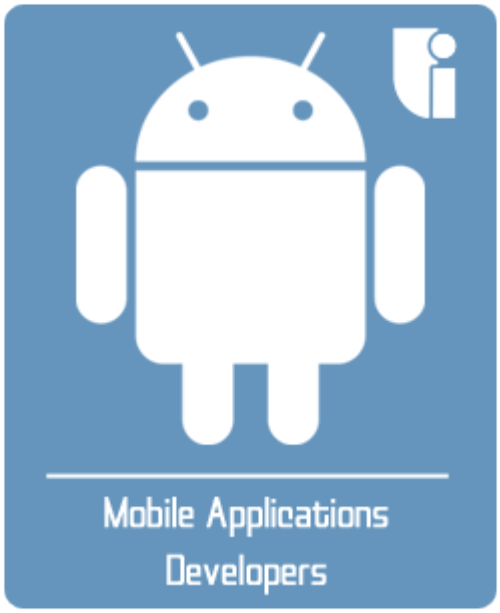
akademia androida



Bluetooth

część VIII

agenda

1. Informacje ogólne
2. Omówienie dostępnych klas do obsługi połączenia
3. Jak nadać uprawnienia ?
4. Konfiguracja połączenia
5. Znajdowanie urządzeń
6. Parowanie urządzeń
7. Odnajdowanie urządzeń dostępnych w okolicy
8. Ustawienie wykrywalności urządzeń na określony czas
9. Łączenie dwóch urządzeń
10. Zarządzanie połączeniem – wysyłanie tekstu
11. Zadanie

1. Informacje ogólne

Platforma Android umożliwia bezprzewodową wymianę danych między poszczególnymi urządzeniami za pomocą Bluetooth.

Dostęp do Bluetooth mamy poprzez Android Bluetooth API. Interfejs pozwala na łączenie się z innymi urządzeniami typu punkt-punkt.

Korzystając z Bluetooth API mamy możliwość:

a. Skanowania jakie urządzenia bluetooth są dostępne w okolicy
b. Parowania urządzeń
c. Przesyłania danych z lub do urządzeń
d. Zarządzania wieloma połączeniami

2. Omówienie dostępnych klas do obsługi połączenia

API Bluetooth jest dostępne w pakiecie android.bluetooth.

Do stworzenia połączenia będą potrzebne nam następujące klasy:

- BluetoothAdapter - klasa odpowiedzialna za wykrywanie innych urządzeń
- BluetoothDevice – reprezentuje zdalne urządzenie Bluetooth
- BluetoothSocket – Jest to punkt połączenia, gdzie aplikacja może wymieniać dane z innym urządzeniem poprzez InputStream i OutputStream
- BluetoothServerSocket - otwiera port serwera, który nasłuchuje przychodzące żądania.

3. Jak nadać uprawnienia?

Aby można było korzystać z funkcji bluetooth w aplikacji musimy zadeklarować jedno z dwóch uprawnień :

`BLUETOOTH` i `BLUETOOTH_ADMIN`.

Przykład:

```
<manifest ... >
  <uses-permission android:name="android.permission.BLUETOOTH" />
  ...
</manifest>
```

Dzięki temu będziemy mieli możliwość przeprowadzenie różnego rodzaju komunikacji takich jak żądanie połączenia, przyjmowanie połączeń oraz transferu danych.

W celu zainicjowania wykrywania urządzeń lub manipulowania ustawieniami musimy zainicjować `BLUETOOTH_ADMIN`

Uprawnienie `BLUTOOTH_ADMIN` nie zwalnia nas z tego aby nie zainicjować uprawnienia `BLUETOOTH`.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

4. Konfiguracja połączeń

Przed połączeniem aplikacji za pomocą bluetooth musimy sprawdzić czy nasze urządzenie ją obsługuje. Metoda getDefaultAdapter() zwraca nazwę urządzenia.

Kod:

```
BluetoothAdapter mBluetoothAdapter =
                            BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) {
    // urządzenie nie obsługuje bluetooth
}
```

Włączanie Bluetooth:

```
if (!mBluetoothAdapter.isEnabled()) {
  Intent enableBtIntent=
            new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
  startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

Metoda isEnabled() sprawdza czy funkcja bluetooth jest włączona. Tworzymy intencję, która obsłuży nam włączanie. Metoda startActivityForResult() wywoła utworzoną intencję. REQUEST_ENABLE_BT – pokazuje, że system pozwala użytkownikowi włączyć bluetooth.

```
private final static int REQUEST_ENABLE_BT = 1;
```

5. Znajdowanie urządzeń

Aby znaleźć dostępne urządzenia musimy skorzystać z klasy BluetoothAdapter.

1. Na początku skanujemy wszystkie urządzenia w pobliżu, pobieramy z każdego takie informacje jak: nazwa urządzenia oraz unikalny adres MAC.
2. Jeśli przechwycimy informacje to następuje zainicjowanie połączenia.
3. Po nawiązaniu połączenia przechodzimy do parsowania urządzeń.
4. Po parsowaniu mamy wgląd do takich informacji jak: nazwa urządzenia oraz jej adres MAC.

6. Parsowanie urządzeń

Do sprawdzenia, które urządzenia są spreparowane używamy getBondedDevices(). Następnie możemy wypisać nazwę i adres urządzenia.

Przykładowy kod:

```
Set<BluetoothDevice> pairedDevices =
mBluetoothAdapter.getBondedDevices();
```

```
// sprawdzamy czy są sparsowane urządzenia

if (pairedDevices.size() > 0) {

    // w pętli znajdujemy urządzenia
    for (BluetoothDevice device : pairedDevices) {

        // wypisywanie znalezionych urządzeń poprzez nazwe I adres
        mArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
    }
}
```

7. Odnajdowanie urządzeń w okolicy

Do znajdowania urządzeń potrzebne nam będzie wywołanie startDiscovery(). Proces jest asynchroniczny i zwraca czy urządzenia zostały znalezione czy nie.

Obsługa transmisji między wykrytymi urządzeniami:

```
// tworzymy BroadcastReceiver dla ACTION_FOUND
// po to aby uzyskać informacje na temat każdego urządzenia
private final BroadcastReceiver mReceiver = new BroadcastReceiver()
{
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        // gdy znajdzemy urządzenie
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // za pomocą Intecji pobieramy objekt BluetoothDevice
            BluetoothDevice device =
     intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // wypisywanie urządzeń
            mArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
        }
    }
};
// rejestrowanie BroadcastReceiver

IntentFilter filter = new
IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
```

8. Ustawienie wykrywalności urządzeń

Domyślnie urządzenie jest wykrywalne przez 120s. W przykładzie tym ustawiliśmy wykrywalność na 300s. W startActivity() wywołujemy utworzoną intencję.

Przykładowy kod:

```
Intent discoverableIntent =
          new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);

discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURA
TION, 300);
startActivity(discoverableIntent);
```

9. Łączenie urządzeń

Do utworzenia połączenia między dwoma urządzeniami, potrzebujemy strony klienta i serwera. Jedno urządzenie będzie otwierać gniazdo serwera, drugie natomiast musi zainicjować połączenie za pomocą adresu MAC.

Połączenie z serwerem

Otwieramy BluetoothServerSocket. Głównym celem gniazda serwera jest nasłuchiwanie przychodzących połączeń, i jeśli jakieś połączenie zaakceptujemy to połączymy się z BluetoothSocket

Jak utworzyć gniazdo i zaakceptować połączenie urządzeń?

1. Trzeba otrzymać BluetoothServerSocket poprzez wywołanie : listenUsingRfcommWithServiceRecord(String, UUID).
   - String jest rozumiany jako nazwa usługi,
   - UUID – identyfikator usługi, występuje wtedy, gdy klient będzie chciał się połączyć z urządzeniem
2. Nasłuchiwanie połączeń poprzez wywołanie funkcji accept(). Funkcja zwróci wynik jeśli połączenie zostało zrealizowane lub gdy wystąpi

wyjątek. Połączenie zostanie zaakceptowane tylko w momencie, gdy urządzenie wyśle informacje zwrotną z UUID. Jeśli zakończy się to powodzeniem, to akcept() zwróci połączenie z BluetoothSocket.

3. Jeśli nie chcesz aby były akceptowane dodatkowe połączenia to należy wywołać funkcje close().

Przykład wątek akceptujący przychodzące połączenia:

```
private class AcceptThread extends Thread {

    private final BluetoothServerSocket mmServerSocket;


    public AcceptThread() {

        // użycie tymczasowgo obiektu, który potem przypiszemy do
mmServerSocket

        // because mmServerSocket is final

        BluetoothServerSocket tmp = null;

        try {

            // MY_UUID w postaci stringu, używane przez stronę
                klienta

            tmp =
mBluetoothAdapter.listenUsingRfcommWithServiceRecord(NAME, MY_UUID);

        } catch (IOException e) { }

        mmServerSocket = tmp;

    }

     public void run() {

        BluetoothSocket socket = null;

        // nasłuchiwanie dopóki gniazdo nie zostanie zwrócone

        while (true) {

            try {

                socket = mmServerSocket.accept();

            } catch (IOException e) {

                break;
```

```
            }
            // gdy połączenie zostanie zaakceptowane
            if (socket != null) {
                //w oddzielnym wątku sprawdzamy czy działa
zarządzanie połączeniami
                manageConnectedSocket(socket);
                mmServerSocket.close();
                break;
            }
        }
    }
     /** zakonczenie nasłuchiwania
    public void cancel() {
        try {
            mmServerSocket.close();
        } catch (IOException e) { }
    }
}
```

Połączenie z klientem

Oto kroki wykonania procedury:

1. Musimy użyć BluetoothDevice aby otrzymać BluetoothSocket. Otrzymujemy poprzez createRfcommSocketToServiceRecord(UUID).
2. Nastepnie trzeba uzgodnić UUID po stronie klienta i serwera.
3. Inicjujemy połączenie za pomocą funkcji connect().

Przykład zainicjowania połączenia:

```
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;

    public ConnectThread(BluetoothDevice device) {
// użycie tymczasowego obiektu, który przypiszemy do mmSocket
        BluetoothSocket tmp = null;
        mmDevice = device;

        // pobranie BluetoothSocket do połączenia się z danymi
BluetoothDevice
        try {
            // MY_UUID is the app's UUID string, also used by the
server code
            tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
        } catch (IOException e) { }
        mmSocket = tmp;
    }

     public void run() {
        // zakończenie  wyszukiwania urządzen, aby nie spowolnić
połączenia
        mBluetoothAdapter.cancelDiscovery();

        try {
            // połączenie urządzenia przez gniazdo
            mmSocket.connect();
        } catch (IOException connectException) {
```

```
            // nie można się połączyć, wyjście z gniazda
            try {
                mmSocket.close();
            } catch (IOException closeException) { }
            return;
        }

        // sprawdzenie w osobnym wątku ,czy działa zarządzanie
połączeniem
        manageConnectedSocket(mmSocket);
    }

    /** anulowanie aktualnego połaczenia oraz zamknięcie gniazda*/
    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) { }
    }
}
```

Funkcję cancelDiscover() wywołujemy w momencie, gdy będziemy mieli połączenie. Close() zamyka połączenie z gniazdem i czyści wszystkie zasoby.

10. Zarządzanie połączeniem – wysyłanie danych

Jeśli połączymy dwa lub więcej urządzeń ze sobą każdy będzie połączony z BluetoothSocket. W tym momencie możemy przesyłać dane między urządzeniami. Procedura składa się z :

1. Pobierz strumień wejściowy i wyjściowy (InputStream OutputStream) za pomocą metod getInputStream() i getOutputStream().
2. Odczytywanie i zapisywanie danych jako strumien danych za pomocą read(byte[]) i write(byte[]).

Musimy użyć wątku do czytania I pisania. Metoda read blokuje połączenia dopóki jest coś do czytania. Metoda write zazwyczaj nie blokuje strumienia.

```
String msg = entry.getText().toString();
msg += "\n";
mmOutStream.write(msg.getBytes());
```

Wysyłanie danych przykład:

```
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket) {
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        // pobranie za pomocą obiektów tymczasowych strumienia
wejściowego i wyjściowego
        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
        } catch (IOException e) { }

        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }
```

```
    public void run() {
        byte[] buffer = new byte[1024];  // rozmiar bufora dla
strumienia
        int bytes; // bity zwracane z funkcji read.

        // nasłuchiwanie strumienia wejściowego  do momentu
pojawienia się wyjątku
        while (true) {
            try {
                // odczytywanie danych ze strumienia wejściowego
                bytes = mmInStream.read(buffer);
                // wysłyanie otrzymanych bajtów
                mHandler.obtainMessage(MESSAGE_READ, bytes, -1,
buffer)
                        .sendToTarget();
            } catch (IOException e) {
                break;
            }
        }
    }

    /* wysyłanie danych z głównej aktywności  do zdalnego urządzenia
*/
    public void write(byte[] bytes) {
        try {
            mmOutStream.write(bytes);
        } catch (IOException e) { }
    }

    /* wywołanie głównej aktywości do zakonczenia połączenia*/
    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) { }
    }
}
```

W konstruktorze przechwytuje strumienie, a wątek czeka na dane które mają przyjść poprzez InputStream. Jeśli metoda read byte[] odczyta wszystkie bajty ze strumienia to dane przesyłamy za pomocą Handlera. Następnie wraca i czeka na następne bajty ze strumienia. Wysyłanie danych wychodzących polega na wywołaniu metody write (byte[]).

Metoda cancel() służy do zakończenia połączenia.

Więcej informacji na stronie:

http://developer.android.com/guide/topics/connectivity/bluetooth.html

11. Zadanie

Napisać aplikację, która obsłuży komunikację bluetooth pomiędzy dwoma urządzeniami. Należy stworzyć osobne przyciski jeden do włączania bluetooth, drugi do wyłączania, trzeci do wysyłania komunikatu, czwarty, który zwróci nam nazwy adresy dostępnych urządzeń w okolicy.. Dodatkowo potrzebujemy EditText do wprowadzania tekstu do wysłania, oraz TextView do wyświetlania wiadomości.

Dziękuję za uwagę !

Kontakt:

- kigras@wi.zut.edu.pl