



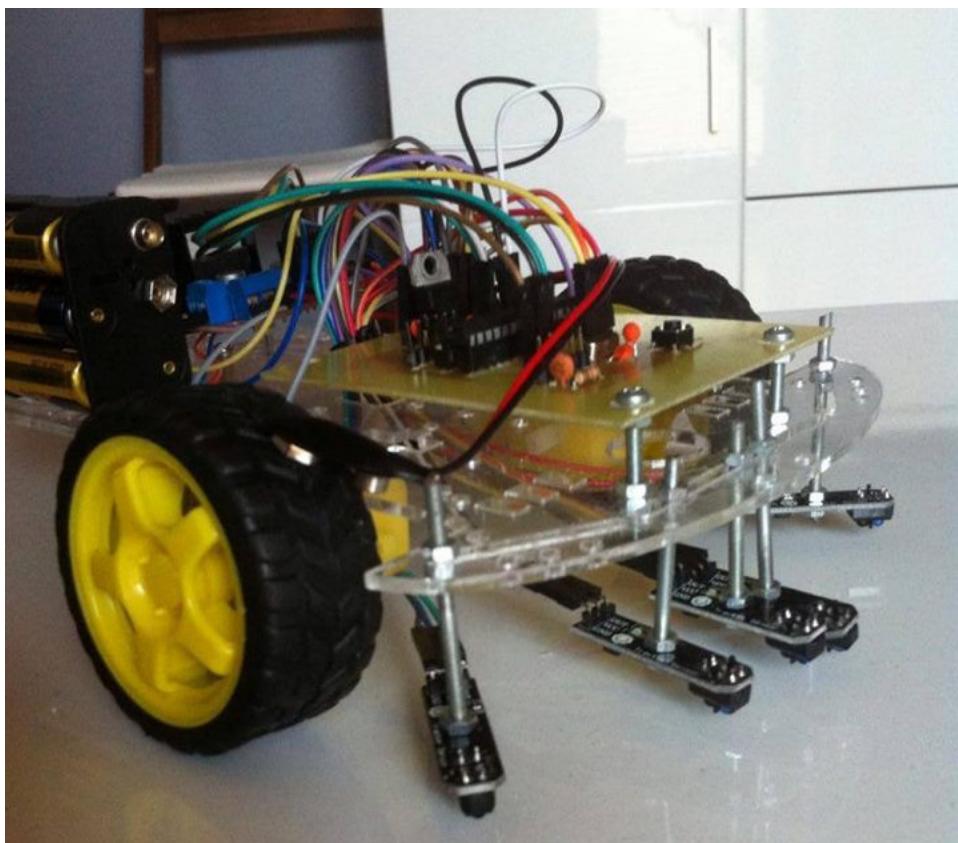
ACADEMY OF CREATIVE ACTION

BASED ON RAFAEL KLAUS TRAINING

Institute of Computing Science Poznan University of Technology

Pszemeg DTR

8 czerwca 2016



Rysunek 1: Pszemeg

imie	nazwisko	przydział zadań
Piotr	Binkowski	zamawianie i kompletowanie podzespołów, projekt płytki drukowanej, nadzorowanie prac
Adam	Ćwian	wytrawianie płytki (termotransfer)
Paweł	Miłończyk	DTR, lutowanie płytki, trawienie płytki, wykonanie termometru odpornego na wrzątek
Juliusz	Sałek	kapitan, wykonanie tras, DTR, fotorelacja, kąpiel płytki w roztworze
Patryk	Scheffler	wytrawianie płytki, fotorelacja, testowanie robota
Jakub	Stańczak	zaprogramowanie robota, wiercenie i czyszczenie płytki, system regulacji czujników
Jorge	Abreau	słuchanie i próba zrozumienia działania robota i kodu

Spis treści

Spis rysunków

1 Opis

Celem robota jest podążanie wyznaczoną linią. Jest w stanie pokonywać zakręty do 90° oraz skrzyżowania. Robot najlepiej radzi sobie na łukach, gdyż trzy środkowe czujniki są zblizone do siebie co umożliwia mu szybkie wprowadzania korekt i płynną jazdę. Problematyczne są ostre zakręty, gdzie linia jest łapana dopiero przez zewnętrzne czujniki. Robot wtedy zwalnia i stara się wrócić na trasę środkowym czujnikiem. W przypadku wypadnięcia z trasy robot będzie wykonywał nawrót w tę stronę, gdzie ostatnio widział linię.

Robot wykorzystuje pięć czujników. Dwa skrajne umieszczone są w znacznym oddaleniu i trochę z tyłu. Napęd robota wykonany jest z dwóch silników z przekładniami, umieszczonych z przodu pojazdu. Zasilany jest z 6 baterii AA. Czas działania jest szacowany na 2h. Nie zdecydowaliśmy się na zakup akumulatorów, ze względu na wysoki ich koszt oraz wymogu posiadania specjalnej ładowarki. Tym sposobem robot kosztował nas zaledwie ok. 91.20zł.

Niestety, ale duża masa, czujniki położone blisko osi skrętnej oraz gotowe podwozie skutecznie uniemożliwiły nam walkę o dobry czas. Postawiliśmy na dokładność i nasz robot, gdy był ukończony, nigdy nie wypadł z trasy. Wypadanie z trasy było charakterystyczne dla szybkich i małych robotów, które musiały wiele razy startować w eliminacjach by ukończyć trasę.

2 Elementy

2.1 Tabela elementów

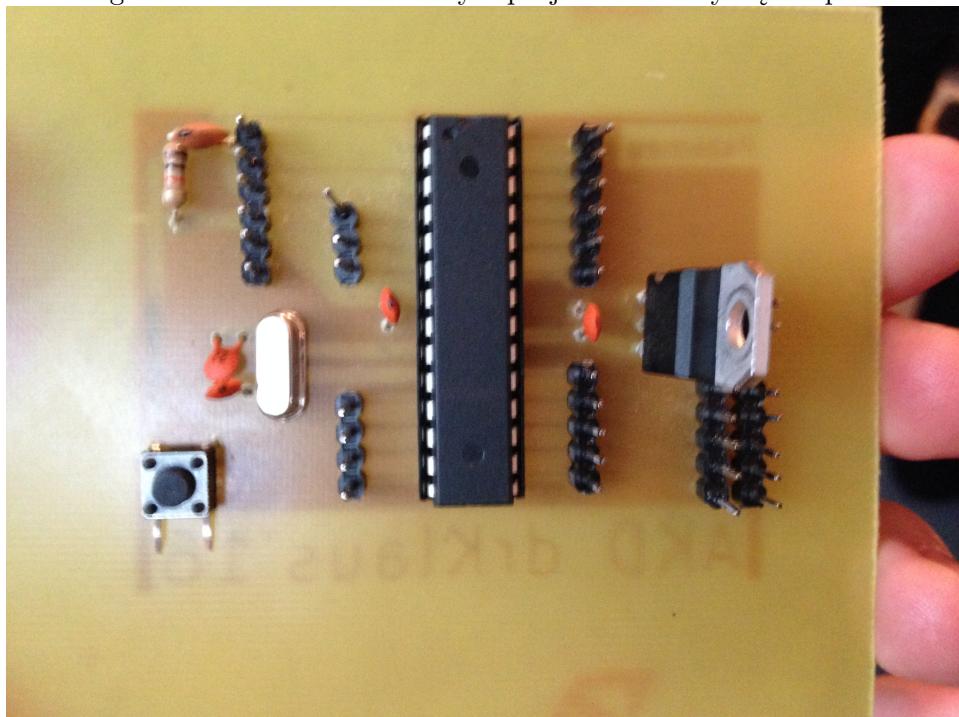
nazwa	cena
2 x silnik 9V z przekładniami i kołami	16,5
5 x czujnik TCRT5000 IR	9,7
podwozie	30
mostek h L289N	6
atmega 328P	6
koszyk na 6 baterii	4
wytrawiacz nadsiarczan sodowy b327	5,5
pcb 7x9cm	2,5
4 x kondensator 68nF	–
oscylator 16Mhz	–
2 x kondensator 22pF	–
4 x kondesator 100nF	–
stabilizator 7805	1
przycisk monostabilny	–
gniazdo dil 28	1
listwa kołkowa jednorzędowa	–
śruby i wiertła	5
przewody	4
żelazko	–
papier kredowy	0,2
cyna	małe ilości
aceton	małe ilości

2.2 Opis elementów

2.2.1 atmega 328P

- Wyprodukowany przez firmę Atmel.
- Obudowa typu dip 28.
- Architektura AVR RISC.
- Napięcie 1,8V - 5,5V
- 8 Mhz na wewnętrznym oscylatorze, możliwość zewnętrznego oscylatora.

Mikrokontroler jest zgodny z arduino, o ile poprawnie umieszczono oscylator. Jeśli oscylator nie działa mikrokontroler może być dalej zaprogramowany przez Arduino IDE ale jako Atmega 8Mhz. Mikrokontroler został dobrany na podstawie ceny i dostępności IDE. Jest również dostępnych na niego wiele bibliotek które w tym projekcie okazały się nie potrzebne.

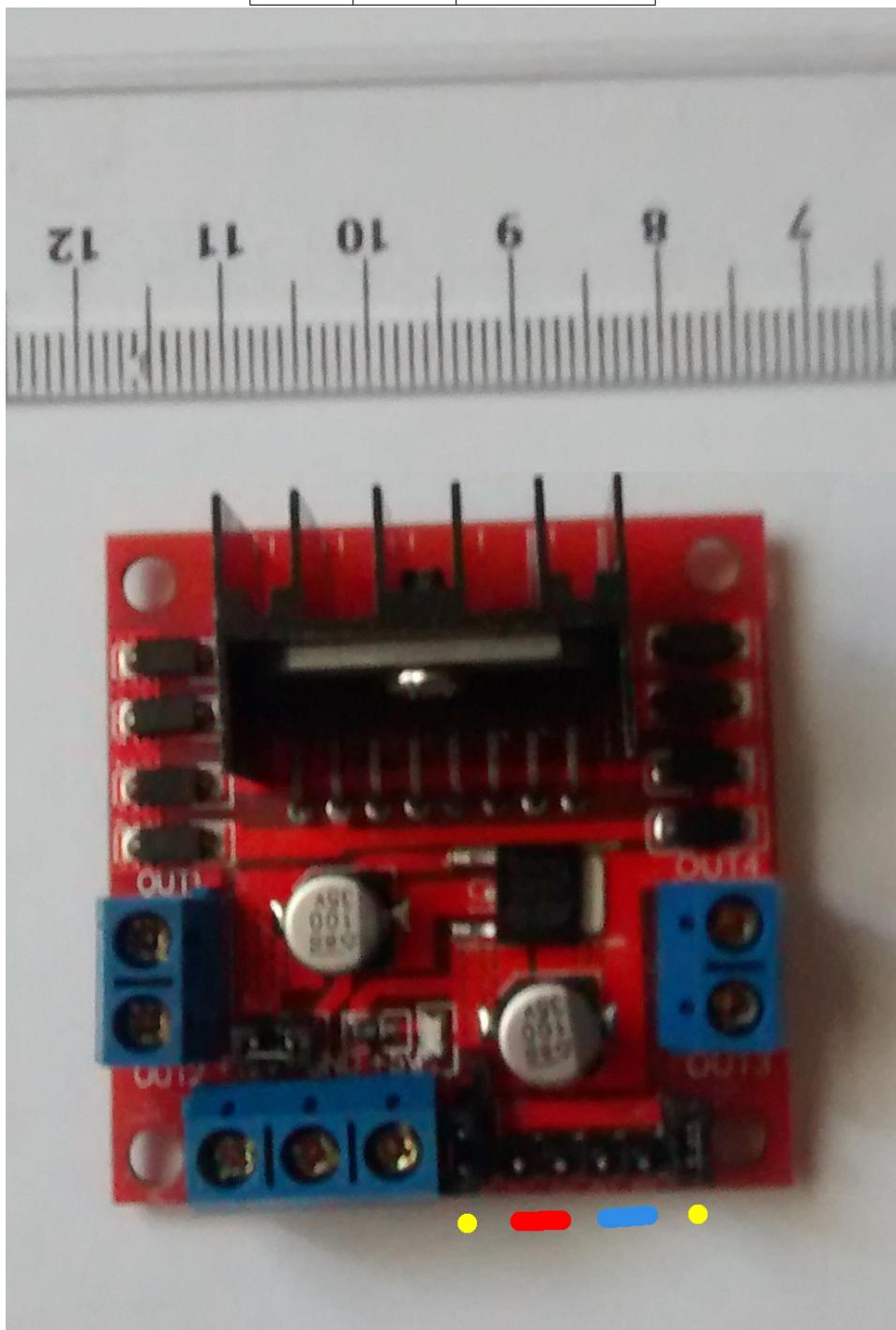


Rysunek 2: atmega
Jest to układ w obudowie DIP, pośrodku zdjęcia.

2.2.2 mostek H L298N

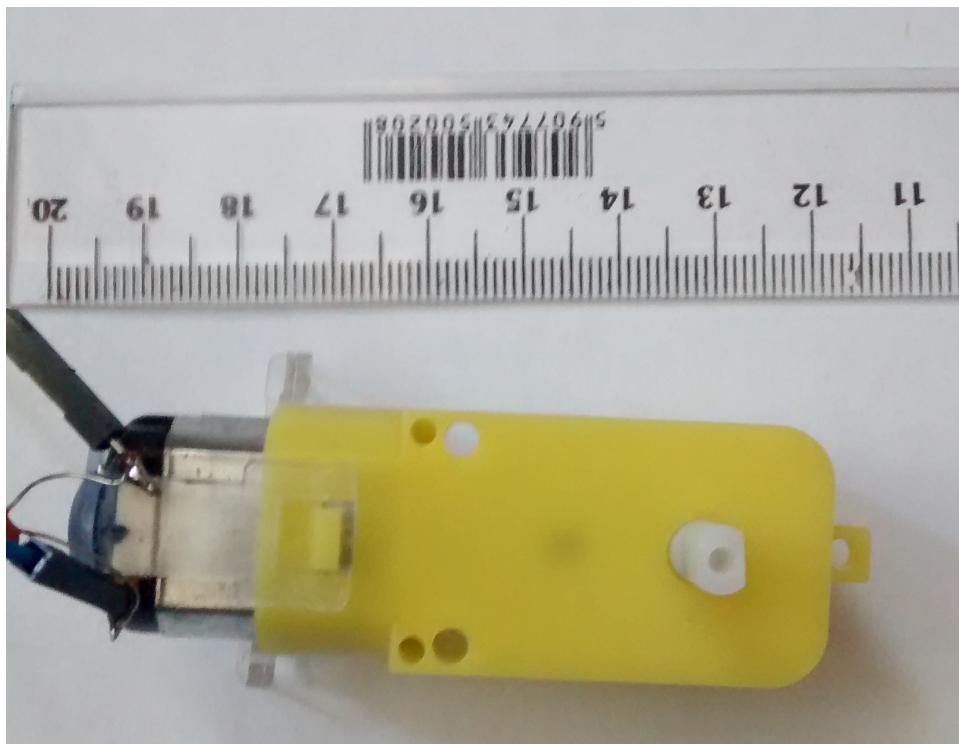
Do wyjść OUT1 OUT2 należy podłączyć jeden silnik. Do OUT3 i OUT4 drugi. Jeśli silnik kręci się w złą stronę należy odwrócić połączenie. Najbardziej skrajne piny (żółte oznaczenie) służą do sterowanie PWM. Czerwone i niebieskie (oznaczone w kodzie jako l_engine.dir i r_engine.dir) to kierunek obrotów oraz wybór hamulca. Przy podaniu zera na sterowanie PWM nie robi różnicy co poda się na pozostałe, silnik zostanie wyłączony.

LOW	LOW	hamowanie
HIGH	LOW	jazda w przód
LOW	HIGH	jazda w tył
HIGH	HIGH	hamowanie

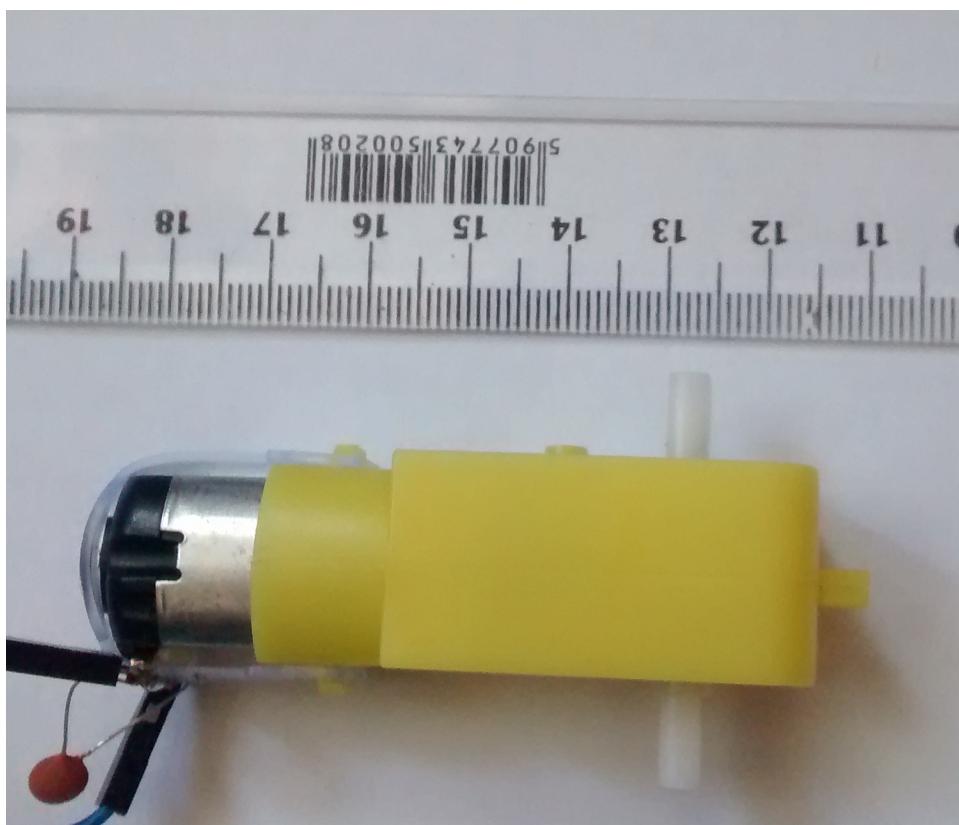


Rysunek 3: mostek L298N

2.2.3 silnik



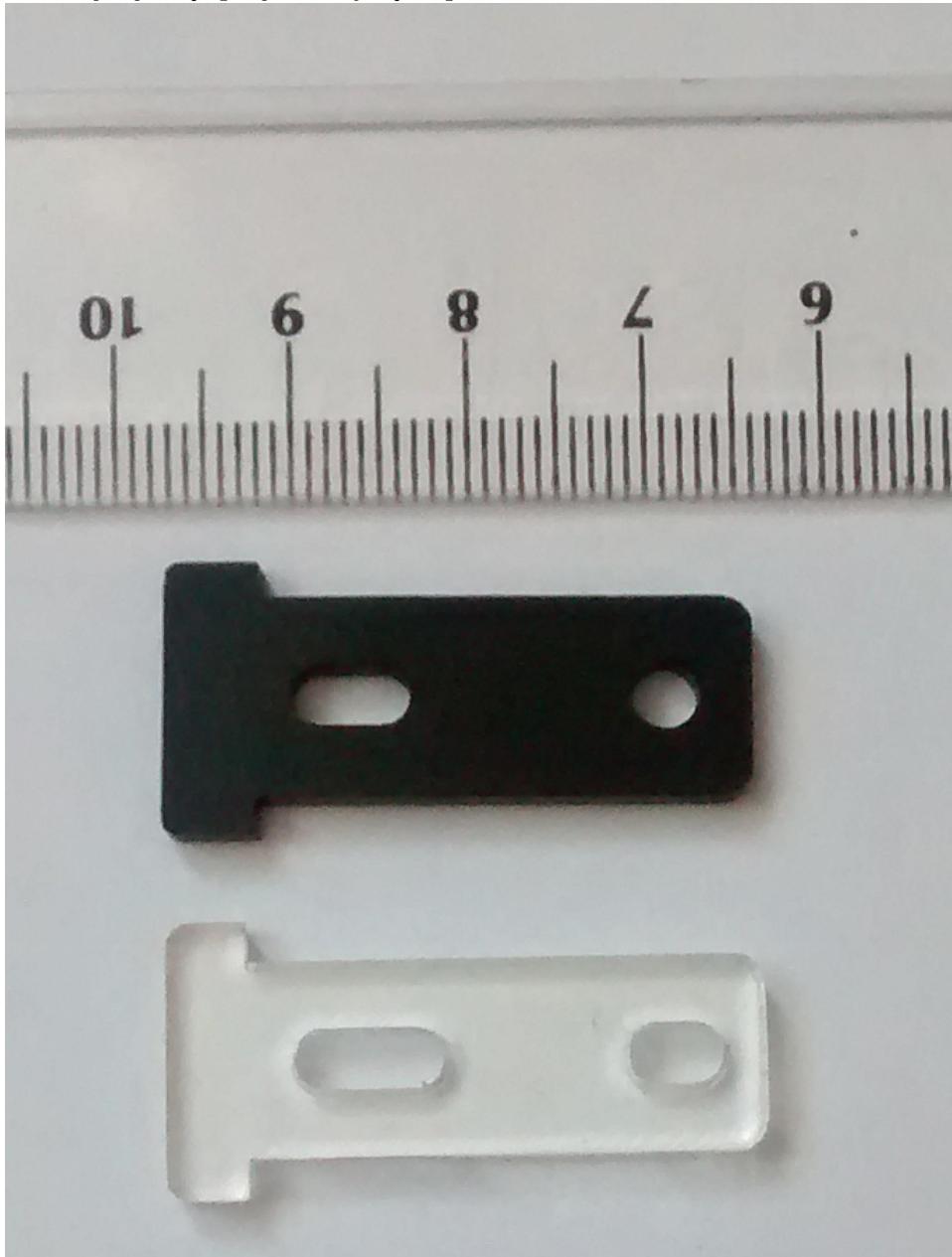
Rysunek 4: silnik bok



Rysunek 5: silnik góra

2.2.4 zaczepy

Przydatne przy mocowaniu silnika, dwa na każdy silnik, po obu stronach mocowane śrubami. W podwoziu znajduje się specjalne wycięcie po obu bokach.



Rysunek 6: zaczepy



Rysunek 7: Mocowanie silnika
Śruby którymi łączy się silnik z zaczepami.

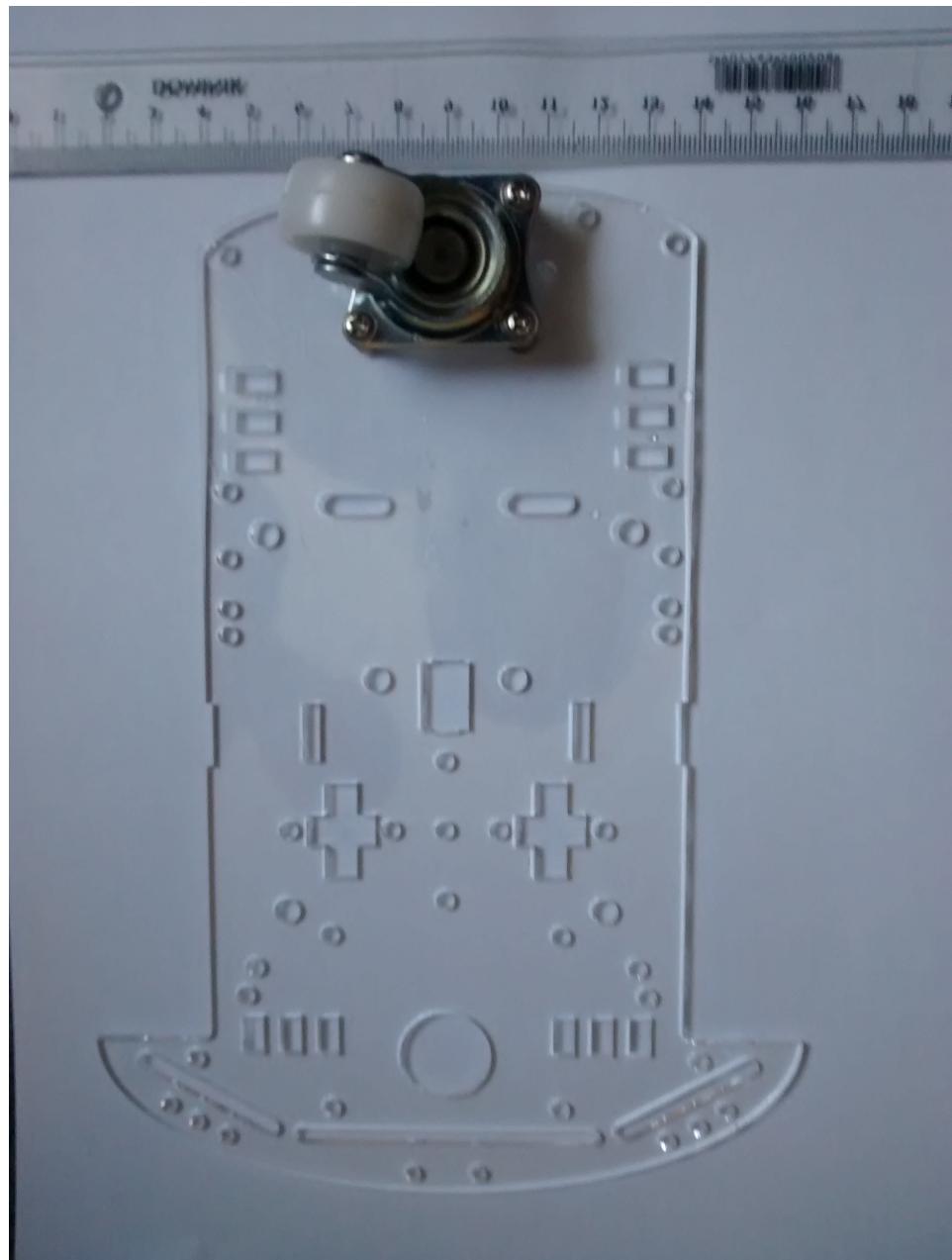
2.2.5 koło

Należy zwrócić specjalną uwagę, aby koło nie dotykało śruby mocującej.

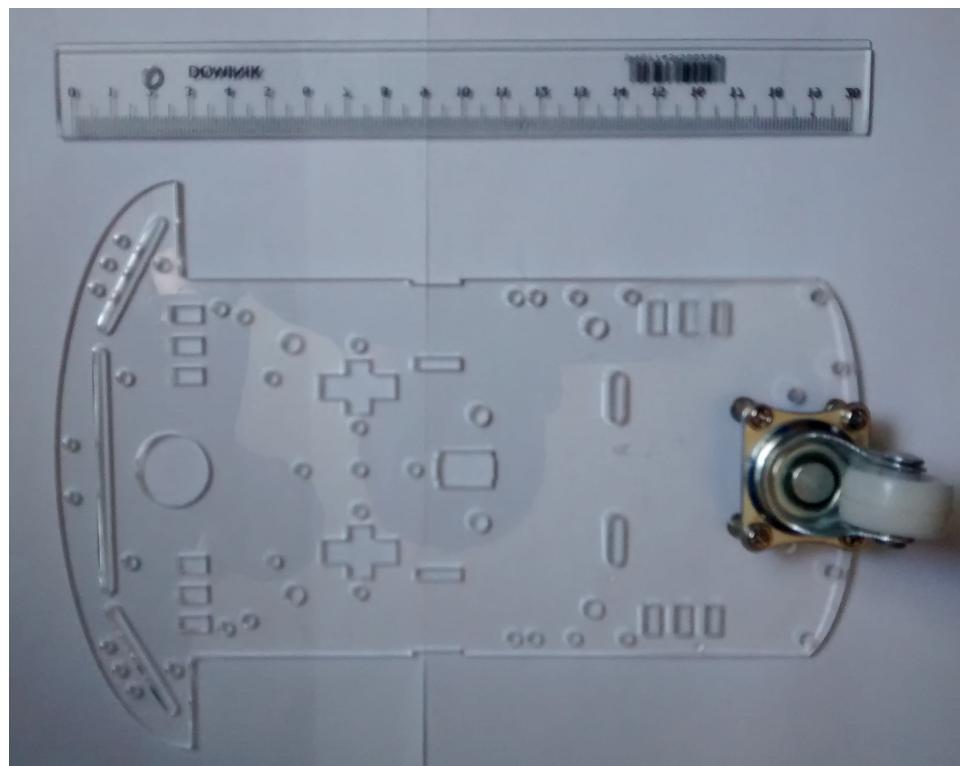


Rysunek 8: koło

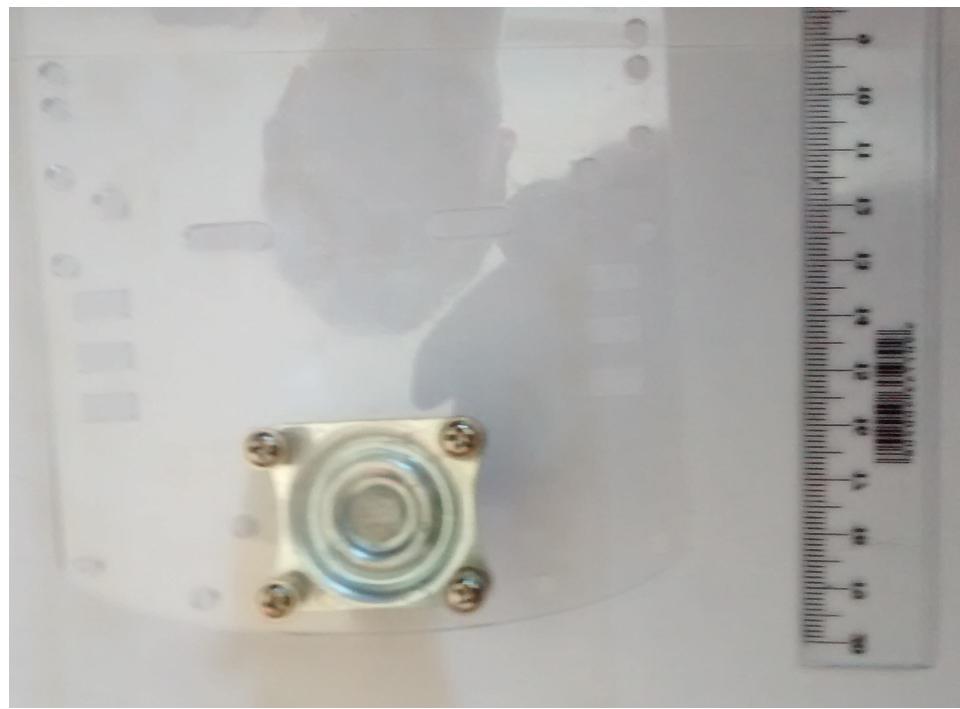
2.2.6 podstawa



Rysunek 9: podstawa



Rysunek 10: podstawa



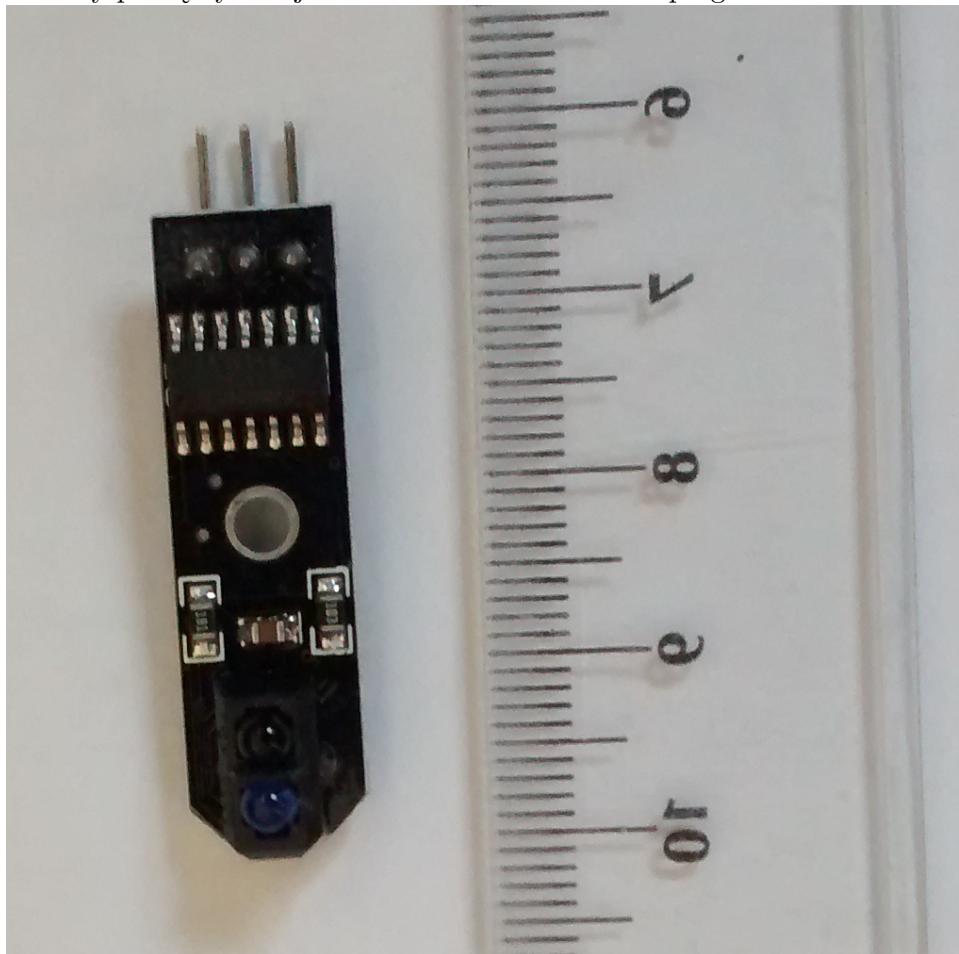
Rysunek 11: podstawa



Rysunek 12: podstawa

2.2.7 czujnik TCRT5000

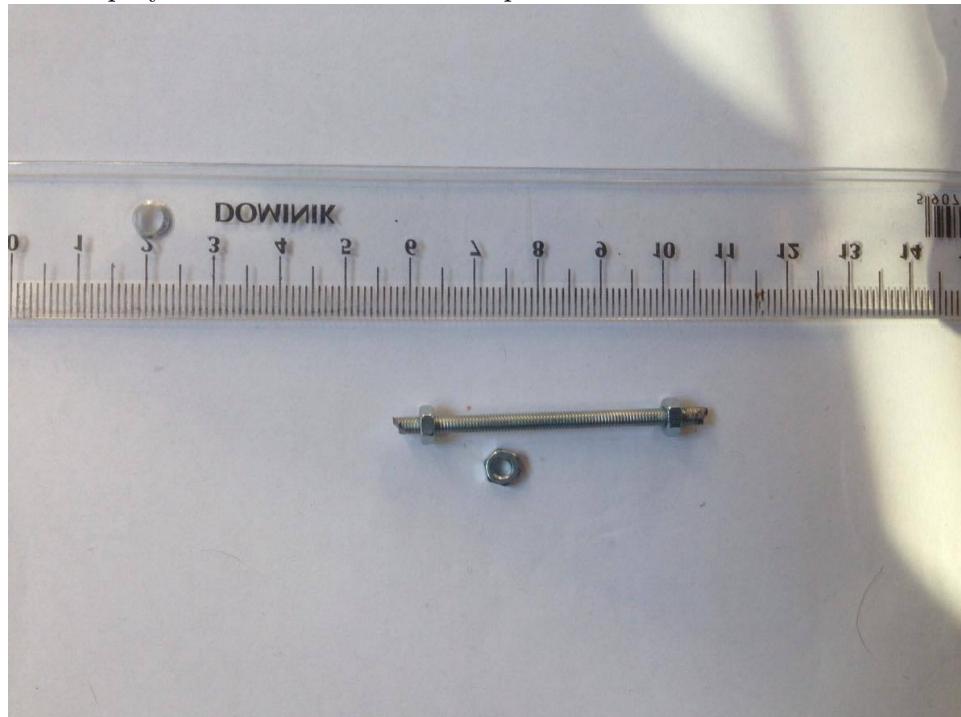
Czujnik wyposażony jest w diodę LED, którą podświetla trasę. Unika tym poleganiu na zewnętrznych źródłach światła. Następnie mierzy wartość światła odbitego. Działa w podczerwieni. Wybrana wersja czujnika wyposażona jest w przerutnik schmitta, który niestety bardziej przeszkadza niż pomaga. Lepiej jest wykorzystać czujniki w pełni analogowe. Czujnik ma również diodę diagnostyczną, która świeci w świetle widzialnym aby ułatwić diagnozowanie. Piny do których należy podłączyć czujniki można łatwo zmienić w programie.



Rysunek 13: czujnik

2.2.8 Mocowanie czujnika

Czujnik zamocowany jest na pociętym pręcie i przykręcony do podwozia. Odległość czujników należy dobrą eksperymentalnie w zależności od podłożu.



Rysunek 14: Mocowanie czujnika

2.2.9 pojemnik na baterie

Łączy 6 baterii szeregowo w wyniku czego napięcie wyjściowe wynosi około 9V.

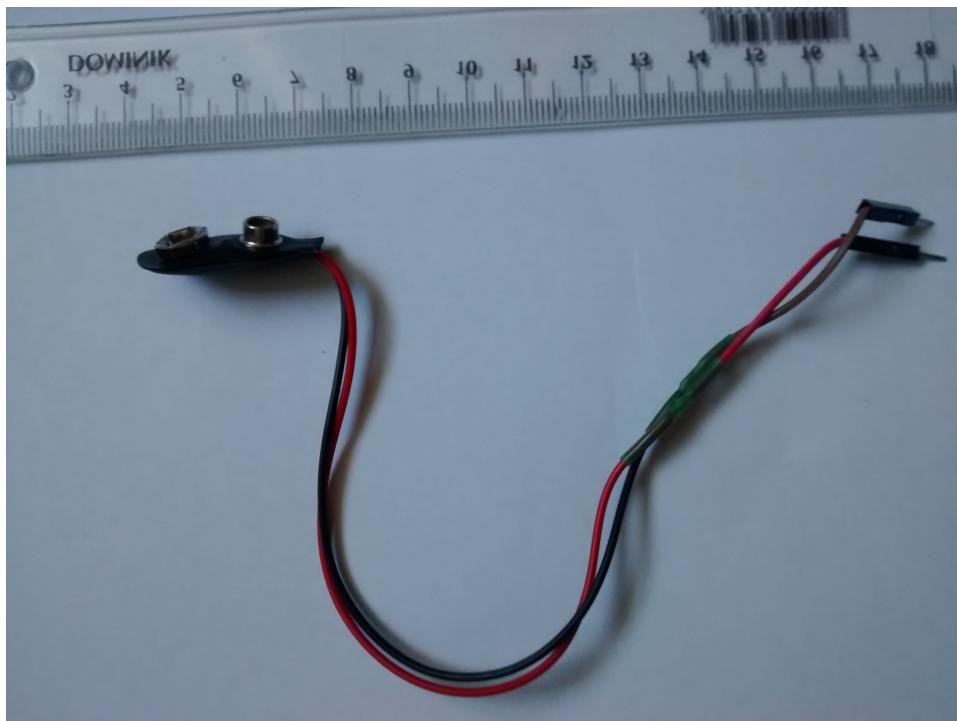


Rysunek 15: pojemnik bok



Rysunek 16: pojemnik góra

2.2.10 wtyczka



Rysunek 17: wtyczka

2.2.11 płytka pcb

Używana do wyprowadzenia wyjść Atmegi oraz jako miejsce na stabilizator 5V kondensatory filtrujące, oraz rezistor i przycisk do resetu. Łączy elementy. Wymiary 7x9cm.



Rysunek 18: Płytki PCB

2.2.12 kondenstor

68nF Wykorzystywany jest do tłumienia zakłóceń powodowanych przez silnik. Można wykorzystać również kondensatory 100nF.

100nF Wykorzystywany jest do tłumienia zakłóceń w układzie zasilania. Jeden wykorzystany do resetu Atmegi.

22pF Wymagany przez oscylator.

2.2.13 przycisk monostabliny

Ręcznie aktywuje reset Atmegi.

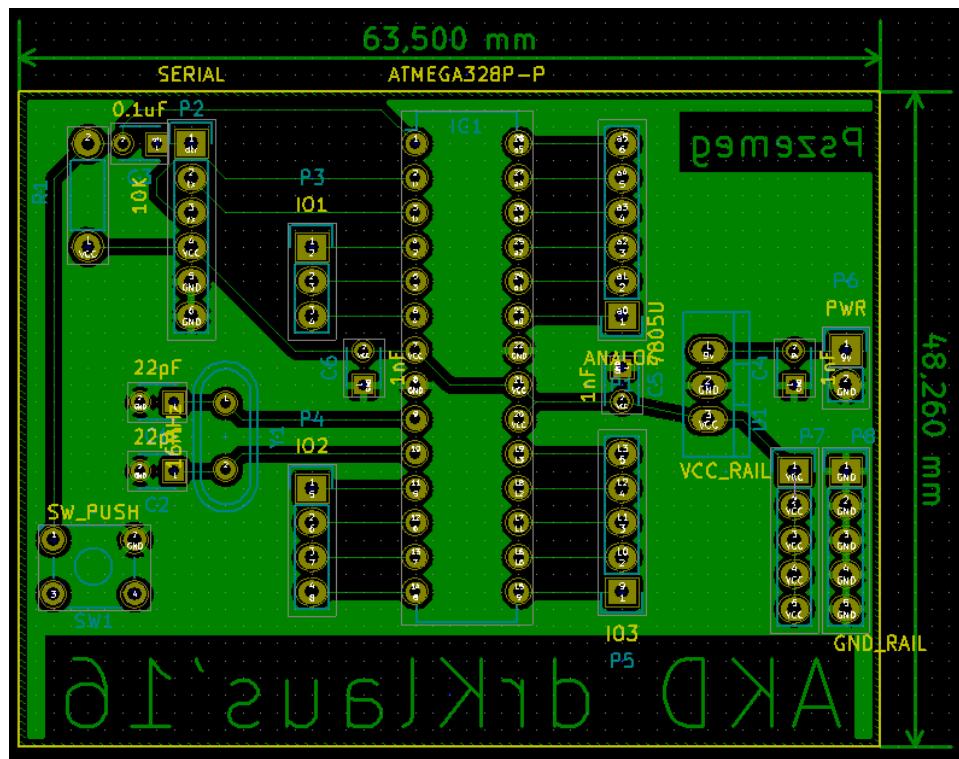
2.2.14 stabilizator 7805

Dostarcza stabilne 5V dla zasilania czujników oraz Atmegi. .

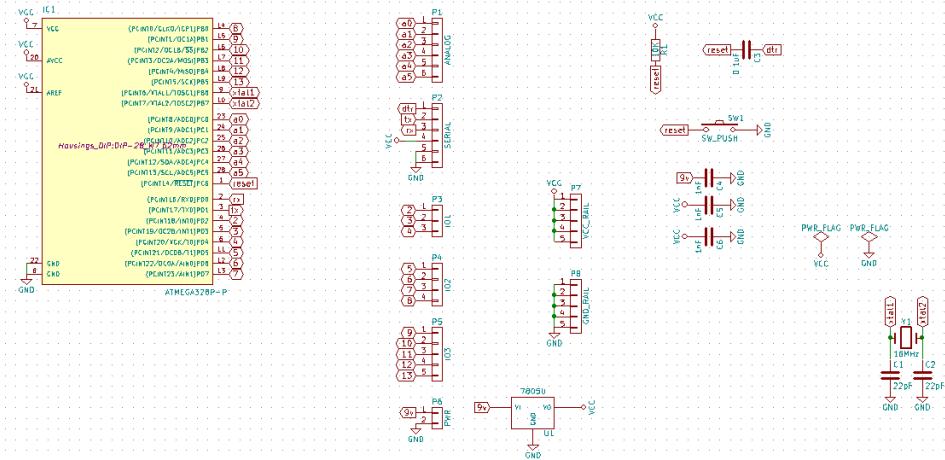
3 Płytki

3.1 Projekt

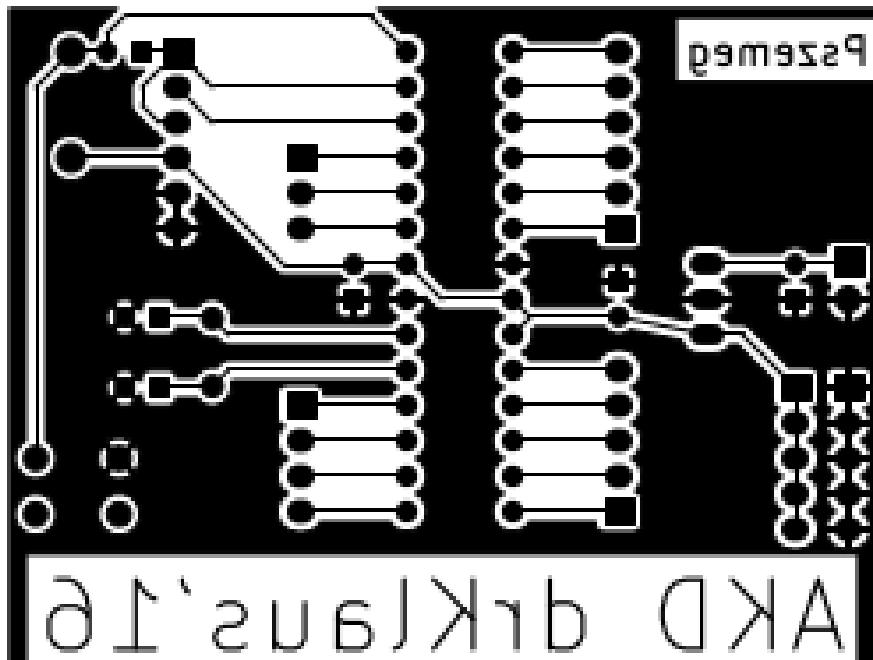
Projekt został wykonany w kiCad oraz z użyciem programu freerouter.



Rysunek 19: PCB projekt



Rysunek 20: schemat



Rysunek 21: schemat do druku
Plik PDF z tym obrazem można znaleźć pod nazwą FTL-B.pdf

3.2 Termotransfer

3.2.1 Przygotowania

Projekt należy wydrukować na papierze kredowym drukarką laserową. Uwaga na zachowanie skali wydruku. Następnie można przejść do przygotowanie miedzianej płytka. Należy oczyścić powierzchnie z tlenków wodnym papierem ściernym 1000. Kolejnie wyczyścić z zabrudzeń mocnym alkoholem. Tak przygotowana płytka jest gotowa na termotransfer.

3.2.2 Termotransfer

Do termotransfetu potrzebne jest żelazko. Należy je rozgrzać na dwie kropki ok (150°C).

Dobrze jest też przygotować odpowiednie stanowisko, by nie przesunąć płytki względem druku podczas operacji.

Można zastosować metodę gdzie żelazko jest na górze lub na dole względem płytki. W wielu poradnikach można znaleźć różne informacje na ten temat, jednak nie wiemy jaka metoda, temperatura, czas jest najlepszy. Za duża temperatura zżółci papier, za niska uniemożliwi termotransfer.

My zastosowaliśmy metodę górną z podkładem z zeszytu i ręczników papierowych. Nie prasowaliśmy też bezpośrednio po papierze kredowym, tylko przykryliśmy go papierem do pieczenia. W ramach nakładki na papier kredowy powinna też sprawdzić się bawełniana szmatka.

Papier kredowy drukiem do dołu przylegał do miedzianej płytki. Temperatura podgrzewała toner umożliwiając przeniesienie warstwy tonera na miedź. By zwiększyć docisk co 40 sek. prasowaliśmy płytke 40 sek. papierową kulką, umożliwiło to dokładniejszy i silniejszy docisk niż żelazkiem. Wykonaliśmy sześć takich zmian.

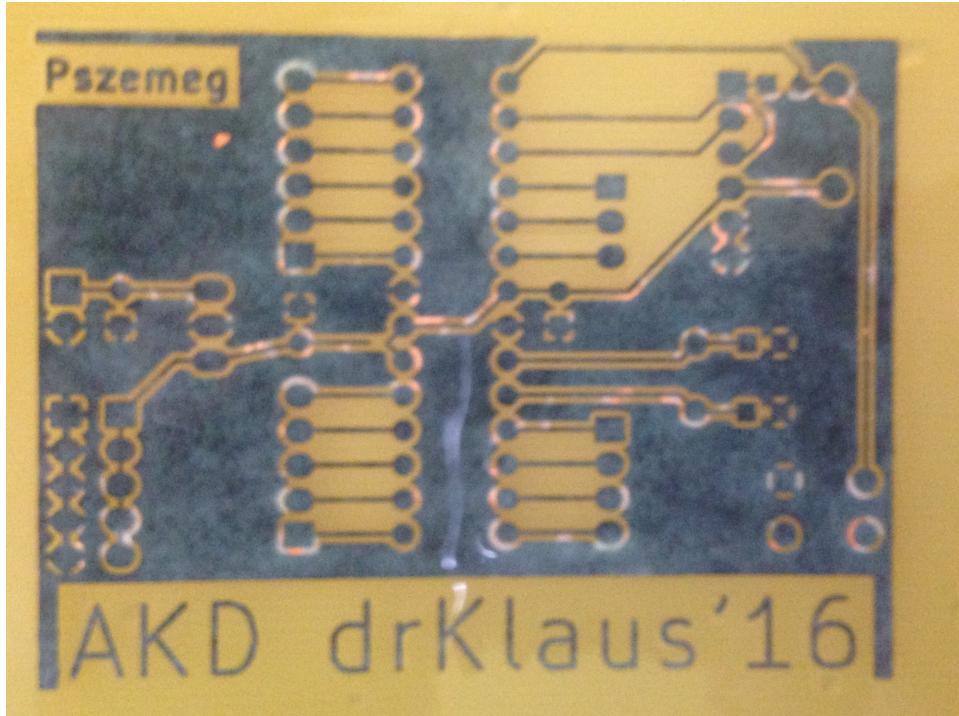
Toner nie powinien łatwo schodzić z płytki, jeśli schodzi po przeciągnięciu palcem, operacje lepiej powtórzyć, stosując trochę wyższą temperaturę lub wydłużając czas grzania.



Rysunek 22: termotransfer - nagrzewanie żelazkiem

3.2.3 Poprawki

Następnie trzeba usunąć papier kredowy. Można to zrobić w ciepłej wodzie z płynem do mycia naczyń, należy poczekać, aż nasiąknie. Kolejnie delikatnie go usuwamy. W niektórych miejscach nie pokrytych tonerem można znaleźć kredę. Jest to wysoce niepożądane, gdyż kreda utrudni dostęp roztworu do trawienia. Można ją usunąć np. wykałaczką.



Rysunek 23: Zabrudzenia kredą, tu niestety po wytrawianiu

Jeśli w niektórych miejscach ścieżki się nie przeniosły można je poprawić markerem niezmywalnym. Jeśli termotransfer nie wyszedł po usunięciu tonera acetonom procedurę rozpocząć od nowa.

3.3 Wytrawianie

Wytrawianie przeprowadziliśmy za pomocą roztworu nadsiarczanu sodowego b327. 0,1kg proszku należy rozpuścić w 0,5L wody. A następnie w temp. 40-50°C przeprowadzić trawienie. Temperaturę można utrzymywać dolewając ciepłej wody do zewnętrznego pojemnika.



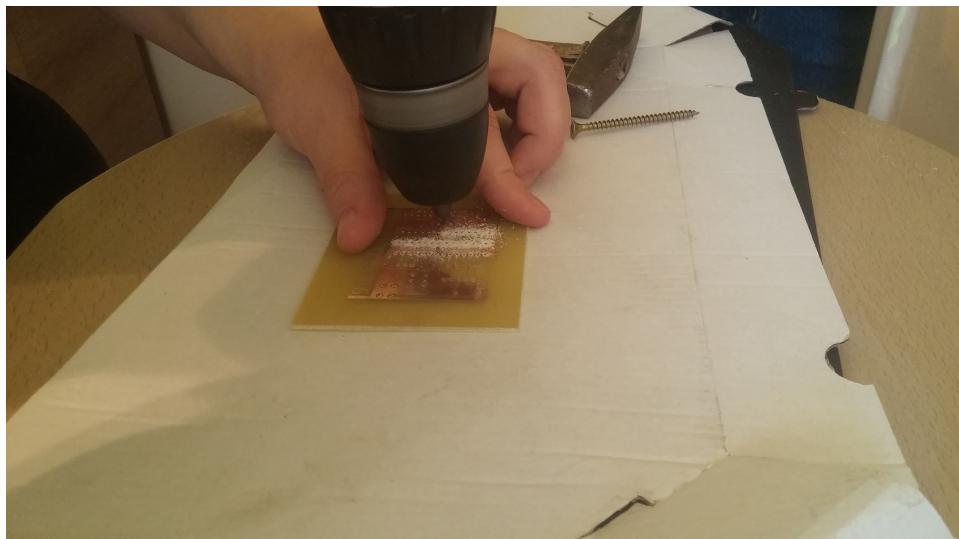
Rysunek 24: prowizoryczna łazienka laboratoryjna

Przydatna może być lepsza izolacja naczynia zewnętrznego. Przedstawione naczynie dość szybko traci ciepło co wiąże się z dolewaniem dużych ilości ciepłej wody.

Płytkę należy włożyć do roztworu i trzymać do zaniku miedzi nieprzykrytej tonerem. Trwa to ok 15. min. Poroszanie płytą lub mieszanie w roztworze ułatwia wytrawianie miedzi. Zbyt długie trzymanie płytka w roztworze spowoduje podtrawianie ścieżek i w konsekwencji ich przerwanie.

3.4 Wiercenie

W wytrawionej płytce należy wywiercić otwory montażowe. W tym celu należy użyć wiertła 1mm. Aby wiertło nie ślizgało się po płytce można zrobić małe dziurki śrubką lub gwoździem.



Rysunek 25: wiercenie

3.5 Lutowanie

Elementy należy przylutować do płytka. Zapewnia to dobre elektryczne połączenie jak i przytwierdza elementy do płytka. Można użyć zwykłej kolbowej lutownicy. W celu poprawy jakości i komfortu lutowanie można zastosować kalafonie, jednak nie jest to niezbędnie. Są dostępne cyny z topnikiem który ma ułatwić lutowanie.

4 Program

4.1 Wgrywanie

Jeśli zakupiono mikrokontroler atmega bez bootloader'a, należy go wgrać lub zdobyć układ z wgranym bootloader'em. Jest też opcja wgrywanie programu bezpośrednio. Jeśli układ ma bootload'er należy podłączyć programator (np. FT232RL) przez port szeregowy poszczególnymi kabelkami. Znajduje się on po lewej od oscylatora. Wgrywanie przeprowadza się poprzez naciśnięcie wgraj w Arduino IDE. Jeśli nie podłączono resetu (przez pin o nazwie DTR), układ należy ręcznie zresetować tuż przed wgrywaniem.

4.2 Opis

Program działa jak regulator PID. Jednak z częścią całkującą pojawiły się pewne problemy dlatego czasem dobrze jest ograniczyć go do sterownika PI.

4.3 Kod

```
1 #define l_engine_speed 10 //pin do kontroli predkosci lewego silnika
2 #define r_engine_speed 9 //pin do kontroli predkosci prawego silnika
3
4 #define l_engine_dir1 5 // pin do kontroli kierunku lewego silnika
5 #define l_engine_dir2 4 // pin do kontroli kierunku lewego silnika
6 #define r_engine_dir1 3 // pin do kontroli kierunku prawego silnika
7 #define r_engine_dir2 2 // pin do kontroli kierunku prawego silnika
8
9 #define changeColor 1 // zamienia wychwytywanie koloru czarnego z bialym
10
11 #define topSpeed 70 // max predkosc jazdy
12 #define maxTurn 70 // max spowolnienie jednego kola do skretu
13
14 int sensorPorts[] = {8, 6, 7, 13, 12}; //porty sensorow
15 int sensorCount = 5; //liczba sensorow
16
17 int kp = topSpeed/4; // wspolczynnik p
18 int kd = 40; // wspolczynnik d
19 int ki = 0; // wspolczynnik i
20 int kiMax = 100; //max wspolczynnika i
21
22 int kiJump = 1; //zwiększa wspolczynnik i o ten wspolczynnik
23 int kiSum = 0; // zmienna do liczenia sredniej z kilku wspolczynników i
24 int kiDelay = 100; // opoznienie i
25 int kiLoop = 0; // zmienna do petli
26
27 int kdDelay = 350; // opoznienie d
28 int kdSum = 0; // zmienna do liczenia sredniej z kilku wspolczynników d
29 int kdLoop = 0; // zmienna do petli
30
31 // kalibracja silników, jeżeli miałyby nierówna moc
32 int kalibracjaL = topSpeed;
33 int kalibracjaR = topSpeed;
34
```

```

35 // zwalnianie na zakretach o ulamek
36 int zwolnij_m = 1; // mnoznik
37 int zwolnij_d = 1; // dzielnik
38
39 // wartosci poczatkowe wspolczynnikow
40 int proportional = 0;
41 int derivative = 0;
42 int integral = 0;
43
44 int error = 0;
45 int lastError = 0;
46
47 void ride(int dir, int ftlSpeedl, int ftlSpeedr);
48 void calcError();
49 int mxor(int a, int b);
50
51 void setup() {
52   Serial.begin(9600);
53
54   pinMode(l_engine_speed, OUTPUT);
55   pinMode(r_engine_speed, OUTPUT);
56   pinMode(l_engine_dir1, OUTPUT);
57   pinMode(l_engine_dir2, OUTPUT);
58   pinMode(r_engine_dir1, OUTPUT);
59   pinMode(r_engine_dir2, OUTPUT);
60
61   for(int i = 0; i < sensorCount; i++){
62     pinMode(sensorPorts[i], INPUT);
63   }
64
65   ride(1, topSpeed * kalibracjaL / topSpeed, topSpeed * kalibracjaR / topSpeed);
66 }
67
68 void loop() {
69   // wylicz p
70   calcError();
71
72   // wylicz d
73   derivative = error - lastError;
74
75   // opoznienie i
76
77   kiSum += (error/2) * kiJump;
78   if(kiLoop >= kiDelay){
79     integral += kiSum / kiDelay;
80     kiLoop = 0;
81     kiSum = 0;
82   }
83   kiLoop++;
84
85   if(integral >= kiMax){
86     integral = kiMax;
87   }
88   if(integral <= -kiMax){
89     integral = -kiMax;
90   }
91   // wylicz skret
92   int turn = proportional*kp + derivative*kd + integral*ki;
93
94   if(turn >= maxTurn)
95     turn = maxTurn;
96   if(turn <= -maxTurn)
97     turn = -maxTurn;

```

```

98
99     int speedL=0;
100    int speedR=0;
101
102 // oblicz skret dla poszczegolnego kola
103 if(turn>=0){
104     speedL = topSpeed * kalibracjaL / topSpeed ;
105     speedR = ((topSpeed - turn)*kalibracjaR)/topSpeed ;
106 }
107 else{
108     speedL = ((topSpeed + turn)*kalibracjaL)/topSpeed ;
109     speedR = topSpeed * kalibracjaR / topSpeed ;
110 }
111
112 // przekaz sterowanie na kola
113 if(proportional == 0){
114     ride(1, speedL, speedR);
115 } else{
116     if(turn == (-maxTurn)){
117         ride(3,topSpeed+30,topSpeed);
118     } else{
119         if(turn == maxTurn){
120             ride(4,topSpeed,topSpeed+30);
121         } else{
122             ride(1, speedL*zwolnij_m/zwolnij_d , speedR*zwolnij_m/zwolnij_d );
123         }
124     }
125 }
126
127 // opoznienie d
128 kdSum += error;
129 if(kdLoop >= kdDelay){
130     lastError = kdSum / kdDelay;
131     kdLoop = 0;
132     kdSum = 0;
133 }
134 kdLoop++;
135
136 }
137
138 // funkcja do sterowania jazda
139 void ride(int dir, int ftlSpeedl, int ftlSpeedr){ //0-stop, 1-do przodu, 2-do
140     tylu, 3-lewe kolo do przodu, prawe do tylu, 4-lewe kolo do tylu, prawe do
141     przodu
142     if(dir == 0){
143         digitalWrite(l_engine_dir1, LOW);
144         digitalWrite(l_engine_dir2, LOW);
145
146         digitalWrite(r_engine_dir1, LOW);
147         digitalWrite(r_engine_dir2, LOW);
148     }
149     if(dir == 1){
150         digitalWrite(l_engine_dir1, LOW);
151         digitalWrite(l_engine_dir2, HIGH);
152
153         digitalWrite(r_engine_dir1, LOW);
154         digitalWrite(r_engine_dir2, HIGH);
155
156         analogWrite(l_engine_speed, ftlSpeedl);
157         analogWrite(r_engine_speed, ftlSpeedr);
158     }
159     if(dir == 2){
160         digitalWrite(l_engine_dir1, HIGH);

```

```

159     digitalWrite(l_engine_dir2 , LOW);
160
161     digitalWrite(r_engine_dir1 , HIGH);
162     digitalWrite(r_engine_dir2 , LOW);
163
164
165     analogWrite(l_engine_speed , ft1Speedl);
166     analogWrite(r_engine_speed , ft1Speedr);
167 }
168 if(dir == 3){
169     digitalWrite(l_engine_dir1 , HIGH);
170     digitalWrite(l_engine_dir2 , LOW);
171
172     digitalWrite(r_engine_dir1 , LOW);
173     digitalWrite(r_engine_dir2 , HIGH);
174
175
176     analogWrite(l_engine_speed , ft1Speedl);
177     analogWrite(r_engine_speed , ft1Speedr);
178 }
179 if(dir == 4){
180     digitalWrite(l_engine_dir1 , LOW);
181     digitalWrite(l_engine_dir2 , HIGH);
182
183     digitalWrite(r_engine_dir1 , HIGH);
184     digitalWrite(r_engine_dir2 , LOW);
185
186
187     analogWrite(l_engine_speed , ft1Speedl);
188     analogWrite(r_engine_speed , ft1Speedr);
189 }
190 }
191
192 //funkcja xor
193 int mxor(int a, int b){
194     if( a==b ){
195         return 0;
196     }else{
197         return 1;
198     }
199 }
200
201 // wylicz odchylenie od prawidlowego kierunku jazdy i zapisz do p
202 void calcError(){
203     int sum = 0;
204     int pos = 0;
205
206     if(mxor(changeColor , digitalRead(sensorPorts[0])) == 1){ // sprawdzenie
207         sensora 1
208         pos += -6;
209         sum++;
210     }
211     if(mxor(changeColor , digitalRead(sensorPorts[1])) == 1){ // sprawdzenie
212         sensora 2
213         pos += -2;
214         sum++;
215     }
216     if(mxor(changeColor , digitalRead(sensorPorts[2])) == 1){ // sprawdzenie
217         sensora 3
218         pos += 0;
219         sum++;
220     }

```

```

218     if(mxor(changeColor, digitalRead(sensorPorts[3])) == 1){ // sprawdzenie
219         sensora 4
220         pos += 2;
221         sum++;
222     }
223     if(mxor(changeColor, digitalRead(sensorPorts[4])) == 1){ // sprawdzenie
224         sensora 5
225         pos += 6;
226         sum++;
227     }
228     if(sum == 0){ // jezli zaden sensor nie jest aktywny
229         error = lastError;
230     } else{
231         error = pos/sum;
232     }
233     proportional = error;
234 }
```

5 Czas wykonania

czynność	przybliżony czas(h)
projekt płytka	6
projekt robota	1
zakupy	1,5
program	5
testy	10
druk projektu	0,7
termotransfer	0,6
trawienie	0,7
wiercenie	0,7
lutowanie	1
składanie robota	0,4
pierwszy rozruch	2
dostosowanie czujników	3

W wykonaniu można zrównoleglić wykonanie płytka z składaniem. Pisanie kodu też może wykonywać inna osoba z grupy. Do dostosowanie czujników lepiej jednak mieć gotowy program i na bieżąco sprawdzać zachowanie robota. Trudno ocenić czas testowanie i pisania programu, gdyż czynności te przeplatały się. Jak można zauważyc dostosowanie czujników to bardzo czasochłonne zajęcie. Zabrało nam to więcej czasu gdyż okazało się, że na naszych trasach wzięliśmy trochę grubszą linię i próbowaliśmy uwzględnić to w kodzie.

Pierwszy rozruch również sprawił problemy, gdyż prawdopodobnie nie wszystkie elementy były dobrze przylutowane. Dodatkowe problemy sprawiły prawdopodobnie mikrokontroler do którego wgrywaliśmy bootloader. Rozwiązaniem może być wzięcie z pewnego źródła gotowego mikroukładu z gotowym bootloader.

6 Serwisowanie i konserwacja

Regularnego przeglądu wymagają czujniki. Robota wyposażymy w system do ich regulacji. Wystarczą do tego obcegi. Należy robota utrzymywać możliwie blisko trasy, ale tak by koła jej nie dotykały. Każdy z czujników na swojej wierzchniej stronie posiada czerwoną diodę LED, które poinformuje nas czy czujnik poprawnie wykrywa linię. Ważną kwestią jest grubość linii.

Najlepsze wyniki uzyskaliśmy dla około 2.0cm. Aktualnie robot jest ustawiony na 1.5cm, które było na RoboDayu. Im szersza linia tym szerzej należy rozstawić trzy wewnętrzne czujniki.

Baterie należy wymieniać po około 2 motogodzinach. Zalecamy wybór baterii alkaicznych AA o napięciu 1.5V - popularne "paluszki". Do pojemnika wchodzi 6 takich baterii i wszystkie są wymagane do poprawnej pracy silników i elektroniki. Silniki i elektronika są zasilane z jednego źródła i nie wykazaliśmy zakłóceń tym spowodowanych.