

**ТЕХНОЛОГИЧЕСКО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

ДИПЛОМНА РАБОТА

Тема: Система за сигурност на мотоциклети

Дипломант:

Николай Романов

Научен ръководител:

Александър Осенов

СОФИЯ

2018



**ТЕХНОЛОГИЧНО УЧИЛИЩЕ
ЕЛЕКТРОННИ СИСТЕМИ
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

Дата на заданието: 06.11.2017 г.

Утвърждавам:.....

Дата на предаване: 06.02.2018 г.

/проф. д-р инж. Т. Василева/

ЗАДАНИЕ

за дипломна работа

на ученика Николай Андреев Романов от 12 Б клас

1.Тема: Система за сигурност на мотоциклети

2.Изисквания:

2.1 Да се създаде механизъм, който има възможност да локализира превозно средство и да следи координатите му в реално време.

2.2 Да се създаде уеб услуга, която да предоставя възможност за синхронизация на данните.

2.3 Да се създаде Android приложение за предоставяне на данните на потребителя.

3.Съдържание

3.1 Обзор

3.2 Същинска част

3.3 Приложения :

Дипломант :.....

/ Николай Андреев Романов /

Ръководител:.....

/ Александър Осенов /

Директор:.....

/ доц. д-р инж. Ст. Стефанова /

Увод

Система против кражба представлява устройство или метод, който предотвратява неразрешено присвояване на елементи, считани за ценни. Кражбите на моторни превозни средства са едно от най-често срещаните престъпления в модерния свят. От изобретяването на първата система за заключване с ключ до представянето на RFID (Radio-frequency identification) и биометричната идентификация, системите против кражба са се развили, за да съответстват на новите изобретения за обществото и за предотвратяване кражба от страна на престъпниците.

Алармата е електрическо устройство, което се инсталира на моторното превозно средство в опит да се предотврати кражбата на самото превозно средство, неговото съдържание или и двете. Алармите за МПС работят като произвеждат силен звук (обикновено сирена, клаксон, клаксона на превозното средство, записан звук за предупреждение или комбинация от всички). Когато предпоставките да се включи алармата са налице, се включват и светлините на превозното средство и (по желание) уведомяване на собственика чрез система за пейджинг и прекъсване на единични или множество електрически вериги, необходими за стартиране на автомобила.

Ранна версия на алармата за коли и използвана като средство против кражба е измислена от непознат затворник от Денвър през 1913 година. Прототипа бил ръчно поставен и задействан, когато някой се опита да завърти двигателя. По-късно била измислена аларма, вдъхновена от стартер с дистанционно, излязла през 1916 година. Тази версия представлявала приемник, който започва да вибрира ако двигателя на колата бъде запален.

Алармите са различно устройство от Immobilizer, въпреки, че целта на двете устройства е една и съща, те работят по различен начин. Обикновено имобилайзерът няма да предупреди звуково или визуално възпрепятстване на кражбата, нито да изисква повече вход от водача, отколкото от водача на автомобил, който няма имобилайзер.

Алармите се разделят на два подтипа: OEM (вградени от за производителя на превозното средство) и Aftermarket (Закупени и инсталирани след като превозното средство е излязло от завода). Immobiliser е електронно устройство, което предотвратява моторът на моторно превозно средство да бъде запален докато не бъде поставен точния ключ. Това предотвратява колата или мотора да бъде „hot wired” (запалена без ключ), след като крадецът е успял да влезе в купето. Изследванията показват, че поставянето на имобилайзери намалява процента за успех при кражба с 40%.

Електрическият имобилайзер е изобретен от St. George Evans и Edward Birkenbuel и е патентиран през 1919 година. Те разработили 3x3 решетка от двуконтактни превключватели на панела, монтиран вътре в автомобила, така че когато ключът за запалването е бил завъртян, токът от батерията отива до запалителните свещи, позволявайки двигателят да стартира или да обезвреди превозното средство и да задейства неговия звуков сигнал.

Много уязвимости били открити в имобилайзерите, предназначени да презпазват модерните автомобили от кражба. Много имобилайзери използват чип Megamos, за който е доказано, че може да бъде разбит. Megamos транспондерът е един от многото различни транспондери, които могат да бъдат открити в днешните имобилайзер системи и също така идва в много различни итерации. Хакването на имобилайзер в реалния свят може да се случи върху превозното средство, а не върху ключа му. Много по-лесно може да бъде програмиран нов ключ, от колкото да се прави опит да се клонира предишен ключ, специално за модерните превозни средства.

Първа глава

1.1 Преглед на устройства за проследяване и защита на автомобили

1.1.1 Oxford Tracker Spy



Oxford Tracker Spy е просто и ефективно устройство, което лесно може да бъде поставено в мотоциклет (или всяко друго превозно средство, или предмет, които искате да следите). Устройството се прикрива на подходящо място в мотоциклета и се активира с мобилен телефон, таблет или дори компютър! Ако мотоциклетът бъде откраднат, отваряте вашето Oxford Tracker Spy приложение за да намерите къде в този момент, навсякъде по света, където има мобилна мрежа, се намира вашето превозно средство. Уредът е компактен и предпазва всеки автомобил или друг движещ се предмет, за който е прикрепен. Oxford Tracker Spy може да бъде монтиран изключително лесно и бързо на всяко превозно средство, както и преместен на друго. Oxford Tracker Spy не изисква никакви допълнителни инструменти или познания в друга област за да бъде инсталиран, необходимо е само да бъде прикрито устройството на избраното подходящо място! Устройството осигурява бърз достъп до неговите основни функции и настройки, използвайки мобилен телефон, таблет или компютър, в зависимост от наличните възможности. Oxford Tracker Spy предоставя покритие в световен мащаб, използвайки сателити за глобално позициониране (GPS) и вграден SIM чип. GPS позицията на

устройството може да бъде видяна през Google maps от компютър или смартфон.

Oxford Tracker Spy съдържа 4 различни режима на ефективно проследяване от мотоциклета или автомобила.

Sleep mode или заспал режим – Това е режимът, зададен по подразбиране, той осигурява местоположението на устройството на всеки 5 минути.

Geofence mode представлява географския район около устройството. Ако автомобилът излезе от обсега на този район, потребителя моментално получава известие за това

движение. Новата локация може да бъде получена всеки 5 минути. Устройството може да изобразява линията на движение на вашето устройство, показвайки неговата траектория на движение до новото местоположение.

Alarm mode или режим на аларма. В случай, че превозното средство бъде откраднато може да активирате Alarm mode, като въведете паролата си. По този начин обявявате мотоциклета или автомобила ви за откраднат.

Функциите, които Oxford Tracker Spy предоставя:

Защитен и сигурен - Компактно проследяващо устройство, което предпазва вашия автомобил или всеки друг движещ се предмет, за което е прикрепено.

GPS сателит – Покритие в целия свят, на всяка точна за земята, използвайки интегриран SIM чип.

Google Maps – Позицията на устройството е изобразена на гугъл карти и може да бъде видяна през вашия компютър или всяко смарт устройство.

Неограничен брой тракъри – Имате способността да проследявате повече от 1 устройство на приложението, проследете вашата кола, мотоциклет, лодка, АТВ или велосипед.

Sleep режим – Това е режимът, зададен по подразбиране, който осигурява местоположението на устройството всеки 5 минути. Нова позиция може да бъде предадена само ако устройството отчете движение.

Извън границата – Настройте границата на вашата къща или гараж, ако устройството напусне района на вашия дом, вие ще получите известие веднага.

Live режим – Когато устройството е в движение, вие можете да видите неговата траектория, по този начин получавате ясна информация точно по кой път, улица или магистрала се е движило.

Alarm режим – С този режим вие получавате email с координатите на вашето устройство всеки 5 минути, по този начин можете да предадете информацията на органите на реда в случай на кражба.

Начин на инсталиране & допълнителни разходи – Устройството се монтира без никакво окабеляване или допълнителни разходи. Просто намирате удобно място, за да скриете вашия Oxford Tracker Spy.

Батерия – Батерията е разработена, така че да издържи до 10 години работа или 14 000 отчитания на локация, така че можете да поставите устройството навсякъде!

Устойчив на атмосферни условия – Изцяло водоустойчив със здрава конструкция.

1.1.2 Autocom GPS Tracker



Autocom GPS Bike Tracker прави наблюдението на автомобила ви лесно и приятно. GPS тракерът няма скъпи годишни договори и е оборудван с функции за сигурност, които включват движение и удар, прецизно откриване на местоположението и ограничаване на движението на GEO-ограда.

Местоположението на вашия мотоциклет може да бъде видяна през смартфон, таблет или компютър. Получете информацията за местоположението на вашия мотоциклет при поискване или автоматично и периодично през определен интервал от време. Устройството продължава да проследява дори когато има загуба в GPS сигнала, предлагайки пълна прозрачност на клиентите. Устройството се зарежда от батерията на мотора и също така има допълнителна резервна батерия. Водостойчив и прахоустойчив Autocom GPS Tracker няма да ви предаде, без значение на къде вие и мотоциклета ви сте тръгнали! Тракерът е невероятно лесен за инсталиране, използва стандартна SIM карта, която може да бъде закупена от всеки мобилен оператор. Autocom GPS Tracker не изисква допълнителен абонамент.

Удобствата, които предлага са: лесен за инсталиране и интегриран за бърза настройка и работа с него, отчита скоростта на движение, засича превишение на скоростта, предупреждава когато има загуба на GPS сигнал. Съдържа включена аларма, която се задейства при запалване на двигателя, предупреждение при ниска батерия, следи за напускане на определена зона, съоръжение или сграда, изобразява информацията на смартфон, таблет или компютър, ограничение до определени телефонни номера, които могат да получават информация за устройството, предупреждение за малко оставаща сметка в SIM картата. Тракерът продължава да проследява дори когато няма сигнал, deep sleep mode удължава живота на батерията, когато мотоциклетът е изключен, прахоустойчива и водостойчив със стандарт IP66, размери 89мм на 50мм на 16мм.

Втора глава

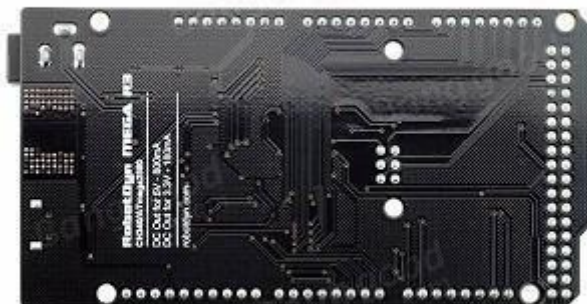
2.1 Избор на хардуерна архитектура

2.1.1 RobotDyn Mega 2560

RobotDyn



**MEGA 2560 R3
+ MicroUSB 2.0 data cable 50cm**



RobotDyn Mega 2560 е микроконтролер, базиран на ATmega2560. Устройството има 54 дигитални пина за вход/изход (14 от които може да бъдат използвани като PWM изходи). 16 аналогови входа, 4 UART (хардуерни серийни портове), 16 MHz-ов кристален осцилатор, USB връзка, захранващ кабел, ICSP хедър и бутон за рестартиране. Съдържа всичко нужно за да поддържа микроконтролера, просто го свързвате към компютър с USB кабел или чрез AC към DC адаптер, или батерия, за да започне да работи.

Характеристика

- Микроконтролер ATmega 2560
- Работно напрежие 5V
- Входно напрежение (препоръчително) 7-9V
- Входно напрежение (ограничения) 6-12V
- 54 цифрови пина (14 от които PWM изход)
- 16 аналогови изхода
- Флаш памет 256 KB , от която 8 KB са заети от bootloader
- 8 KB SRAM
- EEPROM 4KB
- Clock Speed 16MHz

Захранване

RobotDyn Mega 2560 може да бъде захранен чрез USB връзка или чрез допълнително външно захранващо устройство.

Външно захранване (Не USB) може да бъде AC to DC адаптер или батерия. Адаптерът може да бъде свързан с 2.1mm централно-разположен щекер в захранващия вход на борда. Водещите от батерия могат да бъдат поставени в гнездата на Gnd и Vin на конектора POWER.

Платката може да работи на външно захранване от 6 до 12 волта. Ако се захранва с по-малко от 7V, 5V пина може да захранва по-малко от 5 волта и платката може да бъде нестабилна. Ако се захранва с повече от 12V, регулаторът на напрежение може да прегрее и да повреди борда.

Mega 2560 се различава от другите платки по това, че не използва FTDI USB-to serial драйвер чип. Вместо него използва ATmega16U2 (ATmega8U2 във версия 1 и версия 2 на платките), програмиран като USB-serial конвертор.

Ревизия 3 на Mega 2560 съдържа следните подобрения:

1,0 pinout: добавен SDA и SCL пинове, които са близо до AREF пин и два други нови пина, поставени до RESET pin, IOREF позволява на shield-овете да се адаптират към напрежението, което платката им подава. За в бъдеще shield-овете ще бъдат съвместими с двете, платката, която използва AVR и оперира на 5V и с Due версията на платка, която работи с 3.3V.Подобрен RESET алгоритъм. Atmega 16U2 замества 8U2.Пиновете са следните:

VIN. Входа на напрежение към платката, когато използва външно захранващо устройство (различно от 5 волта от USB връзката или друг регулиран източник на захранване). Можете да захраните входното напрежение чрез този пин, или ако се захранва напрежение чрез захранващ кабел, достъп до него през този пин.

5V. Регулираното захранващо напрежение, използвано за да захранва микроконтролера и други компоненти към него. Може да дойде от VIN чрез регулатора или захранено през USB от друг регулиран 5V източник.

3V3. Захранване от 3,3 волта, генерирано от бордовия регулатор. Максималното теглене на тока е 50 mA.

GND. GND пинове.

Памет

Mega 2560 има 256 KB флаш памет за запазване на кода (8 KB от които са заети от bootloader), 8 KB SRAM и 4 KB EEPROM (която може да бъде прочетена и записана с EEPROM библиотеката).

Вход и изход

Всеки един от 54-те цифрови пина на Mega 2560 могат да бъдат използвани като вход и изход, използвайки pin Mode(), digitalWrite() и digitalRead() функции. Те работят на 5 волта. Всеки пин може да осигури или приеме максимум 40 mA и има вътрешен съпротивител (разединен по подразбиране) от 20 до 50 kOhms. В допълнение, някои пинове имат специализирани функции:

Serial 0 (RX) и 1 (TX); Serial 1: 19 (RX) и 18 (TX); Serial 2: 17 (RX) и 16 (TX); Serial 3: 15 (RX) и 14 (TX). Използват се за получаване (RX) и предаване (TX) на TTL serial данни. Пинове 0 и 1 също са свързани към съответните пинове на ATmega16U2 USB-to-TTL Serial чип.

Външни прекъсвания: 2 (прекъсване 0), 3 (прекъсване 1), 18 (прекъсване 5), 19 (прекъсване 4), 20 (прекъсване 3) и 21 (прекъсване 2). Тези пинове могат да бъдат настроени като тригер на прекъсвания при ниска стойност, намаляващ или падащ стойност или промяна в стойността чрез функцията attachInterrupt().

PWM: 0 до 13. Осигуряват 8 битов PWM изход за analogWrite() функцията.

SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Тези пинове поддържат SPI комуникация, използвайки SPI библиотеката.

LED: 13. Съдържа вграден светодиод, свързан с цифров пин 13. Когато пина е със стойност HIGH, светодиодът светва, когато е със стойност LOW, той е изключен.

I2C: 20 (SDA) и 21 (SCL). Поддържат I2C комуникация, използвайки Wire библиотеката. Забележка: тези пинове не са на същото място като I2C пиновете на Duemilanove или Diecimila.

Mega 2560 има 16 аналогови входа, всеки един съдържа 10 битова резолюция (1024 различни стойности). По подразбиране те измерват от земята до 5 волта, въпреки че е възможно да промените горния край на обхвата им, като използвате функцията AREF пин и analogReference().

На платката има още няколко пина:

AREF. Референтно напрежение за аналоговите входове. Използва се с analogReference().

Reset. Използва се за нулиране (LOW) на микроконтролера. Обикновено се използва за добавяне на бутон за на shield-овете с блок.

Комуникация

RobotDyn



RobotDyn Mega 2560 има редица начина за комуникация с компютър, друго Arduino или други микроконтролери. ATmega 2560 предоставя 4 хардуерни UART-а за TTL (5V) serial комуникация. При ATmega16U2 (ATmega8U2 по ревизия 1 и 2 на платката) един от каналите на борда предава през USB виртуален ком порт към софтуера на компютъра (Windows машините имат нужда от .inf файл, но OSX и Linux машините могат да разпознаят борда като COM порт автоматично. Arduino софтуера включва сериен монитор, който позволява прости текстови данни да бъдат изпратени към и от борда. RX и TX светодиода на борда мигат когато се предава информация през ATmega8U2/ATmega16U2 чипа и USB комуникация с компютър (но не за серийна комуникация на пинове 1 и 0).

SoftwareSerial библиотека позволява специална комуникация на всички дигитални портове на Mega2560.

ATmega 2560 също поддържа TWI и SPI комуникация. Arduino софтуера включва Wire библиотека за улеснение на използването на TWI bus. За SPI комуникация се използва SPI библиотеката.

Програмиране

RobotDyn Mega може да бъде програмиран с Arduino софтуера.

ATmega 2560 на платката идва предварително зареден с bootloader, който позволява на програмиста да качи новия код без използване на външна хардуерна програма. То комуникира използвайки оригиналния STK500 протокол (референция, C хедър файлове).

Вие можете за изпуснете bootloader-а и да програмирате микроконтролера през ISCP (In-Circuit Сериийно Програмиране) хедъра. ATmega16U2 е зареден с DFU bootloader, който може да бъде активиран от:

На Rev1 бордове: свързвайки свързващия джъмпер на задната страна на борда и след това нулиране на 8U2.

На Rev2 или по-нови бордове: има резистор, който дърпа 8U2/16U2 HWB линия към земята, което улеснява пускането в DFU режим. След това можете да използвате Atmel FLIP софтуера (Windows) или DFU programmer (Mac OS X и Linux) за да заредите нов фърмуер или да използвате ISP хедъра с външен програмист (презаписване на DFU bootloader-a).

Автоматичен (софтуерен) ресет

Вместо да изисква физическо натискане на ресет бутона преди качване на новия код, RobotDyn Mega2560 е проектиран така, че този процес се прави автоматично от Arduino софтуера. Една от линиите за хардурен контрол (DTR) на ATmega8U2 е свързвана към ресет линията на ATmega 2560 чрез 100 нанофарада кондензатор. Когато тази линия е намалена, ресет линията пада достатъчно дълго, за да нулира чипа. Arduino софтуера използва тази възможност, за да ви позволи да качите код, като просто натиснете бутона за качване в Arduino средата. Това означава, че bootloader-а може има по-малко таймаут време, тъй като понижаването на DTR може да бъде добре кординирано с началото на качването на кода.

Когато Mega 2560 е свързан към работещ компютър, Mac OS X или Linux, то ресетва всеки път, когато е установена връзка с него от софтуера (чрез USB). За следващата около половин секунда bootloader-а започва да работи на платката. Докато е програмиран да игнорира всичко, освен качването на нов код, той ще прочете първите няколко байта данни, изпратени към борда след като връзката е установена. Ако „скеча“ работи на борда, получава еднократна конфигурация или другите данни при първо стартиране, убедете се, че софтуера, с който комуникира изчаква една секунда след отварянето на връзката и преди изпращането на данните.

Платката Mega 2560 съдържа следа, която може да бъде изрязана за да деактивира auto-reset (автоматичния ресет). Подложките на двете страни на следата могат да бъдат запоеени заедно, за да активират отново автоматични ресет, означено е с “RESET-EN”. Вие можете също да деактивирате автоматични ресет, като свържете 110 ohm резистор от 5V към ресет линията.

USB защита

RobotDyn Mega 2560 има полифуза, която може да бъде нулирана. Тя защитава USB портовете на компютъра от по-голям ненужен ток. Макар че по-големите компютри осигуряват тяхна собствена вътрешна защита, предпазителят осигурява допълнителен слой защита. Ако повече от 500 mA ток е приложен към USB порта, предпазителят автоматично ще разруши връзката докато претоварването свърши.

Физическа характеристика и съвместимост

Максималната дължина и широчина на Mega 2560 PCB са съответно 4 и 2.1 инча, като USB конекторът и захранващият жак се простират извън този размер.³ Отворите за винтове позволяват на борда да бъде закачен за повърхност или кутия.

Забележка: разстоянието между цифровите пинове 7 и 8 е 160 мил. (0.16“).

2.1.2 SIM808



SIM808 е Quad-Band GSM/GPRS модул, който комбинира GPS технология за сателитна навигация. Компактния му дизайн, който е интегриран с GPRS и GPS в SMT пакет пести значително количество време и разходи на клиентите, които разработват приложения с GPS модула. Разполагайки със стандартен интерфейс и GPS функция.

Характеристика

- Quad-band 850/900/1800/1900MHz
- GPRS мулти-слот клас 12/10
- GPRS мобилна станция клас B
- Съвместим с GSM phase 2/2+
 - Клас 4 (2 W @ 850/900MHz)
 - Клас 1 (1 W @ 1800/1900MHz)
- Съвместим с Bluetooth: 3.0+EDR
- Контролира се чрез AT команди (3GPP TS 27.007,27.005 и SIMCOM подобрени AT команди)
- Захранващо напрежение: 3.4 ~ 4.4V
- Ниска консумация на ток
- Работна температура: -40°C ~85°C

Начин на свързване към Arduino

Необходим хардуер и софтуер:

- SIM 808 модул
- Arduino борд
- Arduino IDE

Хардуерни връзки

SIM808 трябва да бъде свързан с Arduino борда по следния начин:

- Vcc към 5V
- Gnd към Gnd
- RXD към digital pin 10
- TXD към digital pin 9

Когато борда е включен лампичката PWR ще започне да свети. След дълго натискане на бутона за начало (около 2 секунди) другите 3 лампички (D3, D4, D5) ще започнат да светят. Ако едната от тях започне да премигва, това означава, че SIM808 е свързан правилно и започва работа. Когато захранващият кабел, GSM и GPS антените и SIM картата са свързани правилно към модула лампичката започва да мига бавно (около 3 секунди между всяко премигване). Това сигнализира, че модулет е свързан към мрежата и потребителя може да започне да го използва.

Основните AT команди за използване на SIM808

Команди за GSM

AT Press ENTER

Проверява операцията и връзката с GSM Shield-а. Тази команди ще изпринти ОК връзката към GSM Shield-а е успешна.

ATD+(номер на държава)мобилен номер; Press ENTER

Прави гласово обаждане.

ATH Press ENTER

Прекъсва връзката към Active CallPro.

ATA Press ENTER

Получава и отговаря на обаждане.

AT+CMGF=1 Press ENTER

Изпращане на SMS в текстов формат.

AT+CMGS="мобилен номер" Press ENTER

Когато AT командата е приела „> “ въведете съобщението, което да бъде изпратено чрез SMS. След това натиснете CTRL+Z за да го изпратите. Ако изпращането е било успешно, „OK “ ще бъде изписано на екрана.

AT+CMGF = 1 Press ENTER

Прочита съобщение в текстов формат.

AT+CMGR = num

Number (num) е индексът на съобщението, запазено в SIM картата. За всеки SMS URC ще бъде изписано на екрана като + CMTI: SM 'num' .After this *AT+CMGR=1*.

Команди за Bluetooth

Изпращане на заявка за свързване с друго Bluetooth устройство

AT+BTPOWER=1

Включва Bluetooth.

AT+BTSCAN=1,20

Търсене на други Bluetooth устройства.

AT+BTPAIR=0,6

Опит да се свърже с шестото Bluetooth устройство в списъка(0,6 означава шестия елемент).

AT+BTPAIR =2,0000

Отговор на заявката за свързване в режим с парола

Приемане на заявка от друго Bluetooth устройство

AT+BTPOWER=1

Включва Bluetooth.

AT+BTPAIR=1,1

Приема заявка за свързване.

AT+BTPAIR=2,0000

Приема заявка за свързване когато паролата по подразбиране на другото Bluetooth устройство е 0000.

Свързване към сървис

AT+BTCONNECT=1,2

Свързване със втория профил сървис от устройството, с което има връзка.

Приемане на файл от устройство, с което има връзка

AT+ВТОРРАСПТ=1

Приема файл (записан във вътрешната мемори карта по подразбиране, вход "AT+ВТОРРАСПТ=1,1 ако искате файла да бъде записан във външната памет.

Изпращане на файл към друго Bluetooth устройство

AT+ВТОРРРАСПТ=1, c:\User\BtReceived\link.txt

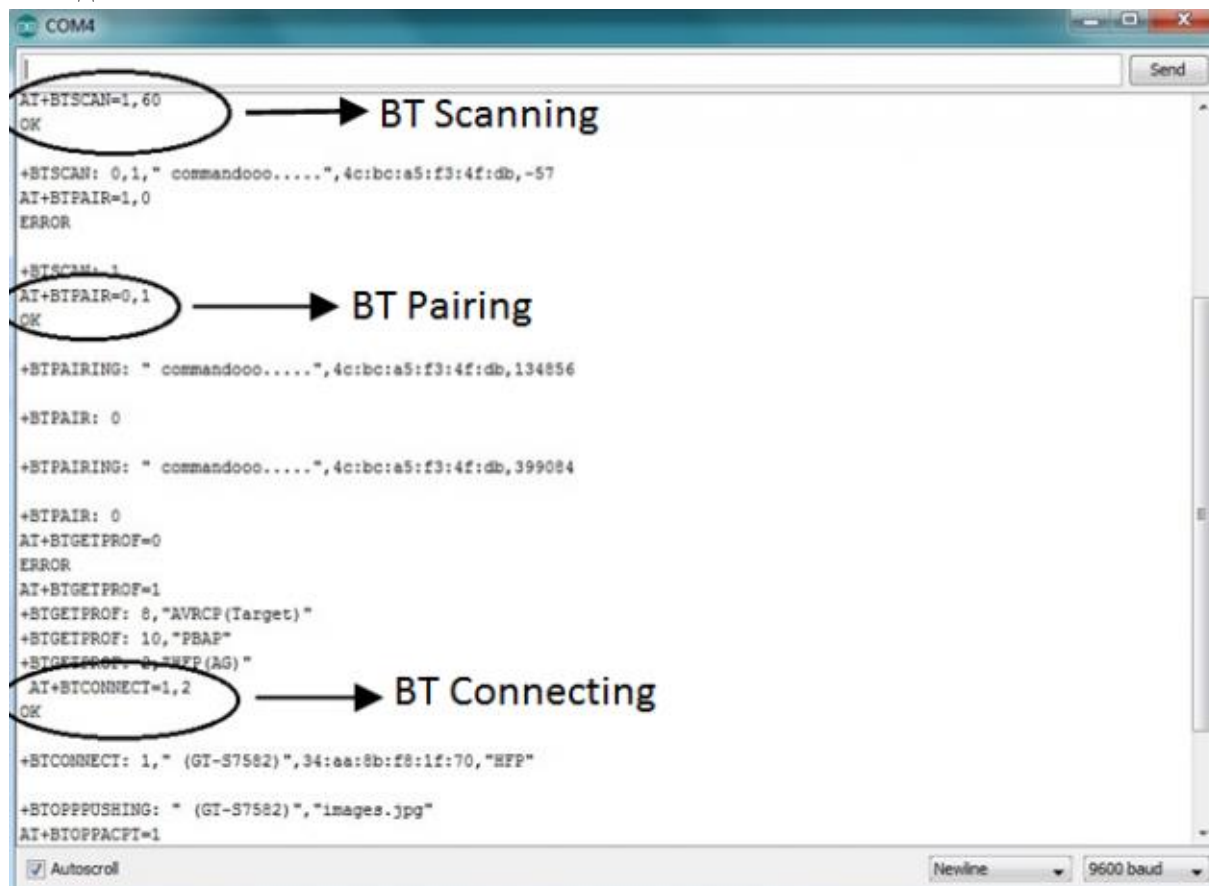
Изпраща файл и изчаква отговор.

Програмиране на SIM808

Примерен код:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9, 10);
void setup()
{
    mySerial.begin(9600);
    Serial.begin(9600);
    delay(100);
}
void loop()
{
    if (Serial.available()>0) mySerial.write(Serial.read());
    if (mySerial.available()>0) Serial.write(mySerial.read());
}
```

Изход:



2.2 Програми, технологии и програмни езици

2.2.1 Програмни езици

Java

Java е общо използван програмен език, който е обектно ориентиран и се базира на обекти. Езикът е специално разработен така, че да има възможно най-малко dependencies (зависимости). Той е предназначен да позволи на програмистите „write once, run anywhere” (WORA). Това означава: компилираният код на Java може да бъде изпълнен на всички платформи, които поддържат Java, без да има нужда програмата да бъде компилирана отново. Java приложенията обикновено се компилират до bytecode, който може да бъде изпълнен на всяка Java virtual machine (JVM), независимо от компютърната архитектура. От 2016 година Java е един от най-известните програмни езици, които се използват в момента, главно за client-server web приложения с над 9 милиона програмисти, които използват езика. Java е разработена от James Gosling в Sun Microsystems (която оттогава е придобита от Oracle Corporation) и пусната през 1995 година като основен компонент от Sun Microsystems Java platform. Езикът наподобява много по синтаксис на C и C++, но има по-малко съражения от ниско ниво от двата езика.

Оригиналната реализация и имплементация на Java компилатори, виртуални машини и класови библиотеки са пуснати от Sun под техен лиценс за собственост. От май 2007 година, в съответствие със спецификите на Java Community Process, фирмата Sun лицензира отново повечето от техните Java технологии под GNU General Public License. Други също са разработили алтернативни имплементации на тези Sun технологии като GNU Compiler for Java (bytecode compiler), GNU Classpath (standart libraries), и IcedTea-Web (browser plugin for applets).

Последната версия на Java е пусната на 21 септември 2017 година. Та е една от двете версии, които в момента се поддържат безплатно от Oracle. По-ранните версии от Java 8 се поддържат от компании с комерсиална цел; пример от Oracle от Java 6 от октомври 2017 година, който все още „силно препоръчват да изтриете pre-Java 8 от повечето компютри“.

James Gosling, Mike Sheridan и Patrick Naughton започват проекта за Java езика през юни 1995 година. Първоначално Java била предназначена за интерактивна телевизия, но била прекалено напреднала за цифровата индустрия за кабелна телевизия по това време. Езикът първоначално се наричал Oak (дъб), носел името на едно дъбово дърво, което се намирало пред Офиса на Gosling. По-късно проекта бил известен под името Green, в последствие отново преименуван на Java, от Java coffee. Gosling проектирал Java със синтаксис подобен на C/C++, за да бъде познат на повечето програмисти по това време.

Sun Microsystems пуска първата публична имплементация на Java под името Java 1.0 през 1995 година. Тя обещава „write once, run anywhere” (WORA), което осигурява безпроблемна работа на известните платформи. Сравнително сигурна и снабдена с конфигурационна сигурност, тя позволявала интернет и файл достъп ограничения. Основните уеб браузъри скоро включили поддръжка на Java в уеб страниците и Java бързо станала популярна. Java 1.0 компилаторът бил написан отново в Java от Arthur van Hoff, за да спазва стриктно спецификите на Java 1.0. С появата на Java 2 (Пусната първоначално като J2SE 1.2 през декември 1998-1999), новите версии имали по няколко конфигурации, създадени за различни типове платформи. J2EE включвала технологии и API-та за enterprise приложения, обикновено изпълнявани в

сървърни среди, докато J2ME предлагала API-та оптимизирани за мобилни приложения. Desktop версията била преименувана на JSSE. През 2006 година за маркетингови цели, Sun преименувала новите J2 версии съответно на Java EE, Java ME и Java SE.

През 1997 Sun Microsystems се обръщат към органа по стандартите на ISO/IEC JTC 1 и по-късно Ecma International формализира Java, но след това прекъсват този процес. Java остава de facto standart, контролирана чрез Java Community Process. По едно време Sun прави повечето от Java имплементациите достъпни без такси, въпреки техния статус на софтуера. Sun придобива приходи чрез продажбата на лицензи за специализирани продукти като Java Enterprise System.

На 13 ноември 2006 година Sun пуска голяма част от техните Java virtual machine (JVM) като free and open-source software (FOSS), съгласно условията на GNU General Public License (GPL). На 8 май 2007 година Sun приключва с този процес, като прави всички JVM core code свободни съгласно условията на free software/open-source, освен малка част от кода, на която Sun не притежава авторски права.

С

С е език за програмиране. С поддържа структурирано програмиране, лексикален променлив обхват и рекурсия, докато система със статичен тип предотвратява много непредвидими операции. По дизайн, С предоставя конструкции, които ефективно картографират типичните машинни инструкции и затова е намерил трайно използване в приложения, който преди това са били кодирани в програмния език асемблер, включващи операционни системи, също така и различни приложения за компютри, вариращи от суперкомпютри до ебедед системи.

С първоначално е разработен от Dennis Ritchie между 1969 и 1973 година в Bell Labs и е използван за да реимплементира на Unix операционните системи. От тогава езикът е един от най-широко използваните програмни езици на всички времена със С компилатори, които варират от различни доставчици на разположение от повечето от съществуващите компютърни архитектури и операционни системи. С е стандартизиран от American National Standards Institute (ANSI) от 1989 година и впоследствие от International Organization for Standardization (ISO).

С е императивен процедурен език. Той е проектиран за да осигури ниско ниво на достъп до паметта и за да предостави езикови конструкции, които ефективно да картографират кода на машинни инструкции и да изискват минимална поддръжка за да могат да бъдат изпълнени. Въпреки способностите му на ниско ниво, езикът е проектиран да насърчи програмиране между различните платформи. Стандартна програма, написана на С може да бъде компилирана за много широко разнообразие от компютърни платформи и операционни системи с малко промени по source кода. Езикът е достъпен на много широк спектър от платформи, от ебедед микроконтролери до суперкомпютри.

Като повечето императивни езици в ALGOL традиция, С има удобствата като структурирано програмиране и позволява лексикален променлив обхват, и рекурсия, докато статична тип система предотвратява много нежелани операции. В С всички executable код се съдържа в подпрограми, които се наричат „функции“ (макар и не в строгия смисъл на функционалното програмиране). Параметрите на функциите винаги се подават по стойност.

Много други езици са взели директно или индиректно от С включвайки: C++, C#, Unix C shell, D, Go, Java, JavaScript, Limbo, LPC, Objective-C, Perl, PHP, Python, Rust, Swift и Verilog (език за описване на хардуер). Тези езици са взели много от контролните и другите прости черти от С. Повечето от тях (С изключение на Python)

приличат много по синтаксис на C, като цяло те комбинират познатите експресивни и стейтмънти от синтаксиса на C с базови тип системи, модели на данни и семантика, които могат да бъдат коренно различни.

Произхода на C е близо свързан с разработката на Unix операционните системи, първоначално имплементирани в асембъл на PDP-7 от Dennis Ritchie и Ken Thompson, включващи някои идеи от колеги. Те решили да свържат чрез портове операционната система с PDP-11. Първоначалната версия на PDP-11 от Unix била разработена на assembler. Създателите обмисляли да пренапишат системата използвайки програмния език B, опростената версия на Thompson от BCPL. Името на езика C било избрано просто защото е следващата буква след B.

2.2.2 Технологии

MySQL

MySQL е open source система за управление на релационни бази данни (RDBMS). Името му е комбинация от "My, името на дъщерята на съоснователя Michael Widenius и SQL, съкращението на Structured Query Language (език за структуризирани заявки). MySQL проектът е open source под GNU General Public лиценз, както и под разнообразие от собствени споразумения. MySQL е била собственост и спонсориран от шведската компания MySQL AB, в момента собственост на Oracle Corporation. За собствена употреба налице са няколко платени версии, които предлагат допълнителни функции.

MySQL е основният компонент от LAMP open-source уеб софтуер стак от приложения. LAMP е акроним на "Linux, Apache, MySQL, Perl/PHP/Python". Приложения, които използват MySQL база данни: TYPO3, MODx, Joomla, WordPress, Simple Machines Forum, phpBB, MyBB и Drupal. MySQL е също използван в много висок профил сайтове в голям мащаб, включвайки Google (не за търсене), Facebook, Twitter, Flickr и YouTube. MySQL е написан на C и C++. Неговия SQL парсър е написан на yacc, но използва домашно приготвен лексикален анализатор. MySQL работи на много системни платформи, включвайки AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, MacOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos и Tru64. Порт към OpenVMS от MySQL също съществува.

MySQL сървърния софтуер и клиент библиотеките използват дистрибуция с двойно лицензиране. Те се предлагат по GPL версия 2, от 28 Юни 2000 година (от 2009 година е удължен с изключение FLOSS лиценз) или под собствен лиценз.

Помощ може да се получи от официалния наръчник. Безплатен съпорт допълнително е достъпен в различни IRC форуми и сайтове. Oracle предлага платен съпорт чрез техните MySQL Enterprise продукти. Те се различават в обхвата на услуги и цена. Допълнително това съществуват редица организации от трети страни, които предоставят съпорт и услуги, включвайки MariaDB и Percona.

MySQL е получил добри отзиви и рецензенти забелязват, че софтуера "се представя изключително добре в среден клас", и че "има интерфейси за програмисти, а документацията (да не говорим за обратна връзка в реалния свят чрез уеб сайтове и подобни) е много, много добра". Софтуера също така е бил тестван за "бърз, стабилен и истински многопотребителски сървър на база данни с множество нишки".

MySQL е създаден от шведската компания MySQL AB, с основатели David Axmark, Allan Larsson и Michael "Monty" Widenius. Първоначално разработването на MySQL започва от Widenius и Axmark през 1994 година. Първата версия на MySQL излиза на

23 Май 1995 година. Първоначално била създадена за лична употреба от mSQL, базиран на ниско ниво език ISAM, който създателите преценили за твърде бавен и негъвкав. Те създали нов SQL интерфейс, докато пазели същото API като mSQL. Чрез поддържането на API-то съвместимо с mSQL системата много програмисти могли да използват MySQL вместо предшественика на SQL, който бил предмет на собственост.

Spring Framework

Spring Framework е framework за Java. Функциите на Spring могат да бъдат използвани от всяко Java приложение. Съществуват разширения за правенето на уеб приложения на платформата Java EE (Enterprise версията). Въпреки, че Spring не предлага специфичен модел на програмиране, той става известен в Java общността като добавка или дори заместник на Enterprise JavaBeans (EJB) модела. Spring Framework е опън сорс.

Първата версия на Spring Framework е написана от Rod Johnson. Той пуска Framework-a с публикацията на книгата си Expert One-on-One J2EE Design and Development през октомври 2002 година. Spring Framework-a излиза под Apache 2.0 лиценс през юли 2003 година. Първото издание с версия 1.0 е пусната през март 2004 година, със следващи издания през септември 2004 и март 2005 година. Spring 1.2.6 Framework печели Jolt productivity award и JAX (Java API for XML) Innovation Award през 2006 година. Spring 2.0 излиза през октомври 2006, Spring 2.5 през ноември 2007, Spring 3.0 през декември 2009, Spring 3.1 през декември 2011 и Spring 3.25 през ноември 2013 година. Spring Framework 4.0 излиза през декември 2013. Забележими подобрения в Spring 4.0 са включена поддръжка на: Java SE (Standard Edition) 8, Groovy 2, някои аспекти от Java EE 7 и WebSocket.

Spring Framework 4.2.0 излиза на 31 юли 2015 година, и веднага е заменен от версия 4.2.1, която излиза на 1 септември 2015, която е съвместима с Java 6, 7 и 8.

Spring Framework 4.3 излиза на 10 юни 2016 година. Версията 4.3.0 RC1 е на разположение „Тя ще бъде последното поколение в рамките на общите изисквания на системата Spring 4 (Java 6+, Servlet 2.5+), подготвя се за удължена 4.3.x поддръжка до 2019 година.

Spring 5 е обявена за изградена върху Reactive Streams, съвместима с Reactor Core.

2.2.3 Програми

IntelliJ IDEA

IntelliJ IDEA е Java интегрирана среда за разработване (IDE) на компютърен софтуер. Разработена е от JetBrains (Познати като IntelliJ) и е налична като Apache 2 лицензирана community версия, и е патентована комерсиална версия. И двете могат да бъдат използвани за комерсиална разработка на софтуер.

Първата версия на IntelliJ IDEA е от януари 2001 година и била една от първите Java интегрирана среда за разработване (IDE) с усъвършенствана навигация в кода и възможности за преструктуриране.

През 2010 година IntelliJ получава най-високия резултат в тест-центъра от четирите най-известни софтуери за разработка на софтуер: Eclipse, IntelliJ IDEA, NetBeans и JDeveloper.

През декември 2014 година Google съобщават за версията 1.0 на Android Studio, open source IDE за Андроид приложения, базирана на open source community версията от IntelliJ IDEA. Други интегрирани среди за разработване на софтуер, базирани на основата на IntelliJ са: AppCode, CLion, PhpStorm, PyCharm, RubyMine, WebStorm и MPS.

Android Studio

Android Studio е официалната интегрирана среда за разработване (IDE) за операционната система Android, базирана на основата на продукта на JetBrains IntelliJ IDEA, Android studio е специално разработена за разработване на Android приложения. Програмата може да бъде използвана на Windows, macOS и Linux базирани операционни системи. Android studio е заместител на Eclipse интегрираната среда за разработване (ADT) , която до сега е основно IDE за разработка на приложения за Android.

Android Studio излиза на 16 май 2013 година на Google I/O конференция. Програмата е била в early access, започвайки от версия 0.1 на 1 май 2013 година до бета версия, която започва от версия 0.8 през юни 2014 година. Първата стабилна версия излиза през декември 2014 година, тя започва от версия 1.0. В момента версията на програмата е 3.0, която излиза през октомври 2017 година.

Arduino IDE

Arduino IDE е кръстосана платформа, написана на Java. Произлиза от другите интегрирани среди за разработване (IDE). Включва редактор за код с опции като: изрязване , копиране, поставяне, търсене и заменяне на текст. Съдържа още автоматично удебеляване на съвпадащи скоби, осветяване на синтаксиса и с едно натискане кодът се компилира и качва на Arduino устройството. В програмата има също така и пространство за съобщения, текст конзола, лента с инструменти, бутони за често срещаните функции и йерархия от менюта с операции за работа.

Програма, написана с IDE за Arduino се нарича „скица“. „Скиците“ се запамятват в компютъра като текстови файлове с разширение .ino. Преди версия 1.0 на Arduino IDE „скиците“ били с разширение .pde.

Arduino IDE поддържа програмните езици C и C++ , използвайки специални правила за структуриране на кода. Arduino IDE предоставя много библиотеки, които осигуряват много процедури за въвеждане и извеждане. Кодът, написан от потребителя изисква само две прости функции: за започване на „скицата“ и основния цикъл, които се компилират и свързват в една изпълнима циклична програма с GNU верига от инструменти, също включена в IDE софтуера. Arduino IDE използва програмата avrdude за да преобразува изпълнимия код в текст файл, който е шестнадесетично кодиран и този файл се зарежда в Arduino устройството.

2.3 Изисквания към програмния продукт

2.3.1 Изисквания към Android приложението

Android приложението трябва да съдържа следните възможности:

- Да предоставя функционалността всеки потребител да може да управлява и следи до 5 устройства от неговия телефон.
- Да показва къде в момента се намират всички устройства, които са свързани с акаунта на потребителя на картата.
- Режим „паркиран“, който да може да бъде активиран и деактивиран от потребителя. Когато режим „паркиран“ е включен приложението, трябва независимо дали приложението е включено да проверява локацията на устройството и в случай на промяна на координатите да изпраща известие на потребителя.
- Да предоставя възможност за избиране на устройство, което да следи в момента от всички устройства на потребителя.
- Да предоставя меню с настройки, което включва:
 - Промяна на периода от време, през който устройството изпраща текущата локация на устройството към сървъра.
 - Проверка на текущото устройство с информация за него.
- Да предоставя опция за добавяне на ново устройство към акаунта на потребителя.

2.3.2 Изисквания към Arduino устройството

- При първоначално включване да се идентифицира пред сървъра като предостави своя уникален пин номер.
- През определен период от време да прави заявка към сървъра, взимайки текущата конфигурация за него.
- Да измерва и изпраща на сървъра текущата си позиция през определения период от време, зададен от текущата конфигурация на устройството.
- Да работи в режим „пестене на енергия, като пести енергия, когато устройство не е заето да комуникира със сървъра.
- При всяка комуникация със сървъра да проверява дали не е в режим „откраднат“. При този режим устройството преминава в „буден режим“. Тогава започва да измерва и изпраща текущата си локация към сървъра през много по-малък период от време.

2.3.3 Изисквания към свързване на устройството

- Всички връзки да бъдат направени по възможно най-здравия и функционален начин за да бъдат устойчиви на сътресения.
- Всички компоненти на устройството и самото то да бъдат запечатани във кутия.
- Да бъде свързано към външно захранващо устройство или външна батерия.
- Захранващата батерия да бъде свързана към акумулаторната батерия на мотоциклета, за да бъде зареждана с електричество само когато мотора е включен.

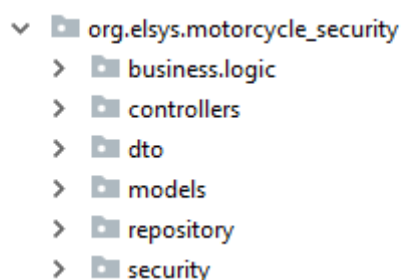
Трета глава

3.1 Реализация на сървъра

Сървърът изпълнява ролята на посредник между базата с данни MySQL, Arduino устройството и Android приложението. Програмиран е на Java, използвайки Spring framework-а в работната среда IntelliJ. Чрез http заявки се осъществява връзката със базата с данни, като по този начин се добавят, променят или вземат записи от там.

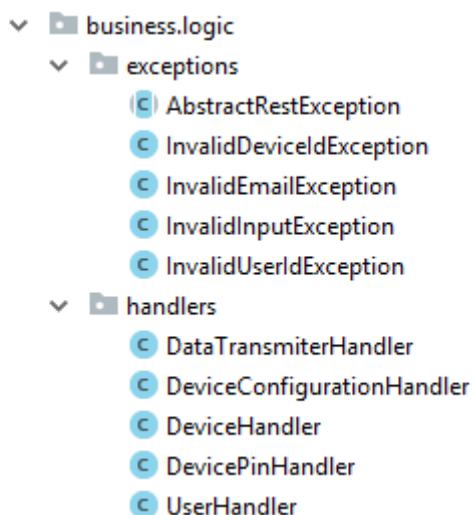
3.1.1 Разделение на кода

Проекта за реализиране на сървъра е разделен на 6 пакета:



3.1.2 business.logic package

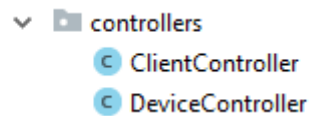
В този пакет се намират handler-ите за всяко отделно entity, както и exception handler-ите.



В подпакета exceptions се съдържат exception-ите, които мога да бъдат хвърлени при всяка една от заявките. В handler класовете се осъществява цялата обработка по данните от базата с данни.

3.1.3 controllers package

В този пакет се намират 2-та контролера, разделени съответно за Arduino устройството и за Android приложението.



Заявките са разделени на два типа: device заявки и client заявки. Device заявките са тези, които са програмирани специално за Arduino устройството. Client заявките са тези, които се използват от Android приложението за комуникация с базата данни.

В ClientController-a се намират всички заявки, които са предназначени за Android приложението:

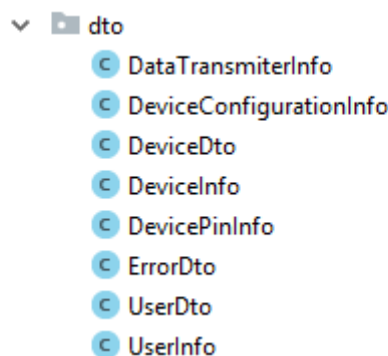
```
/login  
/client/send/parking-status  
/client/send/timeout  
/client/send/parked-coordinates  
/client/send/stolen-status  
/client/send/change-password  
/client/send/create-new-user  
/client/send/create-new-device  
/client/{deviceId}/receive/gps-coordinates  
/client/{deviceId}/receive/gps-coordinates-for-day  
/client/receive/user-account  
/client/{deviceId}/receive/device  
/client/{deviceId}/receive/device-pin
```

В DeviceController-a се намират всички заявки, които са предназначени за Arduino устройството:

```
/device/send/gps-coordinates  
/device/{deviceId}/receive/device-configuration
```

3.1.4 dto package

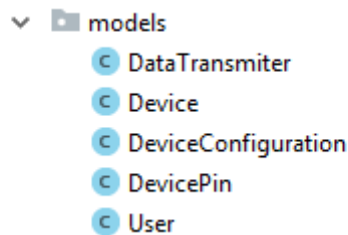
В този пакет се намират Info и Dto класовете за всички модели.



Info класовете служат за структуриране на информацията, която SQL сървърът връща под формат, който да може да бъде прочетен от HTTP апито на Android приложението. Dto класовете служат за структуриране на информацията, която се подава от Android приложението, така че да може да бъде прочетена от сървър.

3.1.5 models package

В този пакет се съдържат моделите на таблиците в базата данни. Чрез тях сървърът генерира MySQL таблиците.



В таблицата DataTransmitter се съдържат координатите по x и y, скоростта на движение, както и датата, в която е изчислена позицията на всяко устройство.

В таблицата Device се съдържат всички устройства с техния идентификационен номер и номера на потребителя, за който е свързано устройството и ако е паркирано, координатите, където е паркирано. Тази таблица съединява устройства с потребителите.

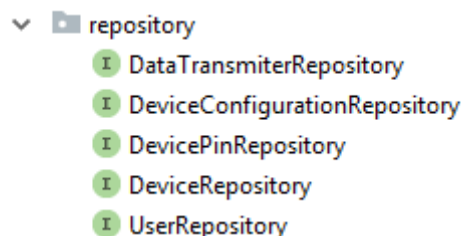
В таблицата DeviceConfiguration се съдържат настройките за всяко устройство: през колко време да изпраща координати устройството и дали е в паркиран режим.

В таблицата DevicePin се съдържат всички възможни номера на устройства.

В таблицата User се съдържат всички потребители и номерата на устройствата, които притежават.

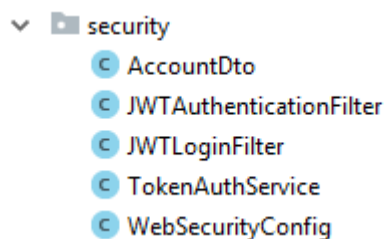
3.1.6 repository package

В този пакет се съдържат репозиторитата за всяка отделна таблица. Чрез тях се взимат, променят или запамятват записите в таблиците.



3.1.7 security package

В този пакет се съдържа системата за сигурност на сървъра. Представява система за сесии. Когато потребителя влезе в акаунта си от Android приложението, той получава токен, чрез който приложението може да получи достъп до заявките на сървъра.



3.1.8 Device заявки

3.1.8.1 /device/send/gps-coordinates

```
@RequestMapping(value="/device/send/gps-coordinates",method=POST)
public ResponseEntity sendGpsCoordinates(@RequestParam(value="deviceId") String
deviceId,
                                         @RequestParam(value="x", defaultValue="0") double x,
                                         @RequestParam(value="y", defaultValue="0") double y,
                                         @RequestParam(value="speed", defaultValue="0") double
speed) {
    try {
        dataTransmitterHandler.updateGPSCoordinates(deviceId, x, y, speed);
    }
    catch(InvalidDeviceIdException exception) {
        return new ResponseEntity(new ErrorDto(exception),HttpStatus.BAD_REQUEST);
    }
    catch(InvalidInputException exception) {
        return new ResponseEntity(new ErrorDto(exception),HttpStatus.BAD_REQUEST);
    }
    return new ResponseEntity(HttpStatus.OK);
}
```

Целта на тази заявка е да изпрати координатите “x” (latitude), “y” (longitude) и скоростта на движение за устройството с посочения “deviceId”. Със заявката се записва и датата, в която е направено изчислението във таблицата data_transmitter.

Тип: POST

Параметри:

String deviceId,

double x,

double y,

double speed

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" - Изтекла или невалидна сесия.

message: "Invalid device id" - Невалидно устройство.

message: "Invalid input" - Въведени са невалидни данни.

```
public void updateGPSCoordinates(String deviceId, double x, double y, double speed)
{
    Device device = deviceRepository.getDeviceById(deviceId);
    if(device == null) throw new InvalidDeviceIdException("Invalid device id");
    if(x == 0 || y == 0) throw new InvalidInputException("Invalid input");
    DataTransmitter d = new DataTransmitter();
    d.setX(x);
    d.setY(y);
    d.setSpeed(speed);
    Long date = System.currentTimeMillis();
    d.setTime(date);
    d.setDevice(device);
    dataTransmitterRepository.save(d);
}
```

Добавя новия запис с координатите x, y, скоростта на движение и текущата дата.

3.1.8.2 /device/{deviceId}/receive/device-configuration

```
@RequestMapping(value="/device/{deviceId}/receive/device-configuration",method=GET)
    @ResponseBody
    public ResponseEntity getDeviceConfigurationDeviceId(@PathVariable
(value="deviceId") String deviceId) {
        DeviceConfigurationInfo deviceConfigurationInfo;
        try {
            deviceConfigurationInfo =
deviceConfigurationHandler.getDeviceConfiguration(deviceId);
        }
        catch(InvalidDeviceIdException exception) {
            return new ResponseEntity(new
ErrorDto(exception),HttpStatus.BAD_REQUEST);
        }
        return new ResponseEntity(deviceConfigurationInfo,HttpStatus.OK);
    }
}
```

Целта на тази заявка е да върне конфигурацията на устройството с горепосоченото „deviceId”.

Тип: GET

Параметри:

String deviceId

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" – Изтекла или невалидна сесия.

message: "Invalid device id" – Невалидно устройство.

При успешна заявка:

```
{
    timeout: long,
    parked: boolean
    stolen: boolean
}
```

```
public DeviceConfigurationInfo getDeviceConfiguration(String deviceId) {
    DeviceConfiguration deviceConfiguration =
deviceConfigurationRepository.getDeviceConfigurationByDeviceId(deviceId);
    if(deviceConfiguration == null) throw new InvalidDeviceIdException("Invalid
device id");
    return new DeviceConfigurationInfo(deviceConfiguration);
}
```

Търси в базата за запис с горепосочения „deviceId” и ако намери такъв, го връща като DeviceConfigurationInfo обект.

3.1.9 Client заявки

3.1.9.1 /client/send/parking-status

```
@RequestMapping(value = "/client/send/parking-status", method = PUT)
public ResponseEntity updateParkingStatusByDeviceId(@RequestParam(value =
"deviceId") String deviceId, @RequestParam(value = "isParked") boolean isParked) {
    try {
        deviceConfigurationHandler.updateParkingStatus(deviceId, isParked);
    }
    catch(InvalidDeviceIdException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    catch(InvalidInputException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    return new ResponseEntity(HttpStatus.OK);
}
```

Целта на тази заявка е да изпрати на сървъра boolean, който показва дали устройството е в режим „паркиран“ или не.

Тип: PUT

Параметри:

String deviceId,
boolean isParked

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" – Изтекла или невалидна сесия.

message: "Invalid device id" – Невалидно устройство.

message: "Invalid input" – Въведени са невалидни данни.

```
public void updateParkingStatus(String deviceId, boolean isParked) {
    DeviceConfiguration deviceConfiguration =
deviceConfigurationRepository.getDeviceConfigurationByDeviceId(deviceId);
    if(deviceConfiguration == null) throw new InvalidDeviceIdException("Invalid
device id");
    if(isParked == true || isParked == false) {
        deviceConfiguration.setParked(isParked);
        deviceConfigurationRepository.save(deviceConfiguration);
        DataTransmitter dataTransmitter =
dataTransmitterRepository.getGpsCoordinatesByDeviceId(deviceId);
        Device device = deviceRepository.getDeviceByDeviceId(deviceId);
        device.setParkedX(dataTransmitter.getX());
        device.setParkedY(dataTransmitter.getY());
        deviceRepository.save(device);
    }
    else throw new InvalidInputException("Invalid input");
}
```

Променя записа с „deviceId“, подаден на функцията като променя isParked полето в true или false. Данните от заявката се записват в таблицата device_configuration.

3.1.9.2 /client/send/timeout

```
@RequestMapping(value = "/client/send/timeout", method = PUT)
public ResponseEntity updateTimeoutByDeviceId(@RequestParam(value = "deviceId")
String deviceId, @RequestParam(value = "timeout", defaultValue = "0") long timeout)
{
    try {
        deviceConfigurationHandler.updateTimeOut(deviceId, timeout);
    }
    catch(InvalidDeviceIdException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    catch(InvalidInputException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    return new ResponseEntity(HttpStatus.OK);
}
```

Целта на тази заявка е да изпрати на сървъра long , който съдържа времето в милисекунди, през което устройството трябва да изпраща текущата си позиция.

Тип: PUT

Параметри:

String deviceId,
long timeout

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" - Изтекла или невалидна сесия.

message: "Invalid device id" - Невалидно устройство.

message: "Invalid input" - Въведени са невалидни данни.

```
public void updateTimeOut(String deviceId, long timeout) {
    DeviceConfiguration deviceConfiguration =
deviceConfigurationRepository.getDeviceConfigurationByDeviceId(deviceId);
    if(deviceConfiguration == null) throw new InvalidDeviceIdException("Invalid
device id");
    if(timeout == 0) throw new InvalidInputException("Invalid input");
    deviceConfiguration.setTimeout(timeout);
    deviceConfigurationRepository.save(deviceConfiguration);
}
```

Променя записа с „deviceId“, подаден на функцията, като променя timeout полето с подадената стойност от функцията. Данните от заявката се записват в таблицата device_configuration.

3.1.9.3 /client/send/parked-coordinates

```
@RequestMapping(value = "/client/send/parked-coordinates", method = PUT)
public ResponseEntity updateParkedCoordinates(@RequestParam(value = "deviceId")
String deviceId,
                                           @RequestParam(value = "x",
defaultValue = "0") double x,
                                           @RequestParam(value = "y",
defaultValue = "0") double y) {
    try {
        deviceHandler.updateParkedCoordinates(deviceId, x, y);
    }
    catch(InvalidDeviceIdException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    catch(InvalidInputException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    return new ResponseEntity(HttpStatus.OK);
}
```

Целта на тази заявка е да изпрати координатите “x” (latitude) и “y” (longitude) за устройството с посочения “deviceId” точно когато устройството влиза в режим „паркиран“.

Тип: PUT

Параметри:

String deviceId,

long timeOut

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" – Изтекла или невалидна сесия.

message: "Invalid device id" – Невалидно устройство.

message: "Invalid input" – Въведени са невалидни данни.

```
public void updateParkedCoordinates(String deviceId, double x, double y) {
    Device device = deviceRepository.getDeviceByDeviceId(deviceId);
    if(device == null) throw new InvalidDeviceIdException("Invalid device id");
    if(x == 0 || y == 0) throw new InvalidInputException("Invalid input");
    device.setParkedX(x);
    device.setParkedY(y);
    deviceRepository.save(device);
}
```

Променя записа с „deviceId”, подаден на функцията като променя parkedX и parkedY полетата с подадените стойности от функцията. Данните от заявката се записват в таблицата devices.

3.1.9.4 /client/send/stolen-status

```
@RequestMapping(value = "/client/send/stolen-status", method = PUT)
public ResponseEntity updateTimeoutByDeviceId(@RequestParam(value = "deviceId")
String deviceId, @RequestParam(value = "isStolen") boolean isStolen) {
    try {
        deviceConfigurationHandler.updateStolenStatus(deviceId, isStolen);
    }
    catch(InvalidDeviceIdException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    catch(InvalidInputException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    return new ResponseEntity(HttpStatus.OK);
}
```

Целта на тази заявка е да изпрати на сървъра boolean, който показва дали устройството е в режим „паркиран“ или не.

Тип: PUT

Параметри:

String deviceId,
boolean isStolen

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" – Изтекла или невалидна сесия.

message: "Invalid device id" – Невалидно устройство.

message: "Invalid input" – Въведени са невалидни данни.

```
public void updateStolenStatus(String deviceId, boolean isStolen) {
    DeviceConfiguration deviceConfiguration =
deviceConfigurationRepository.getDeviceConfigurationByDeviceId(deviceId);
    if(deviceConfiguration == null) throw new InvalidDeviceIdException("Invalid
device id");
    if(isStolen == true || isStolen == false) {
        deviceConfiguration.setStolen(isStolen);
        deviceConfigurationRepository.save(deviceConfiguration);
    }
    else throw new InvalidInputException("Invalid input");
}
```

Променя записа с „deviceId“, подаден на функцията като променя isStolen полето в true или false. Данните от заявката се записват в таблицата device_configuration.

3.1.9.5 /client/send/change-password

```
@RequestMapping(value = "/client/send/change-password", method = PUT)
public ResponseEntity updateTimeoutByDeviceId(@RequestParam(value = "userId") long
userId, @RequestHeader(value = "oldPassword") String oldPassword,
@RequestHeader(value = "newPassword") String newPassword) {
    try {
        userHandler.updatePassword(userId, oldPassword, newPassword);
    }
    catch(InvalidUserIdException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    catch(InvalidInputException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    return new ResponseEntity(HttpStatus.OK);
}
```

Целта на тази заявка е да промени паролата на съществуващ потребител.

Тип: PUT

Параметри:

long userId,

Хедъри:

String oldPassword

String newPassword

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" – Изтекла или невалидна сесия.

message: "Invalid user id" – Невалиден потребител.

message: "Invalid input" – Въведени са невалидни данни.

message: "Invalid old password" – Невалидна стара парола.

```
public void updatePassword(long userId, String oldPassword, String newPassword) {
    User user = userRepository.getUserAccountById(userId);
    if(user == null) throw new InvalidUserIdException("Invalid user id");
    if(newPassword.length() == 0) throw new InvalidInputException("Invalid input");
    if(oldPassword.matches(user.getPassword())) {
        user.setPassword(newPassword);
        userRepository.save(user);
    }
    else throw new InvalidInputException("Invalid old password");
}
```

Променя записа с „userId”, подаден на функцията като променя password полето. Данните от заявката се записват в таблицата users.

3.1.9.6 /client/send/create-new-user

```
@RequestMapping(value = "/client/send/create-new-user", method = POST)
public ResponseEntity createNewUser(@RequestBody UserDto newUser) {
    try {
        userHandler.createNewUser(newUser);
    }
    catch(InvalidInputException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    return new ResponseEntity(HttpStatus.OK);
}
```

Целта на тази заявка е да създаде нов потребител в таблицата users. Заявката изпраща обект от тип UserDto.

Тип: POST

Параметри:

UserDto newUser

Тяло:

```
{
  "email":String,
  "password":String,
  "devices":
  [
    {
      "deviceId":String
    }
  ]
}
```

Отговор от сървър:

При неуспешна заявка:

message: "Invalid input" - Въведени са невалидни данни.

```
public void createNewUser(UserDto userDto){
    ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
    Validator validator = factory.getValidator();
    long violations = validator.validate(userDto).size();
    if(violations>0) throw new InvalidInputException("Invalid input");
    User user = new User();
    user.setPassword(userDto.getPassword());
    user.setEmail(userDto.getEmail());
    userRepository.save(user);
    for(DeviceDto userDeviceDto: userDto.getDevices()){
        Device device = new Device();
        device.setUser(user);
        device.setDeviceId(userDeviceDto.getDeviceId());
        deviceRepository.save(device);
        user.getUserDevices().add(device);
        DeviceConfiguration deviceConfiguration = new DeviceConfiguration();
        deviceConfiguration.setDevice(device);
        deviceConfigurationRepository.save(deviceConfiguration);
    }
}
```

Създава нов запис в таблицата users, създава нов запис в таблица devices и device_configuration. Свързва id-то на потребителя с id-то на устройството.

3.1.9.7 /client/send/create-new-device

```
@RequestMapping(value = "/client/send/create-new-device", method = POST)
public ResponseEntity createNewDevice(@RequestBody DeviceDto newDevice) {
    try {
        deviceHandler.createNewDevice(newDevice);
    }
    catch(InvalidInputException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
    return new ResponseEntity(HttpStatus.OK);
}
```

Целта на тази заявка е да създаде ново устройство в таблицата devices. Заявката изпраща обект от тип DeviceDto.

Тип: POST

Параметри:

DeviceDto newDevice

Тяло:

```
{
  "deviceId":String,
  "userId":long
}
```

Отговор от сървъра:

При неуспешна заявка:

message: "Invalid input" – Въведени са невалидни данни.

```
public void createNewDevice(DeviceDto deviceDto){
    ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
    Validator validator = factory.getValidator();
    long violations = validator.validate(deviceDto).size();
    if(violations>0) throw new InvalidInputException("Invalid input");
    Device device = new Device();
    User user = userRepository.getUserAccountById(deviceDto.getUserId());
    device.setUser(user);
    device.setDeviceId(deviceDto.getDeviceId());
    user.getUserDevices().add(device);
    DeviceConfiguration deviceConfiguration = new DeviceConfiguration();
    deviceConfiguration.setDevice(device);
    userRepository.save(user);
    deviceRepository.save(device);
    deviceConfigurationRepository.save(deviceConfiguration);
}
```

Създава нов запис в таблица devices и device_configuration. Свързва id-то на потребителя с id-то на устройството.

3.1.9.8 /client/{deviceId}/receive/gps-coordinates

```
@RequestMapping(value = "/client/{deviceId}/receive/gps-coordinates", method = GET)
@ResponseBody
public ResponseEntity<DataTransmitterInfo>
getGpsCoordinatesBydeviceId(@PathVariable(value = "deviceId") String deviceId) {
    try {
        DataTransmitterInfo dataTransmitterInfo =
        dataTransmitterHandler.getGPSCoordinates(deviceId);
        return new ResponseEntity(dataTransmitterInfo, HttpStatus.OK);
    }
    catch(InvalidDeviceIdException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
}
```

Целта на тази заявка е да вземе последния запис от таблицата data_transmitter за устройство с горепосоченото “deviceId”.

Тип: GET

Параметри:

String deviceId

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" – Изтекла или невалидна сесия.

message: "Invalid device id" – Невалидно устройство.

```
public DataTransmitterInfo getGPSCoordinates(String deviceId) {
    DataTransmitter dataTransmitter =
    dataTransmitterRepository.getGpsCoordinatesByDeviceId(deviceId);
    if(dataTransmitter == null) throw new InvalidDeviceIdException("Invalid device
id");
    return new DataTransmitterInfo(dataTransmitter);
}
```

Взима последния запис от таблицата data_transmitter за устройството с “deviceId” и го връща като обект от тип DataTransmitterInfo.

3.1.9.9 /client/{deviceId}/receive/gps-coordinates-for-day

```
@RequestMapping(value = "/client/{deviceId}/receive/gps-coordinates-for-day",
method = GET)
@ResponseBody
public ResponseEntity<List<DataTransmitterInfo>>
getGpsCoordinatesForTimeStamp(@PathVariable(value = "deviceId") String deviceId,
@RequestParam(value = "day") String day) {
    try {
        List<DataTransmitterInfo> dataTransmitterInfo =
dataTransmitterHandler.getGPSCoordinatesForDay(deviceId, day);
        return new ResponseEntity(dataTransmitterInfo, HttpStatus.OK);
    }
    catch(InvalidDeviceIdException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
}
```

Целта на тази заявка е да вземе позицията на устройството за всеки час в периода от 0-24 часа за посочената дата като праметър.

Тип: GET

Параметри:

String deviceId,

String day

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" – Изтекла или невалидна сесия.

message: "Invalid device id" – Невалидно устройство.

```
public List<DataTransmitterInfo> getGPSCoordinatesForDay(String deviceId, String day)
{
    DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
    String startFrom;
    String endFrom;
    List<DataTransmitterInfo> dataTransmitters = new ArrayList<>();
    for(int hour = 0; hour<24; hour++) {
        Date start_ = null;
        Date end_ = null;
        if(hour < 10) {
            startFrom = day + " 0" + hour + ":00:00";
            endFrom = day + " 0" + hour + ":59:59";
        }
        else {
            startFrom = day + "" + hour + ":00:00";
            endFrom = day + " " + hour + ":59:59";
        }
        try {
            start_ = formatter.parse(startFrom);
            end_ = formatter.parse(endFrom);
        } catch(java.text.ParseException exception) {}
        DataTransmitter dataTransmitter =
dataTransmitterRepository.getGpsCoordinatesForDay(deviceId, start_, end_);
        if(dataTransmitter != null) {
            dataTransmitters.add(new DataTransmitterInfo(dataTransmitter));
        }
    }
    if(dataTransmitters.size() == 0) throw new InvalidDeviceIdException("Invalid
device id");
    return dataTransmitters;
}
```

Проверява и взима за всеки един час от денонощието първите координати от таблицата data_transmitter за устройството с "deviceId", добавя ги в списък. След това списъкът от координати се връща като отговор на заявката.

3.1.9.10 /client/receive/user-account

```
@RequestMapping(value="/client/receive/user-account",method=GET)
@ResponseBody
public ResponseEntity getUserAccountByUsername(@RequestHeader("email") String
email) {
    try {
        UserInfo userInfo = userHandler.getUser(email);
        return new ResponseEntity(userInfo,HttpStatus.OK);
    }
    catch(InvalidEmailException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
}
```

Целта на тази заявка е да вземе запис от таблицата users, за който съвпада изпратения като хедър String email.

Тип: GET

Хедър:

String deviceId

Отговор от сървъра:

При неуспешна заявка:

message: "Invalid email" - Невалиден email.

```
public UserInfo getUser(String email) {
    User user = userRepository.getUserAccountByEmail(email);
    if(user == null) throw new InvalidEmailException("Invalid email");
    return new UserInfo(user);
}
```

Търси поле от таблицата users с подадения email като String, ако е намерен запис той се връща като обект от тип UserInfo

3.1.9.11 /client/{deviceId}/receive/device

```
@RequestMapping(value="/client/{deviceId}/receive/device",method=GET)
@ResponseBody
public ResponseEntity geDeviceByDeviceId(@PathVariable(value = "deviceId") String
deviceId) {
    try {
        DeviceInfo deviceInfo = deviceHandler.getDevice(deviceId);
        return new ResponseEntity(deviceInfo,HttpStatus.OK);
    }
    catch(InvalidDeviceIdException exception) {
        return new ResponseEntity(new ErrorDto(exception), HttpStatus.BAD_REQUEST);
    }
}
```

Целта на тази заявка е да вземе запис от таблицата devices, за който съвпада изпратения като path параметър String deviceId.

Тип: GET

Path параметър:

String deviceId

Отговор от сървъра:

При неуспешна заявка:

message: "Access Denied" – Изтекла или невалидна сесия.

message: "Invalid device id" – Невалидно устройство.

```
public DeviceInfo getDevice(String deviceId) {
    Device device = deviceRepository.getDeviceByDeviceId(deviceId);
    if(device == null) throw new InvalidDeviceIdException("Invalid device id");
    return new DeviceInfo(device);
}
```

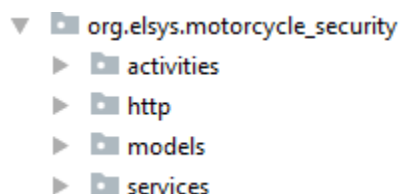
Търси поле от таблицата devices с подадения deviceId като String, ако е намерен запис той се връща като обект от тип DeviceInfo

3.2 Реализация на Android приложението

Android приложението е програмирано на Java, използвайки апито на Retrofit 2.0 в работната среда Android Studio.

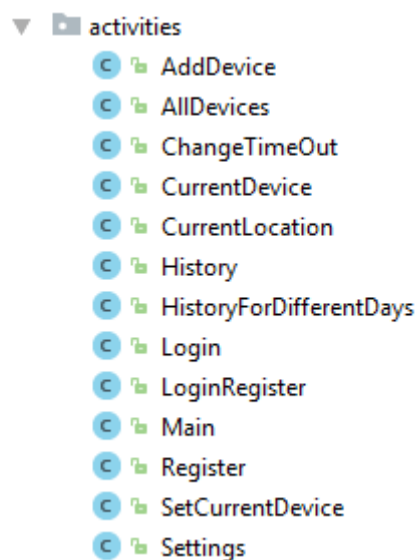
3.2.1 Разделение на кода

Проекта за реализиране на Android приложението е разделен на 4 пакета:



3.2.2 activities package

В този пакет се намират всички активитита на приложението. Всяко активити съдържа различна част от менюто на приложението.



`AddDevice` – Опция от менюто с настройки за добавяне на устройство към акаунта на потребителя.

`AllDevices` – Опция от менюто с настройки, която показва всички устройства, свързани с акаунта на потребителя.

`ChangeTimeOut` – Опция от менюто с настройки, чрез която потребителя може да промени времето, през което неговото устройство изпраща текущата си позиция на сървъра.

`CurrentDevice` – Опция от менюто с настройки, която да показва информация за устройството, което потребителя в момента използва.

`CurrentLocation` – Показва последното местоположение на всички устройства на потребителя, начертано на Google Maps карта.

`History` – Показва меню с избор на локацията на устройството през последните 5 дни.

`HistoryForDifferentDays` - Показва местоположението на устройството на потребителя за избраната дата в период през 1 час, начертано на Google Maps карта.

Login – Мястото, където потребителя попълва email-а и паролата си за да се логне в приложението.

LoginRegister – Това е менюто, на което потребителя избира дали ще се логне в акаунта си или да се регистрира.

Main – Това е главното меню на приложението.

Register – Мястото, където потребителя попълва номера на устройството си, email-а си и паролата си за да се регистрира.

SetCurrentDevice – Опция от менюто с настройки, чрез която потребителя избира (Ако притежава повече от 1 устройство) кое от неговите устройства приложението да управлява.

Settings – Това е менюто с настройки.

3.2.3 http package

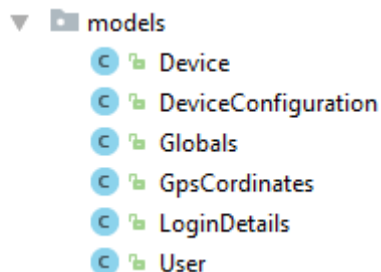
В този пакет се намира настройката на Retrofit апито.



В Api класа е описано IP-то и порта на сървъра, също така и връзките към всички заявки.

3.2.4 models package

В този пакет се съдържат моделите на таблиците от базата данни, но само с информацията, която достига до Android приложението чрез сървъра.



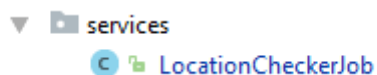
Моделите Device, DeviceConfiguration, GpsCordinates и User съдържат само информацията, която е нужна на Android приложението.

Класът Globals съдържа някои глобални променливи, които се използват само докато приложението работи.

Класът LoginDetails е модел за системата за сесии и съдържа само полетата email и password, за разлика от класът User, който съдържа и id, и списък от устройства.

3.2.5 services package

В този пакет се съдържат сървисите на проекта.



LocationCheckerJob е сървис, който работи само когато устройството на потребителя е в режим „паркиран“. Когато потребителя „паркира“ устройството си, този сървис се включва и започва постоянно да проверява позицията на устройството, като веднага при промяна на координатите, сървисът уведомява потребителя, че неговият мотоциклет или автомобил се движи. Този сървис работи дори когато приложението не е активно.

3.3 Реализация на Arduino устройството

Arduino устройството е програмирано на C/C++, използвайки библиотеката DFRobot_SIM808 в работната среда Arduino IDE.

3.3.1 Свързване

SIM808 модулет е свързан към RobotDyn Mega2560 през портовете 10 (TX) и 11 (RX). И двете устройства се захранват от външна батерия с изходен ток 2.1A. Външната батерия е свързана към акумулатора на мотоциклета чрез USB power адаптер.

3.3.2 Програмиране

Основната работа на устройството е да отчита текущата позиция на устройството и да изпраща координатите на сървъра. То изпраща текущата си позиция през определения за него `timeOut` в мили секунди, като през определено време прави заявка към сървъра, чрез която проверява дали няма промяна в стойността на `timeOut`. Устройството е програмирано, така че да хаби възможно най-малко количество ток за да работи повече време външната батерия, към която е свързано. Затова между заявките устройството е в режим `sleep`, в който пести енергия.

Устройството е програмирано със следния алгоритъм:

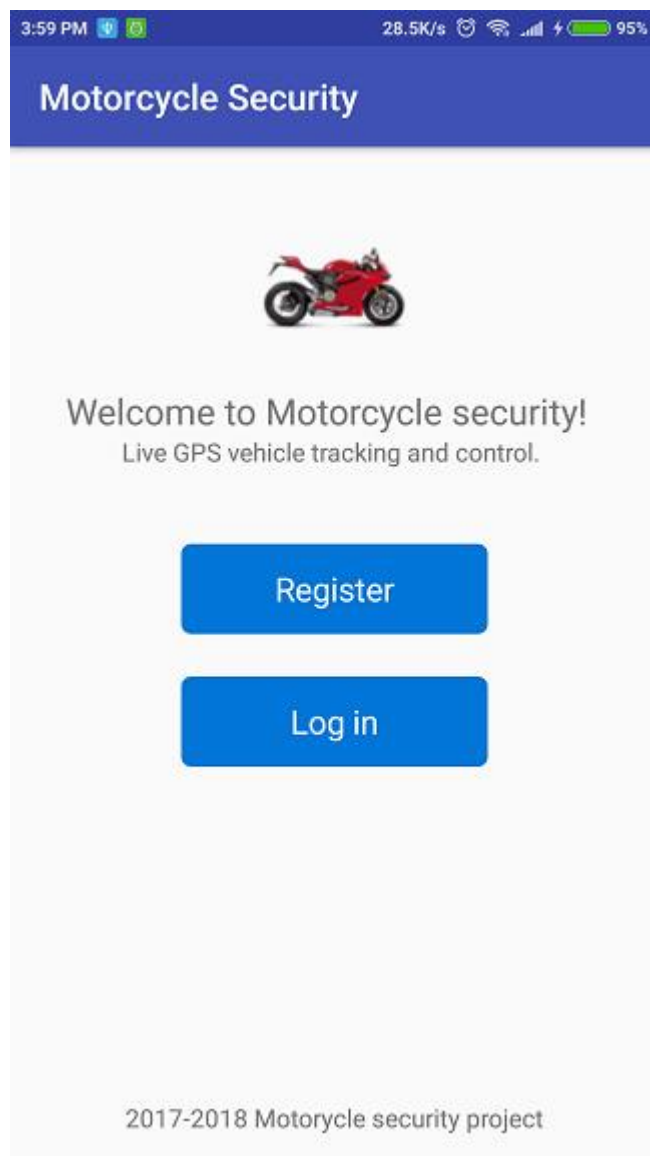
```
void setup()
{
    mySerial.begin(9600);
    Serial.begin(9600);
    initializeConnection();
    initializeTCPConnection();
    formattedGetConfigurationString =
formatGetConfigurationString(deviceId);
    getConfiguration(formattedGetConfigurationString);
}
```

При включване на устройството, то се свързва към сървъра и приема своята конфигурация. След това, то влиза в безкраен цикъл и периодически изпраща координати към сървъра, като проверява дали има промяна в конфигурацията на устройството през определен брой заявки.

Четвърта глава

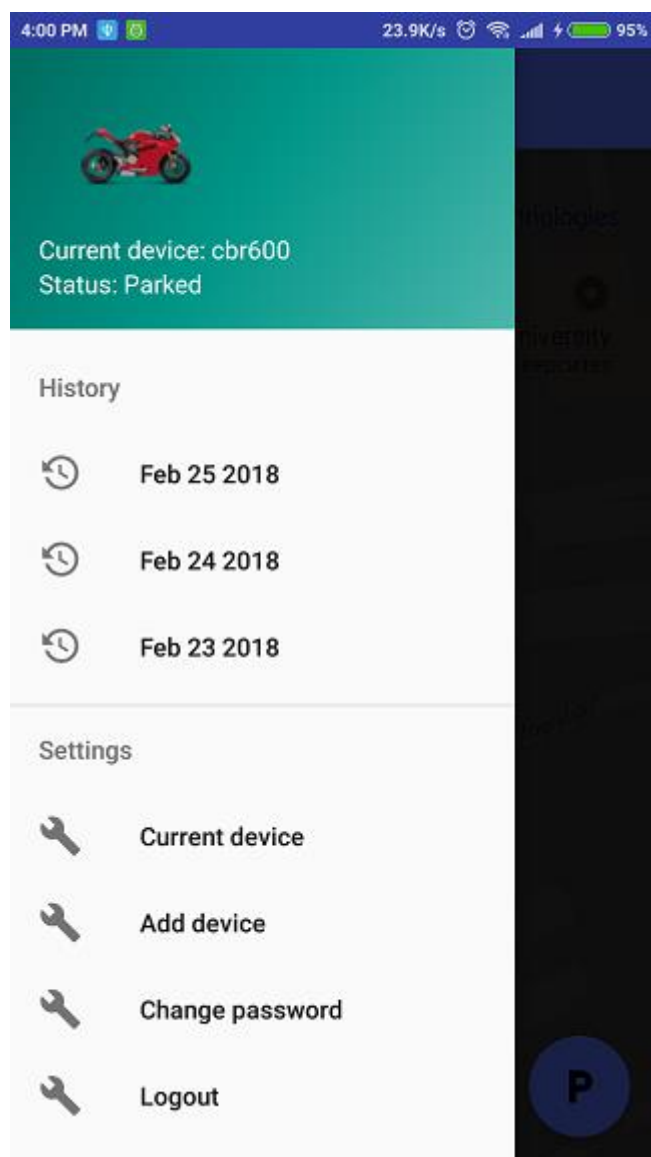
4.1 Наръчник за работа с приложението

4.1.1 Първоначално включване на приложението



Потребителя избира дали да се регистрира или да влезе в акаунта си.

4.1.2 Главно меню



“History” Показва списък с послените 3 календарни дни за избор на показване на локацията на устройството за тези дни в разстояние от 1 час.

“Settings” Показва списък за настройка на приложението.

4.1.3 Местоположение на устройството.

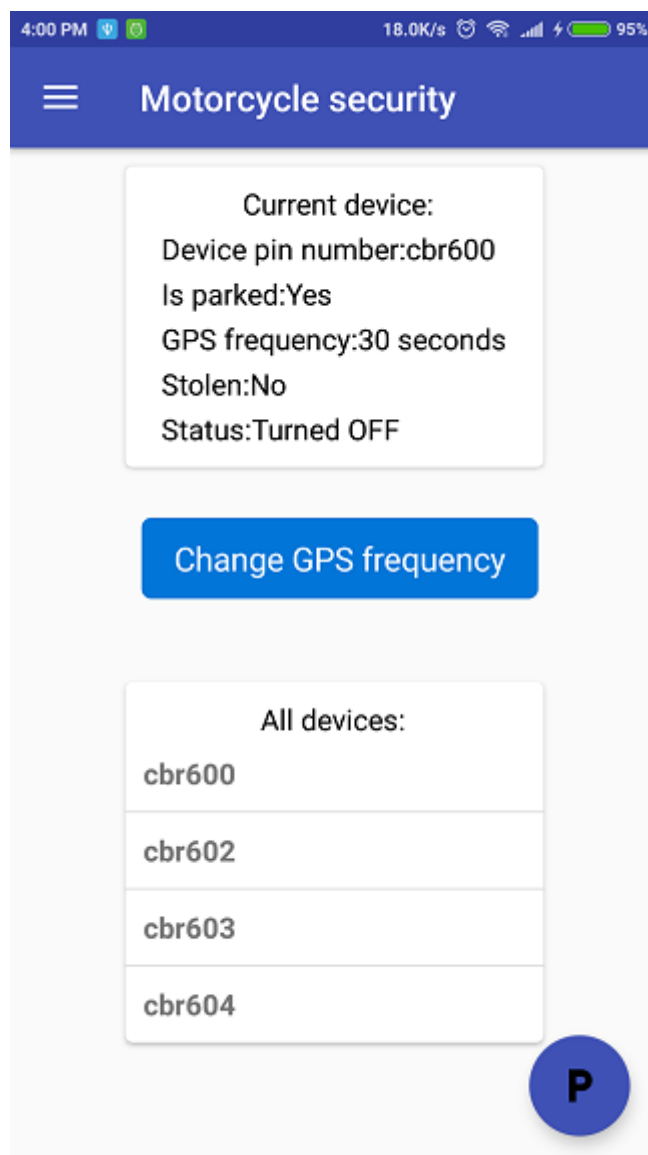


Показва местоположението на нашето устройство: cbr600.

4.1.4 Park (Кръглия Р бутон в долния десен ъгъл)

Поставя устройството в режим „паркиран“ или „непаркиран“. Когато устройството е в режим „паркиран“ фонът на бутона Р е син, когато то е в режим „непаркиран“ фонът на бутона е червен.

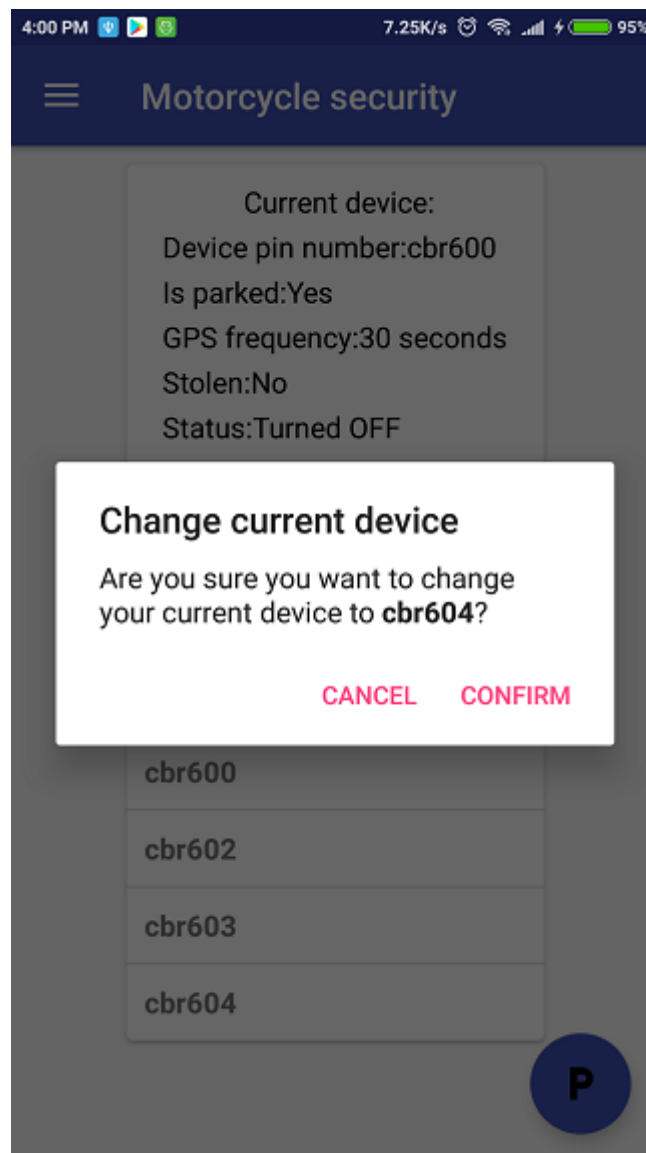
4.1.5 Информация за устройството и всички устройства



Показва информация за устройството на потребителя. С бутона „Change GPS frequency” може да бъде променено времето, през което устройството изпраща координати до сървъра.

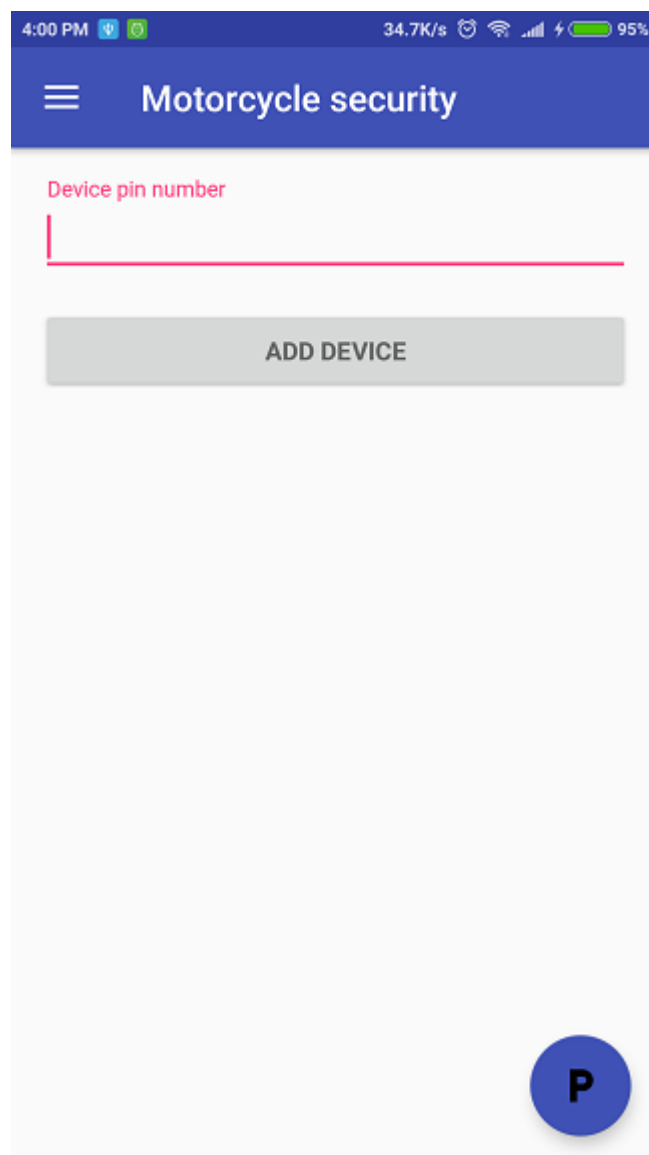
Прозореца „All devices” показва всички устройства на потребителя и предоставя възможност за промяна на устройството, което приложението следи и управлява.

4.1.6 Промяна на устройство



Чрез натискане на едно от устройствата в списъка в “Current device” опцията на приложението, потребителят може да промени кое устройство да се проследява от приложението.

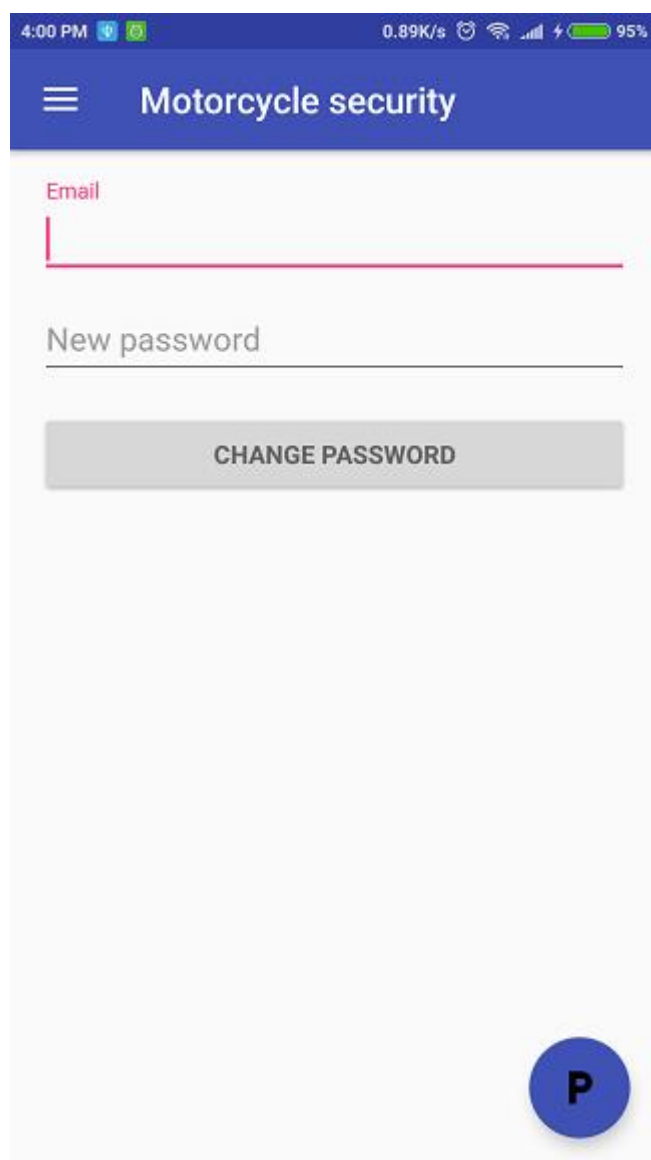
4.1.7 Добавяне на устройство



The screenshot shows a mobile application interface titled "Motorcycle security". At the top, there is a status bar with the time "4:00 PM", signal strength, and battery level at "95%". Below the status bar is a blue header with a hamburger menu icon and the text "Motorcycle security". The main content area is white and contains a red label "Device pin number" above a red input field. Below the input field is a grey button labeled "ADD DEVICE". In the bottom right corner, there is a blue circular profile picture placeholder with a white letter "P".

Добавяне на устройство към акаунта на потребителя. Потребителят трябва да въведе валиден номер на устройство, което се проверява от сървъра дали съществува.

4.1.8 Промяна на парола



The screenshot shows a mobile application interface titled "Motorcycle security". At the top, there is a status bar with the time "4:00 PM", signal strength, and battery level "95%". Below the status bar is a blue header with a hamburger menu icon and the text "Motorcycle security". The main content area is white and contains two input fields: "Email" with a red underline and "New password" with a grey underline. Below these fields is a grey button labeled "CHANGE PASSWORD". In the bottom right corner, there is a blue circular icon with a white letter "P".

Промяна на паролата на акаунта на потребителя. Потребителят трябва да въведе email адреса на акаунта си за да промени паролата си.

Заклучение

В днешно време, когато задръстванията в големите градове и невъзможността да бъде намерено парко място са част от ежедневието ни, все повече млади хора избират транспорта на две колела. По-голямата мобилност, свобода, страст или начин на живот са мотивите на мотористите. От друга страна техните превозни средства са най-уязвими. С развиването на производството на мотоциклетите и мото феновете се развива и престъпността. Крадците на моторни превозни средства стават все по-дързки. Повишената стойност и възможността за кражба на моторите привличат все повече крадци. Затова с развитието на мото-индустрията се развиват и устройства, подобни на Motorcycle security.

Успешно бе създадено устройство за защита и проследяване на мотоциклети, което може да донесе спокойствие на собственика на мотоциклет и да допринесе за безпроблемната и истинска емоция от приключенията на 2 колела. Устройството може да бъде монтирано на мотоциклет бързо и лесно. Не се изискват допълнителни познания за да се инсталира устройството, както и приложението за телефон, което е с улеснен за потребителя интерфейс, за да допринесе за бърза и безпроблемна работа с този продукт.

Устройството може да бъде подоброено чрез по-добър GPS модул, също така и чрез имплементирането на още функции към него, като например автоматично подаване на сигнал на 112 в случай на катастрофа или кражба.

Използвана литература

1. Car alarm - https://en.wikipedia.org/wiki/Car_alarm
2. Oxford Tracker Spy - https://www.sportsbikeshop.co.uk/motorcycle_parts/content_prod/261078
3. Autocom GPS Tracker - https://www.sportsbikeshop.co.uk/motorcycle_parts/content_prod/246089
4. Car alarm - https://en.wikipedia.org/wiki/Car_alarm
5. Arduino Mega 2560 - http://www.geeetech.com/wiki/index.php/Arduino_Mega_2560
6. SIM808 - https://www.itead.cc/wiki/SIM808_GSM/GPRS/GPS_Module
7. IntelliJ IDEA - https://en.wikipedia.org/wiki/IntelliJ_IDEA
8. Android Studio - https://en.wikipedia.org/wiki/Android_Studio
9. Arduino IDE - <https://en.wikipedia.org/wiki/Arduino>
10. MySQL - <https://en.wikipedia.org/wiki/MySQL>
11. Spring Framework - https://en.wikipedia.org/wiki/Spring_Framework
12. Java - [https://en.wikipedia.org/wiki/Java_\(programming_language\)](https://en.wikipedia.org/wiki/Java_(programming_language))
13. C - [https://en.wikipedia.org/wiki/C_\(programming_language\)](https://en.wikipedia.org/wiki/C_(programming_language))
14. Spring Framework docs - <https://docs.spring.io/spring/docs/current/spring-framework-reference/>
15. Android Studio developer - <https://developer.android.com/reference/>
16. Retrofit - <http://square.github.io/retrofit/>

Github repository

Link - <https://github.com/busternr/motorcycle-security>

Съдържание

Задание.....	2
Увод.....	4
Първа глава.....	5
1.1 Преглед на устройства за проследяване и защита на автомобили.....	5
1.1.1 Oxford Tracker Spy.....	5
1.1.2 Autocom GPS Tracker.....	7
Втора глава.....	8
2.1 Избор на хардуерна архитектура.....	8
2.1.1 RobotDyn Mega 2560.....	8
2.1.2 SIM808.....	13
2.2 Програми, технологии и програмни езици.....	17
2.2.1 Програмни езици.....	17
2.2.2 Технологии.....	19
2.2.3 Програми.....	20
2.3 Изисквания към програмния продукт.....	22
2.3.1 Изисквания към Android приложението.....	22
2.3.2 Изисквания към Arduino устройството.....	22
2.3.3 Изисквания към свързване на устройството.....	22
Трета глава.....	23
3.1 Реализация на сървъра.....	23
3.1.1 Разделение на кода.....	23
3.1.2 business.logic package.....	23
3.1.3 controllers package.....	24
3.1.4 dto package.....	24
3.1.5 models package.....	25
3.1.6 repository package.....	25
3.1.7 security package.....	25
3.1.8 device заявки.....	26
3.1.8.1 /device/send/gps-coordinates.....	26
3.1.8.2 /device/{deviceid}/receive/device-configuration.....	27

3.1.9 client заявки.....	28
3.1.9.1 /client/send/parking-status.....	28
3.1.9.2 /client/send/timeout.....	29
3.1.9.3 /client/send/parked-coordinates.....	30
3.1.9.4 /client/send/stolen-status.....	31
3.1.9.5 /client/send/change-password.....	32
3.1.9.6 /client/send/create-new-user.....	33
3.1.9.7 /client/send/create-new-device.....	34
3.1.9.8 /client/{deviceid}/receive/gps-coordinates.....	35
3.1.9.9 /client/{deviceid}/receive/gps-coordinates-for-day.....	36
3.1.9.10 /client/receive/user-account.....	37
3.1.9.11 /client/{deviceid}/receive/device.....	38
3.2 Реализация на Android приложението.....	39
3.2.1 Разделение на кода.....	39
3.2.2 activities package.....	39
3.2.3 http package.....	40
3.2.4 models package.....	40
3.2.5 services package.....	41
3.3 Реализация на Arduino устройството.....	42
3.3.1 Свързване.....	42
3.3.2 Програмиране.....	42
Четвърта глава.....	43
4.1 Наръчник за работа с приложението.....	43
4.1.1 Първоначално включване на приложението.....	43
4.1.2 Главно меню.....	44
4.1.3 Местоположение на устройството.....	45
4.1.4 Park.....	45
4.1.5 Информация за устройството и всички устройства.....	46
4.1.6 Промяна на устройство.....	47
4.1.7 Добавяне на устройство.....	48
4.1.8 Промяна на парола.....	49
Заклучение.....	50
Използвана литература.....	51
Github repository.....	51