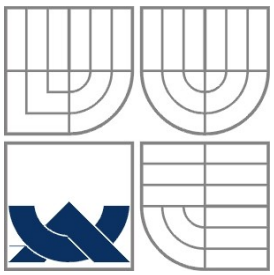
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

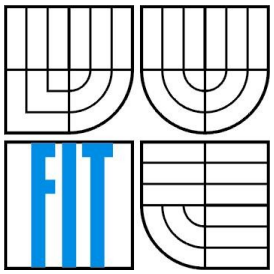
Fakulta informačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2016 Adam Kožušník



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

CHYTRÝ OBOJEK
SMART COLLAR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ADAM KOŽUŠNÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN FUČÍK
SUPERVISOR

Brno, 2016

Abstrakt

Práce se zabývá problematikou inteligentního obojku pro psa. Cílem je návrh a vytvoření aplikace pro monitorování pohybu psa s vhodným grafickým uživatelským rozhraním. Pro snímání pohybu je využito zařízení SensorTag od firmy Texas Instruments. Program je vytvářen jako konzolová aplikace se zobrazením hodnot a pohybu na internetových stránkách.

Abstract

This thesis deals with problem of smart collar for a dog. The goal is to design and implement an application with a suitable user interface, which will monitor movements of a dog. I used Texas Instrument's device named SensorTag from as a motion detector. The program is created as a console application with values and movements displayed on an internet website.

Klíčová slova

Chytrý obojek, C#, MS Visual Studio, AngularJS, HTML, CSS, Databáze MSSQL, Elektronický kompas

Keywords

Smart collar, C#, MS Visual Studio, AngularJS, HTML, CSS, Database MSSQL, Electronic compass

Citace

KOŽUŠNÍK Adam. Chytrý obojek. Brno, 2016. 31 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Fučík Jan

Chytrý obojek

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Fučíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Adam Kožušník
18.5.2016

Poděkování

Zde bych rád poděkoval panu Ing. Janu Fučíkovi za zapůjčení veškerých zařízení, odborné konzultace a podněty při vypracování této bakalářské práce. Dále také firmě CAMEA za zapůjčení veškerého potřebného vybavení.

Obsah

1 Úvod

Tato práce se zabývá problematikou chytrého obojku pro psa. Cílem je navrhnout a implementovat konzolovou aplikaci s vhodným zobrazováním hodnot a pohybu na webových stránkách. Na obojku bude připevněno zařízení SensorTag od firmy Texas Instruments, které disponuje několika různými senzory. Data zasílána z těchto senzorů budou v aplikaci zpracována a z nich se následně zjistí pohyb psa. Program není omezen pro použití pouze na platformě Windows, lze jej používat také na Unixu. Podmínkou je nainstalovaná podpora .NET.

V dnešní době má mnoho lidí a rodin psy. Není vždy jednoduché tyto mazlíčky uhlídat, zvláště na zahradě, kde se často stává, že psi utečou. Použití chytrého obojku sice tomuto nezabrání, může však uživatele ihned upozornit, pokud k útěku dojde. Navrhovaná aplikace bude mít právě tento úkol. Z pohybu zvířete bude vyhodnocovat, zda se stále pohybuje v území zahrady či nikoli. Z dat je pak také možno uživatele informovat o statistikách pohybu zvířete.

Celá práce je rozdělena do pěti kapitol včetně úvodu. Po něm následuje kapitola Současný stav (2), ve kterém jsou popsány dnes dostupné chytré obojky a jejich možnosti používání včetně zhodnocení jejich vlastností, použitých technologií, výhod a nevýhod. Následuje popsání použitého zařízení v kapitole (3) a jeho možných rozšíření. Jelikož je toto zařízení pro práci stěžejní, je popsáno detailněji. Čtvrtá kapitola (4) se věnuje již samotné implementaci, je zde však zahrnut také popis použitých prostředků a návrh celé aplikace, který bylo z technických důvodů nutné několikrát předělat. V podkapitolách jsou uvedeny popisy jednotlivých částí programu jako např. databáze či nutné výpočty. V poslední, páté části (5), je zhodnocení celé práce včetně možných rozšíření celého projektu.

2 Současný stav

V této kapitole jsou popsány již existující chytré obojky, zejména pak způsoby jak fungují. Pokusím se vyzdvihnout jejich výhody a nevýhody. Chytrý obojek se dnes dá koupit prakticky v každém obchodu s potřebami pro zvířata. Metoda, jakým kontrolují pohyb zvířete, se v zásadě neliší. V čem se však obojky liší, je způsob komunikace s přijímacím zařízením.

2.1 Účel chytrých obojků

Dávno pryč jsou doby, kdy obojek sloužil pouze k uchycení vodítka. Možnosti dnešní techniky jsou již mnohem dále. Můžeme se tedy setkat například s obojky, které odnaučí psa štěkat [13]. Štěkot psa je zaznamenán buď mikrofonem nebo snímačem chvění kůže na krku psa. Pokud je štěkání zaznamenáno, psovi se dostane nepříjemného vysokého tónu. Jiné obojky zase poskytují slabé elektrické šoky, vibrace nebo vystříknutí speciální náplně ze spreje. Podle těchto trestů se obojky označují jako ultrazvukové, elektrické, vibrační či sprejové. Ceny se pohybují od 350 Kč až po několik tisíc korun.

Obojky, které lidem pomáhají cvičit jejich domácí mazlíčky, také nejsou ničím novým. Tyto obojky využívají taktéž elektrických impulsů nebo ultrazvukových signálů (případně kombinace obojího). K přijímači, který je umístěn na obojku, zákazník obdrží také ovladač, ze kterého vysílá požadované signály. Intenzita elektrického impulsu a ultrazvukového signálu není pro psa nebezpečná a nemůže mu ublížit. Je pouze nepříjemná a slouží k tomu, aby si pes nepříjemný pocit spojoval s neuposlechnutím povelu. Obojek také může nahradit píšťalku, neboť ji dokáže napodobit. Cena takového obojku je už značně vyšší, pohybuje se na hranici 3500 Kč.

Hitem posledních let jsou obojky pro monitorování pohybu. Těmto obojkům se budu následně podrobněji věnovat, jelikož existuje několik způsobů, jak tyto obojky komunikují.

2.2 Způsob detekce pohybu

Valná většina těchto obojků využívá GPS[1] lokátoru. Použití této technologie je velice rozšířené v celé řadě zařízení od navigací a mobilních telefonů, až po chytré obojky [15]. S tímto systémem je možné určit polohu zařízení prakticky kdekoliv na Zemi. Přesnost okolo 10 metrů je vzhledem k pokrytí celé planety velmi dobrá. Lze ji však ještě zvýšit s použitím pokročilejších optimalizačních metod až na jednotky centimetrů. Jednoduchost použití tohoto systému považuji za velkou výhodu. Pořizovací náklady těchto obojků ovšem nejsou z nejnižších, neboť začínají na částce 3500 Kč. Nejmodernější kusy však stojí až k deseti tisícům. V neposlední řadě je zde také značná spotřeba energie, která je velkou nevýhodou. Výdrž u jednotlivých zařízení se pohybuje v řádech dnů, maximálně týdnů, což je však vzhledem k možnostem obojků dostačující hodnota. Moderní obojky se snaží tento neduh zmírnit přepínáním do úsporného módu. Pokud se zařízení nepohybuje, automaticky se tak funkce GPS vypne a tím šetří baterii. Při zaznamenání pohybu se zařízení GPS opět samo zapne. Kromě polohy lze také zjistit přesný čas kdekoli na zemi. GPS přijímače přijímají signály z jednotlivých družic. Z přijatých dat přijímač vypočte polohu a zjistí přesné datum a čas.

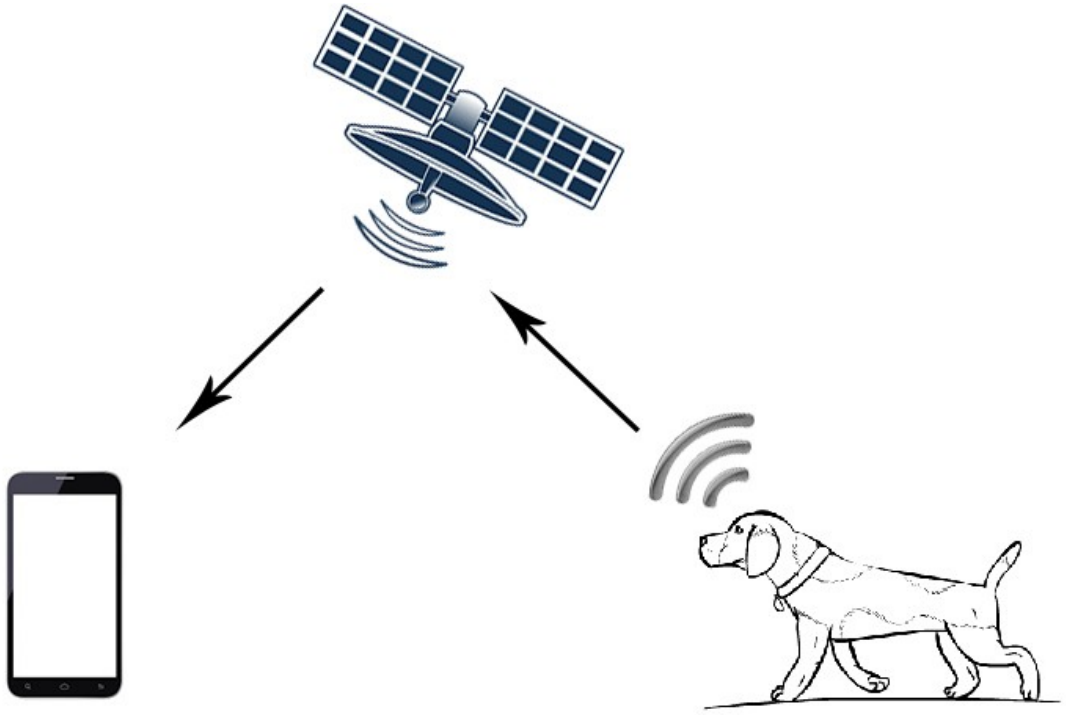
[1] https://en.wikipedia.org/wiki/Global_Positioning_System

Komunikace mezi přijímačem a družicí probíhá pouze jednostranně a to od družic k uživateli, opačně nikoli. V aplikacích, které jsou dodávány k chytrým obojkům, je možné sledovat rychlost pohybu psa či vzdálenost, kterou pes uběhl. Obojky pravidelně vysílají údaje o poloze. Poloha v aplikaci se zobrazuje vzhledem k přijímači, ukazuje tedy vzdálenost psa od pána.

2.3 Typy přenosů dat

2.3.1 GSM

Pokud je obojek propojen s mobilním telefonem, využívá buď spojení GSM [6] nebo Bluetooth [7]. Systém GSM je nejrozšířenějším standardem pro mobilní telefony. Lokalizace využívá satelitů, které jsou rozmístěny kolem celé naší planety. Dosah takovéhoto spojení je tak prakticky neomezený. GSM je buňková síť. Připojení mobilního telefonu či jiného zařízení probíhá přes nejbližší buňku. Buňky se dělí podle velikosti – makro, mikro, piko a deštníkové buňky. Rozdíl mezi těmito buňkami je v oblasti nasazení. Makro buňky mají antény umístěné nad úrovní střech. Mikro jsou pod úrovní střech, takže se hojně nasazují v zastavěných oblastech. Piko buňky se uplatňují zejména uvnitř budov. Deštníkové se používají na vyplnění děr mezi buňkami. V obojcích s tímto typem přenosů jsou často SIM karty[2], která obsahuje informace nutné k přihlášení uživatele do sítě. SIM karta je vložena do obojku a na něj se dá takto dovolat, a tím zjistit jeho poloha. Obojek je tedy možné nalézt téměř kdekoli, kde je signál. Tento druh spojení však vybíjí baterii opravdu velice rychle. Výdrž je pak v rozmezí několika hodin až několika dní. Další nevýhodou je také platba za každé spojení s tímto obojkem, jelikož se jedná prakticky o hovor.



Obrázek 2.1: GSM spojení

[2] https://en.wikipedia.org/wiki/Subscriber_identity_module

2.3.2 Rádiové spojení

Další možností, jak data přenášet, je pomocí rádiového spojení [16], které pracuje na principu vysílačky. Rádiové vlny jsou druhem elektromagnetické radiace. Vlnová délka v elektromagnetickém spektru je delší než u infračerveného světla. Frekvence rádiových vln se pohybuje v rozmezí od 3 kHz do 3 THz s délkou dosahu od mikrometrů do sta kilometrů. Rychlost šíření těchto vln je stejná jako rychlost světla. Aby bylo možné data posílat, je třeba mít rádio vysílač a přijímač. Oproti spojení GSM probíhá komunikace přímo, není potřeba žádné spojovací zařízení. Vysílaný signál může přijímat libovolná stanice. Musí však naslouchat na stejné frekvenci, na jaké jsou data vysílána. Různé vlnové délky mají rozdílnou charakteristiku a zejména možnosti použití v zemské atmosféře. Vlny dlouhé délky mohou obtékat například kolem hor a sledovat tak obrys země, kdežto kratší vlny se mohou odrážet od ionosféry a vrátit se na zem za horizont. Krátké vlny se ohýbají jen velmi málo, takže je vzdálenost, na kterou se mohou šířit, omezena vizuálním obzorem. Chytré obojky pracují na středních či nízkých frekvencích, které odpovídají frekvenci 300 – 3000 kHz, respektive 3 – 30 kHz. Dosah obojků operujících ve středních frekvencích je přibližně do jednoho kilometru. U nízkých frekvencí je dosah až 10 kilometrů. Dosah je samozřejmě závislý na terénu, ve kterém se pohybujeme. Obojky, které využívají tohoto typu komunikace, jsou už velmi propracované a určeny pro náročné uživatele. Poskytují vesměs všechny výše uvedené služby a navíc přidávají další, jako např. topografické mapy, přeposílání hlasových povelů atd. Využívají je především lidé, kteří psy nasazují v terénu, jako jsou například policisté, vojáci či hasiči. Většina těchto přístrojů umožňuje monitorovat více psů najednou, ta nejlepší zařízení až dvě desítky. Ceny těchto obojků samozřejmě odpovídají poskytovaným službám a u základních modelů začínají na částce 8 tisíc korun. Za nejlepší modely je však nutno zaplatit přes 20 tisíc korun [14].

2.3.3 Bluetooth

Mnoho obojků data přenáší pomocí protokolu Bluetooth, který spadá do kategorie osobních počítačových sítí a využívá k posílání dat paketů. Protokol je postaven na struktuře klient-server, kdy server může najednou komunikovat s maximálně sedmi klienty. Role se v průběhu spojení mohou vyměnit. Přenos dat může probíhat pouze s jedním klientem v daný čas. Server určuje, kdy které zařízení může komunikovat. Střídání zařízení probíhá nejčastěji pomocí algoritmu Round-robin, kdy jsou takto zařízení střídána stále ve stejném pořadí znovu a znovu. Existuje několik verzí protokolu Bluetooth, které se liší dosahem a přenosovou rychlostí. Přehled je v tabulce níže.

Verze	Přenosová rychlost	Dosah
1.0	1 Mbit/s	100 m
2.0	3 Mbit/s	10 m
3.0	24 Mbit/s	1 m
4.0	1-3 Mbit/s	<100 m
4.2	1 Mbit/s	>100 m

Tabulka 2.1: Verze Bluetooth

Do roku 2010 byla nejčastěji používána verze 2.0, která se hojně vyskytovala například v mobilních zařízeních. Verze 1.0 se používala zejména v průmyslu a nebyla určena pro běžné používání. Verze

3.0 je využívána v speciálních případech, kde se využije vysoká přenosová rychlost a není potřeba velký dosah. Data se posílají přes jeden ze 79 možných kanálů, každý pracující s šířkou pásma 1 MHz. V roce 2011 byla vyvinuta verze 4.0. Přenosová rychlost je stejná jako u verze 2.0, značně se však zvětšil dosah. Významná změna nastala u verze 4.2, často označovanou jako Bluetooth Smart. Je možné se však setkat s názvem Bluetooth Low Energy. Rozdíl oproti předchozí verzi je zejména ve spotřebě energie, která může být až 100 násobně nižší oproti předchozí verzi. Doba odezvy se také velice snížila ze 100 ms na pouhých 6 ms. Tato verze je především pro použití u maloobjemových přenosů dat, jakými chytré obojky disponují. Protokol Bluetooth je v dnešní době opravdu velice rozšířený a prakticky každé zařízení, které dnes zakoupíme (chytré telefony, hodinky, tablety, televize atd.), poskytuje toto připojení.

2.3.4 ZigBee

ZigBee [8] je poměrně mladým standardem, který byl vyvinut v roce 2004 a stejně jako Bluetooth spadá do kategorie osobních počítačových sítí. Komunikace je možná na vzdálenosti okolo 100 metrů. ZigBee nachází uplatnění v mnoha různých odvětvích od automatizace budov, spotřební elektroniku, počítačové periferie až po průmyslové automatizace. Typicky se používá u přenosů s malým objemem dat. Maximální přenosová rychlost je okolo 250 kbit/s. Přenos dat může být šifrován, a to až 128 bitovým klíčem. Navíc je možné jej použít jak u periodicky se opakujících přenosů (např. data z čidel), tak u nepravidelných přenosů či přenosů s důrazem na malé zpoždění. Umožňuje také tvorbu rozsáhlejších bezdrátových sítí, kde není zapotřebí přenosu velkého množství dat. Mezi hlavní výhody patří spolehlivost, nenáročná implementace a také velmi nízká spotřeba energie. Je navíc velice nenáročný na hardware. Právě díky těmto vlastnostem je ZigBee využíváno u aplikací s napájením z baterií. Výhod má tento protokol tedy spoustu, bohužel kvůli objemu přenesených dat není ZigBee tolik nasazováno. Pro potřeby chytrých obojků je však naprosto dostačující.

3 Použité zařízení

Zde je popsáno zařízení, které jsem k této práci využil. Níže je uveden jeho detailní popis, včetně všech senzorů, kterými disponuje. Dále také uvádím mobilní aplikaci vytvořenou k tomu zařízení. Jak je popsáno v kapitole 4.2.1, původní návrh nebylo možné zrealizovat. Využití této aplikace umožnilo sbírat data ze SensorTagu, proto ji také zmiňuji.

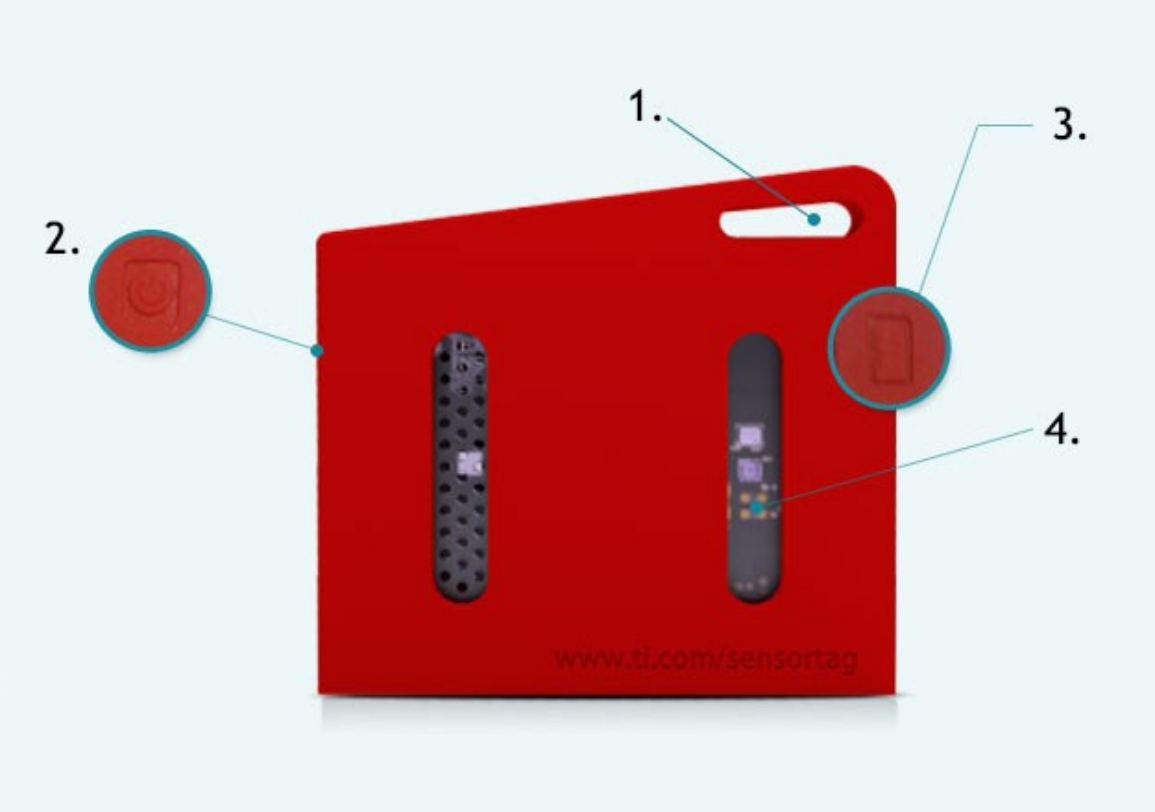
3.1 SensorTag

Obrázek 3.1: SensorTag [10]

Zařízení, které bude připevněno k obojku, je od firmy Texas Instruments a nazývá se SensorTag [1], [5]. Je napájeno baterií CR2032. Pohání jej čip CC2650 s 256 kB paměti flash a k dispozici je paměť RAM o velikosti 8 kB. Komunikace může probíhat přes Bluetooth Smart, ZigBee či 6LoWPAN. Veškeré senzory v tomto zařízení jsou velice malé, aby bylo dosaženo kompaktních rozměrů. Toto zařízení disponuje celkem dvěma tlačítky, dvěma diodami a osmi senzory. Těmi jsou: akcelerometr, gyroskop, magnetometr, teploměr, teploměr objektu nacházející se před zařízením, měřič vlhkosti, světelné čidlo a měřič tlaku.

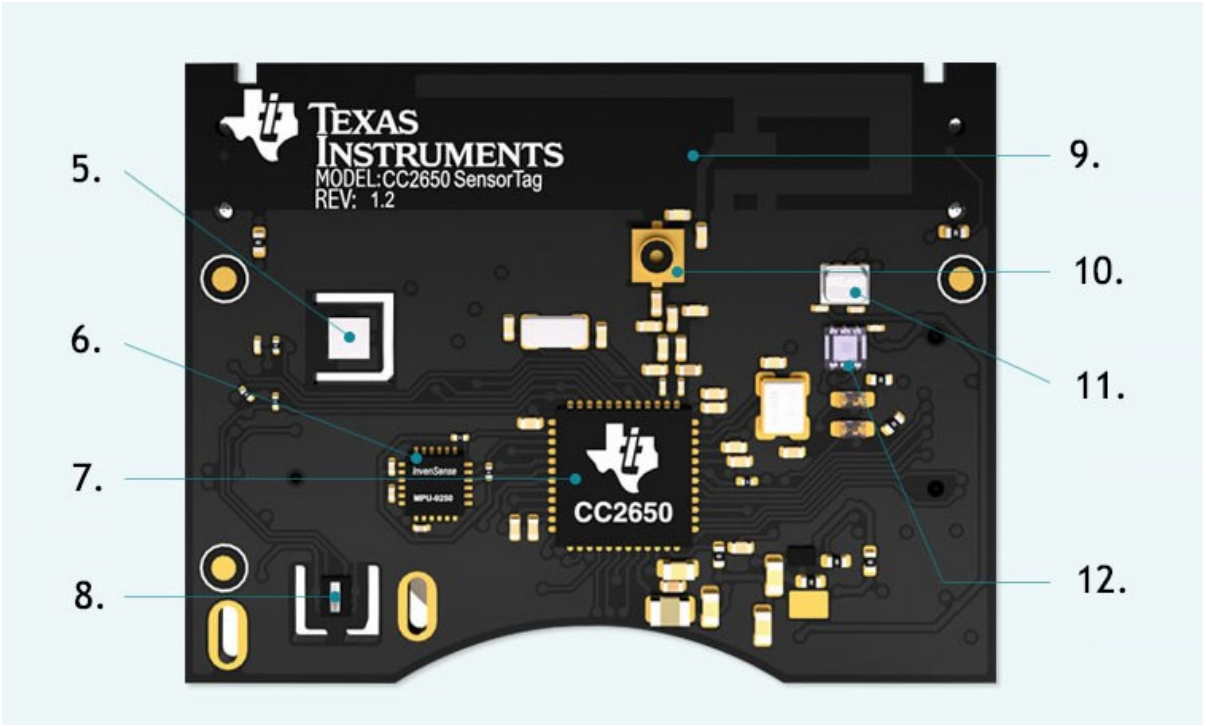
Všechny tyto senzory samozřejmě pro chytrý obojek potřebné nejsou. V zařízení se dá naprogramovat, které senzory budou aktivní a které nikoli. Tím lze také dosáhnout úspory energie. Pro potřeby této práce jsou nejdůležitější akcelerometr a magnetometr. Z nich se pokusím vypočítávat pohyb zvířete. V továrním nastavení zařízení vypočítává hodnoty každých 1000 ms. Pro snímání pohybu však není takto velká perioda dostačující. Jednalo by se totiž o data, která by neměla velkou vypovídací hodnotu kvůli velkým prodlevám při měření. Je tedy nutné snížit periodu snímání alespoň pod 500 ms. Čím nižší periodu nastavíme, tím přesnější data získáme, zároveň však dochází k odesílání většího množství dat a také větší energetické náročnosti. Výdrž zařízení při periodě odesílání dat 1000 ms je přibližně jeden rok. Nejnižší perioda, kterou lze nastavit, je 100 ms. Data

jsou takto již dosti přesná. Dosah zařízení při používání protokolu Bluetooth je okolo 50 metrů, v případě ZigBee pak dokonce 100 metrů. Zařízení je chráněno plastovým krytem, který je rozložitelný, aby se dalo přistupovat k baterii. Má drobné dírky, které jsou nutné pro měření teploty a k chlazení. Přes tento kryt je poté přetažen další gumový, který chrání celé zařízení před nárazy. Rozměry jsou 5 x 6,7 x 1,4 cm včetně gumového krytu. Pro připevnění je na něm umístěni poutko. Cena zařízení je 29 dolarů. Níže následuje detailnější popis.



Obrázek 3.2: SensorTag zezadu [10]

1. Poutko pro uchycení zařízení.
2. Tlačítko vypnout/zapnout – Pro zapnutí či vypnutí stačí pouze zmáčknout. Pro obnovení firmwaru do režimu Bluetooth Smart je potřeba jej podržet spolu s uživatelským tlačítkem po dobu 10 vteřin.
3. Uživatelské tlačítko – Pro přepínání mezi protokoly Bluetooth Smart, 6LoWPAN, ZigBee stačí podržet tlačítko po 3 vteřiny. Pro režim majáku (zařízení je uspáno a vysílá signál, že je aktivní) je třeba jej držet stisknuté 6 vteřin. Obnovení Bluetooth Smart módu proběhne po současném stisknutí s tlačítkem pro vypnutí/zapnutí po dobu 10 vteřin.
4. Zelená/červená dioda – Při načítání senzorů dochází k rychlému problikávání zelené diody. Pomalé blikání zelené diody značí, že zařízení vysílá data. Pokud bliká červená dioda, znamená to, že došlo k nějakému problému.



Obrázek 3.3: Deska SensorTagu [10]

5. Bezkontaktní teploměr TMP007[3] – Senzor je schopný měřit teplotu objektů před senzorem s přesností +/- 1 °C. Vzdálenost nutná pro měření záleží na velikosti objektu. Doporučuje se vzdálenost menší, než je poloměr objektu. Teplotu lesklých materiálů však nelze měřit.
6. Devíti-osý pohybový senzor MPU-9250[4] – Skládá se ze tří zařízení:
   i) Akcelerometr – Akcelerace je měřena ve 3 osách s výsledkem zapsaným na 14 bitech pro každou osu. Může také měřit směr gravitace.
   ii) Gyroskop – Měří rychlost otáčení ve všech 3 osách (X, Y, Z) s rozlišením na 16 bitů pro každou osu. Může měřit pouze změnu směru, nikoli směr gravitace. Při kombinaci s akcelerometrem lze využít k orientaci ve 3D prostoru.
   iii) Magnetometr – Měří magnetické pole ve 3 osách se zápisem na 16 bitů pro každou osu. Data z tohoto senzoru jsou kombinována s daty z akcelerometru pro vytvoření kompasu. Měřit lze jak magnetické pole země, tak i magnetické pole vzniklé z elektroniky.
7. Bezdrátová řídící jednotka CC2650[5] podporující více standardů přenosu dat.
8. Senzor vlhkosti HDC1000[6] – Toto čidlo snímá relativní vlhkost a teplotu okolí. Vlhkost je zaznamenána na 12 bitech, teplota na 14 bitech. Pro správné měření je nutný přímý kontakt se vzduchem, který zajišťuje otvor v krytu. Zároveň však musí být snímač vzduchotěsný vzhledem k vnitřku zařízení. Zařízení také odděluje teplo z elektronických součástek, které jsou uvnitř, aby bylo měření co nejpřesnější.

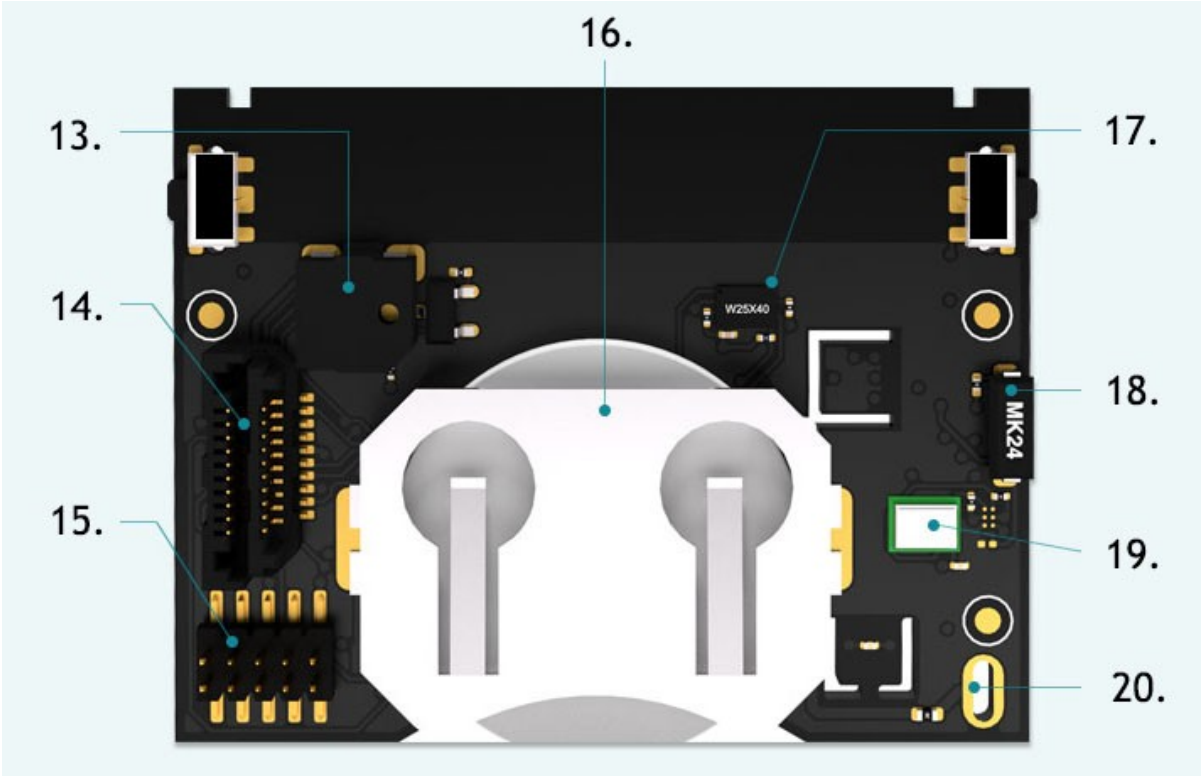
[3] http://www.ti.com/product/TMP007

[4] http://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/

[5] http://www.ti.com/product/CC2650

[6] http://www.ti.com/product/HDC1000

9. Anténa z plošných spojů.
10. uSMA RF konektor.
11. Měřič tlaku BMP280[7] – Měří tlak okolního vzduchu se zapsáním dat na 16 bitech.
12. Světelný senzor OPT3001[8].



Obrázek 3.4: Deska SensorTagu zezadu [10]

13. Bzučák.
14. DevPack[9] rozšiřující konektor.
15. JTAG rozhraní pro programování a ladění.
16. Svorka pro baterii CR2032.
17. 4M Serial Flash paměť – Lze využít pro ukládání dat v režimu offline a podporuje několik protokolů.
18. Magnetický senzor MK24[10].
19. Digitální mikrofon SPH0641LU[11].
20. Pin pro připojení rozšiřující součástky s AAA bateriemi.

[7] https://www.bosch-sensortec.com/bst/products/all_products/bmp280

[8] http://www.ti.com/product/OPT3001

[9] http://www.ti.com/tool/cc-devpack-debug

[10] https://standexelectronics.com/products/mk24-series-reed-sensor-2/

[11] http://www.knowles.com/eng/Newsroom/New-product-Ultrasonic-MEMS-Microphone

3.2 Rozšíření SensorTag

K SensorTagu lze dokoupit a připojit několik rozšíření [12]. První z nich se nazývá SimpleLink SensorTag Debugger DevPack. Toto rozšíření přidává možnost ladění do SensorTagu. Zapojí se do rozšiřující hlavičky DevPack a pomocí programů Code Composer Studio[12] či IAR ARM[13] je možné ladit. Při zakoupení uživatel automaticky obdrží licenci pro používání Code Composer Studia. Dodává se také s USB kabelem pro napájení zařízení během ladění či pro neustálé dobíjení, pokud to aplikace energeticky vyžaduje. Tato ladící souprava tak značně rozšiřuje možnosti použití SensorTagu. Cena je 15 dolarů.

Dalším rozšířením je malý displej, který se k zařízení SensorTag připojí. Displej se taktéž zapojí do konektoru DevPack. Disponuje rozlišením 96 x 96 pixelů, je barevný a poskytuje velmi dobré pozorovací úhly spolu s vysokým kontrastem pro dobrou čitelnost. Je navržen pro nositelné aplikace, meteorologické stanice nebo jakékoliv přenosné zobrazovací aplikace, které prezentují informace z webu. Displej už obsahuje dodatečnou vnější pryžovou objímku, která jej chrání proti poškození. Napájen je z baterie v SensorTagu. Díky použitým technologiím má však displej mimořádně nízký odběr proudu 2 uA. Cena displeje je 19 dolarů.

Posledním originálním rozšířením dostupným na internetových stránkách výrobce je LED Audio DevPack. S tímto rozšířením lze SensorTag využívat k tvorbě světelných aplikací. Stejně jako výše zmíněné rozšíření se i toto zapojuje do konektoru DevPack. Obsahuje čtyři vysoce výkonné LED od značky OSRAM, které disponují barvami červená, zelená, žlutá a bílá. Ovládat LED lze přes Bluetooth Smart z mobilního telefonu skrz mobilní aplikaci od Texas Instruments nebo přes ZigBee. Dodatečná pryžová objímka je taktéž součástí. Pro používání diody již nestačí pouze baterie v SensorTagu, která má pro potřeby LED malou kapacitu. Při plné intenzitě může LED čerpat až 2A proudu, proto je potřeba USB napájení. Cena LED je 19 dolarů.

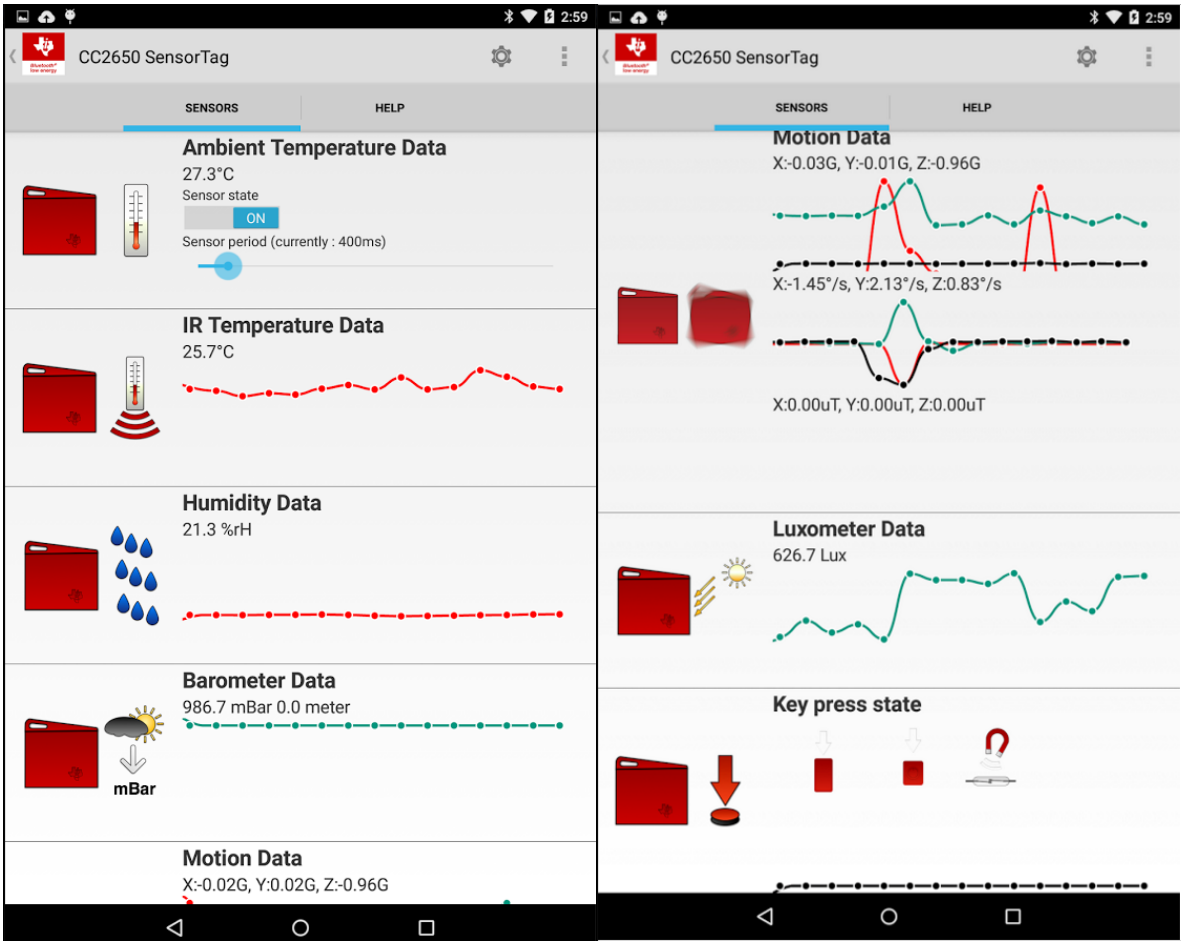
Texas Instruments umožňuje vytvořit si své vlastní moduly a rozšířit tak ještě více funkčnost SensorTagu. Může se jednat o kapacitní dotykové displeje, nové senzory, připojení dalších baterií pro větší výdrž u energeticky náročných aplikací a mnoho dalšího. Na internetových stránkách výrobce je příručka, podle které je možné si vytvořit své vlastní rozšíření. Lze využít zdroje zde dostupné a začít tak od nich.

[12] http://www.ti.com/tool/ccstudio-msp

[13] http://www.ti.com/tool/iar-kickstart

3.3 Mobilní aplikace BLE SensorTag

Tato aplikace je volně ke stažení v obchodech Google Play[14] a AppStore[15]. Je tedy dostupná pro platformy Android a Apple iOS. Po nainstalování je možné ji ihned spárovat se SensorTagem přes protokol Bluetooth. Po připojení se zobrazí nabídka všech senzorů. U každého z nich jsou zobrazovány hodnoty, které naměřily a to včetně příslušných jednotek. Hodnoty jsou doplněny o grafický výstup v podobě grafů, které ukazují, jak se hodnoty mění. Senzory lze jednotlivě ovládat, takže je lze libovolně zapínat/vypínat a měnit jejich periody snímání. V nastavení je možné zvolit také jednotky, ve kterých jsou data prezentována (např. teplota v °C/°F). Dále jak dlouho má být spojení aktivní při přerušení chodu aplikace. Aplikace umožňuje přeinstalování firmwaru. Z nabídky v nastavení si vyberete verzi firmwaru, kterou požadujete a stisknete Start Programming. Aplikace se už o vše postará. Vlastnost, která je pro tuto práci velmi důležitá, je možnost sdílení dat přes internet. Nastavení sdílení je popsáno podrobněji v kapitole 4.3.



Obrázek 3.5: Mobilní aplikace BLE SensorTag [10]

[14] https://play.google.com/store

[15] https://itunes.apple.com/cz/genre/ios/id36?mt=8

4 Implementace

V této kapitole je popsán návrh a implementace celého projektu včetně použitých technologií. Byl jsem nucen návrh několikrát předělávat kvůli technickým problémům se zařízením SensorTag a také kvůli malému množství dostupných materiálů k realizaci. Implementace je rozdělena do několika částí, neboť celý program se skládá z několika vrstev. První z nich je databáze, kde se ukládají jak surová data ze SensorTagu, tak i data vypočítaná. Další vrstvou je samotná aplikace v C#, která vše řídí. Jako poslední vrstva jsou zde internetové stránky, kde se data zobrazují společně s vykreslováním pohybu zvířete.

4.1 Použité prostředky pro programování

Pro tvorbu aplikace, která bude data snímat a vyhodnocovat, jsem se rozhodl využít jazyku C# od společnosti Microsoft. Ačkoli je tento objektově orientovaný jazyk poměrně mladý (vznikl v roce 2000 společně s platformou .NET), velmi rychle se rozvíjí a dnes je hojně používaný. S tímto jazykem jsem neměl doposud žádné zkušenosti. Jelikož je však C# založen na jazycích C++ a Java, v kterých jsem již programoval, učení nebylo nikterak náročné. Syntaxe je velice podobná jazyku C, což učení ještě ulehčilo. Hlavním důvodem, proč jsem se rozhodl použít tento programovací jazyk, je jeho časté používání na reálných projektech. Lze v něm vytvářet téměř všechny druhy programů od konzolových aplikací, databázových programů, webových aplikací a stránek až po formulářové programy pro platformu Windows. Vhodný je však pro vývoj softwarových komponent, nikoli hardwarových. Při učení s tímto jazykem jsem čerpal z webových zdrojů[16], kde jsou popsány a na příkladech vysvětleny různé konstrukce i používání databází či propojení s webovými stránkami. Jako vývojové prostředí jsem zvolil Microsoft Visual Studio[17], které umožňuje vytvářet všechny výše uvedené typy aplikací. Dále také poskytuje velké množství individualizace prostředí včetně chytrého editoru kódu (IntelliSense), které upozorňuje na možné chyby a našeptává další možnosti, což velice usnadňuje práci.

Pro práci s daty jsem využil databázového systému MSSQL, který vyvinula společnost Microsoft. Jedná se o velice výkonný a používaný relační databázový systém založený na architektuře klient/server. Dokáže zvládat velké množství transakcí a je využíván po celém světě v různých odvětvích. Existuje již více než tucet verzí, já jsem využíval verzi 12.0 z roku 2014. Pro práci s tabulkami jsem sáhl po programu SQL Server 2014 Management Studio[18], který práci značně zjednodušuje.

Webové stránky jsem vytvořil v jazyce HTML, který umožňuje vytvářet webové dokumenty. Tento jazyk je založen na značkách, které přidávají a upravují jednotlivé části dokumentu. Značky je možné do sebe zanořovat v různých úrovních a většinou se skládají z počáteční a koncové značky. Grafická úprava webových stránek jen pomocí jazyka HTML je poměrně omezená. Proto jsem použil také Bootstrap. Ten obsahuje již vytvořené šablony pro tlačítka, formuláře či tabulky. Jeho aplikace

[16] http://www.itnetwork.cz/csharp

[17] https://www.visualstudio.com/cs-cz/visual-studio-homepage-vs.aspx

[18] https://www.microsoft.com/en-us/download/details.aspx?id=42299

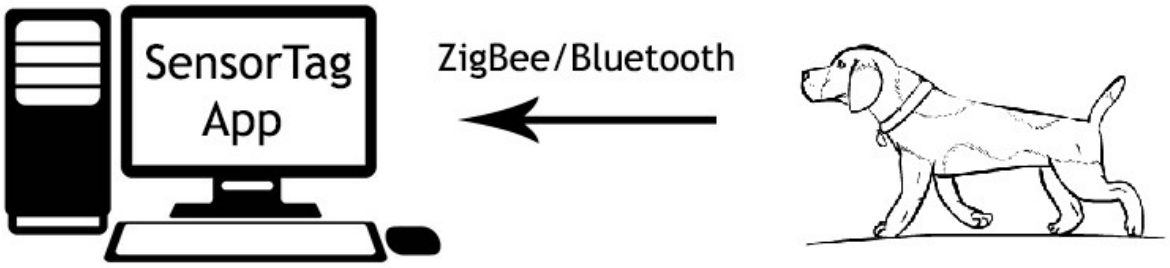
značně ulehčuje formátování prvků na stránkách. Jedná se o soubor šablon založených na HTML a CSS technologiích pro jednoduchou vizuální úpravu stránek.

4.2 Návrh

4.2.1 Původní návrh

Prvotní návrh byl vytvořit okenní aplikaci pro platformu Windows. Aplikace by zobrazovala hodnoty z čidel, také hodnoty přepočítané na vzdálenost obojku od snímače, případně rychlost pohybu. Pracovala by v reálném čase bez ukládání záznamů a sloužila by čistě pro sledování pohybů zvířete. Jednoduché grafické rozhraní by zobrazovalo také pozici psa například na zahradě. Využít jsem chtěl rozhraní Blank App v C#, což je okenní aplikace přenositelná na všechna zařízení s platformou Microsoft Windows 8.1 a Windows 10. Komunikace by probíhala přes protokol ZigBee, který se mě vzhledem k energetické úspornosti a dosahu spojení zdál jako nejvhodnější. Nejdříve tedy bylo potřeba přehrát firmware v zařízení SensorTag, který byl původně nastaven na Bluetooth. Přehrání se dalo provést přes mobilní aplikaci na Android od výrobce zařízení Texas Instruments. Přes tuto aplikaci je pak možné pomocí Bluetooth spojení přehrát firmware na protokol ZigBee. Celý proces zabere sotva pár minut. Další nutností bylo použít USB přijímač ZigBee, neboť počítače jej běžně nemají. K dispozici jsem měl přijímač značky Webee. V C# se mi podařilo přijímač najít mezi všemi připojenými zařízeními a pokusil jsem se z něj číst. Data, která se mi podařilo načíst, však byla velmi krátká a obsahovala pouze informace o zařízení. Rozhodl jsem se je tedy ověřit v programu Packet Sniffer[19], který také pochází od firmy Texas Instruments. Tento program data podrobně zobrazuje. Lze zde nastavit, jestli se data budou přijímat přes protokol ZigBee či protokol Bluetooth. Při zvolení protokolu ZigBee se data opravdu posílala špatně. Neobsahovala hodnoty z kteréhokoliv čidla. Důvodem je nejspíše špatné přehrání firmwaru v SensorTagu. Na oficiálních internetových stránkách výrobce je uvedeno, že přehrání firmwaru přes mobilní aplikaci nemusí být vždy spolehlivé. V případě, že takto přehraný firmware neposílá správně data, je nutné jej přehrát pomocí speciálního zařízení. Jeho cena se však pohybuje okolo 5000 Kč. K této variantě bylo k sehnání malé množství zdrojů, ze kterých bych mohl čerpat. Od tohoto řešení jsem byl tedy nucen upustit. Níže umístěný obrázek demonstruje návrh, který jsem zamýšlel. Platí pro použití protokolů ZigBee, ale i Bluetooth, které jsem popsal dále.

[19] http://www.ti.com/tool/packet-sniffer



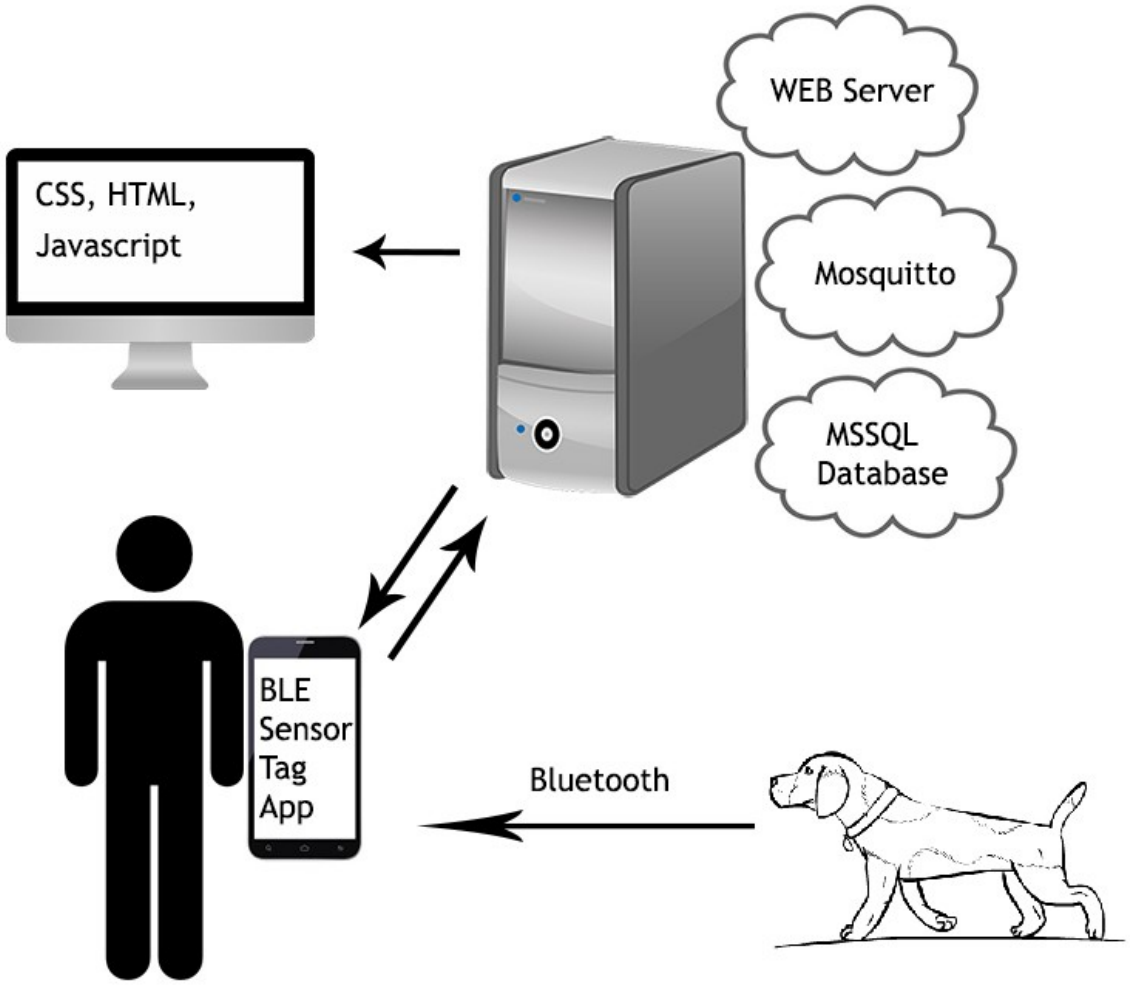
Obrázek 4.1: Zigbee/Bluetooth komunikace

4.2.2 Použití Bluetooth

Návrh s okenní aplikací jsem však chtěl ponechat. Nabízela se tedy možnost posílání dat přes Bluetooth. Nejprve jsem musel zpátky přehrát firmware v SensorTagu na protokol Bluetooth. Přehrání firmwaru jsem provedl stisknutím tlačítek uvedených v kapitole 7 po dobu 6 sekund. Dioda poté začne blikat opět zeleně, což značí, že firmware je zpátky správně přehrán. Jelikož mobilní aplikace od Texas Instruments data přijímá přes protokol Bluetooth, neměl by s přijímáním dat v C# být problém. Při hledání možností programování v C# s protokolem Bluetooth, jsem narazil na knihovnu GATT [11]. Tato knihovna definuje, jak lze mezi dvěma Bluetooth Low Energy zařízeními komunikovat a přenášet data. Koncept této knihovny určuje, že vysílací zařízení je označeno jako GATT Server a přijímací stanice jako GATT Client. Celé spojení je rozděleno do tří základních částí. První je označena jako Profiles. Jsou zde předdefinována různá zařízení, se kterými je možno komunikovat. Jedná se například o měřič srdečního tepu. Další část je Services. Tato sekce umožňuje rozdělení dat do logických částí. Těmi mohou být například různé senzory v jednom zařízení, tak jak je tomu u SensorTagu. Dílčí části jsou identifikovány jednoznačným 128bitovým klíčem nazvaným UUID. Poslední částí je Characteristics, což je z hlediska úrovně nejnižší úroveň celého konceptu. Zde dochází k zapouzdření dat podle například senzorů. Pomocí této knihovny se mi podařilo připojit SensorTag. Bohužel ale nedocházelo k přenosu dat. Problémem mohl být Bluetooth přijímač v mém notebooku. Proto jsem vyzkoušel externí USB přijímač značky Laird[20], který mi byl zapůjčen firmou CAMEA. To však také problém nevyřešilo. Na fóru firmy Texas Instrument jsem se dočetl, že mezi SensorTagem a některými Bluetooth přijímači dochází ke špatné komunikaci. Doporučuje se proto používat přijímače od Texas Instruments. V České republice se však tyto přijímače neprodávají. Bylo by tak nutné jej objednat ze zahraničí, což by tvorbu projektu pozdrželo. Navíc by se cena celého projektu dosti zvýšila. Proto jsem se rozhodl pro jiné řešení.

[20] http://www.lairdtech.com/Products/BT820

4.2.3 Konečný návrh



Obrázek 4.2: Konečný návrh

Vzhledem k výše uvedeným problémům jsem zvolil alternativní řešení, kdy využívám klienta Mosquitto [9] a mobilní aplikaci firmy Texas Instruments, která umožňuje zasílání dat přes internet. Na výše uvedeném obrázku lze vidět, jak celý projekt funguje. Ze SensorTagu umístěného na obojku se data posílají protokolem Bluetooth na mobilní zařízení. Zde běží aplikace BLE SensorTag od Texas Instruments, která přeposílá data přes internet na Mosquitto klienta. Na počítači se veškerá data uloží, zpracují a vyhodnotí. Poté dojde k předání dat webovému serveru, který je zobrazí na webových stránkách. Následuje detailnější popis jednotlivých částí.

4.3 Mosquitto

Klienta jsem si stáhl a nainstaloval podle návodu a poté jej pouze spustil. Po zadání příkazu *netstat –an* v příkazové řádce se vypíšou všechna síťová připojení. U protokolu TCP lze najít adresu 0.0.0.0:1883. Tato adresa říká, že server funguje, avšak není spuštěn na konkrétní adrese, ze které bude data přijímat. Port 1883 je vyhrazen nešifrovanému přijímání dat přes mosquitto.

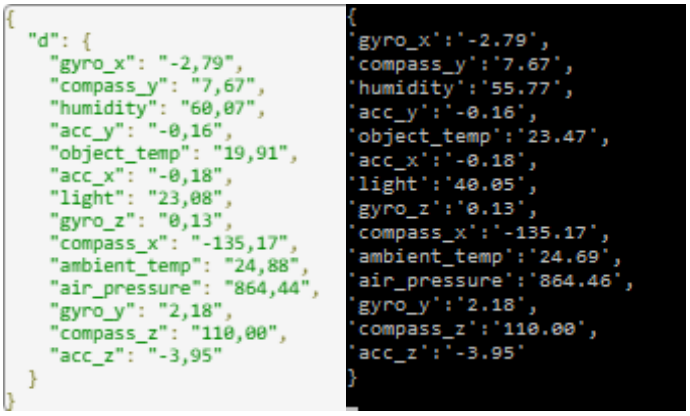
Po předchozích zkušenostech jsem nezačal ihned programovat zmíněného klienta. Funkčnost tohoto řešení jsem si ověřil pomocí rozšíření MQTTLens[21] dostupného v prohlížeči Google Chrome. Po jeho nainstalování a spuštění lze vytvořit spojení. Nastavení je velice jednoduché. Uvede se pouze IP adresa počítače, uživatelské jméno a heslo a tímto je spojení vytvořeno. U něj je poté třeba ještě nastavit název spojení, neboť jich může být spuštěno více najednou. Nastavení spojení v mobilní aplikaci je také velice jednoduché. Po spárování se SensorTagem je možné zvolit sdílení dat přes internet. Zde nastavíme stejné hodnoty jako u klienta, tedy adresu počítače, na kterém klient běží, uživatelské jméno a heslo a zejména pak název spojení. Ještě dodáme port 1883 a můžeme data, která přijmeme ze zařízení, přeposílat. Pokud chceme, aby bylo toto přeposílání možné, je potřeba počítač a telefon připojit ke stejné Wi-Fi síti. Do budoucna je možné mosquitto implementovat jako veřejného klienta. Poté by nebylo nutné, aby mobilní telefon a počítač byly ve stejné Wi-Fi síti. Pro potřeby této bakalářské práce je však současný stav uspokojující. Dále jsem musel vytvořit nové pravidlo v bráně Windows Firewall, které povoluje použití programu mosquitto a portu 1883. Bez něj je přenos zakázaný a data se tedy neposílají. Po provedení všeho výše zmíněného se zobrazují v rozšíření MQTTLens přijatá data. Formát zpráv velice připomíná JSON. Jsou zde však určité odlišnosti jako čárky a použité uvozovky. Formát zprávy je vidět na obrázku 4.3 na straně 18.

Pro programování Mosquitto klienta je nutné využít knihovny *uPLibrary.Networking.M2Mqtt*, která poskytuje veškeré nástroje potřebné k vytvoření klienta. K vytvoření instance třídy *MqttClient* je nutná IP adresa počítače, na které bude přijímat. Vygeneruje se nové ID klienta a vytvoří se spojení, které se doplní o jeho název. Poslední nutností je vytvoření metody, která se spustí při každém přijetí zprávy. Metoda se jmenuje *client_MqttMsgReceived* a zpracovává přijatou zprávu.

4.4 Zpracování dat

Podobnost formátu zprávy s formátem JSON je velice patrná z obrázku umístěného níže. Stačilo tak udělat několik změn v této zprávě a bylo možné pracovat s daty jako s formátem JSON, což značně ulehčilo manipulaci s nimi. Toto řešení mi umožňuje provádět serializaci a deserializaci dat, kterou využívám u předávání dat webovým stránkám a ukládání/načítání dat z databáze. Změny, které jsem udělal, jsou následující. Na začátku a konci zprávy se nacházejí znaky, které jsou oproti JSON nadbytečné. Ty jsem tedy odstranil. Jednalo se o prvních osm znaků a poslední znak řetězce. Druhým rozdílem je výskyt dvojitých uvozovek. JSON jako takový je schopen je zpracovávat. Při serializaci v C# však docházelo k chybám. Poté, co jsem je všechny nahradil jednoduchými, již serializace proběhla úspěšně. Nahrazení bylo dosáhnuto použitím metody *Replace* na celý text zprávy. Posledním rozdílem je použití čárky u desetinného čísla namísto tečky. Všechna čísla však nejsou desetinná (např. signalizace stisku tlačítka). Čárky, které se nevyskytují na konci řádku, jsou nahrazeny tečkou. Porovnání obou formátu je vidět níže. Vlevo se nachází původní formát, vpravo poté JSON.

[21] https://chrome.google.com/webstore/detail/mqttlens/hemojaaeigabkbcookmlgmdigohjobjm

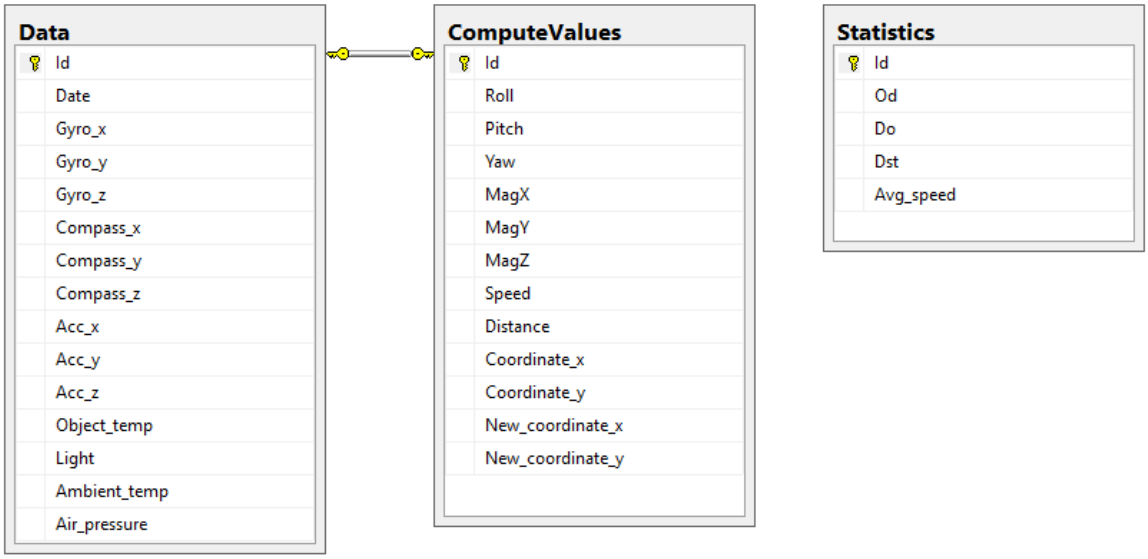


Obrázek 4.3: Porovnání původní a upravené zprávy s daty

Abych mohl provádět deserializaci dat, vytvořil jsem si třídu *JsonConv*. V této třídě jsou parametry, které reprezentují všechna čidla SensoTagu, respektive každou osu pohybových čidel. Deserializaci provádím metodou *DeserializeObject* ze třídy *Newtonsoft.Json*, kterou jsem musel do projektu doinstalovat. Dále už následuje uložení dat do databáze. Ukládání dat je popsáno v kapitole 4.5. Po uložení následuje výpočet hodnot ze získaných dat.

4.5 Ukládání dat

Pro celý projekt jsem vytvořil databázi *SensorTagData* v databázovém nástroji MSSQL. Jak lze vidět na obrázku níže, databáze obsahuje pouze 3 tabulky, více není potřeba. První z nich, pojmenovaná *Data,* obsahuje surová data, která přijdou ze SensorTagu. Druhá, s názvem *ComputeValues*, obsahuje hodnoty, které jsou vypočítány z dat v tabulce *Data*. Jedná se například o úhly, rychlost či vzdálenost. Původně zde nebyly položky Coordinate_x, Coordinate_y, New_coordinate_x a New_coordinate_y. Počítají se však ještě před uložením zbylých hodnot do databáze, tudíž jsem se rozhodl je přidat do této tabulky. Navíc všechna tato data se posílají na webové stránky, kde se zobrazují. Toto řešení vede také k jednoduššímu databázovému dotazu pro výběr dat a není potřeba vytvářet další tabulku s vazbami k tabulkám *ComputeValues* a *Data*. Ke každému záznamu v tabulce *Data* existuje právě jeden záznam s vypočítanými hodnotami v tabulce *ComputeValues*. Proto jsou tabulky propojeny vztahem 1:1. Poslední tabulkou databáze je *Statistics*, ve které jsou uloženy statistiky pohybu zvířete. Jedná se o průměrnou rychlost, celkovou vzdálenost a časový interval říkající odkdy dokdy byla data naměřena. Jako primární klíče všech tabulek jsem zvolil číselný údaj *Id*, který se automaticky navyšuje u každé tabulky s přidáním dalšího záznamu.



Obrázek 4.4: Databáze

Všechny hodnoty, které přicházejí ze SensorTagu jsou desetinná čísla zaokrouhlená na dvě desetinná místa. S těmi si databáze samozřejmě umí lehce poradit. Pro rychlejší odezvu databáze jsem se však rozhodl čísla ukládat jako typ Integer zapsaný na 32 bitech. Každou hodnotu tedy vynásobím stem a až poté uložím. Provedení databázových dotazů je tak rychlejší. Stejně nakládám i s vypočítanými hodnotami, které jsou ukládány naprosto stejně.

Aby bylo v C# možné využívat databázi MSSQL, bylo nutné využít knihovny *SqlClient*. Připojení k databázi se provede ihned při spuštění aplikace pomocí metody *ConnectDB* ze třídy *Database*. Navázání spojení je uskutečněno přes *connectionString*, který obsahuje úplnou cestu k databázi.

Veškerá práce s databází je prováděna právě přes třídu *Database*. Jsou zde metody jak pro zápis dat, tak pro jejich čtení. Každá operace vyžaduje otevření spojení s databází. To se děje přes výše zmíněný *connectionString*, který je ve třídě uveden jako globální proměnná. Po vykonání jakékoliv operace je spojení zase uzavřeno.

Dotazy pro práci s databází jsou napsány v jazyce MSSQL. Použity jsou jako textový řetězec a předány metodě *SqlCommand*, která dotazy vykonává. Celkem se ve třídě *Database* nachází 10 metod.

Provázání tabulek *Data* a *ComputeValues* vztahem 1:1 způsobuje jeden problém. Počítání rychlosti a vzdálenosti je provedeno způsobem, kdy jsou potřebné dva záznamy dat ze SensorTagu. Důležitý je zejména časový údaj pořízení záznamu. Při spuštění aplikace by vznikla situace, kdy by do tabulky *Data* byly přidány dva záznamy a do *ComputeValues* pouze jeden. Záznamy v druhé zmíněné tabulce by tedy neodpovídaly záznamům v tabulce *Data*. S každým novým spuštěním aplikace by se navíc rozdíl v *Id* mezi odpovídajícími záznamy zvyšoval a vypočítané hodnoty by neodpovídaly příslušnému záznamu v tabulce *Data*. Problém jsem odstranil přidáním záznamu s nulovými hodnotami do tabulky *ComputeValues* při spuštění aplikace. Jelikož by hodnoty nebylo možné vypočítat pouze z jednoho záznamu dat ze SensorTagu, jeví se mi toto řešení jako přijatelné. V případě potřeby lze takto také zjistit, kdy byl program spouštěn.

4.6 Výpočty

Z dat, která přichází ze SensorTagu, se provádí několik výpočtů. Je třeba zjistit směr pohybu zvířete, jeho rychlost a dráhu, kterou urazilo. Dále se provádí počítání celkové uražené vzdálenosti za určitou dobu. Metody toto provádějící jsou umístěny ve třídách *Computation* a *ECompass*.

4.6.1 Rychlost a vzdálenost

Vzorce a postupy v této podkapitole jsou převzaty z [2]. Výpočet rychlosti probíhá ve třídě *Computation*. Do metody Speed jsou jako parametry přijatá data ve formátu *JsonConv* a pole typu *DateTime* se dvěma položkami, časovým razítkem předešlé zprávy a s razítkem té současné. První parametr obsahuje zrychlení naměřená pomocí akcelerometru, která jsou pro výpočet nezbytná. Metoda vrací pole s vypočítanou rychlostí a dráhou. Použil jsem následující vzorec:

$$v(t) = \int_{t=0}^{t} a.dt \tag{4.1}$$

Integrováním zrychlení podle času lze získat rychlost pohybu. Jelikož je rychlost zaznamenávána ve třech osách, je nutné integrovat hodnotu z každé z nich samostatně. Jelikož jsou data měřena v přesných intervalech, je možné použít následující počty:

$$vx \mathrel{+}= ax * dt;$$
$$vy \mathrel{+}= ay * dt;$$
$$vz \mathrel{+}= az * dt;$$

kde *a* reprezentuje zrychlení v dané ose, *v* pak vypočítanou rychlost v daném směru a *dt* časový interval mezi dvěma měřeními. Pro přesnost měření je důležité odečíst gravitační zrychlení ve vertikální ose, které činí 0,981 g. Absolutní rychlost se vypočítá následovně:

$$|v| = \sqrt{vx + vy + vz} \tag{4.2}$$

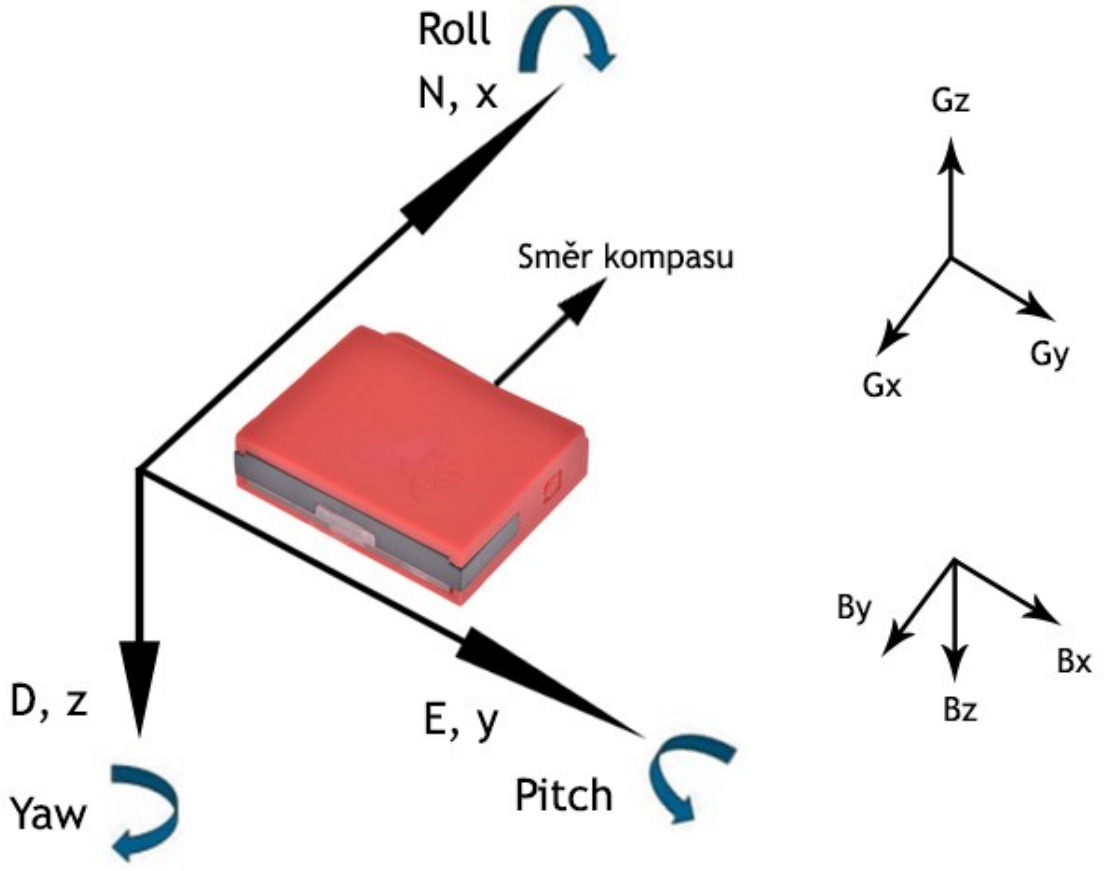
Pokud známe rychlost a časový interval, můžeme pro výpočet vzdálenosti využít základní vzorec pro získání vzdálenosti:

$$s = v * t \tag{4.3}$$

Výpočet rychlosti a vzdálenosti bohužel není zcela přesný. Důvodem jsou intervaly v posílání dat ze SensorTagu. Ta přichází každou vteřinu, což je příliš velký interval. V mobilní aplikaci lze změnit interval snímání hodnot ze senzorů, avšak ne interval odesílání dat. Zkusil jsem tedy změnit nastavení firmwaru v programu Code Composer Studio, kterým lze firmware upravit a následně přehrát. Změnil jsem interval pro odesílání dat na 500 milisekund, data však stále chodila každou vteřinu. Domnívám se, že periodu odesílání dat přes internet má v režii právě mobilní aplikace. Tuto domněnku potvrzuje i fakt, kdy při vypnutí libovolného senzoru ve firmwaru jsou data na straně aplikace zase přijímána. Taktéž při změně periody snímání z čidel se vždy zobrazí výchozí hodnota. Aplikace si tudíž nejspíše sama nakonfiguruje zařízení při párování s ním. Ke zdrojovým kódům jsem se bohužel nedostal a nemohu tak ověřit, zda je tomu skutečně tak. Nepodařilo se mi tedy zjistit, jestli jsou přijatá data nějak zprůměrována za dobu tohoto intervalu, či zda se jedná o hodnotu v danou chvíli, kdy se data odesílají. Podle měření, která jsem provedl, se jedná nejspíše o zprůměrované hodnoty.

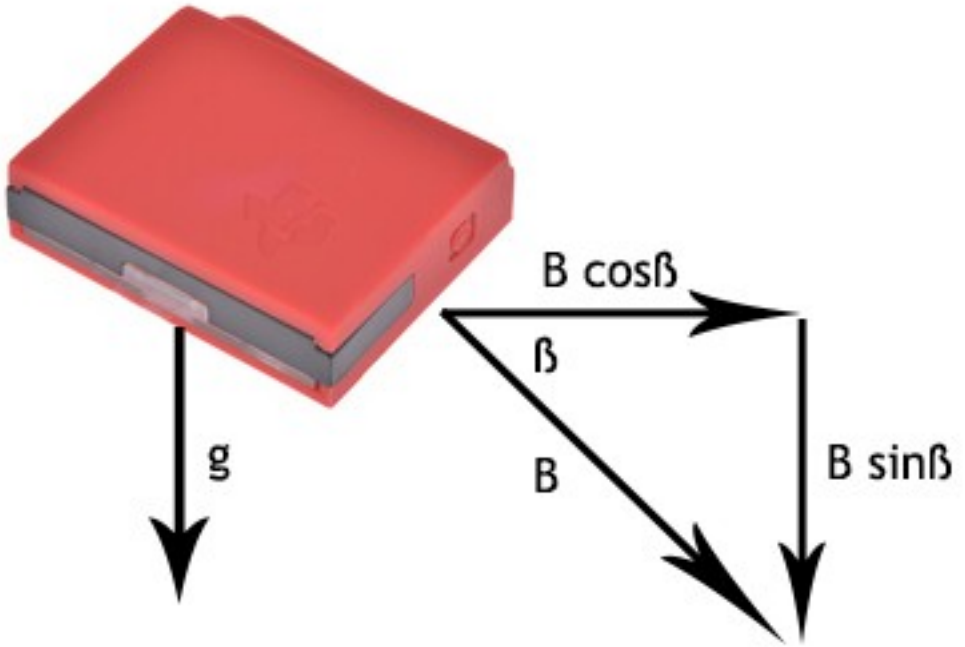
4.6.2 Směr

Tato podkapitola a algoritmy zde uvedené jsou převzaty z [3]. Určení směru je zásadní pro zjištění a vykreslování pohybu zvířete. Výpočet směru se provádí ve třídě *ECompass* a potřebné jsou hodnoty ze všech tří os kompasu a akcelometru umístěných v SensorTagu. Ten využívá standardu NED (North, East, Down) pro koordinaci os zařízení. Vzhledem k upevnění čidel k desce se hodnoty měření akcelerometru a magnetometru mění v závislosti na orientaci zařízení. K výpočtu jsem tedy mohl využít algoritmu uvedeného zde [3]. Jak je vidět na obrázku níže, osa *X* ukazuje k severu, osa *Y* k východu a osa *Z* míří směrem dolů, tedy kolmo k povrchu země. Hodnoty úhlů *Yaw*, *Roll* a *Pitch* mají kladné znaménko ve směru hodinových ručiček. Z obrázku je také patrné, že směr osy *Gy* u akcelometru je stejný s osou y u zařízení. Naproti tomu osy *Gx* a *Gz* mají převrácené hodnoty. U magnetometru je osa *Bz* ve stejném směru, osy *x* a *y* už ale směru os zařízení neodpovídají. Osa *y* by měla být nastavena ve směru *x* a *x-ová* osa do *–By*. Úhel *Yaw* tedy indikuje směr natočení vůči severu. S jeho hodnotou dále počítám při vykreslování směru pohybu.



Obrázek 4.5: Směry os SensorTagu

Kontrolu, že zařízení s osami pracuje stejně, jak je uvedeno ve zdroji, je možné provézt jednoduše pomocí mobilní aplikace. Při položení zařízení na stůl jako je ukázáno na obrázku níže, je u hodnoty osy *Z* vidět přetížení 1g, při otočení zařízení o 180° se hodnota změní na -1g. Horizontální složka geomagnetického pole vždy směřuje k magnetickému severnímu pólu.



Obrázek 4.6: Magnetismus SensorTagu

Výpočet magnetismu je proveden pomocí vzorce:

$$B_r = B \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix} \tag{4.4}$$

Celý algoritmus využívá pouze standardních knihoven jazyka C# a je počítán v celých číslech. Takto nedochází k žádným zaokrouhlovacím chybám. Vzhledem k tomu, že hodnoty z magnetometru jsou měřeny v μT, není důvod k žádnému převodu hodnot, neboť algoritmus pracuje právě s těmito jednotkami. Podrobný popis celého algoritmu je dostupný zde [3]. Níže uvedu jen seznam metod s jejich základním popisem:

iECompass – Hlavní metoda odkud se volají ostatní metody.

iTrig – Počítá úhly sinus a cosinus z obrázku výše, které jsou potřebné k dalším výpočtům.

iHundredAtanDeg – Výpočet úhlu ATAN.

iHundredAtan2Deg – Přiřazení výsledku do správného kvadrantu. Výsledek je úhel ve stupních vynásoben stem. Takto je možné uložit data s chybou pouze 0,01° na 16 bitech.

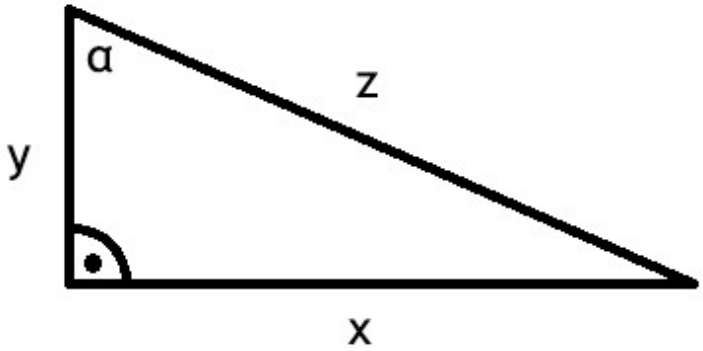
iDivide – Upravuje výsledek podle znaménka čitatele a jmenovatele.

Algoritmus pro vytvoření elektronického kompasu funguje vcelku obstojně. Chyba u měření se pohybovala do 15° a to i na místech, které pro měření magnetismu nejsou vhodné, např. zastavěné oblasti s elektrickým vedením. Nasazením pokročilých optimalizačních algoritmů by bylo možné výsledné hodnoty zpřesnit. V nezastavěných oblastech jsou výsledky velmi uspokojivé. Chyba se pohybuje pouze v jednotkách stupňů. Toto však platí pouze, pokud je zařízení v klidu

nebo konstantním pohybu. Jestliže je však vystaveno příliš náhlým změnám směru a zejména natáčení, výsledky jsou mnohem méně přesné a dochází k velkým výkyvům vypočítaných hodnot. Pro výpočet kompasu se totiž využívá kromě magnetometru také akcelerometr. Ten při pohybu samozřejmě vykazuje zcela jiné hodnoty než v případě kdy je zařízení v klidu. Bohužel psi často provádí rychlé změny směru včetně skákání apod., tudíž je výpočet směru pohybu značně ovlivněn.

4.6.3 Výpočet souřadnic pro vykreslení pohybu

Vykreslování se provádí na webových stránkách. Zvolil jsem komponentu *Canvas* (dále označovanou jako plátno), která je čtvercové tvaru s rozměry 450x450 pixelů. Vykreslování se provádí od středu. Jednotlivé strany reprezentují čtyři světové strany sever, jih, východ a západ. Pohyb je vykreslován vzhledem k severu podle úhlu *Yaw*, jak jsem uvedl výše v kapitole 4.6.2. Pohyb po plátně je realizován pomocí souřadnic na osách *X* a *Y*. Počáteční souřadnice jsou tedy 225 pro obě osy. Nové souřadnice se počítají v metodě *ComputeCoordinates* ve třídě *Computation*. Nové souřadnice lze získat pomocí vzorců pro výpočet odvěsen u trojúhelníku.



Obrázek 4.7: Výpočet nových souřadnic

Ze všech proměnných na výše uvedeném obrázku znám hodnotu alfa (úhel *Yaw*) a přeponu *z*. Tou je vzdálenost, kterou zvíře urazilo za jednotku času. Výpočet probíhá po každém přijetí nových dat ze SensorTagu, proto mohu takto nové souřadnice získávat. Pomocí vzorce (4.5) získáme odvěsnu *y*. Vzorce (4.5) a (4.6) jsou převzaty z [4].

$$y = z * \cos\alpha \tag{4.5}$$

Pro výpočet odvěsny *x* poté už stačí jen použít Pythagorovu větu, neboť se jedná o pravoúhlý trojúhelník.

$$x = \sqrt{z^2 - y^2} \tag{4.6}$$

U počítání hodnoty přepony *x* je nutné zjistit znaménko hodnoty *y*. Pokud je totiž *y* záporné, umocnění této hodnoty v Pythagorově větě způsobí, že by souřadnice *x* pouze narůstala a pohyb by se tak konal pouze v jednom směru osy *x*. V případě, že je hodnota *y* záporná, je změněno i znaménko hodnoty *x*, což zaručuje také odečítání hodnoty *x*. Nové souřadnice pak vzniknou součtem/odečtením stávajících hodnot s přeponami *x* a *y*. U těchto výpočtů bylo nutné použít knihovnu *Math* v C#, která umožňuje počítání cosinu a odmocniny. Poměr vzdálenosti k pixelům na plátně je 1:1, či-li jeden metr v reálu odpovídá jednomu pixelu na plátně. Pokud dojde k překročení hranic plátna, pohyb se znovu vykresluje od jeho středu. Při dalším pokračování na této práci by bylo vhodné toto ošetřit například oddálením obrazu plátna. Takto by bylo možné vykreslovat pohyb ve větším rozsahu. Naproti tomu

k překročení plátno by nemělo nikdy dojít, neboť vzdálenost z jeho středu k okraji odpovídá v reálu 225 metrům. Takto velký dosah Bluetooth v SensorTagu neposkytuje.

4.6.4 Statistiky

Všechna vypočítaná data jsou ukládána do databáze. Nacházejí se v tabulce *Statistics*. Jedná se o hodnoty průměrné rychlosti, vzdálenosti a časového intervalu. Zvolil jsem interval jedné hodiny. Pro každý den tedy v tabulce vznikne maximálně 24 záznamů. Celková vzdálenost je získána sčítáním dílčích vzdáleností, průměrná rychlost pak vydělením celkové vzdálenosti jednou hodinou. Při spuštění programu se jako počáteční čas zvolí započatá hodina prvního záznamu. Dojde tedy k zaokrouhlení na celou hodinu směrem dolů. Až rozdíl počátečního času intervalu a přijatého záznamu přesáhne jednu hodinu, je záznam uložen a počítání započíná znova.

4.7 Prezentace dat

Jak jsem zmínil již v kapitole 4.2.1, od původního návrhu s okenní aplikací jsem upustil a rozhodl se pro prezentování dat na webových stránkách. Toto řešení nabízí širší možnosti použití. Takto je možné data sledovat na všech platformách, včetně těch mobilních. Webové stránky také umožňují poměrně velkou flexibilitu pro případně změny a rozšíření.

4.7.1 Předávání dat

Webové stránky běží na straně klienta. Aplikace napsaná v C# zde pouze umožňuje předávat data těmto stránkám, respektive funkcím napsaným v jazyce JavaScript. To zprostředkovávají metody ve třídě *tagApiController*, které využívají knihovnu *System.Web.Http*. Ta poskytuje veškeré nástroje k tomu potřebné. Třída obsahuje celkem čtyři metody.

Tou první je *getData*, která je typu *HttpGet*. Nejdříve je zde volána metoda pro načtení požadovaných dat z databáze. Jedná se o data z tabulek *Data* a *ComputeValues*, která jsou v dotazu spojena pomocí klauzule *INNER_JOIN*. Ta jsou poté uložena do proměnné třídy *WebData*, která obsahuje všechny položky, které jsou v dotazu vybrány. Vybráno je posledních 10 přidaných záznamů, proto jsou data předávána v datové struktuře List. Dále jsou data serializována do formátu JSON pomocí metody *SerializeObject* z knihovny *JsonConvert*. Poté jsou vložena do proměnné typu *HttpResponseMessage* jako string, který je navrácen funkci *IndexController* v souboru *controller.js*, odkud se *getData* volá.

Druhá metoda se nazývá *getStatistic*, která je taktéž typu *HttpGet*. Její funkce a implementace je prakticky stejná jako u výše zmíněné metody. Jedná se však pouze o data z tabulky *Statistic*, taktéž 10 posledních přidaných záznamů. Ty jsou tentokráte nahrané do proměnné třídy *WebStatisticData*. Následující postup serializace a převedení do formátu JSON je stejné jako u *getData*. *GetStatistic* je volána taktéž ze souboru *controller.js*, přesněji z funkce *StatController*, kam jsou data předána.

Třetí metodou je *getFilterStatistics*, jež je typu *HttpPost*. Ta vybírá záznamy z tabulky *Statistics* podle zvoleného data. Databázový dotaz vybere všechny záznamy s tímto datem, kterých však může být maximálně 24. Další postup je stejný jako u metody *getStatistics*.

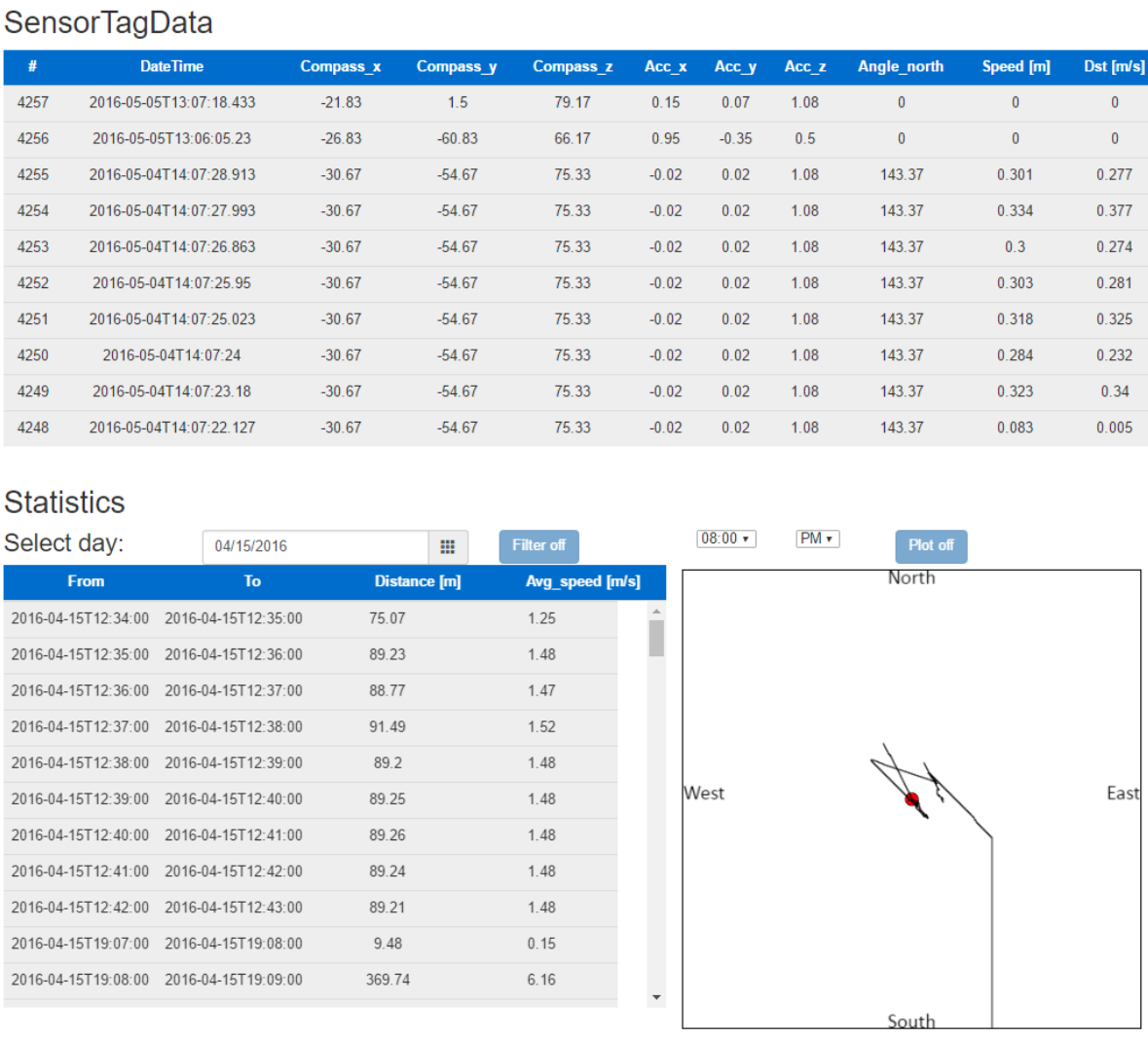
Tou poslední je *getFilterCoordinates*, kde je předán jako parametr časový údaj a je také typu *HttpPost*. Ta volá metodu *LoadCoordinates*, která načte veškeré souřadnice od časového údaje

po tento časový údaj plus jedna hodina. Hodnoty jsou uloženy do proměnné třídy *Coordinates*, serializovány a navráceny zpět.

4.7.2 Webový server

Webový server je implementován ve třídě *StartUp*. Využívá knihoven *Microsoft.Owin* a *System.Web.Http*. Nastavení serveru probíhá v metodě *Configuration*. Jedná se například o cestu k souborům, které slouží k vizualizaci webových stránek nebo typ dat, který se předává. V metodě Run dojde ke spuštění serveru. Pro potřeby této aplikace a její testování bylo dostačující používat URL adresu *http://localhost:8081/*. Do budoucna by bylo vhodné stránky umístit na internet.

4.7.3 Webové stránky



Obrázek 4.8: Webové stránky

Vzhledem k nevelkému množství dat, které se zobrazuje, jsem si vystačil s jednou stránkou. Jsou na ní umístěny tři prvky: dvě tabulky a jedno plátno. Celý web je vidět na obrázku výše. První tabulka s názvem *SensorTagData* obsahuje data z tabulek *Data* a *ComputeValues*. Vždy zobrazuje posledních

10 přidaných záznamů. Seřazeny jsou sestupně od posledně přidaného. Pod touto tabulkou je další, která slouží k zobrazení dat z tabulky *Statistics*. Z databáze je možné si zobrazit statistiky z konkrétního dne. Pro výběr data jsem využil komponenty *datepicker*, kde je možné zvolit rok, měsíc a den. Vedle tabulky se nachází plátno, které vykresluje pohyb zvířete. Červený puntík značí střed, stěny jsou označeny nápisy příslušných světových stran. Vykreslování lze provádět také zpětně. Pomocí *datapickeru* a dvou rolovacích seznamů lze vybrat přesnou hodinu konkrétní dne, pro kterou se poté vykreslí celý pohyb. Po výběru data a času stačí stisknout tlačítko „Plot On“.

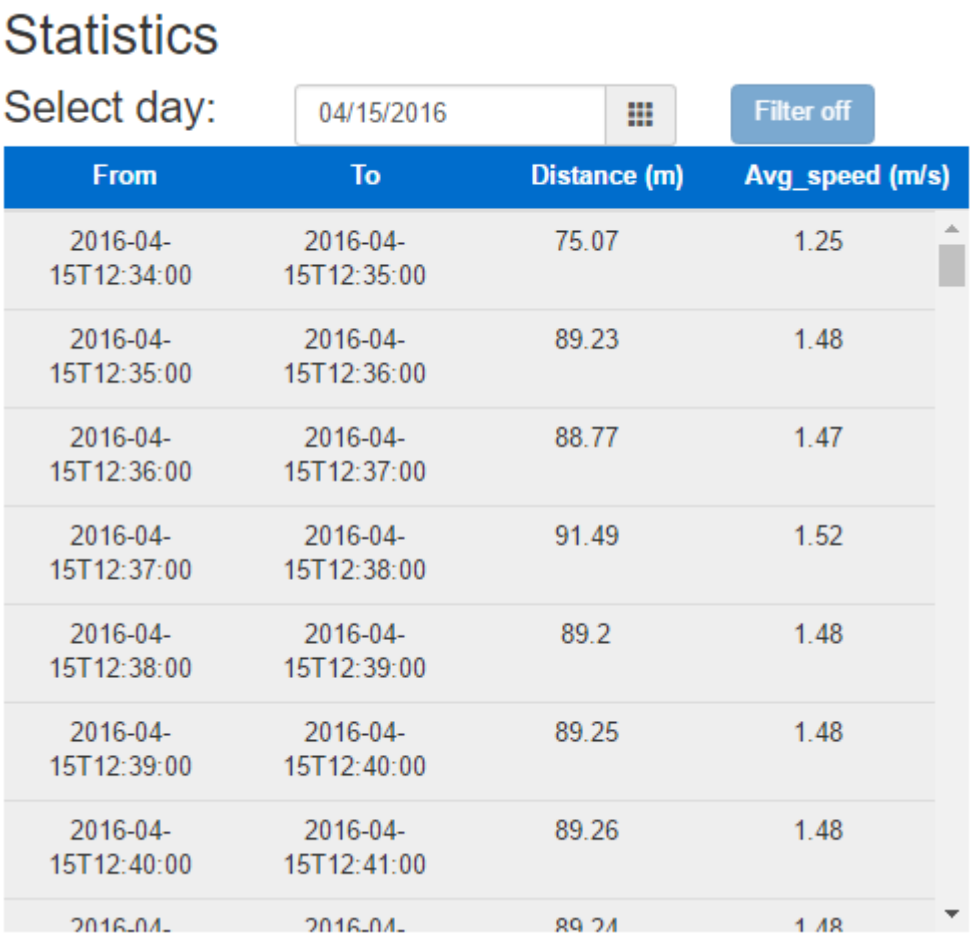
4.7.4 Funkčnost webové stránky

Získávání dat do tabulek je implementováno v jazyce JavaScript. V souboru *controller.js* jsou implementovány dvě funkce: *IndexController* a *StatController*, které komunikují s metodami v C#, které jsou popsány v kapitole 4.7.1.

Prvně jmenovaná volá metodu *getData* v C#, která ji vrátí požadovaná data ve formátu JSON. Ty jsou poté předány do souboru *index.html*, přesněji do dané značky, kde v cyklu dojde k vypsání dat. Tato funkce se periodicky provádí každou vteřinu, která přibližně odpovídá intervalu přidávání nových záznamů do tabulek Data a *ComputeValues*, odkud jsou data načítána. S každým novým záznamem však dochází k volání funkce pro vykreslování pohybu na plátně. Té jsou předány všechny potřebné souřadnice a dojde k vykreslení posledního záznamu.

StatController je rozdělena do tří funkcí. Pokud není zapnut filtr na zobrazení statistik, na stránce je tlačítko „Filter on“. Bez jeho zapnutí probíhá volání funkce *getStatistics*, která navrátí posledních 10 těchto záznamů, které jsou také předány do příslušné značky v souboru *index.html*. V případě, že z komponenty *datepicker* zvolíme libovolné datum a poté stiskneme tlačítko „Filter on“, je aktivní druhá funkce, která datum předá metodě *getFilterStatistics*. S daty se dále pracuje naprosto stejně. Perioda, s jakou se funkce vykonávají, zde může být dosti vysoká. Jelikož se tyto záznamy mohou v databázi měnit nejdříve co hodinu, je možné takovýto interval zvolit i zde. Pokud je filtr zapnut, tlačítko na stránce se změní z „Filter On“ na „Filter Off“. Pokud se stiskne, začnou se znova zobrazovat poslední přidaná data. Pokud stiskneme tlačítko „Plot on“, dojde k vykreslení pohybu zvířete z dané hodiny, které je však zrychlené 50 krát. Vykreslování pohybu v reálném čase je v tu dobu zastaveno, po stisknutí tlačítka „Plot off“ je opět plátno vyčištěno a probíhá standardní vykreslování pohybu v reálném čase.

V HTML jsou použity direktivy AngularJS, které umožňují např. vypisování dat v cyklech atd. Grafika některých prvků stránky je změněna pomocí CSS. Svou roli zde našel rovněž Bootstrap, díky kterému bylo možné stránku upravit tak, že se dokáže přizpůsobit také změně velikosti okna a je možné ji zobrazit i na tabletech či mobilních zařízeních. Tabulky v těchto případech samy mění velikost a řadí se pod sebe. Plátno se takto taktéž přizpůsobí a je umístěno pod obě tabulky. Zde je příklad, jak se změní vykreslení tabulky *Statistics*.



Obrázek 4.9: Tabulka na webových stránkách při změně velikost okna

Jelikož se v této tabulce zobrazuje více záznamů, je zde umístěn posuvník umožňující rolování mezi záznamy. Hlavička však zůstává stále fixní.

Aby byly stránky dostupné i při výpadku internetu, jsou k nim přidány všechny potřebné soubory, které jsou nutné k používání některých prvků. Jedná se o soubory Jquery, Bootstrapu a AngularJS.

5 Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat program, který dokáže sledovat pohyb psa na základě dat ze zařízení SensorTag připojeného k obojku psa. K tomu vytvořit vhodné grafické rozhraní, kde bude možné pohyb sledovat. I přes nutnost několikrát předělávat celý návrh se výsledek práce v zásadě povedl.

V jednotlivých kapitolách jsou detailně popsány jednotlivé části této práce. Současný stav (2) poukazuje na možnosti dnešní techniky v oblasti hlídání/cvičení domácích mazlíčků. Jsou zde vyzdvihnuty nejpoužívanější technologie a popsány jejich pro a proti. Následuje popsání samotného SensorTagu v kapitole Použitá zařízení (3). Uvedeny jsou také možná rozšíření tohoto zařízení, díky kterým jej lze použít pro další různé aplikace. Poté už následuje kapitola věnující se samotné implementaci (4), která detailněji popisuje jednotlivé části celého programu, ale také postupy výpočtů hodnot, jejich ukládání a také konečná prezentace dat na webových stránkách.

Jak jsem však zmínil výše, jsou zde jistá omezení ve výpočtu rychlosti se vzdáleností a poté směru pohybu. Problém intervalu posílání by v případě dalšího pokračování v této práci bylo možné odstranit přímo ve zdrojovém kódu mobilní aplikace. Další možností je použití nové verze SensorTagu, která již bude umět používat Wi-Fi přímo. Odpadla by tak nutnost používat mobilní aplikaci. Data by chodila napřímo ze zařízení přímo do PC. Bohužel tato verze ještě není dostupná a bude na trhu až v průběhu druhé poloviny tohoto roku. I tak však výsledné hodnoty byly uspokojivé. V případě výpočtu směru není tolik možností, jak výsledné hodnoty zpřesnit. Našel jsem pouze pokročilé optimalizační algoritmy pro odstranění magnetismu z přístrojů a elektrického vedení.

Do budoucna se nabízí několik možností rozšíření tohoto projektu. Nejdůležitější je však odstranění neduhu s časovými razítky zprávy. Beze změny zdrojového kódu mobilní aplikace to však nebude úplně možné. Nabízí se spíše řešení s použitím originálních přijímačů Bluetooth/Zigbee. Takto by bylo možné upravit firmware zařízení do požadovaného nastavení. Odstranilo by se také omezení aplikace, kdy musíme s mobilním telefonem být stále nablízku zvířete. Takto by pohyb byl sledován i v lidské nepřítomnosti. Nabízí se také vytvoření mobilní aplikace, která by mohla uživatele automaticky upozorňovat na nežádoucí pohyb psa (např. maximální vzdálenost od přijímače) bez nutnosti ručně kontrolovat webové stránky. Jelikož zařízení disponuje vnitřní pamětí, bylo by také možné data při nemožnosti jejich přenosu ukládat a odeslat je, až to bude možné. Využití by bylo zejména pro statistiky či zpětný pohled na pohyb zvířete. Mobilní aplikace by však nejspíše toto rozšíření neumožnila, neboť data posílá ve stále stejném intervalu. Je tak otázkou, která data by se odesílala po připojení, zda ta uložená nebo z reálného času. Proto by bylo vhodnější rozšíření aplikovat až v případě přímého propojení zařízení s počítačem.

Seznam obrázků

Seznam tabulek

Literatura

[1] Texas Instruments. *IoT made easy* [online]. 2013 [cit. 2016-02-01]. Dostupné z: http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/?INTC=SensorTag&HQS=sensortag

[2] *Calculate speed from accelerometer* [online]. 2014 [cit. 2016-03-20]. Dostupné z: http://physics.stackexchange.com/questions/153159/calculate-speed-from-accelerometer

[3] Ozyagcilar, Talat. *Implementing a Tilt-Compensated eCompass using Accelerometer and Magnetometer Sensors* [online]. 2015 [cit. 2016-03-29]. Dostupné z: http://cache.freescale.com/files/sensors/doc/app_note/AN4248.pdf

[4] *Vzorce pro trojúhelník, jak najít stranu, osu, těžnici, výšku, úhel..* [online]. 2012 [cit. 2016-04-13]. Dostupné z: http://vzorce-matematika.sweb.cz/vzorce-pro-trojuhelnik-jak-najit-stranu-osu-teznici-vysku-uhel.php

[5] Texas Instruments. *CC2650 SensorTag User's Guide* [online]. 2016 [cit. 2016-04-13]. Dostupné z: http://processors.wiki.ti.com/index.php/CC2650_SensorTag_User's_Guide

[6] Haverinen, H., Salowey, J. *Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)* [online]. 2006 [cit. 2016-03-10]. Dostupné z: https://www.ietf.org/rfc/rfc4186.txt

[7] Bluetooth ©. *Bluetooth* [online]. 2016 [cit. 2016-03-02]. Dostupné z: https://www.bluetooth.com/

[8] Zigbee Alliance. *Zigbee* [online]. 2016 [cit. 2016-02-14]. Dostupné z: http://www.zigbee.org/

[9] Pearl, Ori. *Mosquitto* [online]. 2016 [cit. 2016-02-25]. Dostupné z: http://mosquitto.org/

[10] Texas Instruments. *Teardown* [online]. 2015 [cit. 2016-01-14]. Dostupné z: http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/tearDown.html

[11] Townsend, Kevin. *GATT* [online]. 2015 [cit. 2016-02-04]. Dostupné z: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt

[12] Texas Instruments. *IoT made easy* [online]. 2013 [cit. 2016-02-01]. Dostupné z: http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/devPacks.html

[13] Numaxes. *IKI Voice* [online]. 2015 [cit. 2016-01-14]. Dostupné z: http://www.numaxes.com/en/bark-control/163-canicalm-sonic.html

[14] SportDog. *TEK SERIES 2.0* [online]. 2016 [cit. 2016-01-15]. Dostupné z: http://www.sportdogglobal.com/tek2.php

[15] Kippy. *The GPS tracker for cats and dogs* [online]. 2016 [cit. 2016-01-18]. Dostupné z: https://www.kippy.eu/en-gb/gps-dog-cat-tracker-pets-pets-tracker

[16] Jim Lucas. *What Are Radio Waves?* [online]. 2015 [cit. 2016-01-17]. Dostupné z: http://www.livescience.com/50399-radio-waves.html

Příloha A

Obsah CD

- /src/ - adresář se zdrojovými kódy aplikace
- README.txt – návod k překladu a spuštění aplikace
- /text/ – adresář s technickými zprávami ve formátu docx a pdf