# Приложение А.

Код класса *MainActivity*.

package com.example.budilnik;

import static com.example.budilnik.BluetoothLeService.heartRate;

import android.annotation.SuppressLint;

import android.app.AlarmManager;

import android.app.AlertDialog;

import android.app.PendingIntent;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.SharedPreferences;

import android.content.pm.PackageManager;

import android.os.Bundle;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.View;

import android.widget.CompoundButton;

import android.widget.EditText;

import android.widget.Switch;

import android.widget.TextView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import com.google.android.material.snackbar.Snackbar;

import com.google.android.material.timepicker.MaterialTimePicker;

import com.google.android.material.timepicker.TimeFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Date;

import java.util.List;

import java.util.Locale;

import java.util.Objects;

public class MainActivity extends AppCompatActivity {

public static int heartRate;

SimpleDateFormat sdf = new SimpleDateFormat("dd LLLL HH:mm:ss", Locale.getDefault());

public static Integer hour;

public static Integer min;

public static Integer day;

public static Integer hour2;

public static Integer min2;

public static Integer day2;

final Context context = this;

static AlarmManager alarmManager;

public static final int REQUEST\_CODE\_LOC=1;

static Switch switchView;

Calendar calendar\_zavtro = Calendar.getInstance();

Calendar calendar\_segodny = Calendar.getInstance();

public static boolean flagVkl=false;

private int interval;

public static boolean flagVklB=false;

public static boolean flagVklI=false;

@Override

protected void onDestroy() {

flagVkl=false;

SharedPreferences savedData = getSharedPreferences("SavedData",MODE\_PRIVATE);//сохранение данных

SharedPreferences.Editor editor=savedData.edit();//сохранение данных

editor.putBoolean("flag",flagVkl);

editor.apply();

AlarmActivity.hour= hour;

AlarmActivity.min= min;

super.onDestroy();

}

@SuppressLint("SetTextI18n")

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

razrysheniy();

FloatingActionButton fab = findViewById(R.id.fab);

fab.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

final Intent intent = new Intent(MainActivity.this,

MainBluetooth.class);

startActivity(intent);

}

});

FloatingActionButton interval2 = findViewById(R.id.interval);

interval2.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

LayoutInflater li = LayoutInflater.from(context);

View promptsView = li.inflate(R.layout.prompts, null);

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(

context);

alertDialogBuilder.setView(promptsView);

final EditText userInput = (EditText) promptsView

.findViewById(R.id.editTextDialogUserInput);

// set dialog message

alertDialogBuilder

.setCancelable(false)

.setPositiveButton("OK",

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,int id) {

// get user input and set it to result

// edit text

String im =String.valueOf(userInput.getText());

try {

interval=Integer.parseInt (im);

}catch (NumberFormatException e){

Toast.makeText(MainActivity.this, "Вы ввели неправильное число.\n(Вы могли ввести число с буквой или дробное число)", Toast.LENGTH\_LONG).show();

interval=20;

}

SharedPreferences savedData = getSharedPreferences("SavedData",MODE\_PRIVATE);//сохранение данных

SharedPreferences.Editor editor=savedData.edit();//сохранение данных

editor.putInt("interval", interval);

editor.apply();

if (min<10){

if (hour<10){

switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");

}}else{

if (hour<10){

switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");

}

}

}

})

.setNegativeButton("Отмена",

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,int id) {

dialog.cancel();

}

});

// create alert dialog

AlertDialog alertDialog = alertDialogBuilder.create();

// show it

alertDialog.show();

}

});

SharedPreferences savedData = getSharedPreferences("SavedData",MODE\_PRIVATE);//сохранение данных

interval = savedData.getInt("interval", 30);

hour=savedData.getInt("Hours",0);

min=savedData.getInt("Min",0);

flagVkl=savedData.getBoolean("flag",false);

//day="0";

switchView=findViewById(R.id.switch\_bn1);

if (min<10){

if (hour<10){

switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");

}}else{

if (hour<10){

switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");

}

}

proverka();

switchView.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

if (isChecked) {

budilka();

flagVklB=true;

} else {

otmena();

}

}

});

}

@SuppressLint("UnspecifiedImmutableFlag")

private PendingIntent getAlarmInfoPendingIntent(){

Intent alarmInfoIntent = new Intent(this,MainActivity.class);

alarmInfoIntent.setFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP|Intent.FLAG\_ACTIVITY\_NEW\_TASK);

return PendingIntent.getActivity(this,0,alarmInfoIntent,PendingIntent.FLAG\_UPDATE\_CURRENT);

}

@SuppressLint("UnspecifiedImmutableFlag")

public PendingIntent getAlarmActionPendingIntent(){//Что произойдёт когда сработает будильник

Intent intent = new Intent(this,AlarmActivity.class);

intent.setFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP|Intent.FLAG\_ACTIVITY\_NEW\_TASK);

return PendingIntent.getActivity(this,1,intent,PendingIntent.FLAG\_UPDATE\_CURRENT);

}

@SuppressLint("SetTextI18n")

public void budilka(){

SharedPreferences savedData = getSharedPreferences("SavedData",MODE\_PRIVATE);//сохранение данных

hour=savedData.getInt("Hours",0);

min=savedData.getInt("Min",0);

if (min<10){

if (hour<10){

switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");

}}else{

if (hour<10){

switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");

}

}

MaterialTimePicker materialTimePicker =new MaterialTimePicker.Builder()

.setTimeFormat(TimeFormat.CLOCK\_24H)

.setHour(hour)

.setMinute(min)

.setTitleText("Выбирите время для будильника")

.build();

materialTimePicker.addOnPositiveButtonClickListener(view ->{

Date currentTime = Calendar.getInstance().getTime();

if (currentTime.getHours()>materialTimePicker.getHour()) {

zavtro(materialTimePicker,currentTime);

}else {

if (currentTime.getMinutes()>materialTimePicker.getMinute()){

zavtro(materialTimePicker,currentTime) ;

}else{

segodny(materialTimePicker,currentTime) ;

}

}

interval();

SharedPreferences.Editor editor=savedData.edit();//сохранение данных

editor.putInt("Hours", hour);

editor.putInt("Min", min);

editor.putBoolean("flag",flagVkl);

editor.apply();

Log.d("asd","min="+savedData.getInt("Min",0)+"Hours"+savedData.getInt("Hours",0));

if(Objects.equals(day, "0")){

switchView=findViewById(R.id.switch\_bn1);

if (min<10){

if (hour<10){

switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");

}}else{

if (hour<10){

switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");

}

}

}else {

switchView = findViewById(R.id.switch\_bn1);

if (min<10){

if (hour<10){

switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");

}}else{

if (hour<10){

switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");

}

}

}

});

materialTimePicker.show(getSupportFragmentManager(),"Tag\_piker");

}

@SuppressLint("SetTextI18n")

public void otmena(){

SharedPreferences savedData = getSharedPreferences("SavedData",MODE\_PRIVATE);//сохранение данных

hour=savedData.getInt("Hours",0);

min=savedData.getInt("Min",00);

switchView=findViewById(R.id.switch\_bn1);

if (min<10){

if (hour<10){

switchView.setText("0"+hour+":0"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":0"+min+" Интервал ="+interval+" мин");

}}else{

if (hour<10){

switchView.setText("0"+hour+":"+min+" Интервал ="+interval+" мин");

}else{

switchView.setText(hour+":"+min+" Интервал ="+interval+" мин");

}

}

alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);

if (getAlarmActionPendingIntent()!=null){

alarmManager.cancel(getAlarmActionPendingIntent());}//Отменяем будильник, связанный с интентом данного класса

// Toast.makeText(this, "Будильник отменён", Toast.LENGTH\_LONG).show();

flagVkl=false;

switchView.setChecked(flagVkl);

SharedPreferences.Editor editor=savedData.edit();//сохранение данных

editor.putInt("Hours", hour);

editor.putInt("Min", min);

editor.putBoolean("flag",flagVkl);

editor.apply();

}

public void zavtro(MaterialTimePicker materialTimePicker,Date currentTime){

flagVkl=true;

calendar\_zavtro.set(Calendar.SECOND, currentTime.getSeconds());

calendar\_zavtro.set(Calendar.MILLISECOND, 0);

calendar\_zavtro.set(Calendar.MINUTE, materialTimePicker.getMinute());

calendar\_zavtro.set(Calendar.HOUR\_OF\_DAY, materialTimePicker.getHour());

calendar\_zavtro.set(Calendar.DAY\_OF\_MONTH,currentTime.getDate()+1);

day=calendar\_zavtro.getTime().getDate();

min = materialTimePicker.getMinute();//сохранение данных

hour = materialTimePicker.getHour();

alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);

AlarmManager.AlarmClockInfo alarmClockInfo = new AlarmManager.AlarmClockInfo(calendar\_zavtro.getTimeInMillis(), getAlarmInfoPendingIntent());

alarmManager.setAlarmClock(alarmClockInfo, getAlarmActionPendingIntent());

Toast.makeText(this, "Будильник установлен на: " + sdf.format(calendar\_zavtro.getTime()), Toast.LENGTH\_LONG).show();

}

public void segodny(MaterialTimePicker materialTimePicker,Date currentTime){

flagVkl=true;

calendar\_segodny.set(Calendar.SECOND, currentTime.getSeconds());

calendar\_segodny.set(Calendar.MILLISECOND, 0);

calendar\_segodny.set(Calendar.MINUTE, materialTimePicker.getMinute());

calendar\_segodny.set(Calendar.HOUR\_OF\_DAY, materialTimePicker.getHour());

calendar\_segodny.set(Calendar.DAY\_OF\_MONTH,currentTime.getDate());

min = materialTimePicker.getMinute();//сохранение данных

hour = materialTimePicker.getHour();

day=calendar\_segodny.getTime().getDate();

alarmManager = (AlarmManager) getSystemService(Context.ALARM\_SERVICE);

AlarmManager.AlarmClockInfo alarmClockInfo = new AlarmManager.AlarmClockInfo(calendar\_segodny.getTimeInMillis(), getAlarmInfoPendingIntent());

alarmManager.setAlarmClock(alarmClockInfo, getAlarmActionPendingIntent());

Toast.makeText(this, "Будильник установлен на: " + sdf.format(calendar\_segodny.getTime()), Toast.LENGTH\_LONG).show();

}

public void proverka(){

switchView=findViewById(R.id.switch\_bn1);

switchView.setChecked(flagVkl);

}

public void razrysheniy(){

int accessCoarseLocation = this.checkSelfPermission(android.Manifest.permission.ACCESS\_COARSE\_LOCATION);

int accessFineLocation = this.checkSelfPermission(android.Manifest.permission.ACCESS\_FINE\_LOCATION);

int acessWRITEEXTERNALSTORAGE=this.checkSelfPermission(android.Manifest.permission.WRITE\_EXTERNAL\_STORAGE);

List<String> listRequestPermission = new ArrayList<String>();

if (accessCoarseLocation != PackageManager.PERMISSION\_GRANTED) {

listRequestPermission.add(android.Manifest.permission.ACCESS\_COARSE\_LOCATION);

}

if (accessFineLocation != PackageManager.PERMISSION\_GRANTED) {

listRequestPermission.add(android.Manifest.permission.ACCESS\_FINE\_LOCATION);

}

if (acessWRITEEXTERNALSTORAGE != PackageManager.PERMISSION\_GRANTED) {

listRequestPermission.add(android.Manifest.permission.WRITE\_EXTERNAL\_STORAGE);

}

if (!listRequestPermission.isEmpty()) {

String[] strRequestPermission = listRequestPermission.toArray(new String[listRequestPermission.size()]);

this.requestPermissions(strRequestPermission, REQUEST\_CODE\_LOC);

}

}

public void interval(){

if (min-interval<0){

min2=min-interval+60;

if (hour-1<0){

hour2=hour+23;

day2=day-1;

}else{

hour2=hour-1;

day2= day;

}

}else{

min2=min-interval;

hour2=hour;

day2=day;

}

}

}

# Приложение Б.

Код класса MainBluetooth.

package com.example.budilnik;

import android.app.Activity;

import android.app.AlertDialog;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.bluetooth.BluetoothManager;

import android.bluetooth.le.BluetoothLeScanner;

import android.bluetooth.le.ScanCallback;

import android.bluetooth.le.ScanFilter;

import android.bluetooth.le.ScanResult;

import android.bluetooth.le.ScanSettings;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.os.Bundle;

import android.os.Handler;

import android.os.PowerManager;

import android.provider.Settings;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.ListAdapter;

import android.widget.ListView;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;

import java.util.List;

public class MainBluetooth extends AppCompatActivity {

private BluetoothAdapter mBluetoothAdapter;

private BluetoothLeScanner mBluetoothLeScanner;

private boolean mScanning;

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

Button btnScan;

ListView listViewLE;

List<BluetoothDevice> listBluetoothDevice;

List<String> listBluetoothDeviceString;

ListAdapter adapterLeScanResult;

private Handler mHandler;

private static final long SCAN\_PERIOD = 20000;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main\_bluetooth);

Intent intent = new Intent();

String packageName = getPackageName();

PowerManager pm = (PowerManager) getSystemService(POWER\_SERVICE);

if (!pm.isIgnoringBatteryOptimizations(packageName)) {

Toast.makeText(this,

"Пожалуйста, отключите оптимизацию расхода батареи для этого приложения. " +

"\n Это нужно для правильной работы приложения.",

Toast.LENGTH\_LONG).show();

intent.setAction(Settings.ACTION\_IGNORE\_BATTERY\_OPTIMIZATION\_SETTINGS);

//intent.setData(Uri.parse("package:" + packageName));

startActivity(intent); }

// Check if BLE is supported on the device.

if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE\_BLUETOOTH\_LE)) {

Toast.makeText(this,

"BLUETOOTH\_LE not supported in this device!",

Toast.LENGTH\_SHORT).show();

finish();

}

getBluetoothAdapterAndLeScanner();

// Checks if Bluetooth is supported on the device.

if (mBluetoothAdapter == null) {

Toast.makeText(this,

"bluetoothManager.getAdapter()==null",

Toast.LENGTH\_SHORT).show();

finish();

return;

}

btnScan = (Button)findViewById(R.id.scan);

btnScan.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

scanLeDevice(true);

}

});

listViewLE = (ListView)findViewById(R.id.lelist);

listBluetoothDevice = new ArrayList<>();

listBluetoothDeviceString = new ArrayList<>();

adapterLeScanResult = new ArrayAdapter<String>(

this, android.R.layout.simple\_list\_item\_1, listBluetoothDeviceString);

listViewLE.setAdapter(adapterLeScanResult);

listViewLE.setOnItemClickListener(scanResultOnItemClickListener);

mHandler = new Handler();

}

AdapterView.OnItemClickListener scanResultOnItemClickListener =

new AdapterView.OnItemClickListener(){

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

final BluetoothDevice device = (BluetoothDevice) listBluetoothDevice.get(position);

String msg = device.getAddress() + "\n"

+"Bluetooth класс "+ device.getBluetoothClass().toString() + "\n"

+ getBTDevieType(device);

new AlertDialog.Builder(MainBluetooth.this)

.setTitle(device.getName())

.setMessage(msg)

.setPositiveButton("Назад", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

}

})

.setNeutralButton("Подключится", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

final Intent intent = new Intent(MainBluetooth.this,

ControlActivity.class);

intent.putExtra(ControlActivity.EXTRAS\_DEVICE\_NAME,

device.getName());

intent.putExtra(ControlActivity.EXTRAS\_DEVICE\_ADDRESS,

device.getAddress());

if (mScanning) {

mBluetoothLeScanner.stopScan(scanCallback);

mScanning = false;

btnScan.setEnabled(true);

}

startActivity(intent);

}

})

.show();

}

};

private String getBTDevieType(BluetoothDevice d){

String type = "";

switch (d.getType()){

case BluetoothDevice.DEVICE\_TYPE\_CLASSIC:

type = "DEVICE\_TYPE\_CLASSIC";

break;

case BluetoothDevice.DEVICE\_TYPE\_DUAL:

type = "DEVICE\_TYPE\_DUAL";

break;

case BluetoothDevice.DEVICE\_TYPE\_LE:

type = "DEVICE\_TYPE\_LE";

break;

case BluetoothDevice.DEVICE\_TYPE\_UNKNOWN:

type = "DEVICE\_TYPE\_UNKNOWN";

break;

default:

type = "unknown...";

}

return type;

}

@Override

protected void onResume() {

super.onResume();

if (!mBluetoothAdapter.isEnabled()) {

if (!mBluetoothAdapter.isEnabled()) {

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

startActivityForResult(enableBtIntent, RQS\_ENABLE\_BLUETOOTH);

}

}

}

@Override

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

if (requestCode == RQS\_ENABLE\_BLUETOOTH && resultCode == Activity.RESULT\_CANCELED) {

finish();

return;

}

getBluetoothAdapterAndLeScanner();

// Checks if Bluetooth is supported on the device.

if (mBluetoothAdapter == null) {

Toast.makeText(this,

"bluetoothManager.getAdapter()==null",

Toast.LENGTH\_SHORT).show();

finish();

return;

}

super.onActivityResult(requestCode, resultCode, data);

}

private void getBluetoothAdapterAndLeScanner(){

// Get BluetoothAdapter and BluetoothLeScanner.

final BluetoothManager bluetoothManager =

(BluetoothManager) getSystemService(Context.BLUETOOTH\_SERVICE);

mBluetoothAdapter = bluetoothManager.getAdapter();

mBluetoothLeScanner = mBluetoothAdapter.getBluetoothLeScanner();

mScanning = false;

}

/\*

to call startScan (ScanCallback callback),

Requires BLUETOOTH\_ADMIN permission.

Must hold ACCESS\_COARSE\_LOCATION or ACCESS\_FINE\_LOCATION permission to get results.

\*/

private void scanLeDevice(final boolean enable) {

if (enable) {

listBluetoothDevice.clear();

listBluetoothDeviceString.clear();

listViewLE.invalidateViews();

// Stops scanning after a pre-defined scan period.

mHandler.postDelayed(new Runnable() {

@Override

public void run() {

mBluetoothLeScanner.stopScan(scanCallback);

listViewLE.invalidateViews();

Toast.makeText(MainBluetooth.this,

"Scan timeout",

Toast.LENGTH\_LONG).show();

mScanning = false;

btnScan.setEnabled(true);

}

}, SCAN\_PERIOD);

//mBluetoothLeScanner.startScan(scanCallback);

//scan specified devices only with ScanFilter

ScanFilter scanFilter =

new ScanFilter.Builder()

.setServiceUuid(BluetoothLeService.ParcelUuid\_Puls\_ledService)

.build();

List<ScanFilter> scanFilters = new ArrayList<ScanFilter>();

scanFilters.add(scanFilter);

ScanSettings scanSettings =

new ScanSettings.Builder().build();

mBluetoothLeScanner.startScan(scanFilters, scanSettings, scanCallback);

mScanning = true;

btnScan.setEnabled(false);

} else {

mBluetoothLeScanner.stopScan(scanCallback);

mScanning = false;

btnScan.setEnabled(true);

}

}

private ScanCallback scanCallback = new ScanCallback() {

@Override

public void onScanResult(int callbackType, ScanResult result) {

super.onScanResult(callbackType, result);

//TODO

addBluetoothDevice(result.getDevice());

}

@Override

public void onBatchScanResults(List<ScanResult> results) {

super.onBatchScanResults(results);

for(ScanResult result : results){

addBluetoothDevice(result.getDevice());

}

}

@Override

public void onScanFailed(int errorCode) {

super.onScanFailed(errorCode);

Toast.makeText(MainBluetooth.this,

"onScanFailed: " + String.valueOf(errorCode),

Toast.LENGTH\_LONG).show();

}

private void addBluetoothDevice(BluetoothDevice device){

if(!listBluetoothDevice.contains(device)){

listBluetoothDevice.add(device);

listBluetoothDeviceString.add(device.getName());

listViewLE.invalidateViews();

}

}

};

public void oBacking(View view) {

final Intent intent = new Intent(MainBluetooth.this,

MainActivity.class);

if (mScanning) {

mBluetoothLeScanner.stopScan(scanCallback);

mScanning = false;

btnScan.setEnabled(true);

}

startActivity(intent);

}

}

# Приложение В.

Код класса ControlActivity.

package com.example.budilnik;  
  
import static com.example.budilnik.BluetoothLeService.*String\_Name\_switchChar*;  
import static com.example.budilnik.BluetoothLeService.*heartRate*;  
import android.bluetooth.BluetoothGattCharacteristic;  
import android.bluetooth.BluetoothGattService;  
import android.content.BroadcastReceiver;  
import android.content.ComponentName;  
import android.content.Context;  
import android.content.Intent;  
import android.content.IntentFilter;  
import android.content.ServiceConnection;  
import android.opengl.Visibility;  
import android.os.Bundle;  
import android.os.Environment;  
import android.os.IBinder;  
import android.util.Log;  
import android.view.View;  
import android.widget.ExpandableListView;  
import android.widget.SimpleExpandableListAdapter;  
import android.widget.TextView;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.UUID;  
  
public class ControlActivity extends AppCompatActivity {  
 public static String Name\_Device;  
 final String FILENAME = "LOG.txt";  
 private final static String TAG = "321";  
 public static final String EXTRAS\_DEVICE\_NAME = "DEVICE\_NAME";  
 public static final String EXTRAS\_DEVICE\_ADDRESS = "DEVICE\_ADDRESS";  
 private static final String DIR\_SD = "MyFile";  
 private String mDeviceName;  
 private String mDeviceAddress;  
  
 private boolean mConnected = false;  
 private BluetoothGattCharacteristic mNotifyCharacteristic;  
 private BluetoothGattCharacteristic mBatteryNotifyCharacteristic;  
 private BluetoothGattCharacteristic mPulsNotifyCharacteristic;  
 public static BluetoothLeService mBluetoothLeService;  
 static TextView *textViewState*;  
 private ExpandableListView mGattServicesList;  
 private final String LIST\_NAME = "Пульсометр";  
 private final String LIST\_UUID = "0000180d-0000-1000-8000-00805f9b34fb";  
 private ArrayList<ArrayList<BluetoothGattCharacteristic>> mGattCharacteristics = new ArrayList<ArrayList<BluetoothGattCharacteristic>>();  
 private final ServiceConnection mServiceConnection = new ServiceConnection() {  
 @Override  
 public void onServiceConnected(ComponentName componentName, IBinder service) {  
 *mBluetoothLeService* = ((BluetoothLeService.LocalBinder) service).getService();  
 if (!*mBluetoothLeService*.initialize()) {  
 Log.*e*(*TAG*, "Unable to initialize Bluetooth");  
 finish();  
 }  
 // Automatically connects to the device upon successful start-up initialization.  
 *mBluetoothLeService*.connect(mDeviceAddress);  
 }  
  
 @Override  
 public void onServiceDisconnected(ComponentName componentName) {  
 *mBluetoothLeService* = null;  
 }  
 };  
  
 // Handles various events fired by the Service.  
 // ACTION\_GATT\_CONNECTED: connected to a GATT server.  
 // ACTION\_GATT\_DISCONNECTED: disconnected from a GATT server.  
 // ACTION\_GATT\_SERVICES\_DISCOVERED: discovered GATT services.  
 // ACTION\_DATA\_AVAILABLE: received data from the device. This can be a result of read  
 // or notification operations.  
 private final BroadcastReceiver mGattUpdateReceiver = new BroadcastReceiver() {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
 final String action = intent.getAction();  
 if (BluetoothLeService.ACTION\_GATT\_CONNECTED.equals(action)) {  
 mConnected = true;  
 updateConnectionState("GATT\_CONNECTED");  
 } else if (BluetoothLeService.ACTION\_GATT\_DISCONNECTED.equals(action)) {  
 mConnected = false;  
 updateConnectionState("GATT\_DISCONNECTED");  
 View backing = findViewById(R.id.Backing);  
 View bar = findViewById(R.id.progressBar);  
 backing.setVisibility(View.VISIBLE);  
 bar.setVisibility(View.GONE);  
 clearUI();  
 } else if (BluetoothLeService.ACTION\_GATT\_SERVICES\_DISCOVERED.equals(action)) {  
 // Show all the supported services and characteristics on the user interface.  
 displayGattServices(mBluetoothLeService.getSupportedGattServices());  
 } else if (BluetoothLeService.ACTION\_DATA\_AVAILABLE.equals(action)) {  
 }  
 }  
 };  
  
  
 private void clearUI() {  
 mGattServicesList.setAdapter((SimpleExpandableListAdapter) null);  
 }  
  
 private void updateConnectionState(final String st) {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 textViewState.setText(st);  
 }  
 });  
 }  
  
 public static void displayData(String data) {  
 if (data != null) {  
 textViewState.setText(data);  
 }  
 }  
  
 // Demonstrates how to iterate through the supported GATT Services/Characteristics.  
 // In this sample, we populate the data structure that is bound to the ExpandableListView  
 // on the UI.  
 private void displayGattServices(List<BluetoothGattService> gattServices) {  
  
 if (gattServices == null) return;  
 String uuid = null;  
 String unknownServiceString = "Unknown Service";  
 String unknownCharaString = "Unknown Characteristic";  
 ArrayList<HashMap<String, String>> gattServiceData =  
 new ArrayList<HashMap<String, String>>();  
 ArrayList<ArrayList<HashMap<String, String>>> gattCharacteristicData  
 = new ArrayList<ArrayList<HashMap<String, String>>>();  
 mGattCharacteristics = new ArrayList<ArrayList<BluetoothGattCharacteristic>>();  
  
  
 // Loops through available GATT Services.  
 for (BluetoothGattService gattService : gattServices) {  
 HashMap<String, String> currentServiceData = new HashMap<String, String>();  
 uuid = gattService.getUuid().toString();  
 currentServiceData.put(  
 LIST\_NAME, lookup(uuid, unknownServiceString));  
 currentServiceData.put(LIST\_UUID, uuid);  
 gattServiceData.add(currentServiceData);  
  
 ArrayList<HashMap<String, String>> gattCharacteristicGroupData =  
 new ArrayList<HashMap<String, String>>();  
 List<BluetoothGattCharacteristic> gattCharacteristics =  
 gattService.getCharacteristics();  
 ArrayList<BluetoothGattCharacteristic> charas =  
 new ArrayList<BluetoothGattCharacteristic>();  
  
 // Loops through available Characteristics.  
 for (BluetoothGattCharacteristic gattCharacteristic : gattCharacteristics) {  
 charas.add(gattCharacteristic);  
 HashMap<String, String> currentCharaData = new HashMap<String, String>();  
 uuid = gattCharacteristic.getUuid().toString();  
 currentCharaData.put(  
 LIST\_NAME, lookup(uuid, unknownCharaString));  
 currentCharaData.put(LIST\_UUID, uuid);  
 if (uuid==String\_Name\_switchChar){  
 Name\_Device= String.valueOf(gattCharacteristic.getValue());  
 }  
 gattCharacteristicGroupData.add(currentCharaData);  
  
  
 }  
 mGattCharacteristics.add(charas);  
 gattCharacteristicData.add(gattCharacteristicGroupData);  
  
 }  
 final BluetoothGattCharacteristic mBateriCharakteristic =  
 mGattCharacteristics.get(6).get(0);  
  
 mBluetoothLeService.readCharacteristic(mBateriCharakteristic);  
  
 mBluetoothLeService.setCharacteristicNotification(  
 mBateriCharakteristic, true);  
 mBatteryNotifyCharacteristic = mBateriCharakteristic;  
 try {  
 Thread.sleep(100);  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 final BluetoothGattCharacteristic mPulsCharakteristic =  
 mGattCharacteristics.get(5).get(0);  
  
 mBluetoothLeService.readCharacteristic( mPulsCharakteristic);  
  
 mBluetoothLeService.setCharacteristicNotification(  
 mPulsCharakteristic, true);  
 mPulsNotifyCharacteristic = mPulsCharakteristic;  
 if (heartRate!=0){  
MainActivity.flagVklI=true;}  
  
 BackingGl();  
 SimpleExpandableListAdapter gattServiceAdapter = new SimpleExpandableListAdapter(  
 this,  
 gattServiceData,  
 android.R.layout.simple\_expandable\_list\_item\_2,  
 new String[] {LIST\_NAME, LIST\_UUID},  
 new int[] { android.R.id.text1, android.R.id.text2 },  
 gattCharacteristicData,  
 android.R.layout.simple\_expandable\_list\_item\_2,  
 new String[] {LIST\_NAME, LIST\_UUID},  
 new int[] { android.R.id.text1, android.R.id.text2 }  
 );  
 mGattServicesList.setAdapter(gattServiceAdapter);  
  
 }  
 void writeFileSD() {  
 // проверяем доступность SD  
 if (!Environment.getExternalStorageState().equals(  
 Environment.MEDIA\_MOUNTED)) {  
 Log.d(TAG, "SD-карта не доступна: " + Environment.getExternalStorageState());  
 return;  
 }  
 // получаем путь к SD  
 File sdPath = Environment.getExternalStorageDirectory();  
 // добавляем свой каталог к пути  
 sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR\_SD);  
 // создаем каталог  
 sdPath.mkdirs();  
 // формируем объект File, который содержит путь к файлу  
 File sdFile = new File(sdPath, FILENAME);  
 try {  
 // открываем поток для записи  
 BufferedWriter bw = new BufferedWriter(new FileWriter(sdFile));  
 // пишем данные  
 bw.write("Содержимое файла на SD\n");  
 // закрываем поток  
 bw.close();  
 Log.d(TAG, "Файл записан на SD: " + sdFile.getAbsolutePath());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private final ExpandableListView.OnChildClickListener servicesListClickListner = new ExpandableListView.OnChildClickListener() {  
 @Override  
 public boolean onChildClick(ExpandableListView parent, View v, int groupPosition,  
 int childPosition, long id) {  
 if (mGattCharacteristics != null) {  
 writeFileSD();  
 final BluetoothGattCharacteristic characteristic =  
 mGattCharacteristics.get(groupPosition).get(childPosition);  
  
 final int charaProp = characteristic.getProperties();  
 if ((charaProp | BluetoothGattCharacteristic.PROPERTY\_READ) > 0) {  
 // If there is an active notification on a characteristic, clear  
 // it first so it doesn't update the data field on the user interface.  
 if (mNotifyCharacteristic != null) {  
 mBluetoothLeService.setCharacteristicNotification(  
 mNotifyCharacteristic, false);  
 mNotifyCharacteristic = null;  
 }  
  
 mBluetoothLeService.readCharacteristic(characteristic);  
 //TODO characteristic.getValue();  
 }  
 if ((charaProp | BluetoothGattCharacteristic.PROPERTY\_NOTIFY) > 0) {  
  
  
  
  
 mNotifyCharacteristic = characteristic;  
 mBluetoothLeService.setCharacteristicNotification(  
 characteristic, true);  
  
  
 }  
 return true;  
  
 }  
 return false;  
 }  
 };  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_control);  
 View backing = findViewById(R.id.Backing);  
 View bar = findViewById(R.id.progressBar);  
 bar.setVisibility(View.VISIBLE);  
 backing.setVisibility(View.GONE);  
  
 startService(new Intent(this, BluetoothLeService.class));  
 final Intent intent = getIntent();  
 mDeviceName = intent.getStringExtra(EXTRAS\_DEVICE\_NAME);  
 mDeviceAddress = intent.getStringExtra(EXTRAS\_DEVICE\_ADDRESS);  
  
 TextView textViewDeviceName = (TextView)findViewById(R.id.textDeviceName);  
 TextView textViewDeviceAddr = (TextView)findViewById(R.id.textDeviceAddress);  
 textViewState = (TextView)findViewById(R.id.textState);  
  
 textViewDeviceName.setText(mDeviceName);  
 Name\_Device=mDeviceName;  
 textViewDeviceAddr.setText(mDeviceAddress);  
  
 mGattServicesList = (ExpandableListView) findViewById(R.id.gatt\_services\_list);  
 mGattServicesList.setOnChildClickListener(servicesListClickListner);  
  
 Intent gattServiceIntent = new Intent(this, BluetoothLeService.class);  
 bindService(gattServiceIntent, mServiceConnection, BIND\_AUTO\_CREATE);  
  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 registerReceiver(mGattUpdateReceiver, makeGattUpdateIntentFilter());  
 if (mBluetoothLeService != null) {  
 final boolean result = mBluetoothLeService.connect(mDeviceAddress);  
 Log.d(TAG, "Connect request result=" + result);  
 }  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 unregisterReceiver(mGattUpdateReceiver);  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
 unbindService(mServiceConnection);  
 //mBluetoothLeService = null;  
 }  
 private static IntentFilter makeGattUpdateIntentFilter() {  
 final IntentFilter intentFilter = new IntentFilter();  
 intentFilter.addAction(BluetoothLeService.ACTION\_GATT\_CONNECTED);  
 intentFilter.addAction(BluetoothLeService.ACTION\_GATT\_DISCONNECTED);  
 intentFilter.addAction(BluetoothLeService.ACTION\_GATT\_SERVICES\_DISCOVERED);  
 intentFilter.addAction(BluetoothLeService.ACTION\_DATA\_AVAILABLE);  
 return intentFilter;  
 }  
 private static HashMap<String, String> attributes = new HashMap();  
 public void oBacking(View view) {  
 BackingGl();  
 }  
 public void BackingGl(){  
 final Intent intent = new Intent(ControlActivity.this,  
 MainActivity.class);  
 startActivity(intent);  
 }  
}

# Приложение Г.

Код класса AlarmActivity.

package com.example.budilnik;  
  
import android.annotation.SuppressLint;  
import android.app.AlarmManager;  
import android.app.PendingIntent;  
import android.content.Context;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.media.Ringtone;  
import android.media.RingtoneManager;  
import android.net.Uri;  
import android.os.Bundle;  
import android.os.Vibrator;  
import android.view.View;  
import android.widget.Switch;  
import androidx.appcompat.app.AppCompatActivity;  
import org.jetbrains.annotations.Nullable;  
  
public class AlarmActivity extends AppCompatActivity {  
  
 Ringtone ringtone;  
 Vibrator vibrator;  
 public static Integer *hour*;  
 public static Integer *min*;  
 public boolean flagVkl;  
 static Switch *switchView*;  
  
 @Override  
 protected void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 if (MainActivity.*hour*!=null&&MainActivity.*min*!=null){  
 *hour*=MainActivity.*hour*;  
 *min*=MainActivity.*min*;  
 }else {  
 SharedPreferences savedData = getSharedPreferences("SavedData",*MODE\_PRIVATE*);  
  
 *hour*=savedData.getInt("Hours",0);  
 *min*=savedData.getInt("Min",0);  
 }  
 otmena();  
 vibrator = (Vibrator)getSystemService(Context.*VIBRATOR\_SERVICE*);  
  
  
 setContentView(R.layout.*activity\_alarm*);  
  
 Uri notificationUri = RingtoneManager.*getDefaultUri*(RingtoneManager.*TYPE\_ALARM*);  
 ringtone = RingtoneManager.*getRingtone*(this,notificationUri);  
 if (ringtone==null){  
 notificationUri = RingtoneManager.*getDefaultUri*(RingtoneManager.*TYPE\_RINGTONE*);  
 ringtone = RingtoneManager.*getRingtone*(this,notificationUri);  
 }  
 if (ringtone!=null){  
 ringtone.play();  
 }  
 long[] pattern = {0, 2000, 1000, 2500, 1500, 3000};  
 vibrator.vibrate(pattern,1);  
  
 }  
  
 @Override  
 protected void onStop() {  
 super.onStop();  
 otmena();  
 }  
  
 @Override  
 protected void onDestroy() {  
 SharedPreferences savedData = getSharedPreferences("SavedData",*MODE\_PRIVATE*);//сохранение данных  
 flagVkl=false;  
 SharedPreferences.Editor editor=savedData.edit();//сохранение данных  
 editor.putBoolean("flag",flagVkl);  
 editor.apply();  
 if (ringtone!=null&&ringtone.isPlaying()){  
 ringtone.stop();  
  
 }  
 vibrator.cancel();  
 MainActivity.*hour*=this.*hour*;  
 MainActivity.*min*=this.*min*;  
 otmena();  
 super.onDestroy();  
 }  
  
  
 public void cansel(View view) {  
 Intent intent = new Intent(this,MainActivity.class);  
 if (ringtone!=null&&ringtone.isPlaying()){  
 ringtone.stop();  
  
 }  
 vibrator.cancel();  
 otmena();  
 startActivity(intent);  
 }  
  
 public void otmena(){  
SharedPreferences savedData = getSharedPreferences("SavedData",*MODE\_PRIVATE*);//сохранение данных  
 *hour*=savedData.getInt("Hours",0);  
 *min*=savedData.getInt("Min",0);  
 *switchView*=findViewById(R.id.*switch\_bn1*);  
 MainActivity.*alarmManager* = (AlarmManager) getSystemService(Context.*ALARM\_SERVICE*);  
 if (getAlarmActionPendingIntent()!=null){  
 MainActivity.*alarmManager*.cancel(getAlarmActionPendingIntent());}//Отменяем будильник, связанный с интентом данного класса  
 // Toast.makeText(this, "Будильник отменён", Toast.LENGTH\_LONG).show();  
 MainActivity.*flagVkl*=false;  
 MainActivity.switchView.setChecked(false);  
 SharedPreferences.Editor editor=savedData.edit();//сохранение данных  
 editor.putInt("Hours", hour);  
 editor.putInt("Min", min);  
 editor.putBoolean("flag",flagVkl);  
 editor.apply();  
 }  
  
 @SuppressLint("UnspecifiedImmutableFlag")  
 private PendingIntent getAlarmInfoPendingIntent(){  
 Intent alarmInfoIntent = new Intent(this,MainActivity.class);  
 alarmInfoIntent.setFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP|Intent.FLAG\_ACTIVITY\_NEW\_TASK);  
 return PendingIntent.getActivity(this,0,alarmInfoIntent,PendingIntent.FLAG\_UPDATE\_CURRENT);  
 }  
 @SuppressLint("UnspecifiedImmutableFlag")  
 PendingIntent getAlarmActionPendingIntent(){  
 Intent intent = new Intent(this,AlarmActivity.class);  
 intent.setFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP|Intent.FLAG\_ACTIVITY\_NEW\_TASK);  
 return PendingIntent.getActivity(this,1,intent,PendingIntent.FLAG\_UPDATE\_CURRENT);  
 }  
}

# Приложение Д.

Код класса BluetoothLeService.

package com.example.budilnik;  
  
import static androidx.core.app.NotificationCompat.*PRIORITY\_DEFAULT*;  
import static androidx.core.app.NotificationCompat.*PRIORITY\_HIGH*;  
import static androidx.core.app.NotificationCompat.*PRIORITY\_MAX*;  
import android.annotation.SuppressLint;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.app.Service;  
import android.bluetooth.BluetoothAdapter;  
import android.bluetooth.BluetoothDevice;  
import android.bluetooth.BluetoothGatt;  
import android.bluetooth.BluetoothGattCallback;  
import android.bluetooth.BluetoothGattCharacteristic;  
import android.bluetooth.BluetoothGattDescriptor;  
import android.bluetooth.BluetoothGattService;  
import android.bluetooth.BluetoothManager;  
import android.bluetooth.BluetoothProfile;  
import android.content.Context;  
import android.content.Intent;  
import android.content.IntentFilter;  
import android.os.BatteryManager;  
import android.os.Binder;  
import android.os.Build;  
import android.os.Environment;  
import android.os.IBinder;  
import android.os.ParcelUuid;  
import android.util.Log;  
import android.view.View;  
import androidx.core.app.NotificationCompat;  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.nio.charset.StandardCharsets;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.List;  
import java.util.UUID;  
public class BluetoothLeService extends Service {  
 public float batteryPct=0;  
 private final static String *TAG* = "123";  
 private static final String *DIR\_SD* = "MyFile";  
 final String FILENAME = "LOG.txt";  
 public byte[] data;  
 private NotificationManager notificationManager;  
 // Идентификатор уведомления  
 private static final int *NOTIFY\_ID* = 101;  
 private static final int *NOTIFYZU\_ID* = 102;  
 private static final int *NOTIFYZT\_ID* = 103;  
 private static final int *NOTIFYO\_ID* = 104;  
 // Идентификаторы каналов уведомлений  
 private static final String *CHANNEL\_ID* = "Оповещения о входящих данных";  
 private static final String *CHANNELZU\_ID* = "Оповещение о малом уровне заряда пульсометра";  
 private static final String *CHANNELZT\_ID* = "Оповещение о недостаточном уровне заряда телефона";  
 private static final String *CHANNELO\_ID* = "Оповещение об отключении пульсометра от телефона";  
 private BluetoothManager mBluetoothManager;  
 private BluetoothAdapter mBluetoothAdapter;  
 private String mBluetoothDeviceAddress;  
 private BluetoothGatt mBluetoothGatt;  
 private int mConnectionState = *STATE\_DISCONNECTED*;  
 public static int *heartRate*;  
 private static final int *STATE\_DISCONNECTED* = 0;  
 private static final int *STATE\_CONNECTING* = 1;  
 private static final int *STATE\_CONNECTED* = 2;  
 public final static String *ACTION\_GATT\_CONNECTED* =  
 "android-er.ACTION\_GATT\_CONNECTED";  
 public final static String *ACTION\_GATT\_DISCONNECTED* =  
 "android-er.ACTION\_GATT\_DISCONNECTED";  
 public final static String *ACTION\_GATT\_SERVICES\_DISCOVERED* =  
 "android-er.ACTION\_GATT\_SERVICES\_DISCOVERED";  
 public final static String *ACTION\_DATA\_AVAILABLE* =  
 "android-er.ACTION\_DATA\_AVAILABLE";  
 public final static String *EXTRA\_DATA* =  
 "android-er.EXTRA\_DATA";  
 public static Integer *timer*=0;  
 private static final UUID *Battery\_Level\_UUID* =  
 UUID.*fromString*("00002a19-0000-1000-8000-00805f9b34fb");  
 public static String *String\_Name\_ledService* =  
 "00001800-0000-1000-8000-00805f9b34fb";  
 public static String *String\_Name\_switchChar* = "00002a00-0000-1000-8000-00805f9b34fb";  
 public final static UUID *UUID\_Name\_switchChare* =  
 UUID.*fromString*(*String\_Name\_switchChar*);  
 public static String *String\_Puls\_ledService* =  
 "0000180d-0000-1000-8000-00805f9b34fb";  
 public final static ParcelUuid *ParcelUuid\_Puls\_ledService* =  
 ParcelUuid.*fromString*(*String\_Puls\_ledService*);  
 public static String *String\_Puls\_switchChar* = "00002a37-0000-1000-8000-00805f9b34fb";  
 public final static UUID *UUID\_Puls\_switchChare* =  
 UUID.*fromString*(*String\_Puls\_switchChar*);  
 private byte zaryd=0;  
 public BluetoothDevice device;  
 private int priv=0;  
 private boolean zarydFlag=true;  
 private boolean zarydFlag2=true;  
  
  
 @SuppressLint("MissingPermission")  
 public void readCharacteristic(BluetoothGattCharacteristic characteristic) {  
 if (mBluetoothAdapter == null || mBluetoothGatt == null) {  
 Log.*w*(*TAG*, "BluetoothAdapter not initialized");  
 return;  
 }  
 mBluetoothGatt.readCharacteristic(characteristic);  
 }  
 @SuppressLint("MissingPermission")  
 public void writeCharacteristic() {  
 if (mBluetoothAdapter == null || mBluetoothGatt == null) {  
 Log.*w*(*TAG*, "BluetoothAdapter not initialized");  
 return;  
 }  
 BluetoothGattService mCustomService = mBluetoothGatt.getService(UUID.*fromString*(*String\_Name\_ledService*));  
 if(mCustomService == null){  
 Log.*w*(*TAG*, "Custom BLE Service not found");  
 return;  
 }  
  
 BluetoothGattCharacteristic characteristic1 = mCustomService.getCharacteristic(*UUID\_Name\_switchChare*);  
 byte[] value = ControlActivity.*Name\_Device*.getBytes();  
 characteristic1.setValue(value);  
 mBluetoothGatt.writeCharacteristic(characteristic1);  
 }  
 @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
  
 writeCharacteristic();  
 return *START\_REDELIVER\_INTENT*;  
 }  
 private final BluetoothGattCallback mGattCallback = new BluetoothGattCallback() {  
 @SuppressLint("MissingPermission")  
 @Override  
 public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {  
 String intentAction;  
 if (newState == BluetoothProfile.*STATE\_CONNECTED*) {  
 intentAction = *ACTION\_GATT\_CONNECTED*;  
 mConnectionState = *STATE\_CONNECTED*;  
 broadcastUpdate(intentAction);  
 Log.*i*(*TAG*, "Connected to GATT server.");  
 // Attempts to discover services after successful connection.  
 Log.*i*(*TAG*, "Attempting to start service discovery:" +  
 mBluetoothGatt.discoverServices());  
  
 } else if (newState == BluetoothProfile.*STATE\_DISCONNECTED*) {  
 intentAction = *ACTION\_GATT\_DISCONNECTED*;  
 mConnectionState = *STATE\_DISCONNECTED*;  
 Log.*i*(*TAG*, "Disconnected from GATT server.");  
 notifykO();  
 broadcastUpdate(intentAction);  
 }  
 }  
  
 @Override  
 public void onServicesDiscovered(BluetoothGatt gatt, int status) {  
 if (status == BluetoothGatt.*GATT\_SUCCESS*) {  
 broadcastUpdate(*ACTION\_GATT\_SERVICES\_DISCOVERED*);  
 } else {  
 Log.*w*(*TAG*, "onServicesDiscovered received: " + status);  
 }  
 }  
  
 @Override  
 public void onCharacteristicRead(BluetoothGatt gatt,  
 BluetoothGattCharacteristic characteristic,  
 int status) {  
 if (status == BluetoothGatt.*GATT\_SUCCESS*) {  
 try {  
 broadcastUpdate(*ACTION\_DATA\_AVAILABLE*, characteristic);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 }  
  
 @Override  
 public void onCharacteristicChanged(BluetoothGatt gatt,  
 BluetoothGattCharacteristic characteristic) {  
 try {  
 broadcastUpdate(*ACTION\_DATA\_AVAILABLE*, characteristic);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 };  
 private void broadcastUpdate(final String action) {  
 final Intent intent = new Intent(action);  
 sendBroadcast(intent);  
 }  
 @SuppressLint("SetTextI18n")  
 private void broadcastUpdate(final String action,  
 final BluetoothGattCharacteristic characteristic) throws IOException {  
 final Intent intent = new Intent(action);  
  
 UUID uuid = characteristic.getUuid();  
 if (*Battery\_Level\_UUID*.equals(uuid)) {  
 data = characteristic.getValue();  
 if (data != null && data.length > 0) {  
 zaryd = data[0];  
 intent.putExtra(*EXTRA\_DATA*, new String(data, StandardCharsets.*UTF\_8*) + "\n" + data[0]);  
 Log.*d*(*TAG*, "Zaryd" + new String(data, StandardCharsets.*UTF\_8*) + " " + data[0]+"gh "+batteryPct+"%");  
 return;  
 }  
 } else if (*UUID\_Puls\_switchChare*.equals(uuid)) {  
 if (*timer* >= 10) {  
 writeCharacteristic();  
 *timer* = 0;  
 notifyk();  
 } else {  
 *timer* = *timer* + 1;  
 Log.*d*(*TAG*, String.*valueOf*(*timer*));  
 }  
 if (MainActivity.*flagVklB*) {  
 budilnik();  
 }  
 int flag = characteristic.getProperties();  
 int format = -1;  
 if ((flag & 0x01) != 0) {  
 format = BluetoothGattCharacteristic.*FORMAT\_UINT16*;  
 Log.*d*(*TAG*, "Heart rate format UINT16.");  
 } else {  
 format = BluetoothGattCharacteristic.*FORMAT\_UINT8*;  
 Log.*d*(*TAG*, "Heart rate format UINT8.");  
 }  
 *heartRate* = characteristic.getIntValue(format, 1);  
 Log.*d*(*TAG*, String.*format*("Received heart rate: %d", *heartRate*));  
 IntentFilter ifilter = new IntentFilter(Intent.*ACTION\_BATTERY\_CHANGED*);  
 Intent batteryStatus = this.registerReceiver(null, ifilter);  
 int level = batteryStatus.getIntExtra(BatteryManager.*EXTRA\_LEVEL*, -1);  
 int scale = batteryStatus.getIntExtra(BatteryManager.*EXTRA\_SCALE*, -1);  
 batteryPct = level \* 100 / (float)scale;  
 Date cutTime = Calendar.*getInstance*().getTime();  
 if (cutTime.getHours()>20){  
 if (batteryPct<60){  
 if (zarydFlag){  
 notifykZT();  
 zarydFlag=false;}  
 }else {  
 zarydFlag=true;  
 }}  
 if (zaryd<4){  
 if (zarydFlag2){  
 notifykZP();  
 zarydFlag2=false;}  
 else{  
 zarydFlag2=true;  
 }  
 }  
 // проверяем доступность SD  
 if (!Environment.*getExternalStorageState*().equals(  
 Environment.*MEDIA\_MOUNTED*)) {  
 Log.*d*(*TAG*, "SD-карта не доступна: " + Environment.*getExternalStorageState*());  
 return;  
 }  
 // получаем путь к SD  
 File sdPath = Environment.*getExternalStorageDirectory*();  
 // добавляем свой каталог к пути  
 sdPath = new File(sdPath.getAbsolutePath() + "/" + *DIR\_SD*);  
 // создаем каталог  
 sdPath.mkdirs();  
 // формируем объект File, который содержит путь к файлу  
 File sdFile = new File(sdPath, FILENAME);  
  
 // отрываем поток для записи  
 BufferedWriter bw = new BufferedWriter(new FileWriter(sdFile, true));  
 // пишем данные  
 SimpleDateFormat sdf = new SimpleDateFormat("'Date :'dd.MM' and Time :'HH.mm.ss z");  
 String currentDateAndTime = sdf.format(new Date());  
 bw.write("Пульс "+*heartRate* + " : " + currentDateAndTime +" Заряд "+zaryd+ " Заряд телефона "+batteryPct+ "%\n");  
 // закрываем поток  
 bw.close();  
 Log.*d*(*TAG*, "Файл записан" + sdFile.getAbsolutePath());  
  
 intent.putExtra(*EXTRA\_DATA*, String.*valueOf*(*heartRate*));  
  
 } else {// For all other profiles, writes the data formatted in HEX.  
 data = characteristic.getValue();  
 if (data != null && data.length > 0) {  
  
 intent.putExtra(*EXTRA\_DATA*, new String(data, StandardCharsets.*UTF\_8*) + "\n" + data[0]);  
 Log.*d*(*TAG*, "{{{}}}}" + new String(data, StandardCharsets.*UTF\_8*) + " " + data[0]);  
 }  
 }  
  
 //remove special handling for time being  
 Log.*w*(*TAG*, "broadcastUpdate()");  
  
  
  
 sendBroadcast(intent);  
  
  
 //ControlActivity.displayData(intent.getStringExtra(BluetoothLeService.EXTRA\_DATA));  
  
 }  
 public static void createChannelIfNeeded(NotificationManager manager){  
 if(Build.VERSION.*SDK\_INT*>=Build.VERSION\_CODES.*O*){  
 NotificationChannel notificationChannel=new NotificationChannel(*CHANNEL\_ID*,*CHANNEL\_ID*, NotificationManager.*IMPORTANCE\_LOW*);  
 manager.createNotificationChannel(notificationChannel);  
 NotificationChannel notificationChannelZU=new NotificationChannel(*CHANNELZU\_ID*,*CHANNELZU\_ID*, NotificationManager.*IMPORTANCE\_DEFAULT*);  
 manager.createNotificationChannel(notificationChannelZU);  
 NotificationChannel notificationChannelZT=new NotificationChannel(*CHANNELZT\_ID*,*CHANNELZT\_ID*, NotificationManager.*IMPORTANCE\_DEFAULT*);  
 manager.createNotificationChannel(notificationChannelZT);  
 NotificationChannel notificationChannelO=new NotificationChannel(*CHANNELO\_ID*,*CHANNELO\_ID*, NotificationManager.*IMPORTANCE\_DEFAULT*);  
 manager.createNotificationChannel(notificationChannelO);  
 }  
}  
 public class LocalBinder extends Binder {  
 BluetoothLeService getService() {  
 return BluetoothLeService.this;  
 }  
 }  
 @Override  
 public IBinder onBind(Intent intent) {  
 return mBinder;  
 }  
 @Override  
 public boolean onUnbind(Intent intent) {  
 close();  
 return super.onUnbind(intent);  
 }  
 private final IBinder mBinder = new LocalBinder();  
 public boolean initialize() {  
 if (mBluetoothManager == null) {  
 mBluetoothManager = (BluetoothManager) getSystemService(Context.*BLUETOOTH\_SERVICE*);  
 if (mBluetoothManager == null) {  
 Log.*e*(*TAG*, "Unable to initialize BluetoothManager.");  
 return false;  
 }  
 }  
  
 mBluetoothAdapter = mBluetoothManager.getAdapter();  
 if (mBluetoothAdapter == null) {  
 Log.*e*(*TAG*, "Unable to obtain a BluetoothAdapter.");  
 return false;  
 }  
  
 return true;  
 }  
 @SuppressLint("MissingPermission")  
 public boolean connect(final String address) {  
 if (mBluetoothAdapter == null || address == null) {  
 Log.*w*(*TAG*, "BluetoothAdapter not initialized or unspecified address.");  
 return false;  
 }  
  
 // Previously connected device. Try to reconnect.  
 if (mBluetoothDeviceAddress != null && address.equals(mBluetoothDeviceAddress)  
 && mBluetoothGatt != null) {  
 Log.*d*(*TAG*, "Trying to use an existing mBluetoothGatt for connection.");  
 if (mBluetoothGatt.connect()) {  
 mConnectionState = *STATE\_CONNECTING*;  
 return true;  
 } else {  
 return false;  
 }  
 }  
  
 device = mBluetoothAdapter.getRemoteDevice(address);  
 if (device == null) {  
 Log.*w*(*TAG*, "Device not found. Unable to connect.");  
 return false;  
 }  
 // We want to directly connect to the device, so we are setting the autoConnect  
 // parameter to false.  
 mBluetoothGatt = device.connectGatt(this, false, mGattCallback);  
 Log.*d*(*TAG*, "Trying to create a new connection.");  
 mBluetoothDeviceAddress = address;  
 mConnectionState = *STATE\_CONNECTING*;  
  
 return true;  
 }  
 @SuppressLint("MissingPermission")  
 public void disconnect() {  
 if (mBluetoothAdapter == null || mBluetoothGatt == null) {  
 Log.*w*(*TAG*, "BluetoothAdapter not initialized");  
 return;  
 }  
 mBluetoothGatt.disconnect();  
 }  
 @SuppressLint("MissingPermission")  
 public void close() {  
 if (mBluetoothGatt == null) {  
 return;  
 }  
 mBluetoothGatt.close();  
 mBluetoothGatt = null;  
 }  
 @SuppressLint("MissingPermission")  
 public void setCharacteristicNotification(BluetoothGattCharacteristic characteristic, boolean enabled)  
 {  
 if (mBluetoothAdapter == null || mBluetoothGatt == null) {  
 Log.*w*(*TAG*, "BluetoothAdapter not initialized");  
 return;  
 }  
 mBluetoothGatt.setCharacteristicNotification(characteristic, enabled);  
  
 // This is specific to Heart Rate Measurement.  
 if (*UUID\_Puls\_switchChare*.equals(characteristic.getUuid())) {  
 BluetoothGattDescriptor descriptor = characteristic.getDescriptor(UUID.*fromString*("00002902-0000-1000-8000-00805f9b34fb"));  
 descriptor.setValue(BluetoothGattDescriptor.*ENABLE\_NOTIFICATION\_VALUE*);  
 mBluetoothGatt.writeDescriptor(descriptor);  
 }  
 }  
 public List<BluetoothGattService> getSupportedGattServices() {  
 if (mBluetoothGatt == null) return null;  
  
 return mBluetoothGatt.getServices();  
 }  
 public void notifyk(){  
  
 //readCustomCharacteristic();  
 notificationManager=(NotificationManager) getApplicationContext().getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 Intent intent1 = new Intent(getApplicationContext(),BluetoothLeService.class);  
 intent1.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*|Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 PendingIntent pendingIntent = PendingIntent.*getActivity*(getApplicationContext(),0,intent1,PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
 NotificationCompat.Builder notificationBulder=  
 new NotificationCompat.Builder(getApplicationContext(),*CHANNEL\_ID*)  
 .setAutoCancel(false)  
 .setSmallIcon(R.drawable.*ic\_stat\_name*)  
 .setWhen(System.*currentTimeMillis*())  
 .setContentIntent(pendingIntent)  
 .setContentTitle("Входные данные")  
 .setContentText("Пульс = "+ *heartRate*+"\nЗаряд = "+zaryd+"\nЗаряд телефона = "+batteryPct)  
 .setPriority(*PRIORITY\_MAX*);  
 *createChannelIfNeeded*(notificationManager);  
 notificationManager.notify(*NOTIFY\_ID*,notificationBulder.build());  
 MainActivity.*heartRate*=*heartRate*;  
 }  
 public void notifykZP(){  
  
 //readCustomCharacteristic();  
 notificationManager=(NotificationManager) getApplicationContext().getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 Intent intent1 = new Intent(getApplicationContext(),BluetoothLeService.class);  
 intent1.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*|Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 PendingIntent pendingIntent = PendingIntent.*getActivity*(getApplicationContext(),0,intent1,PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
 NotificationCompat.Builder notificationBulder=  
 new NotificationCompat.Builder(getApplicationContext(),*CHANNELZU\_ID*)  
 .setAutoCancel(false)  
 .setSmallIcon(R.drawable.*ic\_chard\_name*)  
 .setWhen(System.*currentTimeMillis*())  
 .setContentIntent(pendingIntent)  
 .setContentTitle("Заряд пульсометра кончается")  
 .setContentText("Заряда пульсометра скорее всего не хватит на всю ночь")  
 .setPriority(*PRIORITY\_HIGH*);  
 *createChannelIfNeeded*(notificationManager);  
 notificationManager.notify(*NOTIFYZU\_ID*,notificationBulder.build());  
 }  
 public void notifykZT(){  
  
  
 notificationManager=(NotificationManager) getApplicationContext().getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 Intent intent1 = new Intent(getApplicationContext(),BluetoothLeService.class);  
 intent1.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*|Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 PendingIntent pendingIntent = PendingIntent.*getActivity*(getApplicationContext(),0,intent1,PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
 NotificationCompat.Builder notificationBulder=  
 new NotificationCompat.Builder(getApplicationContext(),*CHANNELZT\_ID*)  
 .setAutoCancel(false)  
 .setSmallIcon(R.drawable.*baseline\_charging\_station\_24*)  
 .setWhen(System.*currentTimeMillis*())  
 .setContentIntent(pendingIntent)  
 .setContentTitle("Заряд телефона")  
 .setContentText("Заряда телефона Скорее всего не хватит на всю ночь")  
 .setPriority(*PRIORITY\_MAX*);  
 *createChannelIfNeeded*(notificationManager);  
 notificationManager.notify(*NOTIFYZT\_ID*,notificationBulder.build());  
 }  
 public void notifykO(){  
  
  
 notificationManager=(NotificationManager) getApplicationContext().getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 Intent intent1 = new Intent(getApplicationContext(),BluetoothLeService.class);  
 intent1.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*|Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 PendingIntent pendingIntent = PendingIntent.*getActivity*(getApplicationContext(),0,intent1,PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
 NotificationCompat.Builder notificationBulder=  
 new NotificationCompat.Builder(getApplicationContext(),*CHANNELO\_ID*)  
 .setAutoCancel(false)  
 .setSmallIcon(R.drawable.*ic\_chard\_name\_disconection*)  
 .setWhen(System.*currentTimeMillis*())  
 .setContentIntent(pendingIntent)  
 .setContentTitle("Отключилось")  
 .setContentText("Пульсометр отключился от телефона.\n Для полноценной работы приложения подключите пульсометр обратно")  
 .setPriority(*PRIORITY\_DEFAULT*);  
 *createChannelIfNeeded*(notificationManager);  
 notificationManager.notify(*NOTIFYO\_ID*,notificationBulder.build());  
 }  
 public void budilnik(){  
 if (MainActivity.*flagVkl*){  
 Date cutTime = Calendar.*getInstance*().getTime();  
 if (*heartRate*>=60){  
 if (*heartRate*<=80){  
 if (priv>254){  
 priv=0;  
 if (MainActivity.*day2*<=cutTime.getDate()){  
 if (MainActivity.*hour2*<=cutTime.getHours()){  
 if (MainActivity.*min2*<=cutTime.getMinutes()){  
 MainActivity.*flagVklB*=false;  
 final Intent intent = new Intent(this,  
 AlarmActivity.class);  
 intent.setFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*|Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  
 startActivity(intent);  
 }  
 }  
 }  
 }else {  
 priv=priv+1;  
 }  
 }else {  
 priv=0;  
 }  
 }  
 else {  
 priv=0;  
 }  
 }  
}  
  
}