МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА
(повне найменування вищого навчального закладу)

Інститут математики, економіки та механіки
(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем
(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

бакалавра
(освітньо-кваліфікаційний рівень)

на тему Проектування і розробка програмно - апаратного комплексу велосипедиста на базі мікроконтролера

Проектирование и разработка программно - аппаратного комплекса велосипедиста на базе микроконтроллера

Bicyclist software and hardware complex based on microcontroller

Виконав: студент 4 курсу, групи 1
напряму підготовки 6.050102 – «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)

Ємельянов Г.С.
(прізвище та ініціали)

Керівник
к.ф.-м.н., доц. Крапівний Ю.М.
(прізвище та ініціали)

Рецензент
к.т.н., доц. Пенко В.Г.
(прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№_____ від «___»_____р.

Завідувач кафедри

_____________ Є.В. Малахов
(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол №___ від «___» ______р.

Оцінка ________________________
(за 4-х бальною шкалою, за шкалою ECTS, бал.)

Голова ЕК

_____________ О.О. Арсірій
(підпис) (прізвище, ініціали)

Одеса – 2017

АНОТАЦІЯ

При сучасному розвитку технологій все більше з'являється різних проектів націлених на роботу з розумною технікою, а також невеликими апаратно-програмними комплексами які отримали узагальнену назву «Internet of Things». Дана тенденція торкнулася також і ринку вело транспорту.

На сьогоднішній день існують різні варіанти реалізацій, інтеграції та роботи апаратних комплексів на основі мікроконтролерів для досягнення різних цілей в області вело транспорту. Тому в дипломній роботі розробляється програмно-апаратний комплекс велосипедиста на базі мікроконтролера. Основною ж метою даної роботи є створення апаратного комплексу на базі мікроконтролера, вивчення і реалізація предметної області збору статистики з велосипеда, проектування системи з обробки та зберігання отриманих даних з апаратної частини, а також зручний доступ користувача до накопичених даних.

Для досягнення поставленої мети проектується і реалізується апаратний комплекс на основі плати Arduino і набору датчиків, що виконує базову роль у проекті, серверна частина і база даних задовольняє умовам проекту. Також розробляється мобільний клієнт, що дозволяє отримати доступ до накопичених даних, який також служить проміжною ланкою між апаратною і серверною частиною.

Завдяки реалізованій багаторівневій системі, даний комплекс допомагає користувачеві збирати і отримувати певний спектр точних даних.

Даний проект має перспективи для подальшого доопрацювання та розширення функціоналу, так як вже реалізовані основні соціальні аспекти. Це допоможе в майбутньому більш активно розвиватися проекту і об'єднати навколо себе більше користувачів, що сприятиме збільшенню популярності вело транспорту.

АННОТАЦИЯ

При современном развитии технологий все больше появляется различных проектов нацеленных на работу с умной техникой, а также небольшими аппаратно-программными комплексами, которые получили обобщенное название «Internet of Things». Данная тенденция коснулась также и рынка вело транспорта.

На сегодняшний день существуют различные варианты реализаций, интеграции и работы аппаратных комплексов на основе микроконтроллеров для достижения различных целей в области вело транспорта. Поэтому в дипломной работе разрабатывается программно-аппаратный комплекс велосипедиста на базе микроконтроллера. Основной же целью данной работы является создание аппаратного комплекса на базе микроконтроллера, изучение и реализация предметной области сбора статистики с велосипеда, проектирование системы по обработке и хранению полученных данных с аппаратной части, а также удобный доступ пользователя к накопленных данных.

Для достижения поставленной цели проектируется и реализуется аппаратный комплекс на основе платы Arduino и набора датчиков, выполняющие основополагающую роль в проекте, серверная часть и база данных удовлетворяет условиям проекта. Также разрабатывается мобильный клиент, позволяющий получить доступ к накопленных данных, который также служит промежуточным звеном между аппаратной и серверной частью.

Благодаря реализованной многоуровневой системе, данный комплекс помогает пользователю собирать и получать определенный спектр точных данных. Данный проект имеет перспективы для дальнейшей доработки и расширения функционала, так как уже реализованы основные социальные аспекты. Это поможет в будущем более активно развиваться проекта и объединить вокруг себя больше пользователей, что будет способствовать увеличению популярности вело транспорта.

ABSTRACT

With the modern development of technologies, more and more projects are emerging aimed at working with smart technology, as well as with small hardware and software complexes, which received a generalized name "Internet of Things". This trend has also affected the bicycle market.

To date, there are various options for the implementation, integration and operation of hardware systems based on microcontrollers to achieve various goals in the field of bicycle transport. Therefore, in the thesis, a software and hardware biker complex is developed based on a microcontroller. The main goal of this work is to create a hardware complex based on the microcontroller, to study and implement the subject area of statistics collection from a bicycle, to design a system for processing and storing received data from the hardware, and convenient user access to the accumulated data.

To achieve this goal, a hardware complex is being designed and implemented based on the Arduino board and a set of sensors that fulfill the fundamental role in the project, the server part and the database satisfy the project conditions. A mobile client is also being developed that allows access to accumulated data, which also serves as an intermediate link between the hardware and server parts.

Thanks to the implemented multi-level system, this complex helps the user to collect and obtain a certain range of accurate data.

This project has prospects for further refinement and expansion of the functional, since the basic social aspects have already been implemented. This will help in the future more actively develop the project and unite around itself more users, which will increase the popularity of bicycles.

СОДЕРЖАНИЕ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Скорочення:

API – Application Programming Interface;

ASP – Active Server Pages;

DTO – Data Transfer Object;

MVC – Model View Controller;

MVP – Model View Presenter;

MVVM – Model View ViewModel;

POJO – Plain Old Java Object;

REST – Representational State Transfer;

RIP – Passive Infrared Sensor;

SDK – Software Development Kit.

ВСТУП

В даний час все частіше з'являються різні проекти, націлені на роботу з розумною технікою, а також невеликі апаратно-програмні комплекси, які отримали узагальнену назву «Internet of Things».

Найчастіше подібні проекти спрямовані на спрощення взаємодії користувача з великими масивами даних або ж швидке управління повсякденними процесами. Так як подібних процесів в повсякденному житті достатньо багато, не дивно що і число різних «стартапів», спрямованих на оптимізацію подібних процесів, з кожним роком зростає.

Люди все частіше приділяють увагу такій проблемі як високий поріг смертності серед молодого і середнього покоління. Проблема полягає в малоактивному способі життя, а також відмові від будь-якої спортивної діяльності. Для популяризації здорового способу життя було створено велику кількість різних проектів, спрямованих на популяризацію даного підходу. Це дало необхідний результат. У свою чергу швидко почала зростати кількість проектів, націлених на відстеження стану здоров'я і досягнутого результату, оскільки не завжди він може бути візуально оцінений. Така тенденція не могла залишити поза увагою вело ринок. А так як користь від їзди на цьому виді транспорту як для здоров'я, так і для навколишнього середовища була неодноразово доведена вченими, число проектів на базі «Internet of Things», для цього виду транспорту різко зросла.

Метою даного дипломного проекту є проектування і реалізація апаратно-програмного комплексу велосипедиста на базі мікроконтролера, для відстеження певного масиву даних, а також реалізація соціальної комунікації для популяризації цього виду транспорту.

Для досягнення даної мети необхідно вирішити такі завдання:

1) виконати аналіз предметної області «апаратний комплекс на базі мікроконтролера»;

2) розробити модель даних для даної предметної області;
3) створити апаратний комплекс націлений на обробку та передачу даних;
4) створити програмну частину для коректної роботи апаратного комплексу;
5) розробити серверну частину проекту і базу даних яка задовольняє умовам проекту;
6) розробити мобільний додаток, яке наддасть доступ до перегляду статистики, а також передачі даних на серверну частину [1].

ВИСНОВОК

У даній дипломній роботі був спроектований і реалізований програмно-апаратний комплекс велосипедиста на базі мікроконтролера, а також система по збору статистики, аналізу її даних і видачу їх в зручному інтерфейсі користувача, з можливістю інтеграції з «Internet of Things» для збору більш точних і обширних даних.

При виконанні даної роботи був проведений аналіз предметної області даної теми. Було виділено такі основні групи робіт для реалізації даного проекту:

- інтеграція роботи з системою «Internet of Things»;
- реалізація архітектури сервера і реалізації ресурсу бази даних;
- створення мобільного застосування для зручної роботи з зібраними даними.

В рамках розробки системи був спроектований і реалізований весь набір підсистем.

Перша підсистема включає в себе роботу з групою датчиків, націлену на для більш точний збір даних. А також робота з платою Arduino, завдання якої полягає в зборі та аналізі даних з вищезазначених датчиків. Всі ці дані передаються на мобільний додаток за допомогою каналу зв'язку Bluetooth.

Друга підсистема включає в себе роботу з отриманими даними з мобільного додатку. Отримання даних відбувається за допомогою системи Web API, реалізованої на серверній частині проекту. Наявність облікових записів персоналізує роботу користувачів, а реалізації таких соціальних областей як поняття «Друзі» і комунікація за допомогою повідомлень, яка в майбутньому буде інтегрована в мобільний додаток. Це допоможе користувачам системи краще комунікувати один з одним, збільшуючи співтовариство учасників даного проекту, що дозволить розробнику більш активно його розвивати, спираючись на велику базу користувачів.

Також на серверній частині за допомогою різних технологій була зроблена можливість аналізу і зберігання даних в спеціально спроектованій під даний проект базі даних, що значно підвищує зручність використання програми.

Третя підсистема спроектована за допомогою мобільного додатку, який було реалізовано на базі операційної системи Android. Зручний і інтуїтивно зрозумілі інтерфейс допоможе користувачеві негайно включитися в роботу. Користувачеві доступні екрани по статистиці, за якими він зможе відстежити різні стадії свого прогресу, за певний період. Також користувачеві доступна можливість знайти і підключити різні пристрої. Система побудована так, що дані будуть оновлюватися в режимі «реального часу», що дозволить користувачеві зручніше аналізувати отриману інформацію. Наявність в додатку внутрішнього кешу дасть можливість користувачеві переглядати основні дані при наявності проблем з підключенням, які можуть виникнути під час тренувань.

Даний проект має перспективи для подальшого доопрацювання та розширення функціоналу, так як вже реалізовані основні соцальні аспекти, які необхідно впровадити в мобільний додаток для більш комфортної роботи не одного користувача, а цілих груп користувачів.

СПИСОК ЛІТЕРАТУРИ

1. Емельянов Г.С., Числовое программное управление умным домом / Г.С. Емельянов, Ю.Н. Крапивный // Тези докладів чотирнадцятої всеукраїнської конференції студентів и молодих науковців «Інформатика, інформаційні системи і технології». Одеса, 14 квітня 2017 р. – Одеса, 2017.– С.152-154.

2. Мобільний додаток «Runtastic Road Bike» [Електронний ресурс] – Режим доступу: http://forwardvelo.ru/my/apps/runtastic-road-bike – 03.06.17

3. Мобільний додаток «Strava» [Електронний ресурс] – Режим доступу: http://forwardvelo.ru/my/apps/strava – 03.06.17

4. Плата Arduino [Електронний ресурс] – Режим доступу: https://www.arduino.cc/en/Main/ArduinoBoardUno – 05.06.17

5. RIP сенсор [Електронний ресурс] – Режим доступу: http://arduino-diy.com/arduino-infrakrasnyy-PIR-datchik – 05.06.17

6. MC-08 [Електронний ресурс] – Режим доступу: https://www.asutpp.ru/datchiki/gerkon.html – 04.06.17

7. HC-06 [Електронний ресурс] – Режим доступу: https://arduino-ua.com/prod241-Bluetooth_modyl – 04.06.17

8. Клієнт-серверна архітектура [Електронний ресурс] – Режим доступу: http://www.4stud.info/networking/lecture5.html. – 05.06.17

9. MVVM [Електронний ресурс] – Режим доступу: https://msdn.microsoft.com/en-us/library/hh848246.aspx – 02.06.17

10. Material Design [Електронний ресурс] – Режим доступу: https://material.io/ – 01.06.17

11. MVC [Електронний ресурс] – Режим доступу: https://www.asp.net/mvc – 02.06.17

12. Entity Framework [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/entityframework/1.1.php. - 05.06.17

13. ASP.NET WebApi [Електронний ресурс] – Режим доступу: https://smarly.net/asp-net-mvc-4-in-action/mastering-asp-net-mvc/asp-net-web-api/what-is-web-api. – 03.06.17

14. PostgreSQL [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/ – 04.06.17

15. Java [Електронний ресурс] – Режим доступу: https://java.com/ru/about/ – 02.06.17

16. XML [Електронний ресурс] – Режим доступу: http://www.codenet.ru/webmast/xml/ – 01.06.17

17. Реактивне програмування [Електронний ресурс] – Режим доступу: https://habrahabr.ru/post/140719/ – 03.06.17

18. Retrofit [Електронний ресурс] – Режим доступу: http://developer.alexanderklimov.ru/android/library/retrofit.php – 05.06.17

19. Realm [Електронний ресурс] – Режим доступу: https://realm.io/products/realm-mobile-database/ – 04.06.17

ДОДАТОК А
SQL-ЗАПИТИ НА СТВОРЕННЯ ЕЛЕМЕНТІВ БАЗИ ДАНИХ

```
//SQL-запит на створення таблиці SignUp
CREATE TABLE sign_up
(
    base_code text NOT NULL,
    user_id integer NOT NULL,
    password_encode text NOT NULL,
    login_encode text NOT NULL,
    CONSTRAINT sing_up_pkey PRIMARY KEY (base_code),
    CONSTRAINT ok_sing_up_user_id FOREIGN KEY (user_id)
    REFERENCES user_data (user_id) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
);
//SQL- запит на створення таблиці User
CREATE TABLE user_data
(
    user_id serial NOT NULL,
    profile_id integer NOT NULL,
    firebase_token text,
    friend_id_array text,
    communication_id_array text,
    group_id_array text,
    arduino_id_array text,
    friend_possible_id_array text,
    communication_pin_id_array text,
    CONSTRAINT user_pkey PRIMARY KEY (user_id),
    CONSTRAINT ok_user_profile_id FOREIGN KEY (profile_id)
    REFERENCES profile (profile_id) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

```
//SQL- запит на створення таблиці Profile
CREATE TABLE profile
(
     profile_id serial NOT NULL,
     name text NOT NULL,
     last_name text NOT NULL,
     city text NOT NULL,
     photo_url text NOT NULL,
     phone text NOT NULL,
     email text NOT NULL,
     time_last_active timestamp without time zone NOT NULL,
     profile_statistics_id integer NOT NULL,
     CONSTRAINT profile_pkey PRIMARY KEY (profile_id),
     CONSTRAINT ok_profile_statics_id FOREIGN KEY
     (profile_statistics_id)
     REFERENCES profile_statistics (profile_statistics_id) MATCH
SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
);
//SQL- запит на створення таблиці ProfileStatistics
CREATE TABLE profile_statistics
(
     profile_statistics_id serial NOT NULL,
     count_distance_total real NOT NULL,
     middle_speed_total real NOT NULL,
     time_in_trip_total interval NOT NULL,
     calories_total integer NOT NULL,
     count_dangerous_situation integer NOT NULL,
     count_atteempted_theft integer NOT NULL,
     every_day_profile_statistics_id_array text,
     CONSTRAINT profile_statistics_pkey PRIMARY KEY
(profile_statistics_id)
);
```

```
//SQL- запит на створення таблиці EveryDayProfileStatistics
CREATE TABLE every_day_profile_statistics
(
     every_day_profile_statistics_id integer NOT NULL
     count_distance real NOT NULL,
     middle_speed real NOT NULL,
     time_in_trip interval NOT NULL,
     calories integer NOT NULL,
     time_create timestamp without time zone,
     CONSTRAINT every_day_profile_statistics_pkey PRIMARY KEY
(every_day_profile_statistics_id)
);
//SQL- запит на створення таблиці Communication
CREATE TABLE communication
(
     communication_id serial NOT NULL,
     key_dialog text NOT NULL,
     name text NOT NULL,
     photo_url text NOT NULL,
     participant_profile_id_array text NOT NULL,
     message_id_array text NOT NULL,
     creater_profile_id integer NOT NULL,
     add_profile_timestamp_array text,
     is_group boolean,
     CONSTRAINT communication_pkey PRIMARY KEY (communication_id)
);
//SQL- запит на створення таблиці Message
CREATE TABLE message
(
     message_id serial NOT NULL,
     data_message text NOT NULL,
     time_written timestamp without time zone NOT NULL,
     is_read text NOT NULL,
```

```
    profile_id integer NOT NULL,
    type_message integer NOT NULL,
    CONSTRAINT message_pkey PRIMARY KEY (message_id)
);
//SQL- запит на створення таблиці Arduino
CREATE TABLE arduino
(
    arduino_id serial NOT NULL,
    name text NOT NULL,
    mac text NOT NULL,
    CONSTRAINT arduino_id_pkey PRIMARY KEY (arduino_id)
);
```

ДОДАТОК Б
ОСНОВНИЙ КОД ПРОГРАМИ

```
public class MainActivityLogic implements SignOutCallback {

  private Context mContext;
  private FileUtils mFileUtils;
  private DataBaseUtils mDataBaseUtils;
  private ResourceUtils mResourceUtils;
  private CompositeSubscription mSubscription;
  private PreviewProfileDTO mPreviewProfileDTO;
  private UpdateHeaderCallback mUpdateHeaderCallback;

  private boolean checkPhoto;
  public MainActivityLogic(Context context) {
    mContext = context;
    mFileUtils = new FileUtils(mContext);
    mDataBaseUtils = new DataBaseUtils(mContext);
    mResourceUtils = ResourceUtils.with(mContext);
    mSubscription = new CompositeSubscription();
  }
  public void registerCallback(UpdateHeaderCallback callback) {
    mUpdateHeaderCallback = callback;
  }
  public void onResume() {
    if
(mDataBaseUtils.getStateApplication().equals(StateApplication.EN
TER)) {
      actionByStateEnter();
    }
  }
  public void checkStateApplication() {

if(mDataBaseUtils.getStateApplication().equals(StateApplication.
LOGIN)) {
      logOut();
 } else {

chooseFragmentAndTitleById(R.id.navigation_item_statistics);
    }
  }
private void actionByStateEnter() {
    Observable<PreviewProfileDTO> photoUrl =
        mPreviewProfileDTO != null ? inputData() :
getDataProfile();
    Subscription mSubscriptionProfile = photoUrl
        .map(PreviewProfileDTO::getPhotoUrl)
        .observeOn(Schedulers.io())
        .map(this::savePhoto)
        .observeOn(AndroidSchedulers.mainThread())
```

```
        .doOnNext(this::savePhotoPathToDataBase)
        .doOnCompleted(this::uploadFromDataBase)
        .subscribe(new Subscriber<String>() {
          @Override
          public void onCompleted() {
          }
          @Override
          public void onError(Throwable e) {
          }
          @Override
          public void onNext(String photoUrl) {
          }
        });
    mSubscription.add(mSubscriptionProfile);
  }
  private Observable<PreviewProfileDTO> inputData() {
    return Observable.just(mPreviewProfileDTO);
  }
  private Observable<PreviewProfileDTO> getDataProfile() {
    ProfileService service =
RestClient.createClientService(ProfileService.class);
    return service
        .profileData(mDataBaseUtils.getBasic())
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .doOnError(error -> uploadFromDataBase())
        .doOnNext(this::checkSameDataAndWriteToDataBase);
  }

  private void checkSameDataAndWriteToDataBase(PreviewProfileDTO
previewProfile) {
    UserDBModel userDBModel = mDataBaseUtils.getUser();
    String fullName = previewProfile.getFullName();
    boolean isSame =
previewProfile.getPhotoUrl().equals(userDBModel.getPhotoUrl());
    checkPhoto = isSame ? new
File(userDBModel.getPhotoPath()).exists() && isSame : isSame;
    if (!fullName.equals(userDBModel.getFullName())) {
      mDataBaseUtils.transaction(() ->
userDBModel.setFullName(fullName));
    }
    if (!isSame) {
      mDataBaseUtils.transaction(() ->
userDBModel.setPhotoUrl(previewProfile.getPhotoUrl()));
    }
  }
  private String savePhoto(String photoUrl) {
    String pathPath =
mResourceUtils.string(R.string.empty_string);
    if (!checkPhoto) {
      pathPath =
mResourceUtils.string(R.string.file_utils_no_photo_file);
```

```
      try {
        Bitmap bitmapPhotoProfile =
Picasso.with(mContext).load(photoUrl).get();
        pathPath = mFileUtils.savePhotoFile(bitmapPhotoProfile);
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
    return pathPath;
  }
private void savePhotoPathToDataBase(String photoPath) {
    if (!photoPath.isEmpty()) {
      mDataBaseUtils.transaction(() ->
mDataBaseUtils.getUser().setPhotoPath(photoPath));
    }
  }
  private void uploadFromDataBase() {

mUpdateHeaderCallback.updateHeader(convertToProfileModel(mDataBa
seUtils.getUser()));
  }
  private ProfileModel convertToProfileModel(UserDBModel
dbModel) {
    ProfileModel model = new ProfileModel();
    model.setFullName(dbModel.getFullName());
    model.setPhotoUrl(dbModel.getPhotoPath());
    return model;
  }
  public void chooseFragmentAndTitleById(int menuId) {
    switch (menuId) {
      case R.id.navigation_item_statistics:

chooseFragmentAndTitle(StatisticsFragment.newInstance(mPreviewPr
ofileDTO),
            R.string.navigation_item_statistics);
        break;
      case R.id.navigation_item_arduino:
        chooseFragmentAndTitle(ArduinoFragment.newInstance(),
R.string.navigation_item_arduino);
        break;
    }
  }
  private void chooseFragmentAndTitle(Fragment fragment, int
resource) {
    FragmentManager manager = ((MainActivity)
mContext).getSupportFragmentManager();
    manager
        .beginTransaction()
        .replace(R.id.frameLayoutMain, fragment)
        .commit();
    ((MainActivity) mContext).getSupportActionBar()
        .setTitle(mResourceUtils.string(resource));
```

```
  }
  private void logOut() {
    mDataBaseUtils.deleteUser();
    mFileUtils.deleteCacheFile();
    Intent moveToLogin = new Intent(mContext,
LogInActivity.class);
    mContext.startActivity(moveToLogin);
    ((MainActivity) mContext).finish();
  }
  public void setPreviewProfileDTO(PreviewProfileDTO
previewProfileDTO) {
    mPreviewProfileDTO = previewProfileDTO;
  }
  public void onDestroy() {
    if (mSubscription != null) {
      mSubscription.unsubscribe();
    }
    if (mDataBaseUtils != null) {
      mDataBaseUtils.close();
    }
  }

  @Override
  public void signOut() {
    logOut();
  }
}
```

ДОДАТОК В
ОСНОВНИЙ КОД ПРОГРАМИ ARDUINO

```
class StatisticsData {
 float distance;
 float speedBike;
 float timeInTrip;
 int clories;
 int countDangerousSituation;
 public:
    void setDistance (float distanceCurrent){
      distance = distanceCurrent;
    }
    float getDistance() {
      return distance;
    }
    void setSpeedBike (float speedB){
      speedBike = speedB;
    }
    float getSpeedBike() {
      return speedBike;
    }
    void setTimeTrip (float timeTrip){
      timeInTrip = timeTrip;
    }
    float getTimeTrip() {
      return timeInTrip;
    }
    void setCalories (int calor){
      clories = calor;
    }
    int getCalories() {
      return clories;
    }
    void setDangerousSituation (int dangerous){
      countDangerousSituation = dangerous;
    }
    int getDangerousSituation() {
      return countDangerousSituation;
    }
};

#include <SoftwareSerial.h>
int touchSensorPin = 6;
int moveDetectorPin = 7;
int ledPinTouch = 8;
int ledPinMove = 9;

int val = 0;
int countTouch = 1;
```

```
float countTime = 0;
int countDangerousSituation = 1;
float meedleLenth = 3.89;
float meedleWeight = 70;
int countDelay = 2;
int countMetrInKm = 1000;
int countMillisInSec = 1000;
int oneDelay = 1000;

int minSecsBetweenSend = 60;
long lastSend = -minSecsBetweenSend * 1000;
bool checked;

long oldTimestamp = millis();

StatisticsData data;

#define rxPin 2
#define txPin 3

SoftwareSerial blueTooth(rxPin, txPin);

void setup(){
  pinMode(moveDetectorPin, INPUT);
  pinMode(ledPinMove, OUTPUT);
  pinMode(touchSensorPin, INPUT);
  pinMode(ledPinTouch, OUTPUT);
  Serial.begin(9600);
  blueTooth.begin(9600);
}
void loop(){
  workMovementSensor();
  workTouchSensor();
  workBluetooth();
}
void workMovementSensor(){
  long now = millis();
  bool height = digitalRead(moveDetectorPin) == HIGH;
  if (height && !checked) {
    bool timeCheck = now > lastSend + minSecsBetweenSend *
countMillisInSec;
    if (timeCheck)
         digitalWrite(ledPinMove, HIGH);

         data.setDangerousSituation(countDangerousSituation++);
         outputConsole("Movement: ",
data.getDangerousSituation());
         lastSend = now;

         delay(oneDelay);
         checked = true;
   }
```

```
   else if(!height && checked) {
      digitalWrite(ledPinMove, LOW);
      Serial.println("Too soon");
      delay(oneDelay);
      checked = false;
  }
}
void workTouchSensor(){
  val = digitalRead(touchSensorPin);
  if(val == HIGH) {
      digitalWrite(ledPinTouch, HIGH);

      long current = millis();
      long differ = current - oldTimestamp;
      countTime = differ + countTime;

      data.setTimeTrip(countTime / countMetrInKm);

data.setSpeedBike(meedleLenth/(abs((differ/countMillisInSec) -
countDelay)));
      data.setDistance((meedleLenth * (countTouch++)) /
countMetrInKm);
      data.setCalories(meedleWeight * data.getDistance());

      outputConsole("Time seconds: ", data.getTimeTrip());
      outputConsole("Speed m/s ", data.getSpeedBike());
      outputConsole("Distance km: ", data.getDistance());
      outputConsole("Calories count: ", data.getCalories());

      delay(oneDelay);
      oldTimestamp = current;
  }
  digitalWrite(ledPinTouch, LOW);
  delay(oneDelay);
}
      void workBluetooth(){
        sendDataOut(blueTooth, data);
      }
      void outputConsole(String title, float value){
           Serial.print(title);
           Serial.println(value);
      }
      void sendDataOut(SoftwareSerial blueTooth, StatisticsData
data){
        if (Serial.available())
           {
             blueTooth.write(data.getTimeTrip());
             blueTooth.write(data.getSpeedBike());
             blueTooth.write(data.getDistance());
             blueTooth.write(data.getCalories());
           }
}
```