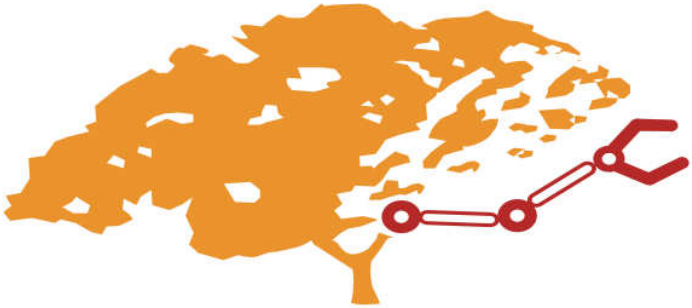


KoNaR

KOŁO NAUKOWE ROBOTYKÓW

Robot microsumo
S.M.A.R.T.

Raport końcowy

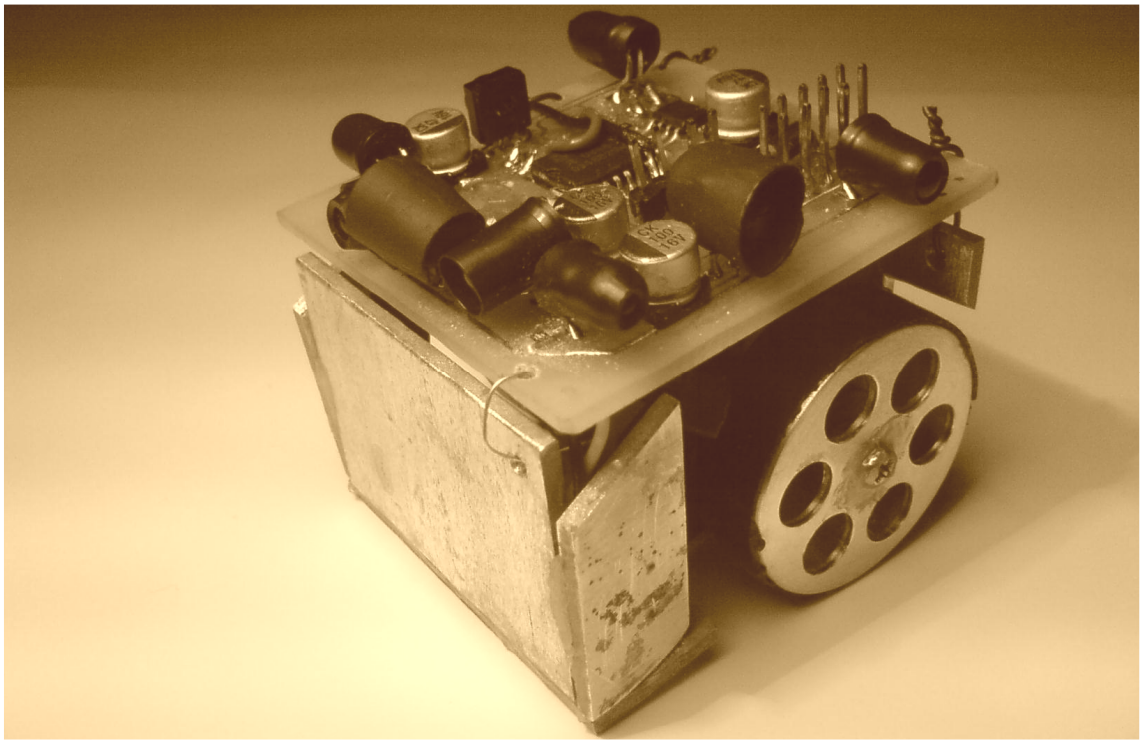
Michał Dziergwa , Maciej Gawron, Mariusz Orda
Koło Naukowe Robotyków „KoNaR”
Wrocław 2010

Spis treści

1. Wstęp

Idea budowy robota S.MA.R.T. zrodziła się na początku roku 2010, na spotkaniu poświęconym omówieniu zrealizowanych projektów w kole naukowym „KoNaR” w roku poprzednim. Słowami, które zrodziły ten pomysł było zdanie kolegi Roberta Budzińskiego *„znowu będziecie robić minisumo, może zróbcie micro”*. Ze swoim skromnym doświadczeniem grupa zabrała się do prac prowadzonych z różną intensywnością, która wzrastała przed kluczowymi zawodami – Robot Challenge 2010 we Wiedniu (na które robot nie został ukończony) oraz Robotic Arena 2010 we Wrocławiu (przed którymi przeszedł całkowitą modernizację i gdzie został zaprezentowany).

Niniejszy raport prezentuje drugiego stworzonego w kole naukowym „KoNaR” robota klasy microsumo. Naszym celem przy jego tworzeniu było zaprezentowanie pewnych wykorzystanych przez nas rozwiązań, pokazanie korzyści wynikających z ich stosowania oraz wad, które zostały zauważone podczas testowania. Chcielibyśmy także zachęcić młodszych członków koła, tak, jak kiedyś zachęcił nas starszy kolega, do budowania w dobie nanotechnologii rzeczy jeszcze mniejszych – sądzimy, że robot 5x5x5 cm o masie nie przekraczającej 100 gram na początek wystarczy.



Rys 1 – Robot mobilny S.M.A.R.T.

2. Część mechaniczna

2.1 Założenia projektowe

Podstawowe ograniczenia, wedle których rozgrywane są zawody robotów microsumo zostały wymienione już we wstępie. Dość dużym problemem jest zmieszczenie wszystkich, relatywnie sporych części niezbędnych w robocie mobilnym – szczególnie układów zasilającego oraz napędowego. Niezbędne było więc, aby oba te elementy były jak najmniejsze, ale jednocześnie zapewniały pożądane parametry.

2.2 Układ napędowy

Opracowano dwie koncepcje napędu robota. Pierwsza koncepcja oparta była o silniki Graupner Speed 265 6V. Pierwsze plany zakładały wykonanie przekładni redukującej ilość obrotów silników, zrezygnowano za nich jednak ze względu na brak odpowiedniej ilości miejsca – wstępny projekt w programie AutoCad pokazał, że niemożliwym będzie wówczas stworzenie jakiegokolwiek rodzaju pługa. W takim przypadku zasilanie musiałoby znaleźć się powyżej układu napędowego, co podwyższałoby punkt ciężkości robota, ułatwiając możliwość jego wywrócenia. Ponadto, nie udało się wyszukać na polskim rynku odpowiednio małych silników (wymiary idealne długości - do 20 mm przy koncepcji nasadzania kół na wał silników).

Zdecydowano się wykorzystać silniki z przekładniami z miniserwa Tower Pro SG-91R, mającymi połowę trybów metalowych. Wymiary serw to 23x12.2x29 mm, co nie zapewnia mieszczenia się w zakładanych wymiarach. Usunięte zostały połowy serw wraz z całą elektroniką sterującą (na kształt litery L) i połączono je klejem cyjano-akrylowym tak, aby oba nałożone na nie koła były współosiowe.

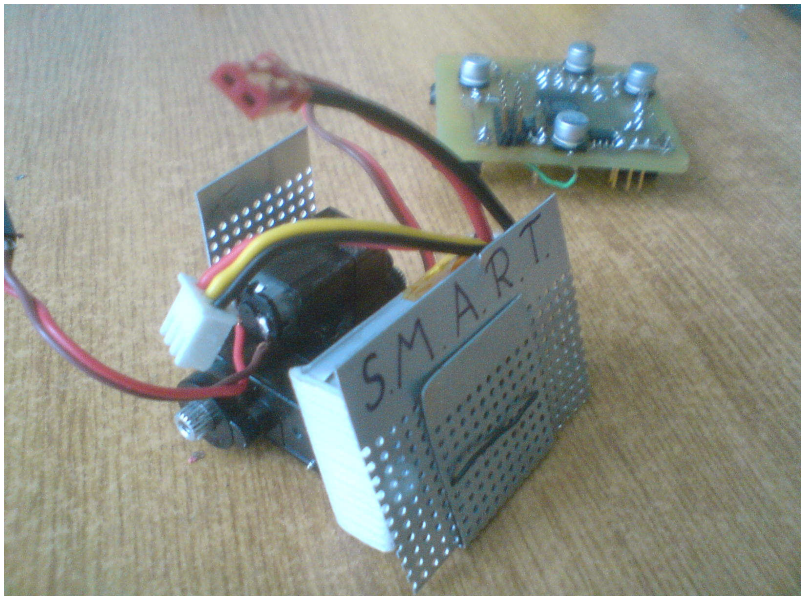
Serwa te wybrano ze względu na korzystne parametry:

* Waga : 11g
* Moment : 2.0 kg (4.8V)
* Prędkość : 0.1 sek./60° (4.8v)

Koła zostały wytoczone z aluminium, mają średnicę 30 mm; przy łączeniu z serwem mają szerokość 2 mm, przy warstwie zewnętrznej mającej kontakt z podłożem – 10 mm. Taka konstrukcja wraz z wywierconymi sześcioma otworami zmniejszającymi masę koła zapewnia odpowiednio mały ciężar – 4 g każde. Na koła możliwe jest założenie odpowiedniego ogumienia, jednak jego maksymalna grubość jest ograniczona do 1,5 mm.

2.2 Obudowa

Robot doczekał się trzech wersji obudów. Pierwsza, wykonana z cieniutkiej blaszki, była obudową prowizoryczną, służącą jedynie do pierwszych testów robota, zdjęcie zamieszczamy jedynie z powodów historycznych.

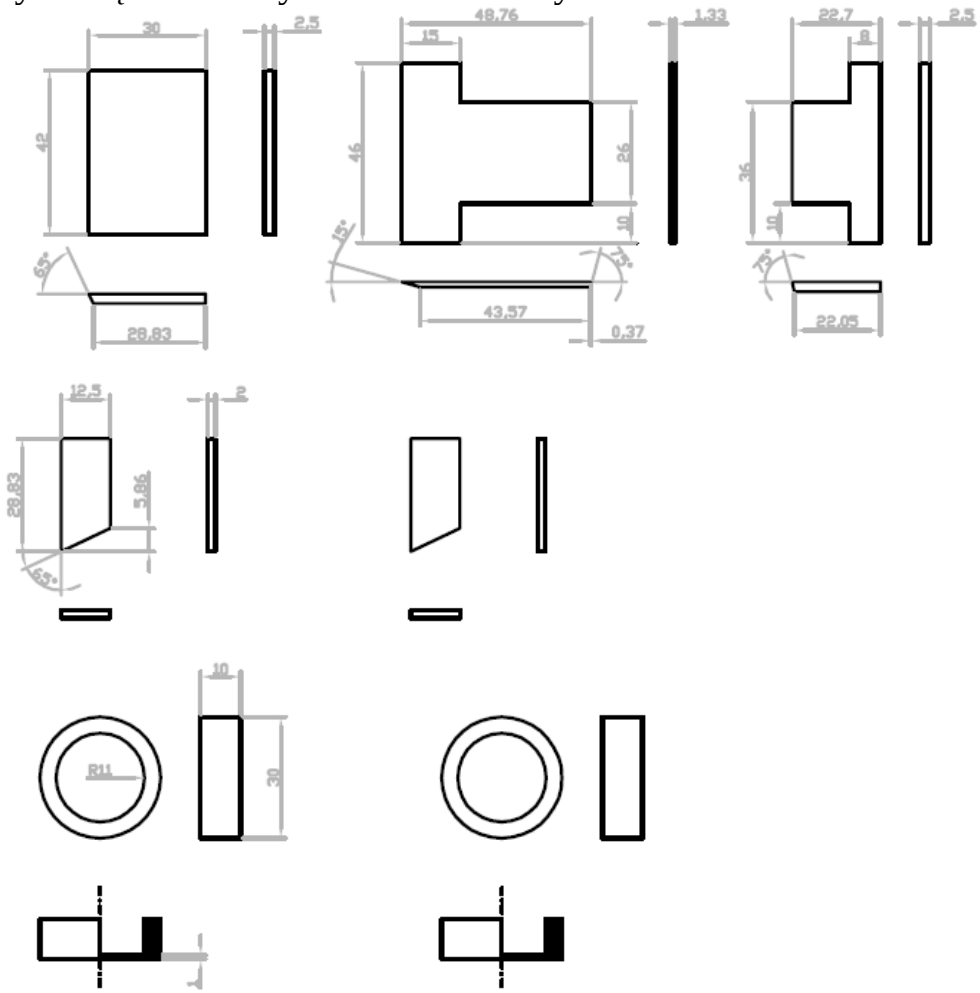


Rys 2 – Wygląd ogólny S.M.A.R.T. przed pierwszym złożeniem: osłona z lipolem oraz serwami, w oddali pierwsza wersja płytki

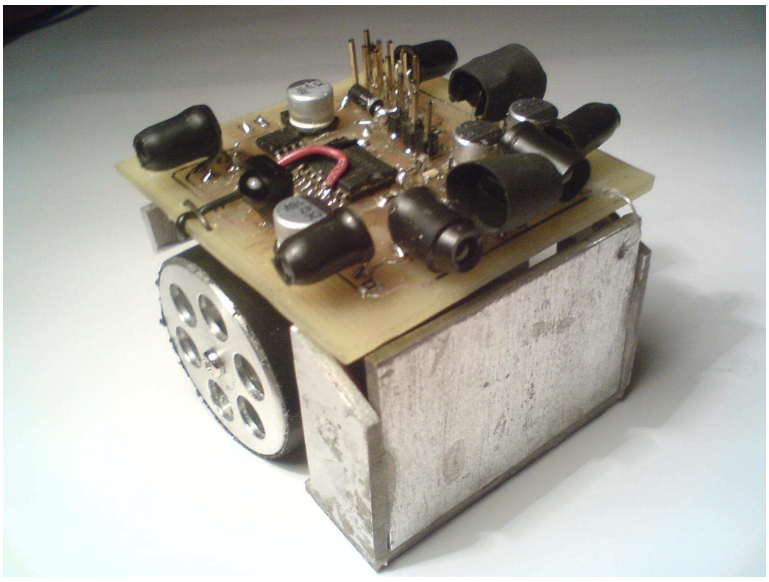
Druga wersja została wykonana z laminatu – poszczególne jej części zostały wycięte przez wiertarkę z tarczą do cięcia i połączone klejem z żywicą epoksydową. Dla lepszego efektu wizualnego pokryto ją brązowym lakierem w sprayu, jednak niestety po tej operacji wyglądała jeszcze gorzej.

Rys 3 – S.M.A.R.T. po złożeniu z drugą obudową

Całość spełniała zadania – była wytrzymała, dobrze zamocować można było na niej płytkę z elektroniką, jednak nie była zbyt reprezentacyjna. Dodatkowo, pierwszą wersję kół stanowiły tam zakrętki od wody mineralnej. Przed zawodami postanowiono o stworzeniu obudowy z prawdziwego zdarzenia – z połączonych części toczonych oraz frezowanych w aluminium.



Rys 4 – Rysunek techniczny obudowy wykonany w programie AutoCad…



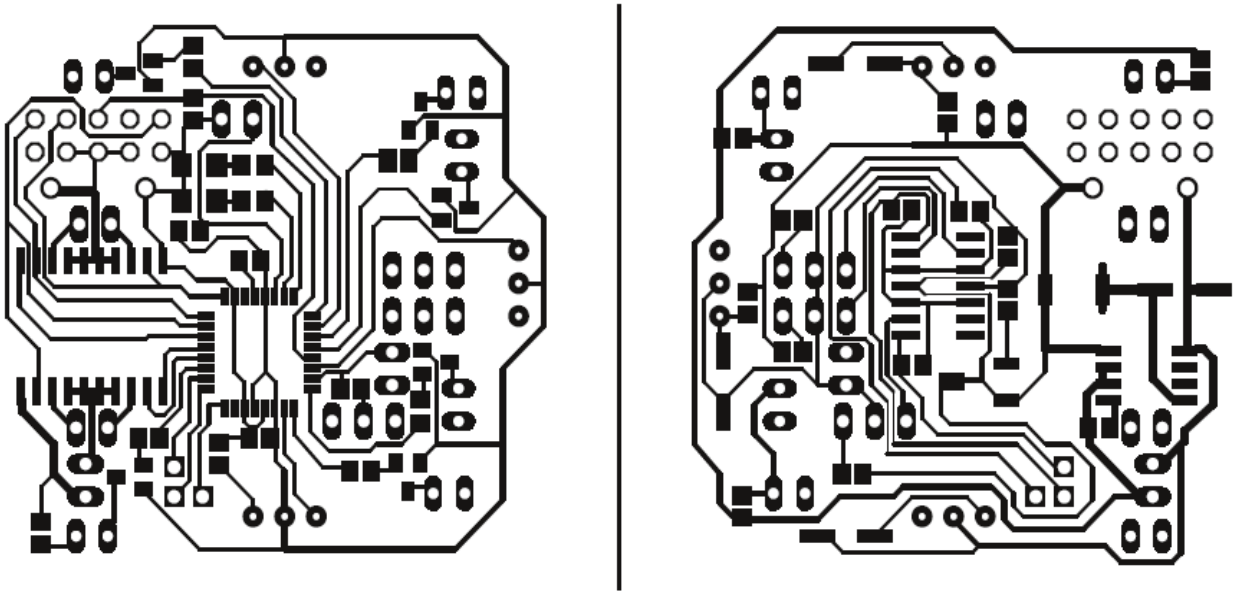
Rys 5 –… i widok robota obecnie, po złączeniu nowej obudowy.

3. Elektronika robota

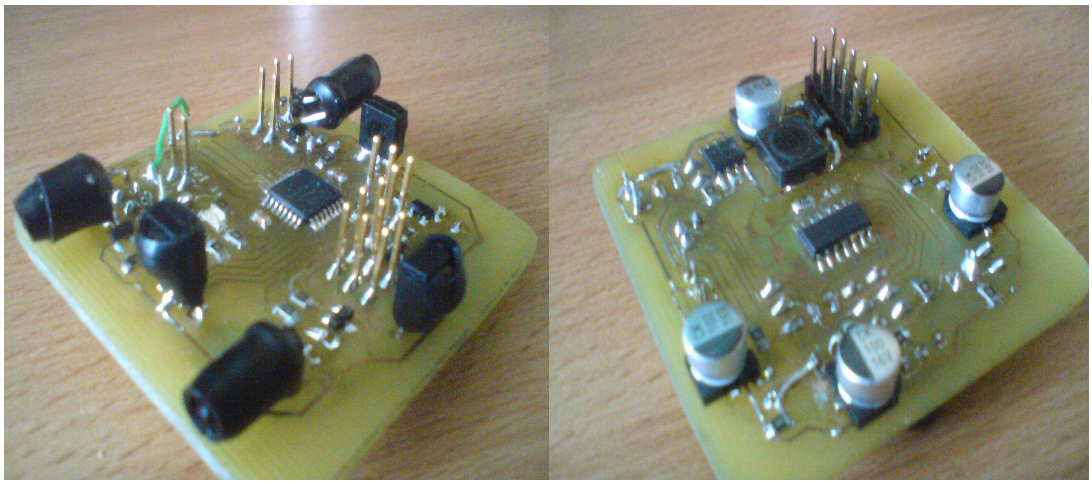
3.1 Płytka PCB

Robot microsumo ma za zadanie wykryć i wypchnąć przeciwnika z ringu. O ile o drugim z tych zadań w ogólności decyduje kwestia napędu, odpowiedniego pługa, opon o pożądanej przyczepności i, wreszcie, lepszego algorytmu to warstwa elektroniczna ma za zadanie wykrywania przeciwnika (oraz białej linii za pomocą wbudowanych sensorów. Niezbędne są także: stabilizator 5V oraz mostek H służący do sterowania silników.

Z powodu ograniczonego miejsca zdecydowano się na stworzenie jednej płytki, dwustronnej w technice montażu powierzchniowego, którą następnie umieszczono w górnej części robota.



Rys. 6 – Projekt płytki PCB…



Rys. 7 - … oraz jej wykonanie

Obrazki prezentują drugą wersję płytki. Wykonywane metodą żelazkową płytek do techniki montażu powierzchniowego charakteryzują się stosunkowo małą odpornością na ponowne nagrzewanie padów –zdarza się, że przy lekko za mocnym grzaniu lutownicą odpadają one, tak jak się stało przy pierwszej wykonanej płytce. Drugą zmianę spowodowała rezygnacja z mostków L293DD na rzecz A3953SLB.

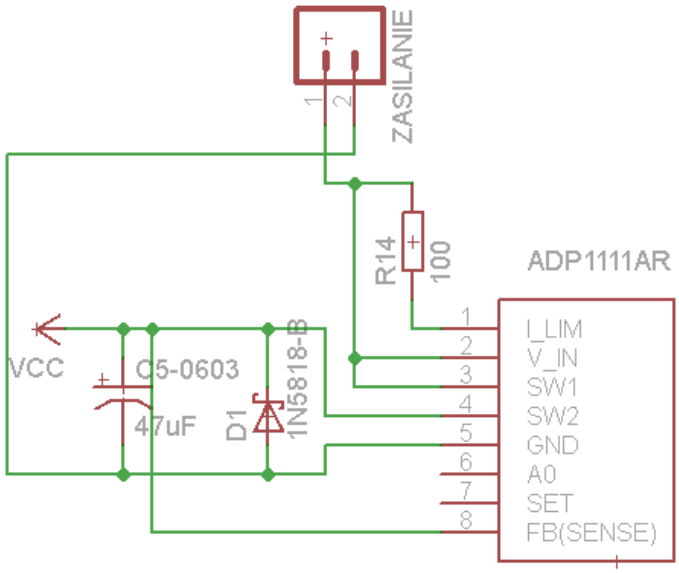
3.2 Układ zasilający

Robot zasilany jest baterią litowo-polimerową MAX FORCE Li-Pol 7,4V 400 mAh 10C firmy 3E . O wyborze zadecydowały jej wymiary – była to jedyna bateria, która odpowiadała naszej koncepcji rozmieszczenia elementów w robocie - mieściła się w zadanych rozmiarach (38 x 25 x 14,5 mm). Dodatkową zaletą była jej bardzo niska waga - 22,7 gram, oraz korzystna cena.

Obecnie nie zauważyliśmy na rynku lepszego rozwiązania w kwestii zasilania robota microsumo, choć radzilibyśmy przyszłym konstruktorom zakup dwóch baterii tego typu – produkty firmy 3E nie są tak niezawodne jak Kokam, zaobserwowano także niepokojące zjawisko jak samoczynne rozładowanie się lipola do 5 V (nie mniej, naładowany działa w robocie do dziś) .

Rys. 8 – Użyta bateria litowo – polimerowa MAX FORCE

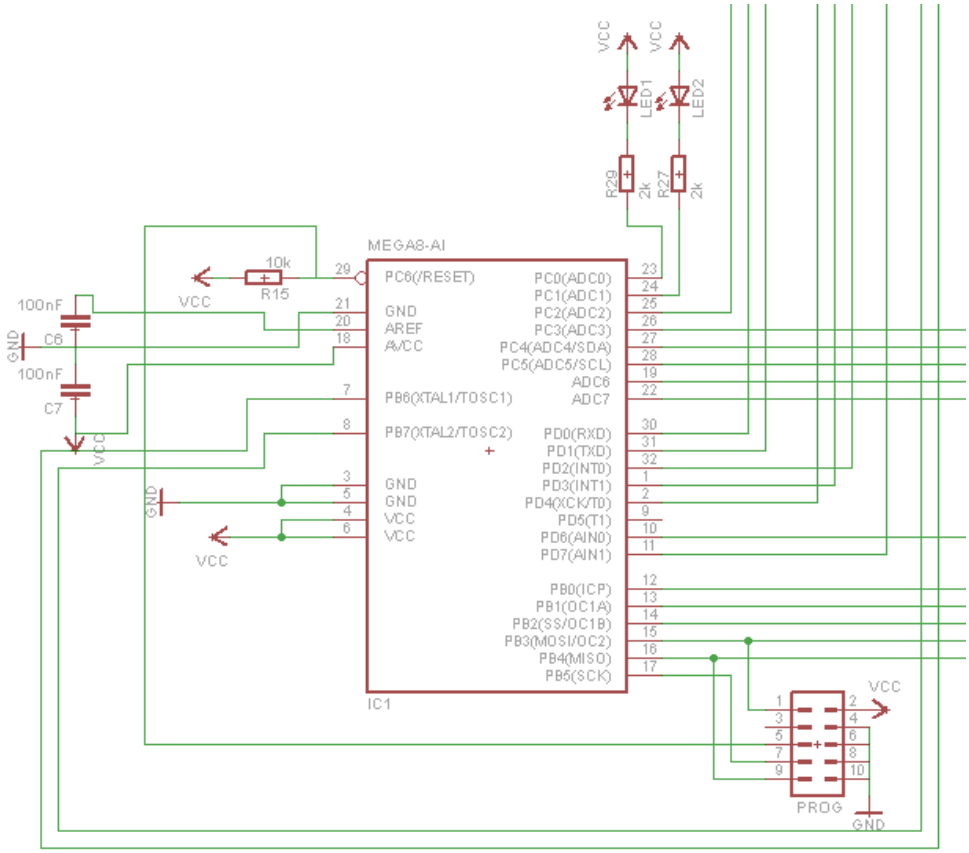
Aby uzyskać napięcie 5 V, niezbędne do prawidłowego działania mikroprocesora oraz czujników zastosowano układ ADP1111 w obudowie SO-8, ustawiając go w trybie Step-Down. Pierwotnie układ zawierał cewkę pomiędzy pinem 4 a napięciem wyjściowym, którą usunięto ze względu na brak miejsca – doświadczalnie stwierdzono jej opcjonalność przy zasilaniu napięciem 6-8 V.



Rys. 9– Schemat podłączenia przetwornicy ADP1111

3.3 Mikroprocesor

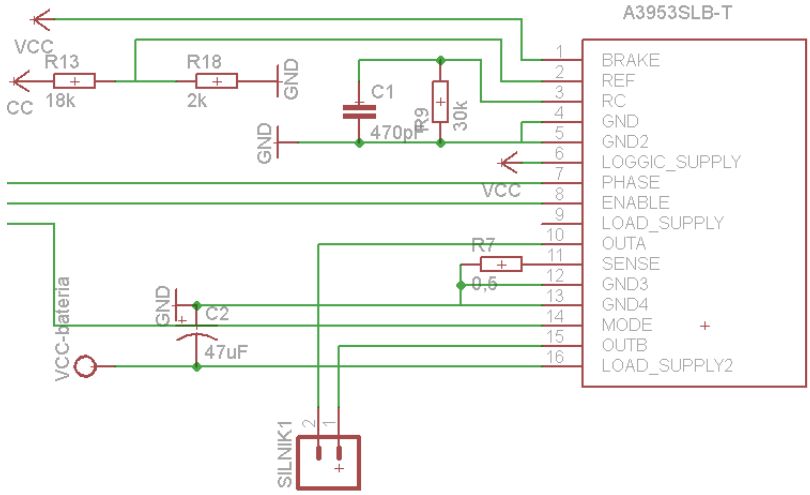
W robocie wykorzystano mikroprocesor Atmega8 w obudowie TQFP. Posiada on 8 wejść 8-bitowego przetwornika analog.-cyfr. 3 z tych linii wykorzystano do obsługi czujników białej linii . Robot działa na wewnętrznym oscylatorze 8 MHz.



Rys. 10– Schemat podłączenia mikroprocesora

3.4 Mostek H

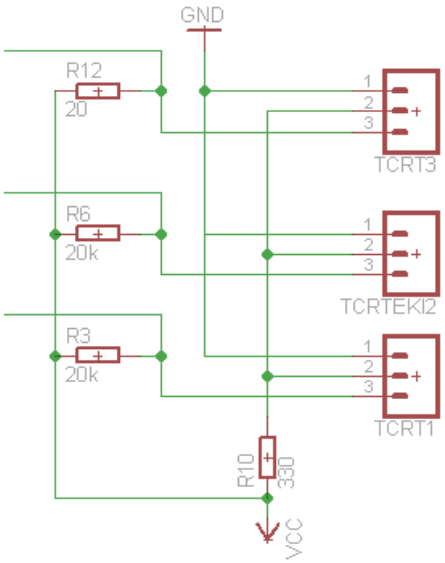
Do sterowania robota użyte zostały mostki h A3953SLBT firmy Allegro Microsystems. Są to układy pozwalające na sterowanie pojedynczym silnikiem. Wybrane zostały ze względu na ich możliwości (różnorodność trybów działania, lepsze statystyki niż w przypadku układów obsługujących 2 silniki, np L298, L293, niewielka ilość dodatkowych elementów). Kierowaliśmy się również chęcią przetestowania rozwiązania innego niż te stosowane w innych konstrukcjach. Mostki sprawują się bez zarzutów, charakteryzują się niskim spadkiem napięcia i jak dotąd nie zauważyliśmy, żeby nadmiernie się nagrzewały. Fakt, że dla każdego silnika potrzebny jest osobny mostek jest też ich największą wadą - dwa, dosyć dużych rozmiarów układy na płytce przeznaczonej dla małego robota znacznie utrudniają projektowanie elektroniki.



Rys 11– Schemat podłączenia mostka H A3953

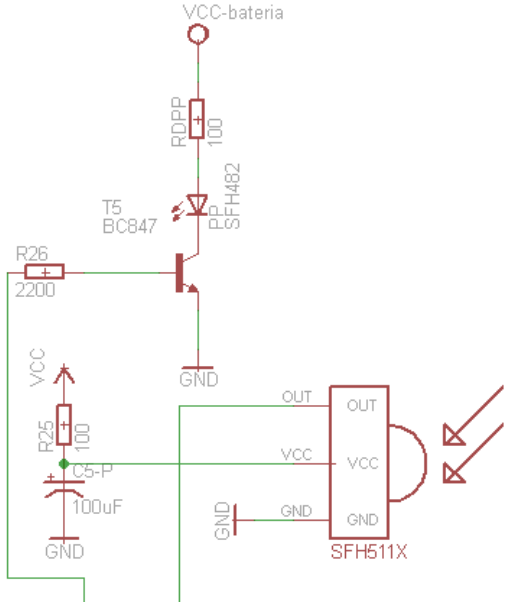
3.5 Czujniki

Do wykrywania białej linii na dohyo użyto trzech czujników TCRT5000.



Rys 12– Schemat przyłączenia czujników TCRT5000

Ważną zmianą w stosunku do założeń projektowych była rezygnacja z dalmierzy optycznych na rzecz diod podczerwonych oraz ich odbiorników. Spowodowane było to koniecznością zminimalizowania wymiarów sensoryki (tak, by mogły mieścić się na płytce PCB) oraz wieloma ich zaletami - niską ceną w stosunku do sharpów, dużym zasięgiem (do 70cm), stosunkowo małą wrażliwość na warunki zewnętrzne oraz niewielkim czasem pomiaru.



Rys 13–Nadawcza dioda podczerwona wraz z odbiornikiem

AT Mega8 generuje sygnał o f=36kHz (przerwania) wysyłany przez mocne diody IR (po obu stronach czujnika-pozwalają dokładniej określić pozycje przeciwnika: lewo, prawo, środek), który po odbiciu od obiektu przechwytywany jest przez czujnik podczerwieni SFH (wykorzystywany w sprzęcie RTV). Zastosowanie tego czujnika pozwoliło zminimalizować wymiary sensoryki (wszystkie elementy mieszczą się na płytce PCB).

Realny zasięg czujnika to 40-70cm – możliwe jest ograniczenia przez zmniejszenie prądu diod (np. zmiana rezystora na PCB), przysłonięcie czujnika/diod, zmniejszenie czasu pomiaru lub częstotliwości świecenia diod(standardowo 36 kHz). Charakteryzuje go niewielka wrażliwość na warunki zewnętrzne oraz niewielki czas pomiaru (50-100us przy ciągłym zasilaniu SFH lub około 5ms przy włączanym jedynie na czas pomiaru). Możliwym minusem są zakłócenia spowodowane odbiciem sygnału od większych płaskich powierzchni oraz pomyłkowe odbieranie sygnału wyemitowanego przez piloty RTV. Obszar wykrywania obiektu zależy od kąta granicznego diod (niestety większość diod jest za słaba i zostają 30 stopniowe). Można jednak zastosować przesłony (np. rurki termokurczliwe), które spowodują bardziej punktowe wykrywanie przeciwnika. Każdy pomiar jest powtażany maksymalnie 7 razy- gdy przeciwnik zostanie wykryty w 3 pomiarach to funkcja szukająca zwraca 1. Należy uważać, aby po każdym pomiarze wyłączać diody.

4. Oprogramowanie S.M.A.R.T.

4.1 Automat stanów

Algorytm robota S.M.A.R.T jest oparty na automacie stanów składającym się jedynie z 4 stanów. Program, napisany w języku C, został zaczerpnięty z robota Macieja Gawrona I prawo Murphy'ego, a zainspirowany kodem robota Shine. Do każdego stanu jest przyporządkowanych kilka sekwencji - kilka ruchów zgrupowanych i zapisanych w jednej tablicy. Każdy stan może wykorzystywać sekwencje innego stanu, jednak w celu ułatwienia późniejszej konfiguracji robota zostały one nazwane – przykładowo Atakuj3L3F jest sekwencją atakująca, która skręca w miejscu w lewo pod kątem pi/2, a następnie jedzie prosto-wszystko przy dużej prędkości).

Pojedynczy ruch to rozkaz typu: lewy silnik ma się poruszać z prędkością X, prawy z Y przez czas Z. W kodzie programu wygląda to następująco:

int8_t AtakujLL3[2][3] = {{PL1, PP4, 2}, {PL4, PP4, 10}};

Jeden cykl programu (sprawdzenie wszystkich sensorów i podjęcie decyzji) trwa mniej niż 2 ms (w przypadku sensorów firmy Sharp należałoby ten czas sztucznie przedłużyć do okresu pomiaru sensora).

S.M.A.R.T. posiada następujące stany:

- **UCIEKAJ** - stan posiada najwyższy priorytet - zostaje włączony w momencie, gdy jeden z czujników wykryje białą linie. Sekwencja ruchów robota zależy od tego, czy widzi przeciwnika oraz czy wcześniej ten stan już wystąpił (jeżeli tak to na jakim sensorze – dzięki temu w teorii robot nie powinien skręcać w złą stronę jeżeli jeden z sensorów już wypadł po za dohya (zwraca czarny kolor) i drugi sensor wykrył białą linię co bez podjęcia środków zapobiegawczych mogłoby spowodować samoistne wyjechanie po za pole walki). W przypadku, jeżeli przeciwnik jest dobrze widoczny przez przednie sensory stan ten jest pomijany (jest to opcja możliwa do wyłączenia np. w pojedynkach deathmatch w których powodowałaby ona wypadania przy pierwszym wygranym pojedynku 1 vs. 1),

- **ATAKUJ** - zazwyczaj stan wywoływany od razu po wykryciu przeciwnika. Jego zadaniem jest wypchnięcie przeciwnika za białą linię lub zbliżenie się do niego jak najszybciej. Ze względu na niewielką prędkość robota nie musieliśmy się martwić zbytnio o bezwładność robota dlatego zazwyczaj robot gwałtownie skręca w stronę przeciwnika, aż nie zostanie on wykryty przez przednie sensory – kierujemy się ma przeciwnika, oba silniki pracują z pełną mocą.

- **DOPADNIJ** - tryb włączany w przypadku, gdy zgubimy przeciwnika w stanie ATAKUJ. Jego zadaniem jest jeszcze trochę pokręcić się w kierunku, gdzie ostatnio widzieliśmy przeciwnika zanim stracimy nadzieję na znalezienie wrogiego robota. S.M.A.R.T. zazwyczaj skręca trochę ostrzej w kierunku, gdzie ostatnio widział robota oraz jedzie trochę do przodu w celu zwiększenia zasięgu czujników robota,

- **SZUKAJ** - stan dla którego nie ma warunku, zostaje wywołany w momencie, gdy w głównej pętli programu nic nie zostanie wykryte i nie ma żadnych danych do przerobienia. Robot wybiera jedną z kilku sekwencji szukających. Np. jazda po dużym kole, małym, ósemce i.t.p. Wybrana sekwencja zależy od wcześniejszych poczynań robota (w tym projekcie wpływ na wybraną sekwencje maja stany: UCIEKAJ, ATAKUJ, DOPADNIJ). W tym stanie robot także rozpoczyna walkę - pierwsza sekwencja (oraz tryb szukania w dalszej walce) jest wybierana przez zawodnika tuż przed rozpoczęciem pojedynku.

Stany są przerzucane w momencie, gdy nastąpi jakaś zmiana na sensorach lub poprzedni stan zostanie zakończony (sekwencja w stanie dojdzie do końca).
Niestety, robotowi brakuje wielu czujników umożliwiających usprawnienie algorytmu oraz zwiększenie ilości stanów – brakuje głównie czujników przemieszczenia (np. czujnika optycznego od myszki, enkoderów). Z tego powodu nie są zaimplementowane sekwencje próbujące atakować przeciwnika od tyłu. Z wcześniejszych doświadczeń wynika, że w większości walk ważniejszy jest ostry pług i większa szybkość robota niż próba zaatakowania przeciwnika od rufy.

4.2 Struktury danych

Wszystkie dane sczytywane z sensorów są przechowywane w strukturze:

```
typedef struct {
    uint8_t Tsop; //IR_LL, IR_LR, IR_L, 0, 0, IR_R, IR_RL, IR_RR
    uint8_t TsopOld; //IR_LL, IR_LR, IR_L, 0, 0, IR_R, IR_RL, IR_RR
    uint8_t Floor; // B, R, L
    uint8_t FloorOld; // B, R, L
    uint8_t Buttons; // połoga, programowanie
    uint8_t SeekOptions; // TSOP_CHANGED, SEEK_LEFT, JUST_RUN, SMALL_CIRCLE,
    //FLOOR_CHANGED,FLOOR_L_RUN, FLOOR_R_RUN, FIRST_IR
} measures;
```

Zazwyczaj wszystko jest ustawiane za pomocą operacji bitowych. Flagi:

TSOP_CHANGED - stan czujników IR się zmienił,
SEEK_LEFT - obracaj się w lewo przy szukaniu,
JUST_RUN - uciekaj!,
SMALL_CIRCLE - wykorzystuj małe obroty przy szukaniu (obracaj się w miejscu)
FLOOR_CHANGED – odczyt czujników podłoża się zmienił,
FLOOR_L_RUN - ostatnio wykrył podłogę (białą linię) po lewej stronie,
FLOOR_R_RUN – ostatnio wykrył podłogę (białą linię) po prawej stronie,
FIRST_IR - przeciwnik został wykryty w pierwszym pomiarze IR - diody świeca słabiej,

Struktura przechowująca informacje o aktualnym stanie:

```
typedef struct {
    uint8_t StateNumber;
    uint8_t PrevStateNumber;
    int8_t (*Sequence)[3]; // wsk na 3 el. tablice sekwencji
    uint8_t MoveNumber; // który ruch z danej sekwencji
    uint8_t MoveLimit; // ile jest ruchów w sekwencji
    uint8_t LastMove; // 1 jeżli zostało kilka us ostatniego ruchu. Oznacza, że
                      // należy zmienić sekwencje lub stan
} state;
```

Wszystkie struktury są globalne. Funkcje operują na nich jak na metodach.

4.3 Wykrywanie przeciwnika

Funkcja odpowiedzialna za wykrywanie przeciwnika:

```
void Timer2_Start(void) {
    TCCR2 |= _BV(CS20); // dzielnik 1, wł timer2
    Wait_us( IrTime );
}
void Timer2_Stop(void) {
    TCCR2 &= ~_BV(CS20); // wył. timer2
    // wył. wsyzstkie diody:
    PORTC &= ~(_BV(PC2));
    PORTD &= ~(_BV(PD1) | _BV(PD2) | _BV(PD4) | _BV(PD7));
    PORTB &= ~( _BV(PB6));
    Wait_us( IR_TIME );
}
```

```
uint8_t TsopMiddle_Check(void) {
    EnemyCounter = 0;
    for ( i = 0; i < TSOPCHECK && EnemyCounter != TSOPCHECKP2; ++i
) {
        Timer2_Start();
        if(PIND & _BV(PD3)) {
            EnemyCounter -= 1; // jak nie ma zmniejszamy
        }
        else {
            EnemyCounter += 1; // zwiększamy o 1 gdy jest
        }
        Timer2_Stop();
    }
    if(EnemyCounter > 0) {
        return 1; // znaleziono przrzeszkode
    }
    return 0;
}
```

Przykładowe wywołanie funkcji w programie:

```
    IrTime = IR_TIME1; // ustalenie czasu świecenia

    DiodaIR = IR_R; // wybranie diody
    if( TsopMiddle_Check() ) {
        Sensors.Tsop |= _BV( IR_R );
    }
    DiodaIR = IR_L; // wybranie diody
    if( TsopMiddle_Check() ) {
        Sensors.Tsop |= _BV( IR_L );
    }
```

Zmienna globalna DiodaIR określa, która dioda aktualnie się świeci. Zmienna globalna IrTime określa czas świecenia diody zanim pomiar zostanie wykonany - przyjmuje jedną z dwóch wartości IR_TIME1 i IR_TIME2 zdefiniowanych w configu. IR_TIME jest czasem gaszenia diody - ustawianym w configu. Gdyby nie była wywoływana funkcja Wait_us (IR_TIME); w Timer2_Stop(void) dioda pomiędzy kolejnymi pomiarami w TsopMiddle_Check(void) świeciłaby cały czas, przez co w każdym kolejnym pomiarze byłoby większe prawdopodobieństwo wykrycia przeciwnika. Czujniki tego typu pozwalają wykryć obiekt na niewielkim dystansie już przy czasie świecenia diody IrTime ustawionym w okolicach 20 us (prawdopodobnie dochodzi do tego duża niedokładność spowodowana wywoływaniem przerwać w czasie odliczania).

5. Zakończenie

5.1 Wnioski, spostrzeżenia, uwagi.

- ważnym czynnikiem ograniczającym proces tworzenia płytki okazały się wymiary robota. Wyjścia z płytki na elementy takie jak silniki, czujniki białej linii, czy zasilanie powinny być bezpośrednio nad samymi elementami aby zmniejszyć ilość okablowania. Zbyt dużo miejsca zajmowały nawet goldpiny, co spowodowało potrzebę wlutowania kabli na stałe.

- w związku z niedoborem miejsca idealnym byłoby znalezienie silników wraz z przekładniami zamiast przerabiania serw, których kształt po sklejeniu znacznie ograniczył przestrzeń wewnątrz robota. W praktyce znalezienie tak małych silników o odpowiednich parametrach jest wyjątkowo trudne.

- zastosowana w końcu obudowa wymaga kolejnej przeróbki, m. in. ze względu na zastosowane rozmieszczenie elementów wewnątrz robota. Kolejnym powodem dla przeprojektowania obudowy są trudności w łączeniu poszczególnych jej elementów oraz chęć zaprojektowania chociaż namiastki pługa.

- doszlifowania wymaga algorytm robota, przeniesiony bezpośrednio z robota I prawo Murphy'ego Macieja Gawrona.

- dużo miejsca wewnątrz robota zajął akumulator. Możliwym ulepszeniem robota byłoby znalezienie akumulatora o mniejszych wymiarach lub rozmieszczenie ogniw w różnych miejscach.

- kolejnym sposobem na zaoszczędzenie miejsca wewnątrz robota oraz obniżenie jego środka ciężkości może być zintegrowanie płytki z elektroniką z obudową robota. Warto także pomyśleć o modułowej konstrukcji płytki (np. osobnej dla czujników) – jedna płytka rozmiarów 4x4 cm znacząco ogranicza możliwości konstrukcyjne.

- rezygnacja z używanych w konstrukcjach dalmierzy laserowych okazała się bardzo dobrym rozwiązaniem – są one zbyt duże do tego typu konstrukcji, nie jest także wymagana precyzja odczytu. Diody podczerwone wraz z odbiornikami dają dużo lepszą możliwość wykrywania przeciwników w robotach microsumo, ponieważ ze względu na mniejsze rozmiary można je rozmieścić we wszystkich kierunkach – są także tańszym rozwiązaniem.