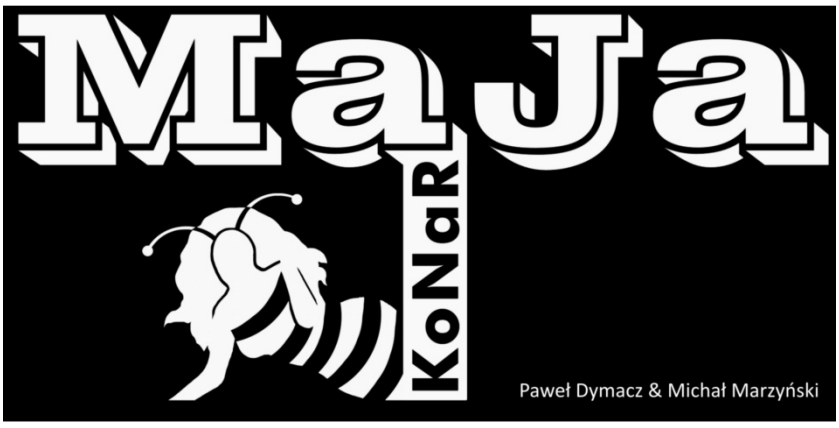
Koło Naukowe Robotyków „KoNaR”

Robot balansujący wykorzystujący algorytm uczenia ze wzmocnieniem.



Paweł Dymacz
Michał Marzyński

Spis treści

Wstęp 4
Konstrukcja mechaniczna 4
Konstrukcja i zasada pomiaru 4
Części 5
Rama 5
Bateria 5
Jednostka napędowa 5
Układ przeniesienia napędu 5
Symulacja komputerowa 7
Zastosowanie 7
Środowisko symulacji komputerowej 8
Physx 8
Sposób sterowania 9
Panel użytkownika 11
Ograniczenia symulacji PhysX 12
Wstęp teoretyczny 12
Implementacja regulatorów P, PI i PID 13
Regulator P 14
Regulator PI 14
Regulator PID 14
Uzyskane wyniki 15
Zestawienie z regulatorami 17
Elektronika 18
Czujniki odległości 18
Zastosowane czujniki do pomiaru odchylenia kątowego 19
Akcelerometr i żyroskop 19
Filtr Kalmana 20
Moduły elektroniki 22
Moduł jednostki arytmetyczno logicznej 22
Moduł sterownika silnika 24
Moduł czujników 26
Podsumowanie 27
Słabe strony algorytmu uczącego się 27
Mocne strony algorytmu uczącego się 28
Słuszność stosowania uczenia się ze wzmocnieniem 29

Błędy w konstrukcji mechanicznej 29
Aktualne zaawansowanie prac nad projektem 29
Bibliografia 30
Podziękowania 30
Galeria V1 31
Galeria V2 36

Wstęp

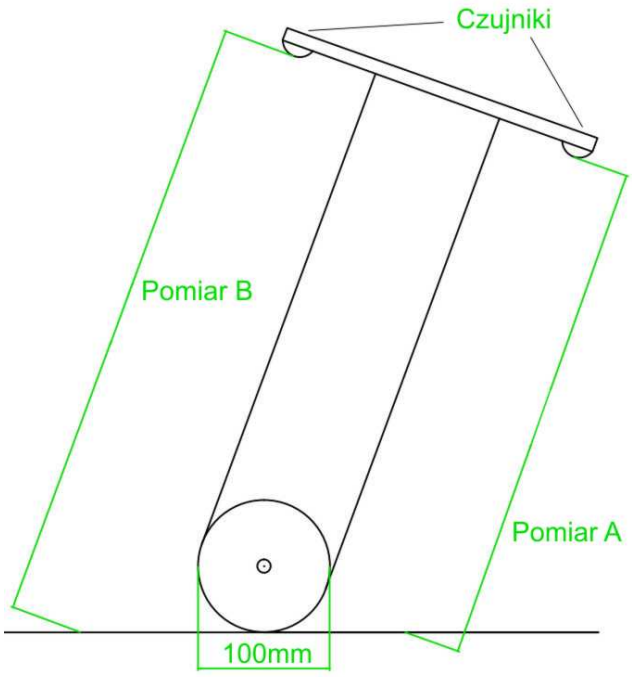
W niniejszym sprawozdaniu przedstawiono aktualnie wykonane prace nad projektem robota balansującego. Zawarte treści obejmują zdobyte doświadczenie i przedstawiają efekt końcowy prac prowadzonych przez semestr letni roku akademickiego 2009/2010, oraz uaktualnienia wprowadzone w semestrze zimowym 2010/2011. Osobami realizującymi niniejszy projekt są : Paweł Dymacz, Michał Marzyński.

Zaplanowany zakres działań do wykonania nie został niestety wykonany w całości. Dokładniejszy opis bieżących postępów, problemów i prac jeszcze koniecznych do wykonania znajduje się w dziale : Podsumowanie – Aktualne zaawansowanie prac nad projektem.

Konstrukcja mechaniczna

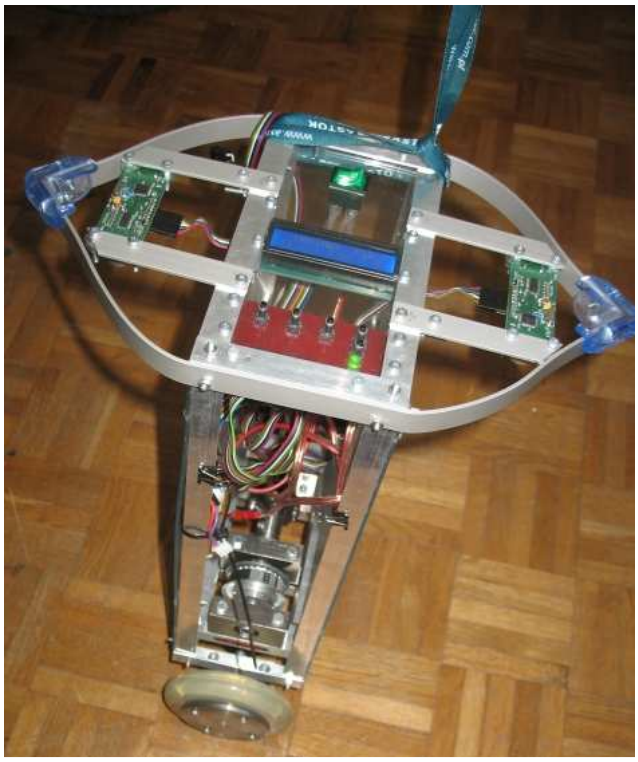
Konstrukcja i zasada pomiaru

Jednym z założeń przyjętych w planach projektu było użycie czujników odległości do pośredniego pomiaru kąta wychylenia robota ze stanu równowagi. Poniżej przedstawiony jest poglądowy schemat zasady pomiaru :



Rysunek 1 Zasada pomiaru

Praktycznie w algorytmie wystarczy mierzyć różnice w zwracanej wartości przez czujniki, ponieważ mierzone odległości A i B są liniowo zależne od kata wychylenia robota od pozycji pionowej. Rama robota mogła by być niższa, jednak umieszczając czujniku odległości wyżej osiągamy mniejszy błąd względny pomiaru. Poniżej zdjęcie przedstawiające rzeczywisty wygląd robota :



Zdjęcie 1 Robot

Części

Rama

Rama robota została wykonana z profili aluminiowych zakupionych w Castoramie. Użyto kształtów : płaskiego, kątowego płaskiego oraz rurek aluminiowych o średnicy zewnętrznej 10mm. Wszystkie elementy ramy robota były skręcane śrubami 3mm, w zależności od punktu mocowania oraz jego wagi w wielu miejscach zostały zastosowane nakrętki z kołnierzem plastikowym zabezpieczającym nakrętkę przed odkręceniem (np. ze względu na wibracje). Zaznaczmy również, że niezmatowione profile aluminiowe są tańsze od tych matowych.

Zastosowanie takich materiałów pozwala na szybkie i łatwe zmontowanie zaplanowanej konstrukcji.

Bateria

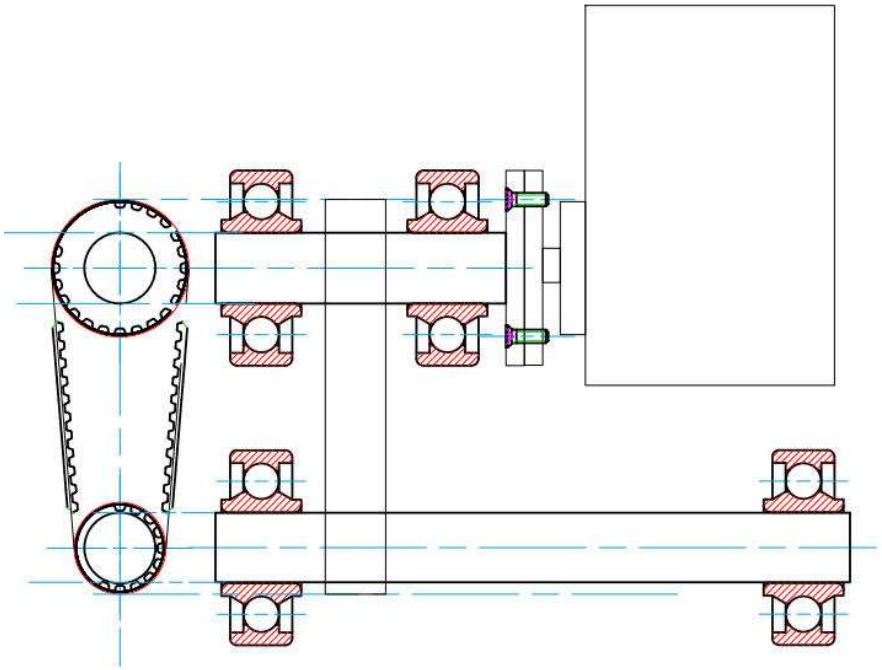
Bateria litowo polimerowa firmy Nosram model VTEC RX 2400 mAh 7,4 V. Bateria została zamontowana na dnie ramy robota, zakryta profilami aluminiowymi oraz elementem z pleksiglasu.

Jednostka napędowa

Serwomechanizm firmy Alturn model AAS-850 Quarter Scale High Torque. Podstawowe parametry to : moment 25,9 kg (przy 6 V), obrót o 60 stopni w 0,13 s (przy 6V), gabaryty 58x75x29 mm, masa 146 gram, metalowe elementy przekładni, podwójnie łożyskowane. Serwomechanizm został pozbawiony elektroniki sterującej i pełni rolę silnika DC scalonego z przekładnią.

Układ przeniesienia napędu

Ze względu na zastosowanie jednego silnika oba koła są połączone na sztywno osią. Orczyk serwa jest przymocowany do drugiej osi. Obie osie są połączone pasem zębatym w przełożeniu 3:2 (ilość zębów koła na osi serwa : osi kół) dzięki czemu uzyskamy większe prędkości obrotowe kół. Ze względu na potrzebę dużego naprężenia pasa zębatego obie osie muszą być zamontowane niezależnie na łożyskach. Poniżej przedstawiony jest poglądowy rysunek sposobu przeniesienia napędu :

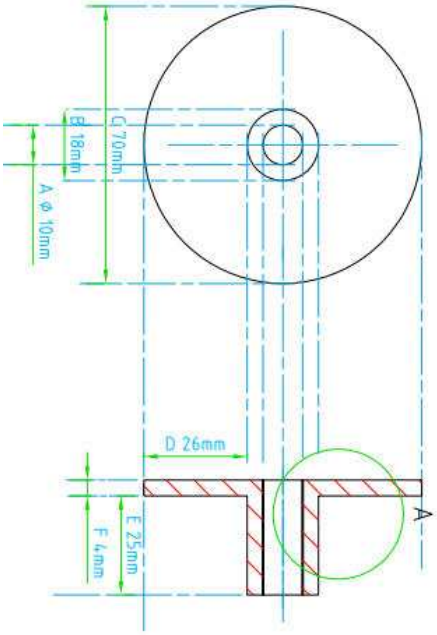


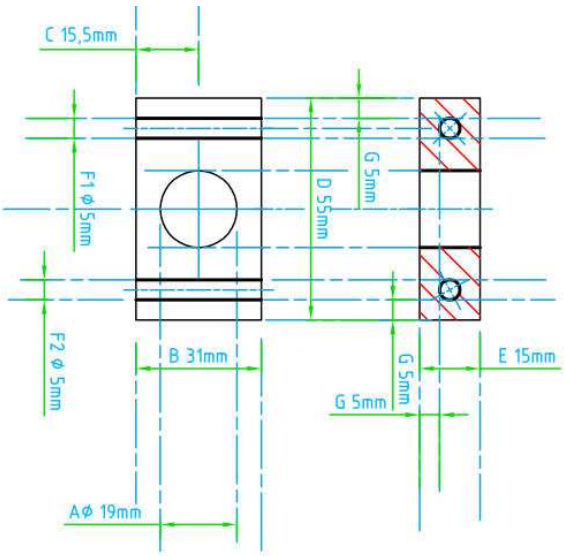
Rysunek 2 Układ przeniesienia napędu

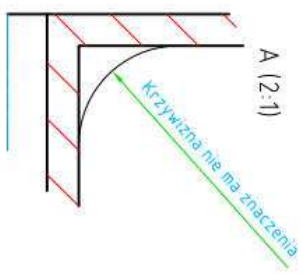
Do zrealizowania powyższego schematu zostały skompletowane następujące części :

- Pas zębaty optibelt ZR 110 XL (11 cali długości całkowitej), podziałka 5,08 mm, profil zamknięty.
- Koła zębate : optibelt ZRS XL, 16 i 24 zęby, materiał : stal/aluminium.
- Łożyska toczne do wału napędowego, 10x19x5, 61800 typ 2RS.
- Rura aluminiowa, średnica zewnętrzna 10mm.

Do zamocowania kół oraz orczyka na osie zostały wykonane elementy na zamówienie, przedstawione poniżej :







Rysunek 4 Łoże

Rysunek 3 Orczyk

Zdjęcie 2Części wykonywane na zamówienie

Zgodnie z dodatkowymi założeniami przedstawionymi w poprzedniej wersji sprawozdania w układzie przeniesienia napędu została wymieniona oś – rura, na oś litą aluminiową. Dzięki temu poprawiła się sztywność oraz sposób blokowania elementów przenoszących moment napędowy : kół zębatych, orczyka oraz kół.

Symulacja komputerowa

Zastosowanie

Metoda uczenia ze wzmocnieniem jest atrakcyjna dzięki możliwości uczenie się algorytmu na podstawie tego w jakie stany wchodzi agent oraz jakie akcje podejmuje aby zrealizować swój cel, otrzymać największą nagrodę. Dokładny opis działania algorytmu jest zawarty w następnym rozdziale.

Stworzony algorytm wstępnie miał służyć do dobrania optymalnych nastaw regulatora PID sterującego robotem balansującym. Ze względu na nieznajomość dokładnych parametrów fizycznych robota (nawet model matematyczny musi być oparty na szeregu przybliżeń) algorytm gwarantujący nauczenie się robota balansować na podstawie wielu prób jest zachwycającą alternatywą. Nie jest wtedy konieczne układanie modelu matematycznego. Metoda staje się atrakcyjniejsza jeśli wchodzimy w przestrzeń sterowania złożonymi modelami robotów których opis matematyczny jest ogromnym problemem dla studenta.

Środowisko symulacji komputerowej

Physx

Termin PhysX odnosi się do „silnika fizyki" – narzędzia programistycznego (Software Development Kit) umożliwiającego uzyskiwanie w grach komputerowych efektów specjalnych ściśle naśladujących zachowanie się rzeczywistych obiektów fizycznych (obiektów które podlegają powszechnym prawom fizyki : siłom grawitacji, tarciu, sprężystości itd.). PhysX wykorzystuje technologię CUDA (umożliwiająca wykorzystanie mocy obliczeniowej procesora graficznego do rozwiązywania ogólnych problemów numerycznych). Silnik PhysX SDK umożliwia nadawanie obiektom podstawowych własności fizycznych (np. masy, położenia, prędkości, przyspieszenia) i definiowanie ich oddziaływań między sobą (np. poprzez zderzenia, tarcie, przeguby) oraz z otoczeniem (np. w skutek działania grawitacji, antygrawitacji, podmuchów wiatru). Symulacja robota była przeprowadzana za pośrednictwem CUDA 2.0 w SDK wersji .8.3 (v2.8.3). Informacje na ten temat jak i software do darmowego ściągnięcia jest dostępny na stornie : http://developer.nvidia.com/object/physx.html

Do stworzenia symulacji została wykorzystana lekcja 202 (Lesson202.cpp) która zawiera zestaw możliwych do utworzenia joint-ów (połączeń pomiędzy implementowanymi obiektami). Poniżej znajduje się krótki opis realizacji podstawowych elementów i określenia interakcji pomiędzy nimi. W celu uproszczenia, obiektem do sterowania jest model odwróconego wahadła. Na jeden z obiektów nie działają prawa fizyki, symbolizuje oś połączenia (zawias), flaga braku podlegania prawo fizyki : `NX_BF_KINEMATIC`. Drugi jest sterowanym wahadłem. Wszystkie obiekty są tworzone jako odpowiedni kształt (kształty są dostępne programowo) :

```
//Ponizsze instrukcje w nawiasie okreslaja kolejno :
//miejsce polaczenia obu obiektow we wspolrzednych wewnetrznych,
//wymiary obiektu, gestosc
box1 = CreateBox(NxVec3(0,4,0), NxVec3(0.2,0.5,1), 10);
box1->raiseBodyFlag(NX_BF_KINEMATIC);
```

Użyte połączenie można zdefiniować jako zawias bez ograniczenia kąta obrotu wahadła. Nazwa połączenia : `NX_RJF_SPRING_ENABLED`. Połączenie to może być wykorzystane również w trybie `motor enabled` które symbolizuje wał silnika o okreslobnych parametrach (w przyszłości, można w ten sposób zamodelowac kompletnego robota w postaci bryły określającej ramę robota z dwoma kołami napędzanymi jednym silnikiem). Parametry określające połączenie :

```
springDesc.damper = 50;              //tłumienie
springDesc.targetValue = NxPi;       //okreslenie kierunki oddziaływania siły
graiwtacji
```

Realizacja ograniczenia maksymalnego wychylenia robota. Ograniczenie było możliwe przez zdefiniowanie połączenia `NX_RJF_LIMIT_ENABLED`, jednak ten typ połączenia nie umozliwiał zdefiniowania parametru tłumienia oraz charakterystyki oddziaływania (sprężystość) na obiekty połączone. Programowo prostszym rozwiązaniem z dostepnością wszystkich potyrzebnych parametrów było utworzenie dwuch odbijaków ograniczających maksymalne wychylenie robota. Zakres uchybu został ograniczony do niespełna 40 stopni, nie ma to wpływu na sam algorytm uczenia ani na porównanie wyników uzyskanych różnymi metodami, wpływa to jednak na skrócenie czasu nauki. Miejsce połaczenia obiektów jest okreslone przez współrzędne sceny :

```
//globalAnchor okresla miejsce zakotwiczenia polaczenia obu obiektow
NxVec3 globalAnchor = NxVec3(0,5,-1);
```

Sposób sterowania

Sterowanie robota rzeczywistego opiera się na sterowaniu prędkości obrotowej kół (obu na raz). Sterowanie zrealizowane w symulacji zostało ograniczone do możliwości przyłożenia siły do srodka ciezkości robota zawsze prostopadle do siły grawitacji. Możliwa wartość przykładanej siły zostałą ograniczona do MaxForce = 3095, ze względu na ograniczenie możliwej do uzyskania prędkości obortowej kół wprawdziwym modelui. Jednak taki model odbiega od rzeczywistego modelu robota który porusza się na kołach. Jednak symulacja ma służyć przedewszystkim do sprawdzenia poprawności działania algorytmu uczącego się i porównania go z podsatawowymi regulatorami. Poprawnie zaimplementowany algorytm uczący się powinien nauczyć się nie tylko regulwoać, sterowac odmiennym modelem ale również sterować w odmiennych warunkach fizycznych, czego standardowe regulatory nie potrafią.

W programie w celu normalizacji zostało przyjęte, że uchyb ma wartość ujemna jeśli robot jest pochylony w lewą stronę (zgodnie z Rysunek 1). Jednocześnie Przykładana siła, która ma zwrot po lewej stronie wektora siły (zgodnie ze zwrotem wektora określającego oś x we współrzędnych wewnętrznych każdego z obiektów) również ma wartość ujemną.

Wartość przykładanej siły regulacji jest obliczana przez regulator, sposób realizowania przykładania siły jest uzależniony od zdefiniowanej metody. Poniżej przedstawiona jest realizacja przykładania siły do robota w zależności od wyjścia regulatora (regulacja) :

```
if (regulacja<0)
  { gForceVec=ApplyForceToActor(actor,NxVec3(1,0,0) ,NxReal( abs(regulacja)
),true); }
else
  { gForceVec=ApplyForceToActor(actor,NxVec3(-1,0,0),NxReal( abs(regulacja)
),true); }
```

Wyjaśnienie wyświetlanych wartości :

Deviation - aktualne wychylenie ze stanu równowagi podawane w stopniach

Max deviation - wartość bezwzględna z maksymalnego wychylenia uzyskanego podczas trwania całej symulacji

Force - aktualna przykładana siła do robota (do środka ciężkości robota)

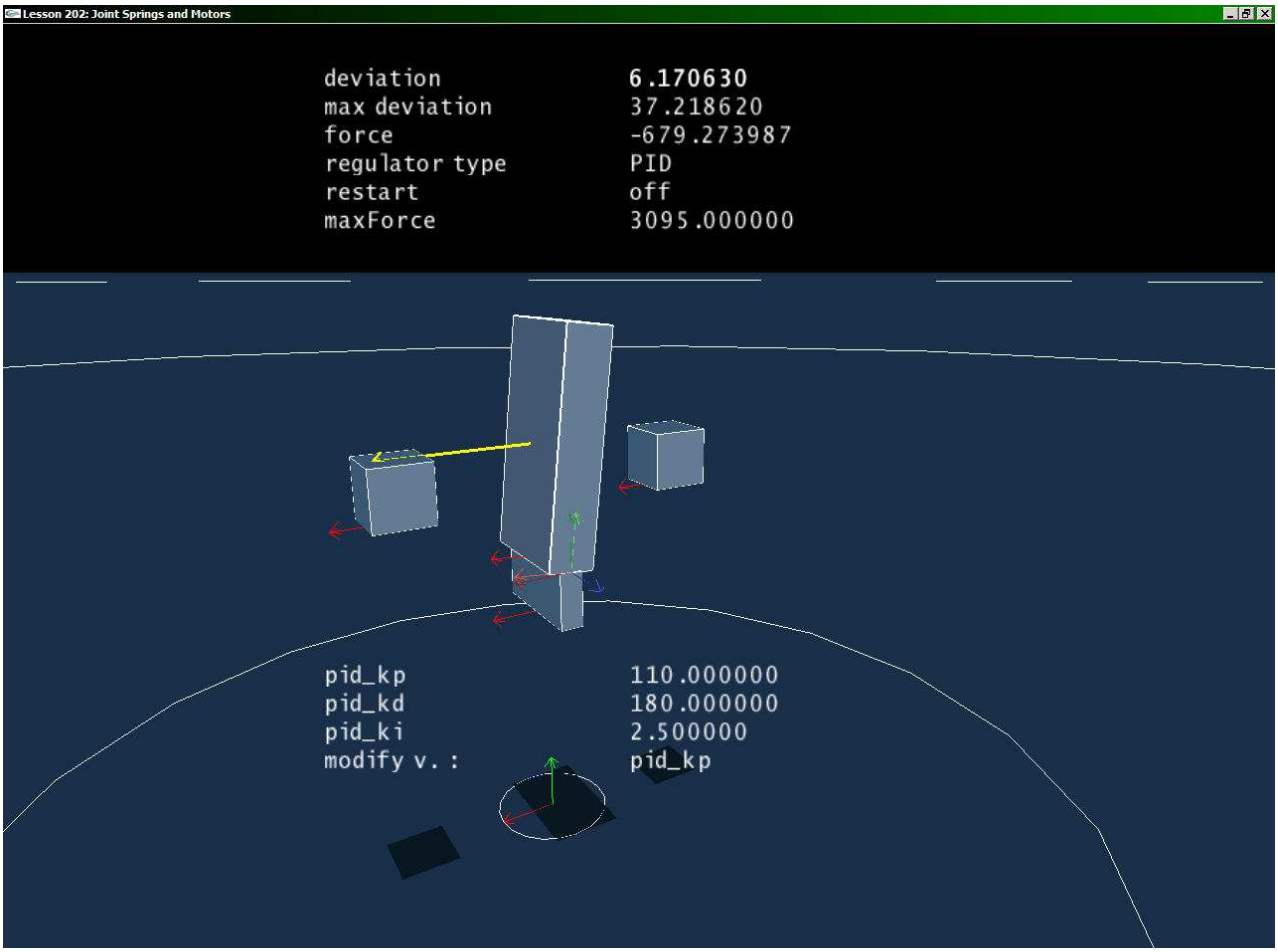
Regulator type - regulator aktualnie sterujący robotem

Restart - opcja restartowania całej symulacji jeśli robot osiągnie określone wychylenie (przyjęte na poziomie 45 stopni).

Max force - wartość bezwzględna z maksymalnej przyłożonej siły do robota

Pid_kp, pid_kd, pid_ki - wartości odpowiednich wzmocnień członów regulatora PID

Modify v. - wybrane wzmocnienie regulatora PID które można modyfikować podczas trwania symulacji



Zdjęcie 3 Wygląd okna symulacji

Widoczne w oknie symulacji strzałki (wektory) określają kierunek i zwrot (wzrost wartości) odpowiednich osi współrzędnych wewnętrznych każdego ze zdefiniowanych obiektów na scenie. Odpowiednio kolor : czerwony – oś x, zielony – oś y, niebieski – oś z. Żółty wektor bezpośrednio przyłożony do bryły robota określa kierunek i zwrot przyłożonej siły określonej przez regulator.

Panel użytkownika

Zdjęcie 4 Wygląd panelu użytkownika

- **Flight Controls -** Używając tych klawiszy oraz/lub lewego przycisku myszki możliwe jest dowolne ustawienie kamery pokazującej symulację.
- **Force Controls -** Przyłożenie siły do środka ciężkości wybranego aktora zgodnie z odpowiednią osią współrzędnych globalnych sceny lub o przeciwnym zwrocie i tym samym kierunku. Siła jaka jest przykładana zawsze ma wartość 3000.
- **Miscellaneous -** Opcje dodatkowe. Kolejno od góry umożliwiają : zatrzymanie symulacji, wybór aktora umożliwiając działanie na niego siłą (używając klawiszy z zakładki Force Controls, siła wyliczana przez regulator jest przykładana niezależnie do obiektu sterowanego), wyłączenie możliwości działania sił zewnętrznych na aktorów (nie działają opcje z zakładki Force Controls), wyświetlenie wszystkich wektorów określających współrzędne układów wewnętrznych brył oraz sceny, włączenie lub wyłączenie wyświetlania cieni aktorów, przesunięcie wybranego aktora w punkt określony współrzędnymi sceny (0,5,0).
- **Select regulator type -** Dostępne opcje umożliwiają wybór regulatora wykorzystywanego do sterowania modelem robota, regulator może być zmieniamy podczas działania symulacji, wybranie opcji y oznacza brak regulacji, opcja n oznacza restartowanie całej symulacji jeśli uchyb

osiągnie wartość 45 stopni (ostatecznie używanie tej opcji zostało zaniechane ze względu na czas potrzebny do ponownego uruchomienia symulacji rzędu paru sekund).
- **Modify PID regulator adjustments -** Opcje służące do zmiany wzmocnienia członów regulatora PID podczas działania symulacji.
- **Files -** Opcje służą do wczytywania i zapisywania macierzy używanej do działania algorytmu Uczenia ze wzmocnieniem. Macierz zapisywana jest pod nazwą „macierzout.bla" (aktualny stan macierz podczas działania symulacji). Wczytywanie macierzy może odbyć się w dowolnym momencie działania symulacji, plik musi nazywać się „macierzin.bla".

Ograniczenia symulacji PhysX

Podstawowym napotkanym problemem jest brak specyfikacji każdej stałej określającej podstawowe parametry fizyczne takie jak : tłumienie, gęstość czy siłę. Wszystkie podawane parametry są bez jednostki i nie byliśmy w stanie określić jednoznacznie w jaki sposób można je przeliczyć na jednostki podstawowe.

Podczas przeprowadzania symulacji napotkaliśmy również problem synchronizacji przeprowadzanych obliczeń. Ze względu na to, że silnik PhysX oraz program zawierający algorytmy wyliczające akcje na podstawie uchybów działają niezależnie. Ze względu na to każda symulacja musiała być przeprowadzana bez żadnych innych programów działających równolegle lub w tle. Skutki nie przestrzegania tej zasady polegały na opóźnieniu działania silnika fizyki lub opóźnieniu przeprowadzanych obliczeń, symulacja dawała przekłamane wyniki i nienaturalne akcje (nawet przy włączaniu się wygaszacza).

Kolejnym problemem jest przenoszalność symulacji. Pomimo działania symulacji na obu komputerach na których był pisany algorytm, wyniki przeprowadzanych testów były rozbieżne. Jest to nie dopuszczalne, przede wszystkim jeśli ma być symulowana fizyka.

Warto również zaznaczyć, że symulacji nie da się przyspieszyć. Nie jest możliwe zwielokrotnianie szybkości obliczania fizyki co determinuje potrzebny czas na badania. Więc czas potrzebny na naukę dla symulacji jest dokładnie taki sam jak dla rzeczywistego modelu robota.

Wstęp teoretyczny

Algorytm Q-learning pozwala na uczenie się funkcji wartości akcji oraz oszacowanie optymalnej funkcji wartości akcji, tak aby móc uzyskać strategię optymalną jako zachłanną względem niej. Zgodnie z algorytmem Q-learning agent wybiera akcję a_t w chwili t na podstawie oszacowania funkcji Q dla aktualnego stanu s_t. Dla danej akcji a_t w stanie s_t i oczekiwanej nagrody R_t funkcja $Q_t(s_t, a_t)$ jest definiowana jako :

$$Q_t(s_t, a_t) = (1 - \alpha)Q_t(s_t, a_t) + \alpha\left[R_t + \gamma \max_{a_t' \epsilon A} Q_t(s_t', a_t')\right]$$

Gdzie:
- A jest zbiorem wszystkich możliwych akcji a_t,
- $0 \leq \alpha < 1$ to współczynnik intensywności uczenia się,
- $0 \leq \gamma \leq 1$ to współczynnik dyskontowania,
- Współczynnik R_t jest nazywany współczynnikiem wzmocnienia (*reinforcement factor*)
- $Q_t(s_t, a_t)$ jest wartością funkcji Q w stanie s_t po wykonaniu akcji a_t.

Algorytm można przedstawić następująco w sposób uproszczony :

inicjalizuj t=0, $R_t = 0$;
while(1)
{
t=t+1;
zarejestruj bieżący stan s_t;
wybierz akcję a_t;
oblicz $R_t(s_t, a_t)$ i wyznacz s_{t+1};
zaktualizuj funkcję Q
{
$\Delta Q_t = \left[R_t + \gamma \max_{a_t} Q_t(s_{t+1}, a_t)\right]$;
$Q_t(s_t, a_t) = (1-\propto) Q_t(s_t, a_t) + \propto \Delta Q_t$;
}
}

W zależności od dobranej wielkości współczynnika α, w kolejnych pętlach działania algorytmu podczas uaktualniania danej wartości funkcji $Q_t(s_t, a_t)$ brana jest pod uwagę mniej lub bardziej wartość, którą modyfikujemy. Dlatego nazywany jest współczynnikiem intensywności uczenia – ponieważ bierzemy pod uwagę to, czego wcześniej się nauczyliśmy.

Współczynnik γ określa wagę, z jaką bierzemy pod uwagę dalsze rezultaty obecnych posunięć. Jest on zawsze co najwyżej równy 1 – czyli mniej lub co najwyżej tak samo ważny jak bezpośredni rezultat danej akcji w danym stanie.

Modyfikacja obydwu tych parametrów może prowadzić do różnych zachowań algorytmu Q-learning. W poniższym sprawozdaniu zostaną przedstawione wyniki działania algorytmu przy różnych parametrach α i γ. Zastosowana w naszym algorytmie funkcja R_t wygląda następująco :

$$R_t = r_s * |\theta|$$

Gdzie :
r_s to odpowiedni całkowitoliczbowy współczynnik
θ to stan s_{t+1} (kąt odchylenia robota od pionu)

Implementacja regulatorów P, PI i PID

Ze względu na jak najlepsze przybliżanie reguł symulacji do rzeczywistego modelu robota wartość siły która może być przyłożona do robota została ograniczona do wartości około 3095. W modelu rzeczywistym maksymalna osiągana prędkość obrotowa kół jest ograniczona parametrami użytego napędu.

Zasada regulowania robotem w symulacji opiera się na następujących powtarzanych w nieskończoność krokach :

1. Pobierz pomiar z czujnika, oblicz uchyb
2. Oblicz wyjście regulatora na podstawie uchybu
3. Przyłóż odpowiednią siłę regulacji do modelu
4. Czekaj 50ms, w tym czasie wartość zadanej siły nie zmieniała się, przyłożona siła cały czas działa na obiekt

Czas wykonywania obliczeń przez regulatory został uznany za nieistotny. Przeciwnie w implementacji algorytmu uczenia się, czas opóźnienia związany z częstotliwością pobierania pomiaru z czujnika zaczyna być liczony wraz z rozpoczęciem obliczeń algorytmu uczącego się.

Regulator P

Sterowanie regulatora proporcjonalnego zdefiniowane jest jako :

$$u(t) = K_p e(t)$$

Gdzie :

$u(t)$ - jest wyjściem regulatora

$e(t)$ - jest wartością uchybu

K_p -jest wzmocnieniem członu proporcjonalnego

Wartości parametrów w programie : `int p_kp = 160;`

Regulator jest w istocie wzmacniaczem z przestrajalnym wzmocnieniem. W programie regulator został zaimplementowany w następujący sposób :

```
regulacja = p_kp*pomiar;
Wait(0.05);
```

Polecenie `Wait(0.05);` zapewnia wymagane opóźnienie definiujące czestotliwość pozyskiwania wyniku wychylenia robota ze stanu równowagi z częstotliwością 20Hz.

Regulator PI

Sterowanie regulatora proporcjonalno całkującego zdefiniowane jest następująco :

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt$$

Gdzie :

K_i - jest wzmocnieniem członu całkującego

Wartości parametrów w programie : `float pi_kp=100; float pi_ki=0.6;`

W programie regulator wraz z ograniczeniem wyjścia w przedziale (-MaxForce, MaxForce), został zaimplementowany w następujący sposób :

```
w_uchyb = w_zadana - pomiar;
regulacja = pi_kp * (w_uchyb - poprzedniuchyb) + pi_ki * w_uchyb +
regulacja;
poprzedniuchyb = w_uchyb;
if(regulacja > MaxForce)
  { regulacja = MaxForce; poprzedniuchyb = 0;}
if(regulacja < -MaxForce)
  { regulacja = -MaxForce; poprzedniuchyb = 0;}Wait(0.05);
```

Regulator PID

Sterowanie regulatora proporcjonalno całkująco różniczkującego (dokładniej w wersji PID-IND) zdefiniowane jest następująco :

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

Gdzie :

K_d - jest wzmocnieniem członu różniczkującego

Wartości parametrów w programie : `float pid_kp=110; float pid_kd=180; float pid_ki=2.5;`

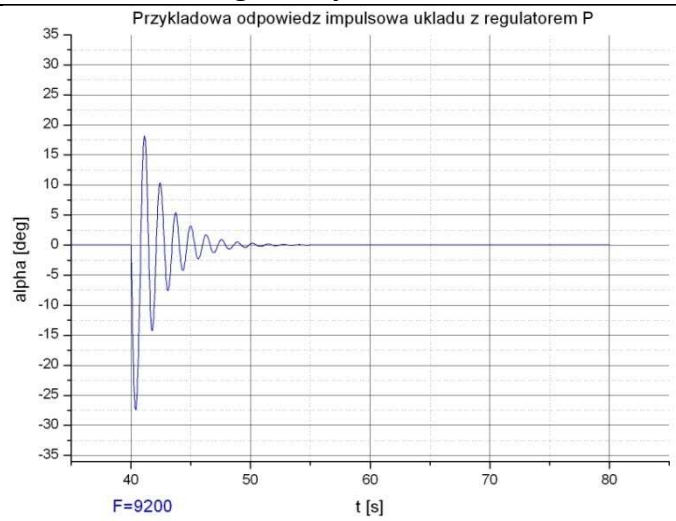
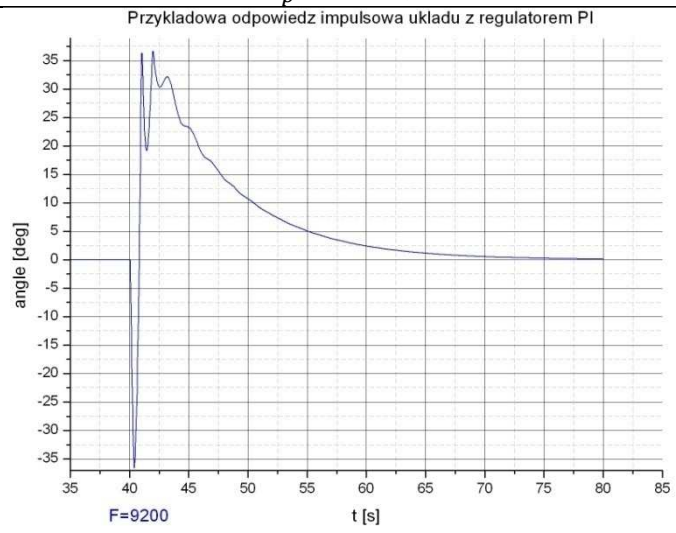
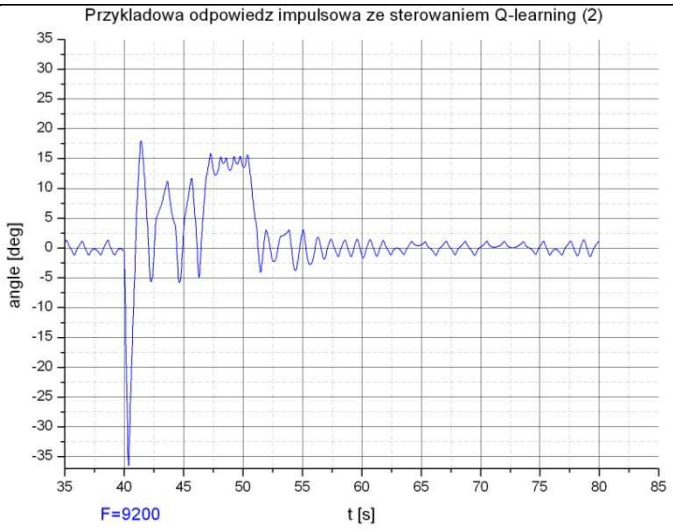
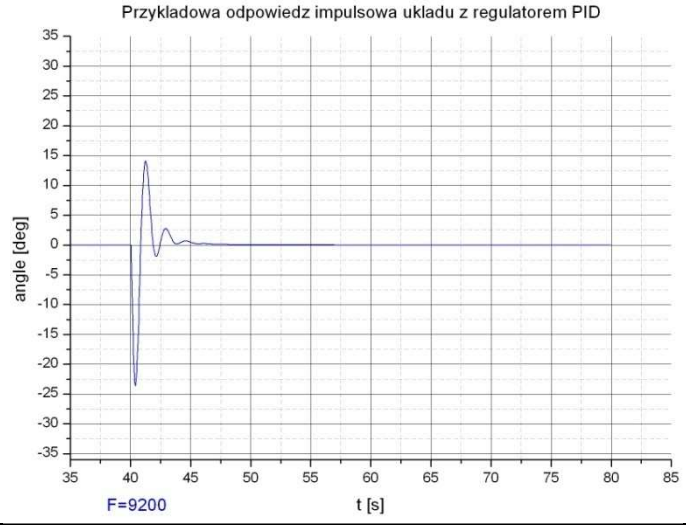
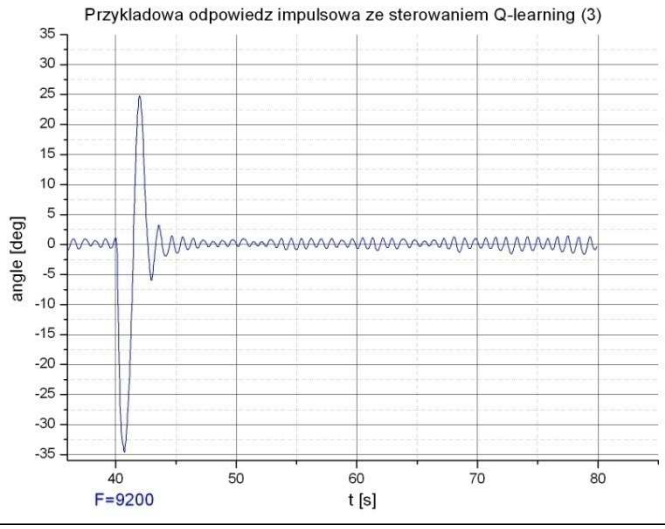
W programie regulator wraz z ograniczeniem wyjścia w przedziale (-MaxForce, MaxForce), został zaimplementowany w następujący sposób :

```
w_uchyb = w_zadana-pomiar;
calka = calka + w_uchyb;
regulacja = pid_kp*w_uchyb + pid_kd*(w_uchyb-poprzedniuchyb) +
pid_ki*calka;
poprzedniuchyb = w_uchyb;
if(regulacja > MaxForce)
  { regulacja = MaxForce; poprzedniuchyb = 0; calka = 0;}
if(regulacja < -MaxForce)
  { regulacja = -MaxForce; poprzedniuchyb = 0; calka = 0;}
Wait(0.05);
```

Sporym problemem okazało się nastawienie regulatora PID. Ze względu na brak modelu matematycznego nastawy mogły być dobrane jedynie ze względu na możliwości jakie dawała symulacja. Wykorzystując I metodę Zieglera – Nicholsa (oscylacyjna), ustawiając wzmocnienie członu proporcjonalnego (rzędu 480-510, w zależności od stanu komputera i włączonych innych programów w tle) tak aby sterowany obiekt wchodził w niegasnące oscylacje uzyskany okres niegasnących oscylacji był rzędu 1,7 sekundy. Zgodnie z tabelami użytej metody przeliczone wartości wzmocnień nie dawały zadowalających efektów. Ostatecznie wystarczające parametry regulatora PID zostały uzyskane umożliwiając użytkownikowi bezpośrednią zmianę wzmocnień podczas symulacji. Przyjętym kryterium był minimalny czas regulacji.

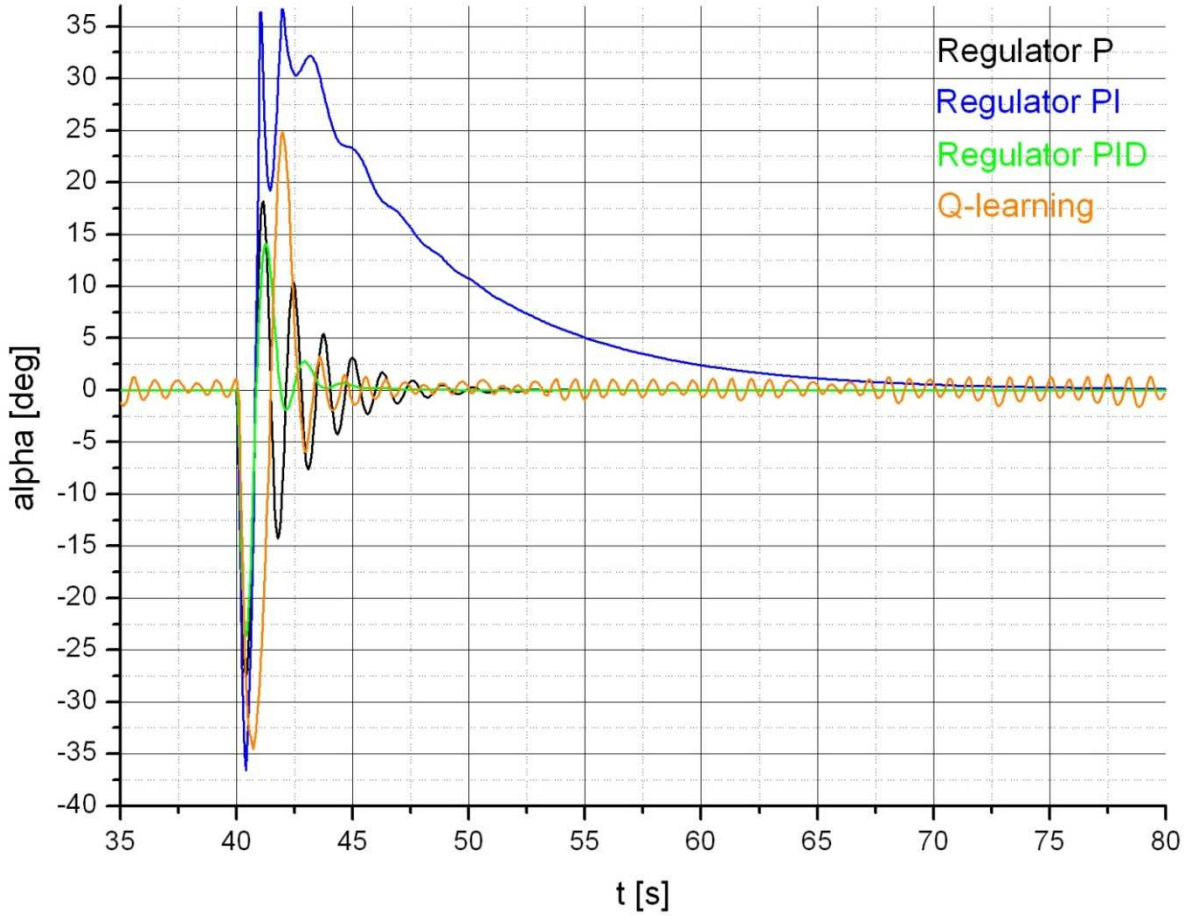
Uzyskane wyniki

Poniższe wykresy przedstawiają rezultaty osiągnięte za pomocą różnych algorytmów regulujących odchylenie robota od pionu. Za każdym razem przykładano impulsowo siłę o tej samej wartości przez 50ms. W przypadku algorytmu Q-learning, przykładana siła powodowałaby przekłamania w uczeniu, zatem na czas pobudzenia impulsowego i prowadzonych eksperymentów na obiekcie algorytm wyłącznie korzysta z wartości zapisanych w QMatrix, ale ich nie modyfikuje, czyli wyłączone jest uczenie się.

Regulatory P, PI i PID	Sterowanie algorytmem Q-learning
	
$k_p = -160;$	QMatrix[21 akcji][91 stanów]; $\propto = 0.2$; $\gamma = 0.8;$
	
$k_p = 100;\ k_i = 0.6;$	QMatrix[81 akcji][91 stanów]; $\propto = 0.2$; $\gamma = 0.8;$
	
$k_p = 110;\ k_i = 2.5;\ k_d = 180;$	QMatrix[81 akcji][91 stanów]; $\propto = 0.1$; $\gamma = 0.9;$

Zestawienie z regulatorami

Poniżej przedstawiono zestawienie odpowiedzi układu z regulatorami P, PI i PID oraz ze sterowaniem Q-learning w najlepszej jakościowo wersji (QMatrix[81 akcji][91 stanów]; $\propto = 0.1$; $\gamma = 0.9$).



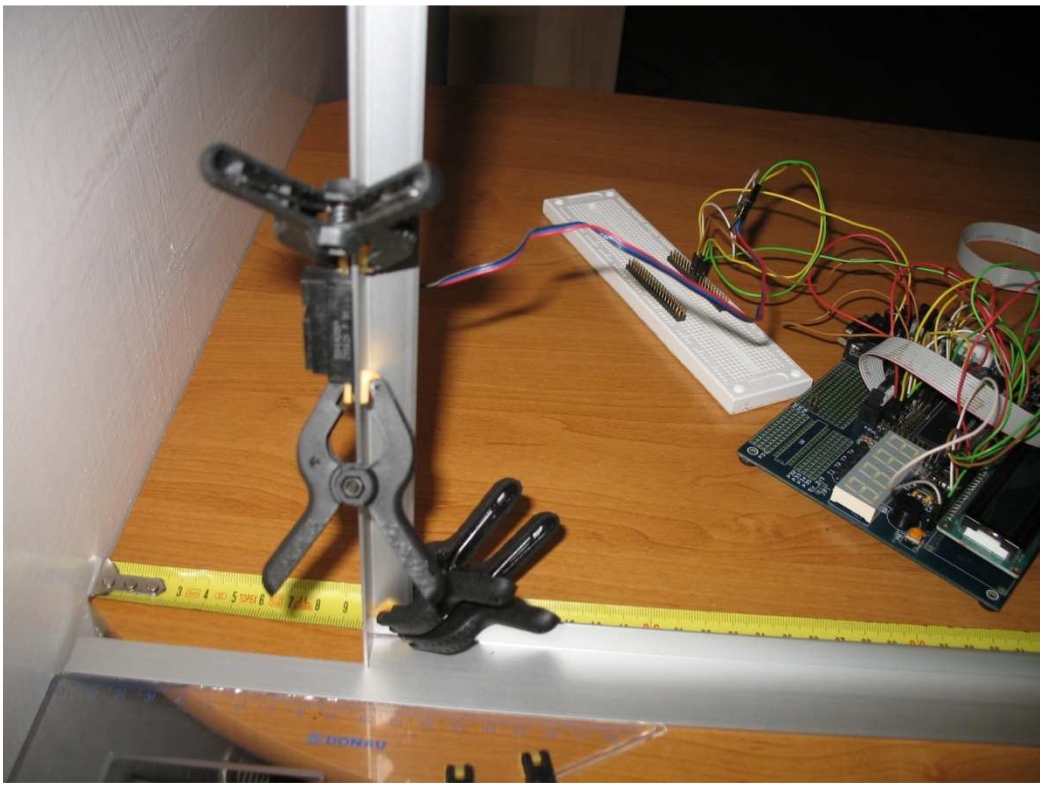
Wykres 1 Zestawienie Q-learning z regulatorami

Jak widać na powyższych wykresach, nie da się niestety doprowadzić układu na stałe do punktu równowagi przy pomocy Q-learning'u, tak jak ma to miejsce przy regulatorach P, PI i PID. Po ustabilizowaniu odchylenia wywołanego przyłożoną impulsowo siłą, występują stałe oscylacje na poziomie $\pm 2.5°$. Jeżeli przyjęlibyśmy ten zakres jako pewnego rodzaju strefę nieczułości, w której oscylacje nie mają dla nas większego znaczenia, czas takiej 'stabilizacji' byłby na poziomie algorytmu PID – lepszy od P i PI. Niestety jednak, w rzeczywistości dążymy do jak najlepszej stabilizacji i oscylacje na takim poziomie są zbyt duże.

Elektronika

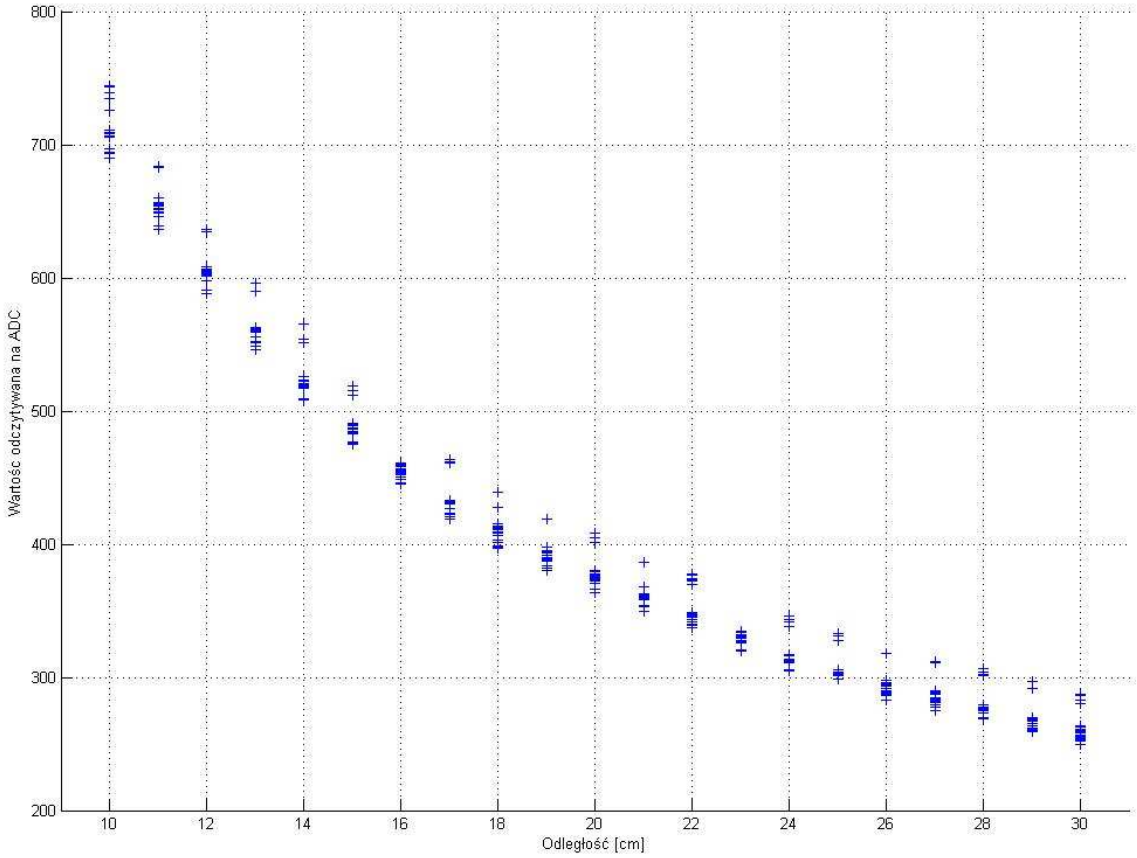
Czujniki odległości

Jednym z założeń projektu było wykorzystanie czujników odległości do pośredniego pomiaru wychylenia robota ze stanu równowagi (pozycja pionowa). Początkowo ze względu na brak elementów potrzebnych do złożenia czujników ultradźwiękowych postanowiliśmy sprawdzić możliwość wykorzystania czujników odległości SHARP 2Y0A21. Do przeprowadzenia pomiarów zostało zmontowane prowizoryczne stanowisko badawcze widoczne na poniższym zdjęciu :



Zdjęcie 5 Testy czujnika SHARP 2Y0A21

Dla odległości czujnika od powierzchni płaskiej, białej, w przedziale od 10 cm do 30 cm, co 1 cm sczytywane były wyniki podawane przez czujnik na ADC procesora ATmega16. Pomiar był wykonywany statycznie – czujnik był nieruchomy. Uzyskane wyniki przedstawia poniższy wykres :



Wykres 2 Wyniki testów czujnika SHARP 2Y0A21

Odchylenie standardowe uzyskiwanych wyników nie było większe niż kilkanaście jednostek (przy 10-bitowym ADC), również rozrzut wartości nie przekraczał kilkunastu procent. Jednak ze względu na małą stromość uzyskanej charakterystyki rozrzut zmierzonych odległości jest rzędu 4-5 cm dla mierzonej odległości około 30 cm oraz 2-3 cm dla mierzonych odległości rzędu kilkunastu centymetrów. Są to zbyt duże niepewności pomiarowe z punktu widzenia możliwości wysterowania robota balansującego na podstawie pomiaru pośredniego wychylenia robota wykorzystując czujniki tego typu.

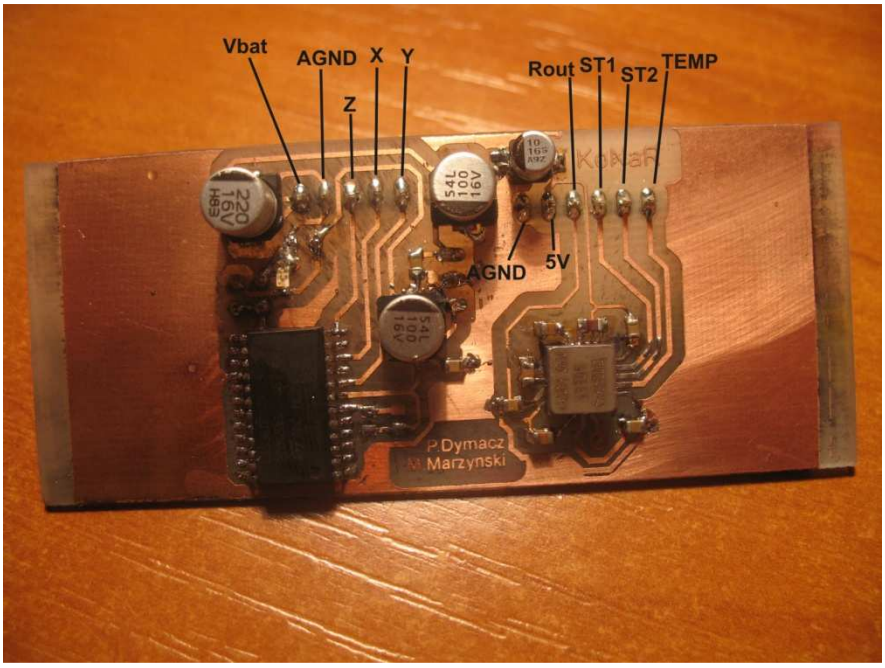
Ostatecznie w robocie zostały zamontowane czujniki ultradźwiękowe w wersji v4 których dokumentacja oraz płytka PCB zostały udostępnione przez koło. Jak się później okazało, ich zastosowanie wymagało zakrycia obudowy robota płytami pleksiglasowymi, ponieważ żebrowany kształt ramy powodował pochłanianie strzałek ultradźwiękowych i uniemożliwiał ich powrót do czujnika – i w rezultacie wyniki były nieprawdziwe. Po zakryciu obudowy problem całkowicie zniknął, a czujniki mierzą odległość od podłoża z dokładnością do 1mm.

Zastosowane czujniki do pomiaru odchylenia kątowego

Akcelerometr i żyroskop

Niestety, idealna wręcz dokładność sonarów ultradźwiękowych przy statycznych pomiarach szybko prysła, kiedy robot zaczął się ruszać – okazało się, że te czujniki wogóle nie nadają się do pomiarów w sytuacji, gdy mierzona wielkość dynamicznie i nieregularnie się zmienia. Ze względu na opłakane rezultaty powyższych sensorów grupa projektowa postanowiła zastosować czujnik przyspieszenia wraz z czujnikiem prędkości kątowej : akcelerometr LIS3L02AS4 i żyroskop ADXRS300.

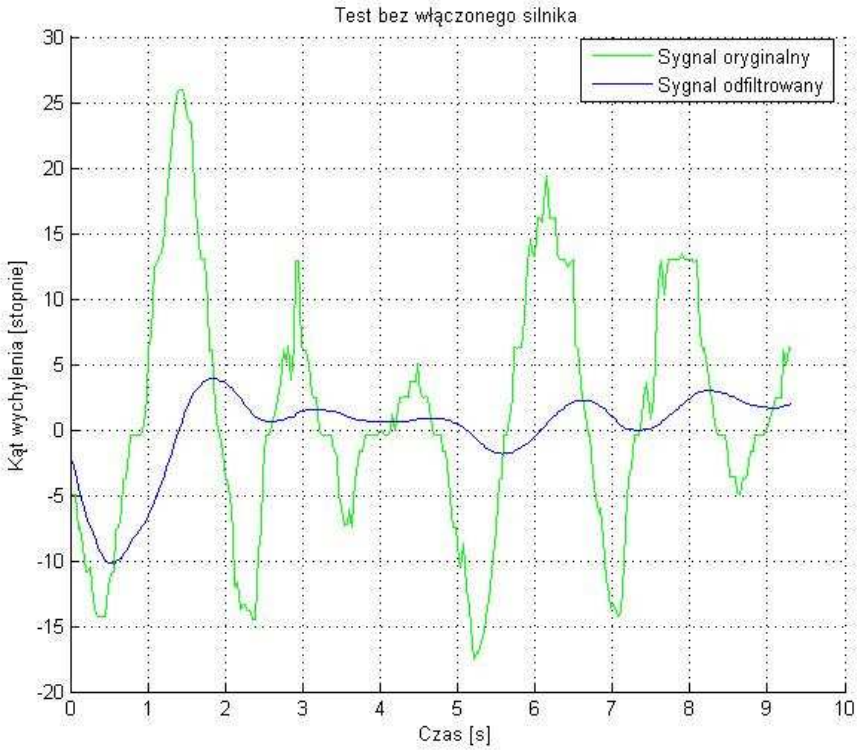
Oba czujniki zostały zamontowane na osobnej płytce w możliwie najniższym punkcie robota, centralnie nad osią. Wybór takiego miejsca montażu minimalizuje wpływ sił odśrodkowych. Moduł płytki został zamontowany na gąbce której zadaniem jest tłumienie występujących drgań mechanicznych. Ze względu na to, że sygnały pomiarowe są w postaci analogowej do przesyłu pomiędzy modułami zostały zastosowane kable ekranowane. Planujemy wykonywać pomiar z czujników z częstotliwością 60Hz przy taktowaniu procesora 16MHz, jednak gotowy algorytm zawierający wszystkie niezbędne instrukcje nie był jeszcze testowany. Ilość instrukcji obciążających procesor może wpłynąć na zwiększenie czasu trwania przerwań w których jest wykonywany pomiar oraz obliczana wartość sterowania.



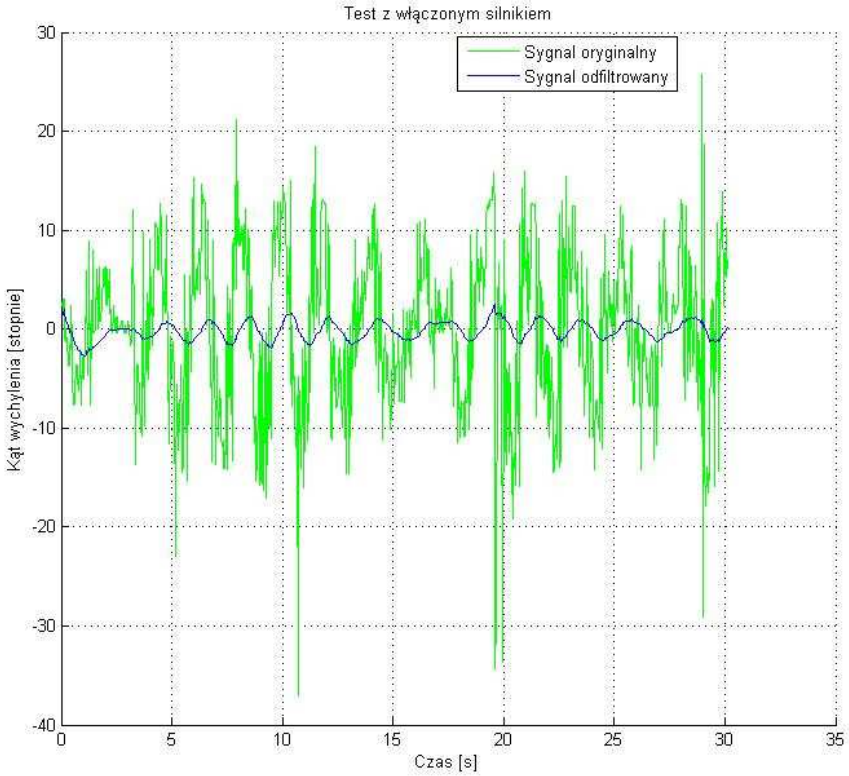
Zdjęcie 6 Moduł płytki z czujnikami

Filtr Kalmana

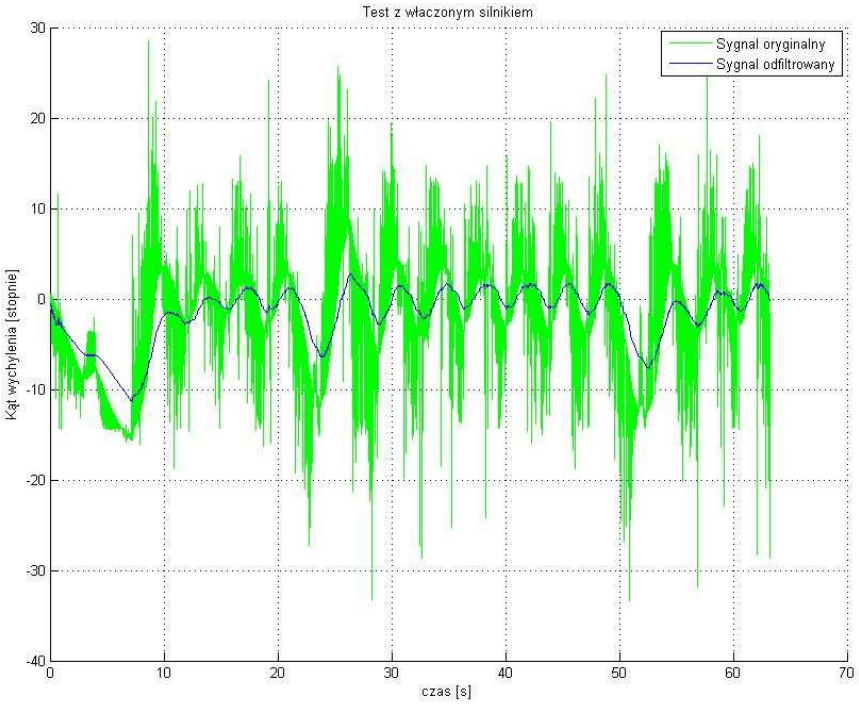
Do poprawnego i optymalnego wykorzystania pomiarów z czujników (również fuzji) został zastosowany filtr Kalmana. W aktualnym stanie na robocie zostały przeprowadzone testy jedynie z zamontowanych akcelerometrem. Dzięki zapewnionej transmisji RS232 zostały sczytane serie danych które jednoznacznie pokazują co zwracają czujnik oraz filtr w robocie. Warto zwrócić również uwagę na ogromny wpływ działania silnika na stabilność całego układu (silnik jest źródłem szumów) oraz na świetne działanie filtru. Poniżej przedstawiamy kilka przykładów danych ilustrujących działanie filtra (aktualnie jest to wersja filtrująca jedynie pomiar z akcelerometru, w nieodległej przyszłości planujemy wykorzystać oba czujniki – żyroskop i akcelerometr, wykorzystując akcelerometr głównie do kalibracji dryftu żyroskopu), poniższe wykresy mają za zadanie przede wszystkim pokazać wpływ włączonego silnika na pomiar, parametry filtra dla wszystkich przykładów : Q=0.000001, R=0.1, pomiar wykonywany co T=32.768ms :



Wykres 3 Filtracja przeprowadzona na komputerze (linia niebieska), sygnał z akcelerometru (linia zielona), silnik wyłączony



Wykres 4 Filtracja przeprowadzona na komputerze (linia niebieska), sygnał z akcelerometru (linia zielona), silnik włączony



Wykres 5 Wynik filtracji przeprowadzonej przez procesor robota (linia niebieska), sygnał z akcelerometru (linia zielona), silnik wyłączony

Moduły elektroniki

Zastosowanym procesorem w robocie jest ATmega32 16AU. Głównym powodem wyboru tego procesora była ilość dostępnej pamięci programowalnej flash potrzebnej w algorytmie uczącym się (32K). Sterowanie silnikiem odbywało się przez układ scalony firmy STMicroelectronics L298, którego odpowiednie wejścia sterujące oraz wyjścia zostały połączone równolegle umożliwiając pracę przy pobieranym przez silnik prądzie 3A (dokładny sposób połączeń pokazany jest w następnym dziale).

Dwa sonary ultradźwiękowe komunikują się z jednostką centralną naprzemiennie przy pomocy interfejsu szeregowego SPI.

Do przechowywania modyfikowalnej macierzy tworzonej w algorytmie uczenia ze wzmocnieniem po wyłączeniu zasilania użyty został eeprom Atmel 24c64 (architektura 8-bitowa, 64K pamięci) wykorzystujący sprzętowy interfejs I2C.

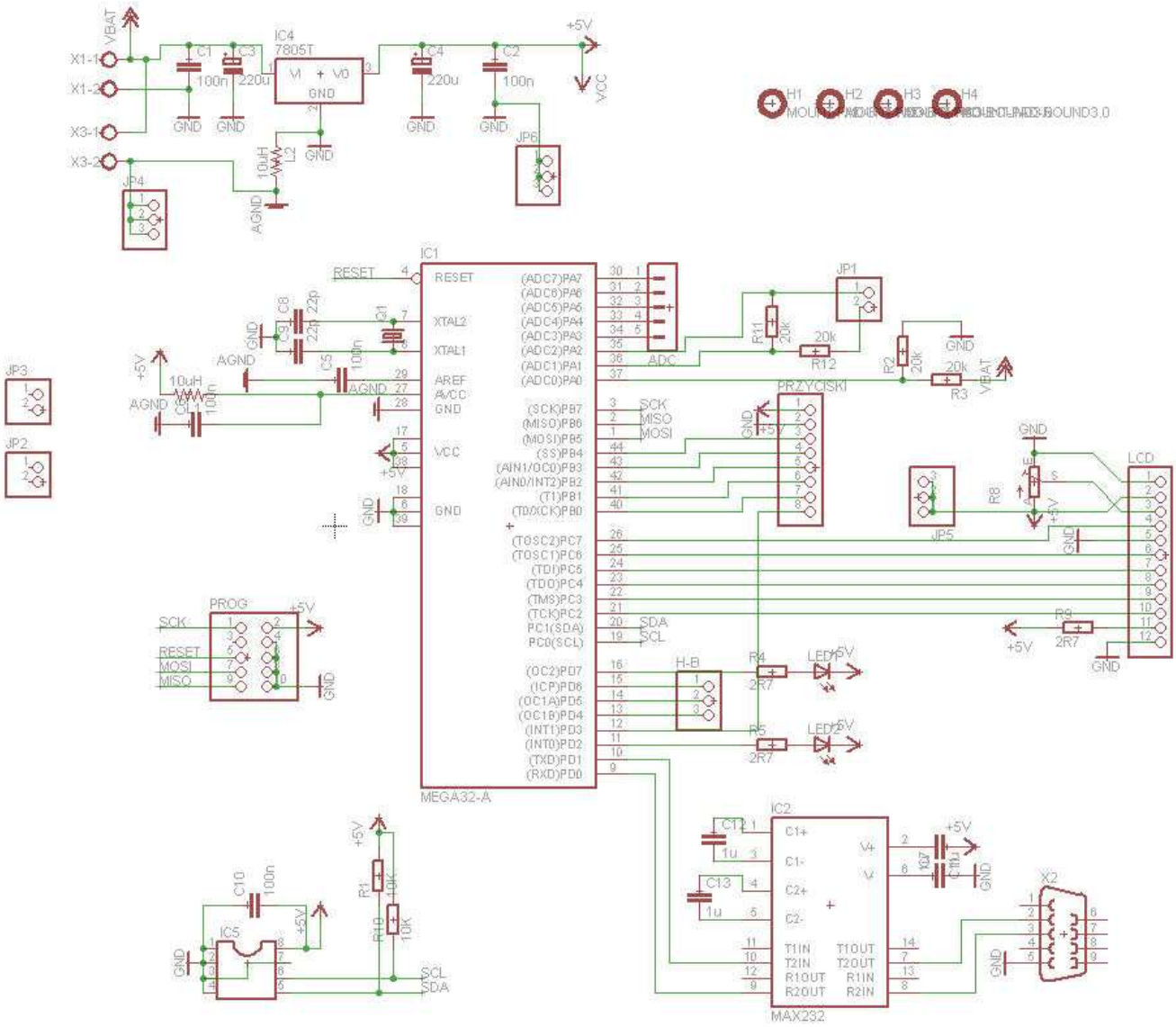
Robot został wyposażony również w wyświetlacz 16x2 znakowy, oraz klawiaturę 5 przyciskową, co umożliwia zmianę parametrów roboczych platformy podczas testów bez potrzeby programowania.

Elektronika była projektowana w darmowej wersji programu Eagle5.7.0. Zgodnie z założeniami przedstawionymi w starszej wersji sprawozdania robot został wyposażony w nową wersję płytek sterujących w których skład wchodzą moduły (pomijając klawiaturę):

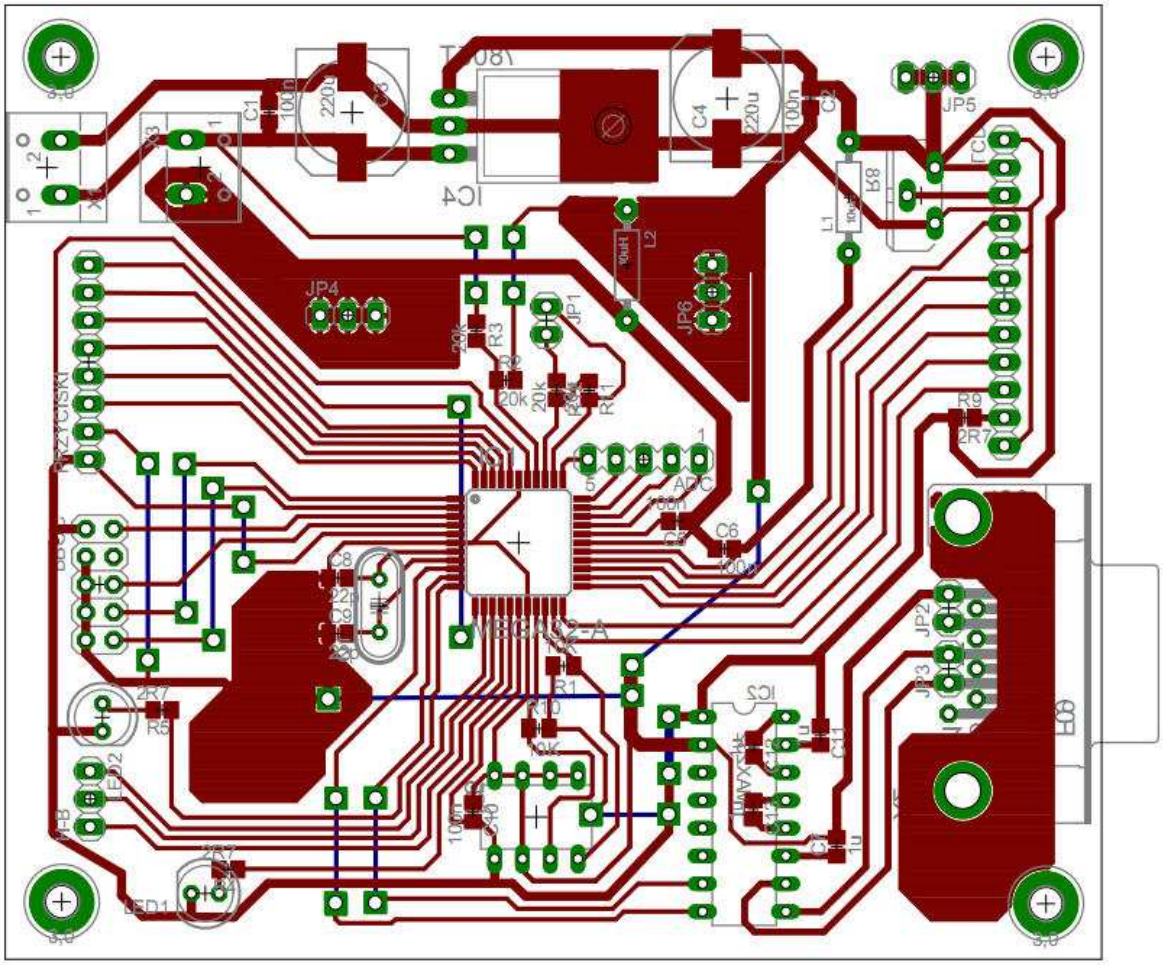
Moduł jednostki arytmetyczno logicznej

Moduł z procesorem sterującym ATmega32, zapewniona komunikacją RS232, możliwością wykorzystania pozostałych portów procesora początkowo zaplanowanych jako nieużywane. W projekcie elektroniki zostały wydzielone masy analogowe i cyfrowe w celu zminimalizowania szumów

powstających głownie w wyniku działania silnika. Poniżej przedstawiam schemat oraz PCB (ścieżki czerwone – warstwa top, ścieżki niebieskie – warstwa bottom) :



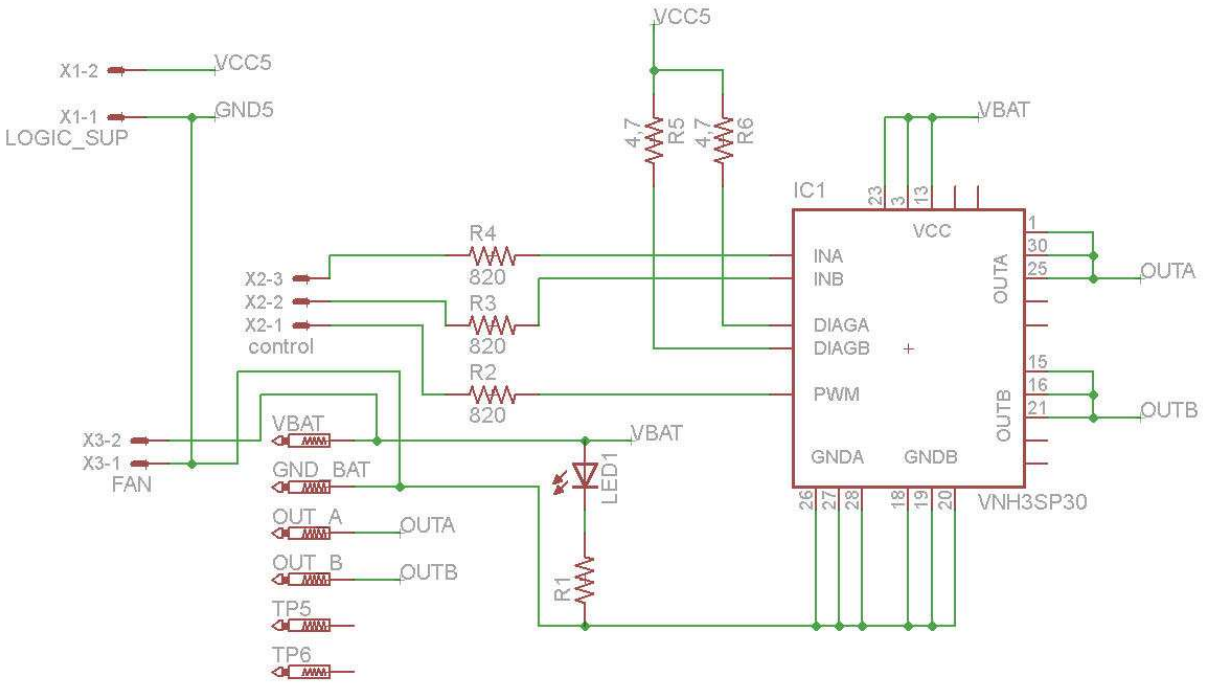
Schemat 2 Elektronika sterująca - schemat



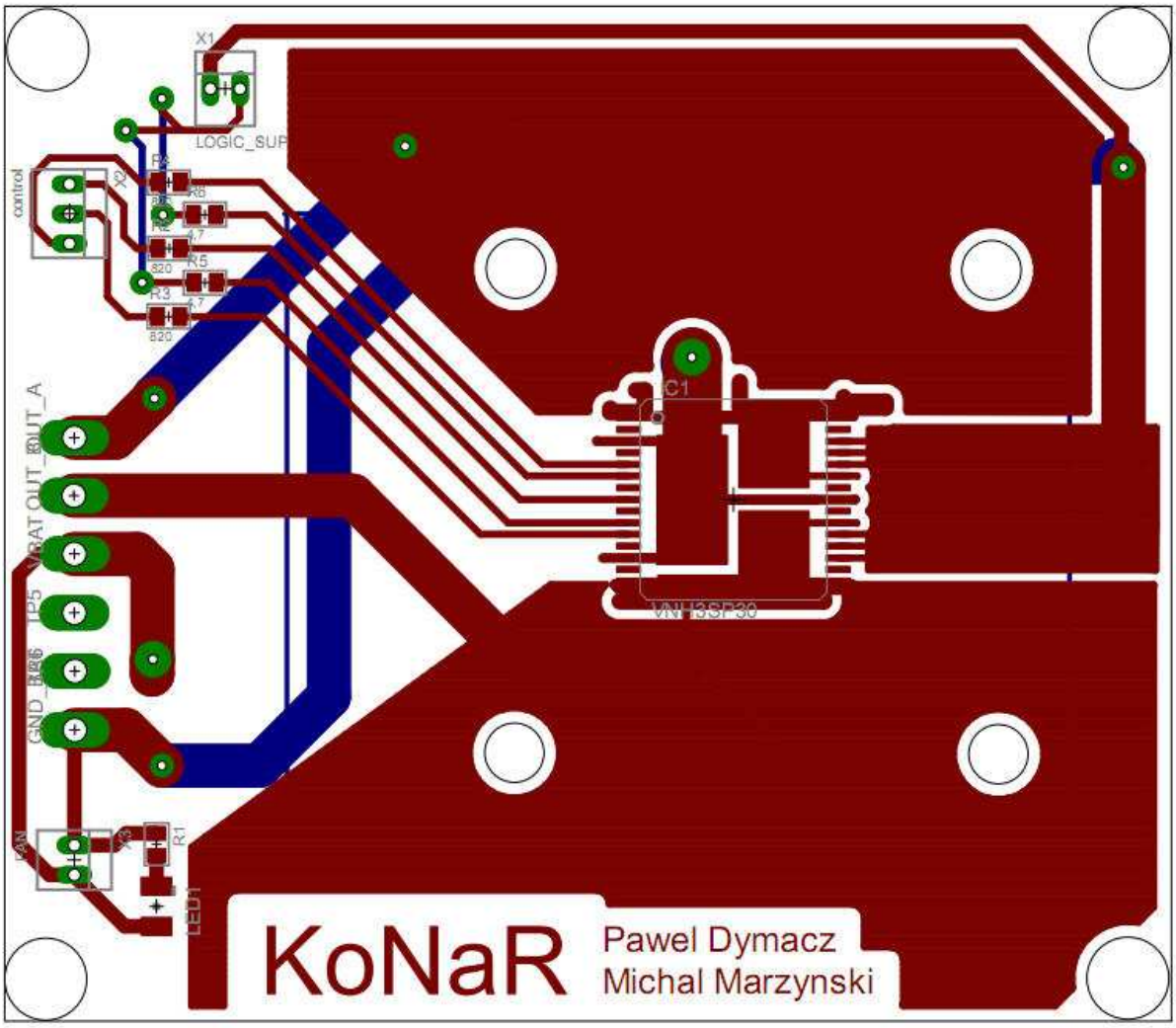
Schemat 3 Elektronika sterująca - płytka

Moduł sterownika silnika

Moduł zawierający scalony sterownik silnika VNH2SP30 (pojedynczy mostek H), dzięki czemu poprawiona została przede wszystkim sprawność sterowania silnikiem, ze względu na zbytnie nagrzewanie się poprzedniego sterownika (L298). Ponadto w poprzedniej wersji sprawozdania zaznaczaliśmy, że moc silnika wydaje się zbyt mała, aktualny sterownik umożliwia zastosowanie silnika pobierającego 30A prądu ciągłego, co znacznie przekracza nasze zapotrzebowanie (jednocześnie pozwala na upgrade w przyszłości)



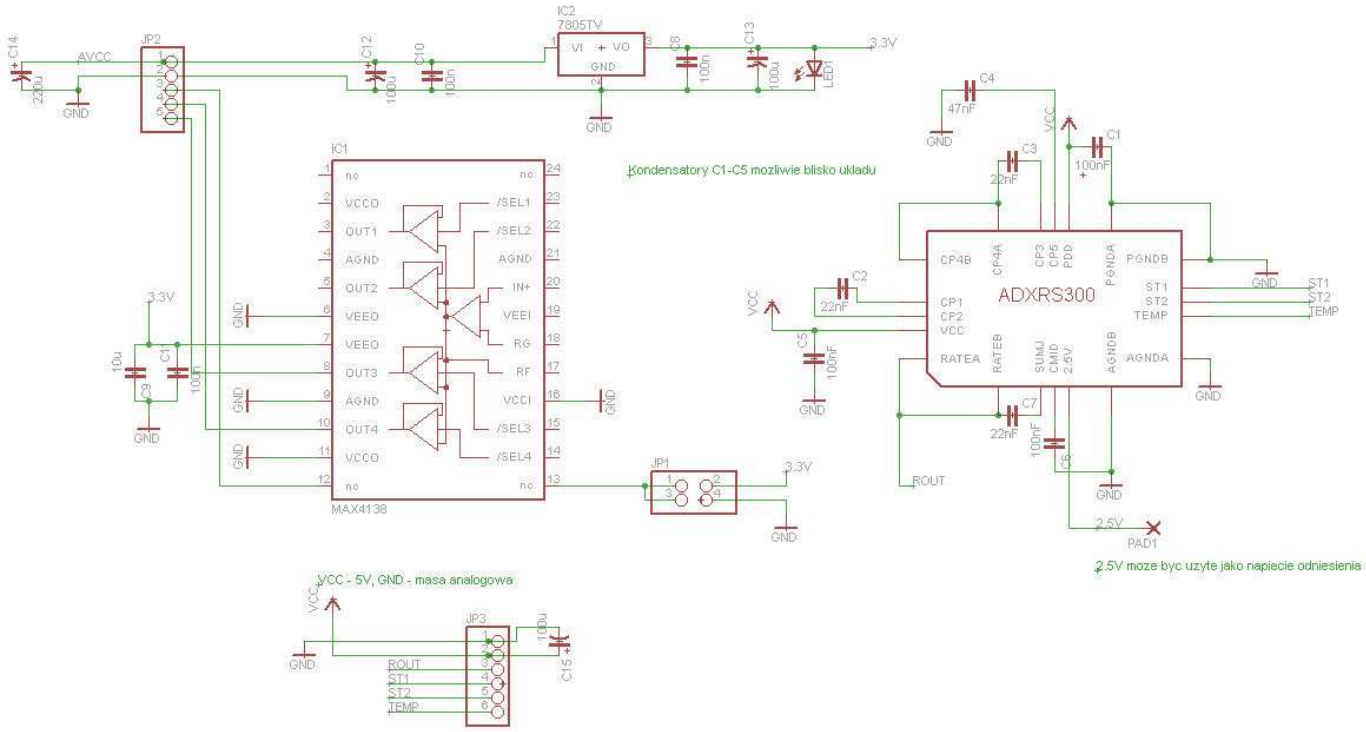
Schemat 4 Moduł mostka-H – schemat



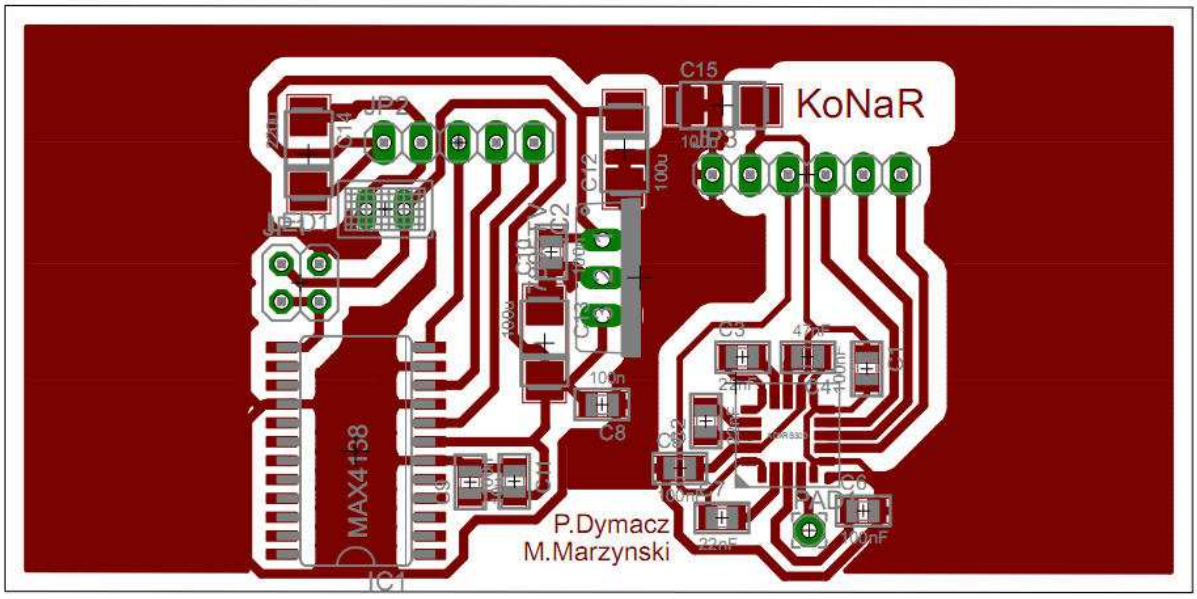
Schemat 5 Moduł mostka-H - płytka

Moduł czujników

Moduł zawierający czujniki do pomiaru odchylenia kątowego robota : żyroskop analogowy ADXRS300 oraz akcelerometr analogowy LIS3L02AS4. Element użytym do stworzenia schematu połączeń akcelerometru ma jedynie identyczną obudowę SO-24 ze względu na brak dostępu do oryginalnej biblioteki tego elementu.



Schemat 6 Moduł czujników - schemat



Schemat 7 Moduł czujników - płytka

Podsumowanie

Słabe strony algorytmu uczącego się

Schemat 8 Elektronika sterująca - schemat

Pomimo dość szybko uzyskanych efektów, czas uczenia od kilku do kilkunastu godzin, symulowane zachowanie robota znacznie odbiega jakościowo od efektów uzyskiwanych za pomocą sterowania regulatorami P, PI i PID.

Sporym problemem okazała się kwestia dyskretyzacji możliwej do wykonania akcji. Cyfrowe implementacje regulatorów zapewniają dokładne wyliczenie odpowiedzi systemu (układ + regulator) na każdy określony stan robota. Algorytm uczenia ze wzmocnieniem niestety musi mieć jednoznacznie określone przedziały wartości określające konkretne akcje które może wykonać. Tu odwołujemy się konkretnie do ilości możliwości podania wypełnienia na silniki robota balansującego. Pierwsze testy algorytmu były przeprowadzane dla macierzy złożonej z 91 kolumn i 21 wierszy (kolumny określają możliwy stan od -45 do 45 stopni wychylenia z pozycji pionowej, wiersze określają 21 możliwości podania wypełnienia na silnik w przedziale od -100% do 100%) co zapewnia podawanie wypełnienia z podziałką co 10% wartości maksymalnej. Jest to fatalny wynik, ponieważ w stanach bliskich równowagi siły potrzebne do regulacji są o wiele mniejsze. Problem ten może być łatwo rozwiązany poprzez zwiększenie macierzy, a dokładniej ilości możliwych akcji, oraz poprzez zmianę funkcji przeliczającej indeksy macierzy na zależności procentowe wypełnienia – na odpowiednią funkcję wykładniczą. Jednak zwiększenie macierzy stanów i akcji prowadzi do ogromnego wzrostu potrzebnej pamięci do przechowywania generowanych danych oraz przede wszystkim wzrasta czas potrzebny do nauczenia się balansowania. Przy założeniu otrzymywania wyników pomiarów czujników co 50ms oraz rozszerzając możliwość osiągania przez robota stanów wychylenia od pionu do prawie +/- 90 stopni czas potrzebny na naukę rzeczywistego robota staje się niewygodnie długi.

Kolejna kwestią wartą poruszenia jest sposób uczenia się. Algorytm powinien zagwarantować przejście robota przez wszystkie możliwe stany i akcje (zapewnia to miedzy innymi start algorytmu z macierzą wypełnioną zerami) w celu zidentyfikowania wzmocnienia w każdej komórce. Przeprowadzenie takiego uczenia w symulacji jest wygodne, robot nie ulega zniszczeniu jeśli straci równowagę. W rzeczywistości przewrócenie robota uzależnia go od pomocy człowieka. Rozwiązaniem może być odrębny system podnoszenia robota do pozycji pionowej lub prawie pionowej w której robot sam potrafił by powrócić do pozycji pionowej odpowiednio przyspieszając (kręcąc kołami). W odniesieniu do wielu innych obiektów regulacji w świecie rzeczywistym jest trudne lub wręcz niemożliwe przeprowadzenie tak dużej ilości prób potrzebnych do nauczenia robota (obiekt regulacji musiał by wielokrotnie przechodzić przez stany w których np. układ mechaniczny jest przeciążony i może ulec zniszczeniu, albo temperatura pewnych elementów jest krytyczna). Częściowym rozwiązaniem tego problemu mogło by być wstępne wypełnienie macierzy stanów i akcji na podstawie regulacji.

Mocne strony algorytmu uczącego się

Najmocniejszą stroną algorytmu jest przede wszystkim fakt, że nie mając żadnej informacji o parametrach fizycznych robota oraz środowiska, w którym się znajduje (np. siła grawitacji, opory ośrodka) algorytm startując z macierzą zainicjalizowaną zerami jest w stanie nauczyć się regulować odchylenie od pionu utrzymując je jak najbliżej zera. W przypadku zwykłych regulatorów P, PI i PID jest to nieosiągalne – wymagają one procesu wyznaczenia odpowiednich nastaw, które i tak wymagają strojenia na rzeczywistym obiekcie, gdyż ten odbiega od założonego modelu matematycznego. Mało tego, gdybyśmy obiekt z dostrojonym już regulatorem przenieśli do innego środowiska, w którym panują inne warunki fizyczne, niż te dla których strojony był regulator, przestałby on spełniać swoją funkcję i układ uległby destabilizacji. Natomiast dzięki algorytmowi Q-learning mamy zapewnioną adaptację agenta do aktualnie panujących warunków. Owszem, adaptacja ta wymaga pewnego nakładu czasu, ale odbywa się zupełnie autonomicznie.

Słuszność stosowania uczenia się ze wzmocnieniem

Przedstawiony w powyższym sprawozdaniu problem – regulacja odchylenia od pionu robota balansującego przy pomocy algorytmu uczenia ze wzmocnieniem była bardzo ciężkim orzechem do zgryzienia, między innymi dlatego, że pracujemy w środowisku ciągłym i jego dyskretyzacja (tym bardziej z niewielką dokładnością) powoduje w połączeniu z niezależnymi od regulatora siłami i przyspieszeniami działającymi w sposób ciągły, duże problemy i niedokładności w tejże regulacji. Tym bardziej, że algorytm z założenia powinien być jak najprostszy i zajmować jak najmniej pamięci, gdyż jego przeznaczeniem jest zastosowanie w mikrokontrolerze. Jednakże, jak pokazały przeprowadzone symulacje, stosunkowo niewielkim kosztem zwiększania złożoności i objętości pamięci wykorzystywanej przez algorytm, da się osiągnąć całkiem przyzwoite wyniki, porównywalne z regulatorami ciągłymi.
Metodę uczenia ze wzmocnieniem stosuje się w robotyce i jest ona dynamicznie rozwijana. Uczenie się celowego zachowania na podstawie interakcji ze środowiskiem można zatem i użyć w regulacji.

Błędy w konstrukcji mechanicznej

Dużym problemem okazało się zamocowanie tulei (do których są przykręcane koła i orczyk serwomechanizmu) oraz kół zębatych do osi. Oś aluminiowa, pomimo wymiany z pustej w środku na litą, ulega ścieraniu i „wyrabianiu" mechanicznemu. Proste przykręcenie tych elementów za pomocą śrub przechodzących na wylot przez oś nie przyniosło zadowalających efektów. Po uruchomieniu robota i próbach nastawienia odpowiednich parametrów regulatorów mocowania dość szybko „wyrobiły się" tworząc nieakceptowalne luzy w układzie przeniesienia napędu.

Aktualne zaawansowanie prac nad projektem

Dotychczasowo w projekcie zostało zrealizowane :

- wykonanie zaplanowanej konstrukcji mechanicznej (rama robota, przełożenie napędu) oraz poprawienie wad konstrukcyjnych omówionych w ostatnim sprawozdaniu – wymiana osi na lite, wymiana kół na lżejsze.
- opracowanie algorytmu uczącego się, Qlearning oraz Reinforcement Learning wraz z wykonaniem symulacji komputerowej, przeprowadzenie badań nad algorytmem, porównanie z regulatorami P,PI,PID.
- Wykonanie części elektronicznych : płytka z elektroniką sterującą, moduł sterownika silnika oraz moduł sensorów.
- Program – komunikacja RS232, filtr Kalmana, implementacja regulatorów P,PI,PID, obsłużenie wszystkich peryferii : akcelerometr, wyświetlacz, klawiatura, dwa ultradźwiękowe czujniki odległości, sterowanie silnikiem, wykorzystywanie pamięci zewnętrznej eeprom, zabezpieczenie baterii przed rozładowaniem (pomiar napięcia). Chcielibyśmy również zaznaczyć, że pełne oprogramowanie (symulacja komputerowa PhysX oraz program sterujący robotem) zostanie udostępnione po zakończeniu prac nad projektem.

Aktualnie trwają prace nad wdrożeniem żyroskopu ADXRS300 (dedykowana płytka została już wykonana), oraz w niedalekiej przyszłości planowane jest wyposażenie robota w moduł bluetooth pozwalający na bezprzewodową komunikację z komputerem. Również wielce pomocna okazałaby się dodatkowa bateria (ta, która jest zainstalowana obecnie jest bardzo mała pojemnościowo i zapewnia zastraszająco krótką pracę, posłużyłaby wyłącznie do zasilania elektroniki, a grupa zakupiłaby większą celem zasilania silnika. Moduły elektroniki i sterownika silników są już przystosowane do takiego rozwiązania).

Bibliografia

- www.wikipedia.org *(stan z dnia 20.05.2010).*
- http://www.elektroda.pl/ (stan z dnia 7.07.2010).
- http://developer.nvidia.com/object/physx.html *(stan z dnia 20.05.2010).*
- http://people.revoledu.com/kardi/index.htm *(stan z dnia 20.05.2010).*
- Paweł Cichosz : *Systemy uczące się* Wydawnictwa Naukowo-Techniczne Warszawa, 2000 ISBN: 83-204-2544-1.
- Tom M. Mitchell : Machine Learning ISBN: 0070428077.
- Jan Kędzierski, Karol Sydor : Sonar ultradźwiękowy. Raport z prac nad czujnikiem w Kole Naukowym Robotyków „KoNaR".
- Jan Kędzierski, Filtr Kalmana – zastosowania w prostych układach sensorycznych. Koło Naukowe Robotyków „KoNaR".
- Datasheet ATmega32.
- Datasheet Atmel 24c64 eeprom.
- Datasheet VNH2SP30.
- Datasheet ADXRS300
- Datasheet LIS3L02AS4

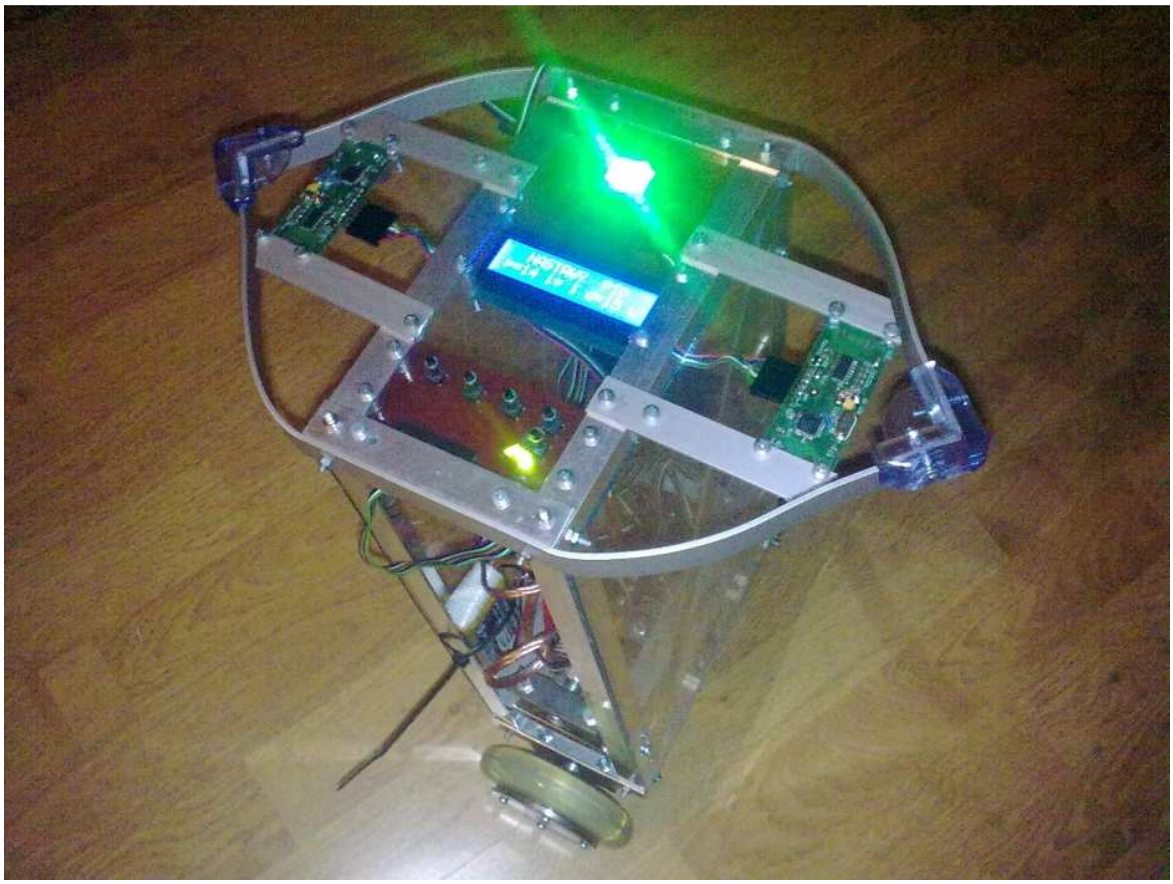
Podziękowania

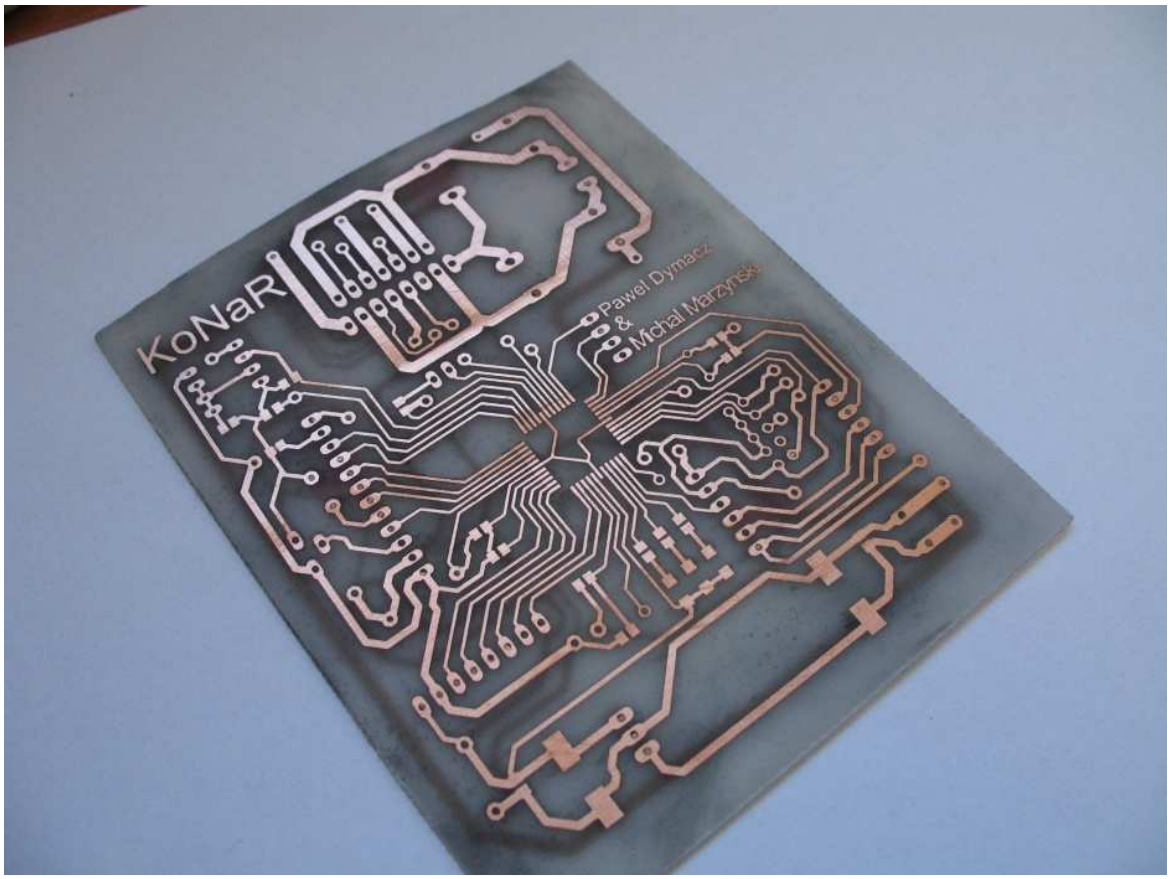
Wyrazy wdzięczności oraz podziękowania dla doktora Andrzeja Kozika i doktora Radosława Rudka za pomoc w implementacji oraz cenne wskazówki dotyczące algorytmu Q-learning oraz środowiska nVidia PhysX.

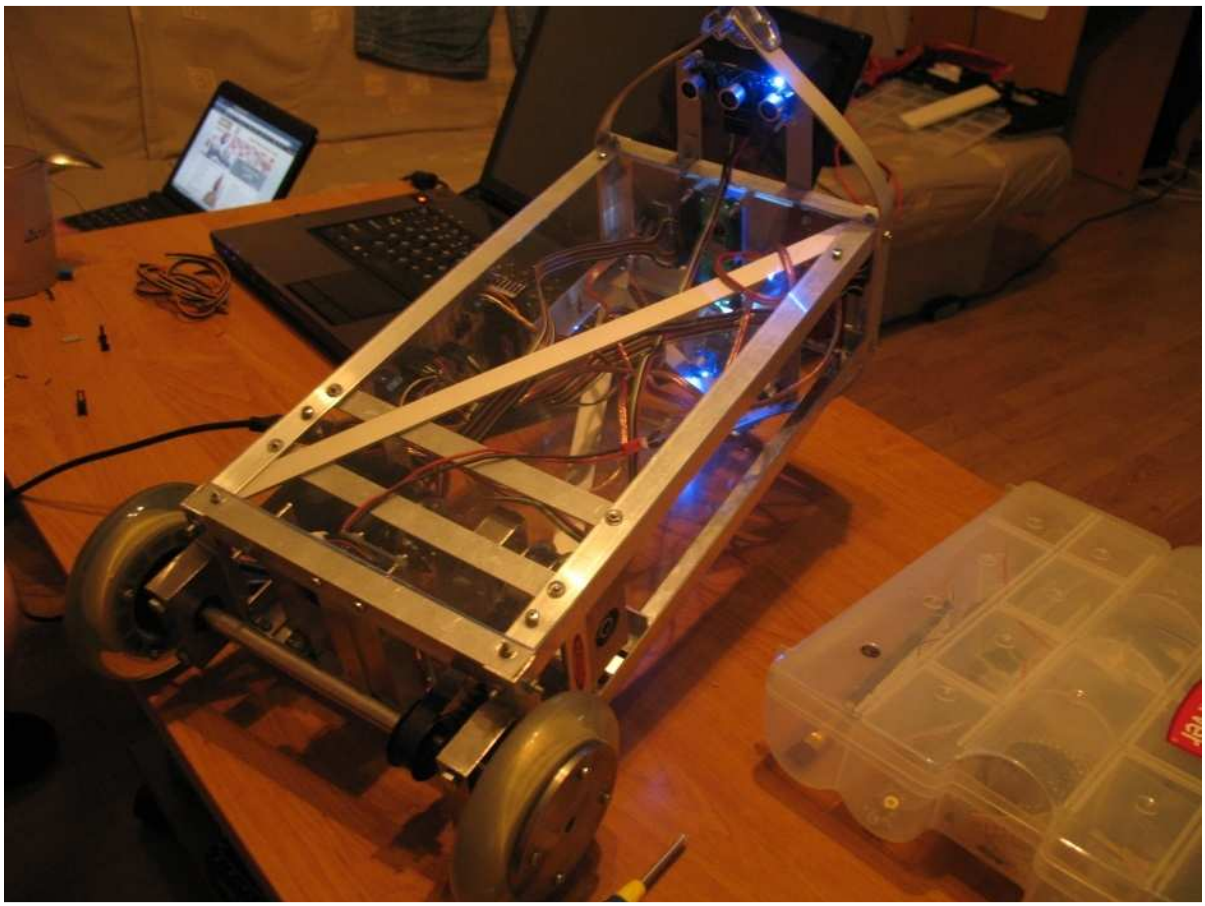
Dziękujemy także doktorowi Markowi Wnukowi za wyposażenie naszej grupy projektowej w żyroskop ADXRS300.

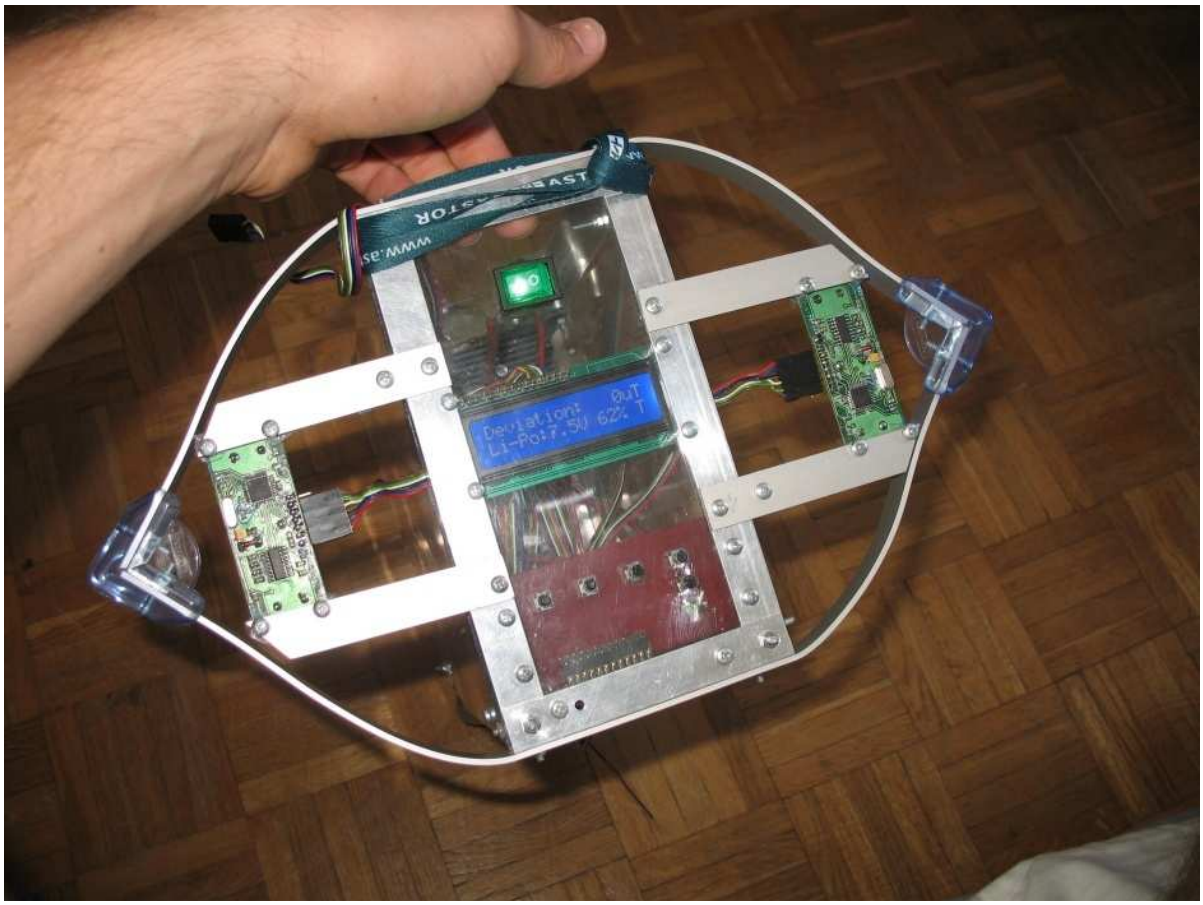
Dziękujemy również osobom z Koła Naukowego Robotyków „KoNaR", przede wszystkim Jankowi Kędzierskiemu za doradztwo, cierpliwość i pomoc w rozwiązywaniu napotkanych przez nas po drodze problemów; osobie która wolała pozostać anonimowa, a bez której stronie mechanicznej robota daleko byłoby do działania bez zarzutu; zarządowi, szczególnie Pawłowi Kaczmarkowi, za pracę włożoną w sprawowanie opieki nad finansowaniem projektu; oraz wszystkim niewymienionym tu osobom, które w mniejszym lub większym stopniu przyczyniły się do powstania tego projektu.

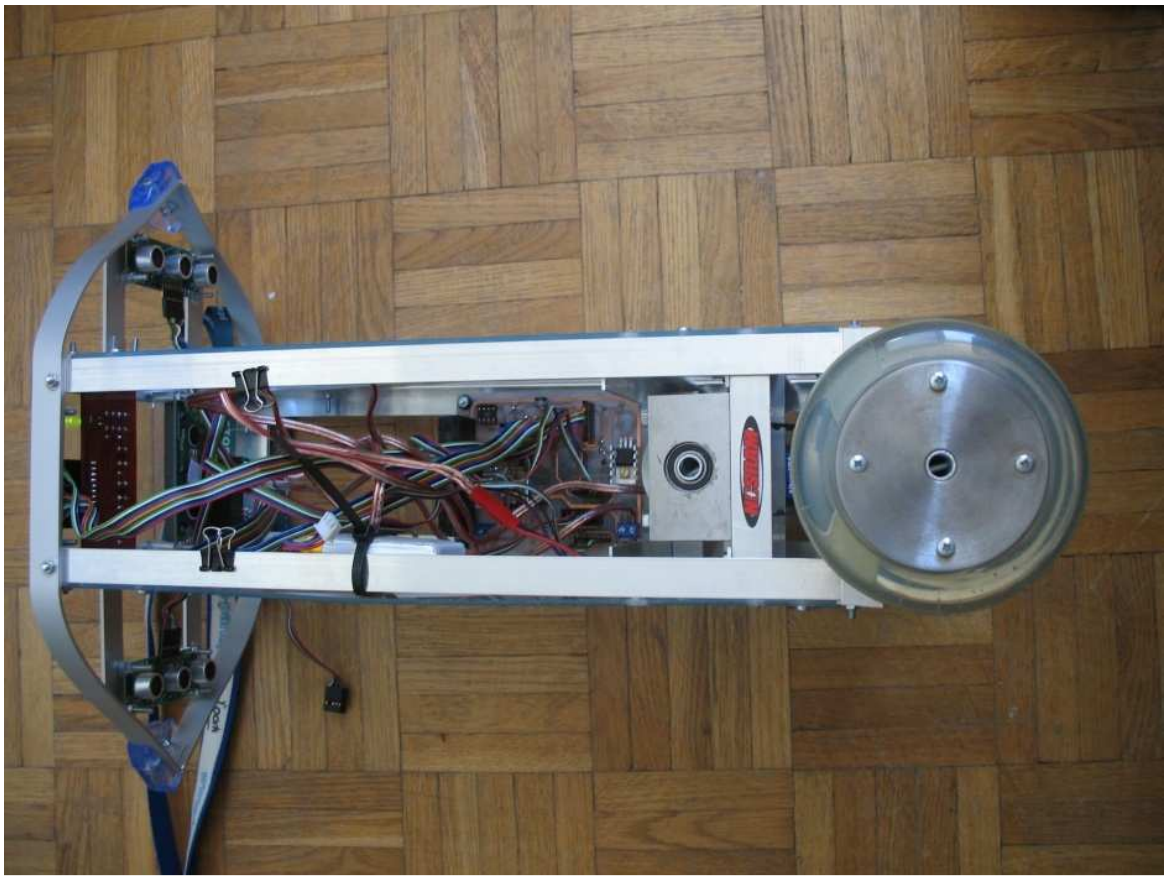
Galeria V1

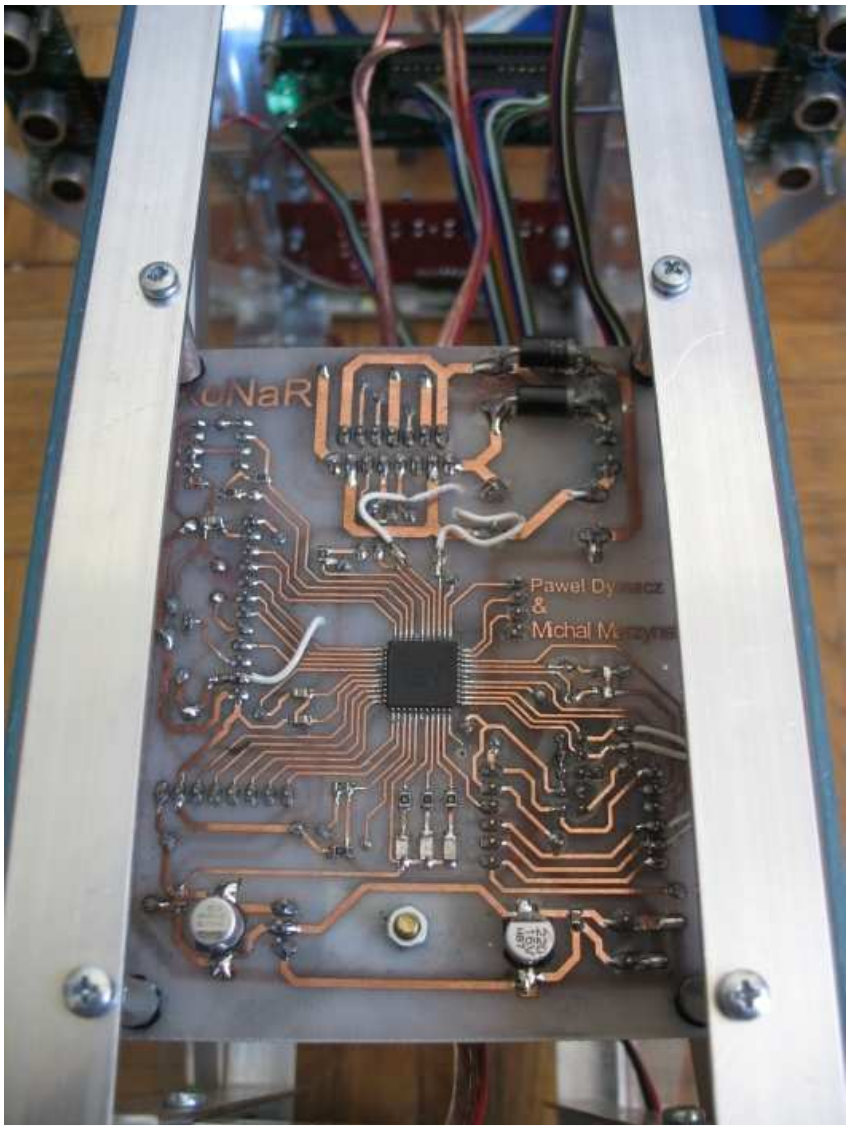



KoNaR
Paweł Dymacz
&
Michał Marzyński

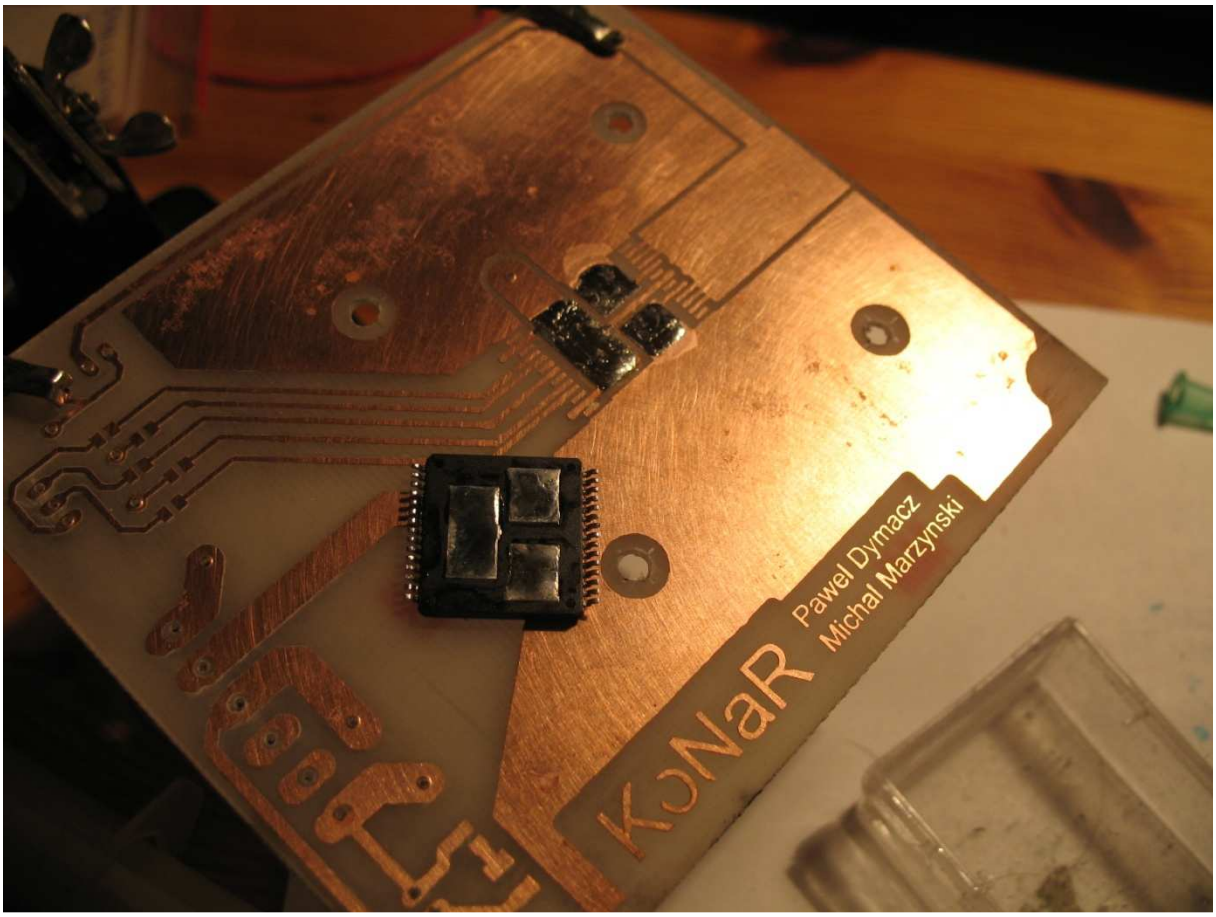



Deviation: 0uT
Li-Pol:7.5V 62% T




Paweł Dymacz
&
Michał Marzyński

Galeria V2


KoNaR
Pawel Dymacz
Michal Marzynski