

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií

Evidenčné číslo: FIIT-104199-72314

Senzorový systém pre Android platformy

Bakalárska práca

Študijný program: internetové technológie

Študijný odbor: 9.2.4. počítačové inžinierstvo

Školiace pracovisko: Ústav počítačového inžinierstva a aplikovanej informatiky

Vedúci záverečnej práce: Ing. Peter Pištek, PhD.

Bratislava 2016

Matúš Pohančénik

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

ZADANIE BAKALÁRSKEHO PROJEKTU

Meno študenta: **Pohančeník Matúš**
Študijný odbor: Počítačové inžinierstvo
Študijný program: Počítačové a komunikačné systémy a siete
Názov projektu: **Senzorový systém pre Android platformy**

Zadanie:

Zariadenia Funtoro umožňujú pripájať cez rôzne komunikačné rozhrania štandardné typy periférnych zariadení. Zariadenia používajú operačný systém Android. Analyzujte možnosti použitia senzorov pre tieto zariadenia, ktoré by mohli komunikovať napríklad cez Bluetooth rozhranie. Navrhňte riešenie pozostávajúce zo senzorového systému a Android aplikácie tak, aby získané údaje zo senzorov vedela aplikácia spracovať v zariadení a zároveň, aby cez vhodne navrhnuté rozhranie poskytovala údaje aj ďalším aplikáciám. Riešenie implementujte pre zariadenia Funtoro alebo iné dostupné zariadenie používajúce operačný systém Android. Výslednú aplikáciu overte v laboratóriu FIIT-Molpir alebo v reálnych podmienkach.

Práca musí obsahovať:

- Anotáciu v slovenskom a anglickom jazyku
- Analýzu problému
- Opis riešenia
- Zhodnotenie
- Technickú dokumentáciu
- Zoznam použitej literatúry
- Elektronické médium obsahujúce vytvorený produkt spolu s dokumentáciou

Miesto vypracovania: Ústav počítačových systémov a sietí, FIIT STU, Bratislava
Vedúci projektu: Ing. Peter Pišteck, PhD.

Termín odovzdania práce v zimnom semestri 9.decembra 2015
Termín odovzdania práce v letnom semestri 10.mája 2016

Bratislava 21.09.2015


Ing. Katarína Jelemenská, PhD.
poverená vedením ústavu



Anotácia

Študijný program: Internetové technológie

Autor: Matúš Pohančeník

Bakalárska práca: Senzorový systém pre Android platformy

Vedúci bakalárskej práce: Ing. Peter Pišteck, PhD.

Máj, 2016

V práci sa zaoberám vytvorením senzorového systému pre Android platformy, ktorý rozširuje možnosti analyzovania stavu prostredia a podnetov z neho.

Bakalárska práca obsahuje analýzu senzorov v Android zariadeniach a možnosti pripájania externých senzorových systémov cez komunikačné rozhrania. Obsah analýzy približuje zariadenia, ktoré by mohli byť využité pri tvorbe tohto senzorového systému a opisuje možnosti pripájania senzorov k týmto zariadeniam.

Senzorový systém je konštruovaný pre využitie aj v extrémnych podmienkach na meranie veličín prostredia. Podobne ako obdobné meteorologické stanice, ktoré sú bežne dostupné na trhu umožňuje sledovanie teploty, vlhkosti a atmosférického tlaku. Môj projekt prináša rozšírenie v podobe merania zrýchlenia, ktoré bude zaujímavé sledovať pri vypustení senzorového systému v module stratosférického balóna, pre ktorý je tento systém vyvíjaný.

Annotation

Degree Course: Internet Technologies

Author: Matúš Pohančéník

Bachelor Thesis: Sensoric system for Android platforms

Supervisor: Ing. Peter Pišteň, PhD.

May 2016

In my project, I deal with creation of sensoric system for Android platforms that expands possibilities of analyzing of environmental conditions and it's impulses.

Bachelors thesis consists of analysis of sensors in Android devices and options of connecting external sensoric systems through communication interfaces. The content of analysis takes a closer look on devices that could be used for production of this sensoric system and describes possibilities of connecting a sensor to these devices.

Sensoric system is designed for usage even in extreme conditions to measure values of an environment. Similar to parallel weather stations that are normally available on market it enables monitoring of temperature, humidity and atmospheric pressure. My project provides enhancement in form of measurement of acceleration which ought to be interesting to monitor while launching a balloon with a sensoric system in it's module for which this system is developed.

Čestné prehlásenie

Podpísaný Matúš Pohančeník čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne podľa odborného vedenia môjho vedúceho a uvedenej literatúry. Som si vedomý prípadných dôsledkov, ak sú tieto informácie nepravdivé.

V Bratislave, máj 2016



Matúš Pohančeník

Pod'akovanie

Ďakujem vedúcemu bakalárskej práce, Ing. Petrovi Pištekov PhD. za jeho rady a postrehy a za čas strávený pri konzultáciách. Rovnako by som sa rád poďakoval kolegovi Bc. Michalovi Valičekovi za jeho rady pri návrhu a implementácii Arduino aplikácie.

OBSAH

ZOZNAM OBRÁZKOV	1
ZOZNAM TABULIEK	2
ZOZNAM POUŽITÝCH POJMOV A SKRATIEK	3
1 ÚVOD	5
2 ANALÝZA	6
2.1 SENZOROVÉ SYSTÉMY PRE ANDROID PLATFORMY	6
2.2 TECHNOLOGIE Z-WAVE, ZIGBEE A BLUETOOTH	7
2.2.1 Z-Wave	7
2.2.2 ZigBee	8
2.2.3 Bluetooth	9
2.2.4 Zhodnotenie	10
2.3 EXISTUJÚCE SENZOROVÉ SYSTÉMY	11
2.3.1 Netatmo	11
2.3.2 SensoDuino	12
2.4 FUNTURO, ARDUINO A RASPBERRY PI PLATFORMY	13
2.4.1 Funtoro HD Infotainment system (1 DIN)	13
2.4.2 Arduino	13
2.4.3 Raspberry Pi	15
2.4.4 Zhodnotenie	15
Tabuľka 1. Porovnanie najdôležitejších parametrov.	16
2.5 BLUETOOTH MODUL, SENZORY A ICH PRIPOJENIE K ARDUINO	16
2.5.1 SPI	17
2.5.2 1-Wire	18
2.5.3 UART	19
2.5.4 I ² C	20
2.5.5 Realizácia pripojenia	22
2.6 ZHODNOTENIE ANALÝZY	22
3 OPIS RIEŠENIA	24
3.1 ŠPECIFIKÁCIA POŽIADAVIEK	24
3.1.1 Špecifikácia funkcionálnych požiadaviek	24
3.1.2 Špecifikácia nefunkcionálnych požiadaviek	24
3.2 ARCHITEKTONICKÝ NÁVRH	25
3.3 PRÍPADY POUŽITIA ANDROID APLIKÁCIE	26
3.4 VÝBER IMPLEMENTAČNÉHO PROSTREDIA	27
3.4.1 Arduino aplikácia	27
3.4.2 Android aplikácia	27

3.5	NÁVRH RIEŠENIA	27
3.5.1	Štruktúra systému	28
3.5.2	Arduino aplikácia	28
3.5.3	Android aplikácia	29
3.5.4	Používateľské rozhranie Android aplikácie.....	31
4	IMPLEMENTÁCIA	33
4.1	ARDUINO APLIKÁCIA	33
4.1.1	Zoznam využívaných knižníc	33
4.1.2	Funkcie programu	34
4.1.3	Výpočet priemeru	35
4.2	ANDROID APLIKÁCIA	36
4.2.1	Povolenia aplikácie.....	36
4.2.2	Rozdelenie do balíkov	37
4.2.3	Grafická časť	39
4.3	OVERENIE RIEŠENIA.....	40
4.3.1	Záver testovania.....	43
5	ZÁVER.....	44
5.1	MOŽNÉ VYLEPŠENIE	45
6	TECHNICKÁ DOKUMENTÁCIA	46
6.1	DOKUMENTÁCIA K ARDUINO APLIKÁCII	46
6.1.1	Pripojenie komponentov k doske Arduino	50
6.2	POUŽÍVATEĽSKÁ PRÍRUČKA K ARDUINO APLIKÁCII	51
6.2.1	Inštalačná príručka.....	51
6.3	DOKUMENTÁCIA K ANDROID APLIKÁCII	52
6.4	POUŽÍVATEĽSKÁ PRÍRUČKA K ANDROID APLIKÁCII	57
6.4.1	Systémové požiadavky	57
6.4.2	Inštalačná príručka.....	57
6.4.3	Opis používateľského rozhrania	57
7	ZDROJE	62

Zoznam obrázkov

Obrázok 1	Porovnanie rýchlostí prenosu technológií Z-Wave, ZigBee a Bluetooth.....	11
Obrázok 2	Počet zariadení pri technológiách Z-Wave, ZigBee a Bluetooth.	11
Obrázok 3	Doska Arduino Mega2560.	15
Obrázok 4	Zapojenie zariadení v technológii SPI.....	18
Obrázok 5	Zjednodušené zobrazenie UART rozhrania.	19
Obrázok 6	Zobrazenie prenosu adresy a údajov protokolom I2C.....	22
Obrázok 7	Architektonický návrh senzorového systému.....	25
Obrázok 8	Prípady použitia.....	26
Obrázok 9	Diagram Arduino aplikácie.	29
Obrázok 10	Priebeh vytvorenia spojenia a záznamu údajov.....	31
Obrázok 11	Používateľské rozhranie počas prebiehajúceho merania.	32
Obrázok 12	Hierarchia balíkov tried Android aplikácie.	37
Obrázok 13	Tabuľka pre uchovávanie údajov o meraní.	38
Obrázok 14	Hierarchia aplikácie s dôrazom na poukázanie grafickej časti.....	40
Obrázok 15	Priebeh teploty počas testu.	41
Obrázok 16	Priebeh vlhkosti počas testu.	41
Obrázok 17	Priebeh atmosférického tlaku počas testu.....	42
Obrázok 18	Priebeh zrýchlenia na všetkých osiach.	42
Obrázok 19	Zapojenie senzoru BMP180 a termistoru KTY81-120.	50
Obrázok 20	Zapojenie Bluetooth modulu HM-10.	51
Obrázok 21	Hierarchia tried zaradených do balíkov.....	52
Obrázok 22	Základná obrazovka aplikácie.	58
Obrázok 23	Inicializácia komunikácie s HM-10.....	59
Obrázok 24	Obrazovka aplikácia počas prijímania údajov.....	59
Obrázok 25	Zobrazenie všetkých uložených meraní.	60
Obrázok 26	Zobrazenie nameraných údajov.....	61

Zoznam tabuliek

Tabuľka 1 Porovnanie najdôležitejších parametrov.....	16
--	----

Zoznam použitých pojmov a skratiek

AC-DC(Alternating Current-Direct Current) - Striedavý a priamy prúd

API (Application programming interface) - časť knižnice, ktorá definuje interakciu s externým kódom

Beacons - Paket používaný na prebúdzanie zariadení

Bootloader - Počítačový kód, ktorý sa spúšťa ešte pred operačným systémom

CO₂ - Oxid uhličitý

CMOS/TTL(Complementary Metal Oxide Semiconductor/Transistor Transistor Logic)

CRC(Cyclic redundancy check) - Kód kontrolujúci správnosť doručeného paketu

CSV(Command separated value) - Špeciálny formát súboru

GATT(Generic attribute profile) - Definuje prenos údajov cez Attribute Protocol

GFSK(Gaussian Frequency Shift Keying) - Typ frekvenčnej modulácie

GHz(Gigahertz) - Jednotka frekvencie

GPU(Graphics processing unit) - Grafický procesor

HCI(Host Controller interface) -Rozhranie zabezpečujúce komunikáciu medzi dvoma časťami protokolu Bluetooth

HDMI(High-Definition Multimedia Interface) - Rozhranie pre prenos audiovizuálneho obsahu

I²C (Inter-Integrated Circuit) - Komunikačný protokol

ID - Unikátny identifikátor

IDE(Integrated Development Environment) - Vývojové prostredie

IDII(Interaction Design Institute Ivrea) - Inštitút v Talianskom meste Ivrea

IEEE(Institute of Electrical and Electronics Engineers) - Svetová asociácia pre vývoj technológií

IFS(Inter frame space) - Prestávka medzi posielaním dvoch paketov

I/O(Input/Output) - Vstupno-výstupné

IoT(Internet of Things) - Sieť zariadení, ktoré medzi sebou komunikujú

ISM(industrial, scientific and medical) - Pásmo rádiového vysielania v priemyselnom, vedeckom a zdravotníckom odbore

Kbps(kilobit per second) - Kilobitov za sekundu

LE(Low Energy) - Označenie pre Bluetooth verziu 4.0

MAC adresa (Media access control address) - Unikátna fyzická adresa zariadenia

mAh(Milliampere hour) - Jednotka odberu

Master - Nadradené zariadenie, ktoré riadi komunikáciu na zbernici v modeli "master/slave"

MHz(Megahertz) - Jednotka frekvencie

MISO(Master in/Slave out) - Linka v SPI komunikácia, prenáša údaje zo slave do master zariadenia

MOSI(Master out/Slave in) -Linka v SPI komunikácia, prenáša údaje z master do slave zariadenia

Paket - Blok prenášaných údajov v počítačovej sieti

PAN(Personal Area Network) - Počítačová sieť na prenos údajov

Point-to-point - Komunikačné spojenie medzi dvoma zariadeniami

PWM(Pulse-width modulation) -Impulzová šírková moduláci

RAM(Random access memory) - Počítačová pamäť s priamym prístupom

RS-232 - Štandard pre sériovú komunikáciu

SCL(Serial clock) - Hodinová zbernica

SDA(Serial data) - Zbernica na prenos údajov

Slave - Podradené zariadenie v modeli "master/slave", ktoré je pripojené na zbernicu a komunikuje, no nemôže ju riadiť

SOC(System on chip) - Inteligentný obvod, ktorý integruje všetky komponenty počítača

UARTs(Universal Asynchronous Receiver/Transmitter) - Zariadenie, pre sériovú komunikáciu

USB(Universal Serial Bus) - Sériová zbernica typu master-slave

V (Volt) - Jednotka napätia

Wi-Fi(Wireless Fidelity) - Bezdrôtová lokálna sieť

XML (eXtensible Markup Language) - Štruktúrovaný formát súborov

1 Úvod

Vývoj a začleňovanie moderných technológií do každodenného života vôbec nie je novinkou. Odkedy sa počítače presunuli z akademickej pôdy aj medzi verejnosť, vývojári prinášajú množstvo inovácií, ktoré sa stávajú neodmysliteľnou súčasťou života človeka. Dnes sú azda najdôležitejšími zariadeniami inteligentné mobilné telefóny, nazývané smartfóny. Tie sú už niekoľko rokov súčasťou nášho každodenného života a podporujú množstvo funkcií, ktoré nám uľahčujú a spríjemňujú život. Jednou z nich je aj možnosť neustáleho prehľadu prostredníctvom internetu, napríklad o vývoji bankových kurzov.

Pretože vývojári sa snažia prinášať stále viac možností pre využitie smartfónov, prišli aj s myšlienkou IoT. Jedná sa o implementáciu softvéru do množstva zariadení a senzorov, ktoré sú spájané do sietí. Vďaka takýmto sieťam môžeme riadiť veľké množstvo akcií, napríklad v dome priamo zo svojho smartfónu. Môžeme nastaviť kúrenie z práce, vypnúť elektrický spotrebič alebo rozsvietiť svetlo a podobne. Tieto funkcie majú na starosti aktivačné zariadenia, ktoré kontrolujú prívod elektrickej energie a tak zabezpečujú ich zapnutie, alebo vypnutie. Súčasťou týchto sietí sú však aj senzory, ktoré nám umožňujú ovládať správanie zariadenia na základe nejakých vonkajších podnetov. Výborným príkladom je rozsvietenie svetla na základe pohybového senzoru. Prítomnosť takýchto senzorov v sieti, ktorá zdieľa údaje zo senzorov prostredníctvom niektorého z komunikačných rozhraní nám prináša neustály prehľad o stave prostredia. Ak napríklad chceme mať vždy aktuálny prehľad o počasi a detailoch ovzdušia, stačí umiestniť niektorý so senzorových modulov vonku a spárovať ho s našim smartfónom, napríklad cez Bluetooth alebo Wi-Fi. Namiesto staromódneho vonkajšieho teplomera máme jednoduchým otvorením aplikácie detailné informácie o vonkajších podmienkach.

Vzhľadom na rozmanitosť využitia senzorových systémov sme sa rozhodli zamerať na možnosť vytvorenia kompaktného pohyblivého senzorového systému. V súčasnosti môžeme vďaka funkciám nášho smartfónu sledovať trasu, kadiaľ sme sa pohybovali. Vďaka pohyblivému senzorovému systému, ktorý si pripojíme k nášmu zariadeniu by sme však mohli získavať omnoho viac informácií. Takéto výhody by mohli prispieť k zaznamenávaniu podmienok prostredia pri rôznych expedíciách do extrémnych podmienok. Napríklad sledovanie radiácie pri objavovaní nových jaskýň, klimatických podmienkach pri turistických expedíciách, alebo vojenských zásahoch.

2 Analýza

Kapitola analyzuje oblasť špecifikovanú podľa prideleného zadania. Zaoberá sa senzormi v kooperácii s operačným systémom Android a opisuje najvyužívanejšie technológie súčasnosti. Približuje niektoré existujúce riešenia sensorových systémov pre Android. V ďalšej časti analyzuje možnosti vytvorenia sensorového systému vhodného pre realizáciu projektu stratosférického balóna.

2.1 Senzorové systémy pre Android platformy

Väčšina zariadení s operačným systémom Android má zabudované senzory, ktoré merajú z pohľadu zariadenia najdôležitejšie veličiny prostredia. Delíme ich do dvoch základných skupín na hardvérové a softvérové. Hardvérové senzory sú fyzickými komponentmi zariadenia a poskytujú priamo zaznamenané údaje. Softvérové senzory nie sú fyzicky umiestnené v zariadení, ale využívajú kombináciu údajov z iných hardvérových senzorov. Takýmto príkladom sú napríklad senzor gravitácie, alebo senzor lineárneho zrýchlenia.

Senzory zaznamenané údaje poskytujú aplikáciám v neupravenej podobe a tie ich následne spracúvajú podľa svojich potrieb. Po analyzovaní údajov dokážu poskytnúť používateľovi informácie o trojdimenzionálnom pohybe, polohe zariadenia, no a v niektorých prípadoch aj podmienkach ovzdušia. Spôsob využitia údajov zo senzoru závisí od typu údajov, ktoré zaznamenáva. Napríklad herné aplikácie najčastejšie využívajú údaje zo senzorov, ktoré merajú zrýchlenie a polohu zariadenia, vďaka čomu dokážu identifikovať napr. trasenie, nakláňanie alebo rotovanie zariadenia. Aplikácie sledujúce podmienky vonkajšieho prostredia využívajú údaje najmä z teplotných senzorov, prípadne senzorov vlhkosti [1].

Android platformy podporujú tri kategória senzorov:

1. Senzory pohybu

Niekoľko senzorov, ktoré umožňujú sledovať pohyb zariadenia v troch osiach. Senzory umiestnené zvyčajne hardvérovo sú gyroskop a akcelerometer. Ostatné senzory sa vyskytujú v oboch formách v závislosti od zariadenia. Sú to senzory gravitácie, lineárneho zrýchlenia a rotácie[2].

2. Senzory prostredia

Základným reprezentantom tejto kategórie je senzor osvetlenia, ktorý ako jediný je súčasťou väčšiny zariadení s operačným systémom Android a slúži na určovanie osvetlenia displeja. Zvyšné senzory prostredia sú podporované platformou Android, no na väčšine zariadení ich nenájdeme. Dôvodom je najmä fakt, že všetky musia byť umiestnené hardvérovo v zariadení. Sú to senzory teploty, tlaku a vlhkosti [3].

3. Senory pozície

Dva senzory umožňujúce sledovanie pozície zariadenia sú senzor geomagnetického poľa a senzor orientácie. Android platforma poskytuje aj senzor vzdialenosti, ktorý slúži najmä na identifikovanie vzdialenosti telefónu od tváre človeka pri telefonovaní. Senzor geomagnetického poľa aj senzor vzdialenosti je umiestnený hardvérový v zariadení. Senzor orientácie je softvérový a využíva údaje zo senzoru geomagnetického poľa a akcelerometra [4].

Senzory, ktoré obsahuje zariadenie sa môžu líšiť podľa výrobcov a s výnimkou senzorov prostredia sa zameriavajú na stav zariadenia. Súčasný trh však ponúka pripojenie periférnych zariadení, ktoré dokážu cez rôzne typy rozhraní prenášať zaznamenávané dáta do aplikácie v telefóne alebo tablete. Práve senzory prostredia sú často pripájané cez rôzne aplikácie k zariadeniam Android, napríklad vo forme domácich meteorologických staníc. Často sa využívajú aj bezpečnostné systémy, ktorými môže používateľ sledovať napríklad zatvorené okná, zamknuté dvere alebo pohyb v budove.

Android aplikácie komerčných senzorových systémov najčastejšie komunikujú so senzormi pripojením k rovnakým internetovým serverom. Tento spôsob prenosu údajov podporujú najmä senzory, ktoré vytvárajú tzv. inteligentnú domácnosť. Princípy takéhoto získavania údajov do Android aplikácie naznačujú úmysel sledovania získaných informácií na diaľku. Minimálne jeden prvok siete musí byť teda inštalovaný ako koordinátor a byť pripojený k sieti internet. Získané dáta sú tak uložené na serveri a používateľ aplikácie sa nemusí starať o poskytovanie dostatočne veľkej pamäte.

Projekty senzorových systémov, pri ktorých nie je možné spoliehať sa na pripojenie k internetovým serverom využívajú pre komunikáciu najmä technológiu postavenú na štandarde IEEE 802.15., nazývanú Bluetooth. Vďaka tomu môžu so senzorovým systémom komunikovať aj bez pripojenia na internet, no len na krátku vzdialenosť.

2.2 Technológie Z-Wave, ZigBee a Bluetooth

Nasledujúce podkapitoly opisujú v súčasnosti najvyužívanejšie technológie pre vytváranie inteligentných domácností.

2.2.1 Z-Wave

Je to komunikačný protokol založený na rádio frekvenčných komunikačných technológiách. Poskytuje nízkoenergetický spôsob komunikácie v sieti, ktorá môže obsahovať až do 232 prvkov. Zariadenia v sieti pritom umožňujú preposielanie údajov a tak podporujú aj efektívnejšie zapojenia siete. To vedie k rozšíreniu rozsahu pokrytia, vďaka možnosti komunikácie vzdialených zariadení sprostredkované cez iné zariadenia.

Na rozdiel od podobných bezdrôtových technológií ako sú Wi-Fi a iné založené na IEEE 802.11, je zameraná na prenos malého objemu údajov. Dosahuje preto rýchlosť prenosu

iba do 100kbit/s [5]. Frekvencia, na ktorej pracuje Z-Wave, sa pohybuje v bezlicenčnom ISM pásme okolo úrovne 900 MHz v závislosti od územia, na ktorom sa technológia používa. Toto členenie reaguje na rozdielne predpisy a normy v Európe, Spojených štátoch amerických a Ázii. Vďaka nižšej frekvencii nedochádza k rušeniu s často využívanými technológiami Wi-Fi, Bluetooth a iné, ktoré komunikujú na preplnenej frekvencii 2.4 GHz.

Každá sieť zariadení obsahuje sieťové ID a všetky zariadenia v sieti majú aj vlastné ID. Základným prvkom siete je primárny ovládač, ten okrem iného umožňuje zdieľanie získaných dát prostredníctvom pripojenia k sieti internet. Môže obsahovať aj sekundárne ovládače, ktoré slúžia rovnako ako primárny ovládač na smerovanie prenášaných údajov a zabezpečenie bezpečnosti v sieti. Smerovanie môžu zabezpečovať aj ostatné zariadenia v sieti, no musia byť vždy aktívne a počúvať [6].

Vďaka svojej nízkej spotrebe energie tvorí jednu z najvyužívanejších technológií pri vytváraní sietí senzorov a iných zariadení, ktoré umožňujú ovládanie na diaľku. V súčasnosti už viac ako 300 certifikovaných spoločností vytvára zariadenia podporujúce túto technológiu.

2.2.2 ZigBee

Technológia je založená na štandarde IEEE 802.15.4, ktorý definuje fyzickú a linkovú MAC vrstvu. Nad nimi ZigBee definuje aplikačnú vrstvu, ktorá umožňuje kompatibilitu zariadení z rôznych spoločností. Táto vrstva tvorí rozhodujúcu nadstavbu oproti štandardnej špecifikácii 802.15.4 a zabezpečuje aj bezpečnosť v sieti. Priama komunikácia medzi dvoma uzlami siete je možná na vzdialenosť do 70 metrov, no uzly môžu komunikovať sprostredkované cez tretí uzol a vzdialenosť tak môže byť rozšírená. Aplikácie založené na štandarde IEEE 802.15.4 sú zväčša monitorovacie alebo ovládacie aplikácie, zamerané na nízko energetické zariadenia, ktoré prenášajú malé množstvá dát. Práve nízka spotreba je kľúčová požiadavka na technológie zabezpečujúce komunikáciu medzi senzormi.

Systém pracuje v troch frekvenčných pásmach 2,4 GHz, 915 MHz pre Severnú Ameriku a 868 MHz pre Európu. Frekvencia 2,4 GHz poskytuje 16 rozdielnych kanálov s maximálnou rýchlosťou prenosu až 250 Kbps. Nižšia frekvencia 915 MHz poskytuje 10 kanálov s maximálnou rýchlosťou 40 Kbps a frekvencia 868 MHz má jeden kanál s maximálnou rýchlosťou prenosu 20 Kbps [7].

Maximálna veľkosť prenášaných paketov je 128 bajtov. Oproti ostatným bezdrôtovým technológiám je to výrazne menej, no štandard IEEE 802.15.4 bol vyvinutý pre komunikáciu s nenáročnými požiadavkami na veľkosť prenášaných údajov. Dôležitá je najmä ich efektívnosť.

Technológia ZigBee umožňuje zapojenie siete do troch rôznych topológií, ktoré sa nazývajú hviezda, sieť a strom[8]. Každá z nich prináša výhody v rôznych situáciách. Hviezda je najčastejšie využívaná, najmä vďaka svojej jednoduchosti. Z názvu vyplýva, že tvar predstavuje niekoľko koncových zariadení pripojených na centrálny uzol. Sieť predstavuje prepojenie jednotlivých uzlov rôznymi cestami, čo zvyšuje stupeň spoľahlivosti komunikácie. Pri problémoch na preferovanej trase prenosu medzi dvoma uzlami môže byť prenos realizovaný inou cestou. Umožňuje to smerovanie, ktoré využíva niektoré uzly v sieti na preposielanie údajov. Posledná topológia strom predstavuje kombináciu hviezdy a siete.

Šetrenie spotrebovanej energie je pri fungovaní siete senzorov a kontrolných zariadení veľmi dôležitým faktorom. Vďaka rámcom, ktoré nazývame "Beacons" je možné synchronizovať komunikáciu medzi koordinačným uzlom siete a koncovými zariadeniami. Rámce periodicky uspávajú a zobúdzajú zariadenia v sieti a umožňujú tak šetrenie energie pri komunikácii. Vďaka tomu môžu zariadenia fungovať aj niekoľko rokov na obyčajnej alkalickú batériu.

Technológia ZigBee teda nebola vytvorená na súperenie so štandardami ako Bluetooth alebo Wi-Fi. Ponúka však ideálne riešenie nízko energetickej komunikácie zariadení napájaných batériami.

2.2.3 Bluetooth

Vývoj tejto technológie začal v roku 1998, keď sa združila skupina spoločností Ericsson, IBM, Intel, Nokia, and Toshiba za úmyslom vytvoriť globálnu technológiu pre bezdrôtovú komunikáciu zariadení na krátku vzdialenosť.

Základom pri vývoji sa stal štandard IEEE 802.15.1, rovnako ako u technológie ZigBee. Bluetooth však vznikol za účelom komunikácie zariadení s väčšou výmenou údajov a tak frekvencia aj rýchlosť prenosu sú väčšie. Používa bezlicenčné ISM frekvenčné pásmo od 2.4 do 2.485 GHz. Po uvedení technológie sa Bluetooth stal nástupcom klasickej káblvej komunikácie cez rozhranie RS-232 a vďaka svojim vlastnostiam sa stal najvyužívanejším spôsob komunikácie dvoch alebo viacerých zariadení na krátku vzdialenosť.

Zariadenia umožňujúce komunikáciu cez Bluetooth rozhranie sú identifikované dvoma parametrami. Prvým je unikátna nemenná 48-bitová adresa, priradená zariadeniu pri výrobe, podobne ako MAC adresa. Druhým parametrom sú hodiny, ktoré tikajú každých 312.5 μ s. Tieto dva parametre umožňujú synchronizovanú komunikáciu zariadení. Pri vytváraní PAN sietí sa delia zariadenia na nadriadené a podriadené. Jeden nadriadený člen siete tak riadi bezpečný prenos údajov vďaka synchronizácií hodín s podriadenými zariadeniami [9].

Bluetooth 4.0

Vývoju nového sveta inteligentných zariadení sa prispôsobuje aj Bluetooth a verzia 4.0 prináša ako najdôležitejšiu zmenu oproti svojim predchodcom nízku spotrebu energie. Táto verzia sa preto nazýva aj Bluetooth Smart alebo Bluetooth Low Energy. Rovnako ako u predchádzajúcej verzie, protokol je rozdelený do dvoch častí, ktoré sa nazývajú Controller a Host. Controller je zvyčajne implementovaný ako SOC a zahŕňa fyzickú a linkovú vrstvu protokolu. Host beží na procesore a riadi funkciu vyšších vrstiev. Komunikácia medzi týmito časťami je realizovaná cez rozhranie nazývané HCI. Aj keď Controller verzie 4.0 prevzal funkcie od svojho predchodcu, tieto časti sú navzájom nekompatibilné. Zariadenia s Bluetooth 4.0 musia preto implementovať aj zásobník klasického protokolu svojich predchodcov. Takéto zariadenia sa potom nazývajú duálne.

Verzia 4.0 poskytuje rovnaký počet 40 kanálov ako staršie verzie, no s dvojnásobne rozšírenou šírkou 2MHz. Delia sa pritom do dvoch skupín na reklamné a údajové, pričom iba tri sú reklamné. Všetky fyzické kanály používajú GFSK moduláciu, ktorá je jednoduchá na implementáciu. Index modulácie je v rozmedzí 0.45-0.55, čo znižuje maximálnu spotrebu energie. Rýchlosť prenosu na fyzickej vrstve je 1Mb/s.

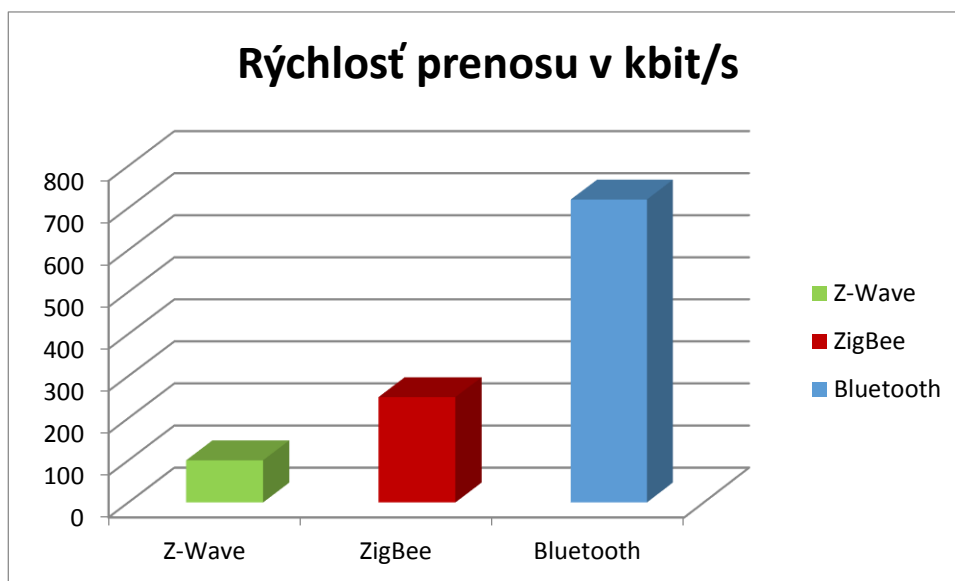
Využitie reklamných kanálov je aj pri vysielaní broadcast paketov. Pre komunikáciu dvoch zariadení je ale potrebné spárovanie, ktoré je tiež realizované cez reklamné kanály. Párovanie je pritom asymetrická procedúra, kde inicializátor nájde zariadenie s ktorým chce začať komunikáciu a posieľa mu žiadosť o spojenie, ktorá vytvára point-to-point spojenie. Po spojení dvoch zariadení prebieha ďalšia komunikácia už po fyzických kanáloch. Doručenie údajov musí podriadené zariadenie vždy potvrdiť odpoveďou. Medzi prenášanými paketmi je definovaná IFS 150 μ s. Každý paket je po doručení ešte kontrolovaný CRC kódom.

Využitie nízko energetickej spotreby novej technológie sa tak posúva aj do sveta senzorov, ktoré sú napájané batériami s objemom 250mAh. Najmä do technológií monitorujúcich zdravie pacientov, ovládanie domácich zariadení, prípadne bezkontaktných platieb [10].

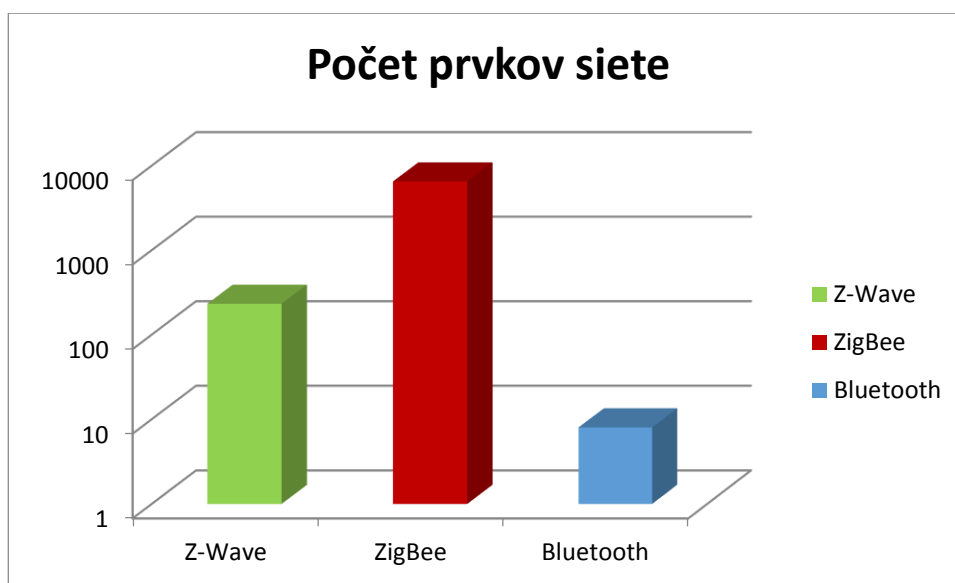
2.2.4 Zhodnotenie

Všetky zhrnuté technológie sú vďaka svojim vlastnostiam budúcnosťou inteligentných domácností. Najmä Z-Wave a ZigBee sú vzájomnou konkurenciou pri vytváraní domácich sietí senzorov a zariadení ovládaných na diaľku. Hlavnou podmienkou v tejto oblasti je nízka spotreba koncových zariadení, čo umožňuje nenáročnosť na množstvo prenášaných dát. Technológia Bluetooth im so svojou verziou 4.0 nepredstavuje významnú konkurenciu v takýchto rozsiahlych sieťach. Naopak, jej zameranie spočíva v prepojení zariadení s mobilnými telefónmi, počítačmi a tabletami. V tejto oblasti má vybudovaný silný základ vďaka svojim predchádzajúcim verziám. Cieľom môjho projektu je teda vytvoriť senzorový systém pre zariadenie s operačným systémom Android a technológia Bluetooth je v tomto smere ideálna. Výhodou je najmä fakt, že nie je potrebné pripájať žiadne

externé zariadenia, ako u technológií ZigBee a Z-Wave. Pretože všetky zariadenia s operačným systémom Android obsahujú Bluetooth rozhranie.



Obrázok 1 Porovnanie rýchlostí prenosu technológií Z-Wave, ZigBee a Bluetooth.



Obrázok 2 Počet zariadení pri technológiách Z-Wave, ZigBee a Bluetooth.

2.3 Existujúce senzorové systémy

Kapitola opisuje existujúce riešenia senzorových systémov, ktoré svojimi vlastnosťami pripomínajú požiadavky môjho projektu. Slúži ako inšpirácia pri vytváraní návrhu implementácie môjho projektu.

2.3.1 Netatmo

Netatmo je súprava vonkajšieho a vnútorného senzorového systému, ktorý pomocou aplikácie poskytuje aktuálne informácie o podmienkach prostredia. Prenos zaznamenaných údajov prebieha cez Wi-Fi router priamo na Netatmo servery. Aplikácia na zariadení je po

spárovaní pripojená k rovnakým serverom a počas pripojenia na internet neustále aktualizuje údaje[11].

Oba oddelené prvky obsahujú takmer rovnaký set senzorov avšak s odlišným rozsahom. Tento fakt je prispôsobený rozdielnym podmienkam vo vnútornom a vonkajšom prostredí. Napríklad teplotný senzor v menšom, vonkajšom module poskytuje rozsah od -40°C do 65°C, naopak vnútorný ponúka znížený rozsah iba od 0°C do 50°C. Vnútorný modul ponúka na viac senzor, ktorý sleduje obsah CO₂(oxid uhličitý) v ovzduší. Pri nadmernom obsahu CO₂ upozorní na kvalitu vzduchu farbou LED svetielka na povrchu modulu a varovaním prostredníctvom aplikácie.

Napájanie zariadení je závislé od prostredia, pre ktoré boli vytvorené. Vnútorný modul je zásobovaný elektrickou energiou prostredníctvom USB rozhrania, pripojením do bežnej elektrickej zásuvky. Vonkajší model je napájaný štyrmi AAA batériami, s potrebou výmeny raz ročne [12].Vzhľadom na to, že komunikáciu so servermi zabezpečuje iba vnútorný modul, musí byť zabezpečené spojenie oboch modulov. Realizované je bezdrôtovým prepojením s frekvenciou 915 MHz alebo 868 MHz s dosahom do 100 metrov.

Aplikácia pre Android je kompatibilná s väčšinou verzií tohto operačného systému a to s verziou Android 2.3.4 a všetkými jej nasledovníkmi [12].

2.3.2 SensoDuino

SensoDuino je aplikácia, voľne dostupná na internetovom obchode Google play pre Android. Zaznamenáva údaje zo senzorov, ktoré sú zabudované v zariadení, na ktorom je inštalovaná.

Získané informácie môže iba zobrazovať na obrazovke, alebo ich súčasne priamo posielat' cez Bluetooth rozhranie na dosku Arduino. Prenos je realizovaný pripojením Bluetooth modulu HC-05 k doske Arduino a spárovaním so zariadením s nainštalovanou aplikáciou. Umožňuje nastavenie frekvencie zaznamenávania dát v rozmedzí od 100 milisekúnd(10Hz) po 10 minút. Nastavená frekvencia záznamu údajov by nemala byť vyššia akou disponuje senzor z ktorého aplikácia zaznamenáva informácie. Dobrým príkladom je napríklad GPS, ktorá vo veľkej väčšine zaznamenáva polohu každú sekundu(1Hz), v prípade nastavenia vyššej frekvencie teda dosiahneme iba duplikovanie rovnakej hodnoty, nie väčšiu presnosť. Frekvencia záznamu údajov nie je jediná, ktorú môžeme upraviť z používateľského rozhrania. Aplikácia používateľovi umožňuje aj rýchlosť prenosu údajov medzi Android zariadením a doskou Arduino [13].

Na druhej strane prenosu zachytáva tzv. "sketch", čo je názov pre program vytvorený v používateľskom prostredí Arduino IDE . Následne konvertuje prenášaný formát reťazca do numerických hodnôt, pripravených pre použitie, napríklad riadenia robota.

Aplikácia slúži ako alternatíva pri tvorbe senzorových systémov pripojených k doske Arduino. Umožňuje využitie senzorov zabudovaných v zariadení s operačným systémom Android, namiesto nákupu externých senzorov k doske Arduino. Vytvára teda finančne nenáročnejšiu cestu pri tvorbe napríklad robotických zariadení.

2.4 Funturo, Arduino a Raspberry Pi platformy

Podkapitola obsahuje analýzu zariadení, ktoré by mohli byť použité pri vytváraní senzorového systému pre operačný systém Android.

2.4.1 Funtoro HD Infotainment system (1 DIN)

Funtoro 1-Din je zábavno-informačný server určený pre využitie v hromadných dopravných prostriedkoch. Svojimi rozmermi pripomína bežné autorádia a preto je možné ho nainštalovať takmer do každej palubnej dosky. Vďaka operačnému systému Android 4.2.2 môže vodič ovládať multimediálny obsah na obrazovkách, zdieľať informácie o trase alebo vozidle. Umožňuje aj pripájanie externých zariadení cez rozhrania HDMI, USB, RS 232. Nedostatkom oproti ostatným zariadeniam s operačným systémom Android je neprítomnosť Bluetooth rozhrania priamo v zariadení. Po odokrytí predného panela umožňuje pripojenie externého disku alebo pamäťových kariet SD a micro SD [14].

2.4.2 Arduino

Myšlienka platformy Arduino vznikla v roku 2005 na severe Talianska v Interaction Design Institute Ivrea. Profesor Massimo Banzi nadobudol pocit, že vtedajší trh neposkytoval dostačujúce možnosti pre rozvíjanie zručností študentov, či nadšencov elektroniky. Dôvodom bola najmä vysoká cena obdobných zariadení a ich nedostatočná funkcionálnosť. Profesor Banzi zároveň chcel platformu, ktorá by bežala aj na počítačoch Macintosh, ktoré sa v IDII najviac využívali [15].

Výsledkom vývoja je voľne šíriteľná platforma, ktorá umožňuje finančne nenáročné riešenie elektronických projektov. Skladá sa z fyzického programovateľného obvodu (často nazývaný mikroovládač) a softvéru, resp. integrovaného vývojového prostredia (IDE).

Arduino v súčasnosti ponúka niekoľko typov zariadení, ktoré obsahujú 8-bitové mikroprocesory od firmy Atmel. Jednotlivé varianty sa líšia v technických parametroch ako napríklad počet I/O pinov, veľkosť pamäte RAM, veľkosťou napájacieho napätia a podobne. Vzhľadom na to, že najdôležitejšími faktormi pri vytváraní Arduino boli nízke finančné náklady a jednoduchosť, tak aj práca s používateľským prostredím (Arduino IDE) je intuitívna. Arduino IDE je implementované pre najpoužívanejšie operačné systémy osobných počítačov Mac OS, Windows a Linux. Poskytuje aj textový editor na písanie programového kódu v jazyku, ktorý je zjednodušenou verziou jazyka C++. Programátor má k dispozícii aj množstvo predprogramovaných knižníc, ktoré môže využívať pri práci

na svojom projekte. Vďaka pripojeniu cez USB kábel je možné nahrávať programy priamo z používateľského prostredia na dosku.

Zariadenia Arduino poskytujú široké spektrum využitia a tešia sa veľkej obľube. Najzákladnejšie dosky sú zväčša používané na menších projektoch pri práci s rôznymi senzormi. Pokročilejšie dosky tvoria základ v IoT projektoch alebo 3D tlačiarňach.

Arduino UNO

Pracovné napätie tejto najvyužívanejšej verzie dosiek Arduino je 5V. Základným prvkom tejto verzie je mikroprocesor ATmega328P.

Arduino UNO obsahuje 14 vstupno-výstupných digitálnych pinov(z toho 6 je možné využiť ako PWM), 6 vstupných analógových pinov, 16 MHz kryštálový oscilátor, USB pripojenie, power jack, ICSP rozhranie pre programovanie bootloadera a reset tlačidlo. Pamäť RAM je veľká 2 KB a pamäť pre uloženie programu ponúka 32 KB.

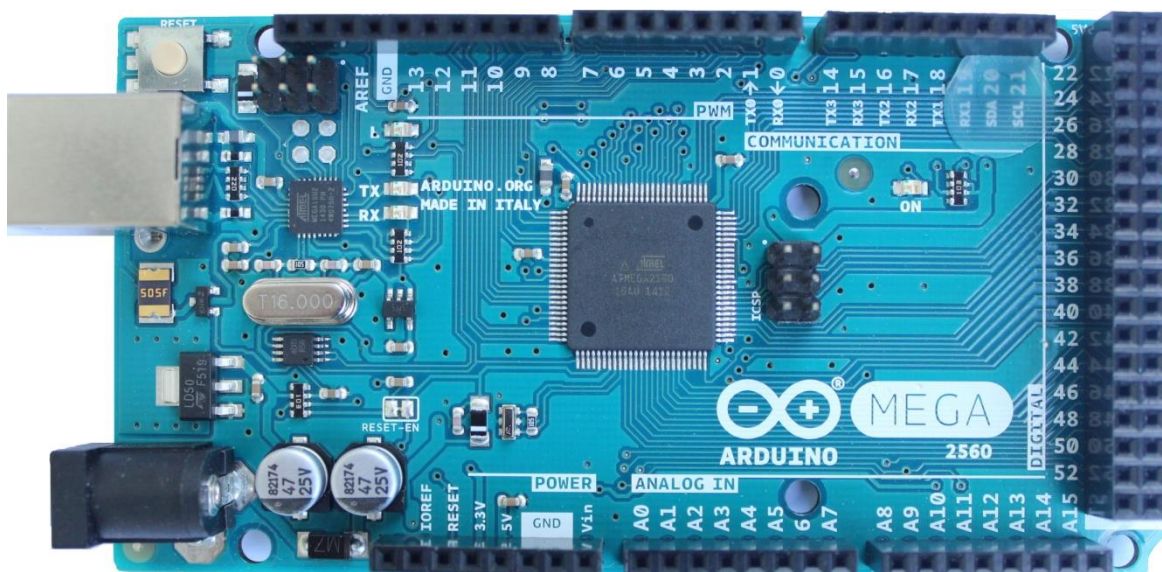
Napájanie je realizovateľné pripojením zariadenia k osobnému počítaču cez USB kábel, alebo cez AC-DC adaptér [16].

Arduino Mega2560

Pracovné napätie dosky Arduino Mega2560 je rovnaké ako pri ostatných doskách 5V. Najzákladnejším rozdielom medzi jednotlivými doskami býva mikroprocesor, v tejto verzii sa nachádza mikroprocesor ATmega2560.

Obsahuje 54 digitálnych vstupno-výstupných pinov(z ktorých 15 môže byť použitých ako PWM), 16 vstupných analógových pinov, 4 UARTs(hardvérové serial porty), 16 MHz kryštálový oscilátor, USB pripojenie, power jack, ICSP rozhranie pre programovanie bootloadera a reset tlačidlo. Na rozdiel od základných verzií Arduino dosiek ponúka zväčšenú pamäť RAM až 8KB. Rovnako aj programová pamäť ponúka výrazne väčší 256 KB pamäťový priestor pre uloženie rozsiahlejších programov.

Ponúka tiež dve formy zásobovania obvodu energiou a to buď cez USB port, alebo cez AC-DC adaptér [17].



Obrázok 3 Doska Arduino Mega2560.

2.4.3 Raspberry Pi

Raspberry Pi je lacný osobný počítač vo veľkosti kreditnej karty. Podobne ako Arduino, aj projekt Raspberry Pi vznikol s úmyslom priniesť finančne nenáročné zariadenie pre vzdelávanie študentov. Na rozdiel od Arduino sa však jedná o počítač a poskytuje tak štandardné vstupno-výstupné rozhrania ako bežné osobné počítače. Poháňaný je čipom BCM2835, respektíve BCM2836 v modeloch novej generácie. Čip BCM2835 obsahuje ARM1176JZFS procesor s frekvenciou 700 MHz a Videocore IV GPU. Zariadenia obsahujú aj možnosť pripojenia počítačového monitora cez HDMI rozhranie, a tiež počítačovú myš alebo klávesnicu cez USB rozhranie. Na niektorých modeloch sa nachádza aj Ethernetové rozhranie. Spoločnosť Raspberry Pi Industrial vyvinula aj operačný systém Raspbian, no v prípade potreby je možné inštalovať aj Ubuntu MATE, Windows IOT Core a ďalšie [18].

Raspberry Pi A+

Model A+, ktorý je nástupcom Modelu A bol predstavený v Novembri 2014. Tento model obsahuje 40 I/O pinov, slot na micro SD kartu, 3,5mm audio jack, HDMI rozhranie, jeden USB port. Nabíjanie zariadenia je realizované pomocou USB portu ¹.

2.4.4 Zhodnotenie

Po analyzovaní všetkých zariadení a zhodnotení možností, ktoré tieto zariadenia ponúkajú som sa rozhodol použiť Arduino Mega2560. V prípade zariadenia Funturo 1-Din je nerealizovateľné jeho využitie pre tvorbu ľahkého a kompaktného senzorového modulu, pretože váha a veľký odber elektrickej energie ho predurčuje skôr pre statické využitie.

¹ <http://docs-europe.electrocomponents.com/webdocs/1354/0900766b81354aa6.pdf>.

Ďalšími nedostatkom je, že neobsahuje ani rozhranie Bluetooth, ktoré je ideálne na komunikáciu na krátku vzdialenosť. Napriek tomu, že Raspberry Pi A+ je oveľa výkonnejší ako dosky Arduino, tento parameter nie je najdôležitejším pre realizovanie môjho projektu. Vzhľadom na cieľ vytvoriť čo najľahší modul, prvým rozhodovacím faktorom bola váha samotnej dosky a následne externej batérie potrebnej pre jej napájanie. Rovnako ako váha dosky, tak aj odber elektrického prúdu sú výhodou dosky Arduino Mega2560. Tá s odberom 200mAh umožňuje použitie ľahšej externej batérie s dostatočnou zásobou elektrickej energie. Napriek tomu, Arduino UNO je v tomto faktore ešte lepšia, no neponúka dostatočnú veľkosť pamäte RAM a ani dostatočnú veľkosť programovateľnej pamäte. Dôvodom je program zabezpečujúci ukladanie dát so senzorov a posielanie ich cez rozhranie Bluetooth. Ten bude vyžadovať aj importovanie externých knižníc, ktoré vyžadujú veľké množstvo pamäte. Existencia týchto knižníc priamo tvorcami platformy Arduino sú tiež veľkou výhodou pri realizácii projektu.

Tabuľka 1. Porovnanie najdôležitejších parametrov.

	Váha	pamäť RAM	Pamäť	Priemerný odber elektrického prúdu
Adruino UNO	25g	2KB	32KB	8mA ²
Adruino Mega 2560	37g	8KB	256KB	20 mA ³
Raspberry Pi A+	45g	256MB	micro SD	100 mA ⁴
Funtoro 1-Din	1435g	2GB⁵	8GB + externá	800 mA⁶

2.5 Bluetooth modul, Senzory a ich pripojenie k Arduino

Dosky Arduino umožňujú pripojenie externých zariadení pomocou pinov umiestnených na doske. Môžeme si pritom vybrať z niekoľkých komunikačných protokolov, ktoré umožňujú prenos údajov na úrovni bitov. Na začiatku si uvedieme rozdelenie zberníc, podľa niekoľkých kategórií.

² <https://code.google.com/p/agritech-ugv-arduino/downloads/detail?name=arduino+uno+datasheet.pdf>

³ http://www.atmel.com/images/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

⁴ <http://raspi.tv/2014/raspberry-pi-a-how-much-power-does-it-need>

⁵ http://shop.molpir.sk/Prilohy/254140002510_0.pdf

⁶ Ing. Martin Kosa . SNP 129, 919 04 Smolenice, Slovakia. 21.Arpl.2016. Osobná komunikácia

Typy zberníc podľa spôsobu riadenia [19]:

- Single-Master (Iba jedno zariadenie v systéme môže riadiť zbernicu.)
- Multi-Master (Na zbernicu je pripojených viacero zariadení, ktoré môžu riadiť zbernicu. V danom čase však môže byť iba jedno riadiace zariadenie.)

Typy zberníc podľa spôsobu synchronizácie [19]:

- synchronná (Prenos je synchronizovaný spoločným hodinovým signálom.)
- asynchronná (Prenos je synchronizovaný odpoveďou podriadeného zariadenia.)

Typy zberníc podľa množstva prenášaných dát [19]:

- sériové (prenos jedného bitu v jednom hodinovom cykle)
- paralelné (prenos viacerých bitov v jednom hodinovom cykle)

Podľa časového multiplexu [19]:

- Multiplexované (Význam údajov prenášaných po zbernici sa mení v čase. Môžu byť prenášané údaje alebo adresa. Typ údajov odlišujú prídavné signály.)
- Nemultiplexované (Význam údajov prenášaných po zbernici sa v čase nemení.)

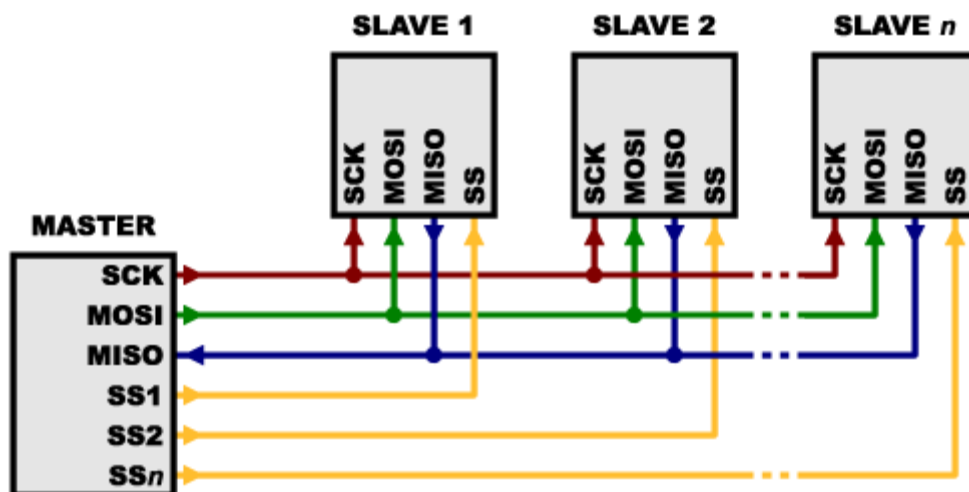
2.5.1 SPI

Je nemultiplexovaná, sériová zbernica, ktorá na prenos údajov používa $3+n$ liniek. Číslo n je definované počtom slave zariadení, pripojených k zbernici. Tá môže obsahovať práve jedno zariadenie master, ktoré zabezpečuje synchronizáciu prenosu pomocou taktovacej linky SCLK.

Prenos údajov medzi zariadeniami master a slave je realizovaný dvoma jednosmernými linkami. Jedna je definovaná pre prenos údajov v smere zo zariadenia slave do master a je označovaná MISO. Prenos v smere do zariadenia master môže realizovať iba jedno zariadenie slave v danom čase. Druhá definuje opačný smer a teda zo zariadenia master do slave a je označovaná ako MOSI. Pretože SPI nepoužíva adresovanie, je vhodná aj pre stream údajov do vybraných zariadení slave. Umožňuje to spôsob akým SPI master vyberá slave zariadenia, s ktorými chce komunikovať. Používa na to signály SS1 až SSN, ktoré sú pripojené na zariadenia a nahrádzajú adresovanie.

SPI protokol

SPI protokol nie je potrebný, pretože sa nepoužíva adresovanie. Výber je realizovaný hardvérovo a tak každé zariadenie obsahuje vlastný protokol, ktorý kontroluje prístup a riadenie. Dôležitým faktom je tiež, že SPI neposkytuje potvrdzovanie doručenia ani riadenie toku údajov a preto ak je to potrebné, nie je vhodný [20].



Obrázok 4 Zapojenie zariadení v technológii SPI⁷.

2.5.2 1-Wire

1-Wire je multiplexovaná zbernica, ktorá umožňuje komunikáciu medzi jedným zariadením master a viacerými 1-Wire slave zariadeniami. Prenos údajov riadi master a tak slave môže posilať údaje na zbernicu až keď master inicializuje komunikáciu. Rovnako komunikácia medzi dvoma slave zariadeniami je možná iba cez master zariadenie.

Protokol 1-Wire používa CMOS/TTL logické úrovne s operáciami špecifikovanými v rozmedzí od 2.8V do 6V. Oba typy zariadení fungujú ako vysielateľ aj prijímač umožňujúce prenos údajov oboma smermi. V danom čase však prebieha komunikácia vždy jedným smerom podľa posledného príznakového bitu LSB.

Prenos údajov je špecifický oproti ostatným opísaným technológiám. Prenášaná logická jednotka je v tomto prípade definovaná potlačením zbernice na hodnotu logickej 0 na čas 15μs alebo kratší. Definícia prenášanej nuly je potlačením zbernice na hodnotu logickej 0 na čas 60μs alebo dlhší. Systémová hodinová linka nie je potrebná, pretože každé zariadenie je časované vlastným oscilátorom, ktorý je synchronizovaný podľa dobežnej hrany master zariadenia.

Napájanie slave zariadení je realizované v čase keď je komunikácia nečinná a zbernica má 5V. V takomto prípade je dióda pol-vlnového oddelovača zapnutá a nabíja kapacitor, ak sa napätie zníži dióda vypne nabíjanie. Takýto spôsob sa nazýva sa parazitný a umožňuje pracovať slave zariadeniu aj keď je zbernica na dolnej úrovni.

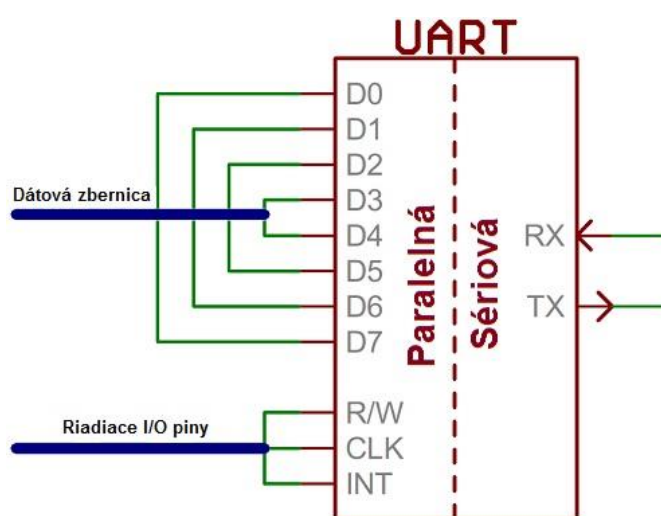
Podržaním zbernice na dolnej úrovni na čas 480μs realizuje resetovanie, za ktorým nasleduje inicializácia novej komunikácia. Tá je realizovaná zistením prítomnosti slave zariadení a synchronizovaním časovania. Ďalej už master pristupuje k zariadeniu podľa jeho 64-bitovej adresy. Tá je tvorená prvým bytom, ktorý definuje typ zariadenia. Ďalších

⁷ <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>

6 bajtov je individuálna adresa a na koniec je CRC byte, ktorý slúži na overenie prenosu adresy [21].

2.5.3 UART

UART je označovaná ako univerzálna asynchrónna zbernica, pretože jej parametre, ako sú rýchlosť, veľkosť prenášaných údajov a ďalšie, nie sú fixne dané. Môžu byť konfigurované podľa požiadaviek danej komunikácie a obe zariadenia musia mať tieto parametre definované rovnako. Predstavuje sprostredkovateľa medzi paralelným a sériovým rozhraním [22]. Na jednej strane je zbernica tvorená ôsmimi dátovými linkami a ďalšími riadiacimi pinmi, zabezpečujúcimi ovládanie a na druhej sú dve linky označované RX a TX.



Obrázok 5 Zjednodušené zobrazenie UART rozhrania⁸.

Technológia je určená na prepájanie dvoch zariadení s krížením liniek RX(prijímač) a TX(vysielač). Vysielač jedného zariadenia musí byť pripojený na prijímač zariadenia s ktorým komunikuje a v opačnom smere rovnako. V prípade komunikácie viacerých zariadení by vzhľadom na vlastnosti UART dochádzalo ku kolíziám, najmä v prípade, ak by sa dve zariadenia pokúšali vysielat' po rovnakej linke. Komunikácia viacerých zariadení by bola možná iba v prípade zapojenia viacerých zariadení, ktoré by slúžili ako prijímače.

UART nájdeme aj ako samostatné obvody, no väčšina mikroovládačov obsahuje minimálne jeden. V prípade potreby ho môžeme implementovať aj softvérovo, napríklad pre dosky Arduino existuje knižnica `SoftwareSerial`, ktorá umožňuje vytvorenie ďalšieho spojenia pomocou UART.

⁸ <https://learn.sparkfun.com/tutorials/serial-communication>

2.5.4 I²C

I²C je dvojlínková, synchronná, obojsmerná zbernica, ktorá poskytuje jednoduchý a efektívny prenos dát medzi dvoma alebo viacerými zariadeniami. Bežne sa pripája jedno master zariadenie, ktoré synchronizuje hodiny na zbernici a ovláda prenos údajov. I²C však umožňuje aj pripojenie viacerých slave zariadení. Najčastejšie sa využíva pri komunikácii na krátku vzdialenosť. Zabezpečuje pritom detekciu kolízií, ktorá chráni údaje pred poškodením, v prípade, že sa viacero nadriadených zariadení pokúsi riadiť zbernicu súčasne [23].

Systémová konfigurácia

I²C používa na prenos dát údajovú linku(SDA) a taktovaciu linku(SCL). Na obe linky je pripojená logická funkcia AND so zvyšovacím (pull-up) odporom.

Údaje sú prenášané medzi zariadením master a jedným zo slave zariadení synchronizovane podľa taktovacej linky SCL po údajovej linke SDA byte za bytom. Každý údajový byte má 8 bitov, pričom každý bit je prenášaný počas jedného hodinového cyklu. Po každom prenesenom byte nasleduje signál ACK, ktorý odosiela príjemca, aby potvrdil správne prijatie údajov. Bity sú prenášané po zbernici SDA vtedy, keď má SCL logickú hodnotu 1. Obsah SDA je teda možné meniť iba v stave, kedy SCL má logickú hodnotu 0. V prípade, že SCL má logickú hodnotu 1, musí byť obsah zbernice SDA stabilný [24].

I²C Protokol

Štandardná komunikácia prebieha v štyroch krokoch:

- 1.Vygenerovanie signálu štart
- 2.Prenos adresy podriadeného zariadenia
3. Prenos údajov
- 4.Vygenerovanie signálu stop

1. Signál START

Nadviazanie novej komunikácie je možné začať vtedy, ak je zbernica voľná, a teda žiadne master zariadenie práve neriadi zbernicu. Ak je zbernica voľná, potom SDA aj SCL majú logickú hodnotu 1. Komunikácia začína odoslaním signálu START (ozn. S) zariadením master. Tento signál je definovaný ako prechod SDA z logickej hodnoty 1 do logickej hodnoty 0, zatiaľ čo SCL má logickú hodnotu 1. Tento signál znamená začiatok nového prenosu údajov.

Opakovaný START je START signál, ktorý bol vygenerovaný skôr ako bola ukončená predchádzajúca komunikácia a teda bol poslaný signál STOP. Zariadenie master ho používa pri komunikácii so zariadeniami slave s ktorými práve neprebieha komunikácia.

Môže ho však použiť aj pri komunikácii so zariadením s ktorým práve komunikuje, no v opačnom smere prenosu.

START signál je generovaný až vtedy, keď je nadstavený STA bit v príkazovom registri a zároveň jeden z bitov RD alebo WR. V závislosti od aktuálneho stavu zbernice sa posiela START alebo opakovaný START.

2. Prenos adresy zariadenia slave

Prvý byte po signáli štart je adresa zariadenia, s ktorým chce master komunikovať. Skladá sa zo 7 bitovej adresy, za ktorou nasleduje RW bit. RW bit signalizuje zariadeniu slave smer prenosu dát. Keďže každé zariadenie pripojené na zbernicu má unikátnu adresu, odpovedá iba to, ktorého adresa bola poslaná na zbernicu. Odpovedá odoslaním tzv. ACK bitu a teda zmenou SDA na logickú hodnotu 0 v 9. hodinovom cykle.

Jadro zaobchádza s prenosom adresy rovnako, ako so zápisom. Stačí uložiť adresu do Transmit registra a nastaviť WR bit. Jadro potom pošle adresu na zbernicu.

3. Prenos údajov

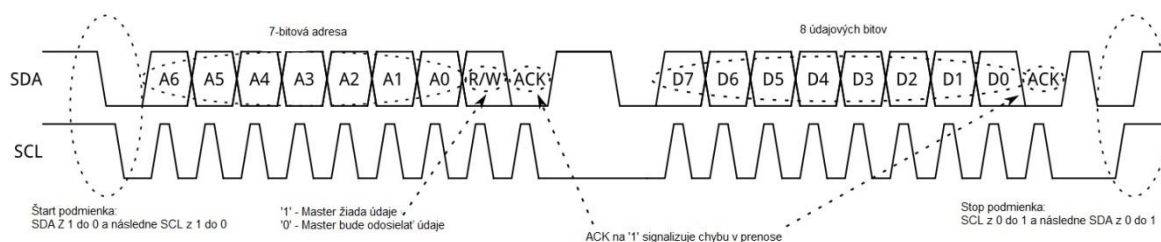
Keď prebehne proces adresovania zariadenia úspešne, môže začať prenos údajov byte po byte v smere ako definoval RW bit v predchádzajúcom kroku. Každý byte musí byť nasledovaný ACK bitom v deviatom hodinovom cykle SCL, rovnako ako pri adrese. V prípade, že slave nepotvrdí doručenie s bitom ACK, môže master generovať signál STOP a zablokovat' prenos údajov. Môže však aj generovať opakovaný START a začať nový cyklus prenosu.

Ak je však master príjemcom a nepotvrdí prijatie údajov, slave uvoľní zbernicu SDA aby mohol master generovať signál STOP alebo opakovaný START.

Pri zapisovaní údajov na slave je potrebné uložiť údaje do Transmit registra a nastaviť WR bit. Pre čítanie je zasa potrebné nastaviť RD bit a jadro počas prenosu nastaví príznak TIP, čo indikuje prenos údajov. Po skončení prenosu sa tento príznak resetuje a v prípade, že prenos prebehol správne, nastaví sa príznak IF. Receive register potom obsahuje platné dáta. Nový príkaz zápisu, alebo čítanie môže byť volaný, ak TIP príznak bol resetovaný.

4. STOP signál

Master môže zrušiť komunikáciu generovaním signálu STOP(ozn. P). Stop je definovaný ako zmena SDA z logickej hodnoty 0 na logickú hodnotu 1, zatiaľ čo SCL má logickú hodnotu 1.



Obrázok 6 Zobrazenie prenosu adresy a údajov protokolom I2C⁹.

2.5.5 Realizácia pripojenia

Súčasný trh ponúka široké spektrum senzorov, ktoré umožňujú sériové pripojenia k doskám Arduino. Ja som sa pre jednoduchosť implementácie rozhodol použiť senzory s pripojením pomocou I²C protokolu. Ten umožňuje jednoduché pripojenie viacerých zariadení na rovnakú zbernicu a realizuje výber zariadenia, s ktorým chce komunikovať softvérovo. To uľahčuje fyzické prepájanie zariadení, napríklad oproti SPI, ktoré vyberá slave zariadenie pre komunikáciu hardvérovým spojením. To vyžaduje ďalšie prepájanie pomocou káblov. V porovnaní s 1-Wire protokolom zase prináša výhodu synchronizovanej komunikácie, vďaka linke SCL, naopak 1-Wire je asynchrónna zbernica.

UART sa zasa využíva na prepájanie iba dvoch zariadení a je najčastejšie využívaný pre pripájanie Bluetooth modulov k doskám Arduino.

2.6 Zhodnotenie analýzy

V kapitole analýza sú opísané oblasti, ktoré priamo súvisia s problematikou podľa zadania mojej bakalárskej práce.

Na začiatku kapitoly je opísaný operačný systém Android v spojení so senzormi, ktoré sa nachádzajú v zariadeniach s týmto operačným systémom. Senzory sú začlenené do troch skupín. Následne sú opísané technológie, ktoré sa využívajú pri vytváraní senzorových systémov pre zariadenia s operačným systémom Android. V porovnaní sú opísané výhody a nevýhody technológií a zdôvodnený výber Bluetooth technológie.

Ďalej sú opísané riešenia podobných senzorových systémov, ktorú slúžia ako inšpirácia pri realizovaní môjho projektu. Výber zariadení pre realizáciu senzorového systému je opísaný v ďalšej časti, kde zdôvodňujem, ktoré vlastnosti zariadení sú rozhodujúcimi pre realizáciu môjho projektu. Ako najvhodnejšie zariadenie vychádza z tohto porovnania doska Arduino Mega2560. Tá by mala slúžiť na pripájanie senzorov, ktoré budú získavať údaje pre odoslanie na Android aplikáciu.

⁹ <https://learn.sparkfun.com/tutorials/i2c>

V závere sú analyzované niektoré možnosti pripojenia externých zariadení k doskám Arduino. Opísané sú sériové protokoly, ktoré sa najčastejšie využívajú pre pripájanie senzorov, alebo Bluetooth zariadení. Výsledkom analýzy je zhodnotenie prostriedkov, ktoré súčasný trh ponúka a zjednodušený návrh senzorového systému pre Android aplikáciu. Ten bude realizovaním pripojením senzorov k doske Arduino, ktorá bude cez Bluetooth modul odosielať údaje na Smartfón s operačným systémom Android. Ďalšia kapitola opisuje detailný návrh tohto projektu.

3 Opis riešenia

Kapitola obsahuje návrh senzorového systému s využitím poznatkov z predchádzajúcej kapitoly. Definuje požiadavky na vytváraný systém, architektonický návrh a návrh aplikácií pre Android a Arduino platformy.

3.1 Špecifikácia požiadaviek

V podkapitole sú opísane funkcionálne a nefunkcionálne požiadavky na vytvárané aplikácie pre Arduino a Android platformy.

3.1.1 Špecifikácia funkcionálnych požiadaviek

Základnou požiadavkou pre aplikáciu na operačný systém Android je umožnenie behu aplikácie pri vypnutej obrazovke zariadenia. Táto požiadavka vyplýva z nefunkcionálnej požiadavky šetrenia energie, pretože práve obrazovka jej v súčasných smartfónoch spotrebuje najviac. Aplikácia by mala umožňovať inicializáciu komunikácie výberom spárovaného zariadenia cez rozhranie Bluetooth. Toto zariadenie sa po tomto procese stane zdrojom zaznamenávaných údajov. Musí umožňovať zadanie mena súboru, do ktorého budú prijímané údaje ukladané. Zároveň by mala spravovať uložené súbory so získanými údajmi a umožniť ich vymazanie. Aplikácia by mala slúžiť v dvoch režimoch, podľa výberu pri inicializácii komunikácie. V jednom režime by mala zobrazovať prijaté údaje priamo v aplikácii a umožniť tak používateľovi sledovať aktuálne údaje. Zároveň však musí tieto údaje ukladať do súboru. Druhý režim by mal jednoducho prijaté údaje ukladať do súboru, no bez zobrazovania priamo v aplikácii. Tento režim by mal slúžiť pri dlhodobých meraniach bez interakcie s používateľom a šetriť tak čo najviac energie pre dlhodobú výdrž batérie.

Požiadavky organizované v bodoch:

- aplikácia musí pracovať pri vypnutej obrazovke zariadenia,
- poskytovať údaje iným aplikáciám vo forme prístupných súborov,
- umožniť zobrazovanie prijímaných údajov v aplikácii.

Aplikácia bežiaca na doske Arduino bude zabezpečovať čítanie údajov z pripojených senzorov. Periodicky bude získavať numerické údaje a vyhodnocovať ich správnosť. Až po kontrole správnosti môžu byť odoslané údaje cez Bluetooth modul.

3.1.2 Špecifikácia nefunkcionálnych požiadaviek

Medzi najdôležitejšie nefunkcionálne požiadavky na aplikáciu Android patrí:

- prehľadnosť zaznamenaných údajov ,
- spoľahlivosť - vyladenie aplikácie pre minimalizáciu chybovosti ,

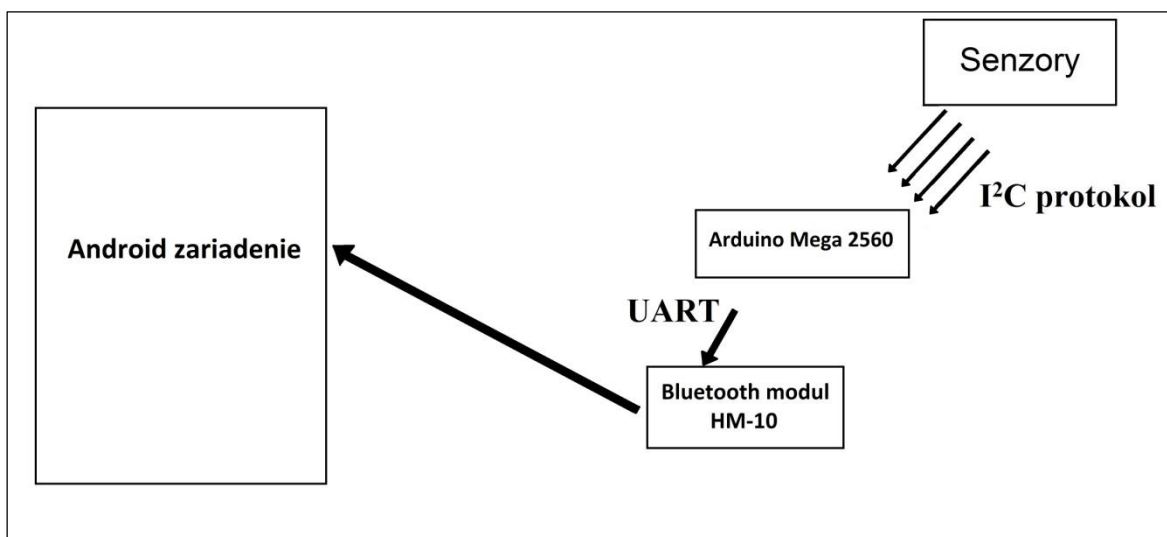
- nízka spotreba elektrickej energie.

Najdôležitejšou nefunkčiou požiadavkou na aplikáciu Arduino je:

- spoľahlivosť - aplikácia musí dokázať obnovenie činnosti programu v prípade zaseknutia sa v chybovom stave.

3.2 Architektonický návrh

Na obrázku nižšie je znázornená architektúra navrhovaného senzorového systému. Projekt bude zložený z dvoch oddelených aplikácií, ktoré budú medzi sebou komunikovať cez rozhranie Bluetooth. Jedna aplikácia bude bežať na doske Arduino a druhá na zariadení s operačným systémom Android. Tok údajov bude realizovaný jedným smerom od pripojených senzorov až k úložnému priestoru v aplikácii na Android zariadení. Pripojenie senzorov a Bluetooth modulu k doske Arduino bude realizované pomocou prepájacích káblov a prepájacej dosky.



Obrázok 7 Architektonický návrh senzorového systému.

3.3 Prípady použitia Android aplikácie



Obrázok 8 Prípady použitia.

Číslo prípadu použitia: UC1

Názov: Vytvorenie spojenia

Popis: Používateľ v zozname spárovaných zariadení vyberá zariadenie s ktorým chce komunikovať a inicializuje komunikáciu.

Číslo prípadu použitia: UC2

Názov: Zadanie mena súboru

Popis: Používateľ pri inicializácii komunikácie zadáva meno súboru, do ktorého sa budú ukladať získavané údaje. Po vytvorení tohto súboru začne prebiehať komunikácia.

Číslo prípadu použitia: UC3

Názov: Vymazanie súboru

Popis: Používateľ má možnosť pri prehľadávaní zoznamu súborov vymazať vybraný súbor.

Číslo prípadu použitia: UC4

Názov: Spravovanie súborov

Popis: Používateľ má možnosť prehľadávať všetky súbory vytvorené aplikáciou.

Číslo prípadu použitia: UC5

Názov: Zobrazenie obsahu súboru

Popis: Používateľ má možnosť pri prehľadávaní zoznamu súborov vybrať konkrétny súbor a zobraziť uložené údaje

Číslo prípadu použitia: UC6

Názov: Zrušenie spojenia

Popis: Používateľ má možnosť zrušiť prebiehajúce spojenie so zariadením.

3.4 Výber implementačného prostredia

Podkapitola uvádza a opisuje výber implementačných prostredí pre obe časti senzorového systému.

3.4.1 Arduino aplikácia

Pre pohodlnú prácu s doskami Arduino bolo vyvinuté aj implementačné prostredie, ktoré sa nazýva Arduino IDE. Umožňuje intuitívne ovládanie s možnosťou využívania predprogramovaných knižníc, napríklad pre komunikáciu cez sériové rozhranie. Rovnako pre vybrané senzory sú implementované knižnice, ktoré plánujem využiť pri implementácii. V tomto prostredí sa používa programovací jazyk ktorý je veľmi podobný jazyku C++.

3.4.2 Android aplikácia

Pre implementáciu Android aplikácie bude použité vývojové prostredie Android Studio od spoločnosti Google. Android Studio umožňuje testovanie aplikácie na virtuálnom zariadení, ktoré simuluje správanie mobilného zariadenia a uľahčuje vývoj aplikácie. Zároveň podporuje tvorbu používateľského rozhrania metódou "drag and drop". Aplikácia bude implementovaná v programovacom jazyku Java.

3.5 Návrh riešenia

Návrh riešenia je vytváraný s ohľadom na špecifikáciu požiadaviek z predchádzajúcej časti. Cieľom je návrhom splniť všetky požiadavky na vytváraný softvér.

3.5.1 Štruktúra systému

Senzorový systém bude tvorený dvoma aplikáciami. Jedna bude zabezpečovať čítanie údajov zo senzorov a bude bežať na doske Arduino a druhá bude tieto údaje spracovávať na platforme Android. Tok údajov medzi oboma aplikáciami bude realizovaný cez rozhranie Bluetooth, ktoré umožňuje bezdrôtovú komunikáciu na krátku vzdialenosť.

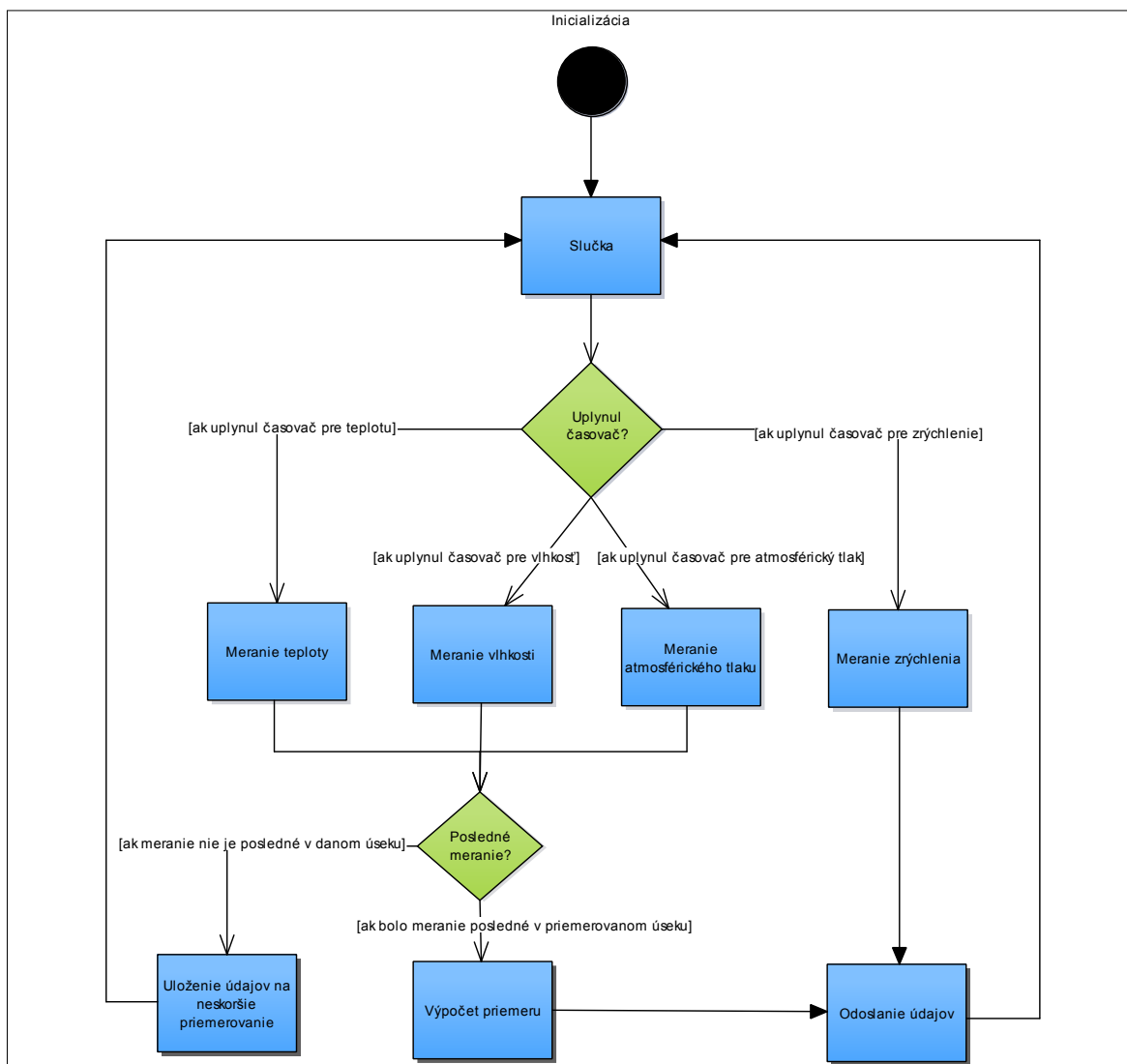
3.5.2 Arduino aplikácia

Na doske Arduino bude v pamäti programu uložený programový kód, ktorý sa spustí po zapojení napájania. Štandardná forma programov je rozdelená do dvoch blokov, ktoré sú reprezentované procedúrami. Na rovnakom princípe bude teda postavená aj moja aplikácia.

V prvom bloku bude prebiehať inicializáciu požadovaných komunikačných rozhraní a jej kód bude vykonaný iba jeden krát na začiatku. Po tejto inicializácii bude program pokračovať v procedúre, ktorá je vykonávaná cyklicky, napovedá to aj jej názov "loop", čo v angličtine znamená slučka. Údaje zo senzorov budú čítané postupne a po každom čítaní budú spracované, alebo priamo odoslané prostredníctvom Bluetooth modulu. Rozdielne sa bude totiž pristupovať k údajom z jednotlivých senzorov. Údaje z akcelerometra bude potrebné ihneď odoslať, no v prípade ostatných senzorov je vhodnejšie vykonať niekoľko meraní a následne vytvoriť priemer hodnôt, ktorý bude odoslaný. Odosielané údaje budú vo formáte String, ktorý bude umožňovať jednoduchú formu spracovania v aplikácii Android. Pre realizáciu čítania zo senzorov budú v cykle volané procedúry, ktoré budú vytvorené pre čítanie z jednotlivých pripojených senzorov. Odosielanie cez Bluetooth rozhranie bude rovnako oddelené v samostatnej procedúre. Tá bude odosielať údaje prijaté prostredníctvom argumentu procedúry.

Pre synchronizáciu komunikácie dosky Arduino a Android zariadenia bude súčasťou odosielaného paketu aj identifikačné číslo, ktoré bude priradené danému senzoru. Toto číslo umožní identifikáciu údajov a ich uloženie v platnom formáte pre daný senzor.

Programový kód vykonáva takéto čítanie a odosielanie údajov neustále počas zapojenia napájania. Ďalšie spracovanie údajov získaných so senzorov bude realizovať aplikácia na zariadení Android.



Obrázok 9 Diagram Arduino aplikácie.

Obrázok vyššie zobrazuje správanie Arduino aplikácie. Inicializácia na začiatku diagramu znázorňuje iniciovanie všetkých potrebných premenných, ktoré program ďalej využíva. Slučka sa vykonáva neustále v cykle. V prípade uplynutia časovača pre daný senzor bude volaná procedúra na čítanie údajov z tohto senzoru. V závislosti od typu senzoru budú údaje buď priamo odoslané, alebo ďalej spracovávané. V prípade akcelerometra bude ihneď po čítaní údaj odoslaný. Ostatné senzory budú vykonávať priemerovanie niekoľkých údajov a až následne bude tento priemer odosielaný. Diagram preto znázorňuje kontrolu posledného merania, ktorá znamená výpočet priemeru a odoslanie. Ostatné merania sa jednoducho uložia do pomocnej premennej pre spracovanie v poslednom meraní danej sekvencie.

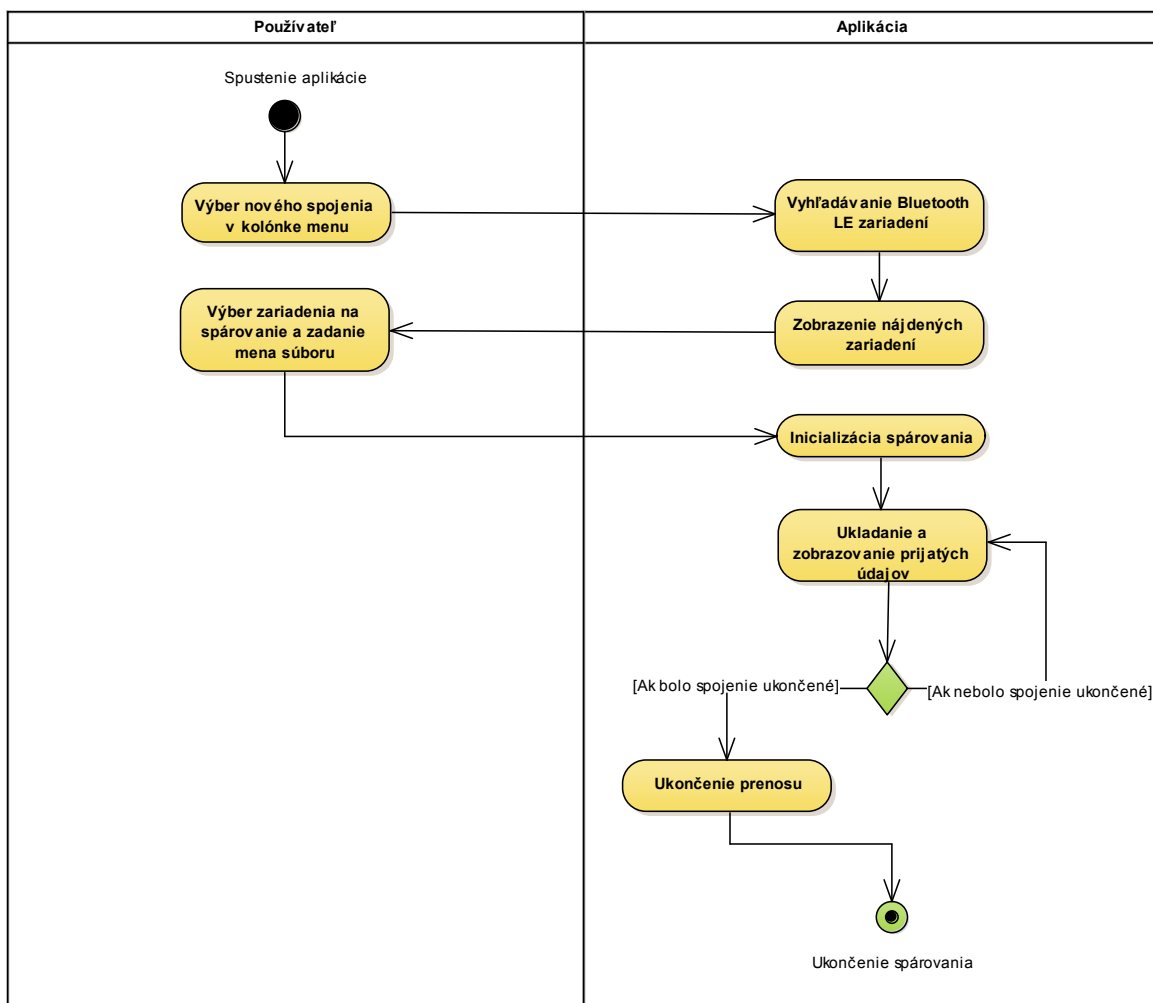
3.5.3 Android aplikácia

Druhá časť senzorového systému bude spracovávať údaje získané zo senzorov a poskytovať ich iným aplikáciám navrhnutým rozhraním. V tomto prípade bude rozhraním súbor vo formáte CSV. Veľmi dôležitou súčasťou je ukladanie údajov do externej pamäte

a teda na pamäťovú kartu. Dôvodom je predpoklad využívania tejto aplikácia a teda aj zariadenia v extrémnych podmienkach, kde môže dôjsť k poškodeniu mobilného zariadenia a tak aj strate údajov. Ukladanie údajov na pamäťovú kartu tak predstavuje bezpečnejšie riešenie pre ochranu nameraných údajov. Po spustení aplikácie bude potrebná inicializácia spojenia s Bluetooth modulom pripojeným k doske Arduino. Po úspešnom spojení bude potrebné vytvorenie súboru pre zápis nameraných údajov. Používateľ si bude môcť okrem zápisu údajov vybrať aj výpis údajov priamo v aplikácii.

Úspešným dokončením všetkých úvodných nastavení sa dostane aplikácia do režimu čítania údajov prijatých cez Bluetooth rozhranie. Primárnou úlohou bude ukladanie získaných informácií do súboru k neskoršej analýze. A teda po úspešnom prijatí platnej a autentickej informácie bude nasledovať zápis do súboru. V prípade, že pri inicializácii nebol vybraný špeciálny režim, prijaté údaje budú zobrazené v prehľadnej forme aj priamo v aplikácii. Čítanie údajov bude prebiehať cyklicky bez prerušenia. Prerušenie bude realizované vstúpením používateľa do programu stlačením tlačidla pre zastavenie komunikácie.

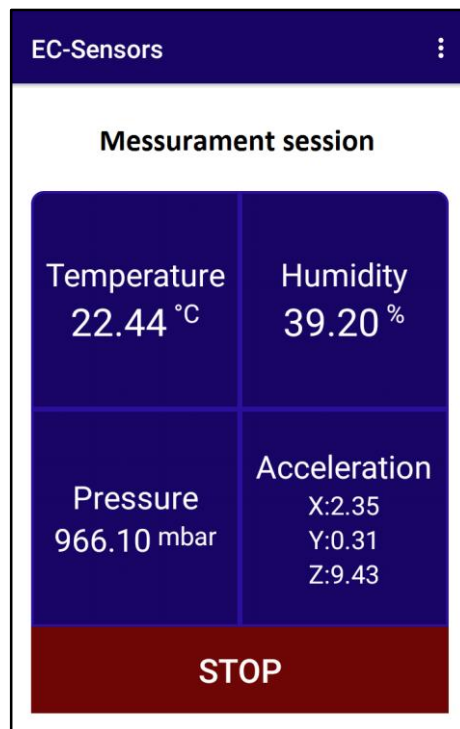
Okrem toho bude aplikácia umožňovať priamo spracovávať súbory s uloženými údajmi. V jednoduchom zozname zobrazí všetky doterajšie merania aj s dátumom vytvorenia. Používateľ bude vedieť vymazávať tieto súbory a zobrazovať ich obsah s možnosťou výberu konkrétneho typu údajov podľa potreby.



Obrázok 10 Priebeh vytvorenia spojenia a záznamu údajov.

3.5.4 Používateľské rozhranie Android aplikácie

Používateľské rozhranie je navrhované s dôrazom na jednoduchosť a prehľadnosť. Práca s aplikáciou by mala slúžiť na jednoduchú inicializáciu komunikácie a základné operácie nad súbormi. Po začiatku merania s výpisom údajov v aplikácii budú zobrazované v navrhnutom formáte podľa obrázku nižšie. Používateľ bude neustále vidieť názov prebiehajúceho merania a aktuálne údaje z príslušného senzoru, bude mať tak neustály prehľad nad zaznamenávanými údajmi.



Obrázok 11 Používateľské rozhranie počas prebiehajúceho merania.

4 Implementácia

Táto kapitola opisuje implementáciu oboch aplikácií podľa vypracovaného návrhu z predchádzajúcej kapitoly. Pre implementáciu Android aplikácie bolo použité vývojové prostredie Android Studio 1.4. Aplikácia bola implementovaná a otestovaná na zariadení Sony Z3 Compact s operačným systémom Android 5.1.1.

Implementácia Arduino aplikácie bola vypracovaná vo vývojovom prostredí Arduino IDE 1.6.6. Pre realizáciu senzorového modulu boli použité senzory teploty KTY81-120, vlhkosti HTU21D, tlaku BMP180 a akcelerometer MMA8452Q. Na odosielanie údajov bol použitý Bluetooth LE modul HM-10.

4.1 Arduino aplikácia

Aplikácia pre platformu Arduino zabezpečuje meranie údajov pomocou pripojených senzorov. Na základe vykonanej analýzy som sa rozhodol použiť senzory s možnosťou pripojenia cez I²C protokol. Výnimkou je napokon senzor teploty, pretože pôvodný senzor, ktorý bol pre tento projekt vybraný potreboval čas na zotavenie 250 milisekúnd po každom meraní. To predstavovalo problém v mojom riešení, pretože by bolo obmedzené meranie z akcelerometru s potrebnou frekvenciou. To však nebolo jediným problémom, vzhľadom na to, že projekt bude využitý pre vypustení stratosférického balónu, je potrebný maximálny rozsah senzorov. Termistor s analógovým výstupom nám umožňuje meranie až do -55 °C.

Senzory s pripojením cez protokol I²C

- HTU21D - senzor vlhkosti
- BMP180 - senzor barometrického tlaku
- MMA8452Q - akcelerometer

Termistor s analógovým výstupom

- KTY81-122

Okrem senzorov je k doske Arduino pripojený aj Bluetooth LE modul s označením HM-10, ktorý zabezpečuje odosielanie údajov. Komunikáciu zabezpečuje protokol UART opísaný v kapitole Analýza.

Pri implementácii tejto aplikácie som využil dostupné knižnice pre prácu s pripojenými zariadeniami, ale aj iné, pre správne fungovanie programu.

4.1.1 Zoznam využívaných knižníc

- avr/wdt.h
- Wire.h
- Timer.h
- SFE_BMP180.h
- HTU21D.h

- SparkFun_MMA8452Q.h

Knižnicu *avr/wdt.h* využívam na zabezpečenie spoľahlivosti programu pri vyskytnutí sa problému. Inicializácia s intervalom 8 sekúnd zabezpečí, že v prípade, ak sa vo vykonávaní programu vyskytne chyba a zasekne sa na dobu aspoň 8 sekúnd, celý program bude reštartovaný a spustený od funkcie *setup()* odznova. Počítadlo, ktoré zabezpečuje sledovanie 8 sekundového intervalu je potrebné vo vykonávaní programu pravidelne reštartovať, takto dáva program vedieť, že sa vykonáva bez problémov a neustále tak predchádza možnému reštartovaniu.

Knižnica *Wire.h* slúži na prácu so seriálovým vstupno-výstupným rozhraním na ktoré je pripojený Bluetooth modul.

Knižnica *Timer.h* umožňuje viacerými funkciami zabezpečovať časovanie vykonávania funkcií podľa potreby. Ja v mojom programe používam funkciu *every()*, ktorej parametrami sú časový údaj v milisekundách a názov funkcie, ktorá má byť volaná. Časový údaj udáva v akej perióde má byť volaná funkcia, ktorej názov je určený ako druhý parameter funkcie *every()*.

Knižnicu *SFE_BMP180.h* implementoval Mike Grusin zo spoločnosti SparkFun Electronics. Táto knižnica umožňuje jednoduché meranie barometrického tlaku, ale aj iných veličín zo senzoru BMP180.

Knižnicu *HTU21D.h* implementoval Nathan Seidle zo spoločnosti SparkFun Electronics. Knižnica umožňuje jednoduchú prácu so senzorom na meranie vlhkosti HTU21D.

Knižnicu *SparkFun_MMA8452Q.h* implementoval Jim Lindblom zo spoločnosti SparkFun Electronics. Knižnica umožňuje jednoduché meranie zrýchlenia z akcelerometru MMA8452Q.

4.1.2 Funkcie programu

Funkcia *void setup()* zabezpečuje inicializáciu všetkých potrebných súčastí programu. Je to napríklad sériové rozhranie, funkcia knižnice *avr/wdt.h*, ktorá spúšťa fungovanie "watch-dog" časovača, sledujúceho správne vykonávanie programu. Ďalej sú inicializované všetky objekty senzorov, ktoré sú pripojené cez protokol I²C. Na konci tejto funkcie definujem za pomoci funkcie *every()* z knižnice *Timer.h* všetky volania implementovaných funkcií. Funkcie zabezpečujúce merania teploty, vlhkosti a tlaku sa vykonávajú s periódou 1000 milisekúnd a meranie z akcelerometra prebieha každých 100 milisekúnd. Nakoniec sú tu inicializované aj hodnoty, ktoré využívam pri realizovaní priemerovania nameraných údajov zo senzorov teploty, vlhkosti a tlaku.

Funkcia *void loop()* vďaka využitiu funkcie časovača z knižnice *Timer.h* obsahuje iba volanie aktualizovania časovača a to funkciou *update()*. Funkcia *loop()* je totiž zodpovedná za beh programu, pretože je vykonávaná neustále dookola. Vo väčšine programov by obsahovala programový kód, no vzhľadom na to, že časovať zabezpečuje

volanie potrebných funkcií v požadovaných intervaloch, táto funkcia obsahuje iba jeho aktualizovanie.

Funkcia *sendMessage()* zabezpečuje odosielanie údajov seriálovým rozhraním. Údaje sú vložené do funkcie ako jej parametre a je to číslo vo formáte *float*, ktoré reprezentuje hodnotu nameranú senzorom a identifikátor typu *string*, ktorý umožňuje identifikáciu hodnoty v Android aplikácii. Súčasťou funkcie je aj prevod odosielanej hodnoty do formátu *string* a zistenie jeho dĺžky, ktorá je tiež odosielaná ako súčasť správy pre ľahšie spracovanie prenášaného údaju. Správy sú dopĺňané pomocou využitia poľa prvkov typu *string*, jedná sa o súvislé biele znaky, ktoré sú v poli umiestnené v danom prvku v rovnakom počte, ako je ich index v poli. Tento fakt následne uľahčuje jednoduchý výpočet indexu prvku, ktorý má byť do správy doplnený pre doplnenie veľkosti.

Funkcia *read_temperatureAnalog()* realizuje čítanie hodnôt z termistoru a ich následné spracovanie. Hneď po čítaní prebieha prevod analógovej hodnoty do digitálnej a to na základe kódu implementovaného na webovej stránke pre Arduino platformy¹⁰. Nasleduje kontrola hodnoty s rozsahom platných hodnôt, ktoré sú určené pre daný senzor. Súčasťou je aj algoritmus zabezpečujúci priemerovanie údajov, ktorý bude opísaný nižšie. Okrem toho má táto funkcia na starosti aj volanie funkcie, ktorá zabezpečuje nulovanie "watch-dog" časovača, ktorí sleduje program a chráni ho pred dlhodobým znefunkčnením.

Funkcia *read_humidity()* realizuje čítanie hodnôt zo senzoru vlhkosti a kontrolu platnosti údajov. Rovnako ako predchádzajúca funkcia realizuje aj priemerovanie údajov s následným volaním funkcie *sendMessage()*. Táto funkcia je volaná každých 20 sekúnd po vykonaní priemeru z dvadsiatich nameraných hodnôt.

Funkcia *read_pressure()* realizuje čítanie hodnôt zo senzoru tlaku a kontrolu platnosti nameraných údajov. Rovnako ako predošlé funkcie realizuje rovnaký algoritmus na priemerovanie hodnôt s odosielaním každých 20 sekúnd. Následne je volaná funkcia *sendMessage()*.

Funkcia *read_acceleration()* realizuje volanie jednotlivých funkcií *read_accelerationX()*, *read_accelerationY()*, *read_accelerationZ()* ktoré vykonávajú čítanie zrýchlenia na jednotlivých súradnicových osiach a následné volanie funkcie *sendMessage()*.

4.1.3 Výpočet priemeru

Výpočet priemeru je realizovaný štyrmi pomocnými premennými pre každý zo senzorov, ktorého hodnoty sa pred odosielaním najprv priemerujú. Rozhodol som sa totiž realizovať merania so senzorov teploty, vlhkosti a tlaku v sekundových intervaloch, z ktorých po 20 meraniach vytvorím priemer. Až tento údaj bude odoslaný na Android zariadenie a uložený pre ďalšiu analýzu údajov

¹⁰ <http://playground.arduino.cc/Main/Kty81-110>

V každom prechode funkcií pre príslušné senzory sa vykonáva porovnanie počítadla, ktoré sa pri každom prechode zníži o hodnotu jedna. Ak je počítadlo rovné hodnote 20, nastaví sa aktuálna hodnota aj ako maximum aj ako minimum danej sekvencie meraní a to z dôvodu, že je prvou v hodnotou v tejto sekvencii. Pri ostatných 19-nástich prechodoch sa kontroluje, či aktuálna hodnota nie je doterajším maximom alebo minimom.

V prvých 19-nástich prechodoch sa zároveň pripočítava aktuálna hodnota k súčtu doterajších nameraných v danej sekvencii. V poslednom meraní sa táto hodnota rovnako pričíta, no vykoná sa aj odčítanie maxima a minima od celkového súčtu s cieľom odstránenia extrémov, či prípadných nepresných meraní senzoru. V tomto poslednom meraní je vypočítaný priemer odoslaný a pomocné premenné sú opäť inicializované na počiatočné hodnoty.

Realizáciu priemerovania formou využitia pomocných premenných som sa rozhodol z dôvodu šetrenia zaťažnosti výpočtovej jednotky. Práca s poľom totiž výrazne viac zaťažuje procesor.

4.2 Android aplikácia

Android aplikácia je implementovaná vo vývojovom prostredí Android Studio 1.4. Cieľovou Android API verziou je v súčasnosti najnovšia API 23, no minimálna verzia na ktorej je možné túto aplikáciu využívať je API 21.

V aplikácii je okrem štandardných knižníc zahrnutá aj knižnica umožňujúca jednoduchý zápis a čítanie zo súborov vo formáte CSV, jej importovanie do projektu zabezpečuje nasledujúci príkaz: `compile files('libs/opencv-3.7.jar')`.

4.2.1 Povolenia aplikácie

Vzhľadom na to, že aplikácia vyžaduje prístup k viacerým zabezpečeným súčastiam zariadenia, musí v súbore *AndroidManifest.xml* obsahovať niekoľko povolení. Vďaka ich prítomnosti v tomto súbore budú aplikácii pri inštalácii pridelené práva využívať prístup k daným súčastiam.

Najdôležitejším súčastou pre príjem dát je prístup k využívaniu Bluetooth rozhrania. Ak chce aplikácia využívať prístup k nejakému hardvérovému prvku zariadenia, musí to deklarovať príkazom *uses-feature*. Takto moja aplikácia deklaruje prístup k Bluetooth LE rozhraniu:

```
<uses-feature android:name="android.hardware.bluetooth_le"  
android:required="true"/>
```

Okrem toho musí byť obsiahnuté aj povolenie pracovať s Bluetooth rozhraním v nasledujúcom tvare:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

Zároveň aplikácia vyžaduje možnosť zapínania Bluetooth rozhrania priamo z aplikácie. Túto možnosť využíva, ak sa používateľ pokúsi vyhľadávať zariadenia, no nie je na zariadení zapnuté rozhranie Bluetooth. Nasleduje žiadosť o povolenie toto rozhranie zapnúť a aplikácia môže pokračovať vo vyhľadávaní prítomných zariadení. Prístup k zapínaniu Bluetooth rozhrania, a ďalším jeho nastaveniam je zabezpečený nasledujúcim príkazom:

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

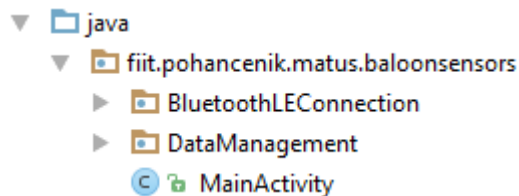
Okrem toho aplikácia pracuje s externou pamäťou, ku ktorej prístup vyžaduje rovnako obsiahnutie povolení. Tieto riadky zabezpečujú aplikácii povolenia na prístup k čítaniu a zapisovaniu do externej pamäte:

```
<uses-permission  
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />  
<uses-permission  
android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

4.2.2 Rozdelenie do balíkov

Programový kód je hierarchicky rozdelený do balíkov podľa najzákladnejších kritérií. Tie sú v tejto aplikácii nasledovné:

- pripojenie a správa prenosu údajov Bluetooth rozhraním,
- správa prijatých údajov a ich prehliadanie.



Obrázok 12 Hierarchia balíkov tried Android aplikácie.

Hlavnú aktivitu s pomenovaním *MainActivity* som sa rozhodol nechať nezariadenú, pretože jej funkcionality zasahuje do oboch zatriedených balíkov. Tento fakt pramení z jej podstaty, pretože tvorí funkčné jadro celého programového kódu. Je východiskovým bodom pre inicializáciu Bluetooth LE spojenia a zároveň spravovania údajov.

Pri úspešnom vyhľadaní a výbere zariadenia na spárovanie, ktoré realizuje aktivita *ScanBLEDevicesActivity* nasleduje pripojenie k zariadeniu. To je iniciované volaním metódy *connect()* z triedy *BluetoothLEService*. Parametrom metódy je adresa Bluetooth LE zariadenia. Po úspešnom spojení trieda *MainActivity* neustále spracováva prijaté údaje pomocou metódy *processData()*. Tá zabezpečuje aktualizáciu údajov v používateľskom rozhraní a zároveň volá metódu triedy *FileManager*, ktorá zabezpečuje zápis údajov do súboru.

Balík *BluetoothLEConnection*

Balík *BluetoothLEConnection* obsahuje triedy, ktorých základom je vzorový program pre prácu s rozhraním Bluetooth LE. Dostupný je priamo vo vývojovom prostredí Android Studio pod názvom BluetoothLeGatt a bol základom tejto časti aplikácie. Triedy boli upravené s cieľom prispôsobenia pre komunikáciu s Bluetooth LE modulom HM-10.

Trieda *SampleGattAttributes* definuje základné GATT atribúty pre komunikáciu s modulom HM-10. Využívaná je v triede *BluetoothLEService*, ktorej úlohou je zabezpečenie vytvorenia spojenia s Bluetooth modulom a prijímanie dát. Rozširuje pritom komponent *Service*, ktorý je využívaný v aplikáciách pri dlhodobých operáciách bez interakcie s používateľom.


Súčasťou tohto balíka je aj trieda, ktorá je rozšírením Android *ListActivity* a nazýva sa *ScanBLEDevicesActivity*. Táto aktivita zabezpečuje inicializáciu vyhľadávania Bluetooth LE zariadení a ich následné zobrazenie v liste. Po výbere zariadenia a následnom zadaní názvu merania táto aktivita odosiela meno zariadenia vybraného pre komunikáciu, jeho adresu a názov merania späť do aktivity z ktorej bola spustená.

Balík *DataManagement*

V tomto balíku sa nachádzajú triedy zabezpečujúce správu prijatých údajov. Pre uchovávanie informácií o jednotlivých meraniach som sa rozhodol využiť jednoduchú implementáciu databázy. Tabuľka obsahuje základné informácie o meraní a jeho súbore. Tieto údaje umožňujú prehľadné zobrazenie všetkých doterajších meraní a vykonávanie akcií nad súbormi.

Vytvorený tabuľka obsahuje nasledujúce údaje:

- cesta k súboru s nameranými údajmi,
- samotné meno merania,
- dátum a čas vytvorenia merania.

sessions	
	id INT PK
	filepath TEXT
	name TEXT
	date TEXT
Indexes	
	PRIMARY id

Obrázok 13 Tabuľka pre uchovávanie údajov o meraní.

Implementácia databázy sa nachádza v triede *DatabaseHandler*, ktorá zabezpečuje všetky operácie nad databázou.

Trieda *SessionInfo* slúži na prehľadné vytváranie nového záznamu o meraní v databáze.

Trieda *FileManager* zabezpečuje všetky operácie s pamäťou. Na začiatku je potrebné získať prístup k externej pamäti zariadenia. V súčasných Android zariadeniach je však ako externá pamäť vnímaná aj interná pamäť. Jedná sa o vytvorenie simulovanej externej pamäte. Pri získavaní priečinku aplikácie sa *FileManager* zameriava na skutočnú externú pamäť a teda pamäťovú kartu. Ak je pripojená k zariadeniu, vráti priečinok na tejto pamäťovej karte. V prípade, že pamäťová karta nie je prítomná v zariadení, aplikácia bude zapisovať údaje do simulovanej externej pamäte. Ďalej táto trieda zabezpečuje vytváranie súborov a ich mazanie. V neposlednom rade sa táto aplikácia stará o zápis a čítanie údajov zo súborov vo formáte CSV.

Aktivita *ShowAllSessionsActivity* zabezpečuje zobrazenie všetkých doterajších meraní, ktoré sú prítomné v pamäti. V zozname meraní sa nachádzajú prvky obsahujúce meno merania a dátum jeho vytvorenia. Vďaka volaniam metód triedy *FileManager* umožňuje používateľovi vymazávať doteraz realizované merania.

Aktivita *ShowSessionDataActivity* umožňuje zobrazovanie údajov zaznamenaných v jednotlivých meraniach. Rovnako ako predošlá aktivita k tomu využíva metódy triedy *FileManager*, ktorá umožňuje čítanie súborov. V tejto aktivite sa nachádza aj privátna trieda *MyAsyncTask*, ktorá je rozšírením triedy *AsyncTask* a zabezpečuje asynchrónne vykonanie operácie čítania zo súboru. Táto trieda totiž obsahuje metódu *doInBackground()*, ktorá zabezpečuje čítanie zo súboru a následne po vykonaní čítania je zavolaná metóda *onPostExecute()*, ktorá prečítané údaje zobrazí v používateľskom rozhraní. V aplikácii je možné zobraziť iba údaje zo senzorov teploty, vlhkosti a tlaku. Údaje zaznamenané akcelerometrom s frekvenciou 10Hz predstavujú obrovské množstvo, ktoré spôsobuje problémy a zasekávanie používateľského rozhrania. Rozhodol som sa preto tieto údaje nezobrazovať.

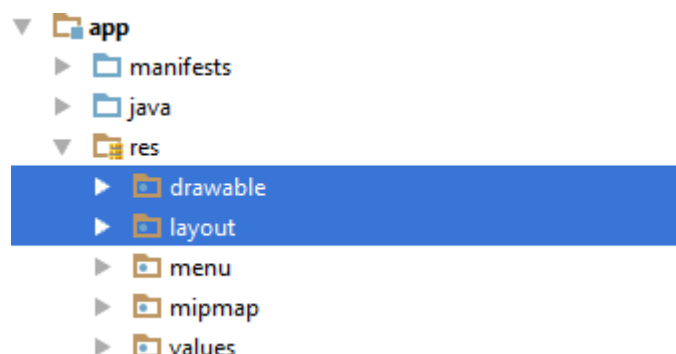
Trieda *CheckBoxDialogAlertFragment* zabezpečuje možnosť výberu zobrazenia údajov podľa potreby používateľa zobrazením dialógového okna. Pri prenose získaných údajov pritom využíva implementované rozhranie *MyDialogFragmentListener*, ktoré zabezpečuje prenos údajov späť do aktivity *ShowSessionDataActivity*.

Pri čítaní zo súboru je využívaný aj enumerátor s názvom *SensorType*. Jeho úlohou pri čítaní je odlíšiť jednotlivé údaje a namiesto identifikačného čísla zobrazí v používateľskom rozhraní údaj s textovým opisom prečítanej hodnoty.

4.2.3 Grafická časť

Grafickú časť a rozloženie prvkov na obrazovke zariadenia majú na starosti v android aplikáciách súbory vo formáte XML. Nachádzajú sa v priečinku *layout*, ktorý je jedným z vyznačených na obrázku nižšie. Nachádzajú sa v ňom všetky súbory definujúce vzhľad aplikácie z pohľadu rozloženia prvkov v daných aktivitách. Druhým vyznačeným

priečinkom je *drawable*, ktorý obsahuje importované obrázky, ktoré sú v aplikácii využívané. Okrem toho obsahuje tiež súbory vo formáte XML, ktoré definujú farebné zobrazenie viacerých prvkov vyskytujúcich sa v jednotlivých obrazovkách aplikácie.



Obrázok 14 Hierarchia aplikácie s dôrazom na poukázanie grafickej časti.

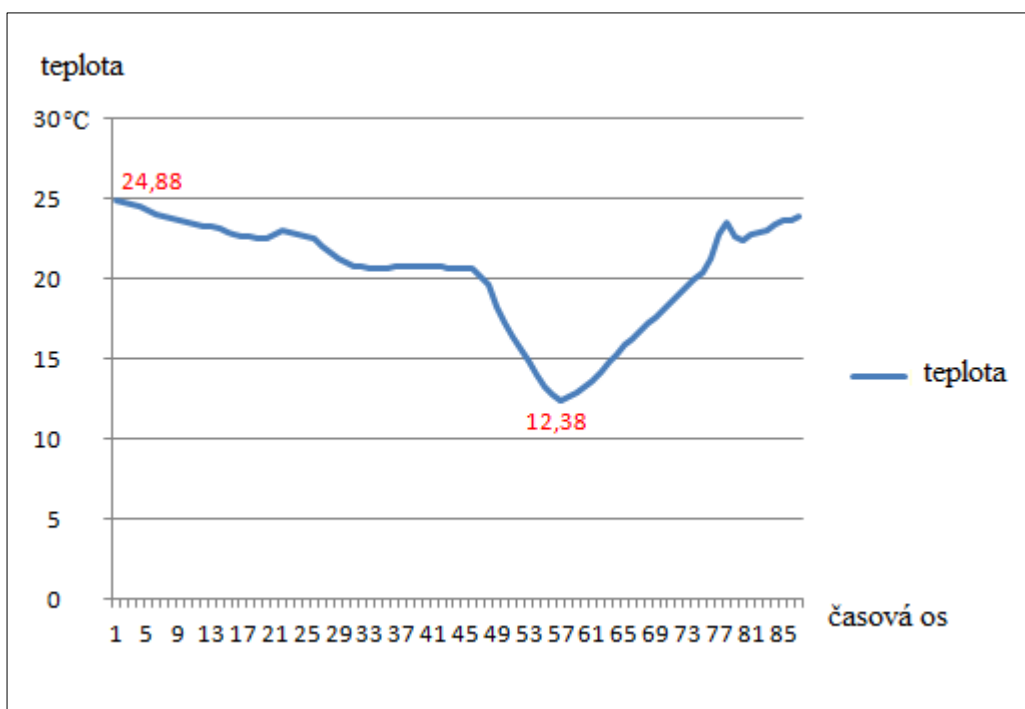
4.3 Overenie riešenia

Testovanie oboch aplikácií bolo realizované počas celého vývoja projektu. Na začiatku bolo najdôležitejším cieľom vytvorenie funkčného prenosu údajov medzi oboma zariadeniami. Testovanie správnosti odosielaných údajov z Arduino aplikácie som spočiatku testoval použitím Android Bluetooth LE terminálu. Neskôršie testovania už boli realizované medzi oboma mnou vytvorenými aplikáciami a zameriaval som sa najmä na správnosť spracovania údajov.

Pri testovaní sa vyskytovali problémy v prípade odosielania údajov s príliš vysokou frekvenciou. Bluetooth modul HM-10 totiž odosielané správy často spájal a tak znemožňoval správne spracovanie v Android aplikácii. Po odladení komunikácie som začal realizovať rôzne testy meraní zo senzorov. Boli vykonané niekoľkohodinové nočné merania z balkóna rodinného domu, ktoré otestovali najmä senzory teploty a vlhkosti.

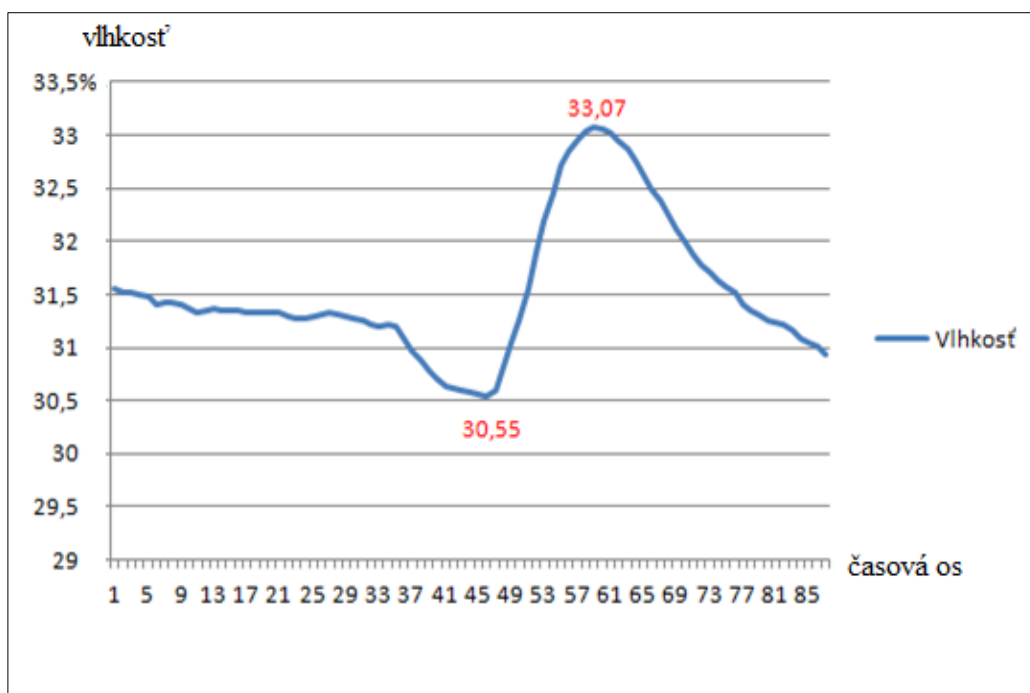
Ďalšia fáza testovania patrila správnosti spravovania údajov a teda ich zobrazovania a vymazávania. Pri zobrazovaní údajov v aplikácii som zistil, že pri mnohohodinových meraniach spôsobuje množstvo údajov zobrazovaných v aplikácii pomalšiu reakciu zariadenia. Množstvo údajov, ktoré musia byť zobrazené používateľovi totiž aplikácia vypisuje pomalšie.

Finálne testovanie bolo realizované v podobných podmienkach ako budú pri lete stratosférického balóna, pre ktorý je tento projekt vyvíjaný. Vykonané bolo meranie počas letu lietadlom pre parašutistov a výsledky meraní senzorov boli spracované v grafoch znázornených nižšie. Priebeh merania trval približne 30 minút.



Obrázok 15 Priebeh teploty počas testu.

Graf údajov, zaznamenaných teplotným senzorom zobrazuje priebeh kolísania teploty. Na x-ovej osi sú zobrazené poradové čísla údajov a na y-ovej je znázornená teplota v stupňoch Celzia. Na začiatku merania je teplota na zemi pred štartom lietadla. Postupným stúpaním lietadla až do výšky 3 km teplota klesla až na hranicu 12,38 °C. Klesanie lietadla bolo rýchlejšie ako to bolo pri stúpaní a tak je na grafe vidno rýchlejšie stúpanie hodnôt.



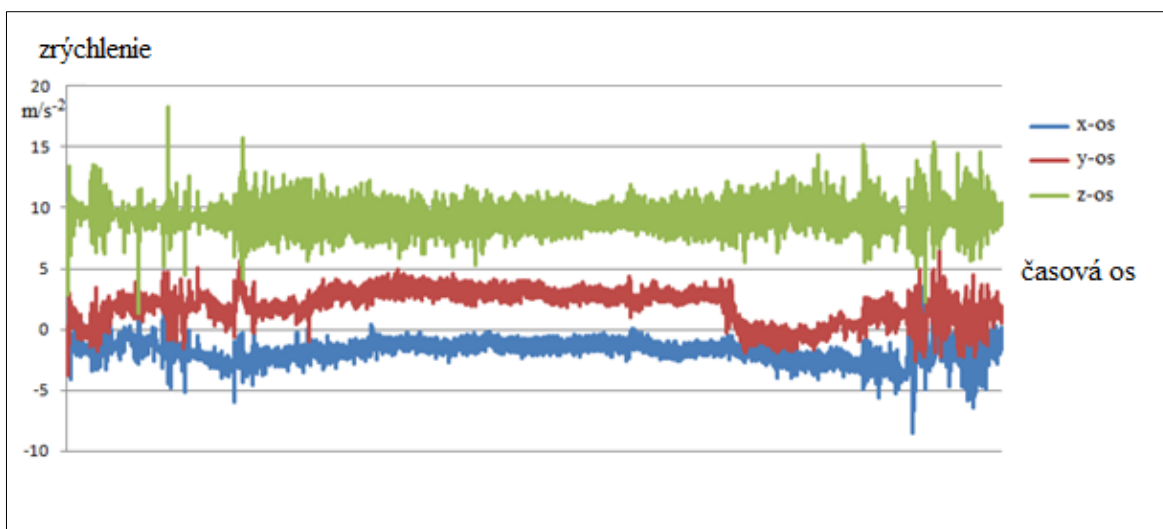
Obrázok 16 Priebeh vlhkosti počas testu.

Graf údajov, zaznamenaných senzorom vlhkosti zobrazuje priebeh hodnôt počas merania. Z grafu vidíme, že vlhkosť sa počas letu lietadlom príliš nemenila. Na x-ovej osi sú zobrazené poradové čísla údajov a na y-ovej je znázornená vlhkosť vzduchu v percentách. Na grafe sú zobrazené minimum a maximum merania.



Obrázok 17 Priebeh atmosférického tlaku počas testu.

Graf údajov, zaznamenaných tlakovým senzorom zobrazuje priebeh hodnôt počas merania. V prípade atmosférického tlaku je tvar grafu podobný ako pri teplote. Stúpaním lietadla totiž klesala hodnota atmosférického tlaku. Na x-ovej osi sú zobrazené poradové čísla údajov a na y-ovej je znázornený atmosférický tlak v milibaroch.



Obrázok 18 Priebeh zrýchlenia na všetkých osiach.

Na grafe z akcelerometra sú zobrazené priebehy zrýchlenia na všetkých osiach.

4.3.1 Záver testovania

Testovanie prebehlo bez problémov a všetky údaje boli zaznamenané korektne. V prípade využitia údajov z akcelerometra pre výpočet výšky, ukázalo testovanie nedostatky. Pri spracovaní údajov sme dospeli k zisteniu, že naklonenie lietadla spôsobuje premietanie stúpania aj do iných súradnicových osí. Výsledkom výpočtu výšky počas letu tak boli skreslené údaje, ktoré spôsobilo rozdielne naklonenie lietadla pri stúpaní a klesaní. Správnosť fungovania oboch častí senzorového systému však dopadla podľa očakávaní.

5 Záver

V úvodnej časti, venovanej analýze zadanej oblasti sú opísané jednotlivé skupiny senzorov v Android zariadeniach. Následne sú opísané v súčasnosti najvyužívanejšie technológie pre vytváranie sietí senzorov a prenos údajov medzi týmito zariadeniami. Ďalej nadväzuje opis existujúcich riešení, ktoré tieto technológie využívajú. Ako inšpirácia pre tvorbu môjho senzorového systému slúžila aplikácia SensoDuino, ktorá zabezpečuje komunikáciu s doskou Arduino prostredníctvom Bluetooth modulu. Smer toku údajov je v tejto aplikácii však opačný ako v mojom projekte.

Vývoj obsahu dokumentu naznačuje smerovanie riešenia tohto bakalárskeho projektu a preto sú ďalej opísané zariadenia, ktoré by mohli byť použité pri vytváraní senzorového systému pre platformu Android. Po dôkladnom porovnaní bola pre implementáciu vybraná doska Arduino Mega 2560, ktorá najlepšie spĺňala požadované kritériá.

V poslednej časti analýzy sú opísané možnosti pripojenia senzorov k doskám Arduino. Opísané sú protokoly, ktoré umožňujú prenos údajov medzi doskami Arduino a rôznymi periférnymi zariadeniami.

Z poznatkov získaných v analýze bol vypracovaný návrh aplikácií, ktoré spolu tvoria senzorový systém. Cieľom bolo vytvoriť Android aplikáciu, ktorá umožňuje zdieľanie údajov s inými Android aplikáciami. Ako rozhranie bol vybraný súbor vo formáte CSV, ktorý umožňuje jednoduchú identifikáciu údajov, na základe príslušného ID senzoru. Pre komunikáciu so senzormi bola vybraný protokol I²C. Počas implementácie som zistil, že malý rozsah teplotného senzoru by spôsobil obmedzenia pri meraní počas letu stratosférického balónu. Použitý bol teda termistor s analógovým pripojením. Pre komunikáciu s Bluetooth LE modulom bola použitá komunikácia cez sériové rozhranie a technológiu UART, ktorá je implementovaná na doskách Arduino. Vzhľadom na cieľ vytvoriť senzorový systém s čo najmenším odberom energie, pre prenos údajov bola použitá verzia Bluetooth 4.0 (Bluetooth LE) a teda modul HM-10.

Výstupom tohto bakalárskeho projektu sú dve aplikácie. Prvou je Arduino aplikácia umožňujúca meranie teploty, vlhkosti, tlaku a zrýchlenia prostredníctvom pripojených senzorov. Získané údaje odosiela prostredníctvom Bluetooth LE modulu. Druhá aplikácia určená pre platformu Android zabezpečuje spracovanie prijatých údajov zo senzorov a ich poskytovanie ďalším aplikáciám. Okrem toho umožňuje aj zobrazovanie aktuálnych údajov priamo v aplikácii okamžite po prijatí.

Pri testovaní senzorového modulu som overoval správnu funkčnosť oboch častí môjho bakalárskeho projektu. Obe časti pracovali bez problémov, no pri spracovaní údajov z letu som zistil, že výpočet výšky z údajov zaznamenaných akcelerometrom nie je možný. Údaje z akcelerometra skresľuje náklon lietadla a stúpanie sa tak prejavuje do všetkých

osí. Riešením tohto problému je implementovanie gyroskopu, vďaka ktorému dokážeme sledovať náklon celého senzorového modulu.

Vypracované zadanie sa podarilo splniť vo všetkých bodoch a projekt bude ďalej využitý pri vypustení stratosférického balóna v spolupráci so Slovenskou organizáciou pre vesmírne aktivity. Projekt bude súčasťou senzorového modulu tohto balóna. Práca obsahuje technickú dokumentáciu a používateľskú príručku.

5.1 Možné vylepšenie

- Implementovanie gyroskopu, pre správny výpočet výšky z údajov akcelerometra
- Zobrazovanie grafov z nameraných údajov priamo v Android aplikácii
- Ukladanie údajov na server

6 Technická dokumentácia

Táto kapitola obsahuje technickú dokumentáciu a opis vybraných častí kódu oboch častí bakalárskeho projektu. Celý projekt je dostupný na nasledujúcej url adrese:

<https://github.com/Empik12/EC-Sensors>

6.1 Dokumentácia k Arduino aplikácii

Celý programový kód Arduino aplikácie sa nachádza v jednom súbore s názvom EC-Sensors-Arduino.ino. Rozdelený je do niekoľkých funkcií, ktoré spoločne tvoria programový kód pre Arduino platformu.

Na začiatku programu je potrebná inicializácia používaných súčastí. Programový kód je vždy spúšťaný od funkcie *setup()*, ktorá túto inicializáciu zabezpečuje.

```
void setup()
{
    Serial.begin(9600);
    Serial.println("REBOOT");
    wdt_enable(WDTO_8S);

    accel.init(); //inicializácia akcelerometra

    if (pressure.begin())// inicializácia tlakového senzoru
        Serial.println("BMP180 init success");
    else
    {
        Serial.println("BMP180 init fail\n\n");
        while(1); // V prípade nesprávnej inicializácie program zastaví
    }
    if(myHumidity.begin()){// inicializácia senzoru vlhkosti
        Serial.println("HTU21D init success");
    }else{
        Serial.println("HTU21D init fail\n\n");
        while(1); // V prípade nesprávnej inicializácie program zastaví
    }
    // Nastavenie časovača na vykonávanie funkcií v zadaných periódach
    t.every(1000, read_temperatureAnalog);
    t.every(1000, read_humidity);
    t.every(1000, read_pressure);
    t.every(100, read_acceleration);

    // inicializácia premenných pre výpočty priemeru
    TemperatureNumber = 0;
    TemperatureCounter = 20;

    HumidityNumber = 0;
    HumidityCounter = 20;

    PressureNumber = 0;
    PressureCounter = 20;
}
```

Hneď po vykonaní tejto funkcie, je v programoch pre Arduino platformy volaná funkcia *loop()*, ktorá je vykonávaná až do konca behu programu cyklicky. Vzhľadom na to, že v programe je používaný časovač, ktorý zabezpečuje volania jednotlivých funkcií, budeme ho musieť neustále aktualizovať. Aktualizácia je vykonávaná práve z funkcie *loop()*.

```
void loop()
{
    t.update(); // Aktualizácia časovača
}
```

Funkcie, ktoré zabezpečujú čítanie údajov zo senzorov teploty, tlaku a vlhkosti na začiatku realizujú čítanie zo senzora a následne vykonávajú rovnaké porovnávanie parametrov. Každá veličina má však vlastné pomocné premenné, ktoré jej umožňujú výpočet priemeru po realizácii dvadsiatich meraní.

Pri čítaní teploty z termistoru musí program vykonávať prevod získaného analógového údaju do digitálnej podoby. Tento výpočet realizuje nasledujúca časť programového kódu.

```
//*****
//táto časť programového kódu je dostupná na url adrese: http://playground.arduino.cc/Main/Kty81-110
unsigned int port = 0;
float temp          = 82;
ADCSRA = 0x00;
ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
ADMUX = 0x00;
ADMUX = (1<<REFS0);
ADMUX |= port;

for (int i=0;i<=64;i++)
{
    ADCSRA|=(1<<ADSC);
    while (ADCSRA & (1<<ADSC));
    temp += (ADCL + ADCH*256);
}

temp /= 101;
temp -= 156;

//*****
```

Čítanie vlhkosti je realizované v jednom riadku vďaka importovanej knižnici pre obsluhu senzoru HTU21D.

```
float humd = myHumidity.readHumidity();
```

Realizácia čítania barometrického tlaku je opäť zložitejšia, keďže používaná knižnica pre obsluhu senzoru BMP180 vyžaduje najprv čítanie teploty týmto senzorom a až následne môže byť odmeraný atmosférický tlak.

```

void read_pressure()
{
    char status;
    double T,P;
    //*****
    //táto časť je implementovaná na základe vzorového príkladu knižnice senzoru BMP180

    status = pressure.startTemperature();
    if (status != 0)
    {

        delay(status);

        status = pressure.getTemperature(T);
        if (status != 0)
        {

            status = pressure.startPressure(3);
            if (status != 0)
            {

                delay(status);

                status = pressure.getPressure(P,T);
                if (status != 0)
                {

```

Takto prebieha čítanie barometrického tlaku, kde po vykonaní tohto kódu sa nachádza číselný údaj v premennej P . Po čítaní údajov nasleduje kontrola a výpočet priemeru, ak sa jedná o posledné meranie v danom intervale. Táto časť kódu je rovnaká pre merania teploty, tlaku a vlhkosti. Ako príklad uvádzam časť kódu pre meranie tlaku.

```

//*****
    if(300 < P && P < 1100 ){

        if(PressureCounter == 20){// ak začína nový cyklus

            PressureNumberMin = P;
            PressureNumberMax = P;

        }else{
            if(PressureNumberMin > P){
                PressureNumberMin = P;
            }
            if(PressureNumberMax < P){
                PressureNumberMax = P;
            }
        }

        if(PressureCounter > 1){//ak je meranie 1-19 iba pripočítaj hodnotu k celkovému súčtu

            PressureNumber = PressureNumber + P;

            PressureCounter--;
        }else{//ak je meranie posledné, opäť pričítaj hodnotu k celkovému súčtu, následne
            //odrátaj minimum a maximum a vypočítaj priemer, potom údaj odošli

            PressureNumber = PressureNumber + P;
            PressureNumber = PressureNumber - PressureNumberMin;
            PressureNumber = PressureNumber - PressureNumberMax;

            float finalPressure = PressureNumber/18;

            sendMessage("[3]", finalPressure);
            PressureCounter=20;
            PressureNumber = 0;
        }}}}

```


Práca s akcelerometrom je trochu odlišná od ostatných senzorov. Nerealizuje sa totiž priemerovanie údajov, no údaje sa okamžite odosielajú. Volanie funkcie *read_acceleration()* zabezpečí volanie funkcií pre čítanie údajov z jednotlivých osí, na ktorých je merané zrýchlenie.

```
void read_acceleration()
{
    if (accel.available())
    {
        accel.read();

        read_accelerationX();
        delay(50);
        read_accelerationY();
        delay(50);
        read_accelerationZ();
    }
}

void read_accelerationX(){

    float numx = accel.cx * 9.81;
    sendMessage("[4]", numx);
}

void read_accelerationY(){

    float numy = accel.cy * 9.81;
    sendMessage("[5]", numy);
}

void read_accelerationZ(){

    float numz = accel.cz * 9.81;
    sendMessage("[6]", numz);
}
```

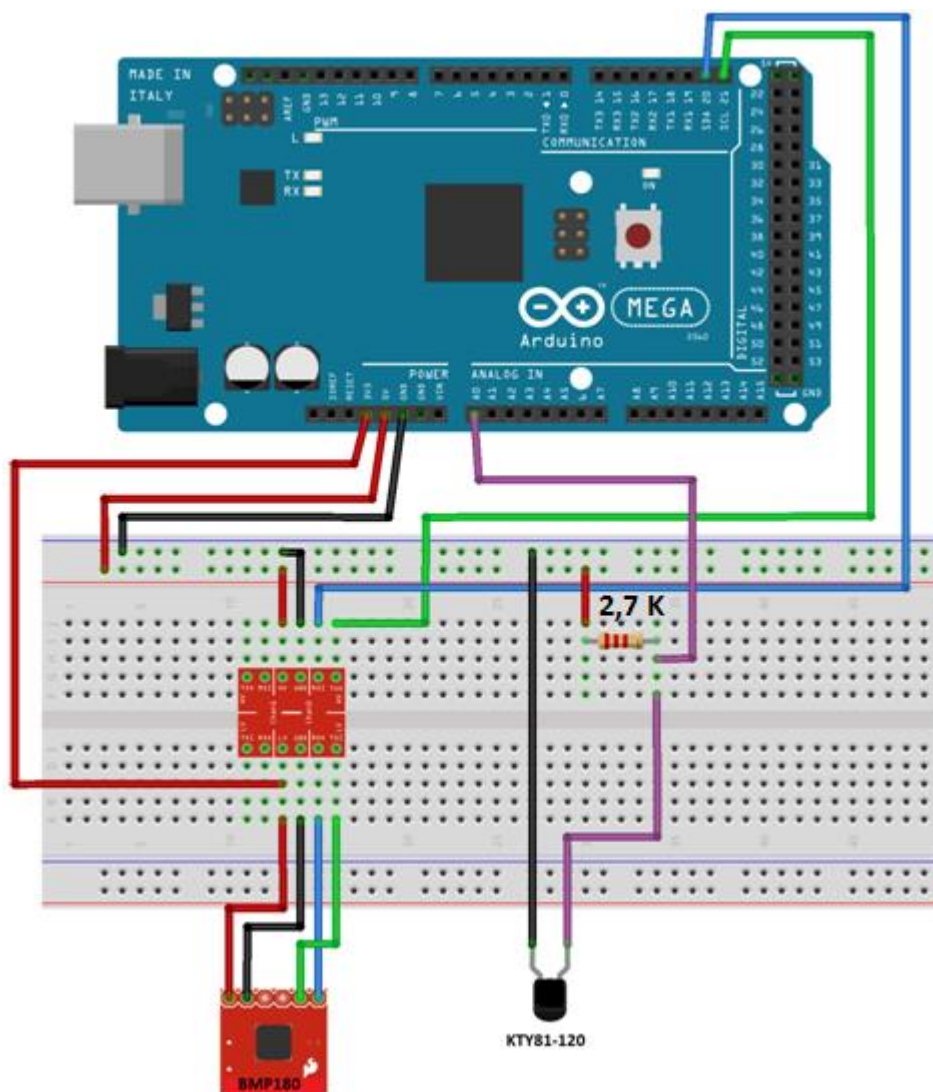
Funkcia pre odosielanie údajov je volaná s parametrom identifikátora odosielaného údaju a samotným údajom.

```
void sendMessage(String ID, float data){

    String Data_N = String(data,2);
    int size = Data_N.length();
    String sizeS = String(size);
    //prvé číslo v správe je identifikátor údaju, nasleduje dĺžka stringu, ktorý obsahuje
    //číselný údaj so senzora, správa je doplnená medzerami a na konci je samotný údaj
    Serial.print(ID+sizeS+spaces[MESSAGE_SIZE-size-4]+Data_N);
}
```

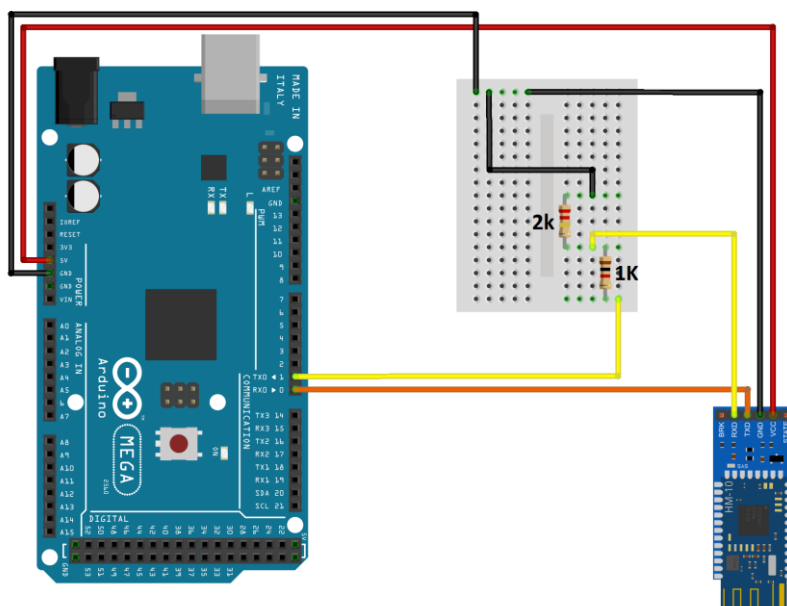
6.1.1 Pripojenie komponentov k doske Arduino

Pre pripojenie zariadení je potrebné využiť prepájacie káble a dosky. Sensory vlhkosti, tlaku a akcelerometer sú pripájané cez protokol I²C. Pre správnu funkciu však potrebujú obojsmerný prevod napätia na linkách, ktoré ich spájajú s doskou Arduino. Použitý je konvertor logických hodnôt od spoločnosti SparkFun. Nasledujúci obrázok znázorňuje pripojenie senzoru tlaku BMP180 ako vzorové pripojenie senzoru cez I²C a analógového termistoru.



Obrázok 19 Zapojenie senzoru BMP180 a termistoru KTY81-120.

Vzhľadom na to, že niektoré pripájacie zariadenia potrebujú pre správne fungovanie napätie 3,3V, používa sa napäťový delič. Na obrázku nižšie je znázornené pripojenie Bluetooth modulu HM-10 k doske Arduino Mega 2560.



Obrázok 20 Zapojenie Bluetooth modulu HM-10.

6.2 Používateľská príručka k Arduino aplikácii

Táto podkapitola obsahuje návod na inštaláciu Arduino aplikácie na dosku Arduino Mega 2560.

6.2.1 Inštalačná príručka

Pre inštaláciu aplikácie na dosku Arduino je potrebné vývojové prostredie Arduino IDE. Je to voľne dostupný softvér prístupný na url adrese:

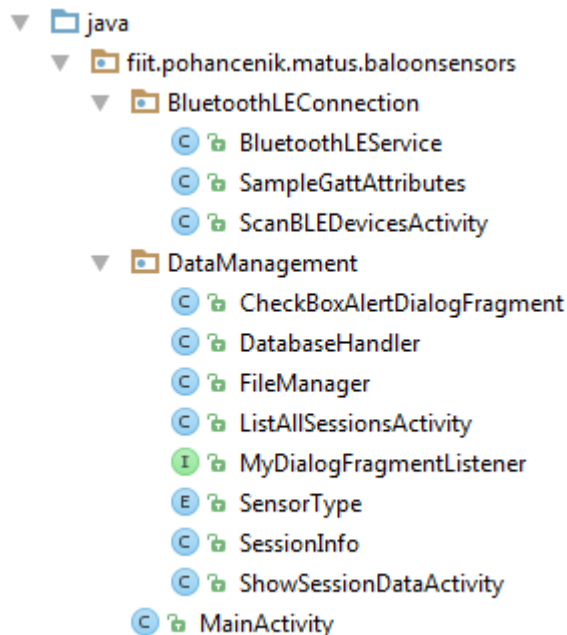
<https://www.arduino.cc/en/Main/Software>

Po úspešnej inštalácii tohto vývojového prostredia je potrebné otvoriť súbor z priloženého média. Jeho umiestnenie je v zložke aplikacie/Arduino/ a volá sa EC-Sensors-Arduino.ino.

Po pripojení dosky Arduino Mega 2560 k počítaču je potrebné v časti Nástroje>Doska vybrať dosku Mega 2560 a v časti Nástroje>Procesor vybrať ATmega2560. Tlačidlom "Nahrat" v ľavom hornom rohu nahráme tento program na dosku Arduino. Následne môžeme pripojiť všetky súčasti podľa obrázkov v časti 6.1.1.

6.3 Dokumentácia k Android aplikácii

V tejto podkapitole sú opísané vybrané časti zdrojového kódu Android aplikácie.



Obrázok 21 Hierarchia tried zaradených do balíkov.

Trieda *SampleGattAttributes* definuje GATT atribúty, ktoré umožňujú identifikáciu Bluetooth modul HM-10.

```
public class SampleGattAttributes {

    public static String HM_10 = "0000ffe1-0000-1000-8000-00805f9b34fb";
    public static String CLIENT_CHARACTERISTIC_CONFIG = "00002902-0000-1000-8000-00805f9b34fb";

}
```

Metóda *connect()*, ktorá je implementovaná v triede *BluetoothLEService* zabezpečuje pripojenie k zariadeniu na základe jeho adresy.

```
public boolean connect(final String address) {
    if (mBluetoothAdapter == null || address == null) {
        Log.w(TAG, "BluetoothAdapter not initialized or unspecified address.");
        return false;
    }
}
```

Na začiatku sa skontroluje či je inicializovaný adaptér a platnosť adresy. Následne sa pokúsi obnoviť už existujúce spojenie. Ak je adresa zariadenia odlišná, pokúša sa vytvoriť nové spojenie

```

    if (mBluetoothDeviceAddress != null && address.equals(mBluetoothDeviceAddress)
        && mBluetoothGatt != null) {
        Log.d(TAG, "Trying to use an existing mBluetoothGatt for connection.");
        if (mBluetoothGatt.connect()) {
            mConnectionState = STATE_CONNECTING;
            return true;
        } else {
            return false;
        }
    }

    final BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
    if (device == null) {
        Toast.makeText(BluetoothLEService.this, "Device not found. Unable to connect.", Toast.LENGTH_SHORT).show();
        Log.v(TAG, "Device not found. Unable to connect.");
        return false;
    }
    mBluetoothGatt = device.connectGatt(this, false, mGattCallback);
    Log.d(TAG, "Trying to create a new connection.");
    mBluetoothDeviceAddress = address;
    mConnectionState = STATE_CONNECTING;
    return true;
}

```

V rovnakej triede sa nachádza aj metóda *setCharacteristicNotification()*, ktorá zabezpečuje nastavenie GATT charakteristiky, ktorej údaje budú čítané počas spojenia.

```

public void setCharacteristicNotification(BluetoothGattCharacteristic characteristic,
                                          boolean enabled) {
    if (mBluetoothAdapter == null || mBluetoothGatt == null) {
        Log.v(TAG, "BluetoothAdapter not initialized");
        return;
    }
    mBluetoothGatt.setCharacteristicNotification(characteristic, enabled);

    if (UUID.SENSORS.equals(characteristic.getUuid())) {
        BluetoothGattDescriptor descriptor = characteristic.getDescriptor(
            UUID.fromString(SampleGattAttributes.CLIENT_CHARACTERISTIC_CONFIG));
        descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
        mBluetoothGatt.writeDescriptor(descriptor);
    }
}

```

Predchádzajúca metóda je volaná z *MainActivity()* po vytvorení spojenia. Zabezpečuje inicializáciu prijímania údajov odosielaných pripojeným zariadením. Toto volanie realizuje zdrojový kód nižšie ako súčasť metódy *startGattService()*.

```

private void startGattService(List<BluetoothGattService> gattServices) {
    if (gattServices == null) return;
    String uuid = null;

    //prechod všetkých Gatt services
    for (BluetoothGattService gattService : gattServices) {
        uuid = gattService.getUuid().toString();
        if(uuid.equals("0000ffe0-0000-1000-8000-00805f9b34fb")) {

            List<BluetoothGattCharacteristic> gattCharacteristics =
                gattService.getCharacteristics();

            //prechod všetkých charakteristik
            for (BluetoothGattCharacteristic gattCharacteristic : gattCharacteristics) {
                uuid = gattCharacteristic.getUuid().toString();
                if(uuid.equals("0000ffe1-0000-1000-8000-00805f9b34fb")) {

                    mBluetoothLeService.setCharacteristicNotification(
                        gattCharacteristic, true);
                }
            }
        }
    }
}

```

Volanie tejto metódy zabezpečuje trieda `BroadcastReceiver` s označením `mGattUpdateReceiver`. Ak už bol príjem údajov zahájený, tato trieda sa stará o spracovanie údajov volaním metódy `ProcessData()`.

```

private final BroadcastReceiver mGattUpdateReceiver = (context, intent) -> {
    final String action = intent.getAction();
    if (BluetoothLeService.ACTION_GATT_CONNECTED.equals(action)) {
        mConnected = true;
    } else if (BluetoothLeService.ACTION_GATT_DISCONNECTED.equals(action)) {
        mConnected = false;
    } else if (BluetoothLeService.ACTION_GATT_SERVICES_DISCOVERED.equals(action)) {
        startGattService(mBluetoothLeService.getSupportedGattServices());
    } else if (BluetoothLeService.ACTION_DATA_AVAILABLE.equals(action)) {
        processData(intent.getStringExtra(BluetoothLeService.EXTRA_DATA));
    }
};

```

Prijaté údaje sa spracúvajú v metóde `ProcessData()`. Na začiatku je potrebná kontrola, či správa prišla v správnom formáte. Najprv sa skontroluje jej dĺžka a následne sa kontrolujú zátvorky na indexoch nula a dva. Medzi nimi sa totiž nachádza identifikátor senzoru. Ak kontrola prebehne úspešne. Zistí sa ešte dĺžka údajov zo senzora, ktorá sa nachádza na treťom indexe správy. Po úspešnom spracovaní správy sa pristupuje ku kontrole aplikačného módu. V ukážke nižšie je znázornený kód, ktorý zabezpečuje výpis údajov aj

v aplikácii a zároveň ukladanie do súboru. Podľa identifikačného čísla určí, ako budú údaje zapísané do súboru a zapíše ho aj s identifikátorom. Identifikátory senzorov sú definované na začiatku triedy *MainActivity*.

Inicializácia identifikačných premenných:

```
private int TemperatureID = 1;
private int HumidityID = 2;
private int PressureID = 3;
private int AccelerationIDX = 4;
private int AccelerationIDY = 5;
private int AccelerationIDZ = 6;
```

Ukážka spracovania prijatej správy:

```
if (data != null && (data.length() == 20)) {

    final String bracket1 = data.substring(0, 1);
    final String bracket2 = data.substring(2, 3);

    if (bracket1.equals("[") && bracket2.equals("]") ) {
        final String Type = data.substring(1, 2);
        if (!APP_MODE) {

            final String DataLenghtString = data.substring(3, 4);
            int DataLenght = Integer.parseInt(DataLenghtString);

            switch (Type) {
                case "1":

                    t1.setText(data.substring(20 - DataLenght));
                    FM.writeToCSVFile(myOutWriter, new String[]{" " + TemperatureID, data.substring(20 - DataLenght)});
                    break;
                case "2":

                    t2.setText(data.substring(20 - DataLenght));
                    FM.writeToCSVFile(myOutWriter, new String[]{" " + HumidityID, data.substring(20 - DataLenght)});
                    break;
                case "3":

                    t3.setText(data.substring(20 - DataLenght));
                    FM.writeToCSVFile(myOutWriter, new String[]{" " + PressureID, data.substring(20 - DataLenght)});
                    break;
                case "4":

                    t4.setText(data.substring(20 - DataLenght));
                    FM.writeToCSVFile(myOutWriter, new String[]{" " + AccelerationIDX, data.substring(20 - DataLenght)});
                    break;
                case "5":

                    t5.setText(data.substring(20 - DataLenght));
                    FM.writeToCSVFile(myOutWriter, new String[]{" " + AccelerationIDY, data.substring(20 - DataLenght)});
                    break;
                case "6":

                    t6.setText(data.substring(20 - DataLenght));
                    FM.writeToCSVFile(myOutWriter, new String[]{" " + AccelerationIDZ, data.substring(20 - DataLenght)});
                    break;
            }

        }

    } else { // ...
    }
```

Pri zobrazovaní údajov zo súboru som implementoval asynchrónnu triedu, ktorá obsahuje dve metódy. Jedna zabezpečuje prečítanie údajov zo súboru a nazýva sa *doInBackground()* a druhá prečítané údaje zobrazí v používateľskom rozhraní, jej názov je *onPostExecute()*. Počas zobrazovania sa využíva aj *enumerátor*, ktorý prevádza číselný údaj identifikujúci

typ čítaného záznamu na text. Tento textový popis jasne prirad'uje hodnotu k senzoru a používateľ nemusí vedieť, aké identifikačné čísla sa pre jednotlivé senzory používajú.

```
@Override
protected Integer doInBackground(String... params) {

    try {
        data = FM.readCSVFile(params[0]);
    } catch (IOException e) {
        e.printStackTrace();
    }

    return null;
}

@Override
public void onPostExecute(Integer result) {
    TextView t = (TextView) findViewById(R.id.data_text_show_session_data);
    t.setText("");

    for (String[] s : data) {

        if (checkedItems[(Integer.parseInt(s[0])) - 1]) {
            t.append(SensorType.values()[(Integer.parseInt(s[0])) - 1].toString() + " : ");
            t.append(s[1] + "\n");
        }

    }

    this.dialog.dismiss();
}
```

Pre akcie čítania a zápisu údajov zo senzorov sú implementované metódy triedy *FileManager*. Čítanie riadkov zo súboru je realizované ukladaním údajov do listu, ktorého prvkami sú dvojprvkové polia typu *string*.

```
public void writeToCSVFile(CSVWriter csvWriter, String[] data) {

    csvWriter.writeNext(data);

}

public List readCSVFile(String filePath) throws IOException {
    List<String[]> dataList = new ArrayList<>();
    CSVReader reader = new CSVReader(new FileReader(filePath),',');
    String [] nextLine;
    while ((nextLine = reader.readNext()) != null) {
        // nextLine[] is an array of values from the line
        dataList.add(new String[]{nextLine[0],nextLine[1]});
    }

    return dataList;
}
```


6.4 Používateľská príručka k Android aplikácii

Táto podkapitola obsahuje používateľskú príručku pre Android aplikáciu. Opisuje základné systémové požiadavky na zariadenie s operačným systémom Android. Zároveň približuje správanie aplikácie na základe zobrazených obrázkov používateľského rozhrania.

6.4.1 Systémové požiadavky

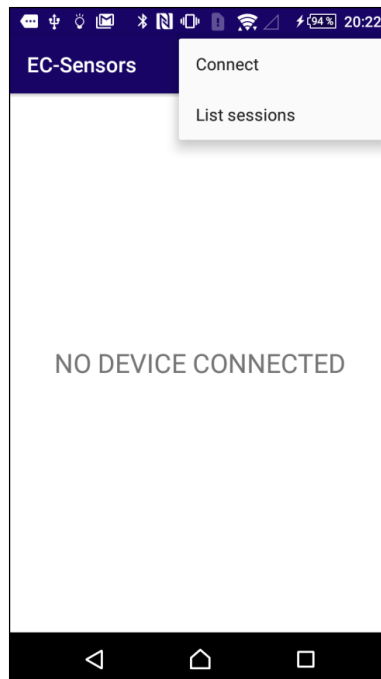
Aplikácia pre platformu Android vyžaduje minimálnu verziu Android 5.0 (API 21). Kompatibilná je teda s touto verziou a všetkými vyššími verziami operačného systému Android. Okrem toho musí Android zariadenie podporovať verziu Bluetooth 4.0, ktorá umožňuje prenos údajov s nízkou spotrebou energie. Pre realizáciu ukladania údajov je potrebné aby zariadenie disponovalo externou pamäťou. Môže sa jednať o priamo pripojenú pamäťovú kartu, ale aj simulovanú externú pamäť. Veľkosť vyžadovanej pamäte je priamo úmerná obsahu meraní, ktoré chce používateľ vykonávať.

6.4.2 Inštalačná príručka

Inštalácia aplikácie na Android zariadenie ja možná skopírovaním súboru EC-Sensors.apk z priloženého elektronického média do pamäte zariadenia. Tento súbor sa nachádza v zložke aplikacie/Android/app/. Pred spustením inštalácie je však potrebná zmena v nastaveniach samotného zariadenia. V časti zabezpečenie musí byť povolená inštalácia aplikácií z neznámych zdrojov. Po vykonaní tejto zmeny stačí kliknúť na súbor EC-Sensors.apk priamo v telefóne a povoliť inštaláciu aplikácie.

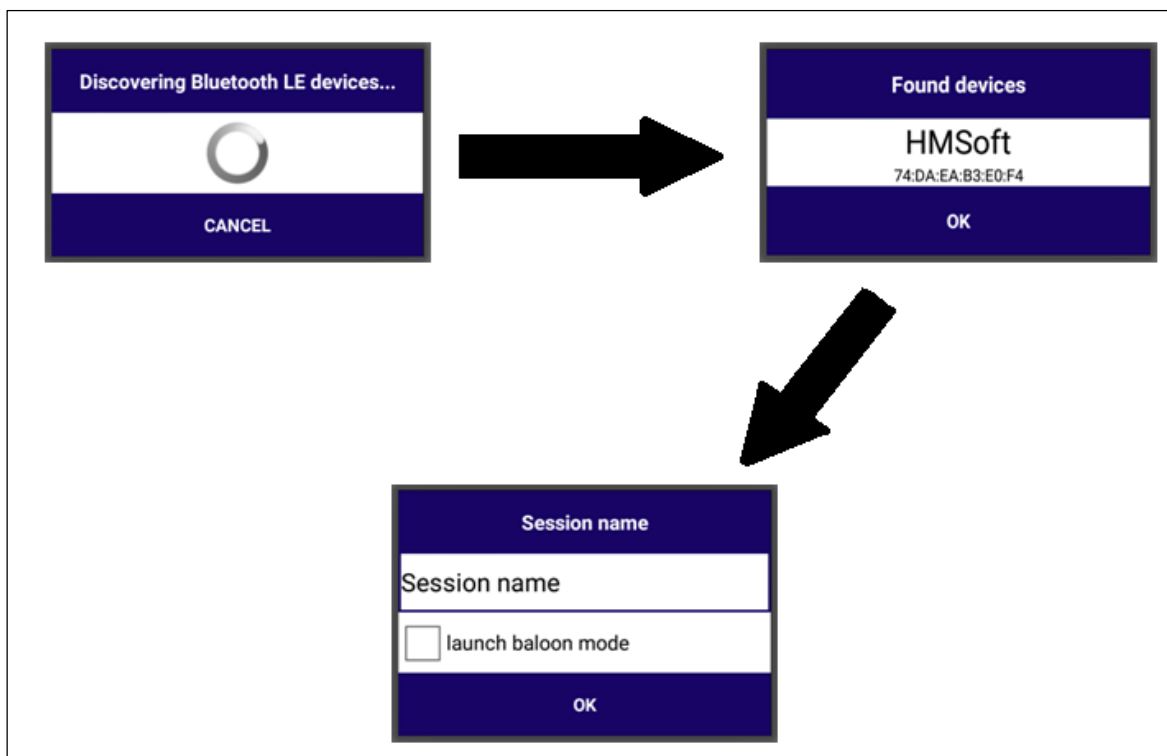
6.4.3 Opis používateľského rozhrania

Po spustení aplikácie sa používateľovi zobrazí základná obrazovka, ktorá dáva používateľovi vedieť, že nie je pripojené žiadne zariadenie a teda neprebíha žiadna komunikácia. Kliknutím na menu tlačidlo v pravom hornom obrazovky používateľ môže prejsť k vytvoreniu nového spojenia, alebo k prehľadávaniu všetkých doteraz vykonaných meraní.



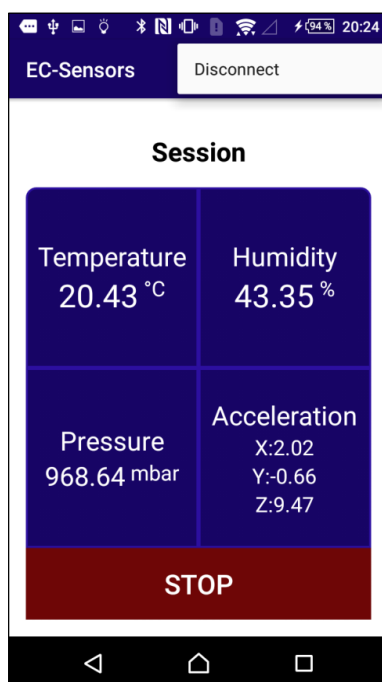
Obrázok 22 Základná obrazovka aplikácie.

Inicializáciu komunikácie používateľ začne tlačidlom "connect" v menu. Dialógové okno okamžite zobrazuje vyhľadávanie Bluetooth LE zariadení. Toto vyhľadávanie je realizované po dobu 5 sekúnd a následne sú všetky nájdené zariadenia zobrazené v liste. Používateľ si jednoducho vyberie zariadenie ku ktorému sa chce pripojiť a svoj výber potvrdí. Nasleduje zadanie mena tohto merania, ktoré sa použije ako meno súboru pre ukladanie údajov. Meno je vždy doplnené dátumom vytvorenia súboru. Používateľ má zároveň možnosť vybrať si režim v akom má aplikácia pracovať. Umožňuje mu to tzv. "checkbox" tlačidlo. V prípade, že toto tlačidlo ostane nezakliknuté, aplikácia bude prijímané údaje zobrazovať aj priamo v aplikácii. Ak používateľ zaklikne tlačidlo, prijímané údaje budú ukladané iba do súboru. Aplikácia bude informovať používateľa o prebiehajúcej komunikácii a móde v ktorom prebieha komunikácia nápisom "LAUNCH BALOON MODE".



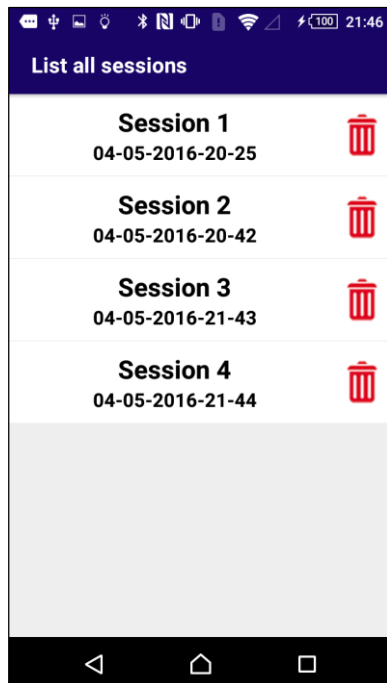
Obrázok 23 Inicializácia komunikácie s HM-10.

V prípade úspešného vytvorenia spojenia a prenosu údajov môže používateľ kedykoľvek vykonať odpojenie od zariadenia a teda zastavenie komunikácie. V menu sa počas pripojenia nachádza tlačidlo "disconnect", ktoré umožňuje okamžité zastavenie komunikácie a zrušenie párovania zariadení. Ak je aplikácia v štandardnom móde a vypisuje údaje priamo v aplikácii, môže používateľ rovnako vykonať zrušenie párovania pomocou tlačidla "stop" v spodnej časti obrazovky.



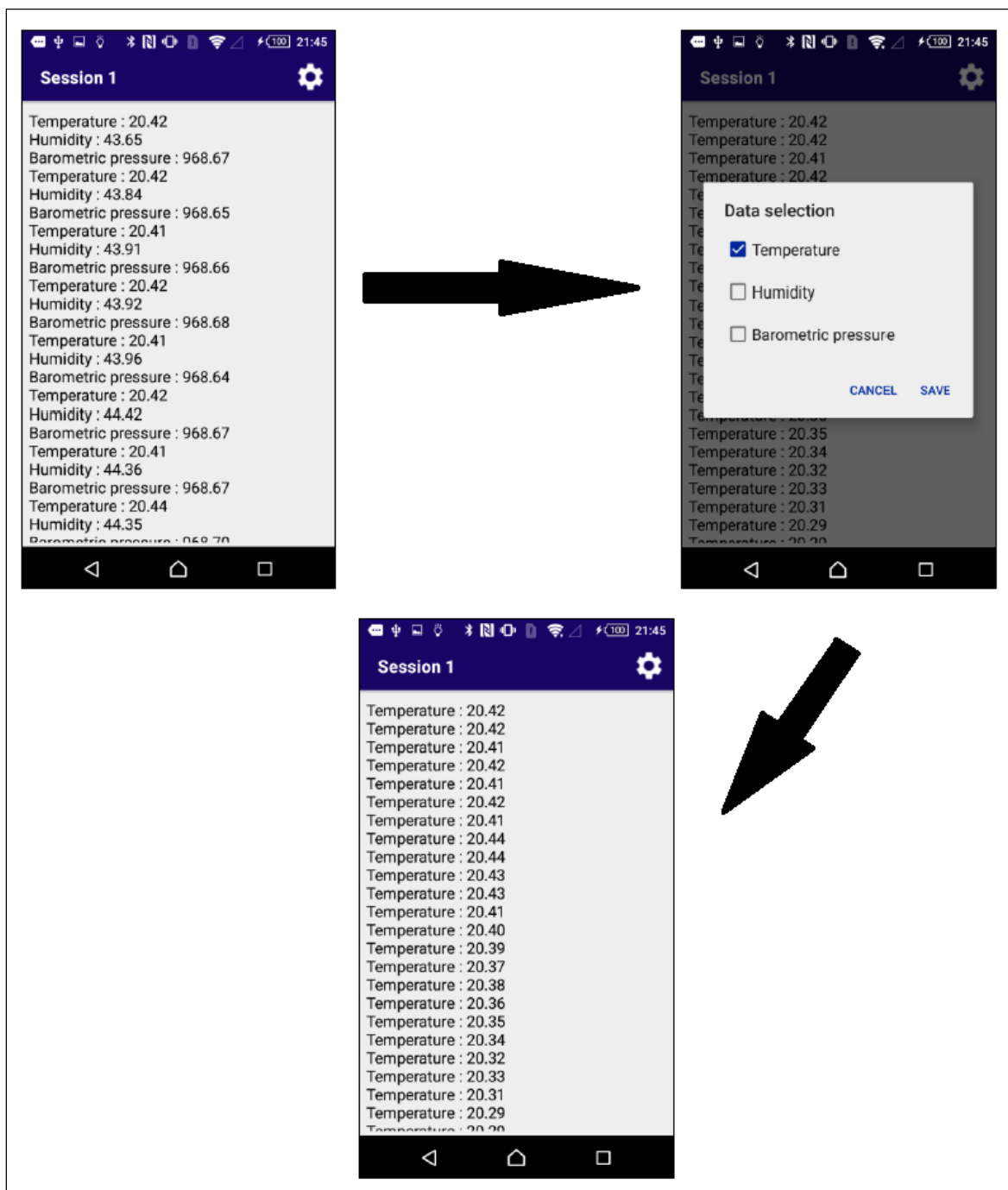
Obrázok 24 Obrazovka aplikácia počas prijímania údajov.

Spravovanie doteraz realizovaných meraní je prístupné z hlavnej aktivity v čase, keď nie je pripojené žiadne zariadenie. Kliknutím na tlačidlo "List sessions" používateľ vyvolá zobrazenie zoznamu všetkých meraní. Každý záznam má pri sebe tlačidlo v tvare koša, ktorým je možné realizovať vymazanie tohto záznamu z pamäte. Kliknutím na celý záznam sa zobrazia zaznamenané údaje z daného merania.



Obrázok 25 Zobrazenie všetkých uložených meraní.

Zobrazenie údajov zaznamenaných vybraným meraním je obmedzené iba na údaje zo senzorov teploty, vlhkosti a tlaku. Používateľ má zároveň možnosť meniť obsah výpisu údajov pomocou tlačidla v pravom hornom rohu a dialógového okna, ktoré je zobrazené po jeho stlačení. V dialógovom okne sú zobrazené možnosti výberu, ktoré používateľ určí jednoduchým zakliknutím daného políčka.



Obrázok 26 Zobrazenie nameraných údajov.

7 Zdroje

1. **Android.** [www.developer.android.com](http://developer.android.com/guide/topics/sensors/sensors_overview.html). [Online] [Dátum: 28. November 2015.] http://developer.android.com/guide/topics/sensors/sensors_overview.html.
2. —. [www.developer.android.com](http://developer.android.com/guide/topics/sensors/sensors_motion.html). [Online] [Dátum: 28. November 2015.] http://developer.android.com/guide/topics/sensors/sensors_motion.html.
3. —. [www.developer.android.com](http://developer.android.com/guide/topics/sensors/sensors_environment.html). [Online] [Dátum: 28. November 2015.] http://developer.android.com/guide/topics/sensors/sensors_environment.html.
4. —. [www.developer.android.com](http://developer.android.com/guide/topics/sensors/sensors_position.html). [Online] [Dátum: 28. November 2015.] http://developer.android.com/guide/topics/sensors/sensors_position.html.
5. **Sigma design.** Willy Event Consultants CO. [Online] 10. Máj 2012. [Dátum: 28. November 2015.] http://www.willypc.com.tw/web/homegridforum-computex2012seminar/Download/07_B_ZWA_Raoul%20W.pdf.
6. **Galeev, Mikhail.** Catching the Z-Wave. [Online] Október 2006. [Dátum: 15. Október 2015.] http://www.eetindia.co.in/ARTICLES/2006OCT/PDF/EEIOL_2006OCT30_RFD_EMS_TA.pdf.
7. **Poole, Ian.** [www.radio-electronics.com](http://www.radio-electronics.com/info/wireless/zigbee/zigbee.php). [Online] [Dátum: 18. Október 2015.] <http://www.radio-electronics.com/info/wireless/zigbee/zigbee.php>.
8. **Kinney, Patrick.** *Communications Design Conference*. December 2003. ZigBee Technology: Wireless Control that Simply. s. 1-15.
9. **Bisdikian, Chatschik.** *An Overview of the Bluetooth Wireless technology*. 12, s.l. : IEEE Communications Magazine, December 2001, Zv. 39, s. 86-94.
10. **Gomez, Carles, Joaquim, Oller a Paradelss, Josep.** Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. [Online] 29. August 2012. [Dátum: 22. November 2015.] <http://www.mdpi.com/1424-8220/12/9/11734/htm>.
11. **Martin, Jim.** Netatmo personal weather station. [Online] 24. Apríl 2014. [Dátum: 3. December 2015.] [http://www.pcadvisor.co.uk/review/ipad-accessories/netatmo-personal-weather-station-review-3442508/..](http://www.pcadvisor.co.uk/review/ipad-accessories/netatmo-personal-weather-station-review-3442508/)
12. **Netatmo.** Netatmo. [Online] 2013. [Dátum: 31. Október 2015.] http://www.netatmo.com/press/zh-CN/release/PR_NETATMO_UNVEIL_EN.pdf.
13. **TechBitar.** [www.techbitar.com](http://www.techbitar.com/sensoduino.html). [Online] [Dátum: 5. November 2015.] <http://www.techbitar.com/sensoduino.html>.

14. **Molpir.** Molpir. [Online] 24. August 2014. [Dátum: 25. November 2015.] http://shop.molpir.sk/Prilohy/254140002551_0.pdf.
15. **Micozzi, Federica.** techmate.eisworld.eu. [Online] [Dátum: 17. Október 2015.] <http://techmate.eisworld.eu/en/arduino-an-italian-success-story/>.
16. **Arduino.** www.arduino.cc. [Online] [Dátum: 20. Október 2015.] <https://www.arduino.cc/en/Main/ArduinoBoardUno>.
17. —. www.arduino.cc. [Online] [Dátum: 20. Október 2015.] <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>.
18. **Raspberry Pi.** Raspberry Pi. [Online] [Dátum: 1. November 2015.] <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>.
19. **Krajčovič, Tibor.** *Počítače*. Bratislava : Vydavateľstvo Slovenská technická univerzita, 2000. s. 47-50. ISBN 80-227-1966-6.
20. **Myers, Paul.** Interfacing using Serial Protocols. [Online] [Dátum: 18. November 2015.] <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.2531&rep=rep1&type=pdf>.
21. **Dallas Semiconductor.** Overview of 1-Wire Technology and Its Use. [Online] 3. December 2002. [Dátum: 17. November 2015.] <http://users.ece.gatech.edu/~hamblen/489X/1-wire.pdf>.
22. **Texas Instrument.** KesyStone Architecture, Universal Asynchronous Reciever/Transmitter. [Online] November 2010. [Dátum: 28. Oktober 2015.] <http://www.ti.com.cn/cn/lit/ug/sprugp1/sprugp1.pdf>.
23. I2C - learn.sparkfun.com. [Online] [Dátum: 20. 11 2015.] <https://learn.sparkfun.com/tutorials/i2c>.
24. I2C_bus_2000 - fm. [Online] 2.1, Január 2000. [Dátum: 19. November 2015.] <http://www.cs.unc.edu/Research/stc/FAQs/Interfaces/I2C-BusSpec-V2.1.pdf>.

Príloha A: Harmonogram práce pre BP1 - zimný semester

- 2. týždeň** - Úvodné stretnutie s vedúcim BP, konzultácia základných súčastí zadania a smerovania bakalárskej práce.
- 3. týždeň** - Analýza možných riešení pre realizáciu zadania a vytvorenie návrhov možných riešení.
- 4. týždeň** - Konzultácia navrhovaných myšlienok a štúdium technickej dokumentácie potencionálnych súčastí riešenia
- 5. týždeň** - Analýza možností bezdrôtovej komunikácie Android zariadení a senzorov, práca na dokumente.
- 6. - 7. týždeň** - Práce na dokumente (analýza existujúcich riešení senzorových systémov), konzultácia návrhu.
- 8. - 9- týždeň** - Konzultácia k špecifikácii požiadaviek, práce na dokumente (Arduino, Raspberry Pi, Funtoro).
- 10. týždeň** - Práca na dokumente(Možnosti pripojenia senzorov k doskám Arduino)
- 11. týždeň** - Implementácia prototypu, dokončenie dokumentu.
- 12. týždeň** - Odovzdanie dokumentácie a predvedenie základnej funkčnosti prototypu.

Zhodnotenie plánu

Harmonogram popisuje plán, ktorý bol zostavený pre prácu na prvej časti bakalárskeho projektu. Týždne, do ktorých je plán rozdelený znamenajú týždne semestra. Vypracovaný plán som sa snažil plniť so všetkých síl, no nie vždy sa mi to darilo podľa mojich predstáv. Celkový výsledok mojej práce na prvej verzii bakalárskeho projektu hodnotím pozitívne, a to aj napriek tomu, že plán bol splnený s malým omeškaním.

Príloha B: Harmonogram práce pre BP2 - letný semester

- 1. týždeň** - Prezentácia vypracovaného prototypu a základnej komunikácie medzi zariadeniami Android a Arduino, zhodnotenie riešenia a konzultácia návrhu.
- 2. - 3. týždeň** - Konzultácia detailov návrhu a práca na programe pre Arduino platformu.
- 4. týždeň** - Prezentácia funkčného merania údajov prostredníctvom senzorov. Konzultácia detailov komunikácie medzi zariadeniami Android a Arduino.
- 5. týždeň** - Práca na dokumente a konzultácia doterajšej práce.
- 6. - 7. týždeň** - Implementácia prehľadného zobrazovania údajov v Android aplikácii.
- 8. týždeň** - Konzultácia detailov implementácie Android aplikácie.
- 9. týždeň** - Implementovanie spravovania súborov v Android aplikácii.
- 10. týždeň** - Práca na dokumente a finalizácia oboch aplikácií.
- 11. týždeň** - Dokončenie dokumentu.
- 12. týždeň** - Odovzdanie bakalárskeho projektu.

Zhodnotenie plánu

Harmonogram popisuje plán, ktorý bol zostavený pre prácu na bakalárskom projekte počas letného semestra. Týždne, do ktorých je plán rozdelený znamenajú týždne semestra. Vypracovaný plán som sa snažil plniť, no v prípade písania dokumentu vznikali značné omeškania. Dôvodom boli náročné zadania počas semestra a omeškanie hardvéru potrebného pre bakalársky projekt. Naopak implementácia Android aplikácie bola dokončená s predstihom a ostatné týždne som len vylad'oval detaily. Našťastie sa všetko podarilo dobehnúť poctivou prácou v záverečných týždňoch semestra.

Príloha C: Obsah elektronického média

./obsah.txt	- Popis obsahu elektronického média
./anotacia.pdf	- Anotácia v slovenskom jazyku
./annotation.pdf	- Anotácia v anglickom jazyku
./dokument_BP.pdf	- Dokumentácia práce
./aplikacie/Android/source/*	- Zdrojové súbory Android aplikácie
./aplikacie/Android/app/EC-Sensors.apk	- Spustiteľný súbor Android aplikácie
./aplikacie/Arduino/EC-Sensors-Arduino.ino	- Zdrojový súbor Arduino aplikácie

Príloha D: Elektronické médium