

# Robot typu line-follower

*sterowany poprzez złącze LPT*

*grupa I2.1, Informatyka 2013*

Imię i nazwisko	Numer indeksu	Rola w zespole
Mateusz Czajka	106596	Mechanika
Marcin Marszałek	106549	Lutownictwo, Elektronika
Mateusz Palichleb	94351	Sprawozdanie
Jakub Peikert	106618	Sprawozdanie
Tomasz Pewiński	106638	Sprawozdanie
Piotr Skonieczny	106650	Elektronika, Koordynator projektu
Adam Szczepański	106593	Elektronika, Oprogramowanie, Lider grupy
Michał Żelazkiewicz	106636	Trasa, Instalacja systemu Windows 95

# Spis treści

[Wstęp](#)

[Uwagi początkowe](#)

[Zarys projektu](#)

[Specyfikacja projektu](#)

[Mechanika](#)

[Opis](#)

[Parametry fizyczne](#)

[Zdjęcia](#)

[Elektrotechnika](#)

[Napęd](#)

[Zasilanie](#)

[Elektronika](#)

[Czujniki](#)

[Komparatory](#)

[Sterowanie silnikami](#)

[Płytnica drukowana](#)

[Złącze LPT](#)

[Schemat](#)

[Oprogramowanie](#)

[Listing kodu](#)

[8259](#)

[8253](#)

[Omówienie algorytmu sterującego](#)

[Serwisowanie i uruchamianie](#)

[Podsumowanie](#)

[Analiza SWOT](#)

[Kosztorys](#)

[Uwagi końcowe](#)

# **Wstęp**

## **Uwagi początkowe**

W związku z zadanymi projektami z przedmiotu Architektura Systemów Komputerowych postanowiliśmy podzielić naszą podgrupę na dwie części. Naszemu zespołowi przypadło stworzenie robota typu line-follower. Po zapoznaniu się ze specyfikacją projektu postanowiliśmy się spotkać w celu ustalenia kolejnych kroków i podziału zadań. Pierwszym wyzwaniem było zdobycie wszystkich wymaganych części i odpowiedniego sprzętu. Po przeszukaniu strychów, piwnic i po zamówieniu reszty potrzebnych elementów w sklepie internetowym rozpoczęliśmy naszą przygodę z budowaniem robota. W ciągu następnych trzech tygodni pracowaliśmy ciężko przez kilkadziesiąt godzin, żeby go stworzyć i zaprogramować. Z uwagi na charakterystyczny kształt naszego robota, postanowiliśmy nazwać go "Skorpion".

## **Zarys projektu**

Projekt polegał na zbudowaniu robota typu line-follower. Zadaniem robota jest podążanie za wyznaczoną na ziemi na białym tle, czarną linią o szerokości 19mm. Sterowanie robotem odbywa się w komputerze PC, poprzez złącze LPT.

# Specyfikacja projektu

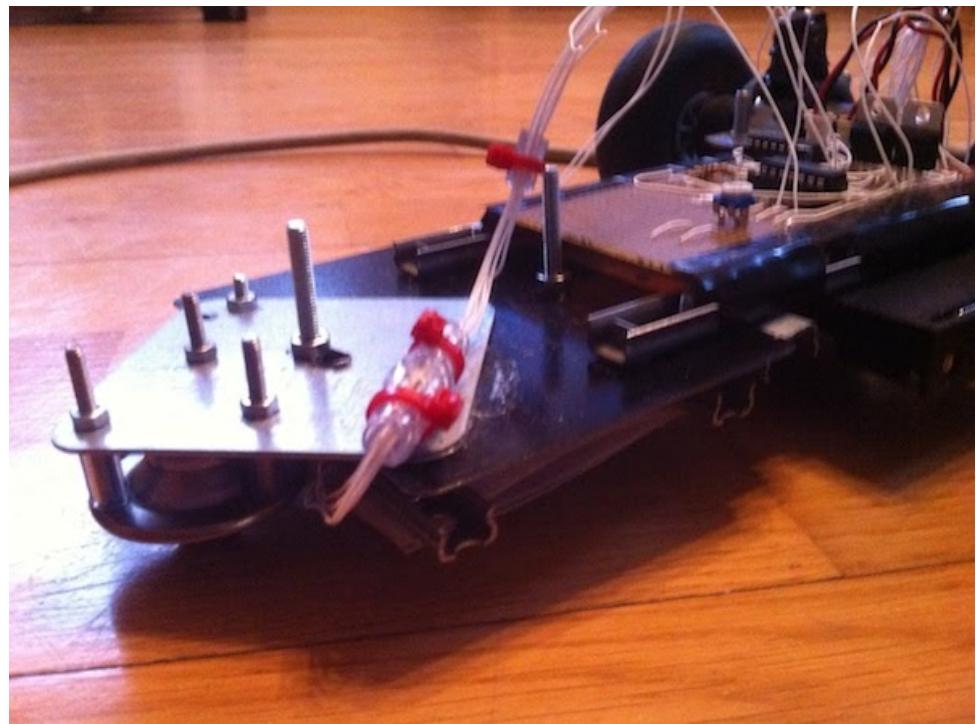
## Mechanika

### Opis

Konstrukcja robota oparta została na metalowym stelażu. Rozważane były również konstrukcje z użyciem LEGO, jednak zostały odrzucone z powodu zbyt wysokich kosztów. Konstrukcja ostatecznie bazuje na zestawie metalowych belek i płyt. Założeniem było stworzenie jak najniższej konstrukcji wraz z jak najniższym środkiem ciężkości - udało się to poprzez zamontowanie koszyków z bateriami poniżej linii silników. Zadbano także o regulowaną odległość diód od podłoga (odbywa się to dzięki śrubom z elastycznymi podkładkami z węża ogrodowego).



W tylnej części zamontowano złącze LPT, dwa silniki prądu stałego model HL149 oraz dwa koła.



W przedniej części zamontowano wysięgnik z zestawem czterech czujników CNY70 oraz kulką podporową





Pod spodem konstrukcji podwieszone zostały 2 koszyczki na 4 baterie typu AA.



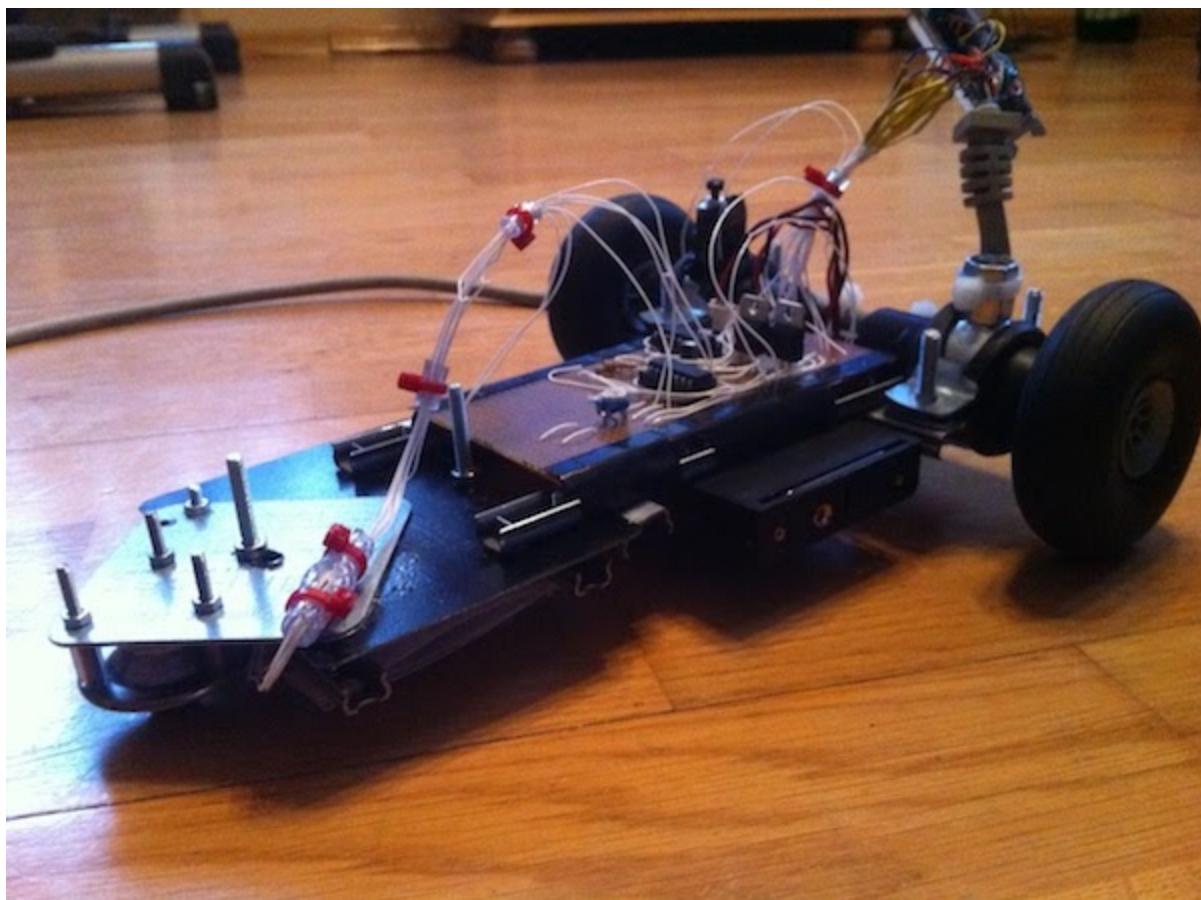
Na wierzchu stelażu przyjmocowano płytke drukowaną z elementami elektronicznymi

## Parametry fizyczne

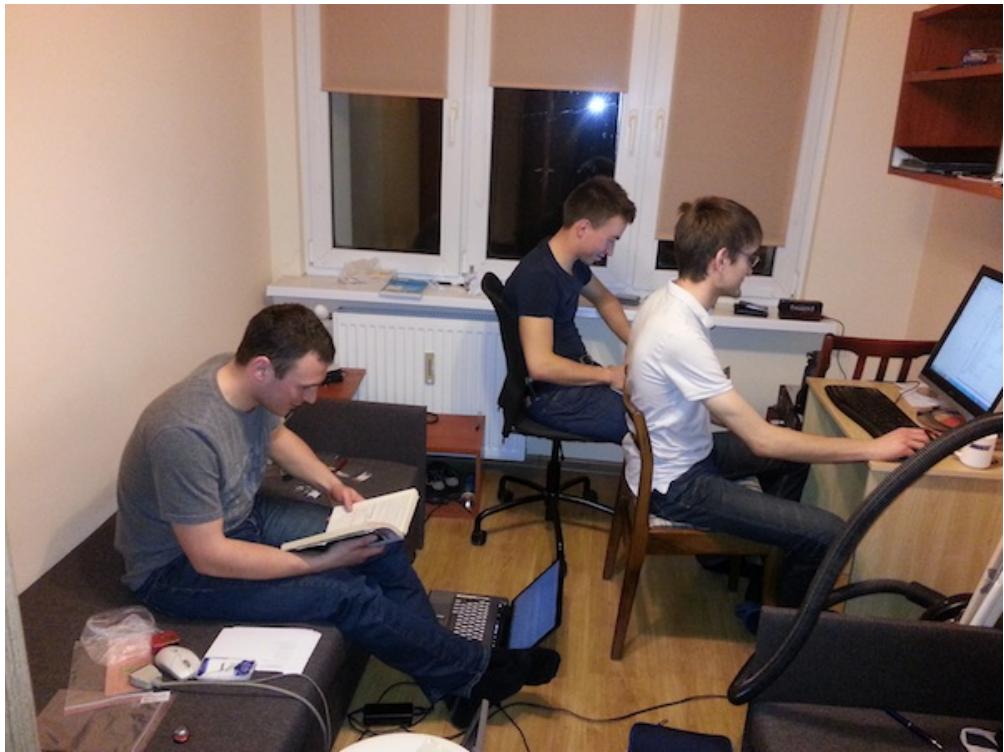
Parametry fizyczne zestawione zostały w poniższej tabelce.

szerokość	215 mm
długość	275 mm
wysokość	130 mm (najwyższe miejsce - LPT)

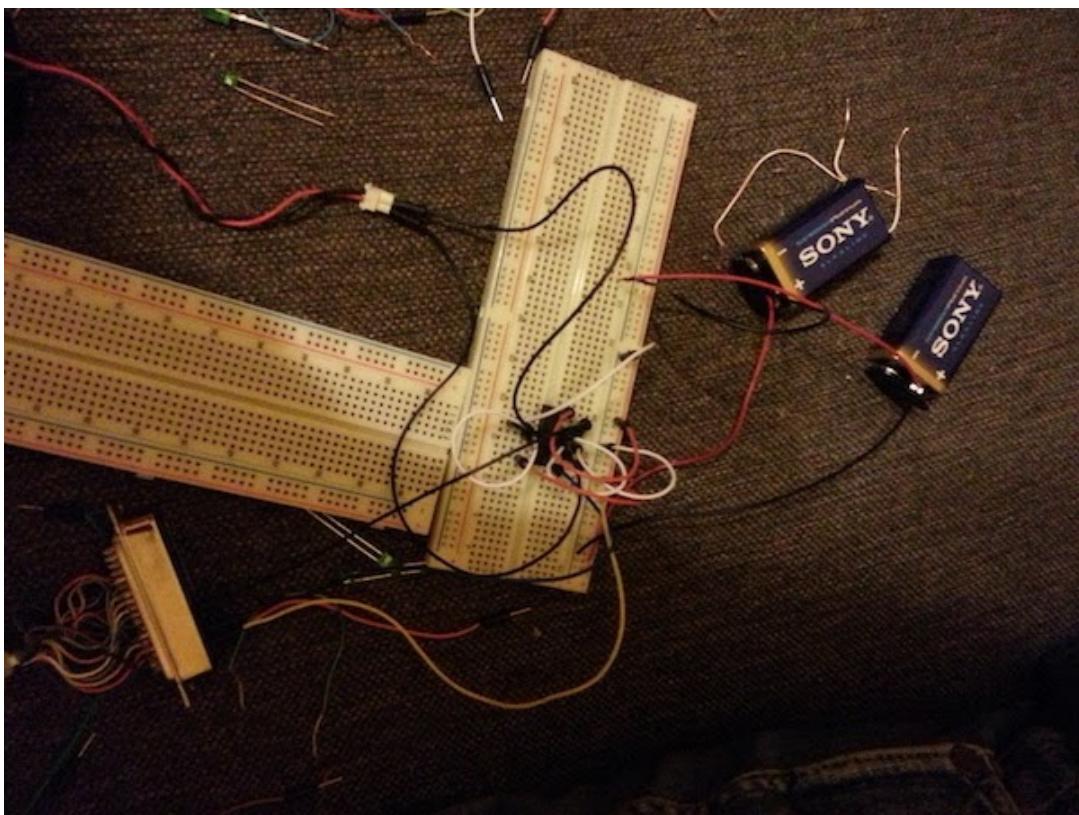
## Zdjęcia



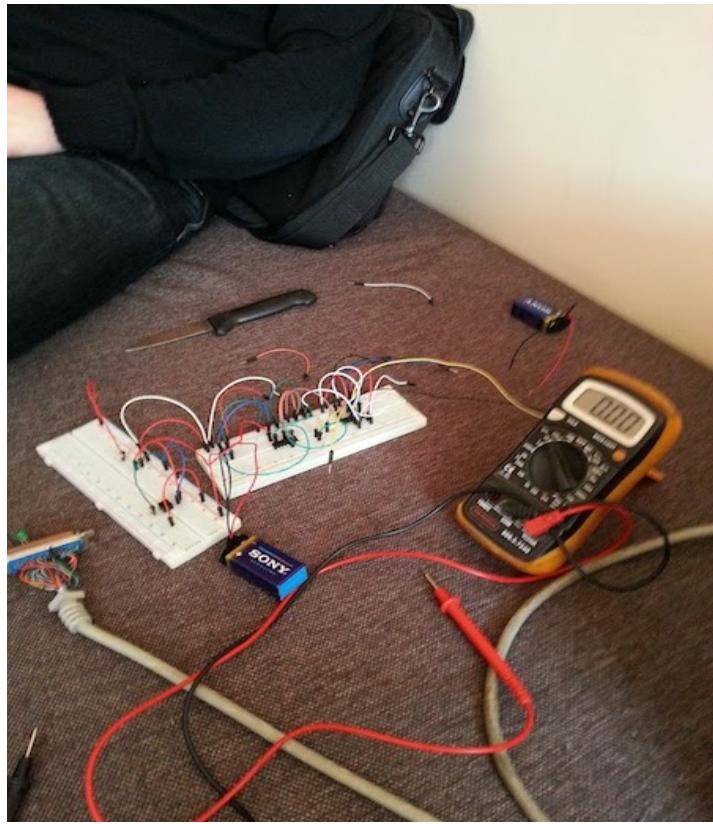
Gotowy, w pełni funkcjonalny robot



Ciężkie i mozolne początki



Pierwsze testy mostka



Pomiary napięc



Lutowanie



Konstrukcja stelażu

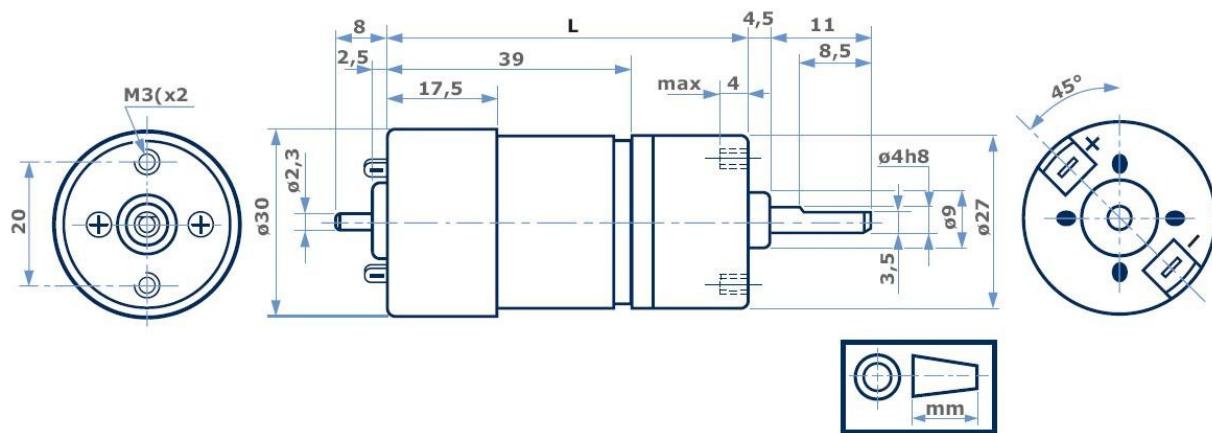
## Elektrotechnika

### Napęd

Jako napęd zastosowaliśmy dwa silniki (po jednym dla każdego koła) HL149 z przekładnią zębatą 10:1 o następującej charakterystyce:

napięcie zasilania	6V
maksymalny pobór prądu	210mA
moment obrotowy	40mNm
masa	100g
prędkość obrotowa biegu jałowego	315 obr/min

Schemat silnika:



Silniki są połączone bezpośrednio z kołami. Zamocowane są one z tyłu line followera przy pomocy dwóch obejm, tworząc jedną oszę.

Napotkaliśmy nieprzewidziany problem, gdyż okazało się, iż średnica osi kół jest większa od średnicy osi silnika, czemu zaradziliśmy poprzez wywiercenie otworu w feldze i przymocowanie śrubą.

Problemem okazało się również uzyskanie odpowiednio wysokiego napięcia na silnikach, czemu udało się zaradzić (kosztem przepalonego zestawu przewodów) za pomocą odpowiedniego połączenia uziemień i zastosowania stabilizatora 8V.

Sterowanie odbywa się poprzez nadawanie odpowiedniego napięcia na dany silnik - im jest ono większe tym większą osiąga prędkość obrotową - robot skręca poprzez odpowiednie dobranie różnych napięć (prędkości) dla każdego z koła.

## Zasilanie

Za źródło zasilania posłużyło nam 8 ogniw elektrycznych R6, AA o napięciu 1.5V umieszczone w dwóch koszykach na baterie, połączonych ze sobą szeregowo.

Cały układ zasilania posiada łączną masę 150g.



Zdecydowaliśmy się na tego rodzaju układ z uwagi na: niskie koszta, łatwość w montażu, estetykę, łatwość serwisowania i wymiany wyczerpanych ogniw, niską awaryjność. Wcześniej rozważany był układ dwóch baterii 6LR61 o napięciu 9V, ale okazał się on gorszy pod względem wymienionych wyżej czynników.

Zdecydowaliśmy się na tego rodzaju układ, gdyż myśleliśmy, iż będzie on wydajny. Jednak mimo łatwości w montażu i serwisowaniu, niskich kosztów oraz estetyki takiego systemu zasilającego, okazał się on dużym błędem, gdyż szybko ulega wyczerpaniu.

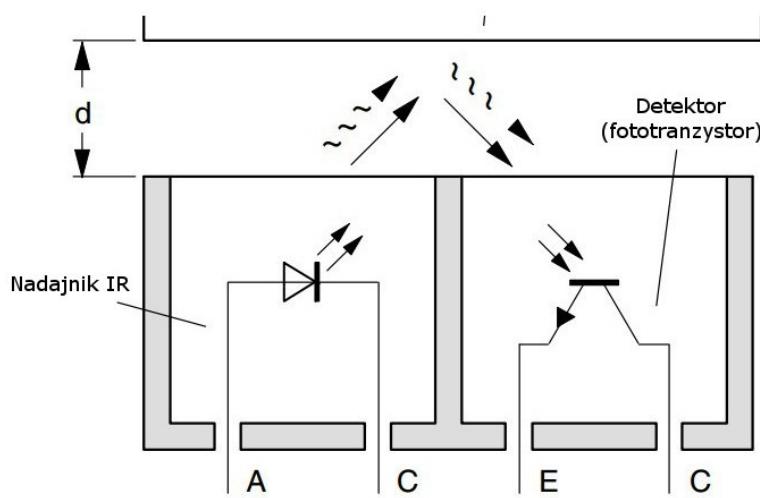
Najlepszym rozwiązaniem naszych problemów byłby akumulator 11.1V, jednakże doszliśmy do tego wniosku zbyt późno i nie byliśmy w stanie wymienić naszego źródła zasilania na wydajniejszy.

# Elektronika

## Czujniki

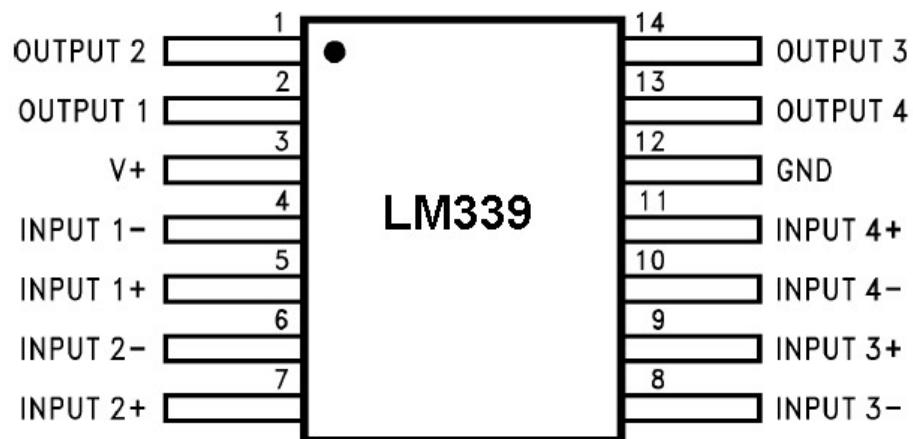
Za rozpoznawanie trasy odpowiada zestaw czterech czujników CNY70, umieszczonych z przodu konstrukcji, 6mm nad ziemią. Model ten został wybrany ze względu na dużą dostępność, optymalne parametry fizyczne i korzystną cenę.

Czujnik wysyła wiązkę promieniowania poprzez nadajnik podczerwieni, a następnie za pomocą fototranzystora, mierzy natężenie światła odbitego. Wyjściem jest sygnał napięciowy, zależny od natężenia światła padającego na ten detektor. Im więcej światła się odbije i dotrze do fotodetektora tym napięcie na wyjściu będzie miało wyższą wartość. Jako że promieniowanie świetlne lepiej odbija powierzchnia jasna (a ciemna pochłania), dlatego napięcie będzie wyższe na białym materiale.



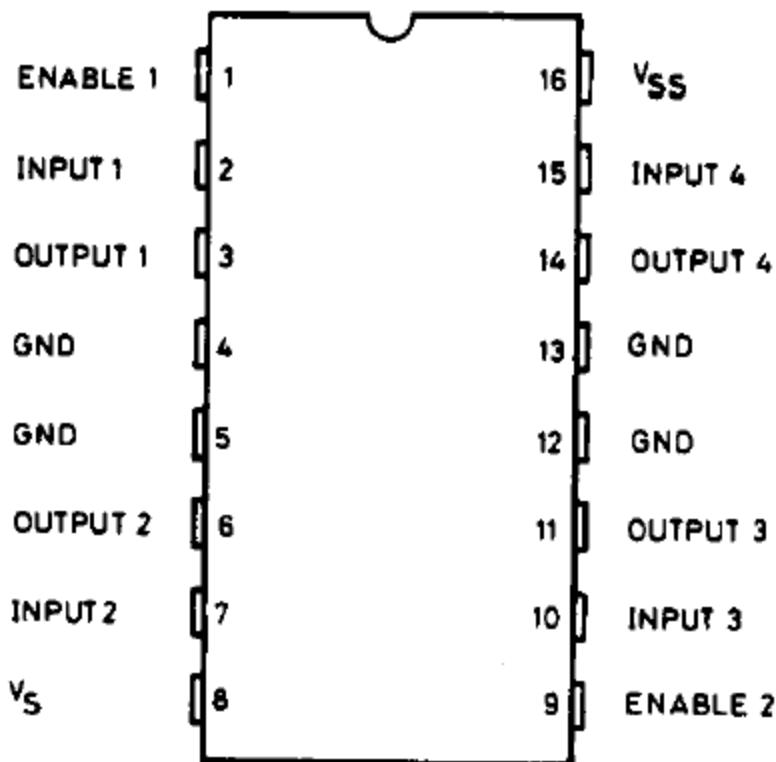
## Komparatory

Do uzyskania wartości logicznej z pomiaru czujników użyto komparatorów LM339. Na jedno wejście podajemy sygnał z czujników, na drugie napięcie odniesienia, które będzie progiem (regulowane za pomocą potencjometru). Na wyjściu otrzymujemy sygnał cyfrowy, który możemy bezpośrednio podłączyć do pinu LPT.



## Sterowanie silnikami

Zastosowany przez nas mostek to dwukanałowy sterownik silników L293D.



Na wejście Vss podajemy napięcie 5V ze stabilizatora, dla układu logicznego.

Na wejście Vs podajemy napięcie 8V ze stabilizatora, dodatkowo zabezpieczone kondensatorem aby chronić przed skokami napięcia.

Chip pozwala na sterowanie parą silników. Piny output podłączone są bezpośrednio do nich. Piny input pozwalają na sterowanie kierunkiem obrotu.

Input 1	Input 2	Kierunek
HIGH	HIGH	stan niewykorzystywany
HIGH	LOW	przód
LOW	HIGH	tył
LOW	LOW	stan niewykorzystywany

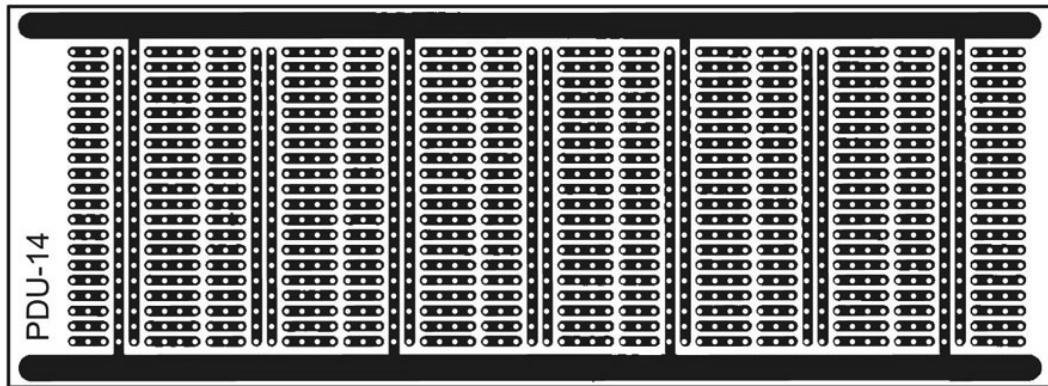
Wejście ENABLE steruje prędkością silników z użyciem metody PWM (pulse-width modulation). Polega ona na odpowiednim wypełnieniu sygnałem HIGH 50 cykli w częstotliwości 819Hz. Dla przykładu 10 cykli HIGH i 40 cykli LOW oznacza 20% prędkości. Metoda ta pozwala na łatwe sterowanie prędkością silnika.

## Płytki drukowane

Połączenia elementów elektronicznych dokonano na płytce PDU-14. Dzięki przecięciu płytki na dwie części mogliśmy umieścić czujniki oddzielnie od pozostałego układu.

Ze względu na ograniczenia czasowe oraz możliwość błędu nie zdecydowaliśmy się na wytrawianie płytki własnoręcznie.

Przed przylutowaniem układu na stałe, jego działanie zostało przetestowane z użyciem płytki stykowej.



## Złącze LPT

Do komunikacji z PC wykorzystano złącze LPT Centronics.

Poniżej zamieszczono wykaz użytych pinów:

pin	typ	użycie
2	wyjście	D0
3	wyjście	D1
4	wyjście	D2
6	wyjście	D4
7	wyjście	D5
8	wyjście	D6
10	wejście	ACK
11	wejście	S3
12	wejście	S4
13	wejście	S5
19	wyjście	GND
21	wyjście	GND
32	wejście	S7

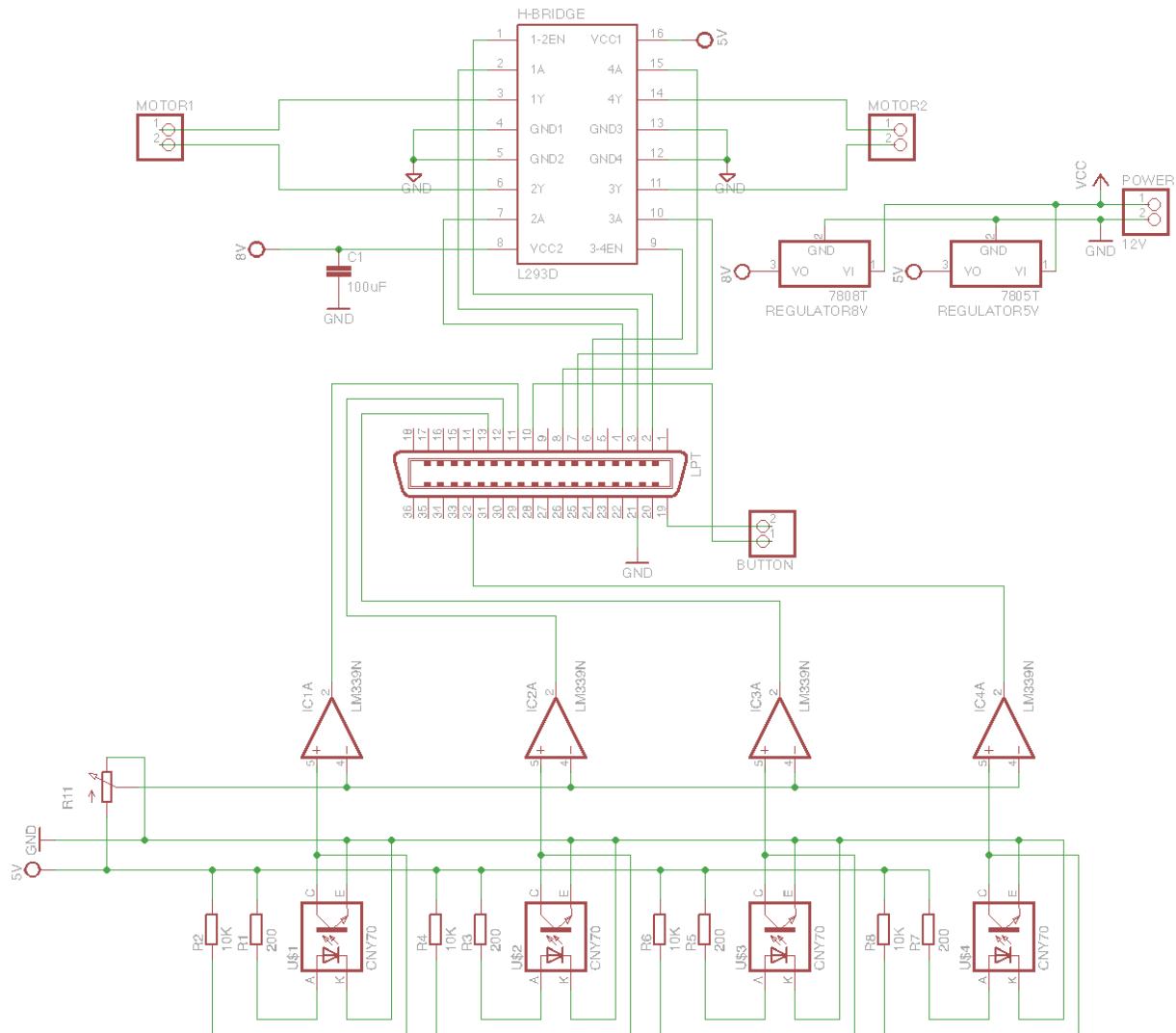
Pin oznaczone D połączone są z mostkiem H i służą do sterowania silnikami (por. sekcja "sterowanie silnikami"). D0,1,2 odpowiadają za prawy silnik, D4,5,6 za lewy silnik.

Pin oznaczone S służą do odczytu stanu logicznego czujników.

Pin ACK służy do uruchomienia robota poprzez przerwanie.

## Schemat

Kompletny schemat, wykonany w programie Eagle, zamieszczono poniżej:



# Oprogramowanie

## Listing kodu

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <conio.h>
4. #include <dos.h>
5.
6. //adresy portu LPT
7. #define DATA 0x378
8. #define STATUS 0x379
9. #define CONTROL 0x37A
10. //adres przerwania generowanego przez 8253
11. #define IRQ0 0x08
12. //adres przerwania generowanego przez LPT (~ACK)
13. #define IRQ7 0x0F
14.
15. //wspolczynniki PID
16. #define kp 17.5
17. #define ki 0.55
18. #define kd 2.6
19. #define Tp 17.0
20. float P,I,D,dif,pdif,rate,turn;
21.
22. typedef unsigned short WORD;
23.
24. void interrupt far (*oldIntIRQ0)();
25. void interrupt far newIntIRQ0();
26. void interrupt far (*oldIntIRQ7)();
27. void interrupt far newIntIRQ7();
28.
29. void setup8259();
```

```
30. void reset8259();
31. void setup8253();
32. void reset8253();
33.
34. WORD newFreq = 819;           // 45x wolniej niz zegar systemowy
35.                                         // dobierac tak zeby newFreq/18.2
   bylo prawie calkowite
36. WORD freqModifier;
37. WORD newTimerFreq;
38.
39. WORD maxPWM, counterPWM;
40. short left_motor=0, right_motor=0;
41. void calculate_pos();
42. float get_sensors();
43. void set_motors();
44.
45. short confirmed;
46.
47. int main() {
48.     confirmed=0;
49.     printf("LPT ROBOT I2.1\n");
50.
51.     dif=0;
52.     I=0;
53.
54.     maxPWM = 50;
55.     counterPWM = 0;
56.
57.     freqModifier = newFreq/18.2;
58.     newTimerFreq = 1193180/newFreq;
59.
60.     setup8253();
61.     setup8259();
```

```
62.  
63.     printf("Nacisnij przycisk na robocie\n");  
64.  
65.     // teraz robot jezdzi  
66.  
67.     getch();  
68.  
69.     reset8259();  
70.     reset8253();  
71.  
72.     // zatrzymanie pracy silników  
73.     outportb(DATA, 0x00);  
74.  
75.     printf("Nacisnij przycisk zeby wyjsc...\n");  
76.     getch();  
77.  
78.     return 0;  
79. }  
80.  
81. void setup8259() {  
82.     // wyłączenie przerwan zeby nic nie przeszkadzało  
83.     asm cli;  
84.  
85.     // pobranie starego wektora przerwan generowanych przez 8253  
86.     oldIntIRQ0 = getvect(IRQ0);  
87.     // ustawienie nowego wektora przerwan  
88.     setvect(IRQ0, nwqIntIRQ0);  
89.  
90.     // pobranie starego wektora przerwan generowanych przez  
91.     // wejście ~ACK na LPT  
92.     oldIntIRQ7 = getvect(IRQ7);  
93.     // ustawienie nowego wektora przerwan  
94.     setvect(IRQ7, newIntIRQ7);
```

```
95.     // odmaskowanie przerwania IRQ7 na 8259
96.     outportb(0x21, (inportb(0x21) & 0x7F));
97.     // ustawienie 4 bitu kontrolnego
98.     // aby wlaczyc przerwania na LPT
99.     outportb(CONTROL, inportb(CONTROL) | 0x10);
100.
101.    // wlaczenie przerwan
102.    asm sti;
103. }
104.
105. void reset_8259() {
106.     // wylaczenie przerwan zeby nic nie przeszkadzalo
107.     asm cli;
108.
109.     // ustawienie starych wektorow przerwan
110.     setvect(IRQ0, oldIntIRQ0);
111.     setvect(IRQ7, oldIntIRQ7);
112.
113.     // wlaczenie przerwan
114.     asm sti;
115. }
116.
117. void setup8253() {
118.     // ustawienie nowej czestotliwosci zegara
119.     asm mov dx, newTimerFreq;
120.
121.     // bity 7-6 00 = generator 0
122.     // bity 5-4 11 = zapis lub odczyt obydwu bajtow licznika CE
123.     // bity 3-1 011 = tryb 3
124.     // bit 0          0      = zliczanie w kodzie binarnym
125.     asm mov a1, 00110110b;
126.     asm out 0x43, a1;
127.
```

```
128.     // zapisywanie do 0x40 - adres Licznika generatora 0
129.     asm mov ax, dx;
130.     asm out 0x40, a1; // LSB (mniej znaczacy bajt licznika CE)
131.     asm xchg ah, a1;
132.     asm out 0x40, a1; // MSB (bardziej znaczacy bajt licznika CE)
133. }
134.
135. void reset8253() {
136.     // ustawienie starej czestotliwosci zegara
137.     asm mov dx, 65535;
138.
139.     // reszta tak samo jak w setup8253()
140.     asm mov a1, 110110b;
141.     asm out 0x43, a1;
142.
143.     asm mov ax, dx;
144.     asm out 0x40, a1;
145.     asm xchg ah, a1;
146.     asm out 0x40, a1;
147. }
148.
149. void interrupt far newIntIRQ7() {
150.     // wylaczenie przerwan
151.     asm cli;
152.
153.     confirmed=1;
154.     //w wyslanie potwierdzenia obsluzenia przerwania do 8259
155.     asm mov a1, 20h;
156.     asm out 20h, a1;
157.
158.     // wlaczenie przerwan
159.     asm sti;
160. }
```

```
161.  
162. void interrupt far newIntIRQ0() {  
163.     // zmienna zliczajaca przerwania  
164.     // po określonej liczbie przerwan wywoływanie jest  
165.     // domyslnie przerwanie, które inkrementuje odpowiednie rejestrty itd  
166.     // ogólnie słuzy temu, żeby czas w systemie płynął  
167.     // w windowsie 95 i nowszych jest osobny sterownik czasu  
168.     // ale w dosię bez tego czas staje  
169.     static WORD InternalCycleCount = 0;  
170.  
171.     // wyłączenie obsługi przerwan  
172.     asm cli;  
173.  
174.     InternalCycleCount++;  
175.  
176.     // zarządzanie licznikiem PWM  
177.     if (counterPWM <= 1) {  
178.         // po całym cyklu zegara obliczane są nowe wartości  
179.         // predkości silników  
180.         calculate_pos();  
181.         // i cykl zaczyna się na nowo  
182.         counterPWM = maxPWM;  
183.         set_motors();  
184.     }  
185.     else {  
186.         counterPWM--;  
187.         set_motors();  
188.     }  
189.  
190.     if (InternalCycleCount < freqModifier) {  
191.         // OCW2 - info o obsłudze przerwania  
192.         asm mov a1, 20h; // 0010 0000 - EOI  
193.         asm out 20h, a1; // 20h - adres głównego rejestrów PIC
```

```
194.    }
195.    else {
196.        // obsluga starego przerwania
197.        // zeby czas systemowy sie nie posypal
198.        InternalCycleCount = 0;
199.        oldIntIRQ0();
200.    }
201.
202.    // wlaczenie obslugi przerwan
203.    asm sti;
204. }
205.
206. void calculate_pos() {
207.     WORD data_pint;
208.     pdif = dif;
209.     dif = get_sensors();
210.
211.     // obliczenia PID
212.
213.     P = dif * kp;
214.
215.     I = I + dif;
216.     I = i * ki;
217.
218.     rate = dif - pdif;
219.     D = rate * kd;
220.
221.     turn = P + I + D;
222.
223.     if (confirmed==1) {
224.         left_motor = Tp+turn <= 50 ? Tp+turn : 50;
225.         left_motor = left_motor >= -50 ? left_motor : -50;
226.         right_motor = Tp-turn <= 50 ? Tp-turn : 50;
```

```
227.         right_motor = right_motor >= 50 ? right_motor : -50;
228.     }
229.     else {
230.         left_motor = 0;
231.         right_motor = 0;
232.     }
233.
234. // kierunek pracy silników
235. data_pins = inportb(DATA);
236.
237.     if (right_motor >= 0) {
238.         data_pins = (data_pins|0x02); // D1 - 1
239.         data_pins = (data_pins&0xFB); // D2 - 0
240.     }
241.     else {
242.         right_motor*=(-1);
243.         data_pins = (data_pins&0xFD); // D1 - 0
244.         data_pins = (data_pins|0x04); // D2 - 1
245.     }
246.     if (left_motor >= 0) {
247.         data_pins = (data_pins|0x40); // D4 - 1
248.         data_pint = (data_pint&0xDF); // D5 - 0
249.     }
250.     else {
251.         left_motor*=(-1);
252.         data_pins = (data_pins&0xBF); // D4 - 0
253.         data_pins = (data_pins|0x20); // D5 - 1
254.     }
255.
256.     outportb(DATA, data_pins);
257. }
258.
259. void set_motors() {
```

```

260.     // prawy silnik
261.     if (counterPWM == right_motor) { // D0 - 1
262.         outportb(DATA, inportb(DATA)|0x01);
263.     }
264.     else if (counterPWM == maxPWM) { // D0 - 0
265.         outportb(DATA, inportb(DATA)&0xFE);
266.     }
267.
268.     // Lewy silnik
269.     if (counterPWM == left_motor) { // D4 - 1
270.         outportb(DATA, inportb(DATA)|0x10);
271.     }
272.     else if (counterPWM == maxPWM) { // D4 - 0
273.         outportb(DATA, inportb(DATA)&0xEF);
274.     }
275. }
276.
277. float get_sensors() {
278.     float res=0;
279.     int count_sensors=0;
280.     WORD status_pins = inportb(STATUS);
281.
282.     if ((status_pins|0xEF) == 0xFF) { // S4 - DIODA 1
283.         res-=1.5; count_sensors++;
284.     }
285.     if ((status_pins|0x7F) != 0xFF) { // S7 - DIODA 2
286.         res-=0.5; count_sensors++;
287.     }
288.     if ((status_pins|0xF7) == 0xFF) { // S3 - DIODA 3
289.         res+=0.5; count_sensors++;
290.     }
291.     if ((status_pins|0xDF) == 0xFF) { // S5 - DIODA 4
292.         res+= 1.5; count_sensors++;

```

```
293.     }
294.
295.     return (count_sensors!=0) ? res/(float)count_sensors : (0.0);
296. }
```

## 8259

Wykorzystaliśmy 2 wejścia: IR0 odpowiedzialne za przerwania generowane przez układ 8253 oraz IR7 odpowiedzialne za przerwanie ~ACK z portu LPT. W tym celu konieczne było pobranie starego wektora przerwań przy użyciu funkcji getvect(void interrupt(\*isr)()) i ustawienie nowego wektora przerwań przy użyciu funkcji setvect(int intr\_num, void interrupt (\*isr)()). Numery wektorów przerwań to dla IRQ0 08h, a dla IRQ7 0Fh. Dla przerwania IRQ7 z portu LPT potrzebne było ponadto ustawienie 4 bitu kontrolnego portu LPT na 1, oraz odmaskowanie przerwania poprzez rozkaz OCW1 (zerowanie bitu 7 i odmaskowanie odpowiedniej linii zgłoszeń). Na zakończenie pracy robota ustawiamy poprzednie wektory przerwań przy użyciu funkcji setvect.

## 8253

Generator 0 sterowany jest zegarem o częstotliwości 1.19318MHz. Oryginalnie zegar systemowy wyzwala przerwania na linii IRQ0 z częstotliwością około 18.2Hz (licznik ustawiony jest na wartość fffffh). Prowadzi to do wywołania procedury INT 08h. Procedura ta inkrementuje każdorazowo stan licznika zlokalizowanego w obszarze danych BIOS. Aby uniknąć problemów z czasem systemowym, co określony czas należy wywoływać oryginalną funkcję obsługi przerwania. Układ 8253 jest programowany po przesłaniu do rejestru sterującego (port 043h) następującego rozkazu:

- bity 7-6 00 = generator 0
- bity 5-4 11 = zapis lub odczyt obydwu bajtów licznika CE
- bity 3-1 011 = tryb 3
- bit 0 0 = zliczanie w kodzie binarnym

Konieczne jest także przesłanie pod adres 40h nowej wartości licznika CE.

## Omówienie algorytmu sterującego

Do sterowania wykorzystaliśmy algorytm PID (proporcjonalny, całkujący i różniczkujący). Algorytm ten przetwarza dane uzyskane z czujników i wylicza nową prędkość silników. Czynnik P jest odpowiedzialny za szybkie reagowanie na zmiany na torze; I zapamiętuje poprzednie wartości, jest wartością akumulowaną; D ogranicza wahania robota wokół linii. Poszczególne współczynniki kp, ki, kd i Tp dobraliśmy doświadczalnie.

## **Serwisowanie i uruchamianie**

Po uruchomieniu programu, aby uruchomic robota należy wcisnąć przycisk generujący przerwanie na pinie ACK. Dopiero wówczas robot rozpocznie pracę.

Koszyczki z bateriami umieszczone zostały na spodniej części robota, co umożliwia łatwy do nich dostęp i ewentualną wymianę.

Silniki zostały umieszczone z tyłu pojazdu, przy pomocy dwóch obejm ułatwiających ich wymianę w przypadku awarii.

## **Podsumowanie**

### **Analiza SWOT**

<b>Mocne strony</b>	<b>Słabe strony</b>
<ul style="list-style-type: none"><li>● dobre stosunki między członkami zespołu</li><li>● dobre umiejętności wydajnego organizowania pracy i czasu</li><li>● dobra komunikacja pomiędzy członkami zespołu</li></ul>	<ul style="list-style-type: none"><li>● brak doświadczenia w projektowaniu i tworzeniu układów elektronicznych</li><li>● brak doświadczenia w tworzeniu dokumentacji technicznej</li></ul>

<b>Szanse</b>	<b>Zagrożenia</b>
<ul style="list-style-type: none"><li>● dostępność poradników / dokumentacji</li><li>● powszechność i łatwa dostępność potrzebnych części</li><li>● łatwo i stale dostępne miejsce pracy</li></ul>	<ul style="list-style-type: none"><li>● nie otrzymanie wszystkich zamówionych części na czas</li><li>● otrzymanie uszkodzonych części</li><li>● przypadkowe uszkodzenie newralgicznych podzespołów</li></ul>

## Kosztorys

<b>nazwa</b>	<b>sztuk</b>	<b>cena jednostkowa [PLN]</b>	<b>cena [PLN]</b>
<b>Czujnik transceptor odbiciowy CNY70</b>	10	3.50	35
<b>Dioda LED 3mm zielona 10 szt.</b>	1	1.90	1.90
<b>Komparator analogowy LM339 - DIP</b>	2	0.50	1
<b>L293D - dwukanałowy sterownik silników</b>	1	3.90	3.90
<b>Potencjometr montażowy leżący 200R - 5szt.</b>	1	1.99	1.99
<b>Kondensator elektrolityczny 0,47uF/50V 105C - 10szt</b>	2	0.99	1.98
<b>Kondensator elektrolityczny 100uF/25V 105C - 10szt.</b>	2	0.99	1.98
<b>Silnik DC HL149 z przekładnią 10:1</b>	2	14.99	29.98
<b>Zestaw rezystorów THT 1/4W - 200 szt.</b>	1	7.90	7.90
<b>Podstawka do układów DIL 16pin zwykła - 5szt.</b>	2	0.80	1.60
<b>Wtyk goldpin 1x40 prosty</b>	5	0.49	2.45
<b>Wtyk goldpin 2x40 kątowy</b>	2	1.50	3
<b>Laminat FR4 dwustronny 200x90mm - 1,5mm</b>	1	3.99	3.99
<b>Zasilanie (kilka kompletów baterii)</b>	-	120	120
<b>Części mechaniczne</b>	-	40	40
<b>Przewód do elektroniki</b>	7	1.30	9.10
<b>Koszulki termokurczliwe</b>	1	5.00	5.00
<b>Stabilizatory napięcia</b>	3	1.50	4.50
<b>zacisk do baterii</b>	2	1.30	2.60

<b>płytki uniwersalna</b>	1	12.00	12.00
<b>koła</b>	1	25.00	25.00
<b>SUMA</b>			314.87

Koszt na osobę: 39.35 PLN

Oprócz tego zmuszeni byliśmy do zakupu niektórych narzędzi, które nie zostały uwzględnione w kosztorysie. Część części nie została ostatecznie użyta. Było to spowodowane zmianami w trakcie prac jak i chęcią zabezpieczenia się na wypadek usterek poszczególnych elementów

### **Uwagi końcowe**

Realizacja projektu postawiła przed naszym zespołem spore wyzwanie. Nikt z nas nie miał w przeszłości bliższej styczności z elektroniką i lutowaniem w projekcie tego typu. Początkowo szczególnie trudności mieliśmy z odpowiednim połączeniem układów scalonych oraz z lutowaniem. Kolejnym wyzwaniem okazało się być dobranie współczynników do algorytmu PID tak, aby robot jechał z jak największą prędkością i jednocześnie nie wypadał z trasy. Uważamy skonstruowanie działającego robota w wyznaczonym czasie za nasz sukces.

Zdobyte doświadczenie wykorzystamy w drugim etapie projektu - konstrukcji robota z mikrokontrolerem.