Yoseph Ayele Kebede

UID: 114196729

Dr. Mitchell

ENPM701 Assignment 7

DC Motor Control using Encoder Count

Assignment #7

DC Motor Control via Encoder Count

Question #1

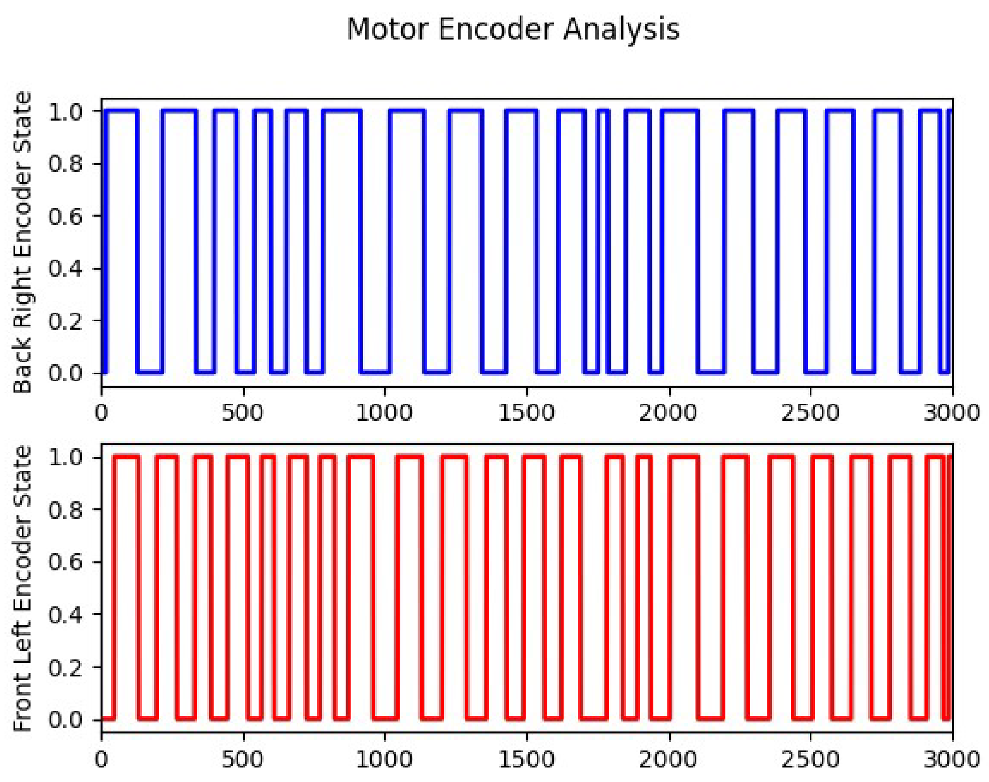
Motor Control Algorithm

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

#### 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)

# Distance to encoding conversion
# x_meters * (1 rev / (2pi*0.0325m)) = # wheel rev = 960 counter
def meter2encoder(x_dist):
    encod = round(float(x_dist / (2*math.pi*0.0325))*960)

    return encod
```

```
def rot2encoder(deg):

    # Approximate radius of rotation computed from Baron estimate
    radius = 0.111 #0.146 # meters
    # Angle needed for robot to rotate
    arc = ((deg * math.pi) / 180) * radius
    distance = round(float(arc / (2*math.pi*0.0325))*960)

    return distance

def main():
    #### Main Code ####
    init()

    counterBR = np.uint64(0)
    counterFL = np.uint64(0)

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


    # initialize pwm signal to control motor
    pwm01 = gpio.PWM(31, 50)  # BackLeft motor
    pwm11 = gpio.PWM(33, 50) # FrontLeft motor
    pwm22 = gpio.PWM(35, 50) # FrontRight motor
    pwm02 = gpio.PWM(37, 50)  # BackRight motor

    pwms = [pwm01,pwm11,pwm22,pwm02]

    valL = 35#25,45,14
    valR = 45#40,30,22

    vals = [valL,valR]

    pwms[0].start(0)
    pwms[1].start(0)
    pwms[2].start(0)
    pwms[3].start(0)

    time.sleep(0.1)

    # Open .txt file to save data
    f = open('FLBR_straight_encoder_states.txt','a')
    outstringF = ''
    # Open .txt file to save data
    rev = open('FLBR_reverse_encoder_states.txt','a')
    outstringRev = ''

    # Open .txt file to save data
    l = open('FLBR_left_encoder_states.txt','a')
    outstringL = ''

    # Open .txt file to save data
    r = open('FLBR_right_encoder_states.txt','a')
    outstringR = ''

    # Drive the robot 3 meters
    x = 2

    # Convert distance from meters to encoders
    encoder_dist = meter2encoder(x)

    encoder_dist = 0
    deg = 0


    decision = input("Do you want to go forward (f), reverse (r) or left (l) or
right (right)?: ")

    if decision == 'f':

        x = input("How far should I go forward? [0-10]meters: ")

        # Convert distance from meters to encoders
        encoder_dist = meter2encoder(float(x))

        # Send forward thruster commands
        #pwms = forward(tf,pwms,vals)

        pwm1 = pwms[0]
        pwm2 = pwms[3]

        pwm1.start(valL)
        pwm2.start(valR)


    if decision == 'r':
        x = input("How far should I go forward? [0-10]meters: ")
        # Convert distance from meters to encoders
        encoder_dist = meter2encoder(float(x))

        # Send forward thruster commands
        #reverse(tf)

        pwm1 = pwms[1]
        pwm2 = pwms[2]

        pwm1.start(valL)
        pwm2.start(valR)
#         pwm1.ChangeDutyCycle(35) #22,45,25,14
#         pwm2.ChangeDutyCycle(40) #30,65,35,22


    else:

#         direction = input("Should I turn left (L) or right (R): ")
        deg = input("How many degrees should I rotate? [0-90]deg: ")

        encoder_dist = rot2encoder(float(deg))

        if decision in ['L','l']:

            valL = 90#90,25,45,14
            valR = 80#80

            pwm1 = pwms[1]
            pwm2 = pwms[3]

        else:

            valL = 90#90,25,45,14
            valR = 94#94

            pwm1 = pwms[0]
            pwm2 = pwms[2]

        pwm1.start(valL)
        pwm2.start(valR)

    print("Meter converted to encoder is: " + str(encoder_dist))
    #pivotleft(tf)
    try:
```

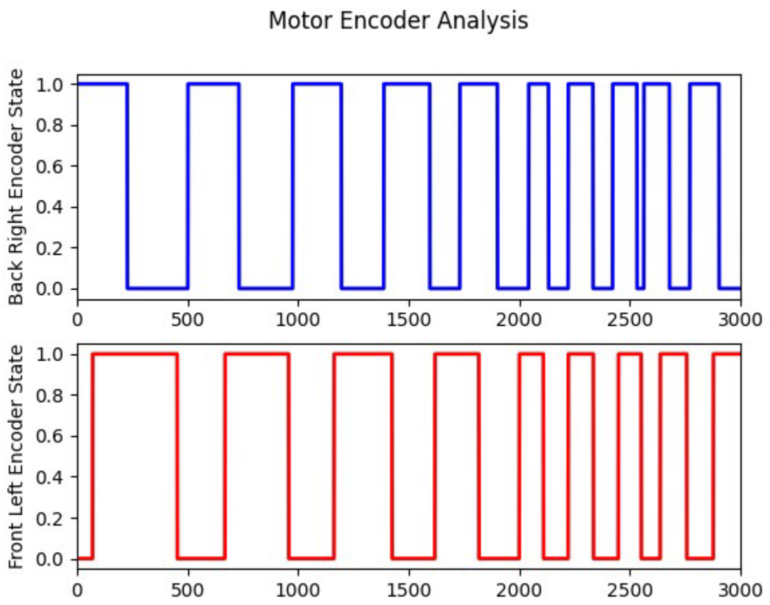
```python
while True:

    if decision in ['f','r']:

        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 > (counterBR + 20)):

            # Reduce the duty cycle of FL
            pwm1.ChangeDutyCycle(35) #22,45,25,14
            pwm2.ChangeDutyCycle(45) #30,65,35,22

        elif (counterBR > (counterFL + 20)):

            # Do vice versa than above.
            pwm1.ChangeDutyCycle(43) #29,60,35,22
            pwm2.ChangeDutyCycle(35) #25,40,25,14

    else:

        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 > (counterBR + 30)):

            if decision == ['L','l']:
                # Reduce the duty cycle of FL
                pwm1.ChangeDutyCycle(85) #85,22,45,25,14
                pwm2.ChangeDutyCycle(90) #90,30,65,35,22
            else:
                # Reduce the duty cycle of FL
                pwm1.ChangeDutyCycle(95) #95,22,45,25,14
                pwm2.ChangeDutyCycle(97) #97,65,35,22
                if (counterBR > (counterFL + 30)): #50 or (not right):

                    if decision == ['L','l']:
                        # Reduce the duty cycle of FL
                        pwm1.ChangeDutyCycle(97) #90,22,45,25,14
                        pwm2.ChangeDutyCycle(90) #80,30,65,35,22
                    else:
                        # Reduce the duty cycle of FL
                        pwm1.ChangeDutyCycle(99) #99,22,45,25,14
                        pwm2.ChangeDutyCycle(90) #90

            if ((counterBR >= encoder_dist) and (counterFL >= encoder_dist)):
                gameover()
                pwm1.stop()
                pwm2.stop()
                gpio.cleanup()

                print("counterBR: ", counterBR, "counterFL: ", counterFL)#, "BR
state: ", gpio.input(12), "FL state: ", gpio.input(7))
                print("Thanks for playing!")
                break

            if decision == 'f':
                print("forward")
                #Record encoder states to txt file
                outstringF = str(buttonBR) + ' ' + str(buttonFL) + '\n'
                f.write(outstringF)
            if decision == 'r':
                print("reverse")
                #Record encoder states to txt file
                outstringRev = str(buttonBR) + ' ' + str(buttonFL) + '\n'
                rev.write(outstringRev)
            if decision in ['L','l']:
                print("left")
                #Record encoder states to txt file
                outstringL = str(buttonBR) + ' ' + str(buttonFL) + '\n'
                l.write(outstringL)
            else:
                print("right")
                #Record encoder states to txt file
                outstringR = str(buttonBR) + ' ' + str(buttonFL) + '\n'
                r.write(outstringR)

    except KeyboardInterrupt:
        gameover()
#          if decision in ['f','r']:
        for pwm in pwms:
            pwm.stop()
        gpio.cleanup()
        print("counterBR: ", counterBR, "counterFL: ", counterFL)
        print("Thanks for playing!")

if __name__ == "__main__":

    main()
```

1.2 Driving Straight



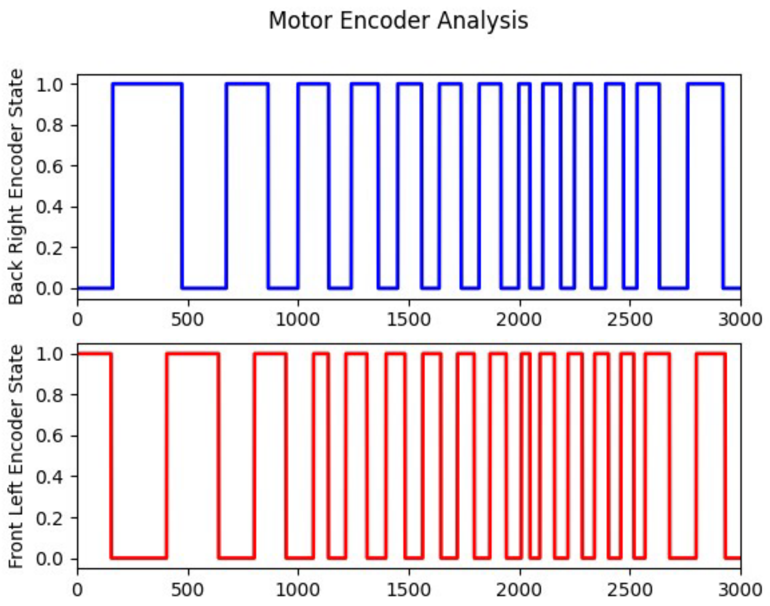
Video: https://youtu.be/ZM8gu7vePHw?si=NP2Ah-XSkil81joz

1.3 Driving Reverse



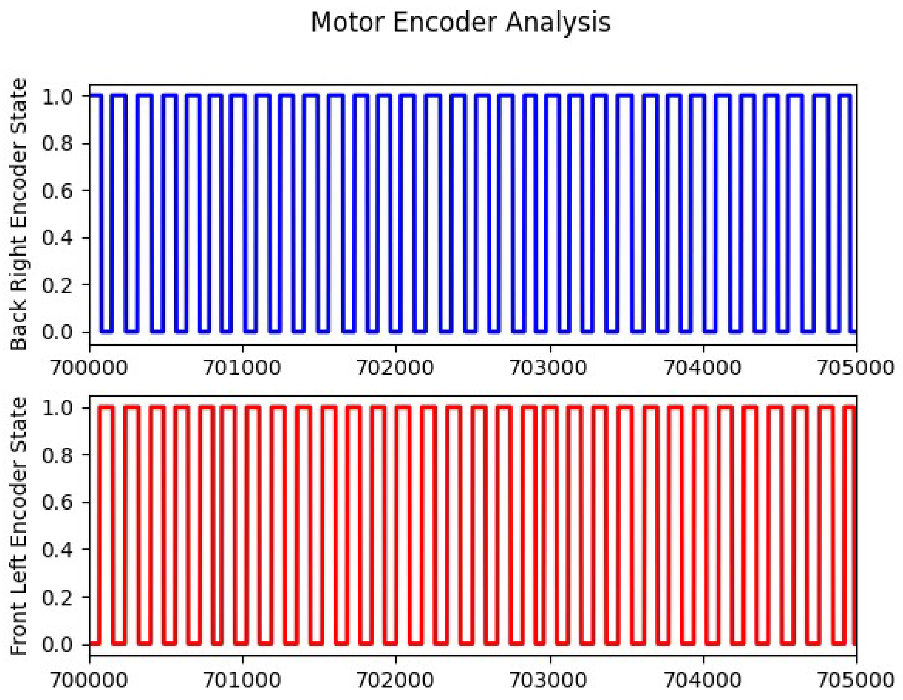
Video: https://youtu.be/LmujiGcn-3w?si=aCCsNXc3x-yS1pct

1.4 Pivot Left



Video: https://youtu.be/strBpSWuMBc?si=_H71qh5lZgcFqacR

1.5 Pivot Right



Video: https://youtu.be/KyMUv-JxjDc?si=5HMuLvxhN753mmjj

Summary:

The encoder data plotted in the above four cases shows that after the encoder state from one of the motors exceeds a certain threshold, as documented in the code above, the executed change in duty cycle shows to reduce the speed of the rotation of the wheel on the side of the wheel that has exceeded by "x" encoder ticks from its counterpart. After repeated testing and iterations on the four types of motion, the front left motor seems to rotate faster than the back right motor which is especially evident in straight line motions. During pivoting however, since higher Vrpm needs to be provided for the PWM pin to pivot the robot on a flat ground floor, the back right motor seems to spin faster although its difficult to say as the wheels kept on slipping. Therefore, the direction of the robot can be kept to traverse in a relatively stable path by reading the wheel encoders, although a use of sensors such as imu's will give greater resolution moving forward.