**Приложение В**

**// AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.aska.bluetothmessanger">

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

**<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"/>**

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_laun\_bluetooth"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher"

android:supportsRtl="true"

android:theme="@style/AppTheme"

android:windowSoftInputMode="adjustResize">

<activity

android:name=".MainActivity"

android:label="@string/app\_name">

</activity>

<activity android:name=".Start\_activity">

</activity>

<activity android:name=".SplashActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN"/>

<category android:name="android.intent.category.LAUNCHER"/>

</intent-filter>

</activity>

</application>

</manifest>

**//MainActivity.java**

public class MainActivity extends AppCompatActivity implements View.OnClickListener{

Button chat, about, close;

public static final int MESSAGE\_STATE\_CHANGE = 1;

public static final int MESSAGE\_READ = 2;

public static final int MESSAGE\_WRITE = 3;

public static final int MESSAGE\_DEVICE\_OBJECT = 4;

public static final int MESSAGE\_TOAST = 5;

public static final String DEVICE\_OBJECT = "device\_name";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_start\_activity);

chat = (Button) findViewById(R.id.button3);

close = (Button) findViewById(R.id.button5);

about = (Button) findViewById(R.id.button4);

chat.setOnClickListener(this);

about.setOnClickListener(this);

close.setOnClickListener(this);

}

@Override

public void onClick(View v) {

switch (v.getId()) {

case R.id.button3: Intent chat = new Intent(this, Start\_activity.class);

startActivity(chat);

break;

case R.id.button4:

AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);

builder.setTitle("About application").setMessage("Bluetooth Messenger\n " +

"Arslan Annaev - KI-15").setIcon(R.mipmap.ic\_laun\_bluetooth)

.setCancelable(false).setNegativeButton("ОК",

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

dialog.cancel();

}});

AlertDialog alert = builder.create();

alert.show(); break;

case R.id.button5: finish(); break;

default:;

}

}

}

**//SplashActivity.java**

public class SplashActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_splash);

AsyncTask<Void,Integer,Void> asyncTask = new AsyncTask<Void, Integer, Void>()

{

@Override

protected Void doInBackground(Void... params) {

sleepSecond();

startAct();

return null;

}

@Override

protected void onProgressUpdate(Integer... values) {}

@Override

protected void onPostExecute(Void aVoid) {

}

private void sleepSecond(){

try{ TimeUnit.SECONDS.sleep(3); } catch (InterruptedException err){}

}

};

asyncTask.execute();

}

void startAct(){

startActivity(new Intent(this, MainActivity.class));

finish();

}

}

**// StartActivity.java**

public class Start\_activity extends AppCompatActivity {

private TextView status;

private Button btnConnect;

private ListView listView;

private Dialog dialog;

private TextInputLayout inputLayout;

private ArrayAdapter<String> chatAdapter;

private ArrayList<String> chatMessages;

private BluetoothAdapter bluetoothAdapter;

private static final int REQUEST\_ENABLE\_BLUETOOTH = 1;

private ChatController chatController;

private BluetoothDevice connectingDevice;

private ArrayAdapter<String> discoveredDevicesAdapter;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

findViewsByIds();

//check device support bluetooth or not

bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

if (bluetoothAdapter == null) {

Toast.makeText(this, "Bluetooth is not available!",

Toast.LENGTH\_SHORT).show();

finish();

}

//show bluetooth devices dialog when click connect button

btnConnect.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

showPrinterPickDialog();

}

});

//set chat adapter

chatMessages = new ArrayList<>();

chatAdapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1,

chatMessages);

listView.setAdapter(chatAdapter);

}

private Handler handler = new Handler(new Handler.Callback() {

@Override

public boolean handleMessage(Message msg) {

switch (msg.what) {

case MainActivity.MESSAGE\_STATE\_CHANGE:

switch (msg.arg1) {

case ChatController.STATE\_CONNECTED: setStatus("Connected to: "+"\n" +

connectingDevice.getName());

btnConnect.setEnabled(false);

break;

case ChatController.STATE\_CONNECTING: setStatus("Connecting...");

btnConnect.setEnabled(false);

break;

case ChatController.STATE\_LISTEN:

case ChatController.STATE\_NONE: setStatus("Not connected");

break;

} break;

case MainActivity.MESSAGE\_WRITE: byte[] writeBuf = (byte[]) msg.obj;

String writeMessage = new String(writeBuf);

chatMessages.add("Me: " + writeMessage);

chatAdapter.notifyDataSetChanged();

break;

case MainActivity.MESSAGE\_READ: byte[] readBuf = (byte[]) msg.obj;

String readMessage = new String(readBuf, 0, msg.arg1);

chatMessages.add(connectingDevice.getName() + ": " + readMessage);

chatAdapter.notifyDataSetChanged();

break;

case MainActivity.MESSAGE\_DEVICE\_OBJECT:

connectingDevice = msg.getData().getParcelable(MainActivity.DEVICE\_OBJECT);

Toast.makeText(getApplicationContext(),

"Connected to:"+"\n" + connectingDevice.getName(),

Toast.LENGTH\_SHORT).show();

break;

case MainActivity.MESSAGE\_TOAST: Toast.makeText(getApplicationContext(),

msg.getData().getString("toast"),

Toast.LENGTH\_SHORT).show();

break;

}

return false;

}

});

private void showPrinterPickDialog() {

dialog = new Dialog(this);

dialog.setContentView(R.layout.layout\_bluetooth);

dialog.setTitle("Bluetooth Devices");

if (bluetoothAdapter.isDiscovering()) {

bluetoothAdapter.cancelDiscovery();

}

bluetoothAdapter.startDiscovery();

//Initializing bluetooth adapters

ArrayAdapter<String> pairedDevicesAdapter = new ArrayAdapter<>(this,

android.R.layout.simple\_list\_item\_1);

discoveredDevicesAdapter = new ArrayAdapter<>(this,

android.R.layout.simple\_list\_item\_1);

//locate listviews and attatch the adapters

ListView listView = (ListView) dialog.findViewById(R.id.pairedDeviceList);

ListView listView2 = (ListView) dialog.findViewById(R.id.discoveredDeviceList);

listView.setAdapter(pairedDevicesAdapter);

listView2.setAdapter(discoveredDevicesAdapter);

// Register for broadcasts when a device is discovered

IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION\_FOUND);

registerReceiver(discoveryFinishReceiver, filter);

// Register for broadcasts when discovery has finished

filter = new IntentFilter(BluetoothAdapter.ACTION\_DISCOVERY\_FINISHED);

registerReceiver(discoveryFinishReceiver, filter);

bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

Set<BluetoothDevice> pairedDevices = bluetoothAdapter.getBondedDevices();

// If there are paired devices, add each one to the ArrayAdapter

if (pairedDevices.size() > 0) {

for (BluetoothDevice device : pairedDevices) {

pairedDevicesAdapter.add(device.getName() + "\n" +

device.getAddress());

}

} else {

pairedDevicesAdapter.add(getString(R.string.none\_paired));

}

//Handling listview item click event

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view,

int position, long id) {

bluetoothAdapter.cancelDiscovery();

String info = ((TextView) view).getText().toString();

String address = info.substring(info.length() - 17);

connectToDevice(address);

dialog.dismiss(); } });

listView2.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {

bluetoothAdapter.cancelDiscovery();

String info = ((TextView) view).getText().toString();

String address = info.substring(info.length() - 17);

connectToDevice(address);

dialog.dismiss();

}

});

dialog.findViewById(R.id.cancelButton).setOnClickListener(

new View.OnClickListener() {

@Override

public void onClick(View v) { dialog.dismiss(); }

});

dialog.setCancelable(false);

dialog.show();

}

private void setStatus(String s) {

status.setText(s);

}

private void connectToDevice(String deviceAddress) {

bluetoothAdapter.cancelDiscovery();

BluetoothDevice device = bluetoothAdapter.getRemoteDevice(deviceAddress);

chatController.connect(device);

}

private void findViewsByIds() {

status = (TextView) findViewById(R.id.status);

btnConnect = (Button) findViewById(R.id.btn\_connect);

listView = (ListView) findViewById(R.id.list);

inputLayout = (TextInputLayout) findViewById(R.id.input\_layout);

View btnSend = findViewById(R.id.btn\_send);

btnSend.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if (inputLayout.getEditText().getText().toString().equals("")) {

Toast.makeText(Start\_activity.this, "Please input some texts",

Toast.LENGTH\_SHORT).show();

} else {

//TODO: here

sendMessage(inputLayout.getEditText().getText().toString());

inputLayout.getEditText().setText("");

}

}

});

}

public void onActivityResult(int requestCode, int resultCode, Intent data) {

switch (requestCode) {

case REQUEST\_ENABLE\_BLUETOOTH:

if (resultCode == Activity.RESULT\_OK) {

chatController = new ChatController(this, handler);

} else {

Toast.makeText(this, "Bluetooth still disabled, turn off application!", Toast.LENGTH\_SHORT).show();

finish();

}

}

}

private void sendMessage(String message) {

if (chatController.getState() != ChatController.STATE\_CONNECTED) {

Toast.makeText(this, "Connection was lost!", Toast.LENGTH\_SHORT).show();

return;

}

if (message.length() > 0) {

byte[] send = message.getBytes();

chatController.write(send);

}

}

@Override

public void onStart() {

super.onStart();

if (!bluetoothAdapter.isEnabled()) {

Intent enableIntent = new Intent(BluetoothAdapter.ACTION\_REQUEST\_ENABLE);

startActivityForResult(enableIntent, REQUEST\_ENABLE\_BLUETOOTH);

} else {

chatController = new ChatController(this, handler);

}

}

@Override

public void onResume() {

super.onResume();

if (chatController != null) {

if (chatController.getState() == ChatController.STATE\_NONE) {

chatController.start();

}

}

}

@Override

public void onDestroy() {

super.onDestroy();

if (chatController != null)

chatController.stop();

}

private final BroadcastReceiver discoveryFinishReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

if (BluetoothDevice.ACTION\_FOUND.equals(action)) {

BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE);

if (device.getBondState() != BluetoothDevice.BOND\_BONDED) {

discoveredDevicesAdapter.add(device.getName() + "\n" + device.getAddress());

}

} else if (BluetoothAdapter.ACTION\_DISCOVERY\_FINISHED.equals(action)) {

if (discoveredDevicesAdapter.getCount() == 0) {

discoveredDevicesAdapter.add(getString(R.string.none\_found));

}

}

}

};

**//ChatController.java**

public class ChatController {

private static final String APP\_NAME = "Bluetooth Messenger";

private static final UUID MY\_UUID = UUID.fromString("8ce255c0-200a-11e0-ac64-0800200c9a66");

private final BluetoothAdapter bluetoothAdapter;

private final Handler handler;

private AcceptThread acceptThread;

private ConnectThread connectThread;

private ReadWriteThread connectedThread;

private int state;

static final int STATE\_NONE = 0;

static final int STATE\_LISTEN = 1;

static final int STATE\_CONNECTING = 2;

static final int STATE\_CONNECTED = 3;

public ChatController(Context context, Handler handler) {

bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

state = STATE\_NONE;

this.handler = handler;

}

// Set the current state of the chat connection

private synchronized void setState(int state) {

this.state = state;

handler.obtainMessage(MainActivity.MESSAGE\_STATE\_CHANGE, state, -1).sendToTarget();

}

// get current connection state

public synchronized int getState() {

return state;

}

// start service

public synchronized void start() {

// Cancel any thread

if (connectThread != null) {

connectThread.cancel();

connectThread = null;

}

// Cancel any running thread

if (connectedThread != null) {

connectedThread.cancel();

connectedThread = null;

}

setState(STATE\_LISTEN);

if (acceptThread == null) {

acceptThread = new AcceptThread();

acceptThread.start();

}

}

// initiate connection to remote device

public synchronized void connect(BluetoothDevice device) {

// Cancel any thread

if (state == STATE\_CONNECTING) {

if (connectThread != null) {

connectThread.cancel();

connectThread = null;

}

}

// Cancel running thread

if (connectedThread != null) {

connectedThread.cancel();

connectedThread = null;

}

// Start the thread to connect with the given device

connectThread = new ConnectThread(device);

connectThread.start();

setState(STATE\_CONNECTING);

}

// manage Bluetooth connection

public synchronized void connected(BluetoothSocket socket, BluetoothDevice device) {

// Cancel the thread

if (connectThread != null) {

connectThread.cancel();

connectThread = null;

}

// Cancel running thread

if (connectedThread != null) {

connectedThread.cancel();

connectedThread = null;

}

if (acceptThread != null) {

acceptThread.cancel();

acceptThread = null;

}

// Start the thread to manage the connection and perform transmissions

connectedThread = new ReadWriteThread(socket);

connectedThread.start();

// Send the name of the connected device back to the UI Activity

Message msg = handler.obtainMessage(MainActivity.MESSAGE\_DEVICE\_OBJECT);

Bundle bundle = new Bundle();

bundle.putParcelable(MainActivity.DEVICE\_OBJECT, device);

msg.setData(bundle);

handler.sendMessage(msg);

setState(STATE\_CONNECTED);

}

// stop all threads

public synchronized void stop() {

if (connectThread != null) {

connectThread.cancel();

connectThread = null;

}

if (connectedThread != null) {

connectedThread.cancel();

connectedThread = null;

}

if (acceptThread != null) {

acceptThread.cancel();

acceptThread = null;

}

setState(STATE\_NONE);

}

public void write(byte[] out) {

ReadWriteThread r;

synchronized (this) {

if (state != STATE\_CONNECTED)

return;

r = connectedThread;

}

r.write(out);

}

private void connectionFailed() {

Message msg = handler.obtainMessage(MainActivity.MESSAGE\_TOAST);

Bundle bundle = new Bundle();

bundle.putString("toast", "Unable to connect device");

msg.setData(bundle);

handler.sendMessage(msg);

// Start the service over to restart listening mode

ChatController.this.start();

}

private void connectionLost() {

Message msg = handler.obtainMessage(MainActivity.MESSAGE\_TOAST);

Bundle bundle = new Bundle();

bundle.putString("toast", "Device connection was lost");

msg.setData(bundle);

handler.sendMessage(msg);

// Start the service over to restart listening mode

ChatController.this.start();

}

// runs while listening for incoming connections

private class AcceptThread extends Thread {

private final BluetoothServerSocket serverSocket;

public AcceptThread() {

BluetoothServerSocket tmp = null;

try {

tmp = bluetoothAdapter.listenUsingInsecureRfcommWithServiceRecord(APP\_NAME, MY\_UUID);

} catch (IOException ex) {

ex.printStackTrace();

}

serverSocket = tmp;

}

public void run() {

setName("AcceptThread");

BluetoothSocket socket;

while (state != STATE\_CONNECTED) {

try {

socket = serverSocket.accept();

} catch (IOException e) {

break;

}

// If a connection was accepted

if (socket != null) {

synchronized (ChatController.this) {

switch (state) {

case STATE\_LISTEN:

case STATE\_CONNECTING:

// start the connected thread.

connected(socket, socket.getRemoteDevice());

break;

case STATE\_NONE:

case STATE\_CONNECTED:

// Either not ready or already connected. Terminate

// new socket.

try {

socket.close();

} catch (IOException e) {

}

break;

} } } } }

public void cancel() {

try {

serverSocket.close();

} catch (IOException e) {

}

}

}

// runs while attempting to make an outgoing connection

private class ConnectThread extends Thread {

private final BluetoothSocket socket;

private final BluetoothDevice device;

public ConnectThread(BluetoothDevice device) {

this.device = device;

BluetoothSocket tmp = null;

try {

tmp = device.createInsecureRfcommSocketToServiceRecord(MY\_UUID);

} catch (IOException e) {

e.printStackTrace();

}

socket = tmp;

}

public void run() {

setName("ConnectThread");

// Always cancel discovery because it will slow down a connection

bluetoothAdapter.cancelDiscovery();

// Make a connection to the BluetoothSocket

try {

socket.connect();

} catch (IOException e) {

try {

socket.close();

} catch (IOException e2) {

}

connectionFailed();

return;

}

// Reset the ConnectThread because we're done

synchronized (ChatController.this) {

connectThread = null;

}

// Start the connected thread

connected(socket, device);

}

public void cancel() {

try { socket.close();

} catch (IOException e) {

}

}

}

// runs during a connection with a remote device

private class ReadWriteThread extends Thread {

private final BluetoothSocket bluetoothSocket;

private final InputStream inputStream;

private final OutputStream outputStream;

public ReadWriteThread(BluetoothSocket socket) {

this.bluetoothSocket = socket;

InputStream tmpIn = null;

OutputStream tmpOut = null;

try {

tmpIn = socket.getInputStream();

tmpOut = socket.getOutputStream();

} catch (IOException e) {

}

inputStream = tmpIn;

outputStream = tmpOut;

}

public void run() {

byte[] buffer = new byte[1024];

int bytes;

// Keep listening to the InputStream

while (true) {

try {

// Read from the InputStream

bytes = inputStream.read(buffer);

// Send the obtained bytes to the UI Activity

handler.obtainMessage(MainActivity.MESSAGE\_READ, bytes, -1,

buffer).sendToTarget();

} catch (IOException e) {

connectionLost();

// Start the service over to restart listening mode

ChatController.this.start();

break;

}

}

}

// write to OutputStream

public void write(byte[] buffer) {

try {

outputStream.write(buffer);

handler.obtainMessage(MainActivity.MESSAGE\_WRITE, -1, -1,

buffer).sendToTarget();

} catch (IOException e) {

}

}

public void cancel() {

try {

bluetoothSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}