

POLITECHNIKA WARSZAWSKA

**WYDZIAŁ ELEKTRONIKI I TECHNIK
INFORMACYJNYCH**

WSTĘP DO ROBOTYKI

Dokumentacja programu robota LEGO EV3 – „Line Follower”

Wykonali:

Patryk Okoński

Adrian Krężel

WEiTI, AIR, s. 15Z

Spis treści

Algorytm podążania za linią	3
Implementacja algorytmu śledzenie linii	3
Sposób dobierania parametrów	3
Wady i zalety zastosowanego rozwiązania	3
Opis budowy robota	4
Układ sterujący	4
Silnik DC.....	4
Czujnik koloru	5
Czujnik natężenia światła	5
Ultradźwiękowy czujnik odległości	6
Czujnik podczerwieni	6
KOD PROGRAMU	7

Algorytm podążania za linią

Algorytm podążania za linią wykorzystuje dwa czujniki: światła oraz koloru. Oba czujniki zwracają informację na temat koloru powierzchni, która odbija emitowane przez nie światło, w postaci natężenia odbijanego światła (im mniejsza wartość odczytu tym czarniejsza powierzchnia).

Czujniki umieszczone są z przodu robota, jeden z lewej strony (czujnik koloru), drugi z prawej strony (czujnik światła) względem linii. Czujniki umieszczone są w takiej odległości od siebie, aby „łapały” tylko kolor biały.

Algorytm podążania za linią bada na bieżąco wartości zwracane przez czujniki i z wykorzystaniem regulatorów PID steruje pracą silnika tak, aby czujniki znajdowały się tylko na białym polu. W sytuacji, gdy czujnik umieszczony z lewej strony robota wykryje kolor czarny, na skutek działania regulatora PID moc silnika lewego zostaje zmniejszona, natomiast moc silnika prawego zostaje zwiększena. W przypadku gdy czujnik umieszczony z prawej strony robota wykryje kolor czarny, na skutek działania regulatora PID moc silnika prawego zostaje zmniejszona, natomiast moc silnika lewego zostaje zwiększena. Całość skutkuje jazdą robota wzdłuż wytyczonej czarnej linii.

Obszerny opis algorytmu znaleźć można w załączonym kodzie programu.

Implementacja algorytmu śledzenie linii

Implementacja algorytmu śledzenia linii wraz z obszernym opisem w postaci komentarzy znajduje się w załączonym kodzie programu.

Sposób dobierania parametrów

W celu zapewnienia poprawnego działania regulatorów konieczne było wyznaczenie wzmacnień członów: proporcjonalnego, całkującego oraz różniczkującego. Z uwagi na brak możliwości podłączania oscyloskopu i wyznaczenia wartości wzmacnień metodą Zieglera – Nicholsa zdecydowano się na wyznaczenie ich metodą doświadczalną.

Pierwszym krokiem było stopniowe zwiększanie wartości wzmacnienia członu proporcjonalnego, aż do osiągnięcia oscylacyjnego ruchu robota. Następnie uzyskaną wartość zmniejszono o połowę i wprowadzono wzmacnienia członów całkującego i różniczkującego. Dalej sprawdzano różne kombinacje wartości dwóch wprowadzonych członów, aż do osiągnięcia zadowalającego efektu. Stopniowe zwiększanie wzmacnienia członu całkującego wzmacniało wpływ zeszłych uchybów na wartość regulatora natomiast zwiększanie wzmacnienia członu różniczkującego wzmacniało reakcję robota na nagłe zmiany uchybu, dzięki czemu dokładniej trzymał się linii. Zbyt duże wartości wzmacnień członów całkującego i różniczkującego powodowały kolejno przesterowywanie robota przy powracaniu do linii oraz przesterowywanie robota przy nagłych zmianach uchybu.

Wady i zalety zastosowanego rozwiązania

Jedną z wad przedstawionego rozwiązania jest konieczność w miarę dokładnego doboru parametrów regulatorów. Niewłaściwe ich wyznaczenie może prowadzić albo do zbyt wolnej reakcji robota na przejeżdżanie linii, bądź też zbyt mocnego przesterowywania robota w wyniku czego może on wypaść poza tor. Właściwe dobranie parametrów regulatora pozwala jednak na płynną jazdę robota i bezbłędne trzymanie się linii.

Zastosowanie dwóch czujników wykrywających najeżdżanie na linię pozwoliło na proste rozwiązanie problemu przejazdu skrzyżowań. Robot może albo rozpoznawać jednoczesny najazd dwoma czujnikami na linię i np. ustawić jednakową prędkość silników do czasu przejazdu skrzyżowania, bądź wskazania regulatorów mogą się wzajemnie wyzerować, co przyniesie jednakowy efekt.

Opis budowy robota

Robot składał się z elementów LEGO Mindstorms EV3 oraz LEGO Technic. Elementy z grupy EV3 stanowiły elektroniczną część robota i były to:

- Układ sterujący
- Dwa silniki DC duże
- Czujnik koloru
- Czujnik światła
- Ultradźwiękowy czujnik odległości
- Czujnik podczerwieni

Elementy z grupy Technic były częściami składowymi budowy robota i posłużyły do połączenia elementów elektronicznych w jedną całość oraz osadzenie ich na podwoziu robota.

Układ sterujący

Stanowi najważniejszy element robota mobilnego. W projekcie układem sterującym jest klocek EV3 o następujących parametrach technicznych:

- 4 porty wejściowe
- 4 porty wyjściowe
- Port PC mini USB
- Port hosta USB
- Port kart Micro SD
- Wbudowany głośnik
- Wyświetlacz

Porty wejściowe umożliwiają podłączenie sensorów do robota natomiast porty wyjściowe służą do sterowania pracą silników. Dzięki specjalnemu specjalnym bibliotekom możliwe jest proste odczytywanie wartości czujników oraz sterowanie silnikami.



Rysunek 1. LEGO MINDSTORMS EV3 – Klock EV3

Silnik DC

Urządzeniami dzięki którym możliwe jest przemieszczanie się robota po torze są dwa silniki prądu stałego z zestawu LEGO Mindstorms. Silniki te są silnikami krokowymi umożliwiającymi precyzyjne sterowanie kątem obrotu z dokładnością do 1 stopnia.



Rysunek 2. LEGO MINDSTORMS – Silnik DC duży

Czujnik koloru

Czujnik koloru służy do wykrywania intensywności światła, na podstawie której rozpoznaje 8 różnych barw. W projekcie robota wykorzystano go wraz z czujnikiem natężenia światła do sterowania torem jazdy robota.



Rysunek 3. LEGO MINDSTORMS – Czujnik koloru

Czujnik natężenia światła

Czujnik ten posiada moduł, który mierzy natężenie światła oraz diodę, która pozwala oświetlić i zmierzyć jasność obiektu bezpośrednio przed czujnikiem. W projekcie wartości odczytywane z czujnika wykorzystywane są przez regulator PID w celu wypracowania sygnału sterującego.



Rysunek 4. LEGO MINDSTORMS – Czujnik natężenia światła

Ultradźwiękowy czujnik odległości

Czujnik służy do pomiaru dystansu pomiędzy obiektem a czujnikiem z wykorzystaniem ultradźwięków. W projekcie zamontowano go na bocznej powierzchni robota. Umożliwia on objazd przeszkody.



Rysunek 5. LEGO MINDSTORMS – Ultradźwiękowy czujnik odległości

Czujnik podczerwieni

Czujnik podczerwieni jest sensorem służącym do wykrywania sygnału podczerwieni, np. z pilota. Można go także stosować do pomiaru odległości. Posiada nadajnik podczerwieni, dzięki któremu może sam emitować sygnał podczerwieni i mierzyć czas pomiędzy jego emisją a powrotem, na podstawie czego oblicza dystans od przeszkody. W projekcie służy do wykrywania przeszkody znajdującej się przed pojazdem.



Rysunek 6. LEGO MINDSTORMS - Czujnik podczerwieni

KOD PROGRAMU

```

1 ######
2 # Wstęp do Robotyki (WR) - Laboratoria
3 # Kod programu robota LEGO EV3 - "Line Follower"
4 # Wykonali:
5 #     PATRYK OKOŃSKI
6 #     ADRIAN KREŽEL
7 #####
8
9 #Import niezbędnych bibliotek.
10 from time import sleep
11 import sys, argparse, time
12 from ev3dev import*
13
14 #Przypisanie do kolejnych zmiennych odpowiadających im czujników i silników.
15 lmotor = large_motor(OUTPUT_D) #Silnik lewy.
16 rmotor = large_motor(OUTPUT_A) #Silnik prawy.
17 irda = infrared_sensor() #Czujnik podczerwieni.
18 sonar = ultrasonic_sensor() #Czujnik ultradzwiekowy.
19 l_kolor = color_sensor() #Czujnik kolorów umieszczony z lewej strony robota.
20 r_kolor = light_sensor() #Czujnik światła umieszczony z prawej strony robota.
21
22 #Sprzwdzenie czy silniki i czujniki wcześniej zadeklarowane są podłączone.
23 #W sytuacji w której jeden z elementów nie będzie podłączany program zostanie zakończony
24 #z informacją dotyczącą błędu.
25 assert lmotor.connected #Sprawdzenie podłączenia silnika lewego.
26 assert rmotor.connected #Sprawdzenie podłączenia silnika PRAWEGO.
27 assert irda.connected #Sprawdzenie podłączenia czujnika podczerwieni.
28 assert sonar.connected #Sprzwdzenie podłączenia czujnika ultradzwiekowego.
29 assert l_kolor.connected #Sprawdzenie podłączenia czujnika kolorów.
30 assert r_kolor.connected #Sprawdzenie podłączenia czujnika światła.
31
32 #Włączenie regulacji prędkości silnika. Kiedy regulują prędkości jest włączona, sterownik
33 #silnika tak reguluje mocą dostarczana, aby silnik utrzymywał prędkość określona
34 #w zmiennej speed_sp.
35 lmotor.speed_regulation_enabled = 'on' #Włączenie regulacji prędkości silnika lewego.
36 rmotor.speed_regulation_enabled = 'on' #Włączenie regulacji prędkości silnika prawego.
37
38 #Ustawienie trybu pracy czujnika koloru. Tryb "COL-REFLECT" powoduje świecenie się
39 #czerwonej LEDY, a wartość zwracana przez sinik jest procentowa wartości intensywności
40 #odbitego światła (0-100%).
41 l_kolor.mode = "COL-REFLECT" #Ustawienie czujnika koloru w tryb "COL-REFLECT".
42
43 #KALIBRACJA
44 #Kalibracja polega na nauczeniu robota wartości koloru białego oraz czarnego dla poszczególnych czujników
45 #(światła/koloru). Najpierw definiowane są niezbędne zmienne, które służą do przechowywania aktualnych
46 #wczytanych wartości z czujników, zmienne przechowujące wypracowane wzorce kolorów oraz zmieniona która pozwala
47 #na prostą zmianę liczb pomiarów dla każdego koloru. Następnie każdy kolor próbowany jest odpowiednia liczba
48 #razy, po czym wyliczana jest jego wartość średnia. Kalibracja wymaga ręcznego przemieszczenia robota tak aby
49 #raz lapal tylko biały kolor a raz tylko czarny.
50 l_odczyt = 0 #Zmienna przechowująca aktualną wartość zwracaną przez czujnik koloru.
51 r_odczyt = 0 #Zmienna przechowująca aktualną wartość zwracaną przez czujnik światła.

```

```

52
53 l_bialy = 0      #Zmienna przechowujaca etalon koloru bialego dla czujnika koloru.
54 l_czarny = 0     #Zmienna przechowujaca etalon koloru czarnego dla czujnika koloru.
55 r_bialy = 0      #Zmienna przechowujaca etalon koloru bialego dla czujnika swiatla.
56 r_czarny = 0     #Zmienna przechowujaca etalon koloru czarnego dla czujnika swiatla.
57
58 l_pom = 10       #Zmienna określająca liczbę pomiarów z których zostanie obliczona średnia danego koloru.
59
60 sound.speak("CAL", True)          #Komunikat dźwiękowy oznaczający początek kalibracji.
61 sound.speak("WHITE", True)        #Komunikat dźwiękowy informujący że probkowany będzie kolor biały.
62 sleep(2)                         #Funkcja wstrzymująca program na 2 sekundy w celu dokładnego ustawienia sensorów po
63 otrzymaniu komunikatu.
64 for k in range(0,l_pom):          #Petla 'for' ktorej liczba wykonan odpowiada liczbie pomiarow.
65     l_odczyt = l_kolor.value()    #Odczyt wartości z czujnika kolorów i zapisanie jej do zmiennej l_odczyt.
66     r_odczyt = r_kolor.value()    #Odczyt wartości z czujnika światła i zapisanie jej do zmiennej r_odczyt.
67     l_bialy += l_odczyt           #Zwiększenie zmiennej l_bialy o wartość zmiennej l_odczyt.
68     r_bialy += r_odczyt           #Zwiększenie zmiennej r_bialy o wartość zmiennej r_odczyt.
69     sleep(0.05)                  #Funkcja sleep.
70 l_bialy = l_bialy/l_pom           #Obliczenie wartości średniej koloru białego dla czujnika koloru.
71 r_bialy = r_bialy/l_pom           #Obliczenie wartości średniej koloru białego dla czujnika światła.
72
73 sound.speak("BLACK", True)        #Komunikat dźwiękowy informujący że probkowany będzie kolor czarny.
74 sleep(2)                         #Funkcja wstrzymująca program na 2 sekundy w celu dokładnego ustawienia sensorów po
75 otrzymaniu komunikatu.
76 for k in range(0,l_pom):          #Petla 'for' ktorej liczba wykonan odpowiada liczbie pomiarow.
77     l_odczyt = l_kolor.value()    #Odczyt wartości z czujnika kolorów i zapisanie jej do zmiennej l_odczyt.
78     r_odczyt = r_kolor.value()    #Odczyt wartości z czujnika światła i zapisanie jej do zmiennej r_odczyt.
79     l_czarny += l_odczyt          #Zwiększenie zmiennej l_czarny o wartość zmiennej l_odczyt.
80     r_czarny += r_odczyt          #Zwiększenie zmiennej r_czarny o wartość zmiennej r_odczyt.
81     sleep(0.05)                  #Funkcja sleep.
82 l_czarny = l_czarny/l_pom          #Obliczenie wartości średniej koloru czarnego dla czujnika koloru.
83 r_czarny = r_czarny/l_pom          #Obliczenie wartości średniej koloru czarnego dla czujnika światła.
84
85 sound.speak("END", True)          #Komunikat informujący o zakończeniu procesu kalibracji.
86
87 #PROGRAM GŁÓWNY
88 #Proram główny dzieli się na dwie części. Część pierwsza zawiera definicje niezbędnych zmiennych wykorzystywanych
89 #przez zastosowane regulatory PID oraz czujniki odległości. Część druga realizuje jazdę robota po linii oraz omijanie
90 #przeszkody.
91 l_Kp = 50                          #Współczynnik wzmacniania członu proporcjonalnego regulatora PID silnika lewego.
92 l_Ki = 10                           #Współczynnik wzmacniania członu całkującego regulatora PID silnika lewego.
93 l_Kd = 0.3                          #Współczynnik wzmacniania członu różniczkującego regulatora PID silnika lewego.
94
95 r_Kp = 8.35                         #Współczynnik wzmacniania członu proporcjonalnego regulatora PID silnika prawego.
96 r_Ki = 1.67                          #Współczynnik wzmacniania członu całkującego regulatora PID silnika prawego.
97 r_Kd = 0.05                          #Współczynnik wzmacniania członu różniczkującego regulatora PID silnika prawego.
98
99 l_i_aw = 200                         #Anty windup członu całkującego silnika lewego.
100 r_i_aw = 2000                        #Anty windup członu całkującego silnika prawego.
100 Td = 0.01                           #Czas różniczkowania.

```

```

101 Ti = Td
102 wz_p = 250
103 i_pamiec = 10
104
105 l_PID = 0
106 l_P = 0
107 l_I = 0
108 l_D = 0
109 l_error = 0
110 l_last_error = 0
111 l_i_error = 0
112 l_ls_error = [0] * i_pamiec
113 l_wz = l_bialy
    czujnika koloru.
114
115 r_PID = 0
116 r_P = 0
117 r_I = 0
118 r_D = 0
119 r_error = 0
120 r_last_error = 0
121 r_i_error = 0
122 r_ls_error = [0] * i_pamiec
123 r_wz = r_bialy
    czujnika swiatla.
124
125 lmotor_ster = 0
126 rmotor_ster = 0
127 ogr_pred = 1000
128 wsp_pred = wz_p/125
    od predkosci zadanej.
129
130 d_irda = 0
131 d_sonar = 0
132 faza = 0
133 faza_ob = 0
134
135 while (irda.value() > 20):
    sleep(0.1)
    sleep(3)
136 #PETLA GLOWNA programu
137 #W petli glownej (nieskonczonej) programu, znajdują sie instrukcje pozwalajace robotowi na podazanie za linia badz tez objad przeskody.
138 #Podazanie za linia realizowane jest gdy spełniony jest warunek faza == 0, natomiast objezdaniu przeskody odpowiada warunek faza == 1.
139 #Program domyslnie pracuje w fazie 0, czyli podaza za linia. Dopiero wykrycie obiektu na jego drodze powoduje zmiane fazy na 1 w wyniku czego
140 #wlaczany jest podprogram objazdu przeskody.
141 #W celu realizacji zadania podazania za linia zastosowano dwa regulatory PID, oddzielne dla silnika prawego oraz silnika lewego.
142 #Kazdy z regulatorow dziala tak, aby robot jechal wzdluz czarnej lini. Czujniki umieszczone sa po lewej i prawej stronie lini

```

```

tak, aby lapac tylko bialy kolor.
145 #W przypadku, gdy czujnik koloru(czujnik lewego silnika) wykryje wjazd na czarna linie, regulator PID zmniejszy moc silnika
lewego i zwiększy moc silnika prawego.
146 #Anallogiczna sytuacja ma miejsce, gdy to czujnik swiatla najedzie na czarna linie. Silnik prawy ma zmniejszana moc, a lewy
zwiększa.
147 #Kazdy z regulator oddzialywuje na kazdy z silnikow jednoczesnie. Dodatkowo wskazania regulatorow sa wzmacniane przez
wspolczynnik predkosci, ktorego wartosc zalezy
148 #od wartosci zadanej predkosci. Wynika to z faktu, ze nastawy regulatorow zostaly dobrane dla wartosci zadanej predkosci
rownej 125. Przy braku wspomnianego wspolczynnika,
149 #przy wiekszych predkosciach robot nie reagował wystarczajaca mocno na zmiany wskazywane przez regulatory. Stad koniecznosc
skalowania wskazan regulatorow w zaleznosci
150 #od predkosci zadanej.
151 #W kazdym obiegu petli program dziala wedle nastepujacego algorytmu:
152 #1. Odczytaj wartosc z czujnika.
153 #2. Zapisz ostatni uchyb do zminnej przechowujacej ostatni uchyb.
154 #3. Oblicz nowy uchyb.
155 #4. Przesun tablice przechowujaca uchybu o 1 w lewo i na ostatnim miejscu zapisz aktualny uchyb.
156 #5. Oblicz skumulowany uchyb na potrzeby czlonu calkujacego.
157 #6. Sprawdz czy obliczony skumulowany blad nie przekracza wartosci granicznych (anty-windup).
158 #7. Oblicz czlon proporcjonalny regulatora.
159 #8. Oblicz czlon calkujacy regulatora.
160 #9. Oblicz czlon rozniczkujacy regulatora.
161 #10. Oblicz wartosc regulatora.
162 #11. Powtorz punkty 1-10 dla drugiego regulatora.
163 #12. Oblicz wartosc mocy silnikow w oparciu o wartosc zadana predkosci oraz wskazania regulatorow.
164 #13. Sprawdz czy obliczene moce nie wykraczaja poza wartosci graniczne (ograniczenie maksymalnej predkosci).
165 #14. Wysteruj silniki.
166 #15. Zapetl.
167 #W fazie 2 robot ma na celu objechanie przeszkody. W tym celu po wykryciu przeszkody okreca sie o 90 stopni w prawo i
rozpoczyna
168 #algorytm objazu przeszkody, ktory mozna zaprezentowac w nastepujacy sposob:
169 #1. Wykryto przeszkode
170 #2. Faza_ob == 0 - Obrot o 90 stopni
171 #3. Ustawienie fazy_ob == 1 i zatrzymanie silnikow
172 #4. Faza_ob == 1 - Sprawdz czy odleglosc od obiektu jest wieksza niz 30.
173 #Jezeli tak ustaw faze_ob == 2, jezeli nie ustaw faze_ob == 3.
174 #(sprawdzane jest w ten sposob, czy robot po wykryciu przeszkody widzi czujnikiem przeszkode czy tez nie).
175 #5. Faza_ob == 2 - Jedz prosto tak dlugo jak dystans jest wiekszy niz 30. Pozniej ustaw faza_ob = 3.
176 #6. Faza_ob == 3 - Jedz prosto tak dlugo jak dystans jest mniejszy niz 30. Pozniej ustaw faza_ob = 4.
177 #(faza_ob == 3 sluzi do okreslenia czy robot jedzie wzdluz pierwszej sciany czy tez juz ja omina.
178 #7. Faza_ob == 4 - Robot omija przeszkode i okreca sie w lewo o 90 stopni i ustawia faza_ob = 5.
179 #8. Faza_ob == 5 - Sprawdz czy odleglosc od obiektu jest wieksza niz 30.
180 #Jezeli tak ustaw faza_ob = 6, inaczej ustaw faza_ob = 7.
181 #9. Robot jedzie do przodu, tak dlugo az wykryje przeszkode w odleglosci mniejszej niz 30. Potem ustawia faza_ob = 7.
182 #10. Faza_ob == 7 - Jedz prosto, tak dlugo jak dystans od przeszkody jest mniejszy od 30. Potem ustaw faza_ob = 8.
183 #11. Faza_ob == 8 - Obroc robota o 90 stopni w lewo i ustaw faza_ob = 9.
184 #12. Faza_ob == 9 - Jedz prosto az czujnik swiatla natrafi na linie czarna. Wtedy obroc sie o 90 stopni w prawo
#po czym wrac do jazdy po lini.
185
186 while True:                                #Nieskonczona petla while.
187     d_irda = irda.value()                   #Odczytanie wartosci z czujnika podczerwieni.
188     if (d_irda < 20):                      #Sprawdzenie czy wartosc zwrocona przez czujnik jest

```

```

mniejsza od 20.
189     faza = 1
190 if faza == 0:
191     #PID DLA SILNIKA LEWEGO(CZUJNIK KOLORU)
192     l_odczyt = l_kolor.value()
193     zapisanie jej do zmiennej l_odczyt.
194     l_last_error = l_error
195     l_error = l_wz - l_odczyt
196
197     for k in range(0,i_pamiec):
198         i_pamiec.
199             if k < i_pamiec - 1:
200                 iteracji petli jest mniejsza od wartosci zmiennej i_pamiec
201                     l_ls_error[k] = l_ls_error[k+1]
202                     l_ls_error o jeden w lewo.
203             else:
204                 l_ls_error[k] = l_error
205
206         l_i_error = 0
207         for k in range(0,i_pamiec):
208             i_pamiec.
209                 l_i_error += l_ls_error[k]
210
211             if l_i_error > l_i_aw:
212                 dodatniej wartosci granicznej.
213                     l_i_error = l_i_aw
214                     dodatniej wartosci granicznej.
215             elif l_i_error < -l_i_aw:
216                 ujemnej wartosci granicznej.
217                     l_i_error = -l_i_aw
218                     ujemnej wartosci granicznej.
219
220             l_P = l_Kp * l_error
221             regulatora PID.
222             l_I = l_Ki * l_i_error * Ti * i_pamiec
223             l_D = l_Kd * (l_error - l_last_error) / Td
224             regulatora PID.
225
226             l_PID = l_P + l_I + l_D
227             #PID DLA SILNIKA PRAWEGO (CZUJNIK SWIATLA)
228             r_odczyt = r_kolor.value()
229             i_zapisanie jej do zmiennej r_odczyt.
230             r_last_error = r_error
231             r_error = r_wz - r_odczyt
232
233             for k in range(0,i_pamiec):
234                 i_pamiec.
235                     if k < i_pamiec - 1:
236                         iteracji petli jest mniejsza od wartosci zmiennej i_pamiec
237                             r_ls_error[k] = r_ls_error[k+1]
238                             r_ls_error o jeden w lewo.
239
240             #Jezeli tak zmien wartosc zmiennej faza na 1.
241             #Sprawdzenie czy wartosc zmiennej faza jest rowna 0.
242
243             #Odczytanie wartosci zwracanej przez czujnik koloru i
244
245             #Przepisanie uchybu do zmiennej l_last_error.
246             #Obliczenie nowego uchybu.
247
248             #Petla ktorej ilosc wykonan zalezy od zmiennej
249
250             #Sprawdzenie czy zmienna uzywana do kolejnych
251             #iteracji petli jest mniejsza od wartosci zmiennej i_pamiec - 1.
252             #Przesuniecie wartosci znajdujacych sie w liscie
253
254             #Wartosc zmiennej k >= wartosci zmiennej i_pamiec.
255             #Zapisanie na ostatnim miejscu listy aktualnego uchybu.
256
257             #Wyzeroowanie skumulowanego uchybu.
258             #Petla ktorej ilosc wykonan zalezy od zmiennej
259
260             #Obliczenie skumulowanego uchybu.
261
262             #Sprawdzenie czy skumulowany uchyb nie przekracza
263
264             #Przypisanie do zmiennej skumulowanego uchybu
265
266             #Sprawdzenie czy skumulowany uchyb nie przekracza
267
268             #Przypisanie do zmiennej skumulowanego uchybu
269
270             #Obliczenie wartosci czlonu proporcjonalnego
271
272             #Obliczenie wartosci czlonu calkujacego regulatora PID.
273             #Obliczenie wartosci czlonu rozniczkujacego
274
275             #Obliczenie wartosci regulatora PID.
276
277             #Odczytanie wartosci zwracanej przez czujnik swiatla
278
279             #Przepisanie uchybu do zmiennej r_last_error.
280             #Obliczenie nowego uchybu.
281
282             #Petla ktorej ilosc wykonan zalezy od zmiennej
283
284             #Sprawdzenie czy zmienna uzywana do kolejnych
285             #iteracji petli jest mniejsza od wartosci zmiennej i_pamiec - 1.
286             #Przesuniecie wartosci znajdujacych sie w liscie

```

```

224
225         r_ls_error[k] = r_error
226
227         r_i_error = 0
228         for k in range(0,i_pamiec):
229             i_pamiec.
230                 r_i_error += r_ls_error[k]
231
232         if r_i_error > r_i_aw:
233             dodatniej wartosci granicznej.
234                 r_i_error = r_i_aw
235             dodatniej wartosci granicznej.
236             elif r_i_error < -r_i_aw:
237                 ujemnej wartosci granicznej.
238                 r_i_error = -r_i_aw
239                 ujemnej wartosci granicznej.
240
241             r_P = r_Kp * r_error
242             regulatora PID.
243             r_I = r_Ki * r_i_error * Ti * i_pamiec
244             r_D = r_Kd * (r_error - r_last_error) / Td
245             regulatora PID.
246
247             r_PID = r_P + r_I + r_D
248             #WYSTEROWANIE SILNIKOW
249             lmotor_ster = wz_p - l_PID * wsp_pred + r_PID * wsp_pred
250             wartosc zadana predkosci oraz wskazania regulatorow.
251             rmotor_ster = wz_p - r_PID * wsp_pred + l_PID * wsp_pred
252             wartosc zadana predkosci oraz wskazania regulatorow.
253
254             if lmotor_ster > ogr_pred:
255                 lewego nie przekracza dodatniej wartosci granicznej.
256                 lmotor_ster = ogr_pred
257                 dodatniej wartosci granicznej.
258             elif lmotor_ster < -ogr_pred:
259                 lewego nie przekracza ujemnej wartosci granicznej.
260                 lmotor_ster = -ogr_pred
261                 ujemnej wartosci granicznej.
262
263             if rmotor_ster > ogr_pred:
264                 prawego nie przekracza dodatniej wartosci granicznej.
265                 rmotor_ster = ogr_pred
266                 dodatniej wartosci granicznej.
267             elif rmotor_ster < -ogr_pred:
268                 prawego nie przekracza ujemnej wartosci granicznej.
269                 rmotor_ster = -ogr_pred
270                 ujemnej wartosci granicznej.
271
272             lmotor.run_forever(speed_sp = int(lmotor_ster))
273             rmotor.run_forever(speed_sp = int(rmotor_ster))

```

#Wartosc zmiennej k >= wartosci zmiennej i_pamiec.
#Zapisanie na ostatnim miejscu listy aktualnego uchybu.

#Wyzeroowanie skumulowanego uchybu
#Petla ktorej ilosc wykonan zależy od zmiennej
#Obliczenie skumulowanego uchybu.

#Sprawdzenie czy skumulowany uchyb nie przekracza
#Przypisanie do zmiennej skumulowanego uchybu

#Sprawdzenie czy skumulowany uchyb nie przekracza
#Przypisanie do zmiennej skumulowanego uchybu

#Obliczenie wartosci czlonu proporcjonalnego
#Obliczenie wartosci czlonu calkujacego regulatora PID.
#Obliczenie wartosci czlonu rozniczkujacego

#Obliczenie wartosci regulatora PID.
#Oblicz wartosc mocy silnika lewego w oparciu o
#Oblicz wartosc mocy silnika prawego w oparciu o

#Sprawdzenie czy obliczona wartosc predkosci silnika
#Przypisanie do zmiennej predkosci silnika lewego

#Sprawdzenie czy obliczona wartosc predkosci silnika
#Przypisanie do zmiennej predkosci silnika lewego
#

#Sprawdzenie czy obliczona wartosc predkosci silnika
#Przypisanie do zmiennej predkosci silnika prawego

#Sprawdzenie czy obliczona wartosc predkosci silnika
#Przypisanie do zmiennej predkosci silnika prawego

#Wysteronanie silnika lewego.
#Wysteronanie silnika prawego.

```

258     if (_kolor.value() < 1.25*l_czarny):           #Sprawdzenie czy wartosc zwracana przez czujnik
259         koloru jest wieksza niz 1.25 wartosci etalonu koloru czarnego dla silnika lewego.
260         if (r_kolor.value() < 1.25*r_czarny):       #Sprawdzenie czy wartosc zwracana przez czujnik
261             swiatla jest wieksza niz 1.25 wartosci etalonu koloru czarnego dla silnika prawego.
262             lmotor.run_forever(speed_sp = wz_p)      #Ustawienie predkosci silnika lewego na wartosc
263             zapisana w zmiennej predkosci zadanej.
264             rmotor.run_forever(speed_sp = wz_p)        #Ustawienie predkosci silnika prawego na wartosc
265             zapisana w zmiennej predkosci zadanej.
266             sleep(0.5)                                #Wstrzymanie wykonywania programu na 0.5 sekundy.
267             sleep(0.01)                               #Wstrzymanie wykonywania programu na 0.01 sekundy.
268         if faza == 1:                            #Sprawdzenie czy wartosc zmiennej faza jest rowna 1.
269             if (faza_ob == 0):                      #Sprawdzenie czy wartosc zmiennej faza_ob jest rowna 0.
270                 lmotor.run_forever(speed_sp = 200)   #Ustawienie predkosci silnika lewego na wartosc 200.
271                 rmotor.run_forever(speed_sp = -200)  #Ustawienie predkosci silnika prawego na wartosc -200.
272                 faza_ob = 1                         #Ustawienie wartosci zmiennej faza_ob na 1.
273                 sleep(1)                           #Wstrzymanie wykonywania programu na 1 sekunde.
274                 lmotor.run_forever(speed_sp = 0)    #Zatrzymanie silnika lewego.
275                 rmotor.run_forever(speed_sp = 0)    #Zatrzymanie silnika prawego.
276             if (faza_ob == 1):                      #Sprawdzenie czy wartosc zmiennej faza_ob jest rowna 1.
277                 if (sonar.value > 30):            #Sprawdzenie czy wartosc zwracana przez czujnik
278                     ultradzwiekowy jest wieksza niz 30. #Ustawienie wartosci zmiennej faza_ob na 2.
279                     faza_ob = 2                   #Wartosc zwracana przez czujnik ultradzwiekowy jest
280             if (faza_ob == 2):                      #Ustawienie wartosci zmiennej faza_ob na 3.
281                 lmotor.run_forever(speed_sp = 200)   #Sprawdzenie czy wartosc zmiennej faza_ob jest rowna 2.
282                 rmotor.run_forever(speed_sp = 200)   #Ustawienie predkosci silnika lewego na wartosc 200.
283                 while (sonar.value() > 30):        #Ustawienie predkosci silnika prawego na wartosc 200.
284                     ultradzwiekowy jest wieksza niz 30. #Sprawdzenie czy wartosc zwracana przez czujnik
285                     sleep(0.0001)                  #Wstrzymanie wykonywania programu na 0.0001 sekundy.
286                     faza_ob = 3                   #Ustawienie wartosci zmiennej faza_ob na 3.
287                     sleep(0.2)                  #Wstrzymanie wykonywania programu na 0.2 sekundy.
288             if (faza_ob == 3):                      #Sprawdzenie czy wartosc zmiennej faza_ob jest rowna 3.
289                 lmotor.run_forever(speed_sp = 200)   #Ustawienie predkosci silnika lewego na wartosc 200.
290                 rmotor.run_forever(speed_sp = 200)   #Ustawienie predkosci silnika prawego na wartosc 200.
291                 while (sonar.value() < 30):        #Sprawdzenie czy wartosc zwracana przez czujnik
292                     ultradzwiekowy jest mniejsza niz 30. #Wstrzymanie wykonywania programu na 0.0001 sekundy.
293                     sleep(0.0001)                  #Ustawienie wartosci zmiennej faza_ob na 4.
294                     faza_ob = 4                   #Sprawdzenie czy wartosc zmiennej faza_ob jest rowna 4.
295                     sleep(0.25)                  #Wstrzymanie wykonywania programu na 0.25 sekundy.
296                     lmotor.run_forever(speed_sp = -200) #Ustawienie predkosci silnika lewego na wartosc -200.
297                     rmotor.run_forever(speed_sp = 200) #Ustawienie predkosci silnika prawego na wartosc 200.
298                     sleep(1)                    #Wstrzymanie wykonywania programu na 1 sekunde.
299             if (faza_ob == 5):                      #Zatrzymanie silnika lewego.
299             if (sonar.value() > 30):            #Zatrzymanie silnika prawego.
300                 ultradzwiekowy jest wieksza niz 30. #Ustawienie wartosci zmiennej faza_ob na 5.
300                 sleep(0.25)                  #Sprawdzenie czy wartosc zmiennej faza_ob jest rowna 5.
300                 if (sonar.value() > 30):            #Sprawdzenie czy wartosc zwracana przez czujnik
300                     ultradzwiekowy jest wieksza niz 30. #Wstrzymanie wykonywania programu na 0.25 sekundy.

```

```

300         faza_ob = 6
301     else:
302         mniejsza niz 30.
303         faza_ob = 7
304     if (faza_ob == 6):
305         lmotor.run_forever(speed_sp = 200)
306         rmotor.run_forever(speed_sp = 200)
307         while (sonar.value() > 30):
308             ultradzwiekowy jest wieksza niz 30.
309             sleep(0.0001)
310             faza_ob = 7
311             sleep(0.2)
312     if (faza_ob == 7):
313         lmotor.run_forever(speed_sp = 200)
314         rmotor.run_forever(speed_sp = 200)
315         while (sonar.value() < 30):
316             ultradzwiekowy jest mniejsza niz 30.
317             sleep(0.0001)
318             faza_ob = 8
319     if (faza_ob == 8):
320         sleep(0.25)
321         lmotor.run_forever(speed_sp = -200)
322         rmotor.run_forever(speed_sp = 200)
323         sleep(1)
324         lmotor.run_forever(speed_sp = 0)
325         rmotor.run_forever(speed_sp = 0)
326         faza_ob = 9
327     if (faza_ob == 9):
328         lmotor.run_forever(speed_sp = 200)
329         rmotor.run_forever(speed_sp = 200)
330         while (r_kolor.value() > 0.8*r_bialy):
331             zwarcana przez czujnik swiatla bedzie wieksza niz 0.8 wartosci etalonu koloru bialego dla silnika prawnego.
332             sleep(0.01)
333             faza_ob = 0
334         lmotor.run_forever(speed_sp = 200)
335         rmotor.run_forever(speed_sp = -200)
336         sleep(1)
337         faza = 0

```

#Ustawienie wartosci zmiennej faza_ob na 6.
#Wartosc zwracana przez czujnik ultradzwiekowy jest
#Ustawienie wartosci zmiennej faza_ob na 7.
#Sprawdzenie czy wartosc zmiennej faza_ob jest rowna 6.
#Ustawienie predkosci silnika lewego na wartosc 200.
#Ustawienie predkosci silnika prawego na wartosc 200.
#Sprawdzenie czy wartosc zwracana przez czujnik
#Wstrzymanie wykonywania programu na 0.0001 sekundy.
#Ustawienie wartosci zmiennej faza_ob na 7.
#wstrzymanie wykonywania programu na 0.2 sekundy.
#Sprawdzenie czy wartosc zmiennej faza_ob jest rowna 7.
#Ustawienie predkosci silnika lewego na wartosc 200.
#Ustawienie predkosci silnika prawego na wartosc 200.
#Sprawdzenie czy wartosc zwracana przez czujnik
#Wstrzymanie wykonywania programu na 0.0001 sekundy.
#Ustawienie zmiennej faza_ob na 8.
#Sprawdzene czy wartosc zmiennej faza_ob jest rowna 8.
#Wstrzymanie wykonywania programu na 0.25 sekundy.
#Ustawienie predkosci silnika lewego na wartosc -200.
#Ustawienie predkosci silnika prawego na wartosc 200.
#Wstrzymanie wykonywania programu na 1 sekunde.
#Zatrzymanie silnika lewego.
#Zatrzymanie silnika prawego.
#Ustawienie wartosci zmiennej faza_ob na 9.
#Sprawdzenie czy wartosc zmiennej faza_ob jest rowna 9.
#Ustawienie predkosci silnika lewego na wartosc 200.
#Ustawienie predkosci silnika prawego na wartosc 200.
#Petla while wykonywana tak dlugo, jak wartosc
#Wstrzymanie wykonywania programu na 0.1 sekundy.
#Ustawienie wartosci zmiennej faza_ob na 0.
#Ustawienie predkosci silnika lewego na wartosc 200.
#Ustawienie predkosci silnika prawego na wartosc -200.
#Wstrzymanie wykonywania programu na 1 sekunde.
#Ustawienie wartosci zmiennej faza na 0.