Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
Петрозаводский государственный университет (ПетрГУ)
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Промежуточный отчет о научно-исследовательской работе

# Анализ проблем адаптации алгоритмов упрощения ломаных на мобильных устройствах (на территории Петрозаводска)

Выполнил:
студент 4 курса группы 22403 И.С. Прохоров

_____
*подпись*

Научный руководитель:
к.т.н., доцент О.Ю. Богоявленская
Оценка руководителя: _____

_____
*подпись*

Представлен на кафедру

« ____ » _____ 2016 г.

_____
*подпись принявшего работу*

Петрозаводск

2016

# Содержание

# Введение

В настоящее время огромное число людей используют мобильные устройства не только как средства связи, но и как переносные хранилища данных, средства съемки, мультимедиа проигрыватели и т.д., почти полностью связывая свои повседневные действия с ними. В условиях сильной загруженности городских дорог транспортом крайне важно умело распределить свое время и определить маршрут, благодаря которому пользователь мог добираться вовремя в любое необходимое ему место.

Таким образом, высока вероятность востребованности программы, способной считывать и запоминать все наиболее используемые пользователем маршруты, упрощая их для сокращения времени преодоления.

Цель:

Разработка мобильного приложения, работающего на основе алгоритма упрощения ломаных.

Задачи:

1) Добавление карты в приложения для визуализации получаемых данных о дислокации.

2) Построение ломаной передвижения на основе полученных данных о местоположении пользователя.

3) Улучшение алгоритма упрощения пути следования.

# 1 Изученные материалы.

## 1.1 API android.location.

Данный интерфейс программирования приложений содержит Framework API классы, которые определяют Android сервисы для определения местоположения объекта. Из данного API были взяты классы Location и LocationManager, первый предоставляет данные о географическом положении пользователя, а второй обеспечивает доступ к сервисам системы определения местоположения. Так же был использован интерфейс LocationListener необходимый для получения уведомлений от LocationManager при изменении местоположения.

## 1.2 Google Maps Android API.

Данное API предоставляет возможность портирования в Android приложение карты, которая основанна на данных сервисов Google Maps. В приложении были использованны оба класса, предоставляющих возможность работы с картой, GoogleMap и MapFragment. GoogleMap является основным классом API Google Maps, поэтому большинство взаимодейтсвий с картой производилось с его помощью. В приложении использовался интерфейс обратного вызова OnMapReadyCallback необходимый для проверки готовности карты к работе и передачи фрагмента для взаимодействия с ней. Также использовался класс Polyline для построения маршрутов передвижений пользователя и метод addMarker() для добавления маркера с местоположением. Сама карта была портированна в приложение с помощью класса MapFragment посредством добавления его компонента fragment в xml-файл. Для создания взаимосвязи между классом MapFragment и GoogleMap использовался метод getMapAsync, возвращающий объект карты при готовности экземпляра GoogleMap.

## 1.3 Средства Android.

android.content.Intent – абстрактное описание для операции, которая должна быть выполнена. В приложении операция используется для перехода к меню включения определения местоположения на телефоне и чтения данных из файла.

android.os.Environment – класс, обеспечивающий доступ к переменным окружения. getExternalStorageState() – метод, возвращающий текущее состояние внешнего носителя по заданному пути. Используется для проверки доступа к SD-карте.

getExternalStorageDirectory() – метод для получения пути к каталогу с файлом о данных местоположения пользователя.

## 1.4 Средства языка Java.

java.text.SimpleDateFormat – класс, использующийся для форматирования и разложения даты в локально-чувствительной манере.

java.util.Scanner - простой текстовый сканер с возможностью анализа примитивных типов и строк посредством использования регулярных выражений. Использовался для считывания данных из файла о перемещении пользователя.

java.util.Timer - класс потокового планирования задач для будущего выполнения. В приложении использовался для определения переодичности выполнения записи в файл с маршрутами передвижений.

java.util.TimerTask - абстрактный класс, описанный в программе как задача по записи данных в файл.

java.io.BufferedWriter - класс для записи текста в поток вывода.

# 2 Поставленные задачи.

## 2.1 Разработка прототипа приложения.

Реализовано корректно работающее приложение, удовлетворяющее пунктам 1 и 2 в категории Задачи.

**Описание возможностей приложения:**

**Нахождение.**

Приложение получает данные о местонахождении пользователя на основе работающего GPS, вышек сотовой связи либо Wi-Fi.

**Запись.**

При нажатии в окне приложения кнопки «Start Write» выводится диалоговое окно, предоставляющее пользователю возможность выбора между обычной записью изменений его дислокации и записью с отображением маршрута на карте. Вне зависимости от выборанного способа учета изменений местоположения данные заносятся в файл. Каждая последующая запись добавляется через каждые 30 секунд до момента повторного нажатия на кнопку «Start Write». Данные записываются в файл «GPS_Coordinates.txt» с указанием времени записи, координат долготы и широты нахождения пользователя. Также создается файл «GPS_Points.txt» с численными данными о дислокации пользователя.

**Чтение.**

При нажатии на кнопку «Read File» пользователь открывает файл «GPS_Coordinates.txt», в котором указаны данные, описанные выше. Открытие файла происходит с помощью встроенной программы для чтения текстов.

**Демонстрация.**

Пользователь имеет возможность моментального получения данных о своей дислокации в виде маркера на карте с помощью нажатия на кнопку «Show Location».

## 2.2 Тесты.

Ниже представлены изображения, демонстрирующие основные функции приложения.

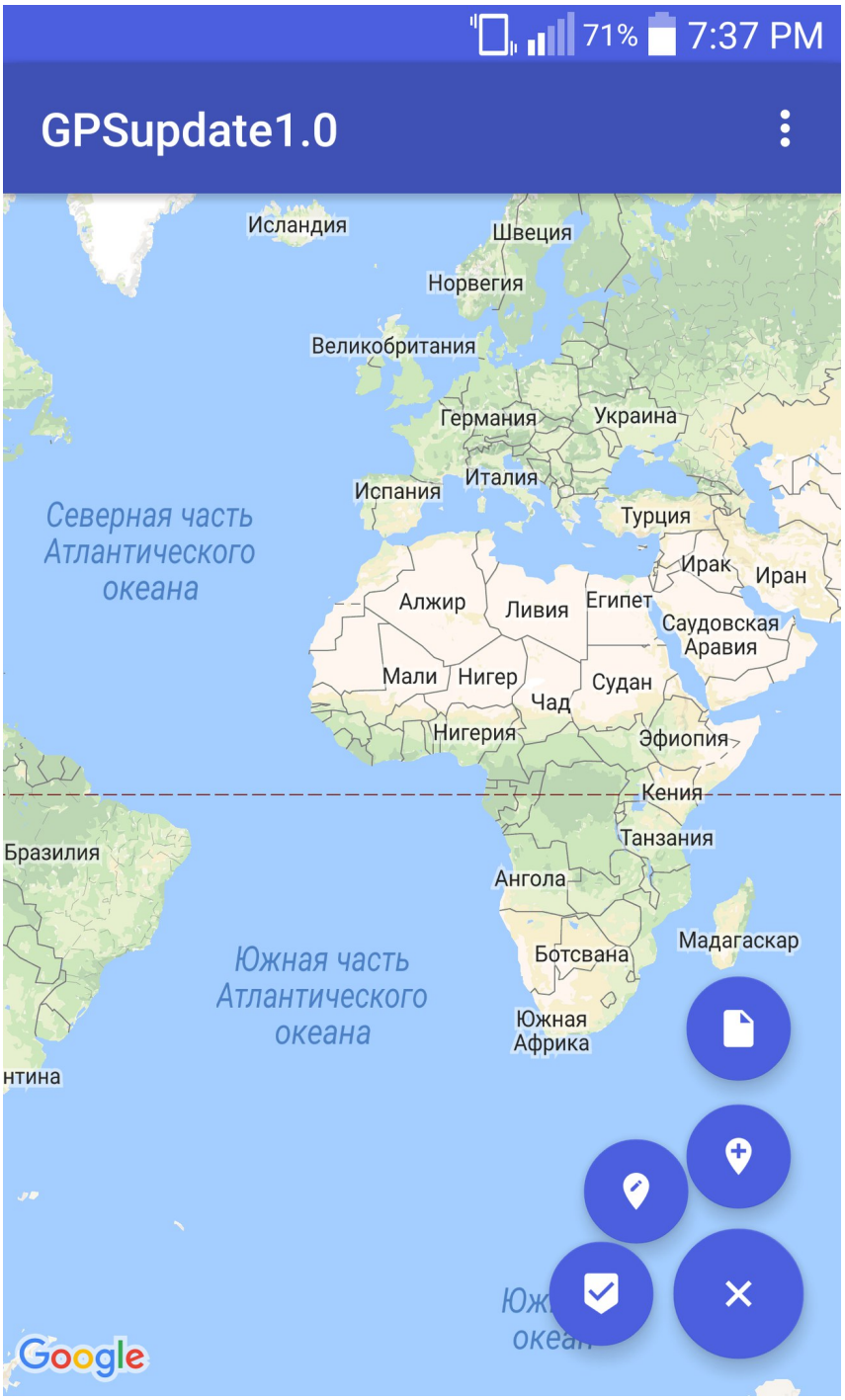


Рис. 1. Главное окно приложения.

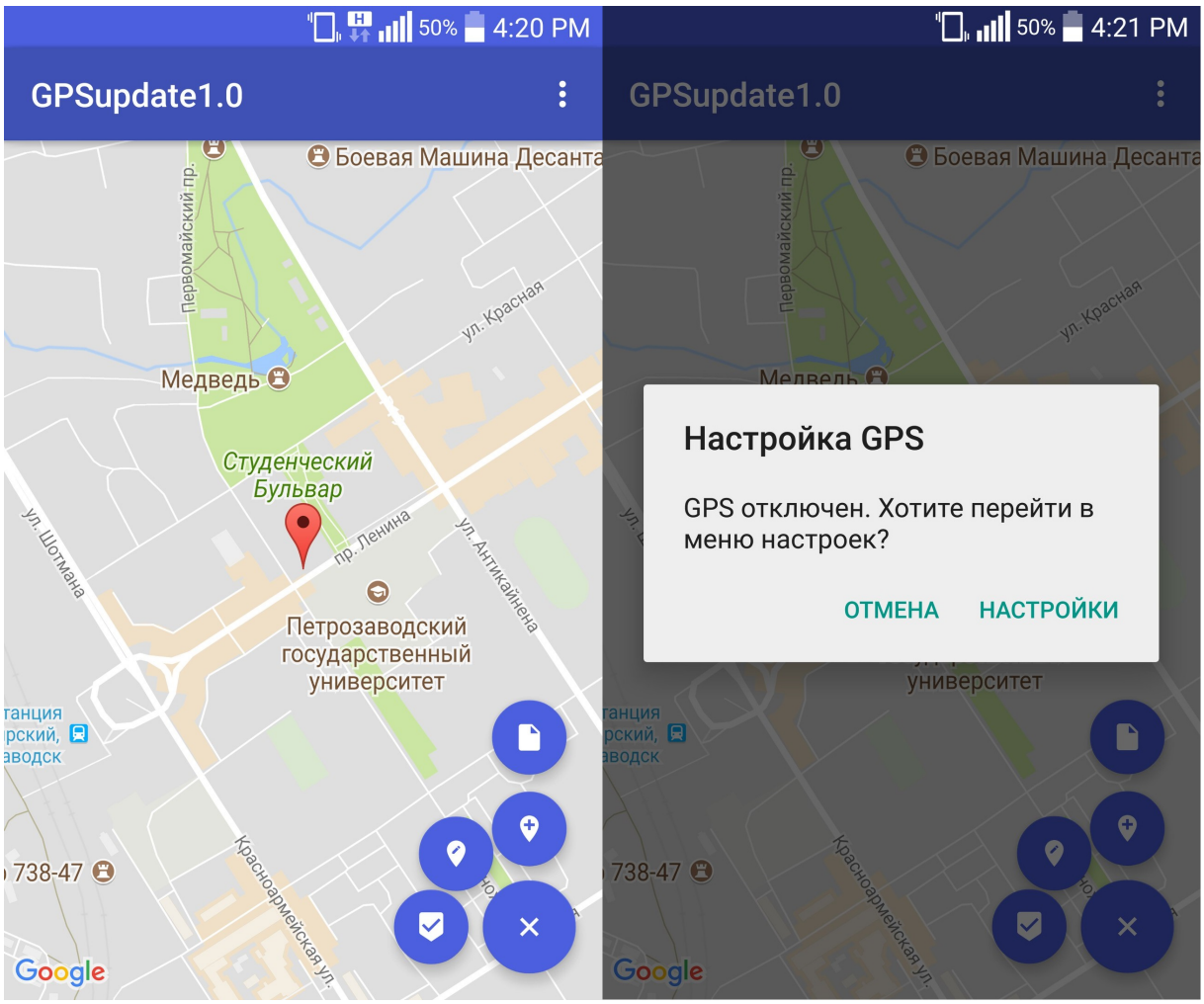


Рис. 2. Нажатие кнопки «Show Location»: отображение дислокации пользователя или вывод диалогового окна для включения функции учета местоположения, если она отключена.

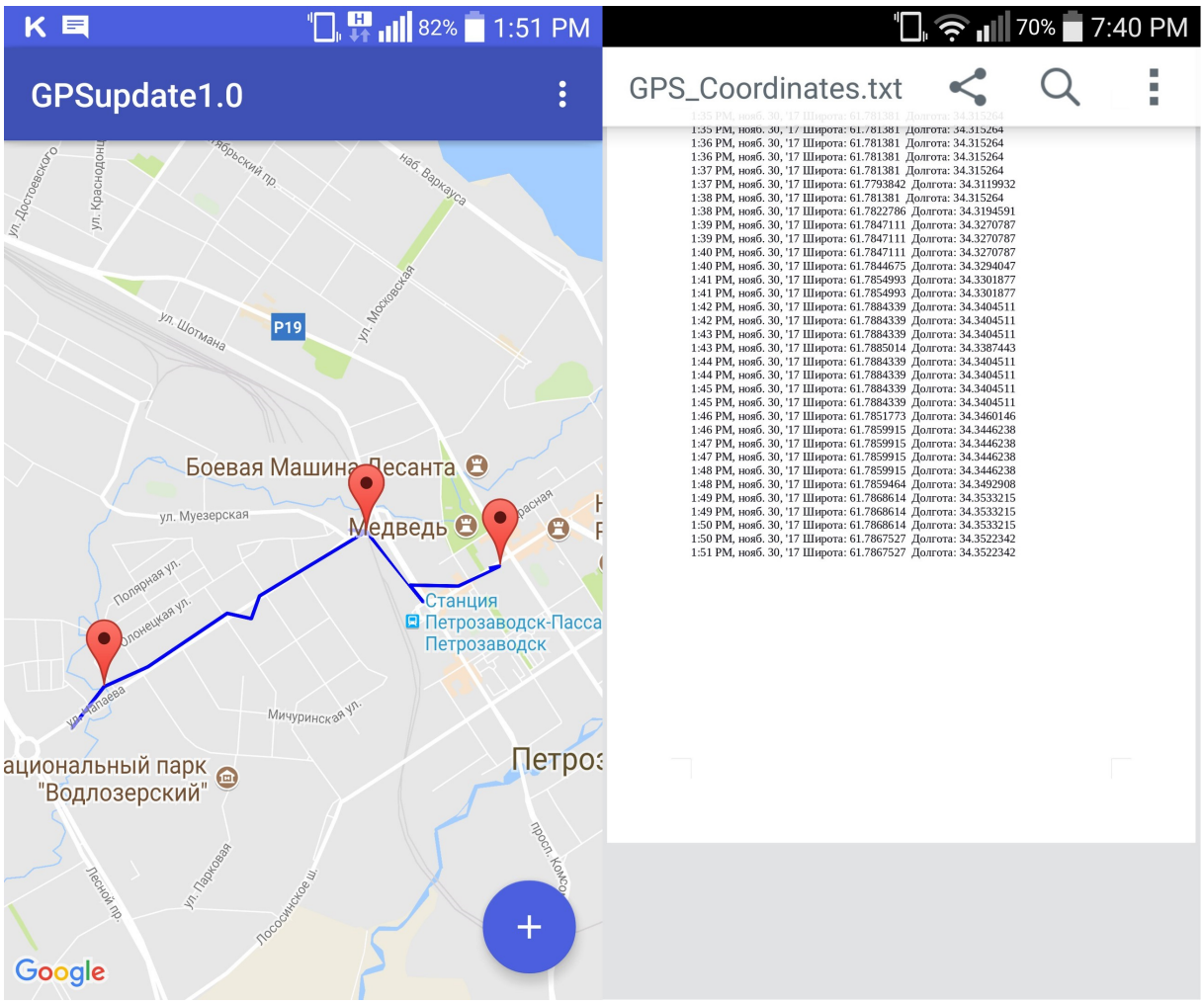


Рис. 3. Кривая, демонстрирующая передвижения пользователя, и файл с аналогичными данными(кнопки «Start Write» и «Read File»).

# 3 Приложение.

Код программы:

MainActivity.java

```
package com.example.ilya.gpsupdate10;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;

import java.io.BufferedWriter;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Locale;
import java.util.Scanner;
import java.util.Timer;
import java.util.TimerTask;

public class MainActivity extends AppCompatActivity
implements OnMapReadyCallback {

    GPSTracker gpsTracker;
    GPSTracker gpsTrackerWrite;

    private GoogleMap map;

    FloatingActionButton fab, fab1, fab2, fab3, fab4;
    Animation fabopen, fabclose, rotforward, rotbackward;
    boolean isOpen = false;

    final String FILENAME_SD = "GPS_Coordinates.txt";
    final String FILE_POINT = "GPS_Points.txt";
    final String DIR_SD = "GPS_Files";
    private BufferedWriter gpswr, pointwr;

    boolean isTimerOn = false;
    boolean isDrawLoc = false;
    private Timer mTimer;
    private MyTimerTask mMyTimerTask;

    private ArrayList<LatLng> latLngArrayList = new ArrayList<>();
```

```
Polyline polyline;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    /*
    Данная часть кода отвечает за обращение к фрагменту карты
    в xml-файле.
     */
    MapFragment fragment = (MapFragment) getFragmentManager()
            .findFragmentById(R.id.map);
    fragment.getMapAsync(this);

    fab = (FloatingActionButton) findViewById(R.id.fab);
    fab1 = (FloatingActionButton) findViewById(R.id.fab1);
    fab2 = (FloatingActionButton) findViewById(R.id.fab2);
    fab3 = (FloatingActionButton) findViewById(R.id.fab3);
    fab4 = (FloatingActionButton) findViewById(R.id.fab4);

    fabopen = AnimationUtils.loadAnimation(this, R.anim.fab_open);
    fabclose = AnimationUtils.loadAnimation(this, R.anim.fab_close);
    rotforward = AnimationUtils.loadAnimation(this, R.anim.rotate_forward);
    rotbackward = AnimationUtils.loadAnimation(this, R.anim.rotate_backward);

    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            AnimFab();
        }
```

```
        });

        /*
        Данная часть кода используется для получения данных о
        местонахождении пользователя и их выводе на экран.
Здесь же указана привязка к кнопке Show Location.
         */
        fab1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                gpsTracker = new GPSTracker(MainActivity.this);
                if(gpsTracker.canGetLocation()) {

                    double latitude = gpsTracker.getLatitude();
                    double longitude = gpsTracker.getLongitude();

                    if(latitude == 0.0 || longitude == 0.0)
                        Toast.makeText(MainActivity.this,
"Местоположение определяется...",
Toast.LENGTH_SHORT).show();
                    else
                        SetMapMark(latitude, longitude);

                } else
                    gpsTracker.showSettingsAlert();
            }
        });

        /*
        Данная часть кода используется для создания взаимосвязи
между кнопкой Read File и соответствующим методом.
         */
        fab2.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View view) {
                readFile();
                Toast.makeText(MainActivity.this,
"Read Button!",
Toast.LENGTH_SHORT).show();
            }
        });

        fab3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(MainActivity.this,
"Simple Button!",
Toast.LENGTH_SHORT).show();
            }
        });

        /*
        Данная часть кода используется для создания взаимосвязи
между кнопкой Start Write и соответствующим методом.
         */
        fab4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                gpsTracker = new GPSTracker(MainActivity.this);
                if(gpsTracker.canGetLocation()) {
                    if (isTimerOn)
                        SetTimer();
                    else
                        PointAlertDialog();
                } else
                    gpsTracker.showSettingsAlert();
            }
        });
```

```
    }

    /*
    Данная часть кода используется для получения пути к каталогу с хранимыми
    на SD-карте данными о передвижениях пользователя и записи в находящийся
    в нем файл строки с данными о местоположении в определенный момент
    времени. Активируется при нажатии на кнопку Start Write.
    */
    class MyTimerTask extends TimerTask {
        @Override
        public void run() {
            if(!Environment.getExternalStorageState().
equals(Environment.MEDIA_MOUNTED)) {
                return;
            }
            File sdPath = Environment.getExternalStorageDirectory();
            sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
            sdPath.mkdirs();
            File sdFile = new File(sdPath, FILENAME_SD);
            File sdPoint = new File(sdPath, FILE_POINT);
            final long date = System.currentTimeMillis();
            final SimpleDateFormat sdf =
    new SimpleDateFormat("K:mm a, MMM d, ''yy");
            try {
                gpswr = new BufferedWriter(new FileWriter(sdFile, true));
                pointwr = new BufferedWriter(new FileWriter(sdPoint, true));
            } catch (IOException e) {
                e.printStackTrace();
            }
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    gpsTrackerWrite = new GPSTracker(MainActivity.this);
```

```
                    String dateString = sdf.format(date);
                    String lat = Double.toString(gpsTrackerWrite.getLatitude());
                    String longe = Double.toString(gpsTrackerWrite.getLongitude())
                    String str = dateString + " Широта: " + lat + " "
    + " Долгота: " + longe + "\n";
                    String strpoint = lat + " " + longe + "\n";
                    try {
                        gpswr.write(str);
                        pointwr.write(strpoint);
                        gpswr.flush();
                        pointwr.flush();
                        gpswr.close();
                        pointwr.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    if(isDrawLoc) {
                        latLngArrayList.add(new LatLng(gpsTrackerWrite.
getLatitude(), gpsTrackerWrite.getLongitude()));
                        if (latLngArrayList.size() == 2) {
                            PolylineOptions polylineOptions =
    new PolylineOptions()
                                    .addAll(latLngArrayList)
                                    .width(10)
                                    .color(Color.BLUE)
                                    .geodesic(true);
                            polyline = map.addPolyline(polylineOptions);
                        } else if(latLngArrayList.size() > 2) {
                            polyline.setPoints(latLngArrayList);
                        }
                    }
                }
            });
        }
```

```
}

private void AnimFab() {
    if(isOpen) {
        fab.startAnimation(rotbackward);
        fab1.startAnimation(fabclose);
        fab2.startAnimation(fabclose);
        fab3.startAnimation(fabclose);
        fab4.startAnimation(fabclose);
        fab1.setClickable(false);
        fab2.setClickable(false);
        fab3.setClickable(false);
        fab4.setClickable(false);
        isOpen = false;
    } else {
        fab.startAnimation(rotforward);
        fab1.startAnimation(fabopen);
        fab2.startAnimation(fabopen);
        fab3.startAnimation(fabopen);
        fab4.startAnimation(fabopen);
        fab1.setClickable(true);
        fab2.setClickable(true);
        fab3.setClickable(true);
        fab4.setClickable(true);
        isOpen = true;
    }
}

/*
Данная часть кода отвечает за установку и корректную работу таймера при
нажатии на кнопку Start Write.
 */
private void SetTimer() {
    if(isTimerOn) {
```

```
            if(mTimer!=null){
                mTimer.cancel();
                mTimer=null;
            }
            Toast.makeText(MainActivity.this,
    "Timer Off!",
    Toast.LENGTH_SHORT).show();
            isTimerOn = false;
        } else {
            if(mTimer!=null)
                mTimer.cancel();

            mTimer = new Timer();
            mMyTimerTask = new MyTimerTask();
            mTimer.schedule(mMyTimerTask, 0, 1000*30);
            Toast.makeText(MainActivity.this,
    "Timer On!",
    Toast.LENGTH_SHORT).show();
            isTimerOn = true;
        }
    }

    /*
    Данная часть кода используется для считывания данных о перемещениях
    пользователя из файла.
     */
    private void SetPointList() {
        try {
            if (!Environment.getExternalStorageState().
equals(Environment.MEDIA_MOUNTED)) {
                return;
            }
            File sdPath = Environment.getExternalStorageDirectory();
            sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
```

```
            File sdPoint = new File(sdPath, FILE_POINT);

            FileInputStream fileInputStream = new FileInputStream(sdPoint);
            Scanner scanner = new Scanner(fileInputStream);
            scanner.useLocale(Locale.ENGLISH);

            while (scanner.hasNextDouble()) {
                latLngArrayList.add(new LatLng(scanner.
nextDouble(),scanner.nextDouble()));
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void PointAlertDialog() {
        AlertDialog.Builder alertDialog =
new AlertDialog.Builder(MainActivity.this);

        alertDialog.setTitle("Предыдущий маршрут");
        alertDialog.setMessage("Хотите увидеть предыдущие перемещения?");
        alertDialog.setPositiveButton("Да",
new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                SetPointList();
                PolylineOptions polylineOptions = new PolylineOptions()
                        .addAll(latLngArrayList)
                        .width(5)
                        .color(Color.BLUE)
                        .geodesic(true);
                polyline = map.addPolyline(polylineOptions);
                isDrawLoc = true;
```

```
                SetTimer();
            }
        });
        alertDialog.setNeutralButton("Запись",
new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                isDrawLoc = false;
                SetTimer();
            }
        });
        alertDialog.setNegativeButton("Нет",
new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                isDrawLoc = true;
                SetTimer();
                dialogInterface.cancel();
            }
        });
        alertDialog.show();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
```

```
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            map.clear();
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    /*
    Данная часть кода используется для проверки готовности карты к работе и
    нанесения маркера на нее с центрированием камеры. Установка маркера
    активируется на кнопку Show Location.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        map = googleMap;
    }

    public void SetMapMark(double latitude, double longitude) {
        LatLng lng = new LatLng(latitude, longitude);
        map.moveCamera(CameraUpdateFactory.newLatLngZoom(lng, 15));
        map.addMarker(new MarkerOptions().position(lng));
    }

    /*
    Данная часть кода используется для получения пути к каталогу и открытия
    файла с данными о передвижениях пользователя, хранимыми на
    SD-карте. Активируется при нажатии на кнопку Read File.
    */
    void readFile() {
```

```
        if(!Environment.getExternalStorageState().
equals(Environment.MEDIA_MOUNTED)) {
            return;
        }
        File sdPath = Environment.getExternalStorageDirectory();
        sdPath = new File(sdPath.getAbsolutePath() + "/" + DIR_SD);
        File sdFile = new File(sdPath, FILENAME_SD);
        Intent i = new Intent();
        i.setAction(Intent.ACTION_VIEW);
        i.setDataAndType(Uri.fromFile(sdFile), "doc/*");
        startActivity(i);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        gpsTracker.stopUsingGPS();
    }
}
```

# Список литературы


1. developer.android.com [Электронный ресурс]: Сайт с информацией для Android-разработчиков. — Электрон. ст. — [USA]. — URL:https://developer.android.com/index.html

2. docs.oracle.com [Электронный ресурс]: Сайт с информацией для Java-разработчиков. — Электрон. ст. — [USA]. — URL:http://docs.oracle.com/javase/7/docs/api/index.html

3. maps.google.ru [Электронный ресурс]: Сервис Google Maps. — Электрон. дан. — [USA]. — URL:https://www.google.ru/maps/