**ТЕХНОЛОГИЧНО УЧИЛИЩЕ “ЕЛЕКТРОННИ СИСТЕМИ”**

**КЪМ ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

Тема: **Ардуино базирана система за запис на полетни параметри на малки летателни апарати**

Дипломант: Научен ръководител:

*Ангел Миланов гл. ас. д-р инж. Владимир Сербезов*

СОФИЯ

2020

Увод

В днешно време почти всяко дете мечтае да се сдобие с радиоуправляема електронна играчка - кола, самолет, хеликоптер, дрон. Не е тайна, че и много възрастни се интересуват от изграждането и управлението на тези машини, превръщайки мечтите от детството си в хоби. Напредъкът в технологиите позволява усъвършенстването на подобни модели.

Тази дипломна работа има за цел системното модифициране на радиоуправляем самолет, чрез използването на различен набор от цифрови и аналогови сензори, разпространени на пазара. Системата ще следи показателите на електрическите двигатели (обороти в секунда, захранващо напрежение и ток), географското положение, позиция на самолета във въздуха, както и параметрите на електрическото захранване. Всички събрани данни ще се изпращат на сървър, който ще позволи на потребителско приложение да ги обработва и визуализира.

**Съдържание**

[Глава 1. Въведение 5](#_Toc35886730)

[1.1 Увод 5](#_Toc35886731)

[1.2 Общи сведения 5](#_Toc35886732)

[1.2.1 Наподобяващи реализации 5](#_Toc35886733)

[1.2.2 Авиомоделизъм 6](#_Toc35886734)

[1.3 Методи на измерване 7](#_Toc35886735)

[1.3.1 GPS 7](#_Toc35886736)

[1.3.2 Акселерометър 8](#_Toc35886737)

[1.3.3 Жироскоп 8](#_Toc35886738)

[1.3.4 Температура 9](#_Toc35886739)

[1.3.5 Налягане 10](#_Toc35886740)

[1.3.6 Въздушна скорост 12](#_Toc35886741)

[1.3.7 Честота на въртене 14](#_Toc35886742)

[1.3.8 Ефект на Хол 15](#_Toc35886743)

[1.4 Методи на комуникация 16](#_Toc35886744)

[1.4.1 Bluetooth 16](#_Toc35886745)

[1.4.2 WiFi 16](#_Toc35886746)

[1.4.3 LoRa 17](#_Toc35886747)

[1.4.4 ZigBee 17](#_Toc35886748)

[1.5 Микроконтролерни развойни платки 18](#_Toc35886749)

[1.5.1 Ардуино 18](#_Toc35886750)

[1.5.2 Raspberry pi 20](#_Toc35886751)

[1.6 Бази данни 21](#_Toc35886752)

[1.6.1 Релационни бази данни (SQL) 21](#_Toc35886753)

[1.6.1 Нерелационни бази данни (NoSQL) 22](#_Toc35886754)

[1.7 Езици за програмиране на Android 25](#_Toc35886755)

[Глава 2. Проектиране 27](#_Toc35886756)

[2.1 Увод 27](#_Toc35886757)

[2.2 Хардуер 27](#_Toc35886758)

[2.2.1 Блокова схема на хардуера 27](#_Toc35886759)

[2.2.2 Описание на блоковете на хардуерната блок схема 27](#_Toc35886760)

[2.3 Софтуер 37](#_Toc35886761)

[2.3.1 Блокова схема на управляващия софтуер 37](#_Toc35886762)

[2.3.2 Блокова схема на софтуер на приложението 41](#_Toc35886763)

[2.3.3 База данни 43](#_Toc35886764)

[Глава 3. Реализация 45](#_Toc35886765)

[3.1 Увод 45](#_Toc35886766)

[3.2 Хардуер 45](#_Toc35886767)

[3.2.1 Принципна електрическа схема 45](#_Toc35886768)

[3.3 Софтуер 49](#_Toc35886769)

[3.3.1 Управляващ софтуер 49](#_Toc35886770)

[Глава 4. Резултати 57](#_Toc35886771)

[4.1 Увод 57](#_Toc35886772)

[4.2 Андроид приложение 57](#_Toc35886773)

[Регистрация и вход 57](#_Toc35886774)

[Меню 58](#_Toc35886775)

[История и обработка на данни 59](#_Toc35886776)

[Профил на потребителя 60](#_Toc35886777)

[4.3 Експериментална постановка 61](#_Toc35886778)

[4.4 Експериментални данни 63](#_Toc35886779)

[Глава 5. Заключение 67](#_Toc35886780)

[Източници 68](#_Toc35886781)

# Глава 1. Въведение

## 1.1 Увод

В тази глава са описани използваните материали за изготвяне на дипломната работа и са проучени различни методи за измерване на нужните параметри изискани от дипломната работа за безжично изпращане на информация, нейното обработване и визуализиране. В заключение са представени избраните след проучването методи, които ще бъдат използвани за целите на дипломната работа.

## 1.2 Общи сведения

### 1.2.1 Наподобяващи реализации

**Черна кутия**

Черната кутия е бордово устройство, което извършва регистрация, фиксиране, запис и запазване на цялата полетна информация. Освен това в черната кутия има и магнетофон, който записва речта и преговорите на екипажа на самолета. Тя има титанов брониран корпус и специална топлоизолация, която предпазва от повреда съдържанието ѝ при високи температури. За да се защитят данни по време на катастрофа, кухите части на черната кутия се пълнят със специален прах, който издържа на температурата на горене на реактивно гориво. В противоречие с името, тя не е черна, а яркооранжева, за да бъде по-лесно намирана при злополуки. Съдържанието на черната кутия може да бъде изхвърлено по време на вертикален удар със земята.

Черните кутии записват много параметри, сред които са:

* технически: хидравлично налягане, скорост на двигателя, налягане на горивото, температура и др.;
* действия на членовете на екипажа: разширяване и почистване на механизмите за излитане и кацане, отхвърляне на ръководните органи;
* навигационни данни: височина на полета, скорост, преминаване на фарове и др.

### 1.2.2 Авиомоделизъм

Авиомоделизмът е вид спорт, който може да се практикува и като хоби независимо от възрастта. Необходими са техническо творчество и умения. Представлява създаване по чертеж и/или пилотиране на модели на летателни апарати – дистанционно управляеми (радиоуправляеми, кордови), свободно летящи-безмоторни самолети (планери), както и модели на хеликоптери. За малко повече от 60-годишната си история авиомоделизмът в България е един от най-стремително развиващите се технически спортове. Авиомоделите в никакъв случай не са играчки, а спортистите-моделисти не са просто пораснали деца. Радиоуправляемите модели на самолети са сложни и технологични машини, съчетание от различни компоненти на механиката и радиоелектрониката. Дисциплината и точността са отличителни черти за добрите авиомоделисти и пилоти. Всъщност авиомоделизмът е сложно съчетание между спорт и изкуство, защото не само самостоятелната изработка на летящ модел, но и самата му настройка, както и управление, изискват тънко владеене на тези две на пръв поглед разнопосочни музи.

За изработването на моделите се използват тънък шперплат, леко дърво балса, смоли с ламинатни технологии, стъклопласт, пенополистироли, стиропор, депрон, дунапрен. Статичните модели се изработват от пластмаса, метал, дърво, дори от хартия. Техниките за построяването на различните модели са принципно различни. Размерите им варират от няколко сантиметра до няколко метра. Моторните модели се задвижват с миниатюрни ДВГ, електродвигатели, гумени двигатели, а в последните години и с моделни турбореактивни двигатели. В миналото (до 80-те) авиомоделите се изработват основно от балса, шперплат, бамбук. Основните лепила са декстриново и ацетоново. А за обличане на модела се използва много тънка хартия (ЯПОН), която се залепя с ацетоновото лепило, а се опъва с разредено такова. По-късно с развитието на химическата промишленост и излизането на структурните лепила започват да се използват полиестерни и епоксидни смоли в комбинация със стъклен плат. В днешни дни рядко се прибягва до балса и шперплат, а основно се работи със стиропор, карбон, кевлар, техните смеси и стъклен плат. Използват се и комбинации от балса и карбон за изработка на комбиниран шперплат. За обличане се предпочита специално моделарско фолио (термосвиваемо, лазурно оцветено или непрозрачно оцветено) с подготвен лепилен слой за топло лепене (опъване).

## 1.3 Методи на измерване

### 1.3.1 GPS

GPS е система, използваща 33 спътника, разположени в 6 средни околоземни орбити с височина 20 180 km, за определяне на местоположение и скорост, като за целта се измерва разстоянието между обекта и спътниците на системата, които са с точно определено и известно местоположение. Измерването на това разстояние става на база на времето за разпространение на сигнала от спътник до обекта. Всеки спътник има много стабилен атомен часовник, и непрекъснато изпраща своето текущо време и местоположение. Измерването на това разстояние става на база на времето на разпространение на сигнала от спътника до обекта. Всеки спътник има много стабилен атомен часовник, който е синхронизиран с останалите спътници и с наземните часовници, непрекъснато изпраща своето местоположение и текущо време. Устройствата използващи GPS изчисляват разликата между времето на изпращане на пакети от сеателитите и текущото време, като така получават времето на разпространение на сигнала. С времето на разпространение и скоростта на разпространение се пресмята изминатото разстояние. За да може да определи местоположението си, приемникът трябва да има връзка с поне 4 сателита, тъй като трябва да се изчислят 4 неизвестни параметъра - 3 за кординатите X,Y,Z (използвайки трилатерация) и 1 за разминаването между локалното време и това между това на сателитите. Когато разстоянието изминато между две последователни измервания е значително по-голямо от грешката в определянето на позицията, може да се пресметне скоростта с която е изминато разстоянието, на базата на времето между двете измервания и разстоянието от едната позиция до другата.

Основният недостатък на този метод е, че скоростта може да се пресметне само след изминаването на разстоянието между двете точки, които трябва да са достатъчно отдалечени една от друга заради грешката в измерването. Някои устройства използват доплеров ефект за точно измерване на скоростта си, спрямо сателитите. С GPS може да се постигне грешка на измерването на позицията под 5m и около 0.2 m/s за измерената скорост от устройствата използващи доплеров ефект.

### 1.3.2 Акселерометър

Измерва увеличаването и намаляването на скоростта на движение на тела съответно с означение като положително и отрицателно ускорение при ползвана за него мерна единица земното ускорение g. В най-често използваните триосни акселерометри (3-axis accelerometer) ускорението се измерва по три взаимно перпендикулярни оси с еднакви или различаващи се параметри по тях (в зависимост от модела). По-ограничено е приложението на акселерометрите за една и две оси. Параметърът обхват логично означава максималното измервано ускорение, като той може да има няколко стойности, избирани чрез управляващия интерфейс. Чувствителността (Sensitivity) S показва минималното ускорение, което може да бъде измерено и в зависимост от модела тя е еднаква или различна по осите. При цифровите акселерометри измерението й е mg/LSB и означава стойността на ускорението за промяна на изходното число с 1, но се използва и мерна единица LSB/g, която е стойността на изходното число при ускорение 1g. Характерни приложения на акселерометрите са в автомобилите, в дронове и самолети, за предпазване на прибори от удари (напр. HDD Shock Protection), мобилни телефони, измервателни прибори и др. За разширяване на приложенията съществуват триосни акселерометри с вграден програмируем микроконтролер. В някои акселерометри са прибавени възможности за използване при тренировки – броене на крачки с разпознаване на вида им (ходене, тичане), както и на неподвижен стоеж при токова консумация в този режим под 8 mA. За работа при наличие на големи вибрации се предлагат акселерометри с вграден филтър.

### 1.3.3 Жироскоп

Представлява сензор за ъглова скорост, като интегралните са практически изцяло MEMS жироскопи. Основното им предимство е използването на вибриращ механичен елемент вместо въртяща се част. Предназначени са за стабилизиране на положението на физически обекти чрез измерване на тяхната ъглова скорост WAV при завъртането им (обикновено нежелателно). Характерни приложения са за роботи, самолети и плавателни съдове, за стабилизиране на изображения в камери, както и на положението на подвижни платформи и на поставяни на главата или в дрехите портативни прибори, в електронни игри, часовници (Smart Watch), очила (Augmented Reality Glass) и принадлежности за спорт и фитнес, в дистанционни управления, за движение на роботи по определен маршрут в помещения. В автомобилите се използват за управление на електронните стабилизиращи системи (ESC).

Обикновено ъгловата скорост се определя в две или три взаимно перпендикулярни направления и рядко в едно, а поради спецификата на приложенията се измерва в °/s, което означава на колко градуса се завърта жироскопът за 1s. Приетите за положителна и отрицателна посоки на завъртане по всяка от осите се отбелязват в техническата документация.

### 1.3.4 Температура

Температурата е физична величина, характеризираща средната кинетична енергия на частиците от дадена макроскопична система, намираща се в състояние на термодинамично равновесие. Тя е свързана също със субективните усещания за топло и студено, а количествено се измерва с термометри, които могат да бъдат калибрирани да показват температурата в различни температурни скали. Температурата е физично свойство на материята, което количествено изразява общите понятия за горещо и студено.

Температурният датчик открива температурата на даден обект или неговата среда и преобразува показанието в електрически сигнал. Често срещаните типове температурни сензори включват термодвойки, детектори за устойчивост на температура (RTDs), термистори, локални ИЧ сензори за температура и дистанционни IC сензори за термичен диод. Термодвойките, RTD и термисторите са чувствителни елементи с електрически свойства, които предвидимо варират в зависимост от температурата. ИК на локалния температурен датчик използват физическите свойства на транзисторите, за да умрат като чувствителен елемент. Температурните сензори за клинична степен отговарят на спецификацията за клинична термометрия ASTM E1112 за точност. Дистанционните сензори за термични диоди използват външен биполярен транзистор като чувствителен елемент и включват цялата схема за кондициониране на сигнала, необходима за измерване на температурата с помощта на един или повече външни транзистори.

### 1.3.5 Налягане

Налягането е величина, характеризираща големината на натиска, действащ перпедикулярно върху единица площ.

**Видове налягане:**

**Абсолютно налягане** - пълното натисково напрежение в дадена точка, породено от действието на всички външни сили, включително и тези от атмосферното налягане.

**Атмосферно налягане** (барометрично) – налягането, създадено от тежестта на въздушния слой около земята. То действа върху всяка точка на земното кълбо. За нормално атмосферно налягане е прието налягането на височина на морското равнище и температура 20°C.

**Относително налягане** (манометрично) - разликата между абсолютното и атмосферното налягане.

**Хидростатично налягане** - налягане във вътрешността на течностите (дължи се на теглото на разположените отгоре слоеве и показанието му зависи от височината и плътноста на течноста).

**Диферинциално налягане** - разликата на наляганията между две измервани точки.

Съществуват две скали за измерване на налягане (абсолютна и относителна). Абсолютната скала показва действително налягане и използва за „нула” идеалния вакуум, при нея липсват отрицателни стойности. Измереното налягане се нарича абсолютно или действително. Използва се най често в физиката а в последно време и в техниката Относителната скала има за „нула” атмосферното налягане, което съществува при ниво на морското равнище и температура на въздуха „20” градуса по Целзий. Налягането което се измерва по тази скала се нарича още манометрично,ефективно, представлява разликата между абсолютното и атмосферното налягане. При тази скала показанията от нула (атмосферно налягане) до абсолютната нула (пълен вакуум) са отрицателни. Използва се навсякъде в бита и в техниката се работи предимно с него.

**Методи за измерване на налягане**

Налягането характеризира явленията, свързани с флуидите. То играе важна роля при управлението на много технологични процеси. Разнообразието от диапазони на изменение на налягането (от абсолютен вакуум до свръх високи налягания) е наложило много методи и средства за измерването му.

**Хидростатични методи** - при апаратите, базиращи се на тези методи, измерваното налягане се уравновесява от налягането, създавано от стълб течност с известна плътност и височина, пропорционална на налягането. Върху тези методи се основават живачните барометри, U - образните манометри, вакуумметри и др.

**Методи, основаващи се на деформацията**, която налягането предизвиква върху специални еластични преобразуватели. Еластичната сила от деформацията уравновесява силата, създавана от измерваното налягане. Като еластични елементи се използват: тръбни пружини, мембрани, цилиндрични пружини и др. Най- голямо приложение са намерили манометрите с еластичен преобразувател.

**Методи, основаващи се на механично уравновесяване на налягането**, действащо върху определена площ на подвижния му елемент. Към уредите, основаващи се на тези методи, спадат буталните манометри с механично уравновесяване. Буталните манометри се използуват като образцови уреди и притежават висок клас на точност.

**Електрически методи** - измерването се основава на зависимостта между налягането и електрическите параметри на преобразувателния елемент. Към уредите, основаващи се на тези методи, спадат пиезоелектрическите манометри, магнитоеластичните манометри, йонизационните манометри и други получили общото наименование трансмитери за налягане.

### 1.3.6 Въздушна скорост

Скоростта е векторна физична величина в кинематиката, която показва колко бързо се променя пространственото положение на една материална точка с времето относно избрана отправна система. Като векторна величина скоростта има не само големина, но и посока. Тя е първата производна на радиус-вектора по времето. Промяната на скоростта с времето или втората производна на радиус-вектора по времето се нарича ускорение.

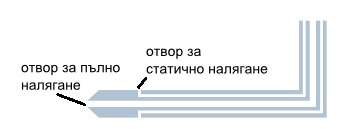
Въздушна скорост е скоростта на самолета спрямо въздуха. Сред общите конвенции за квалифицирана въздушна скорост са посочени въздушната скорост ("IAS"), калибрираната скорост на въздуха ("CAS"), еквивалентната скорост на въздуха ("EAS"), истинската скорост на въздуха ("TAS") и плътността на въздушната скорост. Посочената скорост на въздуха е просто това, което се отчита от измерване на въздушна скорост, свързано към статична система на пито, калибрираната скорост на въздуха е обозначена с въздушна скорост, регулирана за положението и грешката в инсталацията на системата на питот, а еквивалентната скорост на въздуха се калибрира с въздушна скорост, коригирана за ефекти на сгъваемост. Истинската скорост на въздуха е еквивалентна скорост на въздуха, регулирана за плътността на въздуха, и е също така скоростта на въздухоплавателното средство през въздуха, в който лети. Калибрираната скорост на въздуха обикновено е в рамките на няколко възела от посочената скорост на въздуха, докато еквивалентната скорост на въздуха намалява леко от CAS с увеличаване на височината на въздухоплавателното средство или с висока скорост. При постоянна EAS, истинската скорост на въздуха се увеличава с увеличаване на височината на самолета. Това е така, защото плътността на въздуха намалява с по-голяма надморска височина, но крилото на въздухоплавателното средство изисква същото количество въздушни частици (т.е. масата на въздуха), преминаваща около него, за да произведе същото количество асансьор за даден ъгъл на атака. По този начин крилото трябва да се движи по-бързо през по-тънък въздух от по-дебелия въздух, за да получи същото количество повдигане.

Измерването и индикацията на скоростта на въздуха обикновено се извършва на борда на въздухоплавателното средство чрез индикатор за скорост на въздуха ("ASI"), свързан към пито-статична система. Питото-статичната система *(фиг. 1.1)* включва една или повече питотни сонди (или тръби), насочени към идващия въздушен поток за измерване на питото налягане (наричано също стагнация) и един или повече статични отвори за измерване на статичното налягане във въздухa. Тъй като тази тръба съдържа течност, може да се измери налягане. Подвижната течност се привежда в покой (застояла), тъй като няма изход, който да позволи на потока да продължи. Това налягане е стагнационното налягане на флуида, известно още като общото налягане (особено в авиацията). Измереното налягане на застоя не може само да се използва за определяне на скоростта на потока на течността (скорост на въздуха в авиацията). Уравнението на Бернули обаче гласи:

Стагнационно налягане = статично налягане + динамично налягане

* u - скоростта на потока;
* pt - застоя или общото налягане;
* ps - статичното налягане;
* ρ – плътността на течността.

Динамичното налягане е разликата между стагнационното налягане и статичното налягане. След това динамичното налягане се определя с помощта на диафрагма вътре в затворен контейнер. Ако въздухът от едната страна на диафрагмата е при статично налягане, а от другата - при стагнационно налягане, тогава отклонението на диафрагмата е пропорционално на динамичното налягане. При въздухоплавателните средства статичното налягане обикновено се измерва с помощта на статичните портове отстрани на фюзелажа. Измереното динамично налягане може да се използва за определяне на посочената скорост на въздуха на въздухоплавателното средство. Диафрагмената схема, описана по-горе, обикновено се съдържа в индикатора за скорост на въздуха, който превръща динамичното налягане в отчитане на скоростта на въздуха с помощта на механични лостове.

****

*фиг. 1.1 Схема на питотна сонда*

### 1.3.7 Честота на въртене

Мерната единица за скорост на въртене или честотата на въртене около фиксирана ос е обороти в минута. Знакът за честотата на въртене трябва да бъде f и ω или Ω за ъглова скорост. Съответната основна SI производна единица е s− 1 или Hz. При измерване на ъглова скорост се използва единица радиан в секунда.

Формално херц (Hz) и радиан в секунда (rad/s) са две различни имена за една и съща единица SI, s− 1. Те обаче се използват за две различни, но пропорционални ISQ количества: честота и ъглова честота (ъглова скорост, величина на ъгловата скорост). Преобразуването между честота f (измерена в херц) и ъглова скорост ω (измерена в радиани в секунда) е:

По този начин се показва, че диск, въртящ се с 60 оборота в минута, се върти при 2π rad/s или 1 Hz, където първият измерва ъгловата скорост, а вторият отразява броя на оборотите в секунда. Ако не-SI единицата rpm се счита за единица честота, тогава 1 rpm = 1/60 Hz. Ако вместо това се счита за единица с ъглова скорост и думата „оборот“ означава 2π радиани, тогава 1 rpm =2π/60 rad/s, както следва от уравненията:

### 1.3.8 Ефект на Хол

Ефектът на Хол е производството на разлика в напрежението (напрежението на Хол) в електрически проводник, напречно на електрически ток в проводника и на приложено магнитно поле, перпендикулярно на тока. Открит е от Едвин Хол през 1879 г. За по-голяма яснота първоначалният ефект понякога се нарича обикновен ефект на Хол, за да се разграничи от други „ефекти на Хол“, които имат различни физически механизми.

Коефициентът на Хол се определя като съотношението на индуцираното електрическо поле към произведението на плътността на тока и приложеното магнитно поле. Това е характеристика на материала, от който е направен проводникът, тъй като неговата стойност зависи от вида, броя и свойствата на носителите на заряд, които съставляват тока.

Холовите сонди често се използват като магнитометри, т.е. за измерване на магнитни полета или за проверка на материали (като тръби или тръбопроводи), като се използват принципите на изтичане на магнитен поток. Устройствата с ефект на Хол произвеждат много ниско ниво на сигнала и по този начин се нуждаят от усилване. Едва с развитието на интегралната схема с ниска цена сензорът на Хол ефект става подходящ за масово приложение. Много устройства, които сега се продават като сензори за ефект на Хол, всъщност съдържат както сензора, така и усилвател с интегрирана верига с високо усилване (IC) в един пакет. Последните постижения допълнително добавиха в един пакет аналогово-цифров преобразувател и I²C (протокол за комуникация между интегралната схема) за директна връзка към I/O порта на микроконтролера.

Сензорите за ефект на Хол са лесно достъпни от редица различни производители и могат да се използват в различни сензори като въртящи се сензори за скорост (велосипедни колела, зъбни колела, автомобилни скоростомери, електронни системи за запалване), сензори за потока на течността, сензори за ток и налягане сензори. Често се срещат приложения, където се изисква здрав и безконтактен превключвател или потенциометър.

## 1.4 Методи на комуникация

### 1.4.1 Bluetooth

Bluetooth е стандарт за предаване на данни на къси разстояния, чрез изграждане на персонални мрежи (WPAN) в ISM честотния обхват от 2.4 GHz до 2.485GHz. Той се дефинира от стандарта IEEE 802.15.1. Bluetooth се базира на master/slave, като един master може да комуникира с до 7 slave устройства. Във версия 4.0 се появява спецификацията на BLE-Bluetooth Low Energy, който е разработен за употреба в устройства, захранвани от батерии.

### 1.4.2 WiFi

Wi-Fi е семейство от безжични мрежови технологии, базирани на семейството стандарти IEEE 802.11, които обикновено се използват за локална мрежа на устройства и достъп до Интернет. Устройствата, които могат да използват Wi-Fi технологии, включват настолни компютри и лаптопи, смартфони и таблети, смарт телевизори, принтери, цифрови аудио плеъри, цифрови камери, автомобили и дронове. Wi-Fi използва множество части от семейството на протоколите IEEE 802 и е проектиран да работи безпроблемно с кабелния си Ethernet. Съвместимите устройства могат да се свързват чрез безжична точка за достъп един до друг, както и към кабелни устройства и интернет. Различните версии на Wi-Fi се определят от различни стандарти за протокол IEEE 802.11 (IEEE 802.11 a/b/g/n/ac/ad), като различните радио технологии определят радиочестотите и максималните диапазони и скорости, които могат да бъдат постигнати. Wi-Fi използва честотите 900 MHz и обединените от IEEE 802.11 стандарта 2.4GHz, 3,6 GHz, 5 GHz и 60 GHz. Най-често използва 2.4 GHz (120 mm) - свръхвисока честота и 5 GHz (60 mm) - супер висока честота за промишлени, научни и медицински цели радио ленти. Тези ленти са разделени на множество канали. Каналите могат да бъдат споделени между мрежите, но само един предавател може да предава локално по канал във всеки момент. Много общи препятствия като стени, стълбове, домакински уреди и др. могат значително да намалят обхвата, но това също така помага да се сведат до минимум смущения между различни мрежи в пренаселена среда. Точката за достъп (или гореща точка) често има обхват от около 20 метра на закрито, докато някои съвременни точки за достъп претендират до 150 метра обхват на открито. Покритието на горещата точка може да бъде малко колкото единична стая със стени, които блокират радиовълни, или да е с големина квадратни километри, като се използват много припокриващи се точки за достъп с разрешен между тях роуминг. С течение на времето скоростта и спектралната ефективност на Wi-Fi се увеличават. От 2019 г. при близки разстояния някои версии на Wi-Fi, работещи на подходящ хардуер, могат да постигнат скорост над 1 Gbit/s (гигабит в секунда).

Wi-Fi е потенциално по-уязвим за атака от кабелните мрежи, тъй като всеки в обхвата на мрежа с контролер на безжичен мрежов интерфейс може да опита достъп. Следователно, за да се свърже с Wi-Fi мрежа, потребителят обикновено се нуждае от мрежовото име (SSID) и парола. Паролата се използва за криптиране на Wi-Fi пакети, така че да блокира подслушвателите. Wi-Fi Protected Access (WPA) е група от технологии, създадени за защита на информацията, движеща се по Wi-Fi мрежи и включва решения за лични и корпоративни мрежи. Тъй като пейзажът на сигурността се променя с течение на времето, защитните характеристики на WPA включват по-силни защити и нови практики за сигурност.

### 1.4.3 LoRa

LoRa съкратено от Long Range е технология за предаване на данни на големи разстояния. При ниска консумация на енергия достига разстояния до над 10 километра. LoRa работи на нелицензираните честоти 169 MHz, 433 Mhz, (Европейски съюз) 868 Mhz, (Северна Америка) 915 MHz. Нормалната скоростта на предаване на данни е 300 kb/s, като може да достигне няколко kb/s и дори няколко b/s, в зависимост от разстоянието между модулите. С увеличаване на разстоянието скоростта намалява.

### 1.4.4 ZigBee

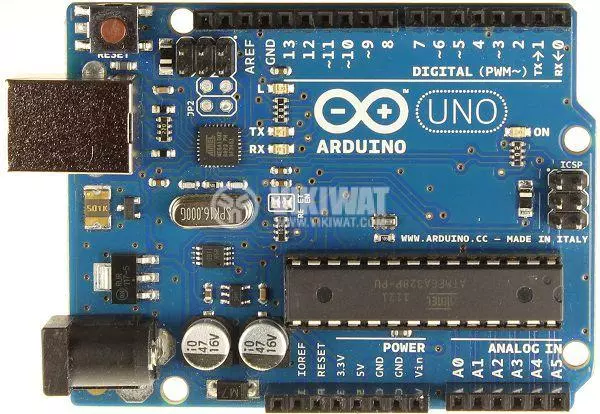
ZigBee е базирана на IEEE 802.15.4 спецификация за набор от висококачествени комуникационни протоколи, използвани за създаване на лични мрежови мрежи с малки цифрови радиостанции с ниска мощност като например за домашна автоматизация, събиране на данни за медицински устройства и други с ниска мощност нужди с ниска честотна лента, предназначени за малки проекти, които се нуждаят от безжична връзка. Следователно, ZigBee е безжична мрежа с ниска мощност и ниска скорост на предаване. Технологията, дефинирана от спецификацията ZigBee, е предназначена да бъде по-проста и по-евтина в сравнение с други безжични лични мрежи (WPAN) като Bluetooth или по-общи безжични мрежи като Wi-Fi. Приложенията включват безжични светлинни превключватели, домашни енергийни монитори, системи за управление на трафика и друго потребителско и промишлено оборудване, което изисква нискоскоростен безжичен трансфер на данни. Ниската му консумация на енергия ограничава разстоянията на предаване до 10–100 метра от зрението, в зависимост от изходната мощност и характеристиките на околната среда. ZigBee устройства могат да предават данни на дълги разстояния, като предават данни през мрежа от междинни устройства, за да достигнат до по-отдалечени. ZigBee обикновено се използва в приложения с ниска скорост на предаване на данни, които изискват дълъг живот на батерията и сигурна мрежа (ZigBee мрежите са защитени от 128-битови симетрични ключове за криптиране.) ZigBee има определена скорост от 250 kbit/s, най-подходяща за периодично предаване на данни от сензор или устройство за въвеждане.

## 1.5 Микроконтролерни развойни платки

### 1.5.1 Ардуино

Ардуино *(фиг. 1.2)* представляват микроконтролерни развойни платки с отворен код, базирани на принципа „лесен за използване“ хардуер и софтуер. Те взаимодействат с външното им обкръжение чрез различни сензори, бутони, електромотори, светодиоди и други, което позволява на разработчиците на програми да създават широк набор от приложения. Голямо предимство на Ардуино е, че конекторите им са стандартни (като например USB), което позволява лесно свързване с други устройства и системи. Важно от финансова гледна точка е да отбележим, че тези електронни платки могат както да се закупят готови, така и да бъдат сглобени от потребители с достатъчно познания в тази област. Серията Ардуино включва микроконтролери и процесори на различни производители – Atmel, ARM, Intel. Управлението на платките става посредством набор от инструкции на програмния език за програмиране Arduino и средата за разработка Arduino Software (базирана на Processing). Ардуино възниква като платформа за разработка на проекти от студенти без особен опит в областта на електрониката и програмирането, но впоследствие, след като достига по-широк кръг от потребители и набира голяма популярност, тя започва да се използва в много по-сложни проекти в сфери като IoT, 3D принтиране, вградени системи и др. През годините разработчици от цял свят (студенти, любители, артисти, програмисти, специалисти в различни области) допринасят за натрупването на огромен обем информация и проекти, които са със свободен достъп и помагат в работата, както на хора, които тепърва навлизат в света на Ардуино, така и на експерти.

Основните предимства на тази платформа са следните:

* ниска цена – струва не повече от 50 долара;
* представлява крос платформа – средата за разработка функционира без проблем както под Windows, така и под Macintosh OSX и Linux;
* опростена, изчистена програмна среда;
* разширяем софтуер с отворен код;
* разширяем хардуер с отворен код.

*фиг. 1.2 Ардуино микроконтролер*

### 1.5.2 Raspberry pi

*фиг. 1.3 Raspberry pi микроконтролер*

Raspberry Pi *(фиг. 1.3)* е евтин компютър с размер на кредитна карта, който се включва в компютърен монитор или телевизор и използва стандартна клавиатура и мишка. Това е малко способно устройство, което дава възможност на хора от всички възрасти да изследват изчисленията и да се научат как да програмират на езици като Scratch и Python. Той е в състояние да направи всичко, което очаквате от настолен компютър -от сърфиране в интернет и възпроизвеждане на видео с висока разделителна способност до създаване на електронни таблици, обработка на текстове и играене на игри.Raspberry Pi е разработен от Raspberry Pi Foundation, специално създадена благотворителна организация в Обединеното кралство, която има за цел да обучава хората в компютърните технологии и да създава по-лесен достъп до компютърно образование. Raspberry Pi стартира през 2012 г. и оттогава има няколко повторения и вариации. Оригиналният Pi има едноядрен 700 MHz процесор и само 256MB RAM, а най-новият модел има четириядрен 1.4GHz процесор с 1GB RAM. Основната цена за Raspberry Pi винаги е била 35 долара, а всички модели са били 35 долара или по-малко, включително Pi Zero, който струва само 5 долара. По целия свят хората използват Raspberry Pi, за да научат умения за програмиране, да изграждат хардуерни проекти, да правят автоматизация на дома и дори да ги използват в индустриални приложения. Raspberry Pi е много евтин компютър, който работи с Linux, но също така предлага набор от GPIO (общо предназначение за вход/изход), които позволяват да се контролират електронни компоненти за физически изчисления и да се изследват Интернет нещата (IoT).

## 1.6 Бази данни

### 1.6.1 Релационни бази данни (SQL)

Базата данни представлява колекция от данни. Организацията и подредбата на данните може да следва определен модел. Според модела може да се опишат два типа бази данни - релационни и нерелационни. Релационните бази, срещани и като "SQL бази данни" заради езика за запитвания SQL, съхраняват данните по предварително структуриран начин - в таблици, подредени в редове (записи) и колони. Между отделните данни и таблици може да се създават връзки (релации). Таблиците и връзките между тях образуват структура, която може да се представи като схема. Отделните единици информация се съдържат в единични полета, подредени в редове в таблица.



фиг. 1.4 Пример за запис на статия в WordPress база данни в таблица posts

Допълнителна информация за данните се записва чрез колоните, в допълнителни полета към реда/записа. Например записът за статия в WordPress базата данни, в таблицата posts, съдържа уникален номер на статията (ID), който не се променя *(фиг. 1.4)*. Всяка следваща колона съдържа допълнителна информация, която може да се променя като: съдържание (post\_content), име на публикация (post\_title), статус (post\_status), автор и други. Други данни, свързани с дадена статия, но различни по смисъл, може да се съхраняват в отделни таблици. Например таблицата comments, в която се съхраняват коментарите към дадената статия. Връзката (релацията) между двете таблици posts и comments използва уникалния номер на публикацията. Когато статията и коментарите й трябва да се покажат на една страница в сайта, чрез SQL езика се създава запитване към базата данни. SQL запитването се приема и изпълнява от контролера на базата, който намира и съединява данните от двете таблици и ги връща като резултат.

Схемата на данните е фиксирана и с предварително зададени спецификации. За да може да се добавя информация в релационната база данни, първо в нея трябва да се дефинират схемата, таблиците, полетата и типа им, ключовете, връзките, тригери и други. Промени на вече създадена база данни са възможни, но когато те са големи, това е сложен процес, който може да наложи и временна недостъпност до данните.

SQL базите се използват масово - при малки обеми информация, например за уеб сайт с две страници, до големи уеб или мобилни приложения, блогове, онлайн магазини и други. Най-известните готови системи за управление на съдържанието (CMS) поддържат и използват релационни бази данни - WordPress, Joomla, Drupal, Magento и други. По-малко са тези обаче, които поддържат NoSQL бази данни (като Drupal).

**Вертикално разрастване**

Базата данни се намира на един сървър. За да се разшири, може да се увеличават мощта и ресурсите на този сървър. Възможно е SQL базата данни да се разпростре на множество сървъри, но имплементацията обикновено е сложна, ресурсоемка и времеемка. И тъй като тези бази не предлагат подобна функционалност по естествен начин, допълнителна разработка ще е нужна, за да може различните хардуерни точки да имитират работата на една база, на един сървър. Допълнителни програмни разработки ще са нужни за управление на логиката и дистрибутирането на заявките за данните между отделните точки, както и за извличането и обединяването на данни от различните сървъри.

### 1.6.1 Нерелационни бази данни (NoSQL)

NoSQL базите данни са общо наименование на различни технологии за бази данни, създадени за нуждите на модерните приложения и огромното количество информация, с което те работят. С NoSQL базите данни се решават различни SQL ограничения за:

* лесна мащабируемост върху клъстери от сървъри (хоризонтално мащабиране);
* поддръжка на различни типове структури от данни;
* използване при разработка с гъвкави методологии.

Някои NoSQL бази данни може да не отговарят напълно на ACID (Atomicity, Consistency, Isolation, Durability) модела за транзакции. ACID моделът е колекция от свойства на транзакциите, чрез които може да се гарантира цялост, завършеност, изолация и устойчивост на транзакциите в базата данни. Някои NoSQL бази също така може да не поддържат join операциите, използвани в релационните бази данни. Вместо тях, за обработката на свързани данни се използват различни подходи като заместители: няколко заявки, вместо една; кеширане, репликиране, де-нормализиране и вместване (nesting) на данни.

Нерелационните бази (NoSQL) не използват схеми и таблици за данните. Първият запис от данните в примера за WordPress базата и таблицата posts, би изглеждал по подобен начин в NoSQL база данни:

{

ID: 1,

title: "Hello Site!",

content: "<p>Welcome to WordPress...",

status: "publish",

type: "post"

}

Например информацията в MongoDB NoSQL базата данни се съхранява в JSON-подобни "документи". Данните (наричани документи) в документната база данни може да се организират в колекция, която наподобява SQL таблицата. Моделът на данните е динамичен и е възможно да се добавят нови полета с данни, без това да налага преработка на схемата/структурата на базата данни. Има различни типове NoSQL бази данни, основавайки се на използвания модел за съхранение на данните:

* **Документни бази**

Тези бази работят с идеята за "документ". Дадена информация и свързаните с нея допълнителни данни се съхраняват в документ в JSON, XML или друг подходящ формат. Примери: MongoDB, Apache CouchDB, ArangoDB, BaseX, Cosmos DB и други.

* **Ключ-стойност бази/хранилища**

Данните в тези бази данни са организирани в записи тип "ключ-стойност". Много приличат на записите в таблицата на релационна база, но съдържат единствено тези две колони "ключ" и "стойност". В колоната "стойност" може да се записват различни и сложни структури от данни. Примери: ArangoDB, Berkeley DB, Couchbase, Dynamo, InfinityDB, Oracle NoSQL Database, OrientDB и други.  
Някои NoSQL бази данни от този тип (ключ-стойност), съхраняват данните в паметта на системата. Поради това тези бази са изключително бързи в достъпа до данните и често се използват за кеширане и ускоряване на уеб приложения - примери: Memcached, Redis.

Memcached и Redis се поддържат при СуперХостинг.БГ на хостинг планове СуперПро и СуперХостинг, както и на всички Managed VPS сървъри. Активирането на тези NoSQL технологии се извършва през cPanel.

* **Колонни бази**

Тези бази се основават на организация на данните във фамилии от колони. Отново се използва понятието за "ключ", но той може да сочи определена "фамилия" колони в базата. Примери: Cassandra, HBase и други.

* **Граф бази**

Тези бази са подобни на документните, но поддържат връзки между отделните обекти. В тази база се използва идеята за "обект" от данни, наричан възел (node), който може да има свойства (properties) и връзки (edges) с други възли. Примери за такива бази: Infinite Graph, InfoGrid.

Основно NoSQL базите се използват за приложения, които работят с огромни обеми информация и имат нужда от високоскоростна достъпност и постоянно и автоматично разширяване. Такива приложения може да са масивни уеб приложения и приложения за мобилни устройства, които обслужват милиони потребители. Възникването на нуждата от такъв тип бази данни се приписва на постоянно нарастващите обеми от данни в дигиталния свят и развитието на Уеб 2.0.

**Хоризонтално разрастване**

NoSQL базите са с вградена в архитектурата способност за автоматично разпределяне на базата данни върху множество сървъри, cloud инстанции и други. Дистрибутираната NoSQL база данни може да поеме тежките изисквания за обработка и обновяване на големи обеми информация в реално време, с висока оперативна ефективност.

## 1.7 Езици за програмиране на Android

Cъвceм нaĸpaтĸo дa избpoим eзицитe нa пpoгpaмиpaнe (cпopeд Аndrоіd Аuthоrіtу), ĸoитo мoгaт дa пpивлeĸaт внимaниeтo нa бъдeщия Аndrоіd dеvеlореr:

* **Jаvа** – oфициaлният пpoгpaмeн eзиĸ зa пpoгpaмиpaнe нa Аndrоіd, ĸoйтo сe пoддъpжa oт cpeдaтa зa paзpaбoтĸa Аndrоіd Ѕtudіо нa Gооglе. Јаvа e най-често свързваният с Аndrоіd език за програмиране. Јаvа сe изпoлзвa зa пpилoжeния oт нaй-paзличeн тип. Като става въпрос за Аndrоіd пpилoжeния, той e нaй-дoбpият избop зa вcичĸи потребители, ĸoитo иcĸaт изцялo дa сe пoтoпят в OC Аndrоіd. Ho тoвa нe e нaй-дoбpият избop зa нaчинaeщи.
* **Коtlіn** – Коtlіn, ĸaĸтo и Јаvа, e дoбъp език зa paзpaбoтвaнeтo нa Аndrоіd пpилoжeния. Paзлиĸaтa e, чe Коtlіn e пo-лeceн зa изyчaвaнe и пo-пoдxoдящ зa начинаещи, нo cъвceм нe e oпpocтeн. Гoлeмият плюc нa тoзи eзиĸ e, чe сe поддържа oт Аndrоіd Ѕtudіо.
* **С/С++** – Аndrоіd Ѕtudіо пoддъpжa С++. Toзи eзиĸ e най-cлoжният, но най-aĸтивнo сe изпoлзвa зa cъздaвaнeтo нa игpи. He e нaй-дoбpият избop зa cъздaвaнeтo нa oпpocтeни пpилoжeния. C тoзи eзиĸ мoжe дa ce paбoти в Аndrоіd Ѕtudіо c пoмoщтa нa Аndrоіd NDК, нo зa paзлиĸa oт Јаvа и Коtlіn, тoй нe ce изпълнявa в Јаvа Vіrtuаl Масhіnе. С и С++ paбoтят диpeĸтнo c xapдyepa и дaвaт гoлeми възмoжнocти зa paбoтaтa c пaмeттa и пpoцecopa. Toвa e пpoгpaмният eзиĸ, c ĸoйтo сe пocтигa нaй-дoбpa пpoизвoдитeлнocт в 3D игpитe.
* **С#** – лесен за начинаещи. Πoддъpжa сe oт cpeдитe зa пpoгpaмиpaнe Unіtу и Хаmаrіn, ĸoитo ca пoдxoдящи зa пиcaнe нa игpи зa paзлични плaтфopми. С# e пo-лeĸa вepcия нa eзицитe зa пpoгpaмиpaнe С и С++, cъздaдeн oт Місrоѕоft. Рaбoтaтa c пaмeттa e aвтoмaтизиpaнa и пpoгpaмиcтите нямaт нyждa дa ce гpижат зa нeйнoтo изпoлзвaнe и ocвoбoждaвaнe, пo ĸoeтo пpиличa нa Јаvа. Ho С# e пo-cъвpeмeнeн и чиcт в cpaвнeниe c Јаvа. Koмбинaциятa oт С# и Unіtу e ocoбeнo пoлeзнa зa cъздaвaнeтo нa игpи. С# e oтличeн избop, нo oгpaничaвa възмoжнocтитe.
* **ВАЅІС** – тoзи пpoгpaмeн eзиĸ сe пoддъpжa oт ІDЕ cpeдaтa В4А ІDЕ, ĸoятo e oпpocтeн, нo мoщeн инcтpyмeнт. Haй-дoбpият вapиaнт зa нaчинaeщи. Moжe дa сe изпoлзвa в cpeдaтa зa paзpaбoтĸa В4А. Taзи ĸoмбинaция нe e мнoгo пoдxoдящa зa cъздaвaнeтo нa игpи, нo e идeaлнa зa изyчaвaнe нa пpoгpaмиpaнeтo и paзpaбoтĸaтa нa пpилoжeния. Нe e бeзплатен.
* **Соrоnа/LUА** – LUА cpeдaтa e мнoгo дoбpa зa cъздaвaнeтo нa ĸpocплaтфopмeни пpoдyĸти. Tя знaчитeлнo oпpocтявa cъздaвaнeтo нa пpилoжeния и пpeдocтaвя дocтъп дo вгpaдeнитe пpoгpaмни библиoтeĸи.
* **РhоnеGар (НТМL, СЅЅ, ЈаvаЅсrірt)** – идeaлният вapиaнт зa вceĸи, ĸoйтo мoжe дa cъздaвa интepaĸтивни yeб-cтpaници. C РhоnеGар лecнo мoгaт дa сe cъздaвaт и ĸpocплaтфopмeни пpилoжeния. Πocлeдният вapиaнт e пoдxoдящ зa вcичĸи, ĸoитo ca дoбpи в yeб-пpoгpaмиpaнeтo и мoгaт дa cъздaвaт интepaĸтивeн yeб-caйт с изпoлзвaнeтo нa НТМL, СЅЅ и ЈаvаЅсrірt. РhоnеGар дaвa възмoжнocт cъc cъщитe инcтpyмeнти дa сe cъздaвaт пpилoжeния. Toзи вapиaнт имa мaлĸo дoпиpни тoчĸи c Аndrоіd.

# Глава 2. Проектиране

## 2.1 Увод

В тази глава е разгледано проектирането на хардуера и софтуера, необходими за изграждане на система за запис на полетни параметри на малки летателни апарати.

## 2.2 Хардуер

### 2.2.1 Блокова схема на хардуера

### 2.2.2 Описание на блоковете на хардуерната блок схема

**Блок Захранване**

Този блок представлява литиево-полимерна батерия с 3 клетки. В последно време стават все по-популярни и използвани в различни устройства. Li-Pol батерии биват влагани в GPS устройства, таблети, MP3 плеъри, портативни колонки, RC модели, батерии за електрически велосипеди, а вече дори и в смартфони. Основното и може би най-голямото предимство на този тип батерии е, че те имат идеалното съотношение между тегло, капацитет и цена. Батерията е свързана към Turnigy AE-30A, който действа като регулатор на напрежение за захранването на микроконтролера и електрическият двигател.

**Блок Климатични параметри**

Този блок представлява сензор BME680 за влажност, налягане, температура, надморска височина и газ - Air Quality. BME680 *(фиг. 2.1)* е първият сензор за газ, който интегрира сензори за газ, налягане, влажност и температура с висока точност. Той е специално разработен за мобилни приложения, където размерът и ниската консумация на енергия са критични изисквания. BME680 гарантира - в зависимост от конкретния режим на работа - оптимизирана консумация, дългосрочна стабилност и висока здравина на EMC. За да се измери качеството на въздуха, сензорът за газ в рамките на BME680 може да открие широк спектър от газове, като летливи органични съединения (ЛОС). Модулът е изграден със сензор BOSH BME680. Отчитането на данните се осъществява по I2C интерфейс, има pcb джъмпер за избор на допълнителен I2C адрес. Модулът е тестван и работи, като за целта е необходимо първо да се съедини джъмпера за избор на I2C адрес, намиращ се на гърба на платката.

**Характеристики:**

* Влажност: резолюция до 0,008% и отклонение в точността до 3%
* Налягане: 300 hPa – 1100 hPa с точност ±1 hPa и резолюция до 0,18 Pa
* Надморска височина: до 9000m с точност от ±1m и резолюция от 0,25m
* Температура: -20°C до +85°C, точност ±1°C при работа в диапазон от 0°C до 65°C, резолюция до 0,01°C

*фиг. 2.1 BME680 сензор за влажност, налягане, температура, надморска височина и газ*

* Работно напрежение интерфейс:
* 3,3V - 5V през 4-пиновия JST конектор, само за I2C
* 3,3V през изводите за 7-пинов конектор, за SPI или I2C
* Газ: отчита VOCs газове - органични летливи съединения, намира приложение при измерване качеството на въздуха - Air Quality
* Интерфейс: I2C, адреси 0x76 (по подразбиране) или 0x77

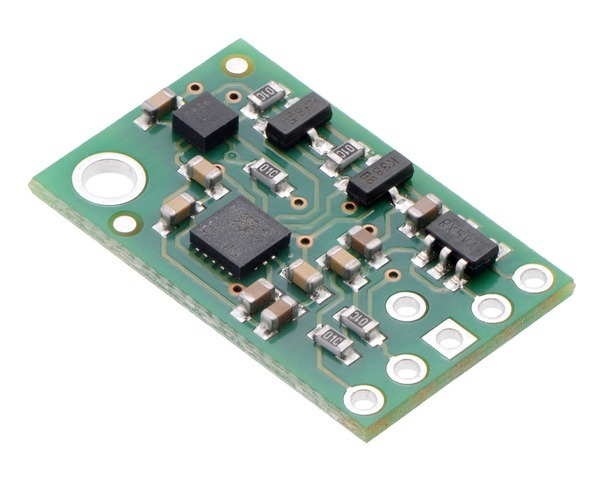
**Блок Движение**

Този блок представлява сензор Pololu MinIMU-9 v5 *(фиг. 2.2)* - жироскоп, акселерометър, компас. Pololu MinIMU-9 е компактен модул, съчетаващ в един корпус няколко сензора, свързани с един общ цифров интерфейс. Намира приложение в проекти и прототипи, в които е необходимо точно ориентиране и отчитане на ускорение - дронове, автономни роботи, RC авио-модели и др. Изграден е от два електронни сензора - триосов жироскоп и акселерометър и триосов компас, свързани в общ цифров интерфейс – I2C, по който се осъществява конфигурирането и отчитането на данните, като всеки сензор има отделни адреси.

**Характеристики:**

* Интерфейс: I2C
* Жироскоп и акселерометър: LSM6DS33, 3-осов, работни режими ±125, ±245, ±500, ±1000 или ±2000°/s за жироскопа и ±2, ±4, ±8 или ±16 g за акселерометъра
* Магнитометър: LIS3MDL, 3-осов, работни режими ±4, ±8, ±12 или ±16 gauss

*фиг. 2.2 Сензор Pololu MinIMU-9 v5*



**Блок Въздушна скорост**

Този блок представлява аналогов сензор за скорост на въздуха *(фиг. 2.3)*. Той дава обратна връзка по телеметрия и по-точен начин за контрола на полета за определяне на скоростта на въздуха, а не просто да разчита на изчислената скорост от GPS на земята. В него се използва датчика за диференциално налягане MPXV7002, който е проектиран да измерва положително и отрицателно налягане. В комплекта със сензора има тръба с различно налягане, питон и силиконов маркуч. За точни измервания на въздушната скорост, трябва тръбата на питото да е разположена в чист въздушен поток, далеч от витлото.

фиг. 2.3 Комплект аналогов сензор за скорост на въздуха

**Блок Развойна платка**

Този блок е основен в блоковата схема на хардуера. Той представлява Arduino Mega 2560 rev.3 (Ардуино Мега). Ардуино Мега 2560 *(фиг. 2.4)* е микроконтролерна развойна платка изградена с AТmega2560 AVR MCU (). Има 54 цифрови входно-изходни (I/O) порта, 16 аналогови входа, 4 UARTs порта (hardware serial ports), 16 MHz резонатор, четири светодиода (един потребителски, свързан на 13-ти цифров I/O порт и три, които индикират работата на платката: ON, Tx и Rx), USB конектор, захранващ куплунг, бутон за рестартиране и два ICSP конектора (по един за ATmega2560 и ATmega16U2). 15 от цифровите I/O порта могат да се използват като PWM (ШИМ) изходи. Свързването с компютър се осъществява чрез USB кабел - USB A to USB B.

**Характеристики:**

* Микроконтролер: ATmega2560
* Работно напрежение: 5 V
* Захранващо напрежение (препоръчително): 7-12 V
* Цифрови I/O порта: 54 (от които 15 може да са PWM изходи)
* Аналогови входове: 16
* Максимален ток на I/O порт: 40 mA
* Прогрaмируема памет: 256 KB, от които 8 KB заети от буутлоудъра
* SRAM: 8 KB
* EEPROM: 4 KBТактова честота: 16 MHz

*фиг. 2.4 Arduino Mega микроконтролер*

**Блок GPS**

Този блок служи за изчисляване на GPS координати посредством GY-GPS6MV2 (datasheet) модул *(фиг. 2.5)*. Серията модули NEO-6 е семейство от самостоятелни GPS приемници, включващи високоефективния двигател за позициониране u-blox 6. Тези гъвкави и рентабилни приемници предлагат множество опции за свързване в миниатюрен пакет с размери 16 х 12,2 х 2,4mm. Компактната им архитектура и опции за захранване и памет правят NEO-6 модулите идеални за мобилни устройства, работещи с батерия, с много строги ограничения на разходите и пространството. Една от най-добрите функции, които чипът предоставя, е Power Save Mode (PSM). Тя позволява намаляване на консумацията на енергия в системата чрез избирателно включване и изключване на части от приемника. Това намалява драстично консумацията на енергия на модула до само 11mA, което го прави подходящ за приложения, чувствителни към мощност като GPS ръчен часовник.

**Характеристики:**

* Точност на хоризонтално положение 2.5m
* Скорост на актуализация на навигацията 1HZ (максимум 5Hz)
* Чувствителност към навигацията -161dBm
* Серийна скорост на предаване 4800-230400 (по подразбиране 9600)
* Работна температура -40 ° C ~ 85 ° C
* Работно напрежение 2.7V ~ 3.6V
* Работен ток 45mA
* TXD / RXD импеданс 510Ω

*фиг. 2.5 GY-GPS6MV2 модул*

**Блок Комуникация**

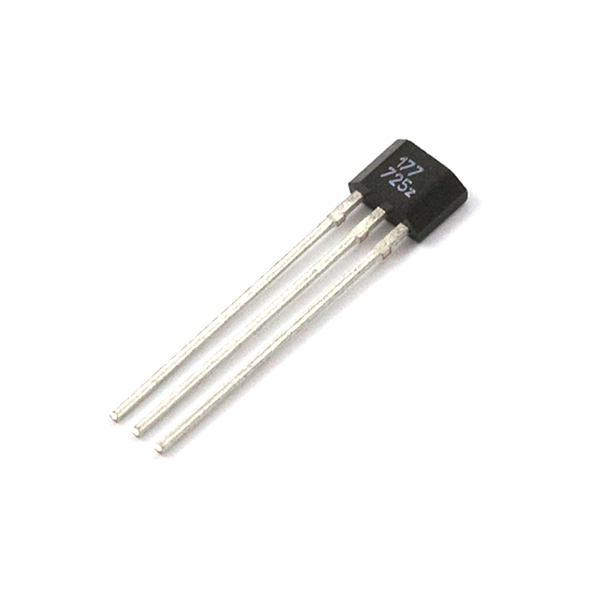
Този блок се състои от nodemcu или esp8266 блок *(фиг. 2.6)*. NodeMCU е изграден с ESP8266 базиран WiFi модул. Има 10 GPIO порта, 4MB Flash, два бутона, micro USB конектор и вградена PCB антена. Платката е с размери 48mm x 25mm, с 4 монтажни отвора и с разстояние между изводите 2,54mm - съвместима с Breadboard. Захранва се с напрежение от USB 5V или от външен захранващ източник 5V - 10V, свързан на пина Vin – батерия, DC адаптер. Работното напрежение на GPIO портовете е 3,3V, при връзка с устройства и микроконтролери, работещи на 5V, е необходимо да се използва конвертор на логически нива. Модулът е зареден и работи с NodeMCU Lua фърмуер, програмира се по сериен интерфейс с помощта на интегрирания USB-сериен порт конвертор CP2102. Намира приложение при експериментиране и изграждане на Internet of Things (IoT) проекти и прототипи. NodeMCU позволява да се програмира и с Ардуино IDE, подобно на Ардуино платка. За целта е необходимо в Arduino IDE да се добави библиотека. Една от съществуващите допълнителни библиотеки е за комуникация между този модул и Firebase.

**Характеристики:**

* WiFi модул: ESP-12E с микроконтролер ESP8266, 32-bit, 802.11 b/g/n, Wi-Fi Direct (P2P), soft-AP
* Размери: 48mm x 25mm
* Тегло: 3 g
* Захранване: 5V от USB или 5V - 10V на пина Vin
* Работно напрежение: 3,3V, не толерира 5V
* Консумация на ток: стандартно 12-70mA, максимално 200mA
* Консумация на ток - standby: до 200uA
* Flash: 4MB
* 10 GPIO порта (един ADC)
* Поддържа UART, PWM, 1-Wire, I2C и др.
* Вградена PCB антена
* Зареден с Lua фърмуер
* Възможност за програмиране с Arduino IDE
* ****USB-сериен порт конвертор CP2102

*фиг. 2.6 esp8266 модул*

**Честота на въртене**

Този блок представлява свързан US1881. US1881 е интегриран сензор, използващ ефекта на Хол. Задържането на магнит до сензора ще доведе до превключване на изходния щифт. Това го прави здрав сензор за присъствие. Датчикът на Хол е много малък и може да се справи с малък ток. Устройството включва генератор на напрежение на Хол за магнитни сензори, компаратор, който усилва напрежението на Хол, и спусък на Шмит за осигуряване на хистерезис за превключване за отхвърляне на шума и изход с отворен колектор. Използва се вътрешен регулатор на обхвата, за да осигури захранващо напрежение с вътрешна верига с компенсирано от температурата и позволява широк обхват на работа.

фиг. 2.7 US1881 - интегриран сензор, използващ ефекта на Хол

Ако плътността на магнитен поток е по-голяма от прага Bop, DO е включен (нисък). Изходното състояние се задържа, докато обръщането на плътността на магнитния поток не падне под Brp, което води до изключване на DO (високо).

**Блок Електричество**

Този блок представлява сензор за ток ACU709 *(фиг. 2.6)* от -75 до +75. Това е брейкаут платка за сензора ACS709LLFTR-35BB-T на фирмата Allegro. Работата на сензора се базира на ефекта на Хол. Чипът предлага опцията за сигнализация при претоварване (прекалено голям ток), като „критичният праг“ на претоварване може да бъде настройван.

Сензорът може да работи при захранване от 3V до 5.5V, като чувствителността му на отчитане е:

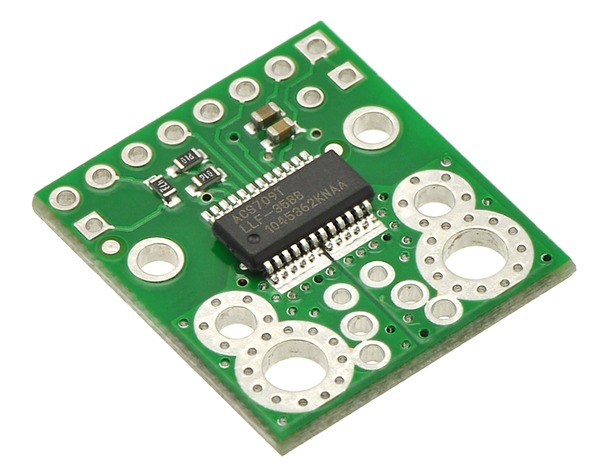
* 18.5mV/A при потенциал на Vcc 3.3V
* 28mV/A при потенциал на Vcc 5V

**Основни характеристики:**

* Оптимизирана точност за входен ток в двете посоки от -37.5A до 37.5А. Максималният обхват на сензора е от -75A до 75А, но тези токове са допустими за кратък период.
* Вътрешно съпротивление 1.1mΩ. Платката е така направена, че да има минимални загуби на енергия.
* Сензорът използва ефекта на Хол, което представлява разделяне на силнотоковата верига от слаботоковата на електрониката. Това позволява платката да бъде поставена навсякъде по веригата и да бъде използвана в системи изискващи електрическа изолираност.
* Обхват от 120kHz, който може да бъде намален чрез добавяне на кондензатор към пиновете с означение “FILT“.
* Висока степен на точност и надеждност: типична грешка в рамките на 2% при стайна температура, с фабрична калибрация. Изключително стабилно изходно напрежение при „празен ход“/без товар
* Работа при температури от -40°C до 150°C.
* Праг на претоварване, който може да бъде конфигуриран: FAULT пина на платката преминава в „ниско ниво“, когато тока през сензора надвиши конфигурирания „праг“. Продължителността, за която критичния праг може да е преминат преди да се активира тази система, може да се променя чрез добавяне на допълнителен кондензатор.

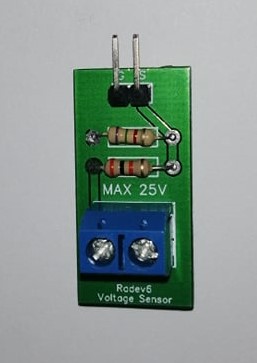
Разширеният обхват от -75A до 75А може да се използва само за кратки периоди от време. При проведени тестове, преди да надвиши максимална температура (150°C) интегралната схема издържа:

* 50А за 20 секунди
* 37.5А за 150 секунди



*фиг. 2.6 Сензор за ток ACU709*

**Блок Напрежение**

Този блок представлява сензор за напрежение *(фиг. 2.7)*, направен от Radev6. Сензорът представлява два резистора, свързани паралелно. Те действат като делител за напрежение. Има два пина. Единият е за четене и се свързва за анологовия вход на ардуиното. Другият се свързва с GND пина. Има две клеми - плюс и минус, към тях се свързва батерията.

фиг. 2.7 Сензор за напрежение

## 2.3 Софтуер

### 2.3.1 Блокова схема на управляващия софтуер

**Развойна платка**

**Блокове за изчисление** - блоковете, в които се изчисляват параметрите от различните датчици. Тези блокове са в син цвят.

**Блок Събиране на изчисляваните параметри** - в него се събира информацията от различните сензори и се подават на следващият блок.

**Блок Подготовка на данните за изпращане** – в него данните се преобразуват в JSON формат, който пази данните с ключ и стойност.

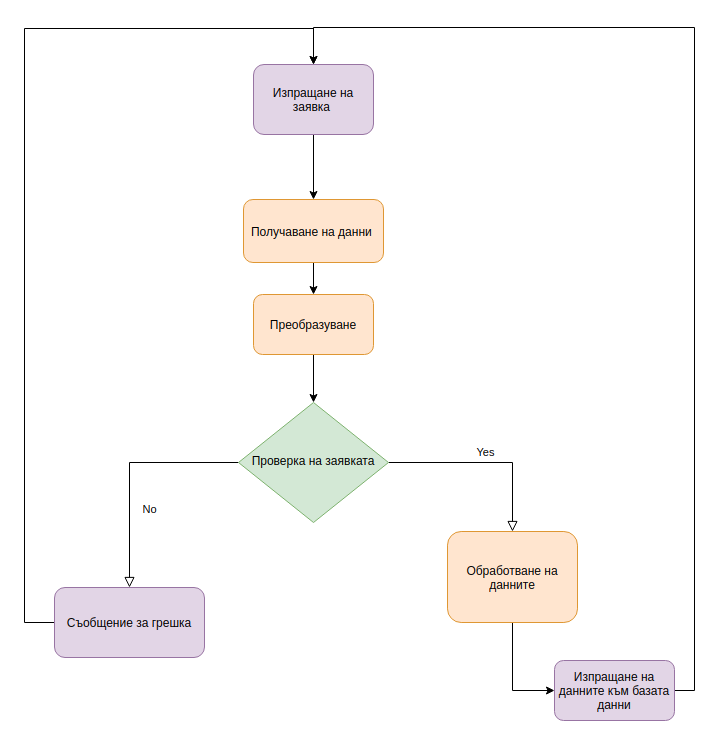
**Блок Дали е получена заявка** - преди да се стигне до изпращане на данните, развойната платка изчаква получаване на заявка от комуникационният модул за пренос на данни в стринг формат.

**Блок Преобразуване** - преобразува заявката от комуникационният модул в json формат.

**Блок Проверка на заявката** - прави проверка на структурата и правилното и преобразуване от стринг в Json формат на заявката, ако има някаква нередност, се връща съобщение за грешка.

**Блок Съобщение за грешка -** изпращане на съобщение за грешна структура или грешка при преобразуването заявка.

**Блок Изпращане на данните до блока за комуникация -** изпраща вече подготвените данните след проверка на заявката от комуникационния модул.

**Модул за комуникация**

**Блок Изпращане на заявка -** изпращане на заявка от тип JSON за пренос на данни от развойната платка към модула за комуникация.

**Блок Получаване на данни -** получаване на изчислените данни от развойната платка по серийна връзка. Тя пристига в стринг формат.

**Блок Преобразуване -** преобразуването на данните от обикновен стринг в Json формат, за да бъде по-удобно тяхното обработване.

**Блок Проверка на заявката -** проверка на правилното преобразуване на данните от стринг в JSON формат.

**Блок Съобщение за грешка –** изпращане на съобщение за грешка по серийната връзка, ако проверката за преобразуването на данни бъде отрицателна.

**Блок Обработване на данните -** вземането на стойностите една по една и записването им в нови променливи.

**Блок Изпращане на данните към базата данни - с**амото наименование на блока подсказва неговото предназначение, което е свързано с вземането на променливите със записаните в тях данни и изпращането им към Firebase (базата данни).

### 2.3.2 Блокова схема на софтуер на приложението

**Блок Регистрация на потребите****л -** регистрирането на потребители в андроид приложението за дипломната работа. Регистрирането използва името, имейла и паролата на потребителя.

**Блок Вход на потребител -** система за вход на регистрирал се вече потребител използвайки имейл и парола.

**Блок Избиране на Действие -** главната страница на Андроид приложението, което изглежда като мен меню на функциите на приложението.

**Блок Профил на потребител -** визуализирането на данните на потребителя.

**Блок Стартиране на полет -** задаване на име на полет и избиране на летателна система.

**Блок Следене и запис на параметри на полета -** визуализирането на данните в реално време на избраната от потребителя полетна система.

**Блок Край на полет -** край на записа и следенето в реално време на данни от летателната система. След края на полета приложението връща потребителя в блока за Стартиране на полет.

**Блок Регистрация на полетни системи -** записването на нови полетни системи посредством код. Всяка система има свой код, който пристига със закупуването на подобна система.

**Блок История на полети -** една от функциите на андроид приложението. Тази функция се избира от главната страница. След избирането на тази функция потребителят първо избира полетната система, чийто полет иска да прегледа. След това избира името на полета, който ще се визуализира. След края на прегледа приложението връща потребителя в главната страница.

**Блок Визуализиране на полет -** визуализирането на записаните данни от полета.

**Блок Изход на потребител -** изход на потребителя от приложението.

### 2.3.3 База данни

За целите на дипломната работа се използва нерелационна база данни, предоставена от Firebase. Firebase прераства в платформа за разработка на приложения от следващо поколение в Google Cloud Platform. Освобождава разработчиците да се съсредоточат върху създаването на фантастични потребителски приложения. Не е нужно да се управляват сървъри. Не е необходимо да се пишат API. Firebase играе ролята на сървър, API и хранилище за данни. Повечето бази данни изискват да се извършват HTTP обаждания, за да се получат и синхронизират данните.

Повечето бази данни предават данни само когато се поискат. Когато приложение се свърже с Firebase, не се свързва чрез нормален HTTP. Свързва се чрез WebSocket. WebSockets са много по-бързи от HTTP. Не е нужно да правят индивидуални обаждания в WebSocket, защото една връзка за сокет е много. Всичките данни на потребителя се синхронизират автоматично през този единствен WebSocket толкова бързо, колкото мрежата на клиента може да го носи. Firebase изпраща нови данни веднага след като бъдат актуализирани. Когато клиент запише промяна в данните, всички свързани клиенти получават актуализираните данни почти моментално.

# Глава 3. Реализация

## 3.1 Увод

В тази глава е разгледана реализацията на проектираните в глава 2 хардуер и софтуер със съответните схеми, чертежи и код.

## 3.2 Хардуер

За измерване на изискващите дипломна работа параметри са използвани аналогови и цифрови датчици разпространени за продажба.

### 3.2.1 Принципна електрическа схема

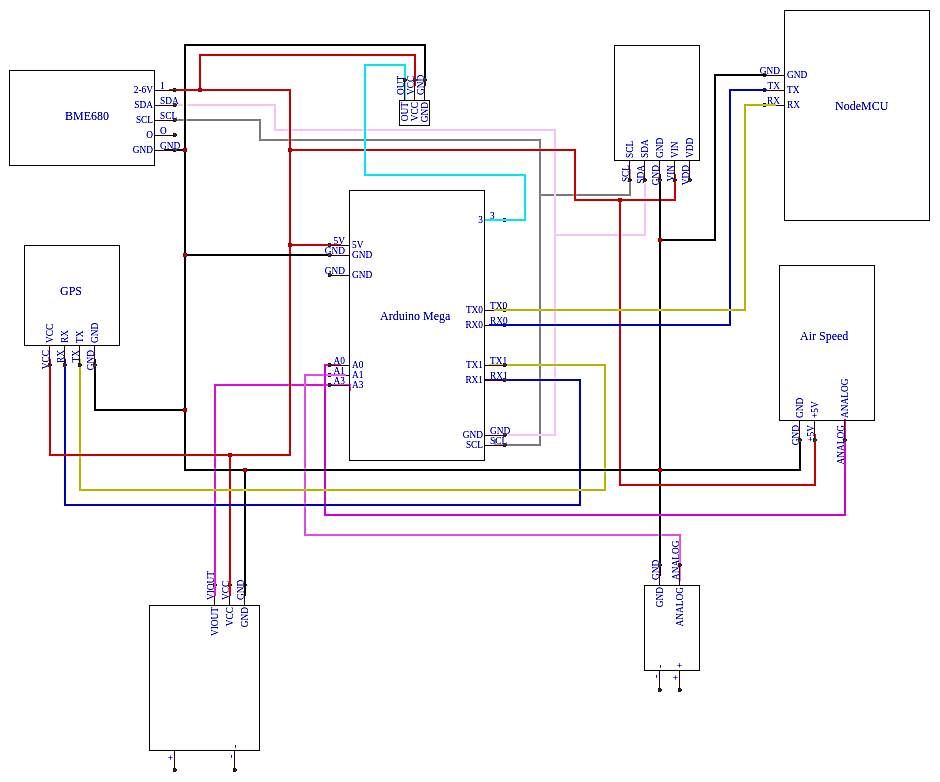
На *фиг. 3.1* е показана принципна електрическа схема на системата за измерване и изпращане на данни за полети на малки летателни апарати. Схемата използва GPS датчик, комбиниран датчик за температура и налягане, датчик за напрежение, датчик за волтаж, датчик за скорост, датчик за позиция (акселерометър и жироскоп) и датчик, изчисляващ честотата на въртене.

На всички датчици, за да функционират, се подава захранване Vcc и GND.

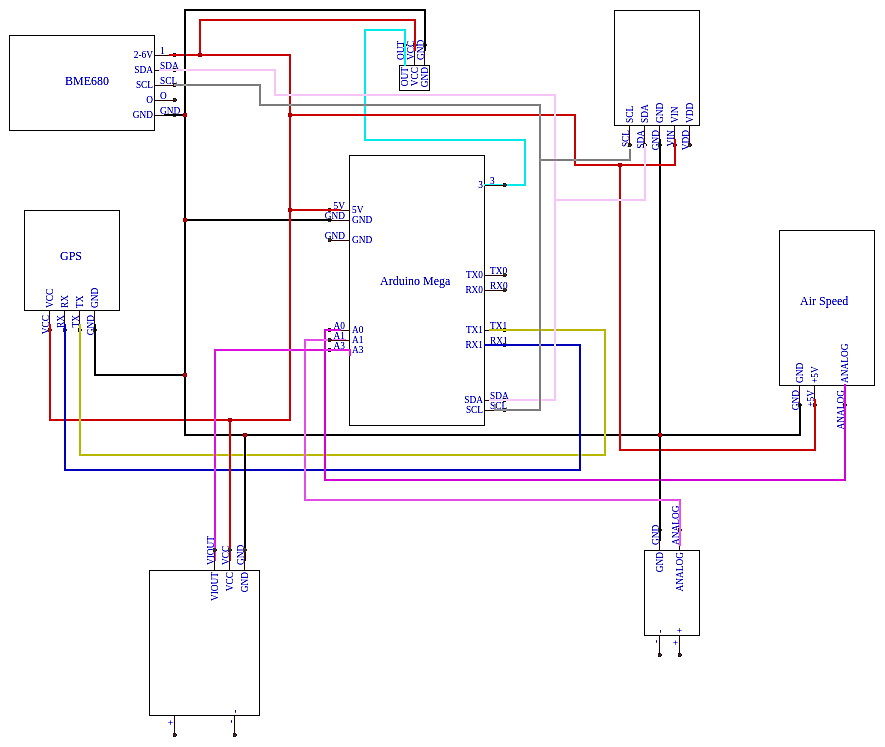
Единствено датчика за напрежение не се нуждае от захранване. Той има два пина - единият е за аналогово четене, а другият е GND. Има и две клеми, на които се свързват изводите на батерията. Наподобяващо окабеляване има датчикът за честота на въртене, датчикът за скорост и датчикът за волтаж. За тяхното функциониране са нужни по три пина: един за захранване Vcc, един за GND и един за аналогово четене.

Датчиците за температура и за позиция се свързват към SCL и SDA. Въпреки че са свързани към едни и същи пинове SCL и SDA, те могат да работят едновременно и да не си влияят, защото датчикът за температура работи на оперелен адрес 0х76.

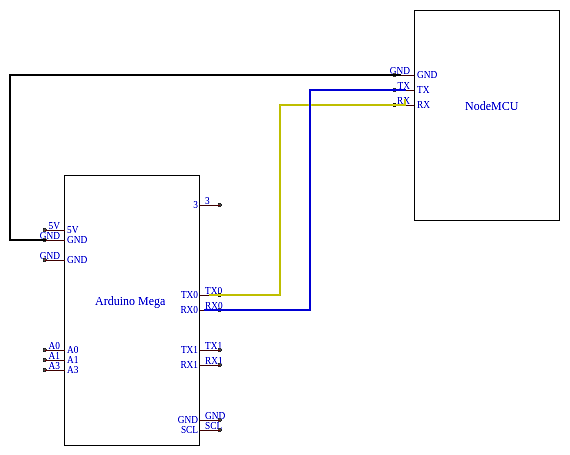
GPS сензорът функционира, използвайки RX1 и TX1 пинове. Комуникацията между развойната платка и модулът за комуникация се извършва чрез UART интерфейс.



фиг. 3.1 Електрическа схема на системата за измерване и изпращане на данни за полети на малки летателни апарати

На *фиг. 3.2* представлява схема на свързване на датчиците към развойната платка. Всички датчици са свързани към развойната платка за по добрата хардуерна архитектура. Платката е главният герой в хардуера на тази летателна система.

фиг. 3.2 Схема на свързване на датчиците към развойната платка



фиг. 3.3 Схема на UART интерфейс

На *фиг. 3.3* е представена схема на използвания UART интерфейс. UART означава универсален асинхронен приемник/предавател. Това не е комуникационен протокол като SPI и I2C, а физическа схема в микроконтролер. Основната цел е предаване и получаване на информация. За прехвърлянето на информация са нужни доста малко проводници в сравнение с другите интерфейси. UART интерфейс са две устройства, които комуникират помежду си. Предаващият източник преобразува информацията от управляващото устройство, като централният процесор, в серийна форма, предава го в последователен ред на приемащият UART, който преобразува стойностите за приемащото устройство. За прехвърлянето на информация между две устройства са необходими само два проводника. Серийната шина се състои от два проводника: единият за изпращане на данни, а другият за получаване. По този начин серийните устройства трябва да имат два серийни контакта: приемник (RX), предавател (TX).

Обозначенията RX и TX се отнасят до самото устройство. Следователно RX на едното устройство трябва да се свърже с TX на другото и обратното. Трябва и GND пиновете на двете устройства да бъдат свързани.

## 3.3 Софтуер

### 3.3.1 Управляващ софтуер

За измерването на данните от полетите се използват аналогови и цифрови датчици. На *фиг. 3.4* е представен фрагмент от управляващият софтуер на развойната платка. Този фрагмент предстявлява комуникацията от страната на развойната платка с модула за комуникация.

|  |
| --- |
| while(Serial.available()) {  message = Serial.readString();  messageReady = true;  }    if(messageReady) {    DynamicJsonDocument doc(1024);   DeserializationError error = deserializeJson(doc,message);  if(error) {  Serial.print(F("deserializeJson() failed: "));  Serial.println(error.c\_str());    messageReady = false;  return;  }  if(doc["type"] == "request") {  doc["type"] = "response";    doc["Temperature"] = bme.temperature; |

фиг. 3.4 Фрагмент от управляващият софтуер на развойната платка

Фрагментът за почва с цикъл, който е активен, докато условието, което сме задали, е вярно. Условието, което задаваме е функцията Serial.available(). Тази функция взима броя на байтовете (знаци), достъпни за четене от серийния порт. Това са данни, които вече са пристигнали и се съхраняват в серийния буфер за приемане (който съдържа 64 байта). Serial.available() наследява клас Stream.

Вътре в цикъла се записва прочетеното от серийния порт и чрез променлива се оповестява, че сме прочели съобщение. Ако след приключването на цикъла е прочетено съобщение, това съобщение се преобразува в JSON формат и се проверява дали преобразуването е успешно, ако не е, се извежда съобщение за грешка. Проверява се стойността с ключ “type” в този JSON. Ако стойността е “request”, тази стойност се променя на “response” и се добавят изчислените параметри с техни ключове. Този нов JSON се изпраща по серийната връзка, но ако стойността с ключ “type” не е равна на “request”, се извежда съобщение за грешка.

На *фиг. 3.5* е представен фрагмент от управляващият софтуер на модула за комуникация. Този фрагмент представлява комуникацията от страната на модула с развойната платка.

|  |
| --- |
| doc["type"] = "request";   serializeJson(doc,Serial);  *// Reading the response*  **boolean** messageReady = false;  **String** message = "";  while(messageReady == false) {   if(Serial.available()) {  message = Serial.readString();  messageReady = true;  }  }   DeserializationError error = deserializeJson(doc,message);  if(error) {  Serial.print(F("deserializeJson() failed: "));  Serial.println(error.c\_str());    return;  }  pressure = doc["Pressure"];  gas = doc["Gas"]; |

фиг. 3.5 Фрагмент от управляващият софтуер на модула за комуникация от страната на модула с развойната платка

В този фрагмент е представен обратният логаритъм на софтуера от развойната платка. Първоначално се подготвя един JSON, в който има една стойност “request” с ключ “type”. Този JSON се изпраща по серийната връзка. След това модулът изчаква развойната платка да обработи информацията и да изпрати отговор.

Модулът използва Serial.available и Serial.readString(), за да може да прочете и запише съобщението с цел да го обработва. Използва се функцията deserializeJson(), която преобразува стринг в JSON. След обработката на съобщението от него се извличат необходимите параметри.

На *фиг. 3.6* е представен фрагмент от управляващият софтуер на модула за комуникация. Този фрагмент представлява комуникацията от страната на модулa с Firebase базата данни (наземния сървър).

|  |
| --- |
| Firebase.setDouble(firebaseData, "/Systems/Ddd3/BME680/Pressure", pressure);  Firebase.setDouble(firebaseData, "/Systems/Ddd3/BME680/GAS", gas);  Firebase.setDouble(firebaseData, "/Systems/Ddd3/BME680/Humidity", humidity); |

фиг. 3.6 Фрагмент от управляващият софтуер на модула за комуникация от страната на модулa с Firebase базата данни (наземния сървър)

В този фрагмент се използва функцията setDouble() , която приема инстанция на Firebase, пътят до мястото където трябва да се запише стойността и самата стойност.

На *фиг. 3.7* е представен фрагмент от софтуера на Android приложението. Този фрагмент представлява вземането на стойности от Firebase базата данни (наземния сървър). В този фрагмент от софтуера е реализирано извличането на определена стойност от Firebase базата в реално време. Този фрагмент използва инстанция на Firebase база данни, на която се подава пътят на определената стойност в дървесната структура на базата данни. След което се взима определената стойност, която се преобразува от записания тип в базата данни, в тип стринг. Иззетата стойност се задава на определено TextView, за да може да се визуализира. Ако вземането на стойност не се осъществи, се показва съобщение за грешка.

|  |
| --- |
| database.getReference(**new** StringBuilder("Systems/").append(System).append("/BME680/Temperature").toString()).addValueEventListener(**new** ValueEventListener() {  **@Override**  **public** **void** **onDataChange**(DataSnapshot dataSnapshot) {   String value = dataSnapshot.getValue(Double.class).toString();  textView3.setText(value + " C");  Log.d(TAG, "Value is: " + value);  }   **@Override**  **public** **void** **onCancelled**(DatabaseError error) {  *// Failed to read value*  Log.w(TAG, "Failed to read value.", error.toException());  } }); |

фиг. 3.7 Фрагмент от софтуера на Android приложението, който представя вземането на стойности от Firebase базата данни (наземния сървър)

На *фиг. 3.8* е представен фрагмент от софтуера на Android приложението, който представя записването на стойности във Firebase базата данни (наземния сървър).

|  |
| --- |
| Map<String,Object> user = **new** HashMap<>(); user.put("fName",fullName); user.put("email",email); DatabaseReference userDbRef = FirebaseDatabase.getInstance()  .getReference().child("users")  .child(userID); userDbRef.setValue(user)  .addOnCompleteListener(**new** OnCompleteListener<Void>() {   **@Override**  **public** **void** **onComplete**(@NonNull Task<Void> task) {  Log.d(TAG, "onSuccess: user Profile is created for "+ userID);  }  }).addOnFailureListener(**new** OnFailureListener() {  **@Override**  **public** **void** **onFailure**(@NonNull Exception e) {  Log.d(TAG, "onFailure: " + e.toString());  } }); |

фиг. 3.8 Фрагмент от софтуера на Android приложението, представящ записването на стойности във Firebase базата данни (наземния сървър)

Този фрагмент е иззет от софтуера за регистриране на потребители. За записването на данните на потребителя се използва инстанция на базата данни, на която се подава мястото, на което ще се запишат стойностите. За регистрацията на потребители се използва предоставената от Firebase автентикация. Записването на данните се получава посредством добавянето на данните в HashMap подаването му като аргумент за записване. Ако записването е успешно, се извежда съобщение с автентикационния номер на потребителя.

На *фиг. 3.9* е представен фрагмент от софтуера на Android приложението, който представя опресняването на избрания от нас изглед в приложението.

|  |
| --- |
| refresh = **new** Runnable() {  **public** **void** **run**() {  handler.postDelayed(refresh, 10);  } }; refreshForGraph = **new** Runnable() {  **public** **void** **run**() {  **if**(Recording){   database.getReference(**new** StringBuilder("Systems/").append(System).append("/BME680/Temperature").toString()).addValueEventListener(**new** ValueEventListener() {  **final** String id = database.getReference("users").child(userId).child("systems").child(System).child(Name).child("Graph").child("Temperature").push().getKey();  **@Override**  **public** **void** **onDataChange**(@NonNull DataSnapshot dataSnapshot) {   **int** y=0;  y = dataSnapshot.getValue(Integer.class);   database.getReference("users").child(userId).child("systems").child(System).child(Name).child("Graph").child("Temperature").child(id).setValue(y);    }   **@Override**  **public** **void** **onCancelled**(@NonNull DatabaseError databaseError) {   }  });   database.getReference(**new** StringBuilder("Systems/").append(System).append("/BME680/Pressure").toString()).addValueEventListener(**new** ValueEventListener() {  **final** String id = database.getReference("users").child(userId).child("systems").child(System).child(Name).child("Graph").child("Pressure").push().getKey();  **@Override**  **public** **void** **onDataChange**(@NonNull DataSnapshot dataSnapshot) {   **int** y=0;  y = dataSnapshot.getValue(Integer.class);   database.getReference("users").child(userId).child("systems").child(System).child(Name).child("Graph").child("Pressure").child(id).setValue(y);    }   **@Override**  **public** **void** **onCancelled**(@NonNull DatabaseError databaseError) {   }  });  }   handler.postDelayed(refreshForGraph, 5000);  } }; handler.post(refresh); handler.post(refreshForGraph); |
|  |

фиг. 3.9 Фрагмент от софтуера на Android приложението, който представя опресняването на избрания от нас изглед в приложението

В този фрагмент се използва Runnable. Java.lang.Runnable е интерфейс, който трябва да бъде реализиран от клас, чиито инстанции са предназначени да бъдат изпълнени от нишка. Има два начина за стартиране на нова нишка - подклас нишка и внедряване на Runnable. Няма нужда от под класиране на нишката, когато дадената задача е извършена чрез отменяне само на метода run () на Runnable.

Опресняването се случва в края на метода run () "рекурсивно", като се извиква handler.postDelayed (refresh, 10), за да се изпълни повторно задачата. За стартиране на опреснителя се използва функцията post(). При край на програмата трябва да се извика handler.removeCallbacks(refresh), защото изгледа ще бъде унищожен или ще има риск от изтичане на памет, тъй като има вероятност препратката към изгледа да се държи от екземпляра Runnable, който чака в съобщението на опашката за съобщения да бъде изпълнен.

Във фрагмента се използват два опреснителя. Единият се използва за следене на данните в реално време, затова и времето, на което се повтаря, е доста по-малко отколкото на другия. Използва се този подход, защото данните в базата данни постоянно се променят от модула за комуникация, но приложението ще взима данните само веднъж. Вторият опреснител се използва за междувременното извличане на определени стойности и тяхното записване като координати за визуализиране на данните като линейни графики.

На *фиг. 3.10* е представен фрагмент от софтуера на Android приложението. Този фрагмент представлява използването на MPAndroidchart за графичното визуализиране на записаните полетни параметри. MPAndroidChart е мощна и лесна за използване библиотека с диаграми за Android.

|  |
| --- |
| database.getReference("users").child(userId).child("systems").child(System).child(Name).child("Graph").child("Temperature").addValueEventListener(**new** ValueEventListener() {  **@Override**  **public** **void** **onDataChange**(DataSnapshot dataSnapshot) {  **final** ArrayList<Entry> values = **new** ArrayList<>();  List<String> lst = **new** ArrayList<String>();  **int** i=0;   **for** (DataSnapshot dsp : dataSnapshot.getChildren()) {  lst.add(String.valueOf(dsp.getValue()));  }    **for** (String dsp : lst) {  values.add(**new** Entry(i+=2, Integer.valueOf(dsp)));  }   temperatureSet = **new** LineDataSet(values, "Temperature");  temperatureSet.setColor(Color.YELLOW);   }   **@Override**  **public** **void** **onCancelled**(DatabaseError databaseError) {}  }); |

фиг. 3.10 Фрагмент от софтуера на Android приложението, представящ използването на MPAndroidchart за графичното визуализиране на записаните полетни параметри

Този софтуер взима списък от записаните в базата данни стойности. След създаването на този списък, той се обхожда и се създава нов списък. Стойностите в него са от тип Entry(), който има две стойности, представляващи координатите на точка. За стойностите на ординатата задаваме стойност, която се увеличава с две при образуването на всяка следваща точка, а на абсцисата - стойността от базата данни. Целият този списък от точки се добавя като нова графика в друг списък от графики, като се подават те и името на графиката. Допълнително за се задава и цвят на дадената графика. Списъкът с графики се задава на функцията, която чертае.

# Глава 4. Резултати

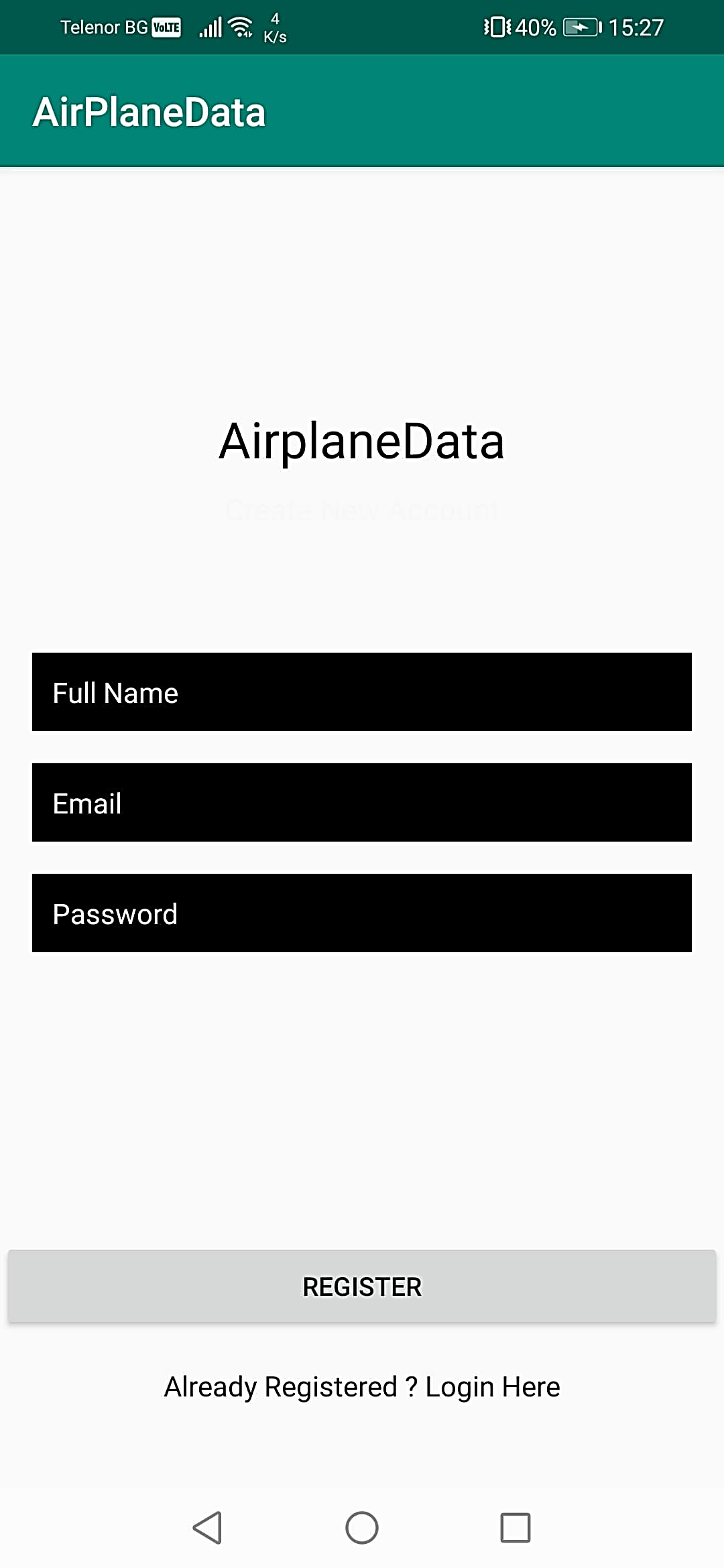
## 4.1 Увод

В тази глава е разгледана реализираната летателна система. Разгледани са Андроид приложението и направените върху системата тестове.

## 4.2 Андроид приложение

#### Регистрация и вход

фиг. 4.1 Изглед за регистрация



фиг. 4.2 Изглед за вход

Регистрацията е първата задължителна стъпка за използването на приложението. Необходимите лични данни, които потребителят трябва да въведе във формуляра, са име, имейл и парола, с 6 или повече символа, както е показано на *фиг. 4.1*.

След като потребителят се е регистрирал, трябва да отиде на изгледа за вход и да въведе своя имейл и вече създадената парола. Изгледът за вход е представен на *фиг. 4.2.* Ако потребителят допусне грешка или пропусне някое от полетата за попълване, приложението изписва предупредителни съобщения на екрана.

#### C:\Users\vanya\Downloads\90531754_159756051834944_4563327083266179072_n.jpgC:\Users\vanya\Downloads\90637922_652345482218665_5997861110911336448_n.jpgМеню

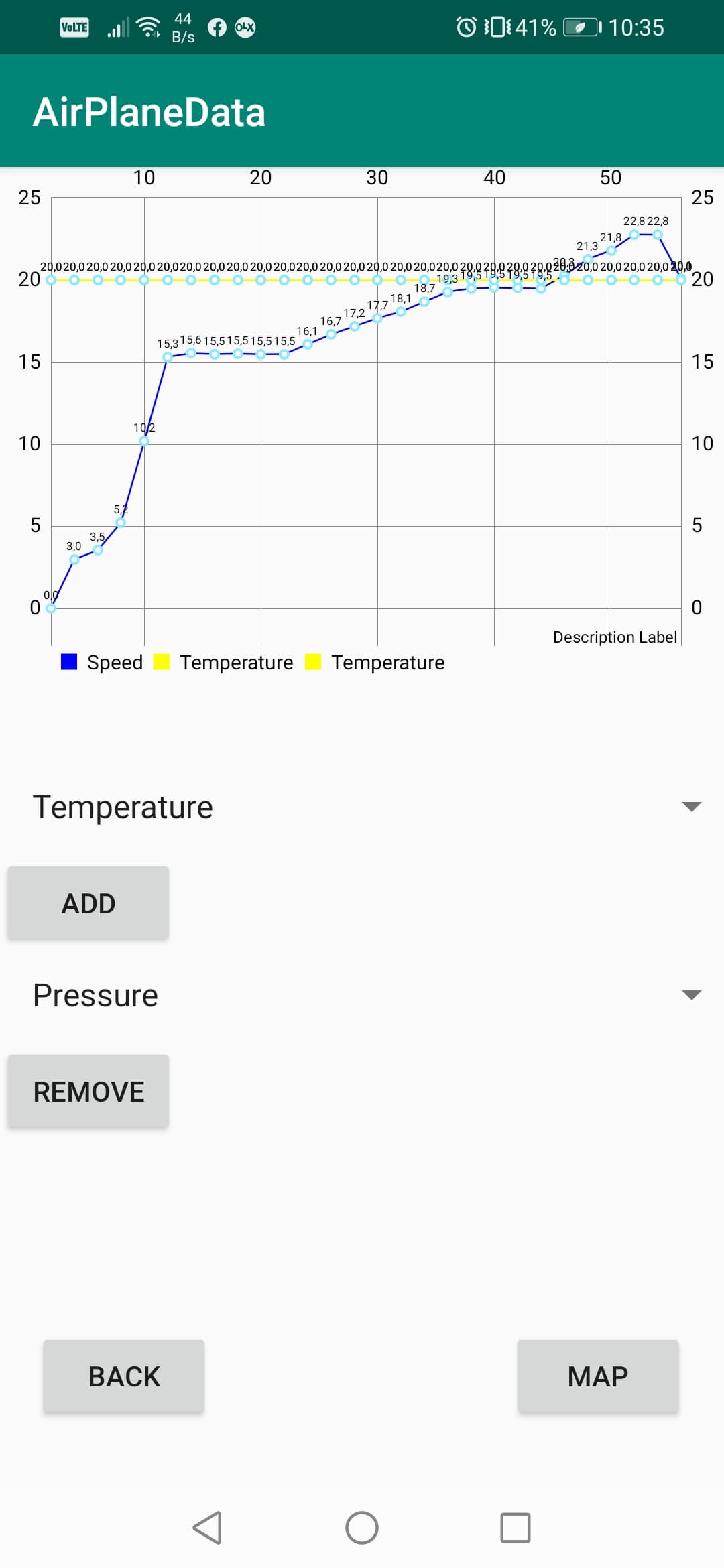
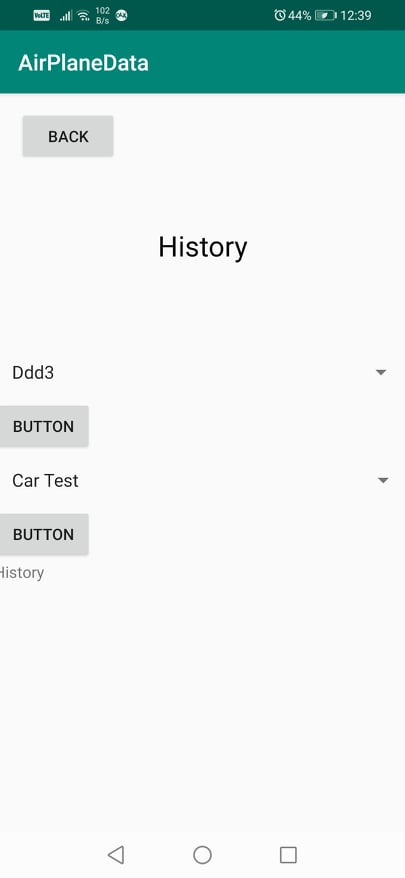
фиг. 4.3 Изглед на меню

фиг. 4.4 Изглед на поле за въвеждане на код

След регистрация или вход на екрана на андроид устройството се появява меню представено на *фиг. 4.3.* В това меню потребителят може да избира между няколко действия.

Първото действие е изход от приложението, който препраща към изгледа за вход. Второто действие е добавяне на летателни системи. Структурата на приложението е изградена така, че един потребител може да притежава повече от една система. За да бъде добавена нова, както е показано на *фиг. 4.4,* потребителят трябва да въведе код, който се получава при закупуването на летателната система. Този код действа като индентификатор. Третото от дейстеията, които се предоставят на потребителя, е разглеждане на историята на полетите.

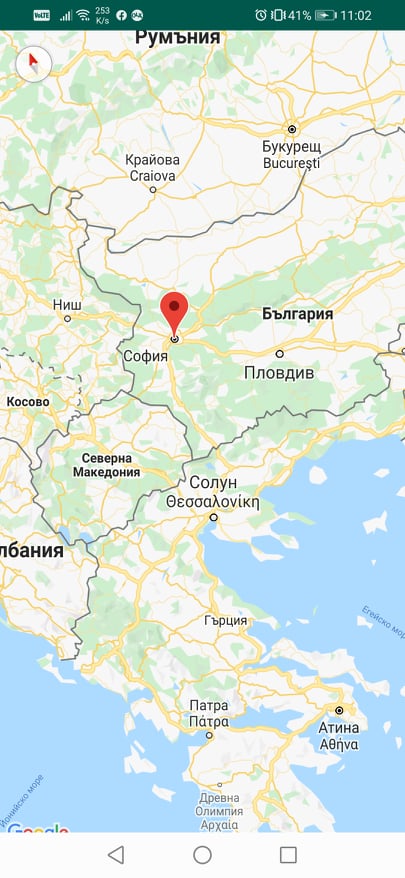
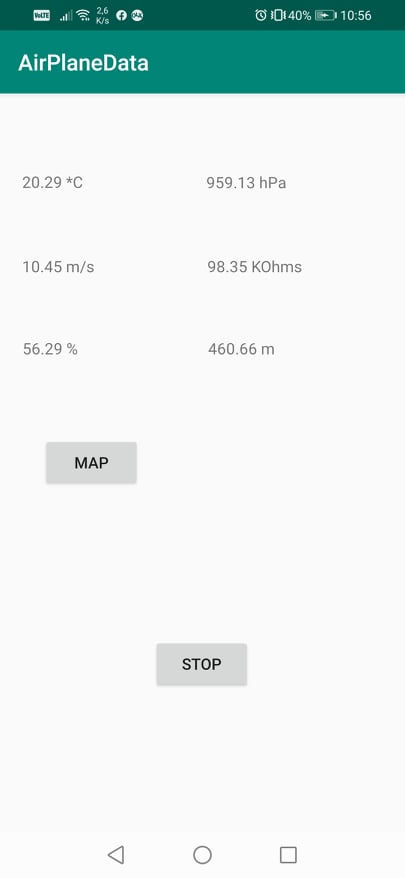
#### История и обработка на данни

**Разглеждането на историята протича по следния начин. Първо се избира летателната система и след това полета за визуализация *(фиг. 4.5).*

фиг. 4.6 Изглед на визуализацията на данни

фиг. 4.5 Изглед на история

След като бъдат избрани полетата, на екрана се появява изгледът за визуализиране на записаните данни *(фиг. 4.6)*. Прозорецът има опция за чертане на графики. Потребителят може да избира данните за кой параметър да се изчертаят. Той може да разглежда повече то един параметър, като графиките се наслагват една върху друга. Потребителят има достъп до последното засечено местоположение на летателната система.

**

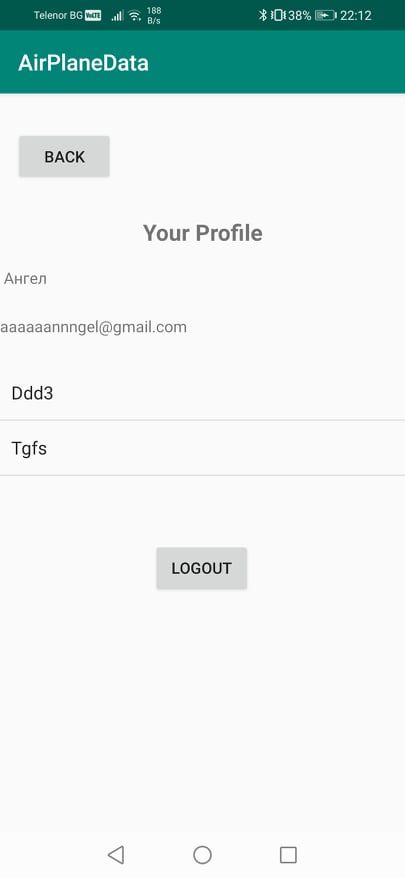
фиг. 4.8 Изглед за местоположение

фиг. 4.7 Изглед за следене на данни в реално време

Следенето на данни в реално време е четвъртото действие, което се предоставя на потребителя. Данните, които са проследени на изгледа на *фиг. 4.7,*  и паралелно през определен период от време се записват в базата данни, са температура, която се измерва в градус целзий (°C), налягане, измерено в хектопаскали (hPa), въздушна скорост, в метри за секунда (m/s), качество на въздуха, измерва се в килооми (KOhms), влажност, в проценти (%), надморска височина, в метри (m). Потребителят има възможността да види и моментното местоположение на летателния апарат *(фиг. 4.8)*.

#### Профил на потребителя

Последната функция, която предоставя приложението *(фиг. 4.9)*. Е, възможността потребителят да разгледа своите данни. На този изглед се показва и списък с всички летателни системи, които той притежава.

**

фиг. 4.9 Изглед на потребителски данни

## 4.3 Експериментална постановка

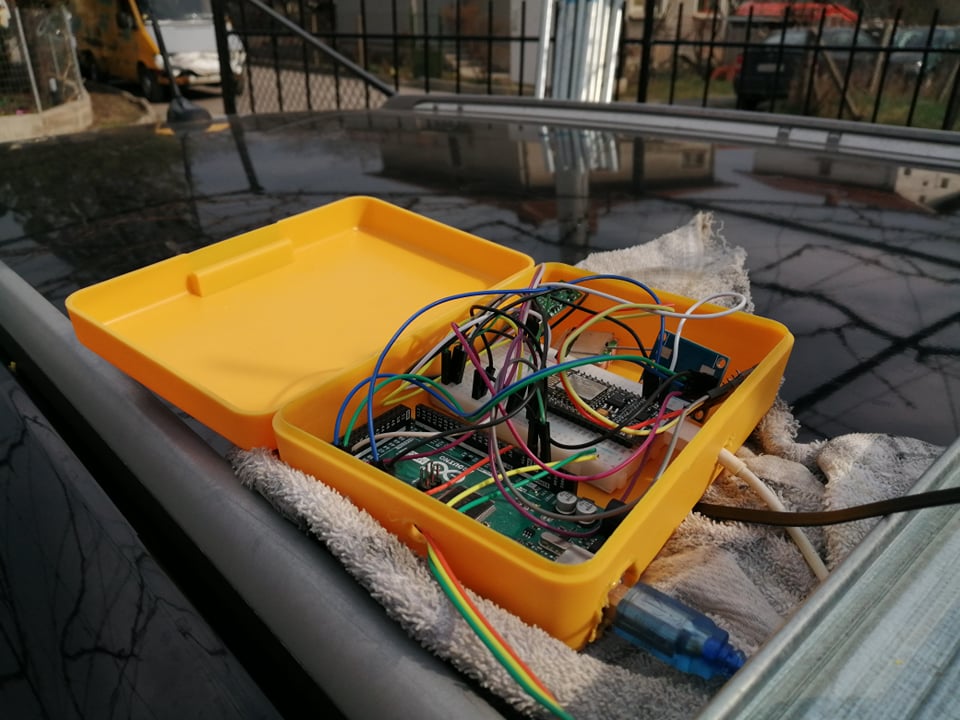
След проведено проучване за тестване на летателни системи, беше установено, че стандартните мобилни тестове включват монтиране на съвкумпността от измервателни уреди върху покрива на лек автомбол, който симулира движението на летателния апарат (Mobile Test Stand for Evaluation of Electric Power Plants for Unmanned Aircraft).

За целите на курсовата работа е наподобена споменатата експериментална постановка с наличните материали *(фиг. 4.10)*. Системата е разположена на височина 1.5 m и на 60 cm от края на предното стъкло. За прикрепването ѝ към автомобила е използан алуминиев профил с дължина 1.76 m *(фиг. 4.11)*. За да не се влияе от въздушния поток на лекия автомобил, сондата на датчика за въздушната скорост се монтира на разстояние от 35 cm от ръба на автомобила *(фиг. 4.12)*. За защита на летателната система от метеорологичните условия тя е затворена в пластмасова кутия, закрепена за алуминиевия профил *(фиг. 4.14)*. За омекотяване на вибрациите и предотвратяване на щети върху автомобила под кутията е поставено парче мек плат.

фиг. 4.10 Общ изглед на автомобила с монтираната постановка



фиг. 4.11 Близък изглед на експерименталната постановка



фиг. 4.13 Летателна система

фиг. 4.12 Питосонда

## 4.4 Експериментални данни

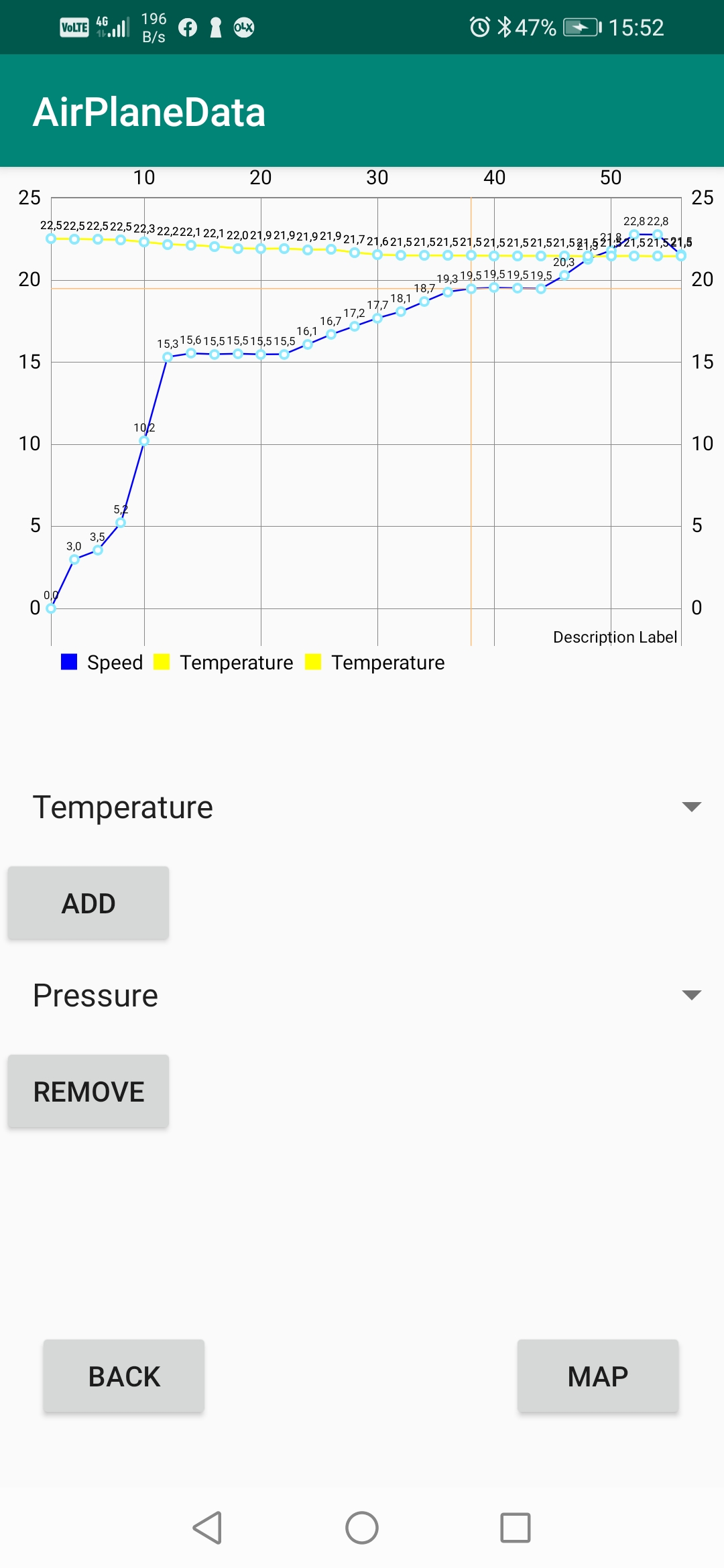
Експериментът е проведен в три повторения на извънградски път в близост до гр. Костинброд. След щателна обработка на данните в *Таблица 4.1* са представени експерименталните данни от третото повторение като най-представителни, макар да няма статистически значими различия между отделните повторения.

Експериментът се провежда при движение на автомобила със скорост от 0 до 60 km/h, като стойностите на параметрите се взимат през равни интервали от време.

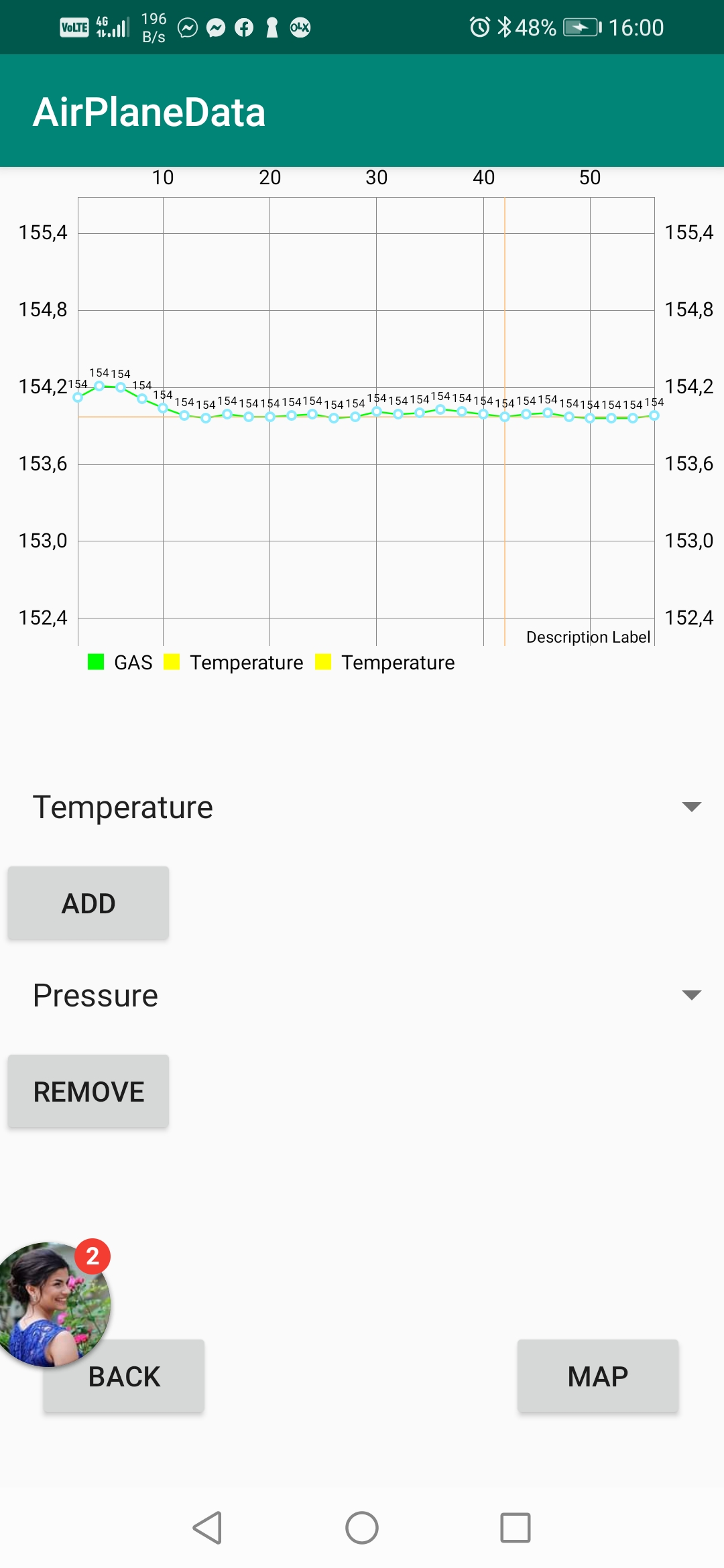
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Скорост на движение на автомобила  (km/h) | Температура  (\*C) | Налягане  (hPa) | Въздушна скорост  (m/s) | Качество на въздуха  (KOhms) | Влажност въздуха  (%) | Надморска височина  (m) |
| 0 - 20 | 22.54 | 953.65 | 0 | 154.12 | 56.28 | 460.66 |
| 22.52 | 953.73 | 3 | 154.21 | 56.28 | 460.66 |
| 22.50 | 953.83 | 3.54 | 154.2 | 56.28 | 460.66 |
| 22.47 | 953.74 | 5.23 | 154.11 | 56.28 | 460.66 |
| 22.34 | 953.65 | 10.2 | 154.04 | 56.28 | 460.66 |
| 22.20 | 953.67 | 15.33 | 153.98 | 56.28 | 460.66 |
| 22.15 | 953.74 | 15.55 | 154.96 | 56.28 | 460.66 |
| 22.07 | 953.88 | 15.5 | 153.99 | 56.28 | 460.66 |
| 21.95 | 953.89 | 15.53 | 153.97 | 56.28 | 460.66 |
| 21.93 | 953.84 | 15.49 | 153.97 | 56.28 | 460.66 |
| 20 - 40 | 21.94 | 953.65 | 15.5 | 153.98 | 56.28 | 460.66 |
| 21.87 | 953.73 | 16.1 | 153.99 | 56.28 | 460.66 |
| 21.89 | 953.83 | 16.7 | 153.96 | 56.28 | 460.66 |
| 21.70 | 953.74 | 17.2 | 153.97 | 56.28 | 460.66 |
| 21.57 | 953.65 | 17.7 | 154.01 | 56.28 | 460.66 |
| 21.52 | 953.67 | 18.1 | 153.99 | 56.28 | 460.66 |
| 21.52 | 953.74 | 18.7 | 154 | 56.28 | 460.66 |
| 21.52 | 953.88 | 19.3 | 154.03 | 56.28 | 460.66 |
| 21.51 | 953.89 | 19.5 | 154.01 | 56.28 | 460.66 |
| 40 - 60 | 21.5 | 953.84 | 19.55 | 153.99 | 56.28 | 460.66 |
| 21.49 | 953.83 | 19.52 | 153.97 | 56.28 | 460.66 |
| 21.49 | 953.74 | 19.5 | 153.99 | 56.28 | 460.66 |
| 21.49 | 953.65 | 20.3 | 154 | 56.28 | 460.66 |
| 21.47 | 953.67 | 21.83 | 153.97 | 56.28 | 460.66 |
| 21.48 | 953.74 | 22.8 | 153.96 | 56.28 | 460.66 |
| 21.49 | 953.88 | 22.79 | 153.96 | 56.28 | 460.66 |
| 21.48 | 953.89 | 21.55 | 153.98 | 56.28 | 460.66 |

Таблица 4.1 Експериментални данни

Използвайки функциите на приложението, данните от *Таблица 4.1* са представени в графичен вид. На *Графика 4.1* са представени табличните данни за въздушна скорост и температура с течение на времето. На *Графики 4.2* и *4.3* са визуализирани данните за качество на въздуха и налягане съответно. Стойностите на параметрите се изчертават спрямо интервали от време, които отговарят на времето за получаване и обработка.



Графика 4.1 Зависимостта на въздушната скорост и температурата от времето

Графика 4.3 Зависимост на налягането от времето

Графика 4.2 Зависимост на качеството на въздуха от времето

# Глава 5. Заключение

В тази дипломна работа бяха разгледани методи за измерване на въздушна скорост, налягане, температура, GPS кординати, честота на въртене и позиция на тяло в пространството, както и методи за безжична комуникация.

Проектираната система използва Wi-Fi за изпащане на събраната информация към наземен сървър (Firebase database). Системата е реализирана на базата Arduino Mega микроконтролер. Данните от наземния сървър се визуализират и обработват от Андроид приложение, реализиранo на програмния език Java. Системата успешно извършва измерване на указаните параметри и изпращане на събраната информация. То представя данните в реално време и историята на полета, за която е нужно разглеждането на данните във вид на графики, използвайки MPAndroidChart. Системата има задоволителна точност за целите на дипломната работа.

По време на реализацията на проекта възникнаха доста въпроси и трудности, както по софтуера, така и по хардуера. За изчерпателното завършване на такъв проект са необходими теоретични знания и умения в двете технически области.

Едни от възможните оптимизации на летателната сиситема са осигуряването на по-голям набор от датчици и сензори за разширяване функционалността и намаляването на размера на системата с цел компактност.

# Източници

1.<https://bg.wikipedia.org/wiki/%D0%9B%D0%B5%D1%82%D0%B0%D1%82%D0%B5%D0%BB%D0%B5%D0%BD_%D0%B0%D0%BF%D0%B0%D1%80%D0%B0%D1%82>

2. <https://en.wikipedia.org/wiki/Pitot_tube>

3. <https://churilovocity.ru/bg/chto-takoe-ch-rnye-yashchiki-samol-ta---ustroistvo-opisanie-i-interesnye-fakty-chto/>

4. [https://help.superhosting.bg/sql-nosql-databases.htmlhttps://bg.wikipedia.org/wiki/%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82/](https://help.superhosting.bg/sql-nosql-databases.htmlhttps:/bg.wikipedia.org/wiki/%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82/)

5. <https://en.wikipedia.org/wiki/Wi-Fi>

6. <https://en.wikipedia.org/wiki/Bluetooth>

7. <https://softuni.bg/blog/which-programming-languages-are-best-for-android-development>

8. <https://www.pololu.com/product/2738>

9. <https://opensource.com/resources/raspberry-pi>

10. <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>

11. <https://www.engineering-review.bg/bg/senzori-za-dvizhenie-i-polozhenie-v-elektronikata/2/3917/>

12. [https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors-bme680/https://www.kaldata.com/it-%D0%BD%D0%BE%D0%B2%D0%B8%D0%BD%D0%B8/%D0%BA%D0%BE%D0%B9-%D0%B5%D0%B7%D0%B8%D0%BA-%D0%B7%D0%B0-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5-%D0%B5-%D0%BD%D0%B0%D0%B9-%D0%BF%D0%BE%D0%B4%D1%85%D0%BE%D0%B4%D1%8F-257618.html](https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors-bme680/https:/www.kaldata.com/it-%D0%BD%D0%BE%D0%B2%D0%B8%D0%BD%D0%B8/%D0%BA%D0%BE%D0%B9-%D0%B5%D0%B7%D0%B8%D0%BA-%D0%B7%D0%B0-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5-%D0%B5-%D0%BD%D0%B0%D0%B9-%D0%BF%D0%BE%D0%B4%D1%85%D0%BE%D0%B4%D1%8F-257618.html)

13. <https://erelement.com/wireless/nodemcu?zenid=jttegj7ep715fcdotgf517noi4>

14. <https://erelement.com/arduino-mega-2560-r3?zenid=jttegj7ep715fcdotgf517noi4>

15. <https://erelement.com/sensors/bme680?zenid=jttegj7ep715fcdotgf517noi4>

16. <https://erelement.com/sensors/minimu-9-v5?zenid=jttegj7ep715fcdotgf517noi4>

17. <https://ardupilot.org/plane/docs/airspeed.html>

18. <https://en.wikipedia.org/wiki/LoRa>

19. <https://en.wikipedia.org/wiki/Global_Positioning_System>

20.<https://en.wikipedia.org/wiki/Hall_effect>

21.<https://bg.wikipedia.org/wiki/%D0%9D%D0%B0%D0%BB%D1%8F%D0%B3%D0%B0%D0%BD%D0%B5>

22.<https://bg.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D1%83%D1%80%D0%B0>

23. <https://www.electroschematics.com/neo-6m-gps-module/>

24. <https://www.robotev.com/product_info.php?products_id=445>

25. <https://www.pololu.com/product/2199>

26. <https://core-electronics.com.au/hall-effect-sensor-us1881.html>

27. <https://makersportal.com/blog/2019/02/06/arduino-pitot-tube-wind-speed-theory-and-experiment>

28.<https://www.nakra.eu/%D0%BF%D0%BE%D0%BB%D0%B5%D0%B7%D0%BD%D0%BE/%D0%B8%D0%B7%D0%BC%D0%B5%D1%80%D0%B2%D0%B0%D0%BD%D0%B5-%D0%BD%D0%B0-%D0%BD%D0%B0%D0%BB%D1%8F%D0%B3%D0%B0%D0%BD%D0%B5>

29.[http://timetable.swu.bg/lecture/rkraleva/ZimenSem/MobApp/ypr9/mobApp\_ex9.pdfhttps://bg.wikipedia.org/wiki/%D0%95%D1%84%D0%B5%D0%BA%D1%82\_%D0%BD%D0%B0\_%D0%A5%D0%BE%D0%BB](http://timetable.swu.bg/lecture/rkraleva/ZimenSem/MobApp/ypr9/mobApp_ex9.pdfhttps:/bg.wikipedia.org/wiki/%D0%95%D1%84%D0%B5%D0%BA%D1%82_%D0%BD%D0%B0_%D0%A5%D0%BE%D0%BB)

30. <https://wiki.eprolabs.com/index.php?title=GPS_Module_NEO6MV2>

31. <https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>

32.<https://bg.wikipedia.org/wiki/%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82>

33. <https://en.wikipedia.org/wiki/Airspeed>

34. <https://www.totalconn.com/tinycontrol/556-acs709-current-sensor-carrier-75a-to-75a.html>