

~/Desktop/Fall 2023/ME 405/Lab0x05/controls.py

```
1  '''
2      @name          lab4_main.py
3      @brief         lab4 main file for circle line following
4      @author        tanner, noah
5      @date          november, 2023
6  '''
7
8  from pyb import Pin, ADC, Timer
9  import romi_driver as driver
10 import encoder_class as encoder
11 import time
12 from math import pi
13
14 # sensor array [ 3, 4, 5, 6, 7, 8, 9 ] with 3 being on left when romi faces forward
15
16 # setup sensor pins
17 P3      = Pin(Pin.cpu.A0, mode=Pin.IN)
18 P4      = Pin(Pin.cpu.A1, mode=Pin.IN)
19 P5      = Pin(Pin.cpu.A4, mode=Pin.IN)
20 P6      = Pin(Pin.cpu.B0, mode=Pin.IN)
21 P7      = Pin(Pin.cpu.A7, mode=Pin.IN)
22 P8      = Pin(Pin.cpu.A6, mode=Pin.IN)
23 P9      = Pin(Pin.cpu.A5, mode=Pin.IN)
24
25 # adc setup
26 adc3    = ADC(P3)
27 adc4    = ADC(P4)
28 adc5    = ADC(P5)
29 adc6    = ADC(P6)
30 adc7    = ADC(P7)
31 adc8    = ADC(P8)
32 adc9    = ADC(P9)
33
34 val_array = [ 0, 0, 0, 0, 0, 0, 0 ]
35 adc_array = [ adc3, adc4, adc5, adc6, adc7, adc8, adc9 ]
36
37 # calibration values:
38 white = [285.7, 279.9, 276.7, 272.9, 292.4, 1836.1, 4059.3]
39 black = [3274.6, 3094.8, 2866.8, 2749.4, 2084.8, 1858.6, 4059.2]
40
41
42 def calibrate(color):
43     cal_array = [ 0, 0, 0, 0, 0, 0, 0 ]
44     idx       = 0
45     iter      = 10
46
47     if color == 'white':
48         while iter > 0:
49             while idx < len(cal_array):
50                 cal_array[idx] += adc_array[idx].read()
51                 idx += 1
52             time.sleep(1)
53             idx = 0
```

```

54         iter -= 1
55         print(cal_array)
56
57         white_cal_array = [ ( val / 10 ) for val in cal_array ]
58         return white_cal_array
59
60     if color == 'black':
61         while iter > 0:
62             while idx < len(cal_array):
63                 cal_array[idx] += adc_array[idx].read()
64                 idx += 1
65             time.sleep(1)
66             idx = 0
67             iter -= 1
68             print(cal_array)
69
70         black_cal_array = [ ( val / 10 ) for val in cal_array ]
71         return black_cal_array
72
73     def read():
74         idx = 0
75         while idx < len(val_array):
76             val_array[idx] = adc_array[idx].read()
77             idx += 1
78
79         return val_array
80
81     def calc_centroid():
82         sensor_vals = read()
83         weighted_sum = sum(i * val for i, val in enumerate(sensor_vals))
84         total_sum = sum(sensor_vals)
85         centroid = weighted_sum / total_sum
86
87         return centroid
88
89     def forward():
90         right.enable()
91         left.enable()
92
93     def turn_right(left_speed):
94         left.set_duty(left_speed, 0)
95         left.enable()
96         right.disable()
97
98     def turn_left(right_speed):
99         right.set_duty(right_speed, 0)
100         right.enable()
101         left.disable()
102
103     def pivot_right(speed):
104         left.set_duty(speed+5, 0)
105         right.set_duty(speed, 1)
106         left.enable()
107         right.enable()
108
109     def pivot_left(speed):

```

```

110     left.set_duty(speed, 1)
111     right.set_duty(speed, 0)
112     left.enable()
113     right.enable()
114
115 def stop():
116     right.disable()
117     left.disable()
118
119 cpr          = 1440          # encoder count per revolution of romi wheel
120 wheel_dia    = 2.83          # romi wheel diameter in inches
121 wheel_circ   = 2.83 * pi
122 def calc_distance(counts):
123     rev = counts / cpr
124     dist = rev * wheel_circ
125     return dist
126
127 # initialize total distance variables
128 dist_left = 0
129 dist_right = 0
130 def loop(threshold, base_speed, kp, ki, kd):
131     total_dist = 0
132     while total_dist < 75.36:
133         enc_right.update()
134         enc_left.update()
135         dist_left = calc_distance(-enc_left.total_position)
136         dist_right = calc_distance(-enc_right.total_position)
137         total_dist = .5 * (dist_left + dist_right)
138         print(f"Distances [ L: {dist_left}, R: {dist_right}, Total: {total_dist}]")
139         sensor_vals = read()
140         centroid = calc_centroid()
141         reference_pt = len(sensor_vals) / 2
142         pid_output = pid_controller(centroid, reference_pt, kp, ki, kd)
143         new_left = base_speed + pid_output
144         new_right = base_speed - pid_output
145
146         # check encoder values to determine when to stop
147
148         if centroid < reference_pt - threshold:
149             turn_left(new_left, new_right)
150             print('turned left')
151         elif centroid > reference_pt + threshold:
152             turn_right(new_left, new_right)
153             print('turned right')
154         else:
155             forward()
156     stop()
157     print('arrived at the start')
158
159 # initialize variables for pid controller
160 prev_error = 0
161 integral = 0
162
163 def pid_controller(centroid, reference_pt, kp, ki, kd):
164     global integral, prev_error
165     error = reference_pt - centroid

```

```

166     integral    += error
167     derivative  = error - prev_error
168
169     pid_res     = kp * error + ki * integral + kd * derivative
170     prev_error  = error
171     print(pid_res)
172     return pid_res
173
174 # left driver
175 tim_left       = Timer(4, freq = 20_000)
176 pwm_left_pin   = Pin(Pin.cpu.B6, mode=Pin.OUT_PP)
177 pwm_left       = tim_left.channel(1, pin = pwm_left_pin, mode=Timer.PWM)
178 dir_left_pin   = Pin(Pin.cpu.B9, mode=Pin.OUT_PP)
179 en_left_pin    = Pin(Pin.cpu.B7, mode=Pin.OUT_PP)
180 left           = driver.romi_driver(tim_left, pwm_left, dir_left_pin, en_left_pin)
181
182 # right driver
183 tim_right      = Timer(8, freq = 20_000)
184 pwm_right_pin  = Pin(Pin.cpu.C9)
185 dir_right_pin  = Pin(Pin.cpu.C8, mode=Pin.OUT_PP)
186 en_right_pin   = Pin(Pin.cpu.C7, mode=Pin.OUT_PP)
187 # configure channels 2, 3, 4
188 pwm_right      = tim_right.channel(4, pin = pwm_right_pin, mode=Timer.PWM)
189 right          = driver.romi_driver(tim_right, pwm_right, dir_right_pin,
190 en_right_pin)
191
192 # encoder left
193 ps             = 0
194 ar             = 1000
195 l_pin_cha      = Pin(Pin.cpu.B4, mode=Pin.OUT_PP) # encoder
196 l_pin_chb      = Pin(Pin.cpu.B5, mode=Pin.OUT_PP) # encoder
197 l_pin_cha      = Pin(Pin.cpu.B4, mode=Pin.OUT_PP) # encoder
198 l_pin_chb      = Pin(Pin.cpu.B5, mode=Pin.OUT_PP) # encoder
199 tim_left_3     = Timer(3, period = ar, prescaler = ps) # encoder
200 1 timer
201 l_cha          = tim_left_3.channel(1, pin=l_pin_cha, mode=Timer.ENC_AB)
202 l_chb          = tim_left_3.channel(2, pin=l_pin_chb, mode=Timer.ENC_AB)
203 enc_right      = encoder.Encoder(tim_left_3, l_cha, l_chb, ar, ps) # encoder
204 1 instance
205
206 # encoder right
207 r_pin_cha      = Pin(Pin.cpu.A8, mode=Pin.OUT_PP) # encoder
208 r_pin_chb      = Pin(Pin.cpu.A9, mode=Pin.OUT_PP) # encoder
209 r_pin_cha      = Pin(Pin.cpu.A8, mode=Pin.OUT_PP) # encoder
210 r_pin_chb      = Pin(Pin.cpu.A9, mode=Pin.OUT_PP) # encoder
211 tim_left_1     = Timer(1, period = ar, prescaler = ps) # encoder
212 1 timer
213 r_cha          = tim_left_1.channel(1, pin=r_pin_cha, mode=Timer.ENC_AB)
214 r_chb          = tim_left_1.channel(2, pin=r_pin_chb, mode=Timer.ENC_AB)
215 enc_left       = encoder.Encoder(tim_left_1, r_cha, r_chb, ar, ps) # encoder
216 1 instance
217
218 # set motor speed and direction ( 0 = forward, 1 = reverse )
219 left.disable()
220 right.disable()
221 # VALUES from testing: Threshold: 0.1, P: -, I: -, D: -

```