| **Nazwa i akronim projektu:**<br>*Elektroniczna szachownica - ES* | **Zleceniodawca:**<br>*dr inż. Henryk Kormański* | **Zleceniobiorca:**<br>*Łukasz Gutowski*<br>*Łukasz Kowalski*<br>*Wojciech Dmitrzak* |
|---|---|---|
| **Numer zlecenia:**<br>*6@KSDE* | **Kierownik projektu:**<br>Łukasz Gutowski | **Opiekun projektu:**<br>prof. dr hab. inż. Zdzisław Kowalczuk prof. zw |

| **Nazwa / kod dokumentu:**<br>**Sprawozdanie Przejściowe – *6@KSDE*** | **Nr wersji:**<br>03 |
|---|---|
| **Odpowiedzialny za dokument:**<br>Łukasz Gutowski | **Data pierwszego sporządzenia:**<br>20.06.2013 |
| | **Data ostatniej aktualizacji:**<br>17.01.2014 |

**Historia dokumentu**

| Wersja | Opis modyfikacji | Rozdział / strona | Autor modyfikacji | Data |
|---|---|---|---|---|
| 01 | Wersja wstępna | Całość | Gutowski Łukasz | 20.06.2013 |
| 02 | Wersja poprawiona | | Gutowski Łukasz | 17.01.2014 |
| 03 | Wersja ostateczna | | Kowalski Łukasz | 08.01.2014 |

# 1 Wprowadzenie - o dokumencie

## 1.1 Cel dokumentu

Celem dokumentu jest przedstawienie stanu realizacji projektu grupowego, postępów oraz wdrożonych rozwiązań.

## 1.2 Zakres dokumentu

- założenia początkowe;
- prezentacja bieżącego stanu realizacji projektu;
- przedstawienie rozbieżności pomiędzy założeniami projektowymi, a prototypem;
- sugestie dotyczące rozwoju projektu oraz przyszłych zmian w prototypie.

Dodatkowe informacje dotyczące projektu znajdują się w dokumentach powiązanych (dokumentacja osprzętu – schemat, plakat, kod źródłowy programu).

## 1.3 Odbiorcy

Katedra Systemów Decyzyjnych

## 1.4 Wykorzystana terminologia

- ARM - *Advanced RISC Machine* – 32-bitowy procesor typu RISC, wykorzystywany w systemach wbudowanych;
- RAM - *Random Access Memory* – pamięć o dostępie swobodnym;
- RISC - *ReducedInstruction Set Computer* – architektura procesorów, charakteryzująca się zredukowaną liczbą rozkazów;
- RJ-45- typ złącza stosowany w przewodach sieciowych;
- SD (*Secure Digital*) - standard kart pamięci o pojemności nie przekraczającej 4 GB

# 2 Stan realizacji projektu

## 2.1 Założenia początkowe

Celem projektu jest stworzenie multimedialnej szachownicy – robota grającego w szachy z wykorzystaniem zestawu silników, umieszczonych pod planszą. Robot oparty na układzie Raspberry Pi. Algorytm zapisany w pamięci mikrokomputera decyduje o kolejnym ruchu na podstawie aktualnej sytuacji na szachownicy. Zmiana położenia figury przez gracza wykrywana za pomocą czujników zmian pola magnetycznego. Za figury poruszane przez robota odpowiada zestaw silników krokowych i magnesów umieszczonych pod planszą.

## 2.2 Osiągnięte rezultaty

### 2.2.1. Oprogramowanie

Jako podstawę systemu wybraliśmy platformę komputerową Raspberry Pi, opartą na układzie Broadcom BCM2835 SoC, składającą się z procesora ARM1176JZF-S 700 MHz, procesora graficznego VideoCore IV GPU i 512 megabajtów pamięci operacyjnej RAM. Układ został wyposażony w kartę pamięci o pojemności 4GB, na której zainstalowaliśmy system operacyjny Raspbian.
Szczegółowe dane na temat parametrów układu przedstawione w tabeli poniżej:

| | |
|---|---|
| Procesor: | 700 MHz ARM1176JZF-S core (ARM11) |
| Układ graficzny: | BroadcomVideoCore IV<br>(ze wsparciem dla OpenGL ES 2.0, 1080p30 h.264/MPEG-4 AVC) |
| Pamięć SDRAM: | 512 MB (współdzielona przez układ graficzny) |
| Dostępne porty | 2 x USB 2.0<br>1 x Composite RCA<br>1 x HDMI 1.4<br>1 x 3,5 mm Jack<br>1 x Ethernet (RJ45)<br>8 x GPIO, UART, szyna I²C , szyna SPI |
| Nośnik danych: | złącze kart SD / MMC / SDIO |
| Połączenia sieciowe: | 10/100 Ethernet (RJ45) |
| Zasilanie: | 700 mA (3.5 W) |

**Tabela 1. Parametry układy Raspberry Pi**

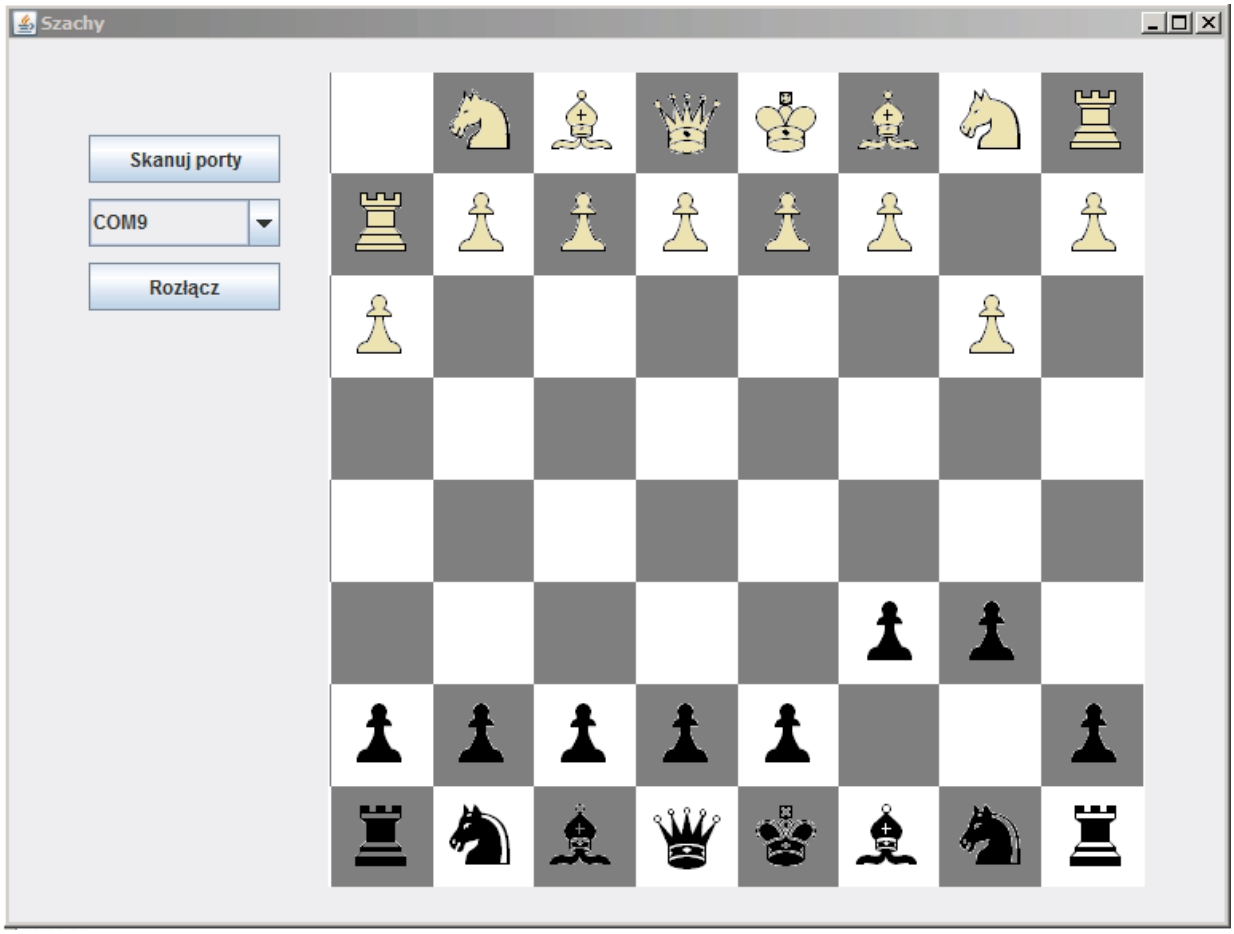
Programowanie układu odbywa się za pomocą komputera klasy PC. Do komunikacji wykorzystana jest sieć lokalna (złącze RJ – 45) oraz portszeregowy (złącze USB). Na układzie Raspberry Pi został uruchomiony serwer VNC. Jako, że transmisja graficzna nadmiernieobciąża łącze, zdecydowaliśmy się ograniczyć do formy tekstowej (konsola poleceń).

Program operujący całym zestawem szachowym został napisany w języku C, z wykorzystaniem bibliotek WiringPi.

### 2.2.2 Komunikacja z komputerem PC

Po zaprogramowaniu układ może działać samodzielnie lub współpracować z komputerem klasy PC. W drugim przypadku do transmisji informacji używany jest interfejs RS232.

Program, służący do realizacji połączenia, został przez nas napisany w języku JAVA z wykorzystaniem środowiska NetBeans. Aplikacja przesyła oraz odbiera dane z portu szeregowego i przedstawia je w formie graficznej, zrozumiałej dla użytkownika. Program działa poprawnie zarówno na systemach z rodziny Windows, jak i na dystrybucjach linuksowych.




**Rysunek 1. Program szachowy**

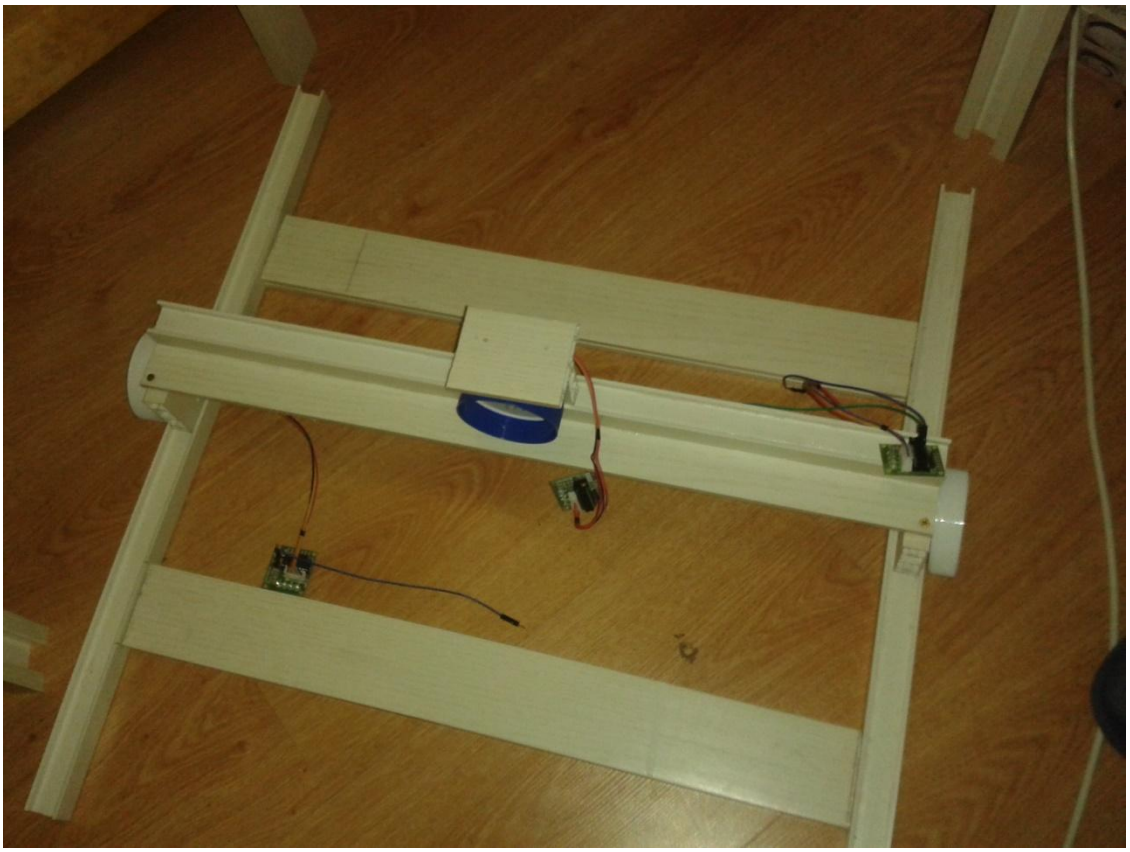
Do komunikacji z komputerem wykorzystaliśmy układ PL-2303HX, konwertujący sygnał z portu szeregowego na USB.

**Rysunek 2. Układ PL-2303HX**

### 2.2.3. Konstrukcja szachownicy

Do wykonania planszy szachownicy wykorzystaliśmy płytę paździerzową o wymiarach 60 cm x 60 cm x 0.5 cm. Płyta została umieszczona na czterech plastikowych wspornikach o wysokości 20,5 cm. Pod płytą znajduje się układ silników. Za ruch figur wzdłuż osi rzędnych odpowiadają dwa silniki, umieszczone równolegle na dwóch prowadnicach, każda o długości 60 cm. Przesuwanie figur po osi odciętych realizowane jest z wykorzystaniem trzeciego silnika, poruszającego się prostopadle do ruchu dwóch poprzednich.

Wszystkie elementy konstrukcyjne (wsporniki, prowadnice), z wyjątkiem planszy szachownicy, zostały wykonane z lekkiego plastiku.



**Rysunek 3. Podstawa szachownicy wraz z silnikami**

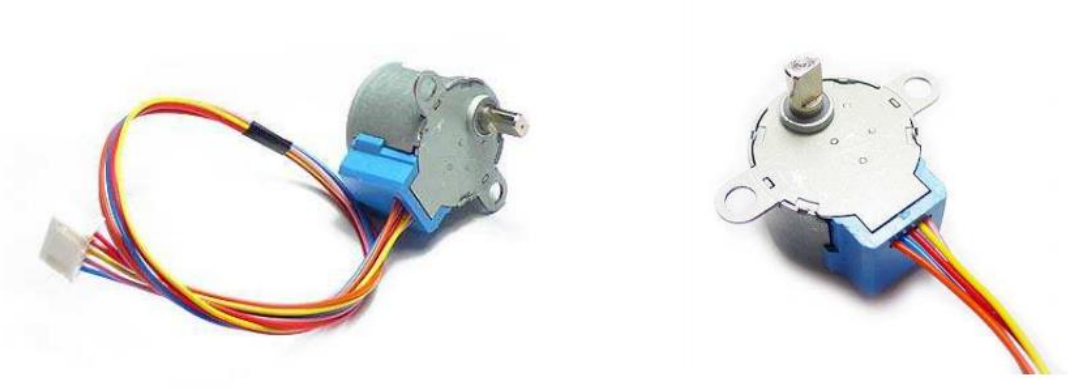
### 2.2.4 Układ sterowania silnikami

Zaprojektowaliśmy i skonstruowaliśmy prototypowy moduł układu sterowania silnikami krokowymi. Jego głównym zadaniem było zmienianie pozycji figur na planszy. Początkowo planowaliśmy wykorzystać dwa sterowniki L293D oraz zestaw silnikówVexta PX-245 – niestety, ze względu na problemy z zasilaniem oraz wysokie błędy pozycjonowania silników, zmuszeni byliśmy znaleźć inne rozwiązanie.

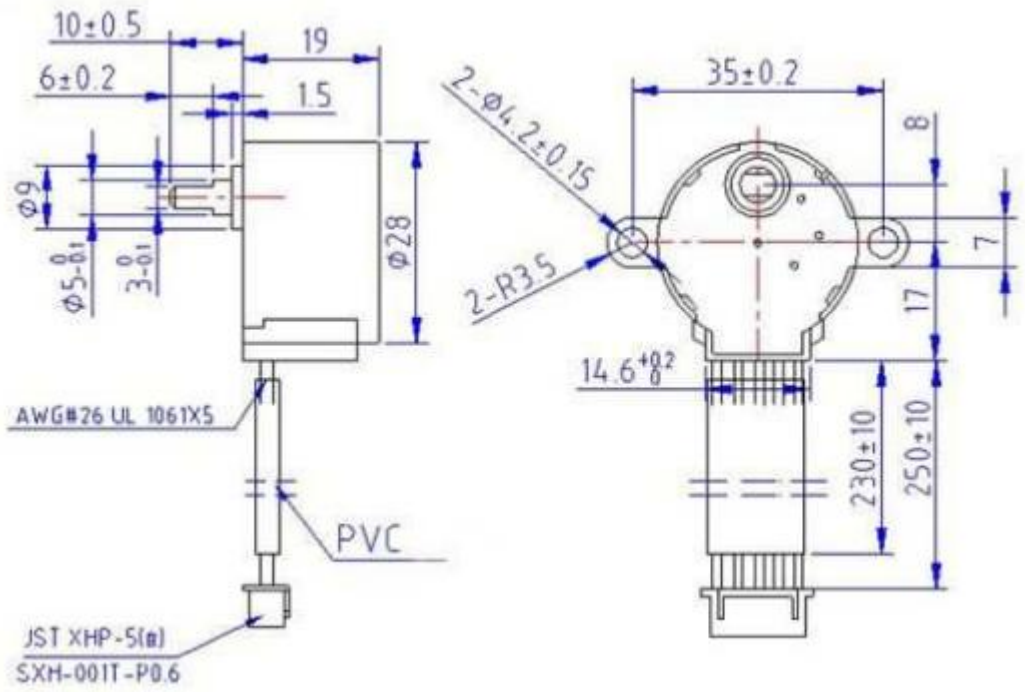
Jako napęd modułu sterowania figurami posłużyły trzy silniki krokowe 28BYJ-48 5VDC o następujących parametrach:

| | |
|---|---|
| Model | 28BYJ-48 5VDC |
| Napięcie zasilania silnika | 5V |
| Ilość faz | 4 |
| Ilość kroków na pełny obrót | 64 |
| Kąt / obrót | 5,625° |
| Rezystancja (25°C) | 50 Ω (±7%) |
| Średnica silnika | 28 mm |
| Odległość otworów mocujących | 35 mm |

**Tabela 2. Parametry silnika 28BYJ-48 5VDC**



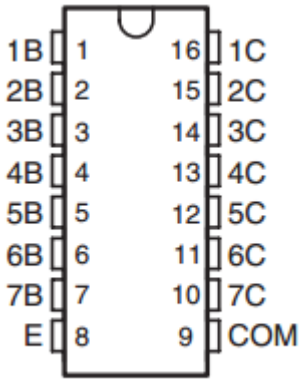
**Rysunek 4. Silniki 28BYJ-48**




**Rysunek 5. Silniki 28BYJ-48 - wymiary**

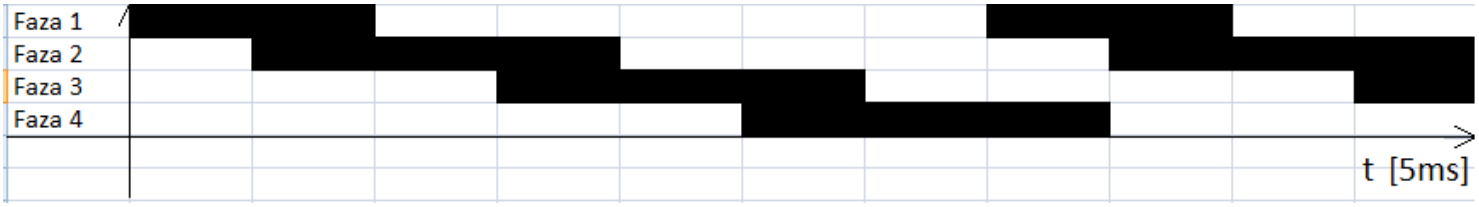
Do sterowania silnikiem zdecydowaliśmy się wykorzystać układ scalonyULN2003 (tranzystor Darlingtona) o następujących parametrach:

| | |
|---|---|
| Prąd kolektora (pojedyncze wyjście) | 500-mA |
| Napięcie na wyjściu | 5V - 50 V (w zależności od napięcia zadanego) |



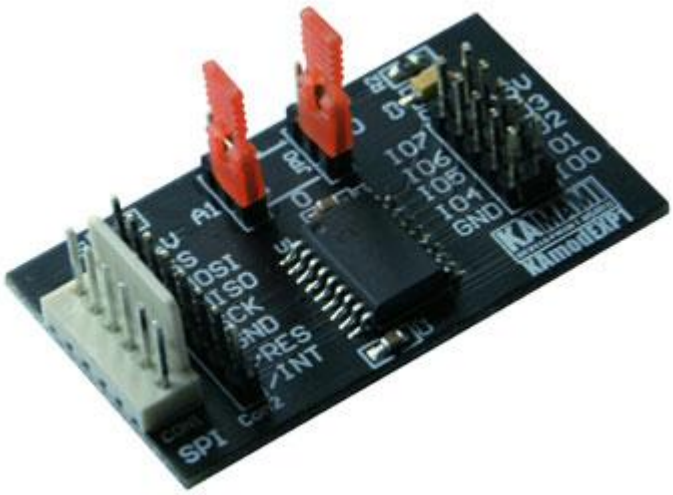

**Rysunek 6. Układ scalony ULN2003**

Sterowanie silnikami odbywa się poprzez załączenie poszczególnych faz przez program, znajdujący się w pamięci mikrokomputera Raspberry Pi. Czas trwania pojedynczej fazy wynosi 5 ms.




**Rysunek 7. Sekwencja faz sterowania silnikami**

Program pozwala na zdefiniowanie długości kroków oraz określenie liczby obrotów. Do sterowania silnikami zostały wykorzystane piny GPIO. Z powodu małej ilości dostępnych złączy, zdecydowaliśmy się na wykorzystanie ekspandera GPIO, sterowanego poprzez interfejs SPI.




**Rysunek 8. Ekspander portów GPIO**

Ekspander oparty jest na układzie scalonym MCP23S08.

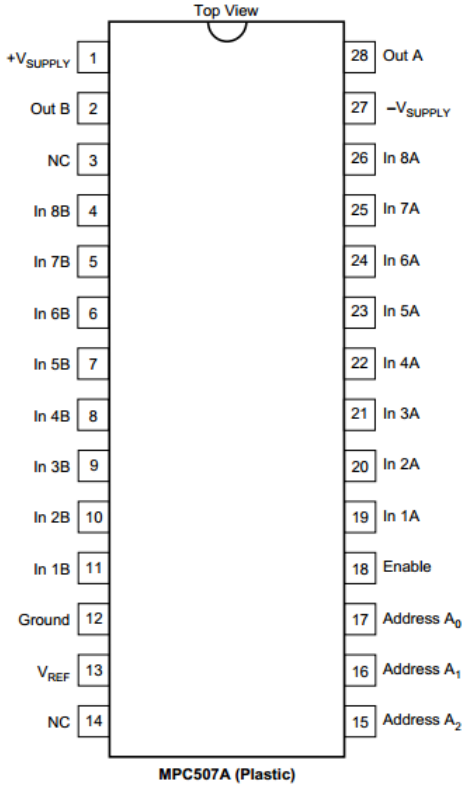
### 2.2.5. Wykrywanie figur na planszy

Do wykrywania zmiany pozycji figury wykorzystujemy kontaktrony magnetyczne typu NO, umieszczone pod poszczególnymi polami szachownicy. Podstawowe parametry czujników zostały przedstawione w tabeli.

| *Parametr* | *Jednostka* | *Wartość* |
|---|---|---|
| Zakres działania | AT | 18-32 |
| ReleaseRange | AT | 7=27 |
| Typowy czas pojedynczej operacji | ms | 0.25 (40 AT) |
| Czas drgania styków (typ.) | ms | 0.15 (40 AT) |
| Release Time (max) | µs | 30 (40 AT) |
| Częstotliwość rezonansowa (typ.) | Hz | 5500 |

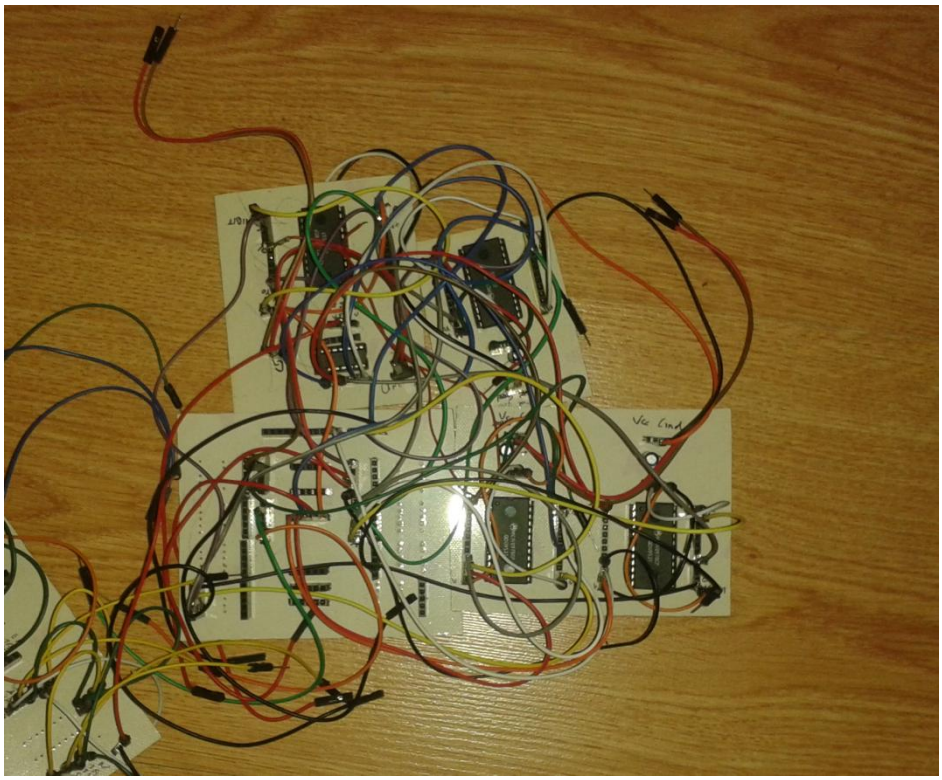
**Tabela 3. Parametry kontaktronów magnetycznych**

Do odbioru sygnałów z 64 czujników pod planszą wykorzystujemy zbiórmultiplekserów. Zdecydowaliśmy się użyć czterech układów scalonych MPC507. Każdy z nich odbiera sygnał analogowy z szesnastu czujników jednocześnie. Multipleksery zostały umieszczone na wytrawionych przez nas płytach PCB.



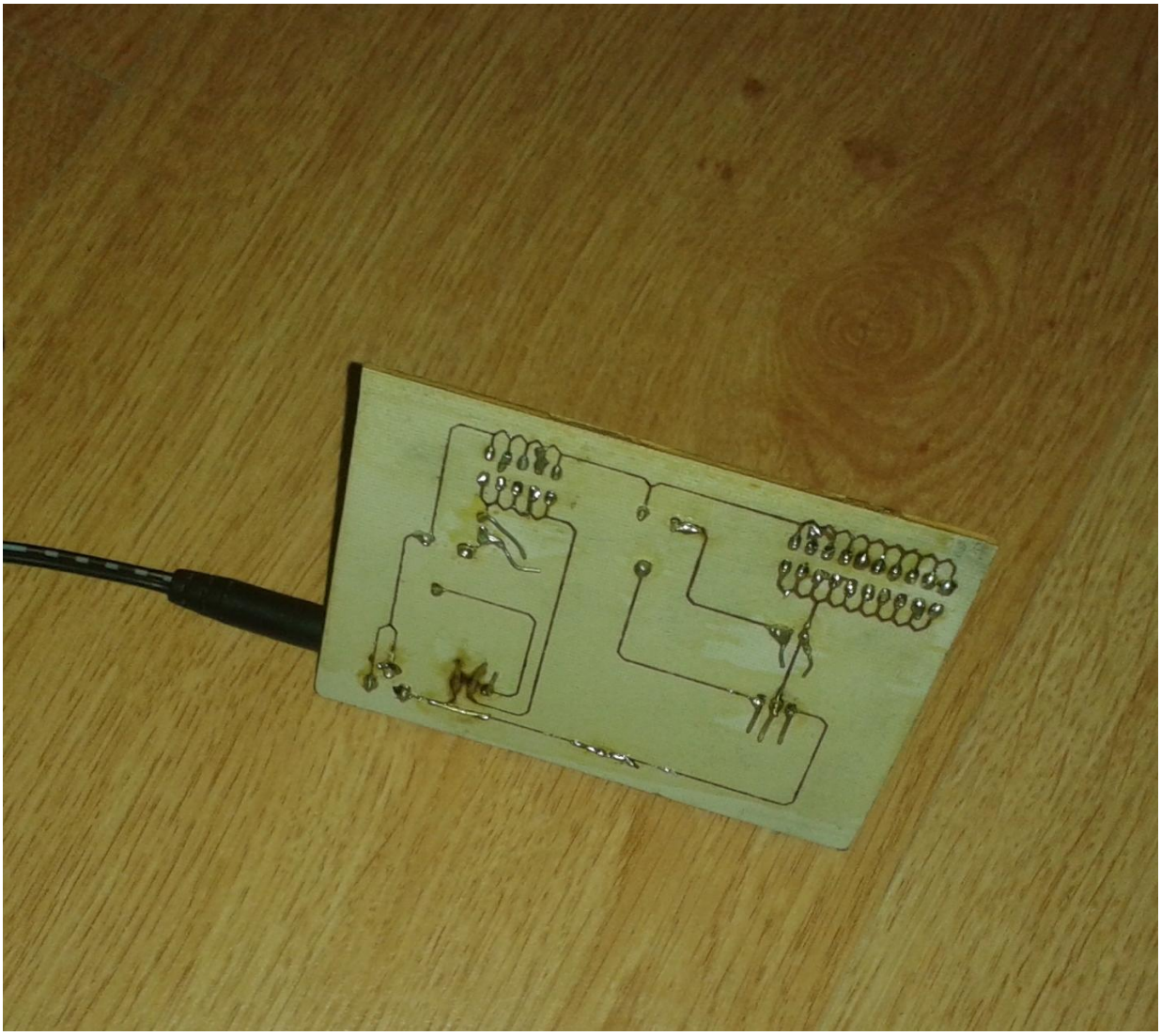

**Rysunek 9. Układ scalony MPC507**

Sposób łączenia multiplekserów z czujnikami oraz mikrokomputerem Raspberry Pi, został przedstawiony na schemacie, dołączonym do sprawozdania.



**Rysunek 10. Układ logiczny z ośmioma multiplekserami**

**Rysunek 11. Płytka zasilająca**



**Rysunek 12. Płytka zasilająca –wytrawione ścieżki**

### 2.2.6. Algorytm określający kolejne ruchy silników krokowych

Aby umożliwić zautomatyzowany ruch figur po szachownicy zaprogramowaliśmy algorytm obliczający optymalną ścieżkę figury z pola startowego do końcowego. Występowanie figur o „nielogicznych" dla komputera ruchach (przykładowo: skoczek, przeskakujący przeszkody lub wykonanie manewru roszady) wymusiło na naspowiększenie tablicy dostępnych ruchów o krawędzie każdego z pól. W ten sposób z tablicy 8x8 utworzyliśmy tablicę 19x17 (dodatkowe 2 pola poza szachownicą używane do usuwania zbitych przez komputer figur z planszy).

Przy obliczaniu optymalnej drogi figury po szachownicy wykorzystaliśmy algorytm Propagacji Fali. Algorytm Propagacji Fali znajduje najkrótszą, bezkolizyjną ścieżkę między dwoma punktami na mapie rastrowej o zadanej metryce. Jak sama nazwa wskazuje działa na zasadzie rozchodzenia się fali. Przyrównując do zjawisk fizycznych działanie algorytmu dobrze obrazuje rozpływająca się w labiryncie woda. Woda płynąc po najmniejszej linii oporu sama znajdzie wyjście z labiryntu. W przypadku przeszukiwania grafu w celu znalezienia najbardziej optymalnej ścieżki będzie nią ta, którą algorytm znajdzie najszybciej.

Każdemu polu na szachownicy przyporządkowuje się wagi, którymi kieruje się algorytm przy wyznaczaniu ścieżek. Obszary zajęte przez przeszkody (figury) oraz punkt

początkowy i końcowy są omijane. To etap początkowy działania metody, który rozmieszcza wartości zgodnie z odległością od punktu docelowego, korzystając z wybranej metryki.

Nadawanie wartości pól nie należącym do przeszkód:

- Znajdź punkt początkowy i nadaj mu wartość zero.
- Sąsiadującym punktom nadaj wartość 1.
- Rozpatruj kolejne punkty sąsiadujące z już opisanym i nadaj im wartość o 1 większą niż sąsiadowi.
- Rozrastająca się w ten sposób fala dociera do punktu końcowego, o ile taki istnieje. Przerwij działanie algorytmu jeśli nadano wartość wszystkim możliwym polom i punkt końcowy nie został znaleziony. Przerwij działanie algorytmu jeśli punkt końcowy został znaleziony (nie nadawaj wartości kolejnym polom, skoro cel został osiągnięty). W przeciwnym przypadku wróć do poprzedniego kroku.

Jeśli algorytm znajdzie punkt końcowy to oznacza, że istnieje ścieżka łącząca punkt początkowy z końcowym. W tym momencie zaczyna się drugi etap algorytmu, czyli odtworzenie ścieżki optymalnej. Zakładamy, że figura znajduje się już w punkcie końcowym, ponieważ tam zatrzymał się algorytm. Celem jest przemieszczenie figury od punktu końcowego do początkowego.

Schemat rekonstrukcji ścieżki:

- Znajdź punkt końcowy
- Sprawdź sąsiadujące z tym punktem pola wolne od przeszkód. Szukaj pola o wartości różniącej się o 1 od rozpatrywanego punktu. Dodaj znalezione pole do listy (rozwiązania)
- Przejdź do właśnie dodanego punktu i powtarzaj czynność z punktu poprzedniego
- Powtarzaj poprzednie dwa punkty, aż znajdziesz punkt o wartości zero (punkt początkowy). Zakończ algorytm.

Otrzymana lista pól zawiera zawsze optymalną ścieżkę, dla zadanej mapy i konfiguracji przeszkód. Niemniej jednak, może istnieć więcej optymalnych ścieżek przy zadanej konfiguracji przeszkód, jednak algorytm Propagacji Fali wybiera i przedstawia tylko pierwszą ze znalezionych.

## 2.3 Rozbieżności i sugestia zmian w realizacji projektu

O ile teoretyczny opis projektu nie nastręczył większych trudności, jego realizacja nie obyła się bez problemów.

- Użyte silniki krokowe charakteryzują się niską dokładnością pozycjonowania, co bardzo utrudnia sterowanie. Rozważamy zmianę na inny model lub wykorzystanie serwomechanizmów;
- wykorzystywane przez nas czujniki pola magnetycznego (tzw. kontaktron, z angielskiego Reed Switch) okazały się zawodne jak i trudne w zainstalowaniu tak, aby nie zaburzały nawzajem swoich odczytów. Możliwe, iż zdecydujemy się na wykorzystanie innych czujników (rozważamy użycie w zamian fotodiod półprzewodnikowych);
- kolejny problem związany jest z komunikacją bezprzewodową. Z racji trudności z programowaniem układu Wi-Fi, zmuszeni byliśmy używać tradycyjnych kabli sieciowych RJ-45.

## 2.4 Dalszy rozwój projektu

W najbliższych miesiącach zamierzamy rozwijać projekt. Oprócz usprawnienia wyżej opisanych niedogodności, chcemy poszerzyć działanie układu o nowe funkcje.

Algorytm szachowy nie został jeszcze w pełni zaimplementowany; wybrany ogólnodostępny program szachowy (FAILE, na licencji MIT) jest przez nas nadal analizowany aby dostosować go do wymagań projektu (ujednolicenie zmiennych, dostosowanie programu, aby odpowiadał na wszystkie sytuacje wymagane w projekcie). Po zakończeniu wyżej wymienionej implementacji, następnym krokiem pozostanie już tylko ulepszanie systemu, aby funkcjonował w jak najdokładniejszy i najszybszy sposób.

# 3 Załączniki

| L.p. | Nazwa dokumentu | Data przyjęcia | Nazwa pliku | Podpis Opiekuna Projektu |
|---|---|---|---|---|
| 1. | Plakat | | | |
| 2. | Schemat układu | | | |
| 3. | Kod źródłowy programu | | | |

# 4 Bibliografia

[1] www.raspberrypi.org, 04.02.2014, 20:30
[2] www.wiringpi.com, 04.02.2014, 21:00
[3] www.faile.sourceforge.net, 04.02.2014, 21:00
[4] "Stepping Motors: A Guide to Theory and Practice (4th Edition)", Paul Acarnely, 01.01.2002, IET
[5] " Wyznaczanie ścieżek bezkolizyjnych dla platformy mobilnej w oparciu o mapy rastrowe", Wojciech Dmitrzak, Przemysław Hirsz, 2013