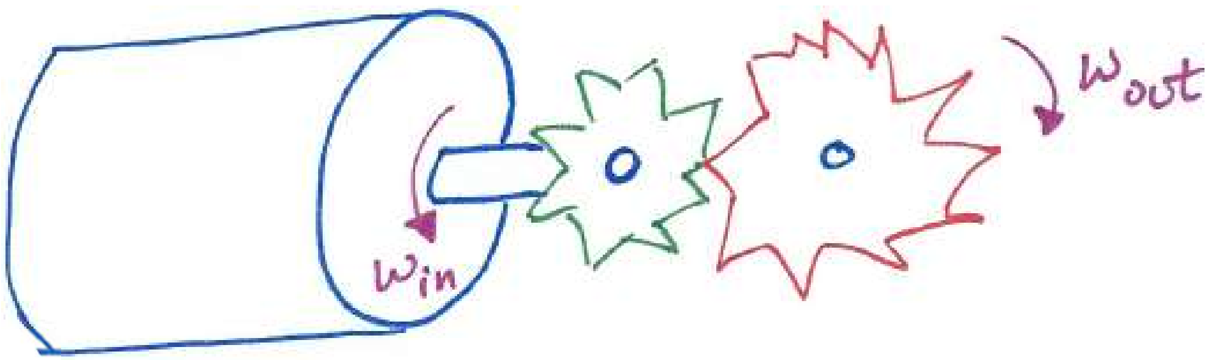


Encoders



https://www.youtube.com/watch?v=oLBYHbLO8W0

DC motor gearing

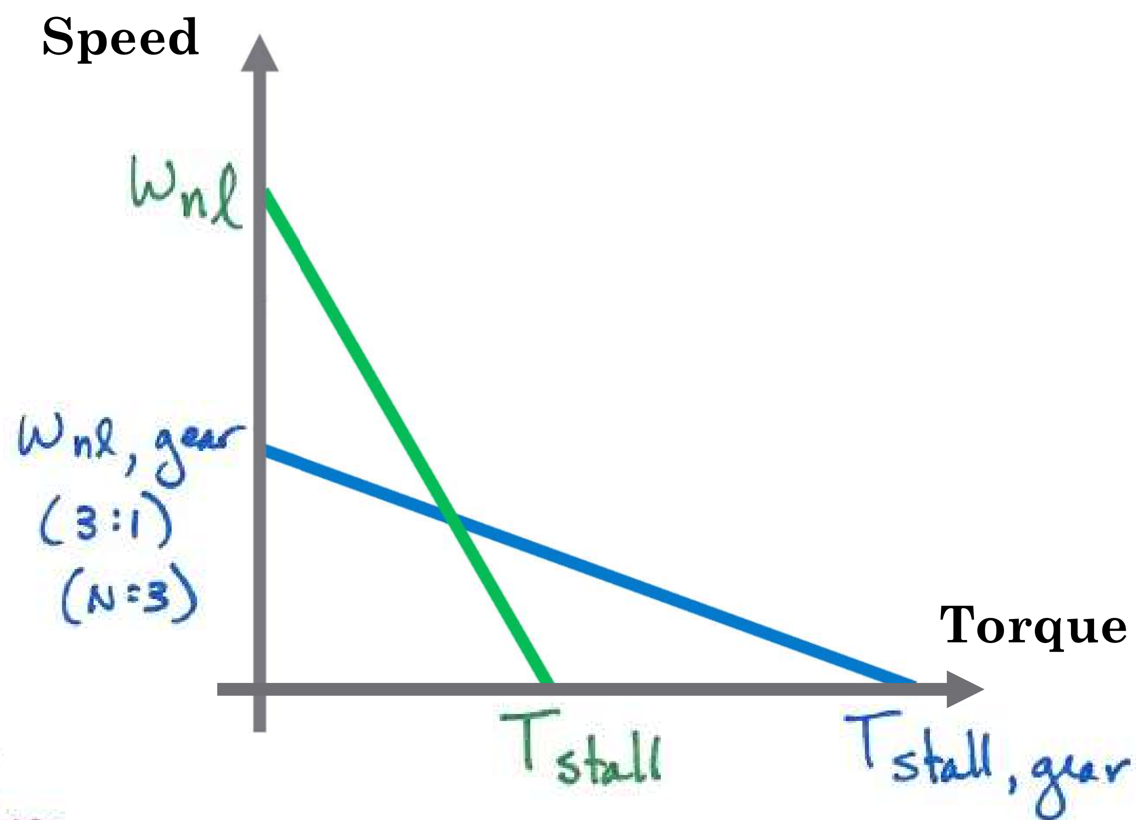


Gear Ratio

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

$\omega_{nl} = 1000$ rpm

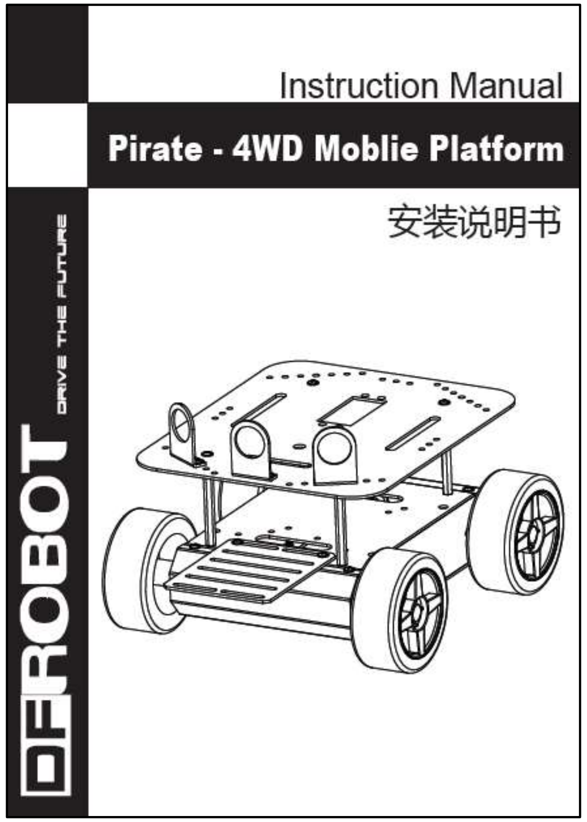
10:1 gear ratio $(N=10)$

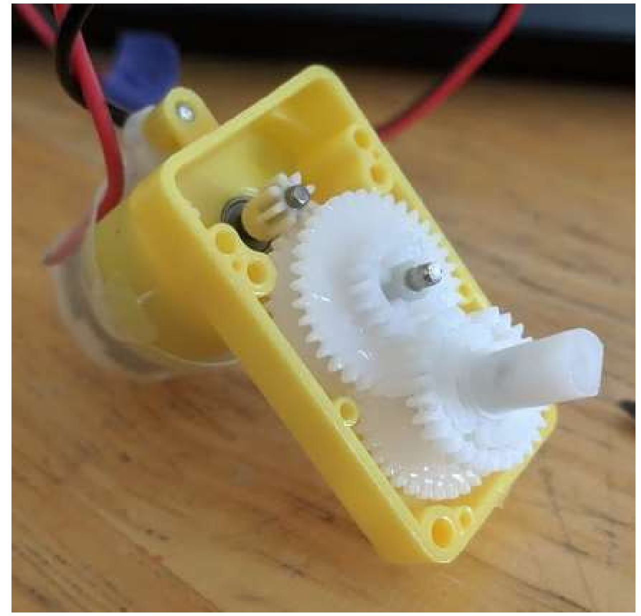


$\eta_{gearbox} = \frac{T_{out}\,\omega_{out}}{T_{in}\,\omega_{in}}$

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

DC motor gearing



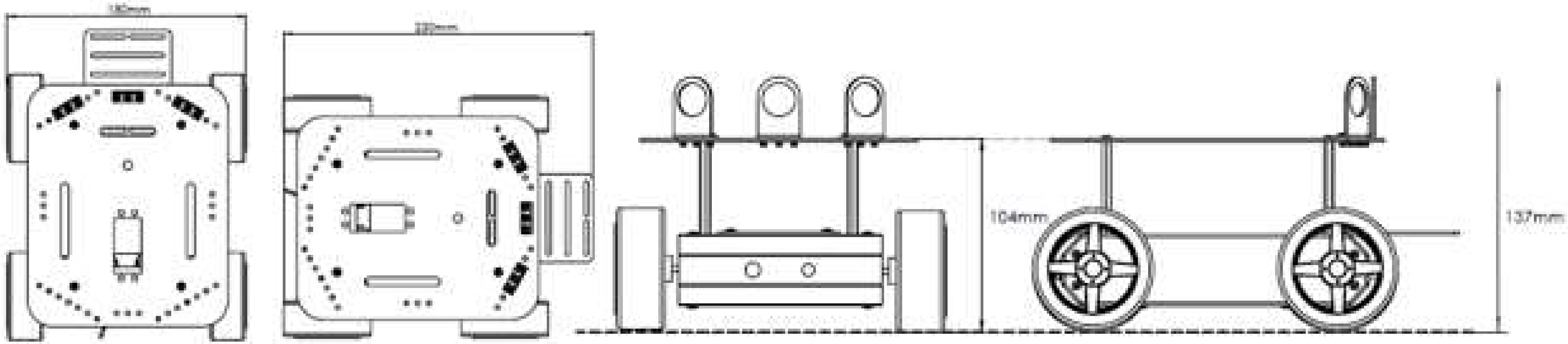


Motor Feature :

- 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
- Torgue (6V): 1.92Kgcm
- Weight : 45g

Localization

- Motor diameter & gear ratio



Complete Machine Weight: 710g
Wheel Diameter : 65mm
Highest Speed: 61cm/s

Motor Feature :

- Gear Ratio : 1:120

Localization

- Motor revolutions required to drive **1 meter in a straight line**:

$$\left(\frac{120\ motor\ rev}{1\ wheel\ rev}\right)$$

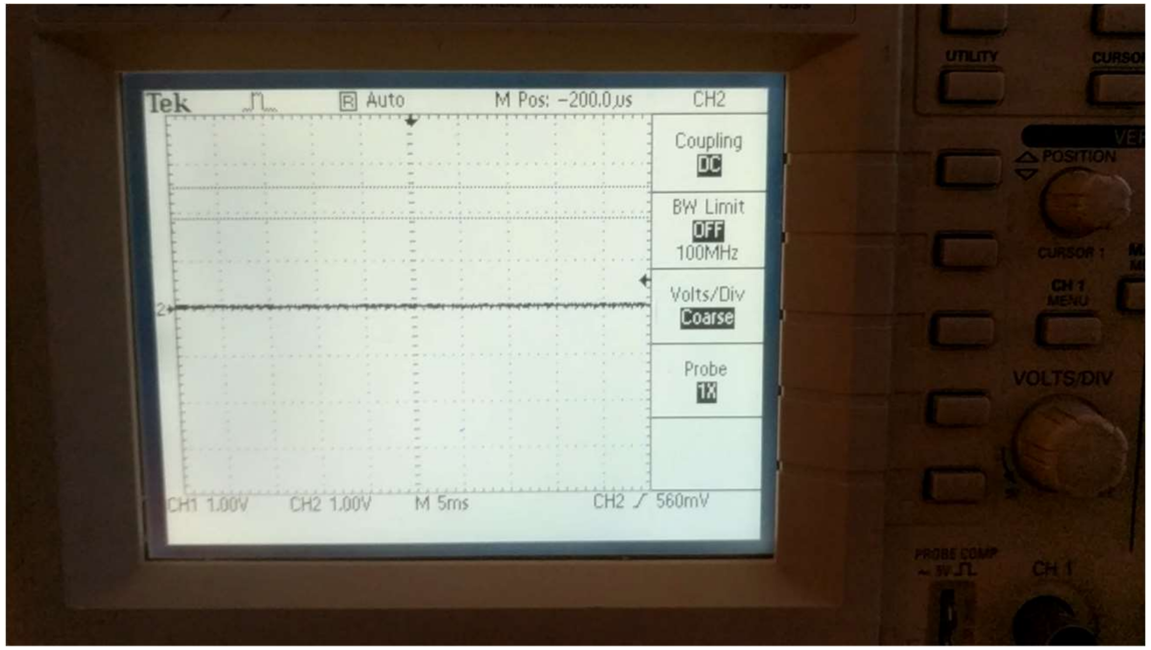
$$\left(\frac{120\ motor\ rev}{1\ wheel\ rev}\right)\left(\frac{1\ wheel\ rev}{2\pi r\ meter}\right)$$

$$\left(\frac{120\ motor\ rev}{1\ wheel\ rev}\right)\left(\frac{1\ wheel\ rev}{2\pi(0.0325)\ meter}\right)(1\ meter) = \boldsymbol{587.6\ motor\ rev}$$

Pirate

Localization

- Encoder ticks required to drive **2 meters in a straight line**:



$$\left(\frac{1\ wheel\ rev}{120\ motor\ rev}\right)$$

$$\left(\frac{1\ wheel\ rev}{120\ motor\ rev}\right)\left(\frac{1\ motor\ rev}{8\ encoder\ ticks}\right) = \left(\frac{1\ wheel\ rev}{960\ encoder\ ticks}\right) = \left(\frac{960\ encoder\ ticks}{1\ wheel\ rev}\right)$$

$$\left(\frac{1\ wheel\ rev}{2\pi(0.0325)\ meter}\right)\left(\frac{960\ encoder\ ticks}{1\ wheel\ rev}\right) = \left(\frac{4701.2\ encoder\ ticks}{meter}\right)$$

$$\left(\frac{4701.2\ encoder\ ticks}{meter}\right)(2\ meters) = \mathbf{9402.4\ encoder\ ticks}$$

Pirate

Localization

- Encoder ticks required to drive **2 meters in a straight line**:

$$\left(\frac{1\ wheel\ rev}{120\ motor\ rev}\right)$$

$$\left(\frac{1\ wheel\ rev}{20\ encoder\ ticks}\right) = \left(\frac{20\ encoder\ ticks}{1\ wheel\ rev}\right)$$

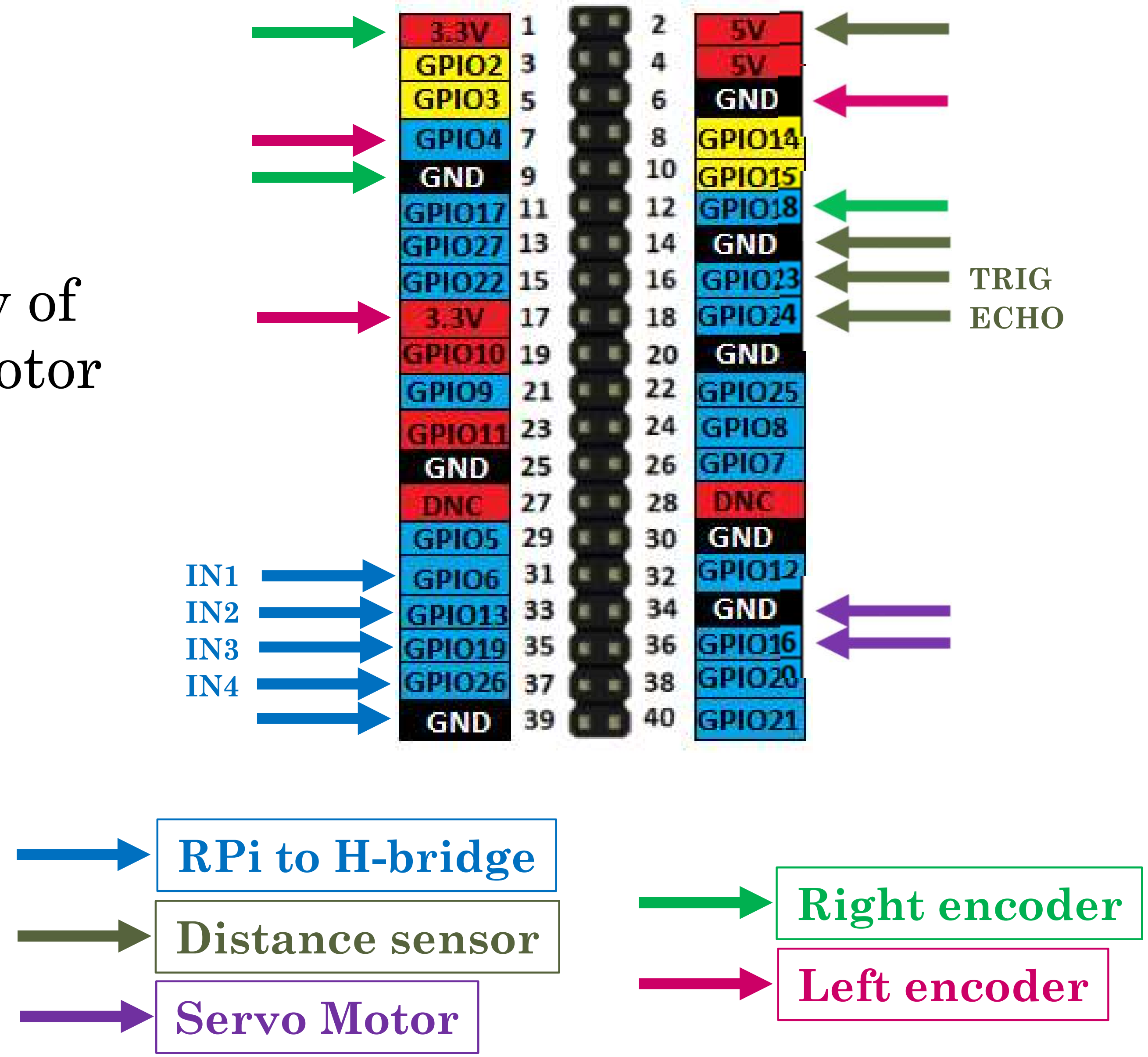
$$\left(\frac{1\ wheel\ rev}{2\pi(0.0325)\ meter}\right)\left(\frac{20\ encoder\ ticks}{1\ wheel\ rev}\right) = \left(\frac{98\ encoder\ ticks}{meter}\right)$$

$$\left(\frac{98\ encoder\ ticks}{meter}\right)(2\ meters) = \boldsymbol{196\ encoder\ ticks}$$

Baron

Teleoperation

- Import RPi.GPIO library
- Utilize PWM functionality of GPIO library to control motor speed



https://pypi.org/project/RPi.GPIO/

Tracking Encoder Counts

- Setup GPIO pins as inputs

```
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

1. Attach interrupts

```
GPIO.add_event_detect(channel, GPIO.RISING)
```

2. Polling
   - Processor intensive

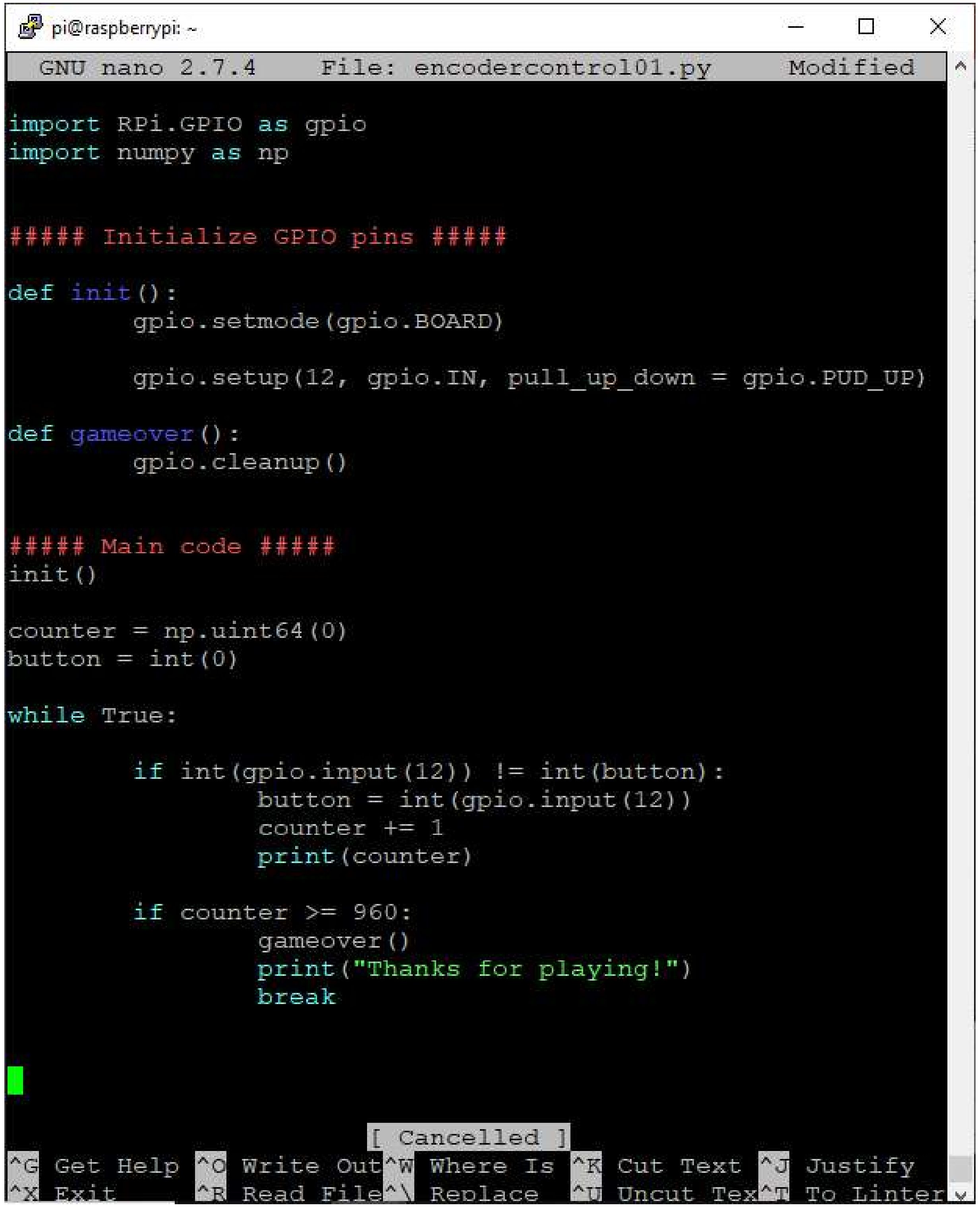
```
if GPIO.input(channel):
    print('Input was HIGH')
else:
    print('Input was LOW')
```

https://sourceforge.net/p/raspberry-gpio-python/wiki/Inputs/

Teleoperation

- Create new Python script: ***encodercontrol01.py***
- Monitor back right encoder

- Run ***encodercontrol01.py***
- Spin wheel slowly **by hand** through 1 revolution

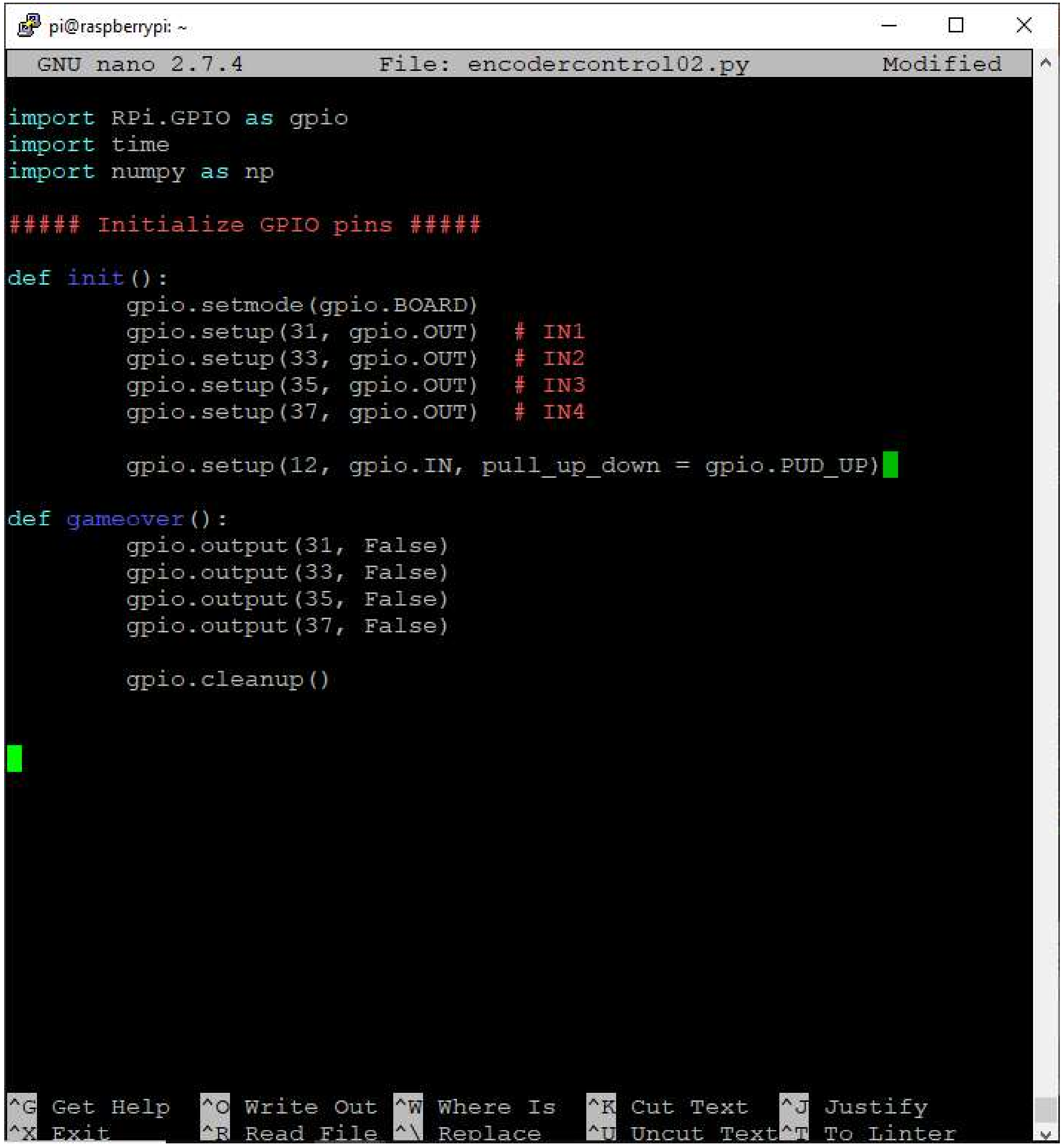


```
pi@raspberrypi: ~
  GNU nano 2.7.4        File: encodercontrol01.py        Modified

import RPi.GPIO as gpio
import numpy as np


##### Initialize GPIO pins #####

def init():
        gpio.setmode(gpio.BOARD)

        gpio.setup(12, gpio.IN, pull_up_down = gpio.PUD_UP)

def gameover():
        gpio.cleanup()


##### Main code #####
init()

counter = np.uint64(0)
button = int(0)

while True:

        if int(gpio.input(12)) != int(button):
                button = int(gpio.input(12))
                counter += 1
                print(counter)

        if counter >= 960:
                gameover()
                print("Thanks for playing!")
                break


                         [ Cancelled ]
^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

- Create new Python script: ***encodercontrol02.py***

- Drive both right-side motors using PWM
- Monitor back right encoder




```python
import RPi.GPIO as gpio
import time
import numpy as np

##### Initialize GPIO pins #####

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

        gpio.setup(12, gpio.IN, pull_up_down = gpio.PUD_UP)

def gameover():
        gpio.output(31, False)
        gpio.output(33, False)
        gpio.output(35, False)
        gpio.output(37, False)

        gpio.cleanup()
```

Teleoperation

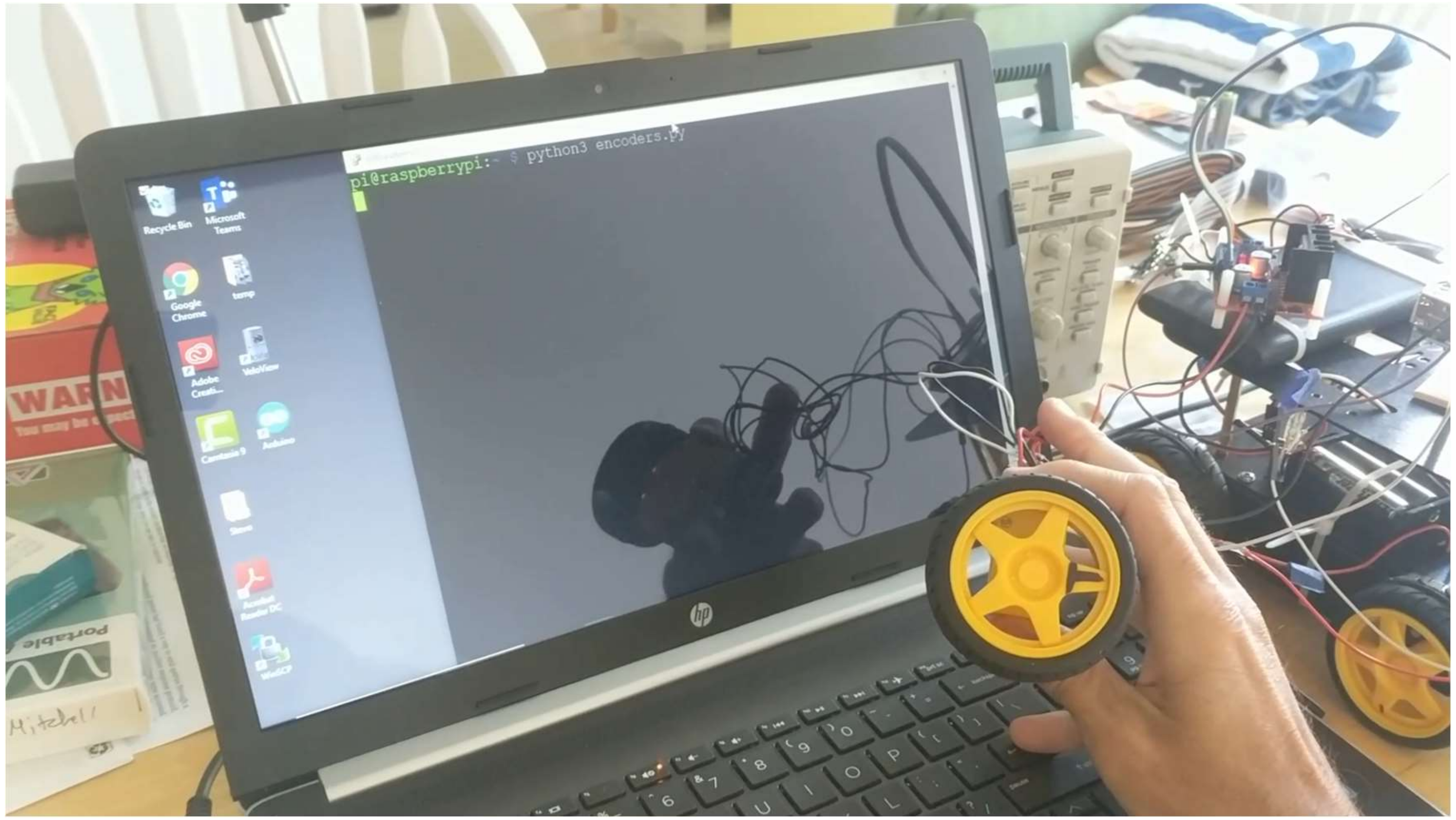
- Create new Python script: ***encodercontrol02.py***

- Drive both right-side motors using PWM
- Monitor back right encoder

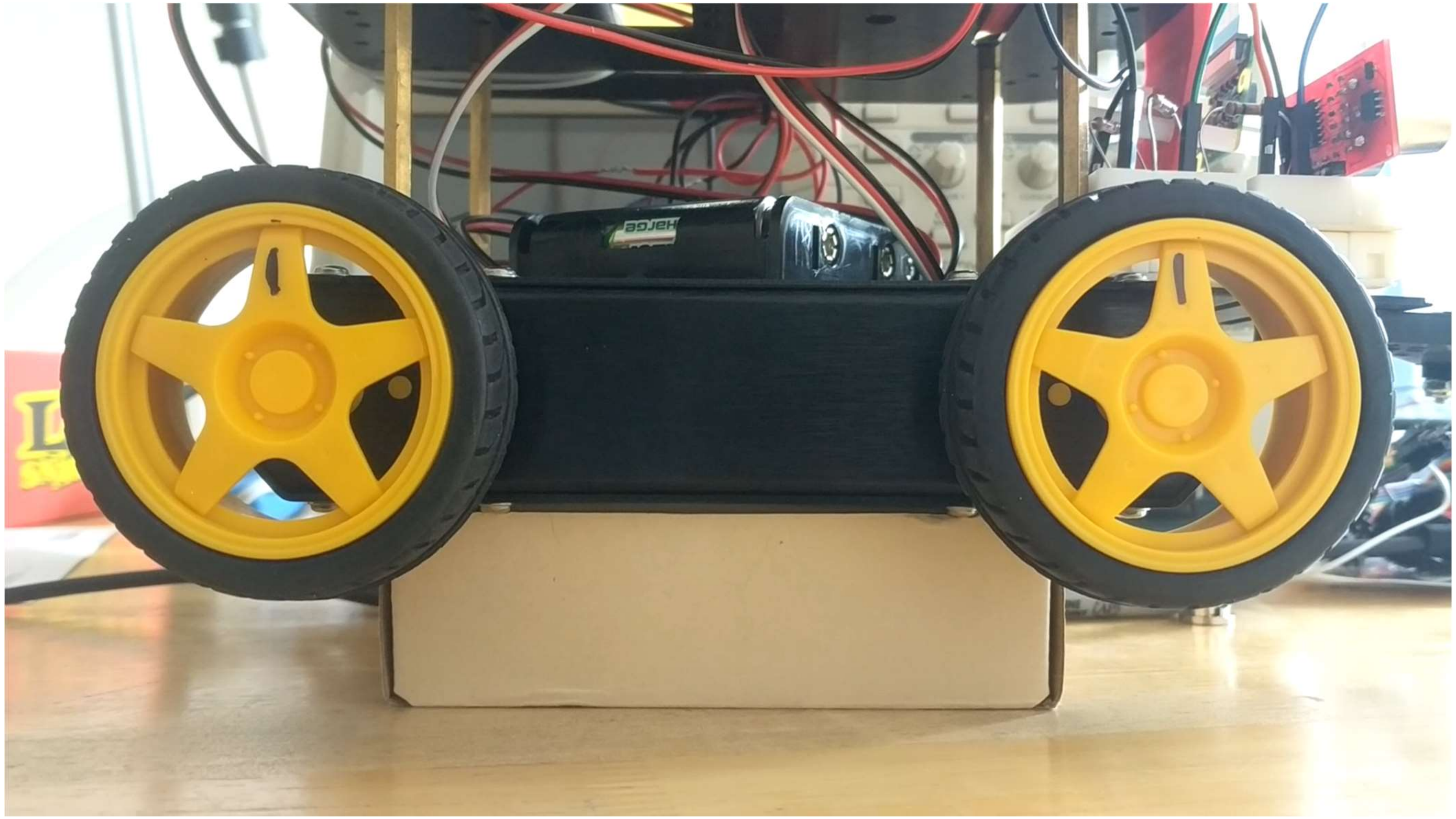
```
pi@raspberrypi: ~
  GNU nano 2.7.4            File: encodercontrol02.py            Modified

##### Main code #####
init()

counter = np.uint64(0)
button = int(0)

# Initialize pwm signal to control motor
pwm = gpio.PWM(37, 50)
val = 14
pwm.start(val)
time.sleep(0.1)

for i in range(0,100000):

        print("counter = ", counter, "GPIO state: ", gpio.input(12))

        if int(gpio.input(12)) != int(button):
                button = int(gpio.input(12))
                counter += 1

        if counter >= 960:
                pwm.stop()
                gameover()
                print("Thanks for playing!")
                break



^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 ***encodercontrol02.py***
- Confirm wheels rotate through 1 rev & 960 encoder ticks

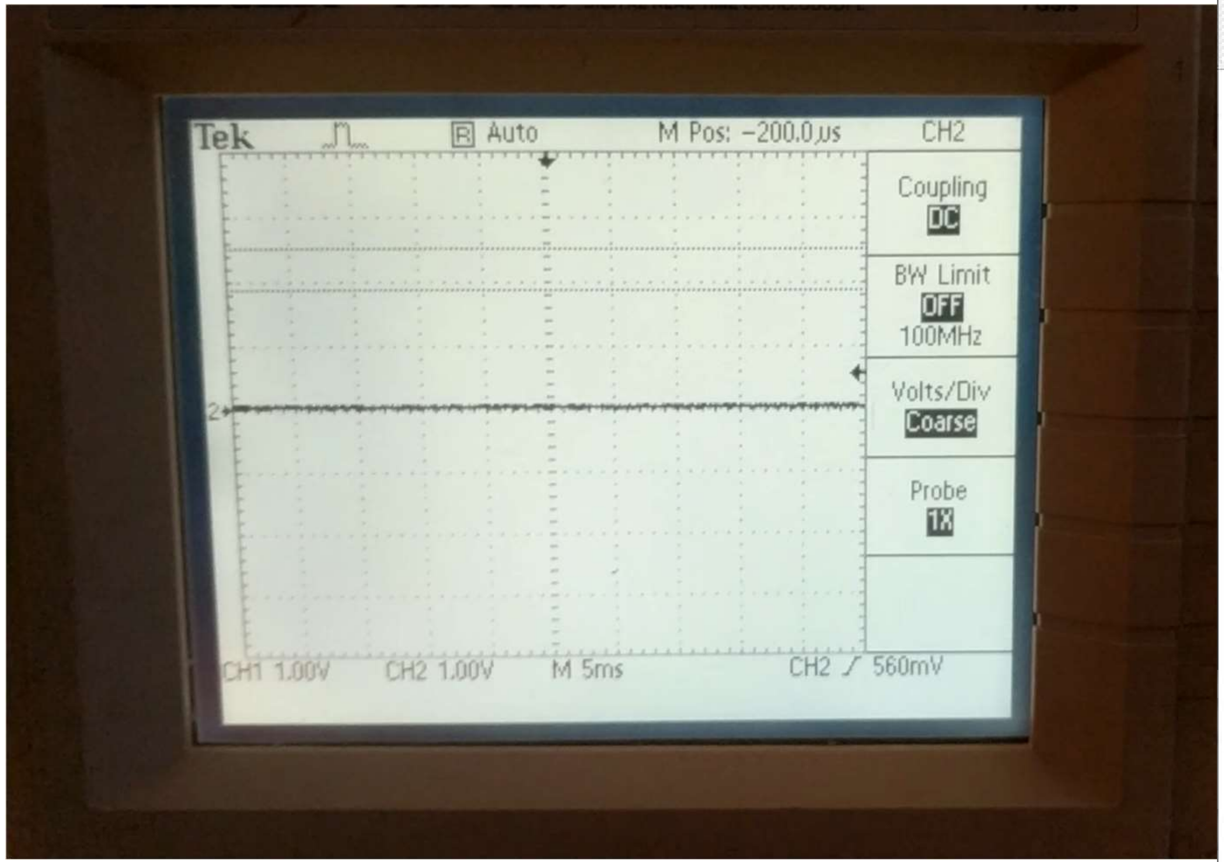


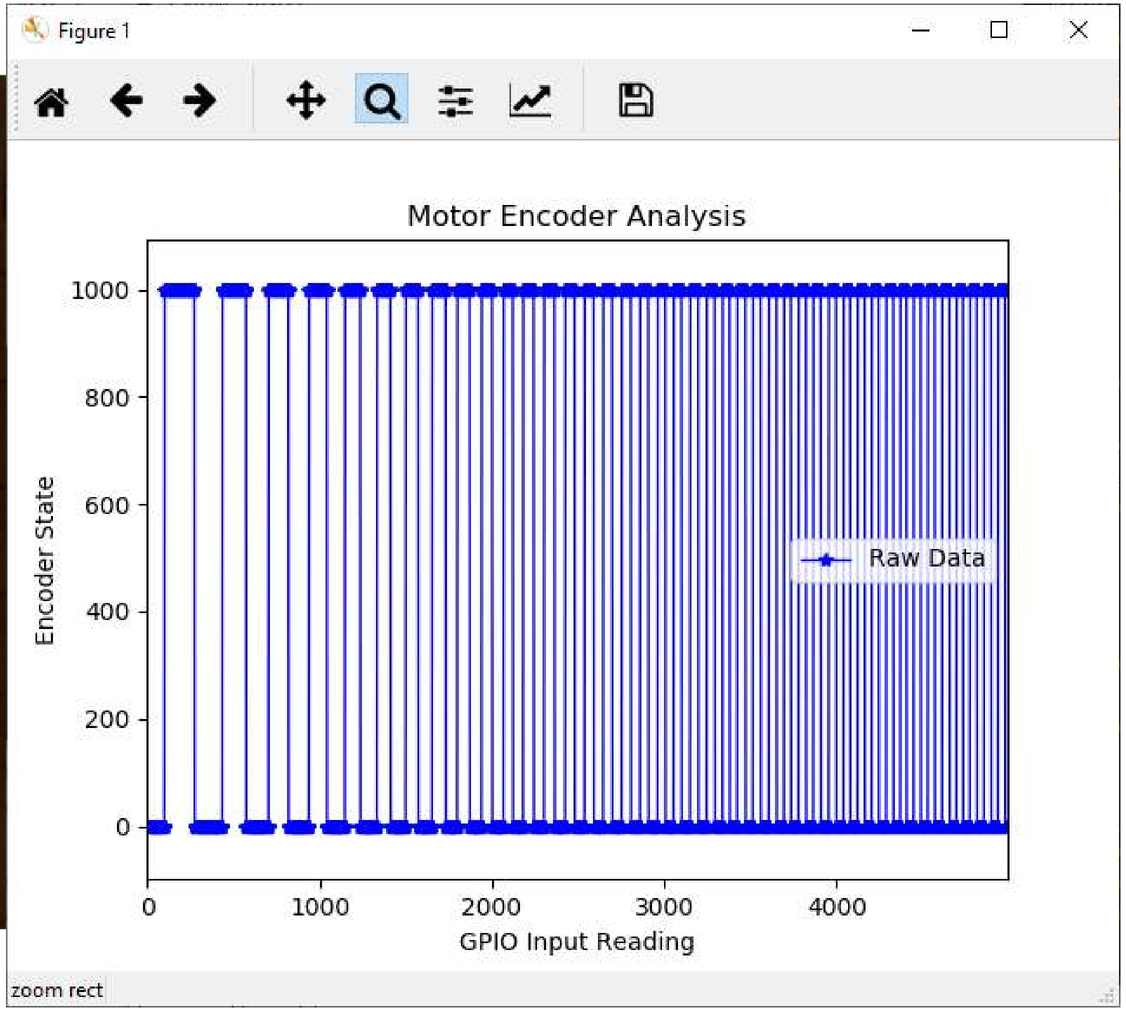
Localization Errors



In-Class Exercise

- Update ***encodercontrol02.py*** to save encoder states to .txt file
- Open & plot data in Matplotlib





ENPM 809T: Autonomous Robotics

Teleoperation

- Create new Python script: ***encodercontrol03.py***

- Drive right-side wheels only
- Track both encoders (left & right)

pi@raspberrypi: ~

GNU nano 2.7.4 File: encodercontrol03.py Modified

```python
import RPi.GPIO as gpio
import time
import numpy as np

##### Initialize GPIO pins #####

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

        gpio.setup(7, gpio.IN, pull_up_down = gpio.PUD_UP)
        gpio.setup(12, gpio.IN, pull_up_down = gpio.PUD_UP)

def gameover():
        gpio.output(31, False)
        gpio.output(33, False)
        gpio.output(35, False)
        gpio.output(37, False)

        gpio.cleanup()
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos ^Y Prev Page M-\ First Line
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line ^V Next Page M-/ Last Line

pi@raspberrypi: ~

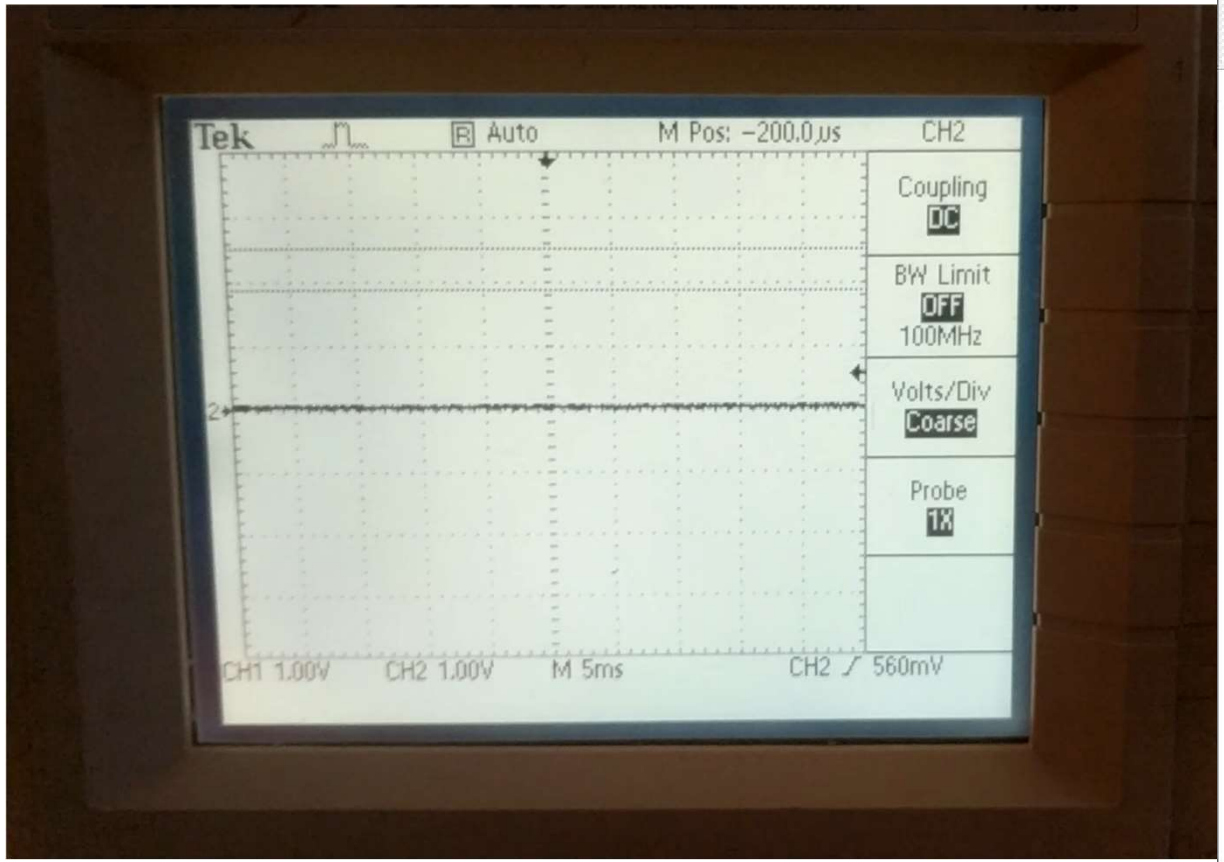
GNU nano 2.7.4 File: encodercontrol03.py Modified

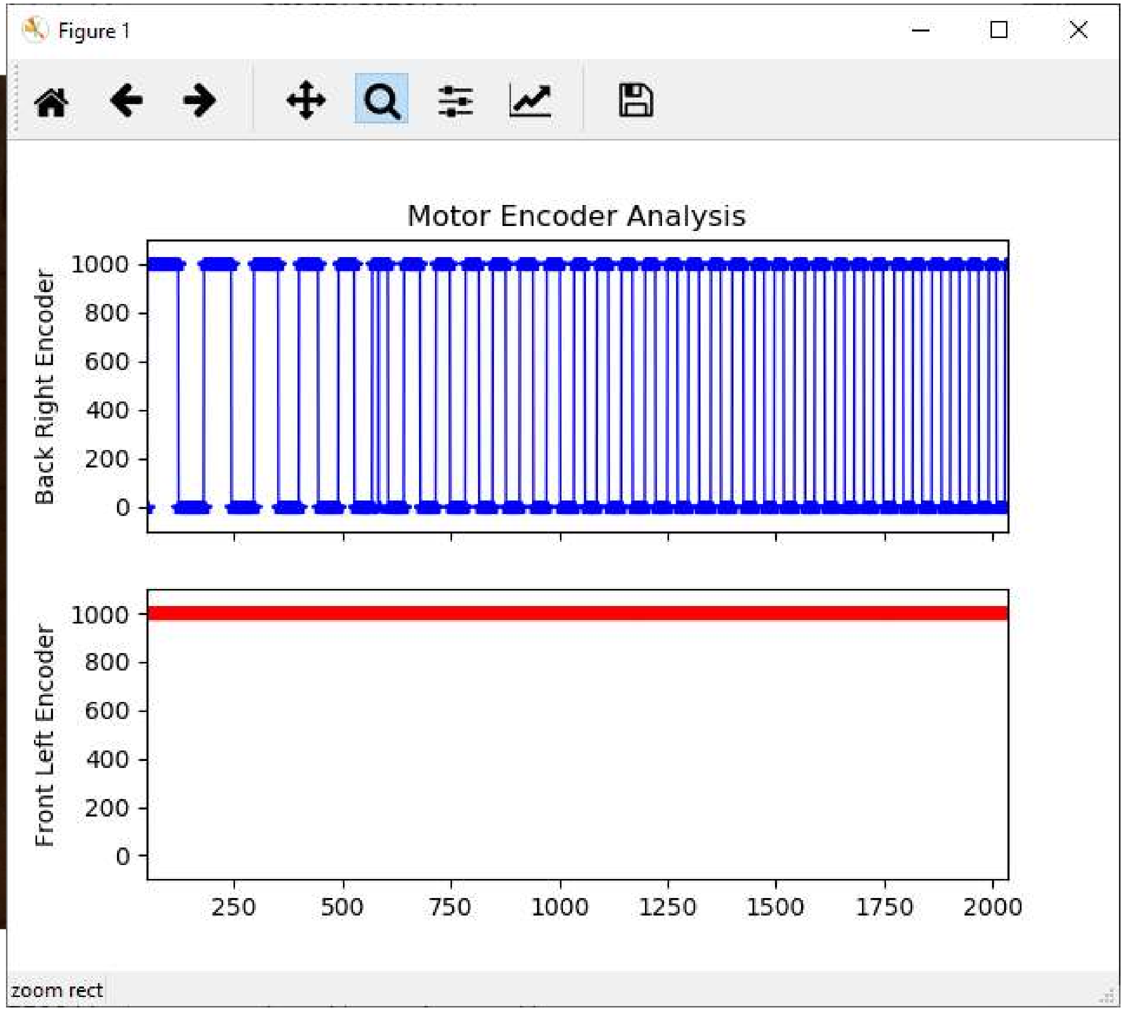
```python
##### Main code #####
init()
counterBR = np.uint64(0)
counterFL = np.uint64(0)

buttonBR = int(0)
buttonFL = int(0)

# initialize pwm signal to control motor
pwm = gpio.PWM(37, 50)
val = 16
pwm.start(val)
time.sleep(0.1)

for i in range(0,200000):

        print("counterBR: ", counterBR, "counterFL: ", counterFL, "BR state: ", gpio.input(12), "FL state: ", gpio.input(7))

        if int(gpio.input(12)) != int(buttonBR):
                buttonBR = int(gpio.input(12))
                counterBR += 1

        if int(gpio.input(7)) != int(buttonFL):
                buttonFL = int(gpio.input(7))
                counterFL += 1

        if counterBR >= 960:
                pwm.stop()
                gameover()
                print("Thanks for playing!")
                break
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos ^Y Prev Page M-\ First Line
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line ^V Next Page M-/ Last Line

In-Class Exercise

- Update ***encodercontrol03.py*** to save <u>both</u> encoder states to .txt file
- Open & subplot all data in Matplotlib





Localization Errors

Teleoperation

- Create new Python script: ***encodercontrol04.py***

- Drive all four motors
- Track both encoders (left & right)

pi@raspberrypi: ~

GNU nano 2.7.4 File: encodercontrol04.py Modified

```python
import RPi.GPIO as gpio
import time
import numpy as np

##### Initialize GPIO pins #####

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

        gpio.setup(7, gpio.IN, pull_up_down = gpio.PUD_UP)
        gpio.setup(12, gpio.IN, pull_up_down = gpio.PUD_UP)


def gameover():
        gpio.output(31, False)
        gpio.output(33, False)
        gpio.output(35, False)
        gpio.output(37, False)

        gpio.cleanup()
```

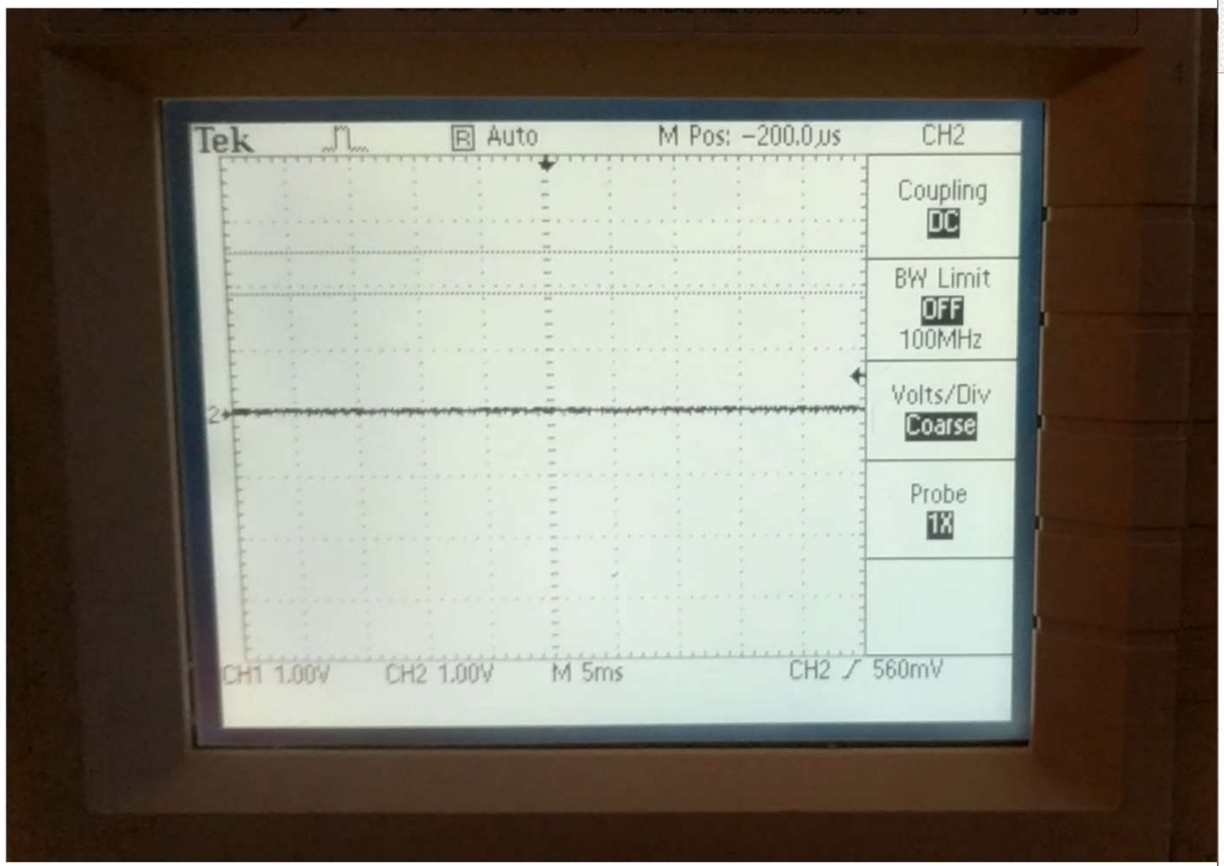
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos ^Y Prev Page M-\ First Line
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line ^V Next Page M-/ Last Line

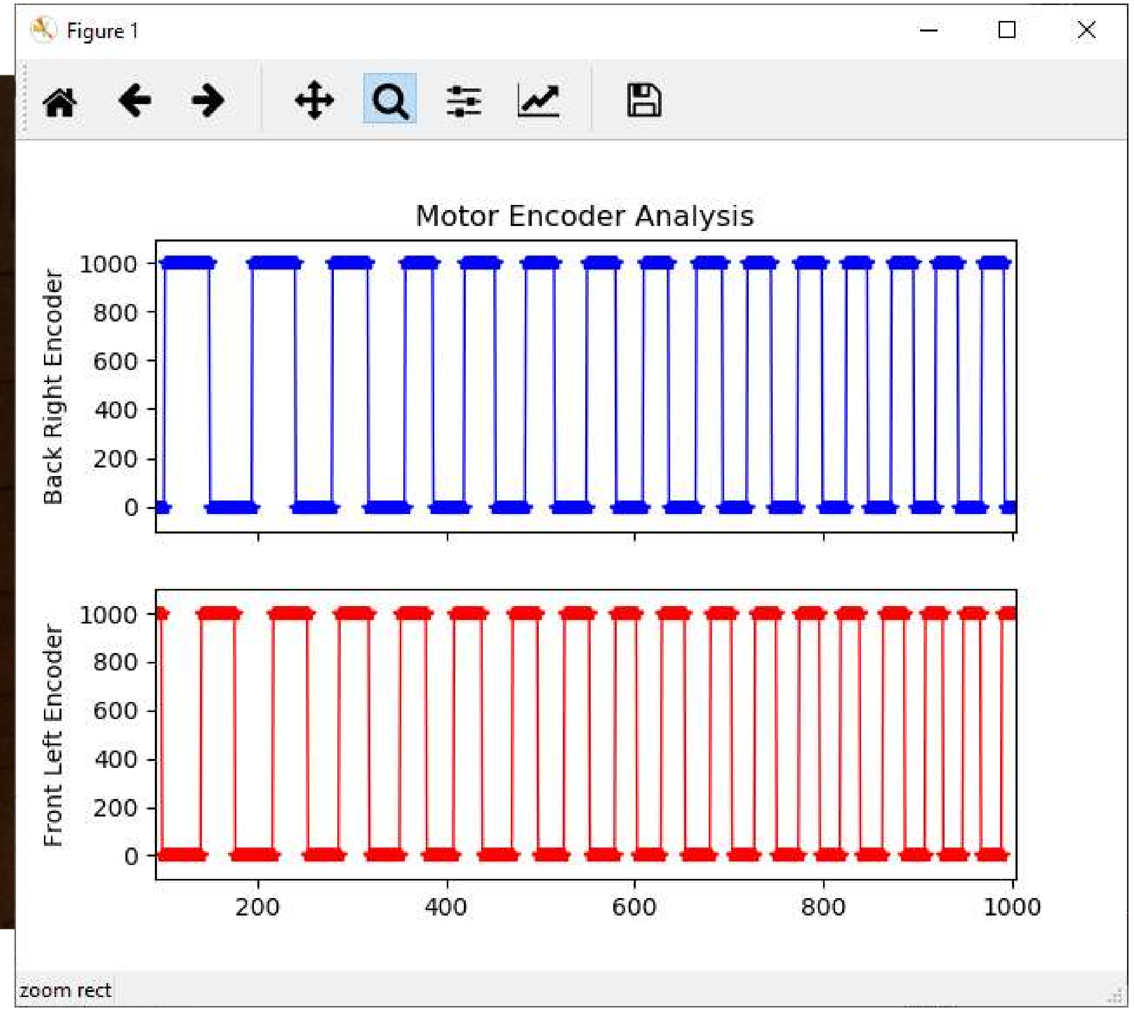
pi@raspberrypi: ~

```
  GNU nano 2.7.4                    File: encodercontrol04.py                    Modified

##### Main code #####
init()
counterFL = np.uint64(0)
counterBR = np.uint64(0)

buttonFL = int(0)
buttonBR = int(0)

# Independent motor control via pwm
pwm1 = gpio.PWM(31, 50)  # BackLeft motor
pwm2 = gpio.PWM(37, 50)  # FrontRight motor
val = 22
pwm1.start(val)
pwm2.start(val)
time.sleep(0.1)

for i in range(0,100000):
        print("counterBR: ", counterBR, "counterFL: ", counterFL, "BR state: ", gpio.input(12), "FL state: ", gpio.input(7))
        if int(gpio.input(12)) != int(buttonBR):
                buttonBR = int(gpio.input(12))
                counterBR += 1

        if int(gpio.input(7)) != int(buttonFL):
                buttonFL = int(gpio.input(7))
                counterFL += 1

        if counterFL >= 960:
                pwm1.stop()

        if counterBR >= 960:
                pwm2.stop()

        if counterBR >= 960 and counterFL >= 960:
                gameover()
                break

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     ^Y Prev Page   M-\ First Line
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter   ^_ Go To Line  ^V Next Page   M-/ Last Line
```

In-Class Exercise

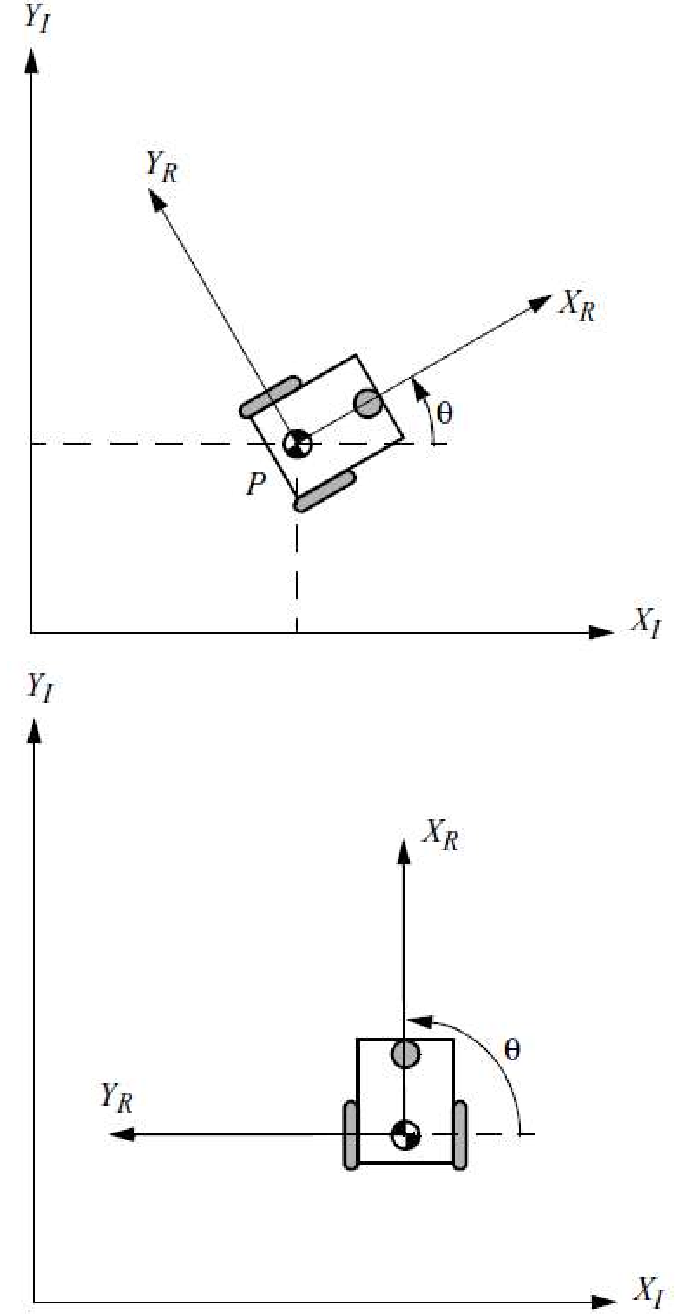
- Update ***encodercontrol04.py*** to save <u>both</u> encoder states to .txt file
- Open & subplot all data in Matplotlib



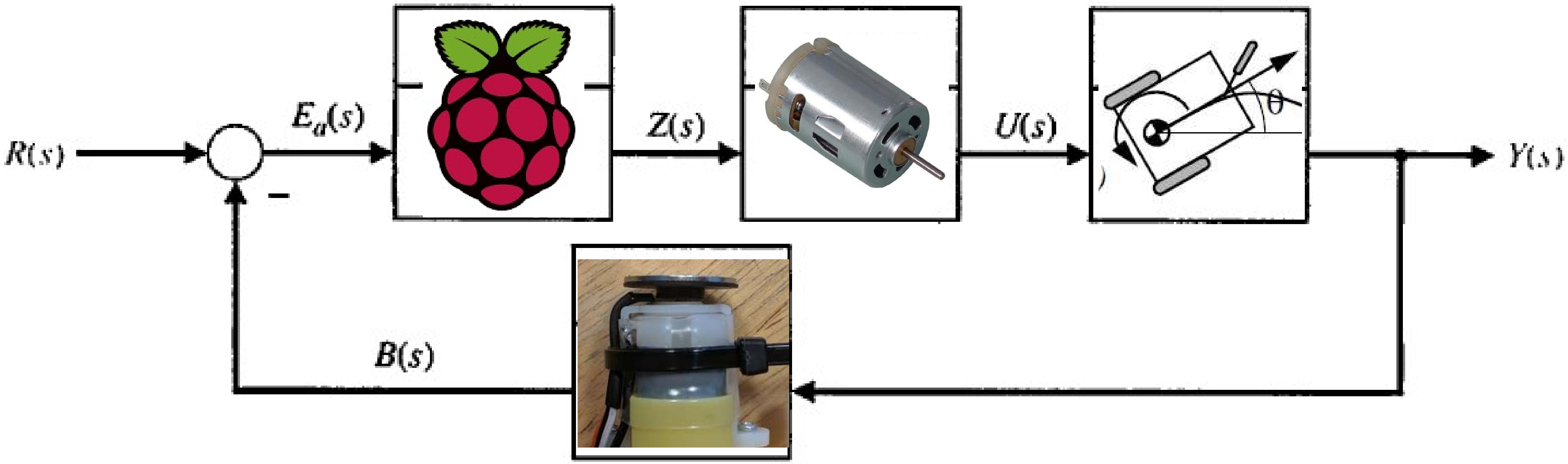


Kinematics

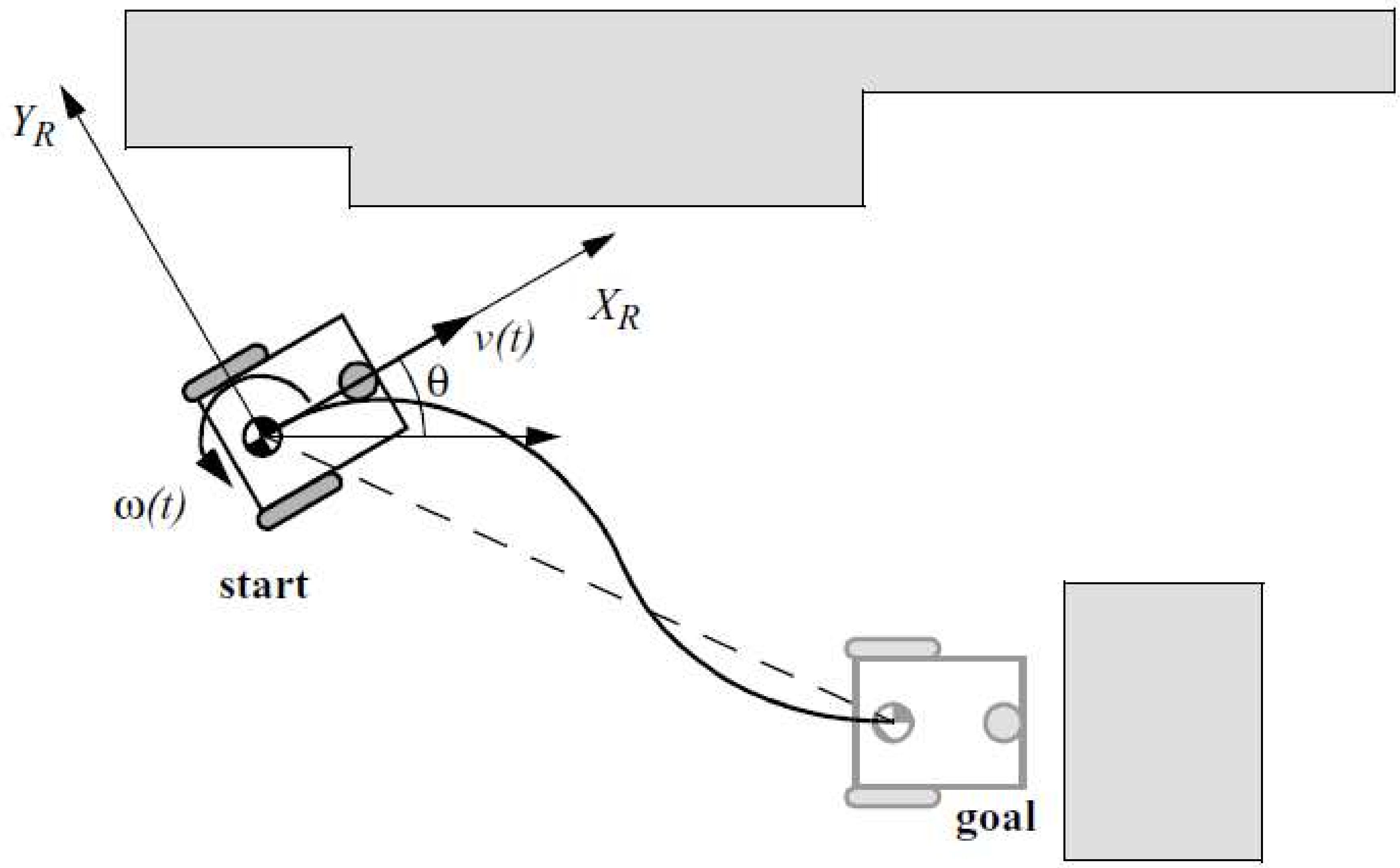
- Study of how mechanical systems behave
- Mobile robotics
  - Design for specific tasks
  - Create control software to drive hardware
- Global reference frame & robot local reference frame



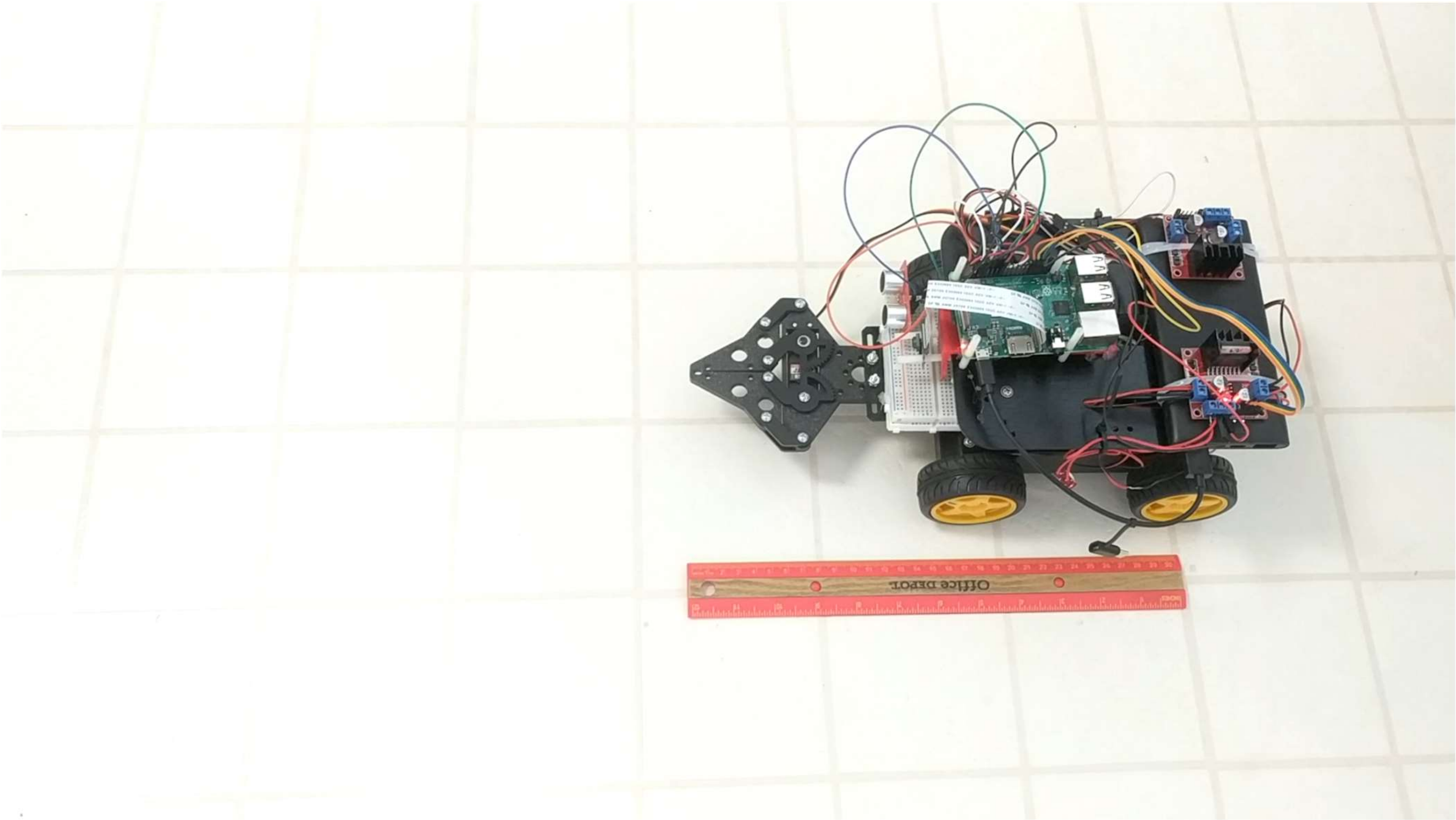
Control Theory



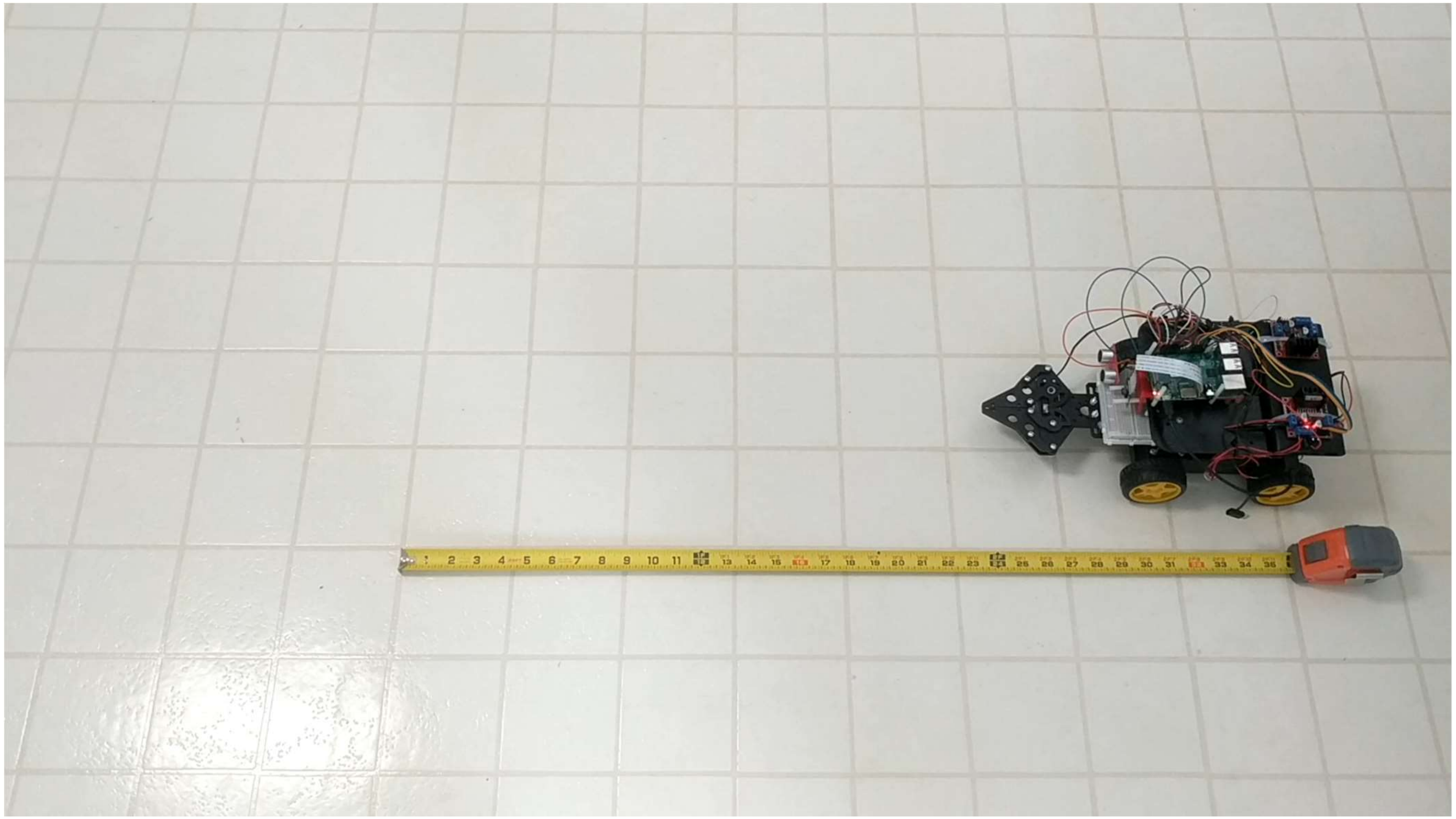
Kinematics & Localization



Kinematics & Localization



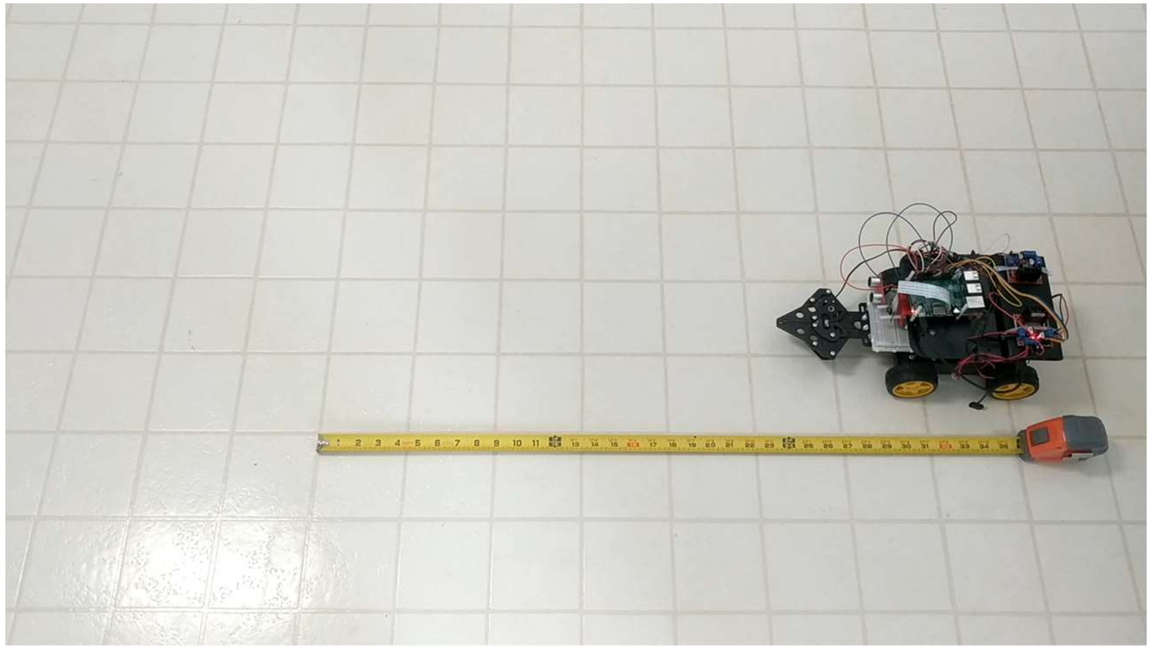
Kinematics & Localization

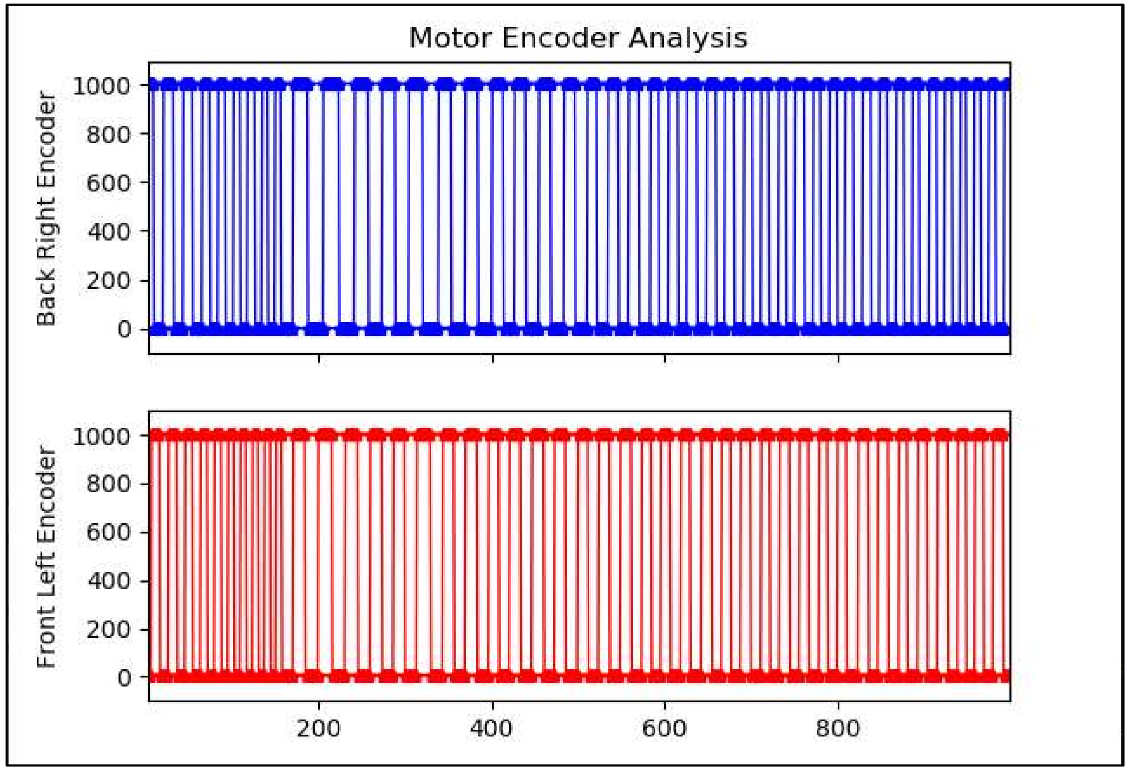


In-Class Exercise

- Create new Python script ***encodercontrol05.py***
- Script must:
  1. Drive robot in <u>straight</u> line for a user-defined distance
  2. Record encoder data from both encoders
- Open & subplot all data in Matplotlib

$$\left(\frac{1\, wheel\, rev}{2\pi(0.0325)meter}\right)(x\, meters) = \#\, wheel\, rev$$





References

- *Introduction to Autonomous Mobile Robots*, Siegwart
  - Chapter 5
- *Wheel Encoder Kit*, Sparkfun
  - https://www.sparkfun.com/products/12629
- RedBot Assembly Guide
  - https://learn.sparkfun.com/tutorials/redbot-assembly-guide-rev-02/wheel-encoder?_ga=2.100473117.2032092491.1562591960-231095067.1560280769
- Rpi.GPIO
  - https://pypi.org/project/RPi.GPIO/
- Resolution, Accuracy, and Precision of Encoders
  - https://cdn.usdigital.com/assets/general/usd-encoder-precision.pdf