1.1. 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 1. 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.

```
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 2. 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 3. 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 4. 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 5. 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 6. 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 7. 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 8. 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 9. Fragment kodu metody onActivityResult

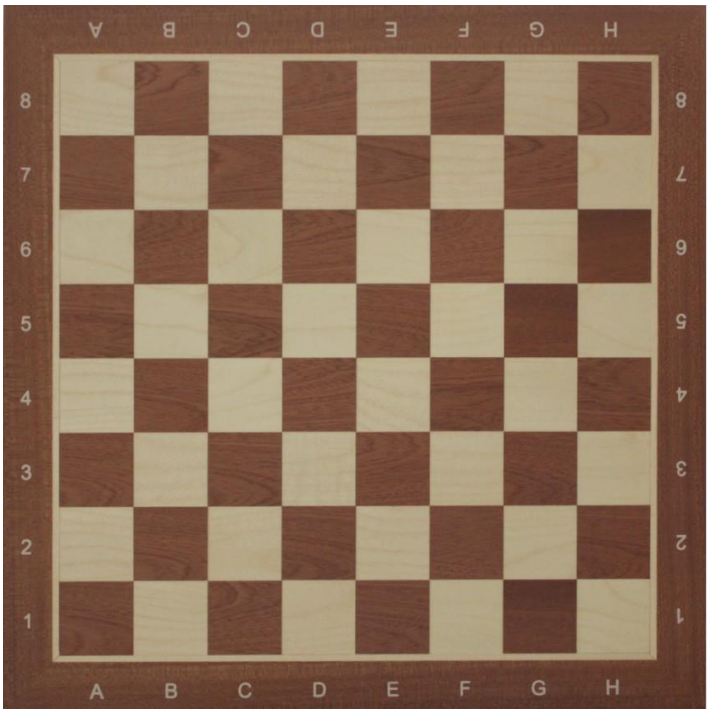
Metoda „onActivityResult” przesyła wynik rozpoznanej mowy przez biblioteki Google.

1.2. 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.

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.

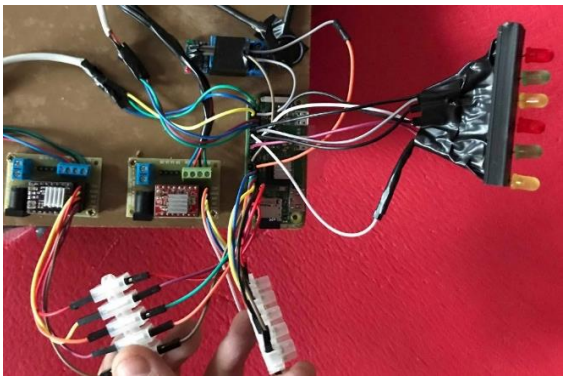


Rysunek 10. Przykładowe pole szachownicy z opisanymi osiami współrzędnych.

1. 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.

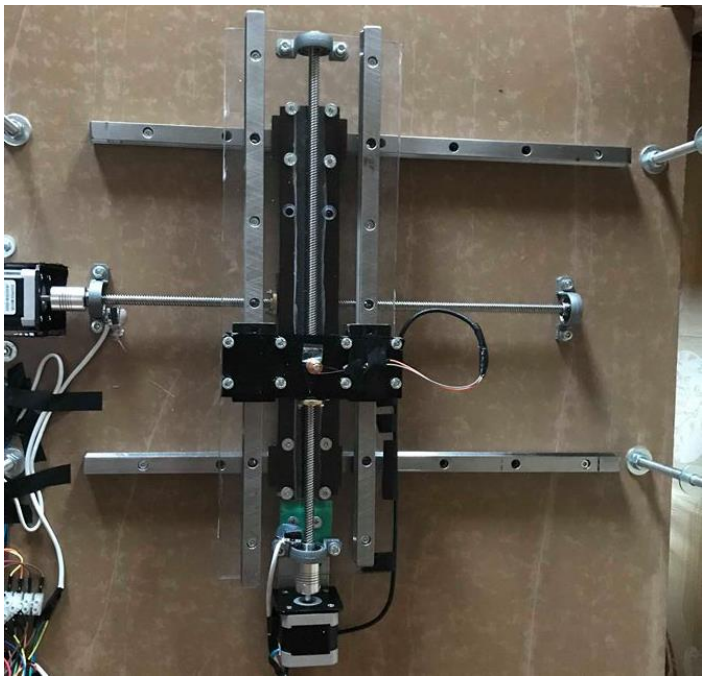
2.1. Połączone wszystkie urządzenia i elementy elektroniczne



Rysunek 11. 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.

2.2. Gotowy układ mechaniczny



Rysunek 12. 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.

2.3. Gotowa szachownica

Rysunek 13. 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 14. Przykładowy pion używany do rozgrywki na szachownicy