Politechnika Wrocławska
Wydział Mechaniczny

Kierunek: Mechatronika

Praca Inżynierska

Budowa i oprogramowanie zautomatyzowanej szachownicy sterowanej głosowo

Construction and software development an automated chess game controlled by voice

Autor:

Adrian Kurzawski

Promotor:

Dr inż. Paweł Krowicki, Katedra technologii laserowych, automatyzacji i organizacji produkcji.

Spis treści

1. Wstęp

Nasza cywilizacja rozwija się w niesamowitym tempie. Obecnie przeciętny człowiek, bez żadnych specjalistycznych uprawnień posiada dostęp do wszelakich technologii. Dzięki sieci internetowej mamy dostęp do ogromnych zasobów wiedzy zgromadzonej przez ludzkość na przestrzeni dziejów. Wobec tego każdy może skonstruować urządzenie składające się z układu mechanicznego, elektronicznego, sterowane w określony sposób. Ogranicza nas tylko ilość poświęconego czasu na zdobycie odpowiedniej wiedzy i oczywiście środki finansowe.

Rozrywka jest nieodłączną częścią naszego życia. Biorąc pod uwagę ilość wolnego czasu, najczęściej rozrywce oddają się dzieci. W dzisiejszych czasach dzieci częściej czas wolny spędzają przed komputerem, czy smartphone‘m. Rodzicom coraz większy problem sprawia znalezienie pociechom kreatywnego, czy pouczającego zagospodarowania wolnego czasu. Istnieje wiele gier planszowych, karcianych, które wymagają od graczy kreatywnego myślenia oraz poprawiają kontakty interpersonalne. Jedną z takich gier są szachy. Tak właśnie zrodził się pomysł zbudowania zautomatyzowanej szachownicy sterowanej głosowo. Zastosowanie dostępnych technologii do poruszania figur zdalnie czyni grę w szachy dużo ciekawszą. Przesuwanie figur ręcznie po polach szachownicy raczej nie jest interesującym zajęciem samym w sobie. Ponieważ rozgrywka jest skomplikowana, wymaga taktycznego myślenia i przewidywania wielu kroków na przód. Szachy nie przyciągają dzisiaj do siebie ogromnych ilości graczy. Urozmaicenie w postaci możliwości wydawania komend głosowych, które powodują ruch figury po polach szachownicy przykuwa uwagę najmłodszych. Sama rozgrywka staje się bardziej absorbująca i może przynieść wiele radości osobom, które dopiero uczą się zasad gry i odkrywają schematy prowadzenia rozgrywki. Taka zautomatyzowana szachownica powinna zainteresować grą w szachy wielu, dotąd nie nastawionych do tej gry entuzjastycznie.

Inspiracją do sposobu poruszania pionem stał się film „Harry Potter i kamień filozoficzny”. W fantastycznym świecie magii i czarodziejów, dwaj główni bohaterowie grali w nazywane przez nich „Szachy Czarodziejów”. Na pierwszy rzut oka nie różnią się niczym od standardowych, znanych ze świata rzeczywistego. Mimo identycznych zasad rozgrywka odbywała się bez udziału kończyn górnych. Gracze wydawali polecenia głosowe, a figury poruszały się po szachownicy za pomocą tajemniczej siły nazywanej w tym uniwersum „magią”. Jedną z możliwości odwzorowania tego sposobu rozgrywki jest umieszczenie pod szachownicą układu mechanicznego przemieszczającego elektromagnes, który generując pole magnetyczne oddziałuje na ferromagnetyczne figury. Telefon umożliwia rejestrowanie komend, a odpowiednia aplikacja przesyła komendy do mikrokomputera sterującego układem mechanicznym. Takie rozwiązanie zostało opracowane szerzej przez autora pracy do zbudowania zautomatyzowanej szachownicy sterowanej głosowo.

2. Cel i zakres pracy

Celem pracy jest zbudowanie w pełni zautomatyzowanej szachownicy sterowanej głosem. Należało zaprojektować i wykonać układ kinematyczny służący do przemieszczania figur. Konieczne jest zbudowanie dwóch modułów translacyjnych pozwalających przemieszczać się w dwóch zależnych osiach. Chwytanie figur w celu ich przemieszczenia powinno być realizowane bezdotykowo, np. za pomocą sterowanego pola magnetycznego. Ponieważ większość elementów jest od siebie zależna, konieczne jest odpowiednie rozplanowanie prac i przygotowanie projektu.

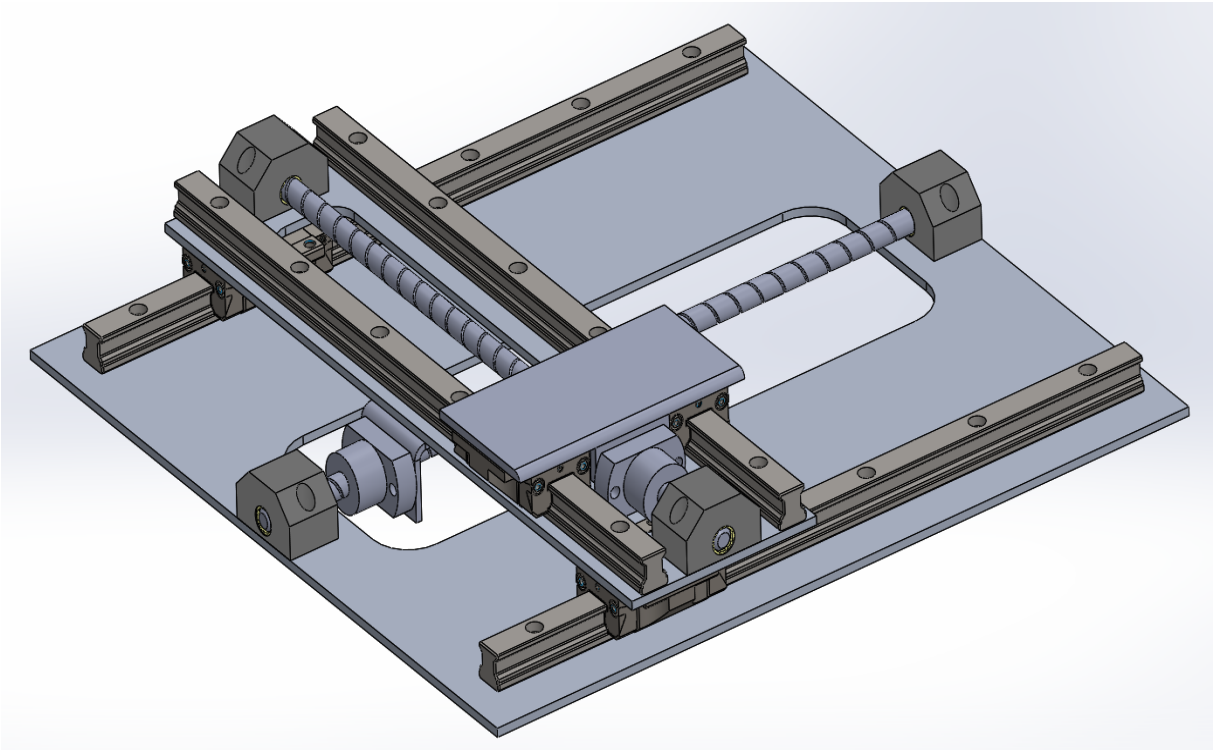
Kolejnym, bardzo istotnym krokiem jest opracowanie aplikacji rozpoznającej mowę i przekazanie komend do mikrokomputera. Należało napisać aplikacje dla telefonu z systemem Android. Trzeba było skorzystać z odpowiednich bibliotek, które realizują rozpoznawanie mowy za pośrednictwem Internetu, stąd niezbędne jest połączenie z siecią. Przekazanie komend z telefonu na mikrokomputer powinno odbywać się poprzez komunikację bezprzewodową. Wszystko powinno odbywać się podczas działania jednej aplikacji. Połączenie bezprzewodowe musi odbywać się automatycznie i oprócz włączenia odpowiedniego modułu na telefonie, nie powinno wymagać żadnej ingerencji od użytkownika. Po włączeniu aplikacji użytkownik powinien mieć możliwość wybrania momentu, w którym zdecyduje się wypowiedzieć odpowiednią komendę.

Kolejnym etapem jest zaproponowanie algorytmu sterowania układem kinematycznym. Algorytm powinien być wykonywany na mikrokomputerze z systemem Linux. Komendy odebrane za pośrednictwem komunikacji bezprzewodowej, muszą być przetwarzane przez algorytm na odpowiednie sygnały sterujące układem kinematycznym. Należało użyć odpowiednich modułów lub układów scalonych do sterowania poszczególnymi elementami układu kinematycznego.

Ostatnim etapem jest przeprowadzenie testów. Należało sprawdzić poprawność rozpoznawania mowy oraz działanie systemu odpowiedzialnego za przemieszczanie figur po szachownicy. Testy powinny być przeprowadzone na kilkuosobowej grupie kontrolnej oraz przy wykorzystaniu odpowiedniego oprogramowania. W celu sprawdzenia wszystkich współpracujących ze sobą elementów i urządzeń, koniecznością jest rozegranie przykładowej partii szachów.

3. Projekt szachownicy

Planowanie pracy zostało rozpoczęte przez opracowanie modelu 3D w systemie CAD (rysunek 1.) przy pomocy programu SolidWorks. Dostępność niektórych elementów, takich jak; szyny, wózki, płyty aluminiowe, blachy gięte zdefiniowała rozwój prac. Elementy te zostały zamodelowane oraz umieszczone w systemie CAD, na tej podstawie dobrano pozostałe fragmenty. W ten sposób powstał wstępny model układu szachownicy, na podstawie którego zbudowano zespół mechaniczny oraz dobrane zostały części elektroniczne. Opracowanie modelu okazało się sporym wyzwaniem. W celu maksymalnego obniżenia kosztów budowy urządzenia, autor pracy starał się wykorzystać jak najwięcej dostępnych przed budową elementów. Przygotowanie modelów poszczególnych elementów z dostępnych półfabrykatów, pozwoliło na ich bezproblemowe wykonanie w domowym warsztacie, przy użyciu narzędzi takich jak; wiertarka, szlifierka kątowa, piła tarczowa, imadło, młotek, piła ręczna. Do znalezienia odpowiednich zastosowań niektórych elementów posłużono się literaturą [1].



Rysunek 1. Projekt części mechanicznej szachownicy

Jednostką sterującą układem kinematycznym jest Raspberry Pi. Elementy elektroniczne takie jak; sterowniki silników, czujniki, diody LED, przekaźniki, zostały tak dobrane, aby w możliwie wygodny sposób można było je podłączyć do mikrokomputera za pomocą portów GPIO, tzn. bez konieczności stosowania oddzielnego zasilania, czy wzmacniania sygnału sterującego, np. za pomocą tranzystorów. Jedynie zasilanie silników i elektromagnesu wymaga zewnętrznego zasilania.

4. Dobór elementów konstrukcyjnych szachownicy

Po wykonaniu projektu należało przystąpić do prac nad budową układu kinematycznego służącego do przemieszczania figur, posłużono się wiedzą z literatury[2]. Wykorzystano do tego prowadnice i wózki, które są napędzane zespołem śruby trapezowej z nakrętką, połączonej sprzęgłem z silnikiem krokowym. Chwytanie figur, w celu ich przemieszczenia, realizowane jest za pomocą elektromagnesu. Jako elementy łączące poszczególne podzespoły wykorzystano płyty drewniane, hdf, płyty ze szkła akrylowego oraz blachę ze stali nierdzewnej. Wszystkie elementy układu zostały fizycznie wykonane z dostępnych dla autora pracy elementów lub zakupione w punktach z odpowiednim do budowy wyposażeniem.

Płyty główne

Rysunek 2. Zastosowane w pracy płyty główne

Jako płytę główną użyto drewnianą płytę o wymiarach 600x600mm o grubości 8mm, dostępną w momencie budowy szachownicy. Do tej płyty zamontowany jest dolny moduł translacyjny oraz płyta górna. Płyta górna to trzymilimetrowa płyta hdf o wymiarach 390x600mm i grubości 3mm. Na płycie hdf znajduje się pole szachownicy.

Pole szachownicy

Rysunek 3. Przestrzeń szachownicy – widok od góry.

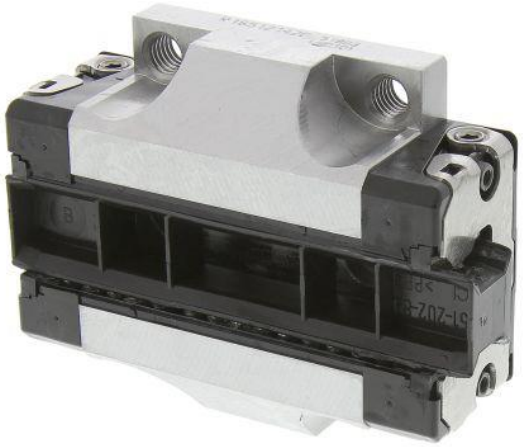
Do płyty górnej hdf została przykręcona mata ochronna pod fotel grubości 0,5mm odpowiednio docięta na wymiary 350x350mm. Na matę ochronną przyklejono okleinę w teksturze szachownicy. To właśnie ta okleina przyklejona na matę tworzy pole, po którym poruszają się figury. Dzięki niewielkiej grubości elektromagnes może poruszać zaraz pod pionami.

Prowadniki liniowe – szyny

Rysunek 4. Prowadnice liniowe Bosch Rexroth

Aby wykonać ruch konieczne jest zdefiniowanie toru. Do zdefiniowania przemieszczenia liniowego zostały użyte prowadnice liniowe firmy Bosch Rexroth, do których miałem dostęp. Są to szyny o wysokości 15mm, szerokości 15mm i długości 420mm. Zostały użyte 4 takie szyny.

Wózki



Rysunek 5. Wózek poruszający się po szynie

Wózki składają się z dwóch układów kulek tworząc łożysko, poruszające się po szynie. Stosując odpowiednie smarowanie, wózki firmy Bosch Rexroth wraz z szynami tej samej firmy stanowią podstawowy zespół do przemieszczeń liniowych, generujący bardzo mały opór. Zostały użyte 4 takie wózki.

Śruba trapezowa z nakrętką

Rysunek 6. Śruba trapezowa Tr8x8 z nakrętką

Do wykonania przemieszczenia liniowego użyty został układ śruby trapezowej Tr8x8x400mm wraz z nakrętką. Zastosowano śrubę o skoku 8mm ponieważ dzięki większemu skokowi niż przy standardowej śrubie M8 przy tej samej prędkości obrotowej silnika, można uzyskać większą prędkość liniową nakrętki względem podłoża. Zostały zastosowane dwie takie pary, do przemieszczeń w dwóch prostopadłych osiach.

Uchwyt z łożyskiem

Rysunek 7. Uchwyt z łożyskiem z otworem o średnicy 8mm.

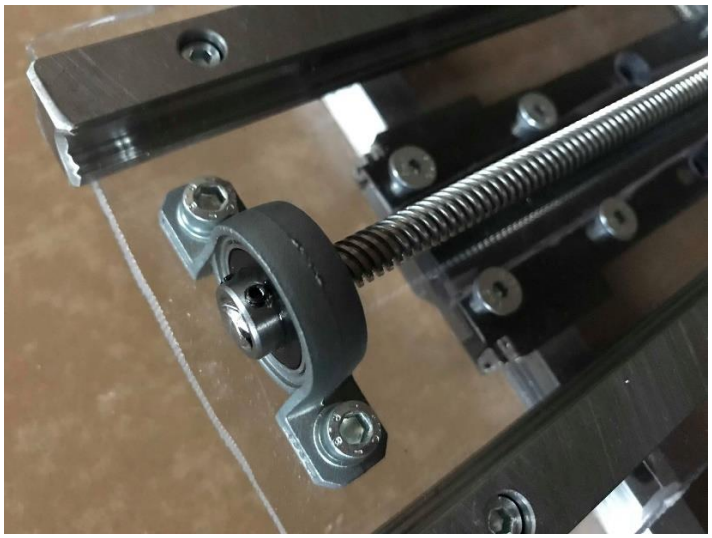
Mocowanie śruby zostało zrealizowane za pomocą uchwytu z łożyskiem. Dzięki temu elementowi śruba może bez większych oporów obracać się względem własnej osi. Zastosowane zostały dwa takie uchwyty na każdą ze śrub.

Sprzęgło aluminiowe

Rysunek 8. Łożysko aluminiowe z otworami o średnicy 8mm oraz 5mm.

Połączenie wału silnika ze śrubą realizowane jest za pomocą sprzęgła aluminiowego. Sprzęgło zostało tak dobrane, aby zminimalizować modyfikacje potrzebne do połączenia elementów. Standardowe średnice większości wałów silników z dostępnego przedziału cenowego to 5mm. Sprzęgło posiada otwór o średnicy 8mm, za pomocą którego zrealizowano połączenie ze śrubą oraz otwór 5mm, za pomocą którego połączono się z wałem silnika. Dzięki zastosowaniu sprzęgła podatnego nie jest konieczna dokładna koncentryczność wału silnika względem śruby. Sprzęgło pozwala na tolerancje współliniowości osi rzędu kilku milimetrów.

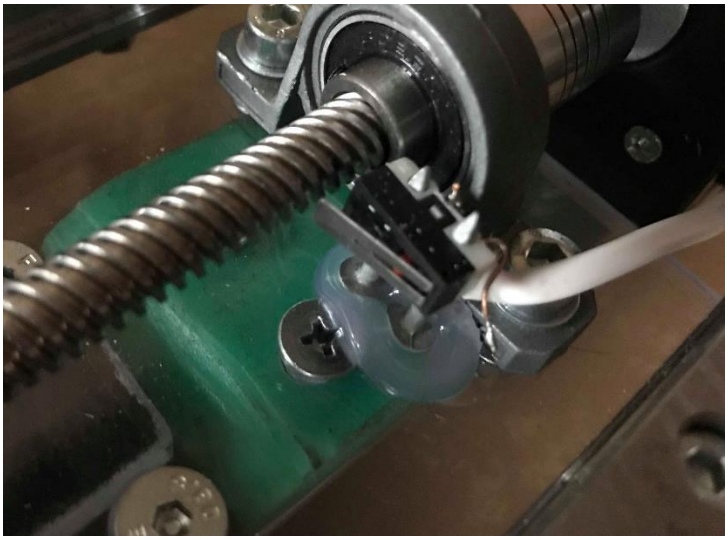
Płyta łączeniowa



Rysunek 9. Górny moduł translacyjny mocowany do płyty ze szkła akrylowego.

Integracja zespołu prowadnic i wózków z zespołem śruby z nakrętką została zrealizowana za pomocą szkła akrylowego(Plexiglas) grubości 4mm, do którego autor miał nieograniczony dostęp. Plexiglas jest materiałem łatwym do obróbki w warunkach domowych, a jednocześnie zachowuje wystarczające właściwości mechaniczne.

Polioksymetylen - POM



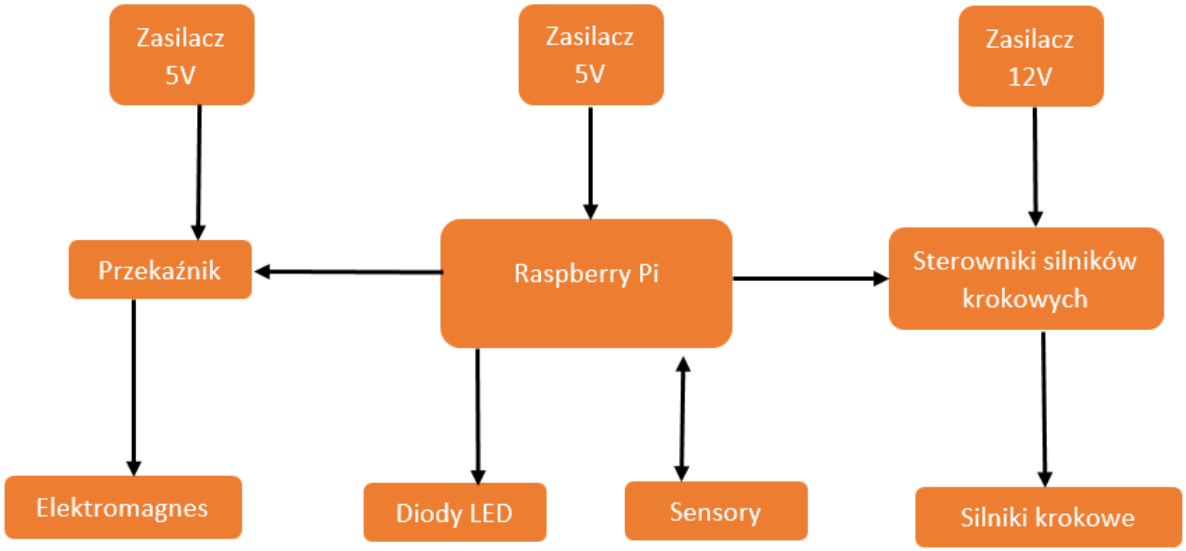
Rysunek 10. Miejsce mocowania silnika do górnego modułu translacyjnego.

Jako element dystansowy została użyta kostka z zielonego polioksymetylenu. Jest to materiał o odpowiednich do tego zastosowania właściwościach mechanicznych, a jednocześnie łatwy w obróbce w warunkach domowych. Konieczne było zastosowanie elementu dystansowego, ponieważ silnik musiał być zamontowany poniżej płaszczyzny płyty górnego modułu translacyjnego.

5. Dobór elementów układu elektronicznego

Opracowano odpowiedni układ elektroniczny, aby umożliwić sterowanie zespołami mechanicznymi. Zespoły odpowiedzialne za translacje liniową napędzane są przez silniki krokowe. Zostały użyte specjalne sterowniki silników krokowych. Wiedza zdobyta na uczelni została poszerzona o literaturę [4].

Elektromagnes potrzebny do bezdotykowego chwytania figur na planszy szachownicy został zbudowany własnoręcznie przez autora na podstawie informacji zaczerpniętych z odpowiedniego źródła [5].Następnie podłączony do przekaźnika, aby umożliwić przepływ dużego prądu. Natomiast do sterowania elementami elektronicznymi wykorzystano mikrokomputer Raspberry Pi Zero W. Oprócz wymienionych elementów elektronicznych do mikrokomputera podłączono również kilka diod elektroluminescencyjnych. Układ elektroniczny zasilany jest z kilku niezależnych źródeł w zależności od potrzeb.



Rysunek 11. Schemat układu elektronicznego.

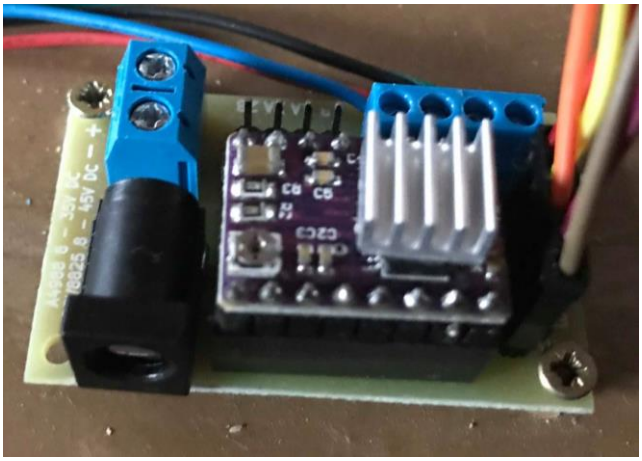
Schemat widoczny na Rysunek 11 przedstawia rozmieszczenie dobranych elementów elektronicznych. Mikrokomputer (Raspberry Pi), jako jednostka centralna, steruje wszystkimi procesami odbywającymi się podczas działania urządzenia. Odbywa się to w sposób bezpośredni lub pośredni.

Silniki elektryczne krokowe JK42HS40-0504

Rysunek 12. Silnik krokowy JK42HS40-0504

Jako jednostka napędzająca użyte zostały dwa silniki krokowe zasilane napięciem stałym 12V. Podczas nominalnej pracy pobór prądu tych silników to 0,5A, generują one moment obrotowy 0,43Nm. Po uwzględnieniu siły potrzebnej do poruszania tym modułem przy pomocy śruby z nakrętką okazało się, że potrzebny moment obrotowy będzie musiał wynosić 0,25Nm. Wybrano te silniki, ponieważ ich moment nominalny jest wystarczający i nie ma konieczności stosowania przekładni, co upraszcza układ mechaniczny.

Moduł sterownika silnika krokowego ze sterownikiem A4988 lub DRV8825

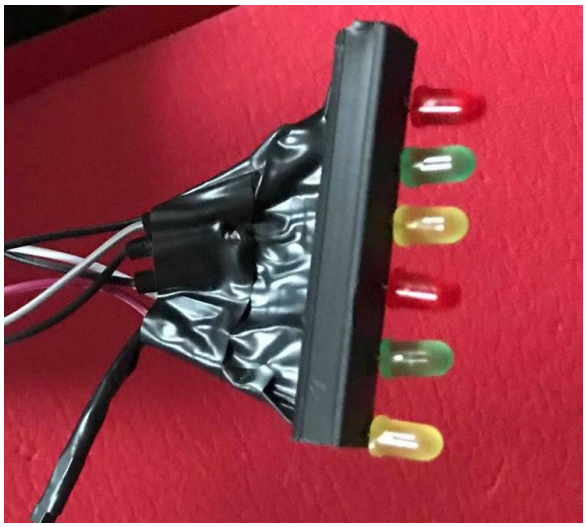


Rysunek 13. Moduł sterownika silnika ze sterownikiem DRV8825

Do sterowania silnikami użyto sterowników A4988 oraz DRV8825. Zakupione zostały wraz z modułem posiadającym wygodne wyprowadzenia oraz pozwalającym w łatwy sposób podłączyć zasilanie silników. Początkowo użyto wyłącznie sterowników A4988, jednak podczas testów uległy awarii, z powodu braku dostępności do jednego z silników. Użyto zamiennika – DRV8825. Sterowniki A4988 oraz DRV8825 mają bardzo zbliżone

parametry, jak i cechują się takim samym sposobem sterowania i połączenia z mikrokontrolerem, więc modyfikacje nie były problemem.

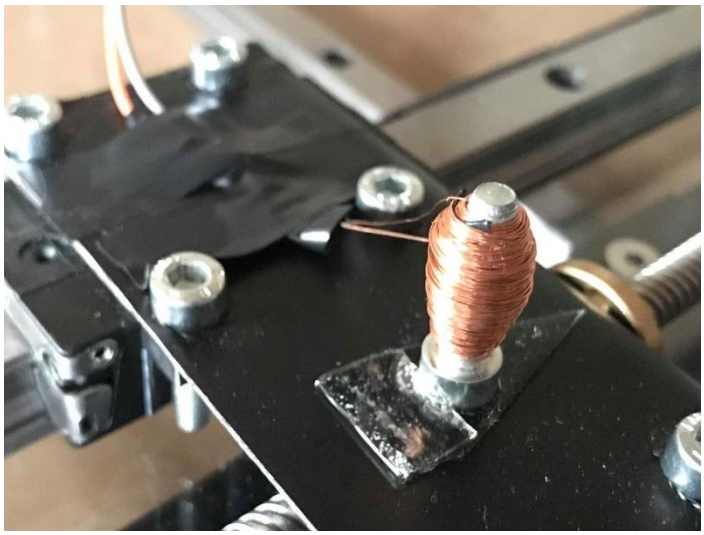
Diody elektroluminescencyjne – LED



Rysunek 14. Układ sześciu diod LED

Podczas testów do monitorowania przebiegu programu bardzo wygodne było użycie kilku diod LED, które zapalały się lub gasły. Autor zdecydował się na pozostawienie układu sześciu diod LED, ponieważ docelowe działanie szachownicy odbywa się bez monitora i użytkownicy (gracze) powinni mieć możliwość otrzymania informacji o błędzie, w celu jego usunięcia lub poprawy rozgrywki.

Elektromagnes



Rysunek 15. Elektromagnes zamontowany w pozycji pracy.

Przestawianie pionów na planszy szachownicy odbywa się przy pomocy elektromagnesu, który w zależności od tego, czy generuje pole magnetyczne, czy nie, chwyta bądź nie pion. Elektromagnes został zbudowany w warunkach domowych. Do jego budowy użyto śruby M5x20, która działa jako rdzeń ferromagnetyczny oraz drutu DNE wyciągniętego

z cewki starego silnika prądu stałego. Podłączony do zasilania 5V DC elektromagnes pobiera około 0.8A prądu.

Przekaźnik

Rysunek 16. Przekaźnik SONGLE z podłączonymi odpowiednimi przewodami.

Sterowanie elektromagnesem odbywa się przy pomocy przekaźnika. Został tak przystosowany przez autora, aby można było za pomocą złącza USB podłączyć zasilanie elektromagnesu ze standardowej ładowarki do telefonu. Sam przekaźnik zasilany jest bezpośrednio z mikrokontrolera napięciem 3,3V.

Wyłącznik, czujnik krańcowy z dźwignią prostą - WK315

Rysunek 17. Dolny moduł translacyjny.

Określanie punktu zerowego dla programu sterującego silnikami odbywa się przy pomocy dwóch czujników krańcowych WK315. Zostały one zamontowane na skrajach dwóch osi w taki sposób, aby przesuwająca się nakrętka w skrajnym położeniu wyłączała czujnik. Dzięki zastosowaniu tych czujników sterowanie przemieszczeniem nakrętek jest powtarzalne

przy każdym ponownym uruchomieniu programu, niezależnie od tego, gdzie zakończył się ruch podczas poprzedniego działania programu.

Zasilanie układu elektronicznego

Rysunek 18. Zasilacz 12V DC, zmodyfikowany.

Zasilanie silników prądem stałym 12V odbywa się za pomocą zasilacza impulsowego, który na wyjściu generuje napięcie 12V o maksymalnym natężeniu prądu 1,0A. Zasilacz został specjalnie zmodyfikowany, aby umożliwić podłączenie obu silników z jednego zasilacza. Wyprowadzone są dwa złącza podłączone równolegle względem siebie. Ponieważ silniki podczas pracy pobierają maksymalnie 0,43A wydajność prądowa zasilacza pozwala na zasilenie dwóch takich silników.

Rysunek 19. Zasilacz 5V DC.

Większość elementów elektronicznych w układzie elektronicznym szachownicy zasilana jest napięciem stałym 5V. Użyto do tego dwóch standardowych zasilaczy do telefonu. Jeden zasilacz o wydajności prądowej 2,1A użyto do zasilania mikrokomputera sterującego. Drugi zasilacz o wydajności prądowej 2,0A użyto do zasilania elektromagnesu.

Mikrokomputer z systemem Raspbian.

Rysunek 20. Raspberry Pi Zero W.

Jednostka sterująca użyta do sterowania szachownicą to Raspberry Pi Zero W. To mikrokomputer, posiadający procesor Broadcom BCM2835 taktowany zegarem 1Ghz. Jest on wyposażony w 512 MB RAM, Wifi, Bluetooth, złącze microUSB, port miniHDMI, slot na kartę pamięci micro SD oraz 40 wyprowadzeń pinów GPIO. Raspberry Pi Zero W obsługuje również podstawowe interfejsy komunikacje, takie jak I2C, SPI, UART. Odpowiednie informacje dotyczące wyprowadzeń mikrokomputera, znajdują się na stronie internetowej poświęconej specyfikacji Raspberry Pi [9].

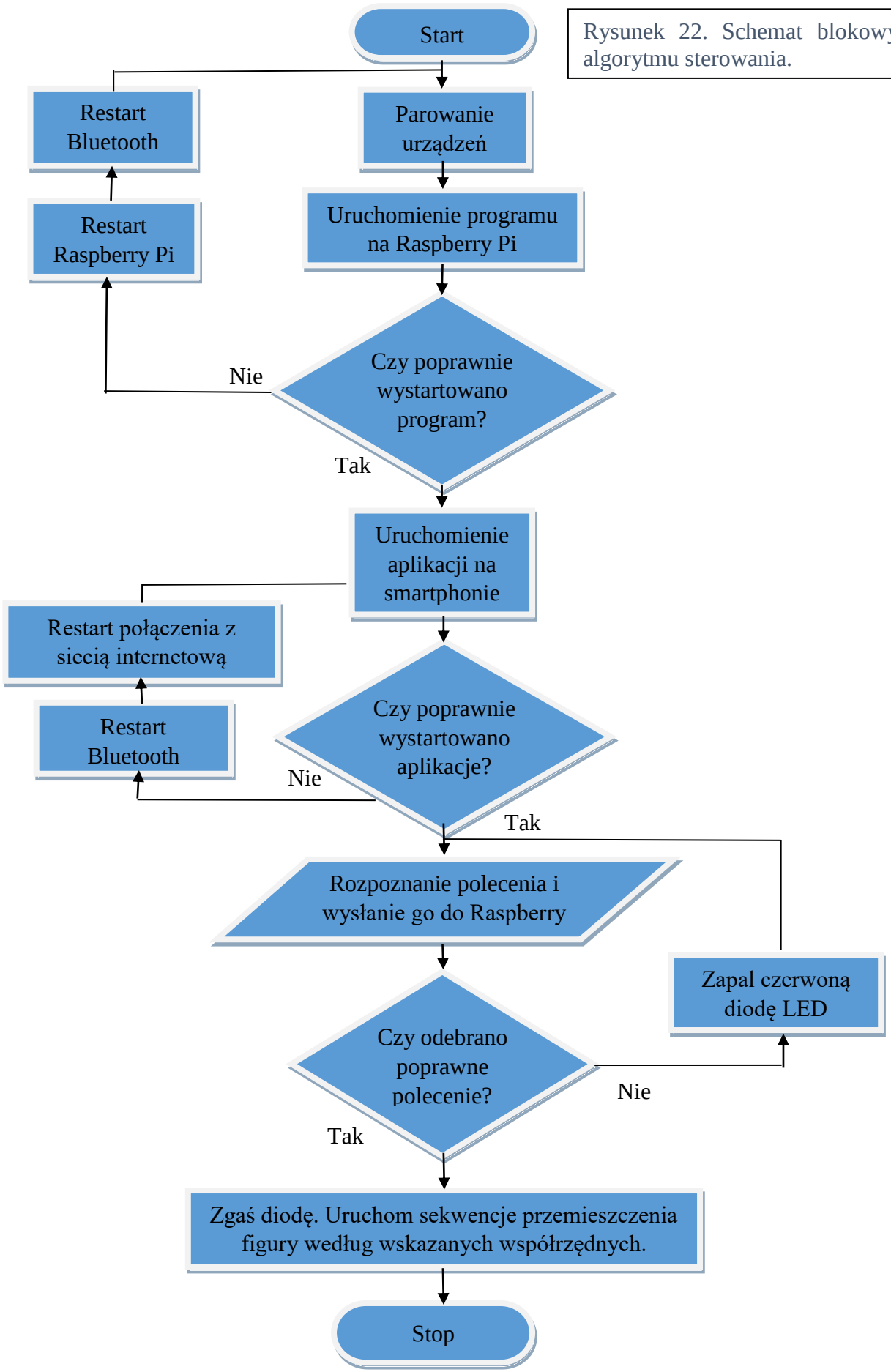
Telefon z systemem android.

Rysunek 21. Telefon z systemem Android.

Sterowanie mikrokomputerem odbywa się zdalnie, za pomocą telefonu z system Android KitKat 4.4 lub nowszym. Telefon używany jest do wydawania poleceń głosowych, które są wysyłane za pośrednictwem Bluetooth'a do Raspberry. Niezbędne jest więc, aby telefon wykorzystywany do sterowania był wyposażony w moduł Bluetooth.

6. Projekt układu sterowania

Opracowano 2 algorytmy sterowania odpowiedzialne za wykonywanie odpowiednich zadań. Jeden algorytm przygotowany przy pomocy poradnika „Adroid Studio"[6]. Pracujący na telefonie odpowiedzialny jest za rozpoznawanie mowy oraz przekazanie wydanego polecenia. Drugi algorytm oparty o publikację Simon'a Monk'a[3], pracujący na mikrokomputerze, ma za zadanie odebrać polecenie. Następnie przetworzyć je na odpowiednie sygnały sterujące sterownikami silników krokowych, przekaźnikiem czy diodami LED. Pierwszy algorytm napisany jest w języku Java, drugi natomiast w języku C. Opracowano uproszczony schemat blokowy algorytmu sterowania.



Rysunek 22. Schemat blokowy algorytmu sterowania.

6.1. Aplikacja dla systemu Android

Wydawanie poleceń szachownicy odbywa się za pomocą telefonu. W tym celu została opracowana aplikacja dla systemu Android 4.4 KitKat lub nowszego. Aplikacja umożliwia rozpoznanie głosowe komend oraz komunikację z Raspberry Pi za pośrednictwem Bluetooth'a. Aby możliwe było rozpoznawanie mowy oraz komunikacja Bluetooth, skorzystano z odpowiednich bibliotek oraz klas udostępnianych na stronach deweloperskich systemu android.

Rysunek 23. Podgląd aplikacji otwartej na telefonie.

Po środku znajduje się pomarańczowy przycisk, za pomocą którego uruchamiamy proces nagrywania naszej komendy. Rozpoznana mowa zostaje wyświetlona zamiast TextView po zakończeniu procesu nagrywania. U dołu ekranu aplikacji znajduje się przycisk POWTÓRZ, który służy do wymuszenia na szachownicy wykonania ostatniej podanej komendy jeszcze raz. Podczas wykonywania aplikacji korzystano z przykładów i schematów podobnych aplikacji [8] [12].

```
private BluetoothAdapter mBluetoothAdapter;
private BluetoothDevice mBluetoothDevice;
private BluetoothSocket mBluetoothSocket;
final Handler handler = new Handler();
private ConnectedThread mConnectedThread;

private String bluetooth_macadress = "B8:27:EB:1C:E4:2F";
```

Rysunek 24. Fragment kodu aplikacji, odpowiedzialny za komunikację Bluetooth.

Powyższy fragment kodu odpowiedzialny jest za deklaracje zmiennych „BluetoothAdapter", „BluetoothDevice" i „BluetoothSocket". „ConnectedThreaed" to klasa komunikacyjna Bluetooth. Użytkownikowi aplikacji nie jest udostępniona możliwość zmiany urządzenia, z którym komunikujemy się na potrzeby aplikacji. Adres MAC ustalany jest na sztywno. Bezpośrednio w kodzie aplikacji odpowiedzialna jest za to stała „bluetooth_macadress".

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txtOutput = (TextView) findViewById(R.id.txt_output);
    btnMicrophone = (ImageButton) findViewById(R.id.btn_mic);
    btnMicrophone.setOnClickListener((v) → { startSpeechToText(); });

    Button button = (Button) findViewById(R.id.button);
    button.setOnClickListener((v) → { SendGeneratedCode(); });

    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();


    connect();


}
```

Rysunek 25. Fragment kodu przedstawiający klasę onCreate.

Metoda „onCreate" jest wykonywana podczas pierwszego otwarcia aplikacji na telefonie. Jest ona odpowiedzialna za wywołanie podstawowych metod. Inicjalizujemy połączenie Bluetooth z urządzeniem o adresie MAC podanym w klasie „BluetoothAdapter". Wywoływane są też klasy odpowiedzialne za funkcjonalność przycisków na ekranie aplikacji.

```
public void connect() {
    //if (mBluetoothAdapter.isEnabled()) {
        mBluetoothDevice = mBluetoothAdapter.getRemoteDevice(bluetooth_macadress);
        showToast( msg: "Connected");
        btConnect();
    //}
}
```

Rysunek 26. Fragment kodu metody deklaracji połączenia Bluetooth

Metoda „connect” odpowiedzialna jest za deklarację połączenia Bluetooth z konkretnym urządzeniem. Adres MAC konkretnego urządzenia podawany jest przez stałą „bluetooth_macadress”.

```
private void btConnect()    {
    if(mBluetoothDevice == null)
        return;
    try{
        mBluetoothSocket = mBluetoothDevice.createRfcommSocketToServiceRecord(uuid);
        mBluetoothSocket.connect();

        mConnectedThread = new ConnectedThread(mBluetoothSocket, b: this);
        mConnectedThread.start();

    }
    catch(IOException e)
    {
       btDisconnect();
    }
}

public void btDisconnect()
{
    if(mBluetoothSocket == null)
        return;
    if(mConnectedThread != null)
    {
        mConnectedThread = null;
    }
    try{
        mBluetoothSocket.close();
    }
    catch(IOException e){
    }
    mBluetoothSocket = null;

}
```

Rysunek 27. Fragment kodu przedstawiający metody odpowiedzialne za połącznie z socketem.

Metoda „btConnect” odpowiedzialna jest za połączenie z „socket”. W przypadku nieudanej próby wywoływana jest metoda „btDisconnect”, która zeruje wszystkie zmienne.

```
//////////////////////////////      Bluetooth   //////////////////////////////
private class ConnectedThread extends Thread {

    private InputStream mmInStream = null;
    private OutputStream mmOutStream = null;
    //private Vector blue= new Vector();


    public ConnectedThread(BluetoothSocket socket, MainActivity b) {
        try {
            mmInStream = socket.getInputStream();
            mmOutStream = socket.getOutputStream();

        } catch (IOException e) {
        }
    }


     public void write(int one) {
        try {
            mmOutStream.write(one);
            showToast( msg: "x1");
        } catch (IOException e) {
            showToast( msg: "x2");
        }

     }

    public void write2(String one) {
        try {
            mmOutStream.write(one.getBytes());

            showToast( msg: "x1");
        } catch (IOException e) {
            showToast( msg: "x2");
        }

    }
```

Rysunek 28. Fragment kodu klasy odpowiedzialnej za komunikacje Bluetooth.

Klasa „ConnectThread” odpowiedzialna jest za komunikację Bluetooth. Metoda „ConnectThread” otwiera strumień komunikacyjny. Metody „write” oraz „write2” wyświetlają krótkie wiadomości na ekranie aplikacji, co umożliwia użytkownikowi testy komunikacji Bluetooth.

```
btnMicrophone = (ImageButton) findViewById(R.id.btn_mic);
btnMicrophone.setOnClickListener((v) → { startSpeechToText(); });
```

Rysunek 29. Fragment kodu metody OnCreate.

W metodzie „OnCreate” dochodzi do deklaracji zmiennej podpiętej do „widgeta” „button” o nazwie „btn_mic” oraz podpięcie metody „startSpeechToText” pod pomarańczowy przycisk.

```
private void startSpeechToText() {
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
            value: "Speak something...");
    try {
        startActivityForResult(intent, SPEECH_RECOGNITION_CODE);
    } catch (ActivityNotFoundException a) {
        Toast.makeText(getApplicationContext(),
                text: "Sorry! Speech recognition is not supported in this device.",
                Toast.LENGTH_SHORT).show();
    }
}
```

Rysunek 30. Fragment kodu metody startSpeechToText.

Metoda „startSpeechToText” odpowiedzialna jest rozpoczęcie nagrywania dźwięku i rozpoznawania mowy. Po wciśnięciu odpowiedniego przycisku, wyświetlany jest komunikat w nowym oknie, w języku angielskim „Speak something…”. Jest to informacja dla użytkownika, że należy rozpocząć wydawanie komendy.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
        case SPEECH_RECOGNITION_CODE: {
            if (resultCode == RESULT_OK && null != data) {
                ArrayList<String> result = data
                        .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
                String text = result.get(0);
                txtOutput.setText(text);
                if(mConnectedThread != null){
                    mConnectedThread.write2(text);
                    //mConnectedThread.write(p);
                    //   showToast("xD");
                }
            }
            break;
        }
    }
}
```

Rysunek 31. Fragment kodu metody onActivityResult

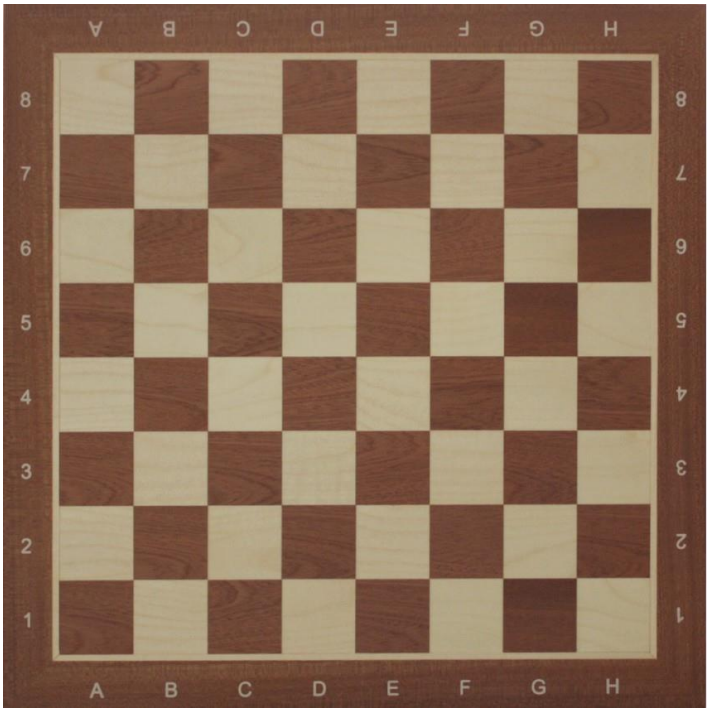
Metoda „onActivityResult” przesyła wynik rozpoznanej mowy przez biblioteki Google.

6.2. Komunikacja Bluetooth z poziomu Raspberry

Do umożliwienia komunikacji Bluetooth konieczne jest sparowanie urządzeń. Aby możliwa była wymiana danych pomiędzy mikrokomputerem, a telefonem należy odpowiednio przygotować urządzenie. Wiedzę zaczerpnięto z odpowiednich portali internetowych [10] [11].

W tym celu należy uruchomić okno terminala w systemie Raspbian. Poleceniem „hcitool scan" wyszukujemy dostępne urządzenia, aby w następnej kolejności znaleźć na liście dostępnych urządzeń nasz telefon. Następnie należy sparować się z telefonem, używając do tego wyświetlony w oknie adres MAC. Telefon musi zostać ustawiony jako zaufany. Znając adres MAC możemy tego dokonać poleceniem „trust". Komendy odebrane z telefonu zapisywane są w pliku tekstowym „/dev/rfcomm0". Plik ten jest tymczasowym plikiem, tworzonym podczas podłączania jakiegoś urządzenia na danym kanale Bluetooth. Przed rozpoczęciem korzystania z tego pliku moduł Bluetooth musi być przestawiony w tryb nasłuchiwania. Można tego dokonać poleceniem z poziomu terminala „sudo rfcomm listen rfcomm0". Po połączeniu się telefonu z Raspberry można już przesyłać dane.

Napisano odpowiedni program w języku C, który umożliwia sterowanie szachownicą po odczytaniu komend z pliku rfcomm0. Przykładowe polecenie wydawane przez użytkownika powinno składać się z podania dwóch współrzędnych w układzie współrzędnych szachownicy. Każde pole na szachownicy opisane jest odpowiednią kombinacją jednej litery (A,B,C,D,E,F,G,H) i jednej cyfry (1,2,3,4,5,6,7,8). Użytkownik podaje pierwszą współrzędną, określając z jakiej pozycji chce przemieścić pion, a następnie na jakie pole chce ten pion przemieścić. Przykładowe polecenia mogą więc wyglądać następująco: „A2 na A3", „E3 na B6", „C4 na G4". Program został przygotowany do operowania na komendach w dokładnie takiej formie. Nie ma możliwości wydania polecenia ruchu w inny sposób.



Rysunek 32. Przykładowe pole szachownicy z opisanymi osiami współrzędnych.

```
void Bluetooth(int tab[]){

    digitalWrite(22,1);
        FILE* f = fopen("/dev/rfcomm0", "rw");
    if (f==NULL) {
        perror("fopen");
        exit(1);
    }
    char pozycja[5];
    char wsp[9];

    fgets(wsp, sizeof(wsp), f);
    if (wsp != NULL) digitalWrite(22,0);
    printf("Polecenie: %s \n", wsp);
    digitalWrite(21,0);
    pozycja[0]=wsp[0];
    pozycja[1]=wsp[1];
    pozycja[2]=wsp[6];
    pozycja[3]=wsp[7];
    printf("pozycja: %s \n", pozycja);
```

Rysunek 33. Fragment kodu odpowiedzialny za przetworzenie danych odebranych za pośrednictwem Bluetooth.

Funkcja „Bluetooth” za argument przyjmuje tablicę globalną „tab[]” czterech znaków typu „char”. Po otwarciu pliku „/dev/rfcomm0” i sprawdzeniu poprawności operacji za pomocą funkcji „fgets” zostaje pobrana linia zapisana w pliku. Komenda zostaje zapisana do zmiennej „wsp[]” typu „char”. Polecenie składa się z 4 znaków określających współrzędne, 2 spacji oraz wyrazu, na co łącznie daje 8 znaków. Każda linia zakończona jest znakiem końca linii, stąd ilość miejsca przydzielona zmiennej „wsp” jest równa 9. Operowanie na takiej zmiennej byłoby niewygodne, zastosowano więc dodatkową zmienną pomocniczą „pozycja[]”, która uwzględnia już tylko znaki odpowiedzialne za określenie

współrzędnych. Kolejnym etapem realizowanym przez funkcję „Bluetooth” jest zapisanie w tablicy globalnej „tab[]” parametrów liczbowych, które w późniejszych etapach działania programu będą odpowiedzialne za sterowanie przemieszczeniem elektromagnesu.

```
//odczytanie pierwszej litery//
if (pozycja[0]=='A'){
    tab[0]=7;
}else if(pozycja[0]=='B'){
    tab[0]=6;
}else if(pozycja[0]=='C'){
    tab[0]=5;
}else if(pozycja[0]=='D'){
    tab[0]=4;
}else if(pozycja[0]=='E'){
    tab[0]=3;
}else if(pozycja[0]=='F'){
    tab[0]=2;
}else if(pozycja[0]=='G'){
    tab[0]=1;
}else if(pozycja[0]=='H'){
    tab[0]=0;
}else {
    printf("pozycja0: blad \n");
}
```

Rysunek 34. Fragment kodu prezentujący realizację przekazania komend do tablicy globalnej "tab[]"

Przed przystąpieniem do sterowania silnikami, elektromagnesem czy diodami należy odpowiednio ustalić, które porty będą działały jako wejścia, a które jako wyjścia oraz ustalić ich stany początkowe. Do sterowania portami GPIO wykorzystano bibliotekę „wiringPi”. Umożliwia ona sterowanie portami w bardzo prosty sposób. Należy jednak mieć na uwadze, że biblioteka ta wprowadza inną numeracje portów niż sprzętowa(domyślna).

```
void Pinout(void){

    //Ustalenie parametrów portów GPIO //
    pinMode(1, OUTPUT); //Sterownik Y Step
    pinMode(2, OUTPUT); //Sterownik Y Dir
    pinMode(16, OUTPUT); //Sterownik Y Enable
    pinMode(3, OUTPUT); //Sterownik X Step
    pinMode(4, OUTPUT); //Sterownik X Dir
    pinMode(0, OUTPUT); //Sterownik X Enable
    pinMode(25, OUTPUT); //elektormagnes
    pinMode(12, OUTPUT); //elektormagnes
    pinMode(26, OUTPUT); //LED
    pinMode(22, OUTPUT); //LED
    pinMode(21, OUTPUT); //LED
    pinMode(30, OUTPUT); //LED
    pinMode(31, OUTPUT); //LED
    pinMode(11, OUTPUT); //LED
    pinMode(28, OUTPUT); //Czujnik Y 5V
    pinMode(24, OUTPUT); //Czujnik X 5V
    pinMode(29, INPUT); //Czujnik Y on/off
    pinMode(23, INPUT); //Czujnik X on/off
    digitalWrite(25,0);
    digitalWrite(12,1);
```

Rysunek 35. Fragment kodu funkcji odpowiedzialnej za ustalenia stanów początkowych na pinach GPIO.

Funkcja „Pinout” oprócz ustalenia, które porty będą działały jako wejścia, a które jako wyjścia, zadaje wartości początkowe odpowiednie do ich przeznaczenia. Funkcja ta powoduje zgaszenie wszystkich diod LED, blokadę silników, uruchomienie przekaźnika elektromagnesu oraz uruchomienie czujników. Jej wywołanie w programie jest konieczne do prawidłowego działania programu. Sterowanie silnikami odbywa się pośrednio przez sterownik, który umożliwia wybór kierunku obrotu wału silnika, blokadę silnika, a także włączenie lub nie obrotów silnika z zadaną prędkością obrotową w postaci sygnału PWM. Aby móc generować sygnał PWM wykorzystano bibliotekę „softPwm”. Pozwala ona korzystać z kilku przydatnych funkcji, niezbędnych do poprawnego sterowania silnikami.

```
// Najazd na pole H1 //
digitalWrite(4,1);
softPwmCreate(3,10,20);
softPwmWrite(3,10);
delay(300);
softPwmWrite(3,0);

digitalWrite(2,1);
softPwmCreate(1,10,20);
softPwmWrite(1,10);
delay(600);
softPwmWrite(1,0);
```

Rysunek 36. Fragment kodu prezentujący funkcje potrzebne do wygenerowania sygnału PWM.

Silniki krokowe użyte w pracy nie posiadają en kodera. Ilość obrotów nie jest więc w żaden sposób zliczana. Sterowanie silnikami odbywa się za pomocą funkcji „delay();”, która tworzy w programie opóźnienie czasowe. Użyto tej funkcji do określania czasu działania silnika, co odpowiada przemieszczeniu elektromagnesu wzdłuż osi, do której zamontowany jest dany silnik. W celu umożliwienia powtarzalnej pracy programu sterującego, bez względu na to jak przemieszczony został elektromagnes przed awarią lub przypadkowym zamknięciem programu, zastosowano czujniki krańcowe. Odpowiednia funkcja odpowiedzialna jest za bazowanie, tzn. umieszczenie elektromagnesu w pozycji początkowej w sposób powtarzalny.

```
while(1){
    x = digitalRead (23);
    if (x<1){
        printf("X %d \n", x);
        delay(100);
    } else if (x>0){
        printf("X %d \n", x);
        softPwmWrite(3,0);
        digitalWrite(26,0); // LED X off
        break;
    }
}

digitalWrite(2,0);
softPwmCreate(1,10,15);
softPwmWrite(1,10);

while(1){
    y = digitalRead (29);
    if (y<1){
        printf("Y %d \n", y);
        delay(100);
    } else if (y>0){
        printf("Y %d \n", y);
        softPwmWrite(1,0);
        digitalWrite(31,0); // LED Y off
        break;
    }
}
```

Rysunek 37. Fragment kodu funkcji odpowiedzialnej za bazowanie.

Ze względu na budowę modułu mechanicznego szachownicy oraz minimalizację czasu potrzebnego do bazowania, pozycja bazowa została określona jako pole H1, które znajduje się w prawym dolnym rogu pola szachownicy. Określono czas potrzebny na pokonanie odległości równej szerokości jednego pola. To właśnie od pola H1 zaczyna się ruch do każdego pola, z którego pobierany jest pion. Za ruch elektromagnesu odpowiedzialne są funkcje „Ruch1 oraz „Ruch2".

```
int Ruch1(int L1, int C1){

int Ruch2(int L1, int C1, int L2, int C2){
```

Rysunek 38. Fragment kodu przedstawiający argumenty wejściowe funkcji.

Funkcja „Ruch1" za argumenty przyjmuje pierwszą współrzędną literę i cyfrę. Funkcja ta realizuje przemieszczenie elektromagnesu na pole o zadanej współrzędnej. W zależności od liczb przekazanych ze zmiennej „pozycja[]", elektromagnes przemieszcza się o odpowiednią wielokrotność szerokości pola w każdej z osi.

Funkcja „Ruch2" za argumenty przyjmuje wszystkie współrzędne. Jest to konieczne do wyznaczenia ruchu z innej pozycji niż bazowa. Funkcja odejmuje współrzędne L1, C1 od współrzędnych L2, C2. W ten sposób otrzymujemy informację o ile pól musimy się z tego miejsca przemieścić, aby dostać się na pole podane współrzędnymi L2, C2. Funkcja sprawdza znak po operacji odejmowania i ustawia odpowiedni kierunek obrotu silników. Do sterowania silnikami używane są wartości dodatnie. W celu sprowadzenia wyniku do wartości bezwzględnych użyto polecenia abs();.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wiringPi.h>
#include <softPwm.h>

int main (int argc, char **argv)
{
    printf("*** Program Sterowania Szachownicy *** \n");

    if (wiringPiSetup() == -1)  {printf("Nie mozna wystartowac wiringPi \n"); return 1;}

    void Pinout(void);
    void Bazowanie(void);
    void Bluetooth(int tab[4]);
    int Ruch1(int L1, int C1);
    int Ruch2(int L1, int C1, int L2, int C2);
    int pozycje[4];

    Pinout();
    Bazowanie();
    while (1){ //rozgrywka//
    digitalWrite(11,1);
    Bluetooth(pozycje);
    printf("Pozycje: %d \n", pozycje[0]);
    printf("Pozycje: %d \n", pozycje[1]);
    printf("Pozycje: %d \n", pozycje[2]);
    printf("Pozycje: %d \n", pozycje[3]);
    Ruch1(pozycje[0], pozycje[1]);
    digitalWrite(25,1);
    delay(500);
    Ruch2(pozycje[0], pozycje[1], pozycje[2], pozycje[3]);
    digitalWrite(25,0);
    digitalWrite(11,0);
    Bazowanie();
    }
```

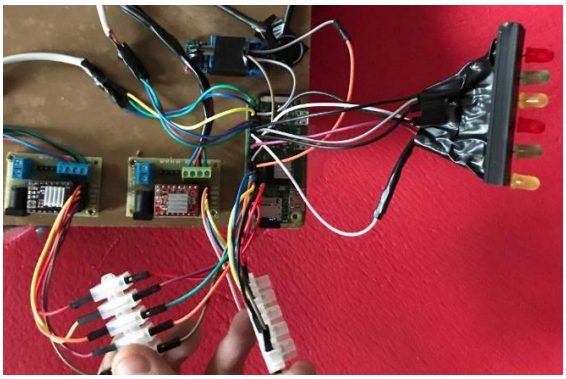
Rysunek 39. Fragment kodu funkcj main.

Główna funkcja programu (main) informuje użytkownika o rozpoczęciu programu oraz o ewentualnym błędzie w przypadku błędnej inicjalizacji biblioteki wiringPi na której opierają się wszystkie funkcje. Wywoływane w odpowiedniej kolejności są tutaj wszystkie funkcje programu. Użytkownik ma podgląd na przekazywane wartości w zmiennej „pozycje[]”, jest to niezbędna właściwość przy wykonywaniu testów. Po wykonaniu funkcji początkowych w pętli „while” wykonywany jest program, dzięki czemu po każdym wydaniu komendy i przemieszczeniu figury wystarczy odczekać kilka sekund, aż układ wróci do pozycji bazowej i można wydawać kolejne polecenia.

7. Wyniki pracy

W tym rozdziale przedstawione zostaną wyniki pracy. Wszystkie zdjęcia przedstawiają rzeczywiste urządzenie, spełniające wymogi postawione na początku pracy. Należy zaznaczyć, że zbudowana szachownica jest interfejsem do gry w szachy. Urządzenie nie kontroluje poprawności ruchów, użytkownicy muszą znać zasady gry i ich przestrzegać, aby urządzenie działało poprawnie. Układ elektroniczny został zabezpieczony skromną obudową, aby uniemożliwić ingerencje przeciętnemu użytkownikowi.

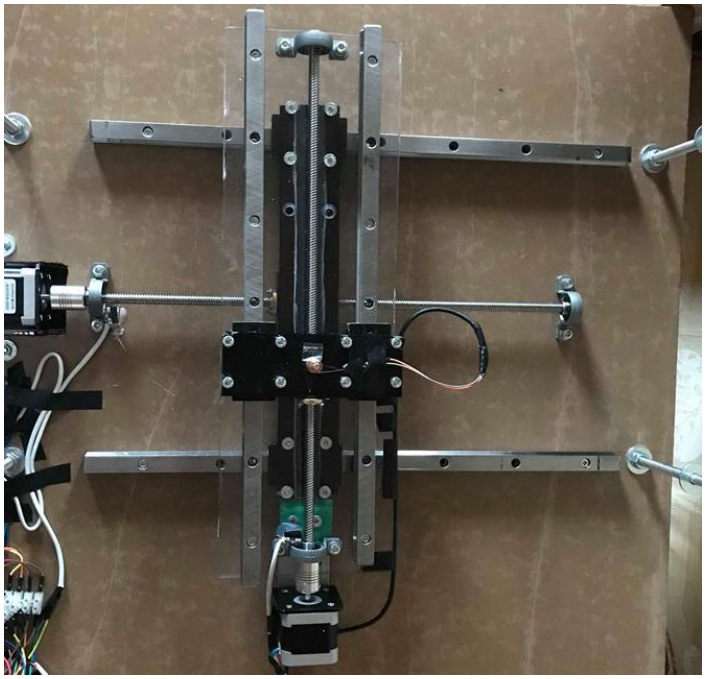
7.1. Połączone wszystkie urządzenia i elementy elektroniczne



Rysunek 40. Mikrokomputer i jego peryferia

Urządzenia peryferyjne do mikrokomputera podłączone zostały za pomocą przewodów z końcówkami żeńsko-męskimi. Wszystkie przewody zostały ze sobą skręcone lub zlutowane oraz zaizolowane taśmą izolacyjną. Wszystkie elementy zostały przykręcone za pomocą wkrętów do płyty głównej lub przyklejone za pomocą klejenia na gorąco w odpowiednich do swojego przeznaczenia miejscach.

7.2. Gotowy układ mechaniczny



Rysunek 41. Gotowy układ mechaniczny.

Wszystkie elementy zostały połączone przy pomocy śrub oraz nakrętek z podkładkami. Prowadnice liniowe przykręcone są za pomocą śrub M4. Uchwyty z łożyskami mocujące śrubę Tr8x8 zostały przykręcone do płyt za pomocą śrub M5.

7.3. Gotowa szachownica

Rysunek 42. Gotowa szachownica przygotowana na rozgrywkę

Płyta górna hdf jest przymocowana do płyty głównej dolnej przy pomocy pręta gwintowanego pociętego na 4 pręty o długości 140mm. Każdy z nich jest przykręcany do płyty głównej na 2 nakrętki oraz w ten sam sposób do płyty górnej. Dzięki zastosowaniu dwóch nakrętek do mocowania płyty górnej hdf mamy możliwość odpowiedniego dostosowania wysokości, na której znajduje się płyta dolnymi nakrętkami, a następnie skręcenie wszystkiego na sztywno następnymi nakrętkami od góry. Do rozgrywki należy używać pionów, które u podstawy mają przyklejony drobny element metalowy lub magnes.

Rysunek 43. Przykładowy pion używany do rozgrywki na szachownicy

8. Podsumowanie i wnioski

Podczas budowy szachownicy autor pracy napotkał kilka znaczących problemów. Pierwsze napotkane problemy zaczęły się już na etapie montażu. Okazało się, że dostępne wózki są przystosowane do działania pod wpływem obciążenia wstępnego. Ręczne przesuwanie wózków po szynie wymagało użycia sporej siły. Dopiero przykładając obciążenie w postaci masy 20kg można było przesuwać wózki bez zauważalnych oporów. Aby pozbyć się problemu usunięto uszczelki zapobiegające dostawaniu się zanieczyszczeń w strefę łożyskową. Zeszlifowano też powierzchnie boczne szyn na frezarce numerycznej, zwiększyć luzy między szyną a wózkiem.

Początkowo planowano użyć płyty hdf o grubości 3mm jako pola szachownicy. Jednak po przygotowaniu elektromagnesu okazało się, że jest on za słaby, gdy jest zasilany przez 5V. Pierwsza próba wyeliminowania problemu to zwiększenie zasilania do 12V. Niestety w tym przypadku cewka elektromagnesu grzała się do bardzo wysokich temperatur, a elementy niemetalowe w otoczeniu cewki zaczęły się topić. Zrezygnowano więc ze zwiększania mocy cewki i wykorzystano matę ochronną pod fotel, która ma grubość 6 razy mniejszą niż płyta hdf. To rozwiązanie pozwoliło pozbyć się destrukcyjnego problemu.

Elektromagnes przy zasilaniu 5V pobiera prąd o wartości około 0.8A. Sterowanie bezpośrednio z portów GPIO Raspberry Pi jest więc niemożliwe, ponieważ z tych portów można pobierać maksymalnie prąd o wartości 30mA. Zakupiony został tranzystor Mosfet, który miał służyć jako przełącznik. Podłączono jeden z przewodów zasilających elektromagnes do źródła oraz drenu. Do bramki natomiast podłączono przewód połączony z wyjściem logicznym na Raspberry Pi. Sterowanie w ten sposób nie przyniosło sukcesów. Pole magnetyczne nie było stabilne, elementy metalowe w pobliżu cewki drżały. Aby wyeliminować problem posłużono się przekaźnikiem opisanym w punkcie 5.5.

Podczas próby sterowania silnikami krokowymi napotkano problem w postaci niekontrolowanych obrotów silnika. Do sterownika silnika krokowego podłączono zasilanie bezpośrednio z Raspberry Pi oraz wyjścia logiczne do złączy Step oraz Dir. Złącze Enable pozostało niepodłączone. Silnik działał tylko wtedy, gdy podano sygnał PWM na złącze Step, a w pobliżu sterownika znajdowała się ręka autora pracy. Sytuacja była powtarzalna niezależnie od miejsca, w którym odbywały się testy. Dopiero po podaniu konkretnej wartości logicznej na złącze Enable problem został wyeliminowany. Złącze enable odpowiedzialne jest za włączenie lub wyłączenie sterownika. Najprawdopodobniej, kiedy złącze nie było podłączone „zbierało" zakłócenia pola elektromagnetycznego z przestrzeni, a autor pracy swoją ręką ingerował w pole, co powodowało zmianę wartości logicznej na tym złączu.

Największe problemy napotkano podczas próby stworzenia poprawnej komunikacji Bluetooth między telefonem, a mikrokomputerem. Do testów aplikacji wykorzystano program Hercules SETUP utility. Bez problemów wysyłano komendy z telefonu na komputer PC oraz odbierano komendy wysłane za pomocą programu z komputera PC na Raspberry Pi. Jednak próby połącznia telefonu z Raspberry kończyły się porażką i komunikatem „no usable services are available for the raspberry pi". System Raspberry Pi najprawdopodobniej nie pozwalał utworzyć połączenia Bluetooth, ponieważ nie znajdował możliwości jego wykorzystania. Po długich bataliach i próbach połączenia na różne sposoby, rozwiązaniem okazała się modyfikacja kilku plików systemowych. Wymuszono utrzymywanie modułu Bluetooth w trybie nasłuchiwania. Konieczne również było dodanie adresu MAC telefonu do

listy urządzeń, które przy starcie modułu Bluetooth mają być automatycznie przełączane w tryb połączono. Oprócz tego, aby algorytm sterowania mógł poprawnie działać, należy z poziomu terminala w oddzielnym oknie wpisać polecenie „sudo rfcomm listen hci0” i po jego wywołaniu przez cały czas okno musi być otwarte i działać w tle.

Po wyeliminowaniu wszystkich problematycznych niezgodności i ukończeniu szachownicy przystąpiono do pierwszych testów. Na grupie kontrolnej składającej się z 4 osób, dwóch mężczyzn i dwóch kobiet, wykonano test aplikacji rozpoznawania mowy. Każda z osób musiała wydać 10 różnych komend oraz wypowiedzieć 5 dowolnych słów. Autor pracy sprawdzał i porównywał wypowiedziane przez grupę kontrolną słowa z tym, co zostało wyświetlone na ekranie aplikacji. Skuteczność wynosi około 90% ponieważ raz na 10 prób aplikacja błędnie rozpoznała mowę.

Przeprowadzono również test sprawności algorytmu sterowania szachownicą. Wydając polecenia przez telefon dwóch graczy, w tym autor pracy, rozegrali krótką partię szachów. Po przystosowaniu się do bardzo wyraźnego wymawiania komend rozgrywka przebiegała bez większych problemów. Szachownica nie została jednak wyposażona w system pozbywania się wyeliminowanych z gry figur. Tą czynność użytkownicy wykonywali samodzielnie, odkładając wyeliminowane z gry figury. Kolejną niedogodnością była potrzeba resetowania połączenia Bluetooth po około 15 ruchach. Raspberry Pi z niewiadomych przyczyn po kilkunastu minutach zrywa połączenie Bluetooth. Jednak, ponieważ cała procedura łączenia się urządzeń została zautomatyzowana, proces restartowania przebiegał sprawnie.

Część napotkanych problemów można by wyeliminować projektując szachownicę od podstaw. I tak, np. zamiast szyn i wózków skorzystać z wałków i suwających się po nich tulei łożyskowych, co spowodowało by zmniejszenie siły potrzebnej na wygenerowanie ruchu. Zespół śruby i nakrętki zastąpić zespołem koła zębatego i pasa klinowego, co ułatwiło by montaż elementów. Zamiast sterować silnikami na podstawie odległości przebytej w danym czasie, lepiej byłoby zaopatrzyć się w enkoder lub odpowiednią ilość czujników rozmieszczonych tak, by monitorować położenie przy każdym z pól.

Próba stworzenia urządzenia na podstawie dostępnych elementów wygenerowała problemy. Zaoszczędzono w ten sposób dostępne środki finansowe, jednak poświęcono sporo czasu na przystosowanie dostępnych elementów do potrzeb. Jest to niewątpliwie pouczająca lekcja dla autora pracy.

9. Bilbiografia

[1] Poradnik Mechatronika, Rea, 2015

[2] Prof. dr. hab. inż. Joachima Potrykusa, Poradnik Mechanika, Rea, 2014

[3] Simon Monk, Zrób to sam, generowanie ruchu, światła i dźwięku za pomocą arduino i raspberry pi, Helion, 2017

[4] Dobrowolski A. Jachna z. Maj, Elektronika alez to bardzo proste, Wydawnictwo BTC, 2013

[5] John Watson, Elektronika, WKiŁ, 2004

[6] Marcin Płonkowski, Android Studio. Tworzenie aplikacji mobilnych, Helion, 2017

[7] Hercules SETUP utility https://www.hw-group.com/products/hercules/index_en.html

[8] Android Speech https://developer.android.com/reference/android/speech/package-summary.html

[9] Raspberry Pi Pinout https://pinout.xyz/

[10] GPIO Interface library for the Raspberry Pi http://wiringpi.com/

[11] Kurs Raspberry Pi od podstaw https://forbot.pl/blog/kurs-raspberry-pi-od-podstaw-wstep-spis-tresci-id23139

[12] Harry's Developer Blog https://wingoodharry.wordpress.com/2014/04/15/android-sendreceive-data-with-arduino-using-bluetooth-part-2/

[13] Develop API Guides Connectivity https://developer.android.com/guide/topics/connectivity/bluetooth.html