Slovenská technická univerzita
Fakulta informatiky a informačných technológii
Ilkovičova 2, 812 19 Bratislava

Tímový projekt – Stratosférický balón

Dokumentácia k inžinierskemu dielu

Tím 10

Cvičiaci/Vedúci: Ing. Michal Valiček.

Akademický rok: 2016/2017

Autori: Bc. Tomáš Urban
Bc. Martin Oravský
Bc. Márius Rak
Bc. Jakub Findura
Bc. Maroš Frkáň
Bc. Ján Pánis
Bc. Dominik Pisarovič

Obsah

Zoznam obrázkov

Zoznam tabuliek

1 Úvod

Oblasť stratosférických letov je mimoriadne zaujímavá a sama o sebe značne špecifická. Táto práca obsahuje dokumentáciu k projektu, ktorým by sme chceli nadviazať na úspešné vypustenie stratosférického balóna v máji 2016. V nasledujúcich častiach popisujeme ciele, ktoré plánujeme dosiahnuť v rámci vývoja nášho servisného modulu, ktorý je hlavnou časťou našej práce. Ďalej popisujeme históriu vypúšťania balónov do stratosféry, ich využitie a následne aj náš návrh riešenia. Tento návrh opisuje všetky moduly nášho systému, od analýzy, cez návrh až po implementáciu. Taktiež popisujeme nami zvolený spôsob propagácie nášho projektu, ako aj jeho výsledkov.

1.1 Ciele projektu

Cieľom nášho projektu je plne funkčný a spoľahlivý servisný modul umožňujúci pravidelné vypúšťanie balónov s experimentami rôznych projektov našej fakulty. Tento servisný modul je špeciálne zariadenie, ktoré pracuje v špecifických podmienkach ako je radiačné pozadie a extrémne zmeny teplôt. Tiež jeho neoddeliteľnou súčasťou je schopnosť zasielať informácie o svojej polohe za účelom úspešného dohľadania balóna po dopade naspäť na zem.

1.2 Globálne ciele pre zimný semester

Počas zimného semestra máme v pláne zamerať sa v prvom rade na kvalitnú analýzu problému, z ktorej následne budeme vychádzať. Správne pochopenie problému je nesmierne dôležité pre dobré nasmerovanie projektu. Po ukončení analýzy sa zameriame na prípravu hardvéru a jeho vzájomnú kompatibilitu a overenie požadovanej funkčnosti. Popri vývoji softvéru a jeho testovaní, by sme sa tiež chceli venovať špecifikácií metodík, pre efektívny a plynulý kolaboratívny postup v našej práci. Spolu s navrhnutím architektúry sa plánujeme taktiež venovať implementácii prototypu s obmedzenou funkcionalitou. Prototyp nášho servisného modulu síce nebude spĺňať všetky požiadavky umožňujúce let do stratosféry, avšak zabezpečia základ pre zachytávanie a preposielanie dát a tiež propagáciu s využitím základných informácii z modulu.

1.3 Stratosférický balón

Stratosférické balóny sú balóny vypúšťané do atmosféry za rôznym účelom. Ich cieľom je dosiahnuť stratosféry, vrstvy zemskej atmosféry, ktorá sa nachádza vo výške približne 10 až 50 km nad zemským povrchom. Pod balónom je upevnený náklad, ktorý je potrebné vyniesť

balónom do stratosféry. Väčšinou sa jedná a meteorologické prístroje, vedecké experimenty, rôzne senzory či kamery alebo fotoaparáty.

Balón neobsahuje žiadny aktívny pohon. Jeho jedinou hybnou silou je nižšia hustota plynu vo vnútri balónu oproti okolitej atmosfére. Vďaka tomu sa balón pôsobením vztlakovej sily dokáže vznášať a vystúpiť do požadovanej výšky. Pretože s narastajúcou výškou klesá tlak atmosféry, plyn v balóne sa rozpína, čím sa zväčšuje priemer balóna. Keď je vnútorný tlak väčší ako pružnosť balónu, dôjde k jeho roztrhnutiu, často označovanému anglickým výrazom „burst“. Po roztrhnutí nasleduje voľný pád nákladu, ku ktorému je pripevnený menší padák, ktorý spomalí finálny pád a riziko poškodenia nákladu pri dopade.

Na to aby bol úspešne splnený cieľ vypustenia balónu, musí celá konštrukcia obsahovať niekoľko komponentov. Hlavným je samozrejme balón naplnený potrebným plynom. Ďalšou časťou je náklad, ktorý je v ochrannom obale, ktorý ho izoluje od nepriaznivých podmienok vo veľkých výškach atmosféry. Obsahom nákladu sú prístroje a experimenty, ktoré je potrebné vyniesť a obslužná elektronika, ktorá sa stará o lokalizáciu balóna a odosielanie jeho polohy aby bolo možné nájsť náklad po dopade.

1.3.1 Náklad

Pod pojmom náklad môžeme rozumieť dve veci. V prvom prípade sa jedná o všetko zariadenie vrátane ochranného obalu, ktoré je pod balónom zavesené. V užšom význame môžeme pod nákladom chápať zariadenia uložené v priestore, ktorý poskytneme iným v našom balóne.

Zásadným obmedzením je hmotnosť, keďže balóny majú obmedzené množstvo hmotnosti, ktorú dokážu vyniesť do atmosféry. Hlavným faktorom ovplyvňujúcim hmotnosť nákladu je veľkosť balóna. Čím je balón väčší, tým má väčší vztlak a dokáže vyniesť väčšiu hmotnosť. Hmotnosť nákladu tiež ovplyvňuje parametre samotného letu, konkrétne výšku stúpania a nadmorskú výšku, pri ktorej sa balón roztrhne. Preto je prioritou čo najviac minimalizovať hmotnosť nákladu, čím urýchlime vzostup a zvýšime maximálnu dosiahnuteľnú nadmorskú výšku a použitie menšieho balóna zníži náklady na jeho vypustenie.

Celý náklad je uložený v nádobe z polystyrénu, ktorá poskytuje hlavne ochranu pred poveternostnými vplyvmi a tepelnú ochranu elektroniky, pričom môže byť obalená v ďalších materiáloch zabezpečujúcich ochranu pred UV žiarením alebo radiáciou. Vo vnútri sa nachádzajú obslužné zariadenia pre riadenie letu a zariadenia alebo predmety potrebné pre splnenie účelu letu balóna.

Primárnym účelom servisného modulu je lokalizácia balóna a odosielanie jeho polohy riadiacemu stredisku pre monitorovanie pohybu a jeho vyzdvihnutie po dopade na zem. Servisný modul obsahuje riadiacu jednotku, GPS modul na určenie polohy, polohové senzory ako akcelerometer, kompas či gyroskop a rôzne ďalšie senzory pre meranie teploty, tlaku alebo vlhkosti. Odosielanie telemetrie riadiacemu stredisku je zabezpečené cez rádiové vlny alebo cez GSM siete pri nižších výškach v dosahu pozemnej infraštruktúry operátora mobilných sietí. Nezanedbateľnou časťou aj z pohľadu hmotnosti je zdroj elektrickej energie v podobe batérie prípadne v kombinácii so solárnymi článkami.

Vynášaný náklad môže byť v princípe čokoľvek, čo je v rámci rozmerových a hmotnostných limitov. Najčastejšie to bývajú meteorologické prístroje, vedecké experimenty, fotoaparáty a iné zariadenia.

1.3.2 Legislatíva a bezpečnosť

Pred vypustením balóna by sme sa mali zamyslieť nad bezpečnosťou a rizikami spojenými s vypustením balóna do voľného priestoru. Pretože balón nie je možné v drvivej väčšine riadiť pri vzostupe ani pri páde, môže potenciálne ohroziť majetok a zdravie iných ľudí.

Počas manipulácie je potrebné si dávať pozor na citlivú elektroniku, ktorá by sa mohla poškodiť. Pokiaľ bude balón naplnený vodíkom, je obzvlášť potrebná vysoká obozretnosť kvôli vysokej výbušnosti.

Pred vypustením je potrebné sa oboznámiť s legislatívou, ktorá sa zaoberá letovou prevádzkou a v prípade potreby kontaktovať dané úrady. Všetky dopravné prostriedky v letovom priestore musia byť oboznámené s prítomnosťou balóna aby nedošlo ku kolízii prípadne vážnejšiemu nešťastiu. Pri dopade je potrebné myslieť na miesto dopadu, tak aby nedošlo k poškodeniu majetku alebo poraneniu iných osôb. K tomu slúžia rôzne softvérové nástroje na výpočet predpokladanej trasy letu a čiastočnému naplánovaniu miesta dopadu.

V rámci Slovenskej republiky vypúšťanie balónov, resp. akýchkoľvek predmetov, do atmosféry reguluje zákon č. 143/1998 Zb., Zákon o civilnom letectve. Zákon nehovorí priamo o balónoch, no riadi a obmedzuje využívanie vzdušného priestoru Slovenskej republiky s ohľadom na bezpečnosť. Relevantné sú pre nás hlavne §5 až §8. Okrem toho je vydávaná Letecká informačná príručka (AIP) [1] odborom manažérstva leteckých informácií (AIM), ktorý patrí pod štátny podnik Letové prevádzkové služby Slovenskej republiky. Táto príručka je určená hlavne pre riadenie leteckej dopravy nad územím Slovenskej republiky, avšak určuje aj pravidlá

pre vypúšťanie stratosférických balónov. Okrem tejto príručky vydáva aj rôzne civilné predpisy týkajúce sa okrem iného aj vypúšťania balónov.

Stratosférickým balónom sa venuje predpis **L2 Pravidlá lietania** vychádzajúci z Dohovoru o medzinárodnom civilnom letectve, konkrétne Oddiel 3, kapitola 1,

SERA.3140 Neobsadené voľné balóny

„Lety neobsadených voľných balónov sa vykonávajú spôsobom, ktorým sa minimalizuje nebezpečenstvo pre osoby, majetok alebo pre ostatné lietadlá a ktorý je v súlade s podmienkami stanovenými v dodatku 2.“ [2]

Ďalej je neobsadeným voľným balónom, ich klasifikácii a pravidlám venovaný Dodatok 4 predpisu L2. [3]V rámci Európskej únie je tento predpis s miernymi zmenami prijatý v nariadení č. 923/2012. Neobsadeným voľným balónom je venovaný dodatok 2. Z predpisov okrem iného vyplýva, že na prevádzkovanie stratosférických balónov je potrebné povolenie leteckého úradu.

1.4 História vypúšťania experimentálnych balónov

Vo francúzsku v roku 1783 sa prvý krát verejne vypustil balón naplnený vodíkom pred zrakmi 300 000 ľudí. Jacques Charles, francúzsky profesor fyziky spoločne s bratmi Robertovcami, ktorý pracovali ako renomovaní výrobcovia vybavenia určeného pre fyzikálne experimenty. Balón s názvom Charlière, ktorý dosahoval veľkých rozmerov bol napúštaný vodíkom po dobu 5 dní. Štartovacím miestom bol verejný park Champ de Mars, ktorý sa nachádza neďaleko Eiffelovej veže. Expanzia plynu v balóne spôsobila jeho roztrhnutie po 45 minútach od štartu a balón sa dostal do výšky 20km nad Paríž.

Ďalším z významných použití, ktoré boli zdokumentované bol meteorologický balón, ktorý zaznamenal francúzsky meteorológ Leon TEISSERENC de Bort. Aktívne začal vypúšťať meteorologickej balóny už v roku 1896. Práca tohto meteorológa významne pomohla pri objavení a najmä bližšom preskúmaní troposféry a stratosféry. Jedná sa o jedinečné vrstvy v zemskej atmosfére, ktoré už v dnešnej dobe môže skúmať hocikto z nás práve vypustením vlastných meteorologických balónov. Vzhľadom na to, že práca vedca Leon Teisserenc de Bort bola tak inštrumentálna, bol poctený tým, že bol po ňom pomenovaný kráter na Mesiaci a rovnako aj na Marse.

Začiatkom 20. storočia, meteorológ a geofyzik menom Alfreda Wegener použitím meteorologického balónu vykonal experimenty, ktoré ho viedli k objaveniu významnej teórie pohybu kontinentov, niekedy aj často označovanou ako kontinentálny drift. Práca, ktorá skúmala túto teóriu bola zverejnená už v roku 1912. Táto teória sa stretla s veľkým odporom a

akceptovaná bola až v roku 1960, čo je viac ako 30 rokov po jeho smrti. Taktiež, podobne ako de Bort bol poctený tým, že na Mesiaci a na Marse boli po ňom pomenované dva krátery.

Ďalším dôležitým vedcom bol americký vedec James Van Allen, ktorý v 50. rokoch 20. storočia vykonával rôzne balónové experimenty. Tieto experimenty viedli k objaveniu radiačných pásov v okolí Zeme. Tieto pásy niekedy označujeme aj ako Van Allenove radiačné pásy. Časopis Time ocenil Van Allena ako muža roka „Man of the Year" v roku 1960 práve kvôli jeho vedeckým zásluhám.

Obrázok 1.1 Študentský amatérsky stratosférický balón vypustený v americkej púšti Mojave.

1.5 Súčasné prístupy

1.5.1 NASA

Veľké bezpilotné balóny, plnené najmä héliom, poskytujú americkej vládnej agentúre NASA nenahraditeľné možnosti ako vyniesť rôzne vybavenie do kozmického priestoru. Unikátne vlastnosti takýchto programov sú kľúčové pre vývoj nových technológií. Mnoho dôležitých objavení a technológií ale aj náklady (angl. payload) pre sofistikovanejšie vesmírne lety boli práve výsledkom experimentov v ktorých boli podklady získané zo stratosférických balónov. Konkrétne sú to napr. pozorovania Röntgenového a gama žiarenia, kozmického žiarenia alebo podklady pre odbor infračervenej astronómie. Novo vyvinuté prístupy sú zamerané na tenko vrstvové stratosférické balóny určené pre dosiahnutie vysokej nadmorskej výšky a ich dlhšie pôsobenie v atmosfére.

V roku 2009 boli vykonané viaceré dlhotrvajúce experimenty, ktoré boli prevádzkované v Antarktíde počas letného obdobia na južnej pologuli. Tieto experimenty boli výsledkom spolupráce organizácie NASA (National Aeronautics and Space Administration) a organizácie

NSF (National Science Foundation). NSF zabezpečila komunikáciu a logistiku a NASA zabezpečila satelitnú komunikáciu. Unikátne vlastnosti atmosféry nad Antarktídou umožňujú vedcom vypustiť balón z vedeckej stanice McMurdo a následne ho z takmer rovnakého miesta obnoviť aj po uplynutí niekoľkých týždňov. Balóny vypustené v Antarktíde sú dlhotrvajúce, kvôli javu nazývanému polárny vír (angl. polar vortex). Tento vír spôsobuje stále, veľké nízkotlakové správanie v atmosfére, najmä kvôli nízkym teplotám. Konštantné denné svetlo v Antarktíde znamená, že sa teplota prechodom zo dňa na noc nemení. Tento fakt pomáha balónu zostať na v rovnakej výške po dlhý časový interval.

Obrázok 1.2 Balón vypustený kanadskou vesmírnou agentúrou z roku 2011.

1.5.2 Balón Universum

Popri svojich študentských aktivít sa na Žilinskej univerzite jeden zo študentov Ondrej Závodský so svojím tímom vypustil 25.9.2010 jeden z prvých stratosférických balónov na Slovensku. Vlastný balón si zakúpil v Prahe, následne ho nafúkali tak aby bol schopný nadvihnúť 2l fľašu s vodou, pripevnili modul a vypustili. Pri prvom vypustení sa stretli s množstvom komplikácií. Prvý vysielač odišiel z dôvodu odtrhnutia antény už pri vzlietnutí. Druhý GPS modul zamrzol pri páde 9km nad zemou a preposielal rovnaké info, ktoré po dopade prestalo vysielať z dôvodu uvoľnenia batérií [4].

Celkový let im trval 2 hodiny a 20 minút, z čoho 1 hodina 38 bol čas stúpania a 42 minút čas klesania. Balón dokopy prešiel vzdialenosť 85 km s maximálnou dosiahnutou

rýchlosťou 90km/h. Maximálna dosiahnutá výška bola 25 946m a minimálna teplota dosiahla -33,29°C.

V ďalších riešeniach sa autor chystá zamerať na rôzne otázky ako zabránenie rotácie buď gyroskopom alebo pomocnými krídelkami za účelom stabilizácie fotoaparátu, vyriešiť riziko pádu balóna do vody, vysielanie súradníc pomocou APRS, prípadne solárny panel alebo riadený pád pomocou kĺzavého padáku. [5]

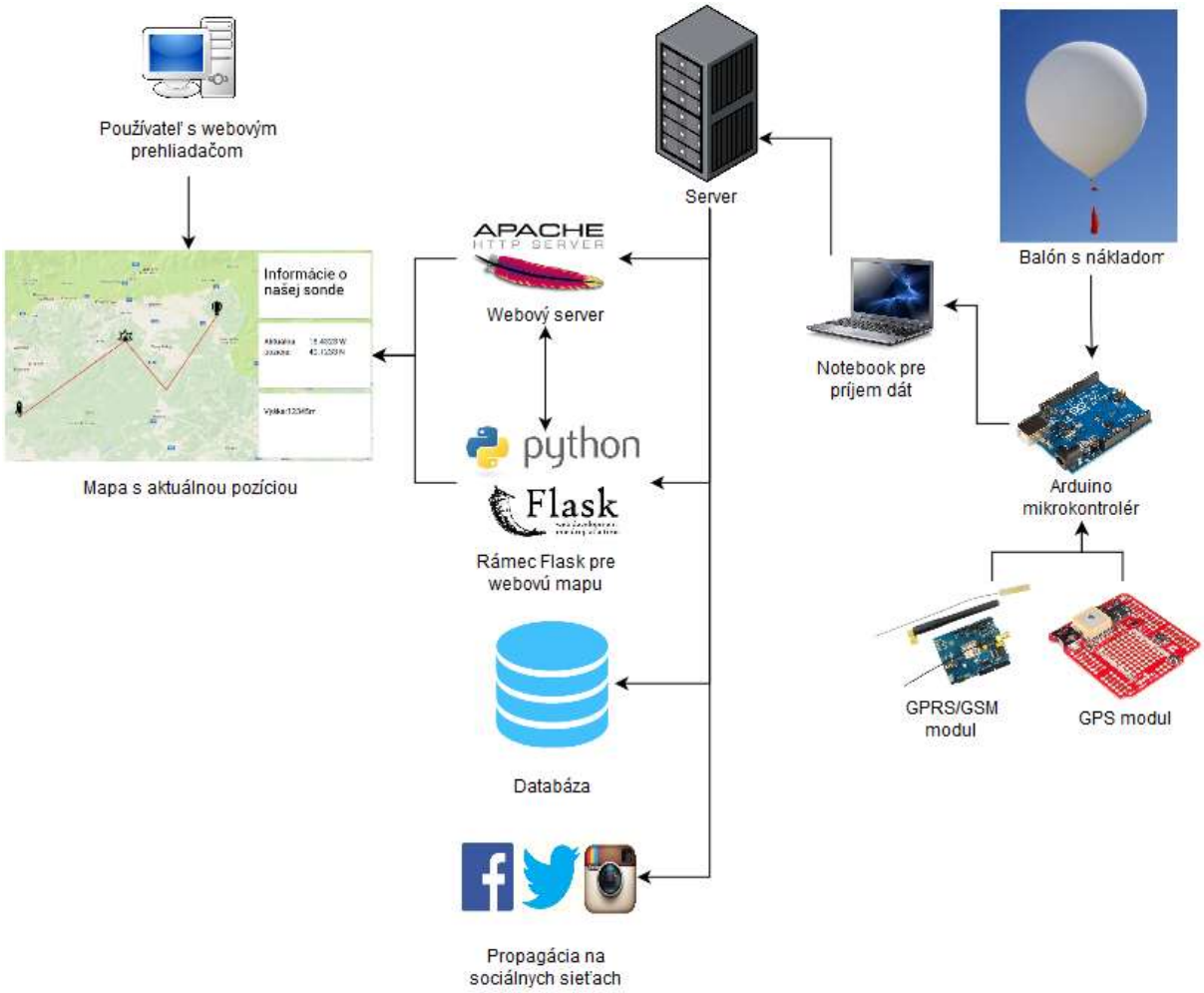
Z jeho riešenia sme sa v mnohom poučili a jeho spísané skúsenosti využili pri implementácii nášho riešenia.

1.5.3 Lietajúci experiment BU60-1

Samotný BU60-1 balón vážil 34,37kg a dosahoval dĺžku 74,5m a 53,7m po obvode. Celková váha balónu pred štartom dosahovala 39,77kg s padákom vážiacim 0,8kg spoločne s pozorovacími prístrojmi, ktorých váha zaťažila balón o 4,6kg. Prístrojmi, ktoré zaznamenávali servisné informácie o experimente boli: dve ITV kamery pre tvorbu fotiek monitorujúce nafúknutie balóna a GPS modul pre presné meranie výšky balónu.

Balón bol vypustený v ranných hodinách 23. mája 2002 o 6:35. Dosahované rýchlosť stúpania balónu dosahovala 260m za minútu a dosiahla doteraz najvyššiu zaznamenanú výšku a to 53km o necelé 4 hodiny v 10:07. Tento úspech pokoril americký rekord z 1972, ktorý dosiahol hranicu 51,8km.

2 Celkový pohľad na systém



Obrázok 3 Celkový model projektu

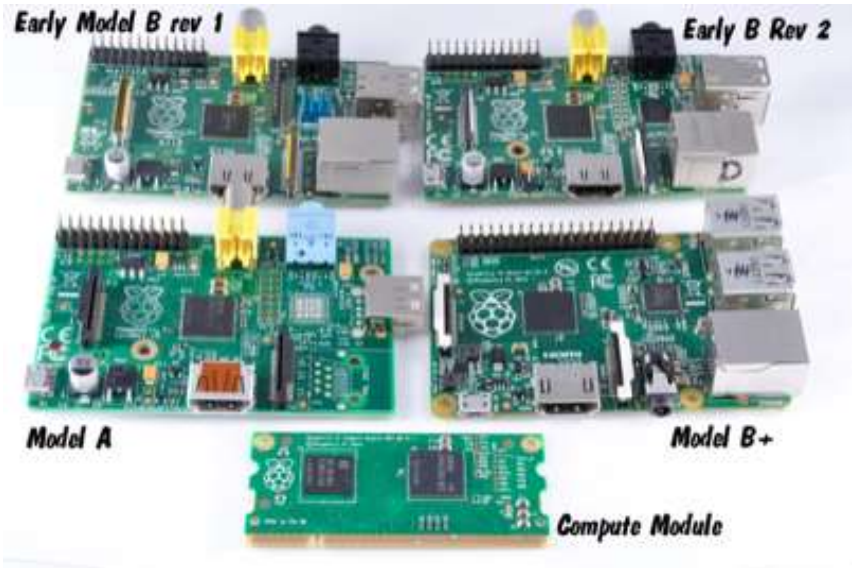
3 Moduly systému a integrácia

3.1 Platformy a komponenty

3.1.1 Platforma Raspberry Pi

Raspberry Pi je miniatúrny jednodoskový počítač vyvíjaný Anglickou spoločnosťou Raspberry Pi Foundation. Cieľom tvorcov Raspberry Pi je vývoj a predaj relatívne lacných jednodoskových počítačov s výkonom postačujúcim na fungovanie základných funkcií Linuxových operačných systémov.

Počítače Raspberry Pi používajú procesory s ARM architektúrou. V prípade prvej generácie ide o ARM1176JZF-S s taktovacou frekvenciou 700MHz. Aktuálna generácia obsahuje výkonnejší procesor ARM CortexA-7 s frekvenciou 900MHz.



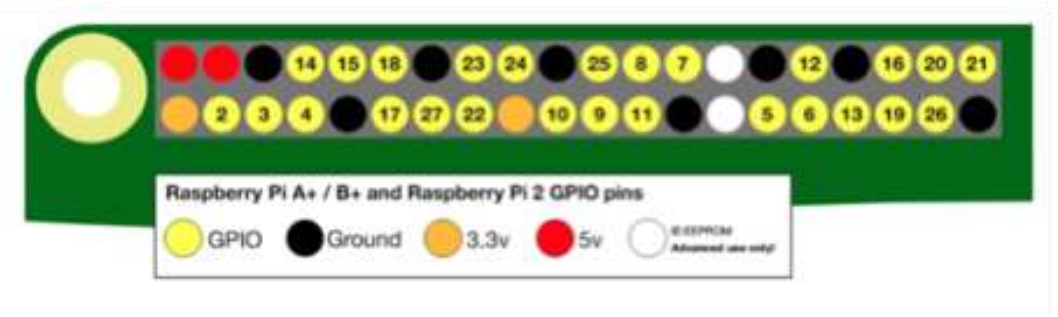
Obrázok 3.1Raspberry Pi

Na pripojenie rôznych štandardných aj neštandardných periférií slúži sériová linka GPIO. GPIO predstavuje 40 pinové vstupno-výstupné rozhranie. Z počtu 40 pinov 26 slúži ako GPIO a ostatné ako napájacie a uzemňovacie piny. Dva piny slúžia na prístup k EEPROM pamäti.

Pri použití GPIO pinu ako výstupu ho dokážeme softvérovo ovládať. Dokážeme ovládať hodnotu napätia na výstupnom pine. Pri zapnutom stave je táto hodnota 3,3V a pri vypnutom 0V.

Pri použití GPIO pinu ako vstup na ňom dokážeme detegovať úroveň el. napätia. Pri úrovni 0V Raspberry Pi deteguje vstup ako vypnutý a pri 3,3V ako zapnutý. Medzi týmito

hodnotami Raspberry Pi deteguje tzv. „floating“ stav, kedy nie je možné presne určiť stav pinu. Riešením je používanie tzv. pull-up a pull-down rezistorov integrovaných na doske.



Obrázok 3.2Rozloženie pinov na Raspberry Pi

Raspberry Pi je možné použiť v kombinácií s niekoľkými operačnými systémami. Oficiálne sú podporované tri linuxové distribúcie: Pidora(založené na distribúcii Fedora), Archlinux a Raspbian(Debian). Použitím týchto operačných systémov získavame plnohodnotný linuxový počítač.

Vďaka jeho univerzálnosti v kombinácii s nejakou z linuxových distribúcií má Raspberry Pi dôležité postavenie aj v odvetví IoT(Internet of Things). Ide však najmä o odvetvie rôznych domácich projektov a nekomerčných riešení.

Jedným z vhodných programovacích jazykov pre vývoj programov pre Raspberry Pi je nepochybne C/C++. Vhodných prostredí na vývoj je hneď niekoľko, no medzi najpoužívanejšie patria NetBeans, Eclipse, XCode a podobne. Po väčšine závisí od osobných preferencií konkrétneho používateľa.

Raspberry Pi je napájané 5V jednosmerným napätím. Výrobca na svojich stránkach odporúča zariadenie napájať zdrojom s kapacitou aspoň 1,8A v prípade druhej generácie zariadenia a 2.5A v prípade tretej. V prípade čistej dosky zariadenia, bez akýchkoľvek periférií, výrobca udáva spotrebu 330-400mA. Konkrétna energetická náročnosť však závisí od celého systému a na počte a type použitých periférií.

3.1.2 Prototypovacia platforma Arduino

Raspberry Pi nemá vstavaný A/D prevodník, preto sú na tento účel často používané iné mikrokontroléry ako napr. Arduino resp. kombinácia s nimi.

Arduino je open-source platforma pre vývoj rôznych prototypov a interaktívnych zapojení. Arduino dosky sú schopné čítať vstupy, vyhodnocovať akcie a udávať výstupy [6]. Platforma je

založená na programovateľnom mikrokontroléri ATMega od spoločnosti Atmel a na trhu je dostupných niekoľko už skompletizovaných verzií.

3.1.2.1 Arduino Uno

Arduino Uno je jeden zo základných modelov tejto platformy a prvý z modelov s USB pripojením. Je založený na čipe ATMega328P.

Obsahuje:

- 14 digitálnych vstupno-výstupných pinov (6 z nich je možné použiť ako PWM [7] výstupy),
- 6 analógových vstupných pinov,
- 16MHz oscilátor,
- USB pripojenie,
- napájací konektor(7-12VDC),
- resetovacie tlačidlo.

Na vytvorenie IP spojenia s inými zariadeniami je možné použiť Arduino Ethernet Shield. Na spoluprácu Arduina a Ethernet Shieldu je potrebné použiť vhodnú knižnicu. V prípade takýchto spojení dostávame veľmi silný nástroj na implementáciu aj zložitejších algoritmov použiteľných pri rôznych druhoch prototypov.

Na vytváranie programov pre Arduino je dostupný vývojové prostredie s rovnakým názvom. Programuje sa v jazyku založenom na C/C++. Pre programovanie sa využíva konvertor z USB na seriálový port [8].

Arduino Uno je napájané 5V jednosmerným napätím. Odporúčané napätie je však 7-12V. Arduino má po zapnutí len približne 20-25mA. Vďaka takejto malej spotrebe sa Arduino stáva jedným z vhodných kandidátov na použite pre účely servisného modulu. Konkrétna energetická náročnosť však závisí od celého systému a na počte a type použitých periférií pripojených k Arduinu.

Sumarizačná tabuľka:

Mikrokontrolér	ATmega328
Operačné napätie	5 V

Vstupné napätie (doporučené)	7 - 12 V
Vstupné napätie (limity)	6 - 20 V
Digitálne V/V kolíky	14
Analógové vstupné kolíky	6
Prúd pre 5 V kolíky	40 mA
Prúd pre 3,3 V kolíky	50 mA
Flash pamäť	32 KB (0,5 KB je použitá pre program boot-ovania)
SRAM pamäť	2 KB
EEPROM pamäť	1 KB
Frekvencia kryštálu	16 MHz

Tabuľka 1 Sumarizačná tabuľka mikrokontroléra Arduino Uno [8].

Obrázok 3.3Mikrokontrolér Arduino Uno [8]

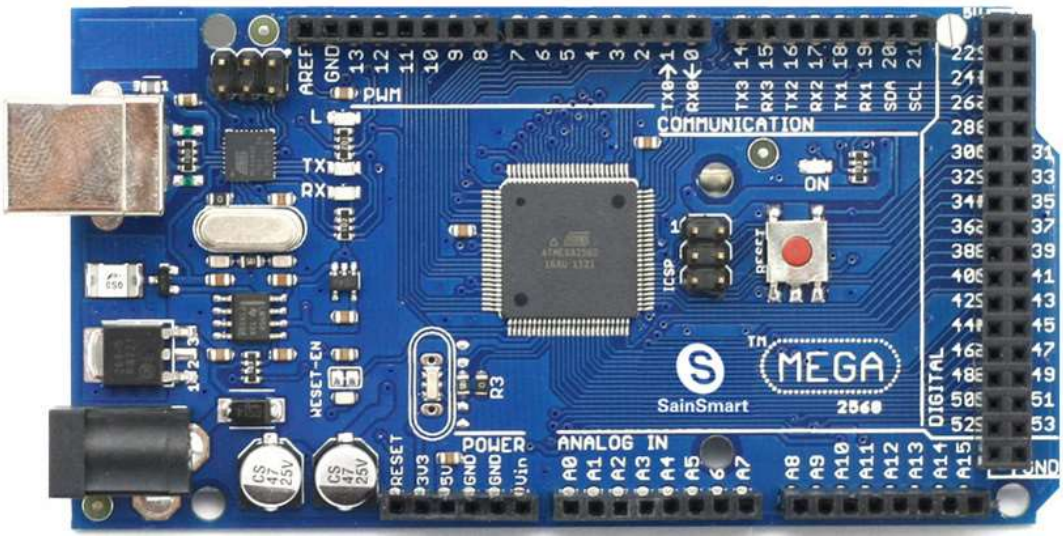
3.1.2.2 Arduino MEGA

Arduino Mega 2560 je mikrokontrolér založený na čipe ATmega2560. Má 54 digitálnych vstupno/výstupných kolíkov (z ktorých 14 môže byť využitých ako PWM výstupy), 16 analógových vstupov a 4 UART. 16MHz keramický kryštál, USB pripojenie, napájací vstup a resetovacie tlačidlo. Obsahuje všetko, čo musí mikrokontrolér obsahovať. Napájať ho môžeme pomocou USB kábla, adaptéru ale aj batérie. Pre programovanie sa využíva konvertor z USB na seriálový port [9].

Mega je väčšia prototypovacia platforma, oproti klasickému Arduinu UNU, avšak je spätne kompatibilné so štítmi z menších mikrokontrolérov ako sú: UNO, Leonardo, Duemilanovo, Diecimala a iné...

Tabuľka 2 Sumarizačná tabuľka mikrokontroléra Arduino Mega 2560 [9]

Mikrokontrolér	ATmega2560
Operačné napätie	5 V
Vstupné napätie (doporučené)	7 - 9 V
Vstupné napätie (limity)	6 - 20 V
Digitálne V/V kolíky	54
Analógové vstupné kolíky	16
Prúd pre 5 V kolíky	40 mA
Prúd pre 3,3 V kolíky	50 mA
Flash pamäť	256 KB (8 KB je použitá pre program boot-ovania)
SRAM pamäť	8 KB
EEPROM pamäť	4 KB
Frekvencia kryštálu	16 MHz



Obrázok 3.4 Mikrokontrolér Arduino Mega 2560 [9]

3.1.3 Intel Galileo

Doska Intel Galileo Gen 2, momentálne najvýkonnejšia dosky so zbernicou Arduino. Srdcom vývojového kitu je výkonný 32-bitový procesor, presnejšie SoC (System on Chip), Intel Quark X1000a, ktorého inštrukčná súprava je kompatibilná s Pentiom. K dispozícii má 256 MB DDR3, 8 MB NOR flash a 8 kb EEPROM.

Popri štandardných digitálnych aj analógových vstupoch a výstupoch je k dispozícii aj slot mini-PCI Express, ktorý je možné použiť pridanie modulu Wi-Fi, 100Mbit ethernetový port, slot na kartu microSD alebo port. Na porovnanie, doska Arduino Ethernet má procesor ATmega328 16 MHz, 1 kb EEPROM, 2,5 kB RAM a 32 KB flash. Ako operačný systém možno využiť Linux (zostava Yocto 1.4 Poky), prípadne upravenú verziu Windows 8.1, ktorá je dostupná na stránke Microsoftu venovanej IoT. Windows 10 IoT Core pre túto dosku k dispozícii nie je.

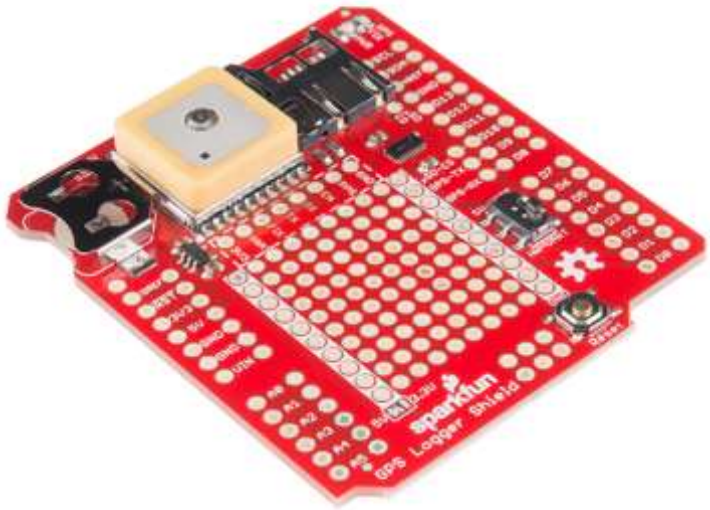
Vysoký výkon Galilea je pochopiteľne vykúpený nepomerne vyššou spotrebou, takže tento kit sa hodí tam, kde je dostatok energie. V prípade použitia na účel sondy ktorá je súčasťou stratosférického balóna sme limitovaný hmotnosťou, a tým pádom aj kapacitou zdroja energie – batériou. V prípade neobmedzenej hmotnosti by bola táto platforma vďaka svojmu výkonu určite vhodnou voľbou.

3.1.4 SparkFun GPS zaznamenávací štít

Tento štít umožňuje Arduinu extra funkcionality. Sú to zaznamenávanie GPS súradníc, čo je pre nás kľúčovo dôležité, miesto pre microSD kartu (zaznamenávanie nameraných hodnôt na záložné médium), ale taktiež aj voľný priestor na usadenie ďalších senzorov alebo akčných členov [10].

Základom je čip GP3906-TLP (GPS prijímač), ktorého hlavnou výhodou je ultra nízka spotreba energie (okolo 20mA). Práve tento parameter je pre nás kľúčový. Taktiež jeho ďalšou nemenej podstatnou výhodou je práceschopnosť v rozmedzí teplôt od -30 do +85°C. Pracuje na 3,3V logike, avšak to nie je problém, keďže Arduino poskytuje aj napájaciu hodnotu napätia 3,3V [11].

Ďalšia výhoda tohto štítu je, že je priamo predpripravený pre prototypovaciu platformu Arduino UNO, ktorá je spätne kompatibilná aj s platformou Arduino MEGA. To je pre nás veľmi výhodné z dôvodu, že nebudeme musieť riešiť vývoj vlastných dosiek, do ktorých by bolo možné jednoducho osadiť daný, alebo podobný čip.



Obrázok 3.5SparkFun GPS zaznamenávací štít [10]

3.1.5 Meranie teploty

Meranie fyzikálnej veličiny teploty, nie je pre náš projekt až tak dôležité, avšak vďaka tomu vieme zistiť, aké sú okolité teploty v rozličných nadmorských výškach. Pomocou nameraných údajov, budeme sa v budúcnu vedieť lepšie pripraviť na okolité nežiadúce podmienky a využiť lepšie materiály, senzory alebo hardvér. Podľa analýzy sme zistili, že teplota v stratosfére väčšinou býva v rozmedzí od -60°C po 0°C, je tomu nutné prispôsobiť aj náš hardvér. Podľa funkcionality delíme teplotné senzory na:

1. **Teplomery so záporným tepelným súčiniteľom (NTC) – Termistory**: slovo termistor vychádza s kombinácie 2-ch slov (TERM-álne senzitívny rez-ISTOR). Je to špeciálny typ rezistora, ktorý predvídateľne mení svoj odpor na základe okolitej teploty. NTC termistor poskytuje vysokú odolnosť v nízkych teplotných podmienkach. Jeho efektívny pracovný rozsah je väčšinou od -50°C po 250°C (líši sa od typu).
2. **Odporový teplotný detektor (RTD)**: je tiež známy ako odporový teplomer. Meria teplotu na základe korelácií odporov medzi RTD prvkom a teplotou. RTD prvok je vyrobený s vysoko čistých, vodivých kovov ako sú: platina, meď alebo nikel. Ich hlavnou výhodou je, že ponúkajú pomerne presný lineárny výstup, ktorý je veľmi presný. Častokrát ich pracovný rozsah presahuje hranice -200°C až 600°C. Majú však jednu nevýhodu – cenu (sú extrémne drahé, v porovnaní s ostatnými).

3. **Termočlánok ako tepelný senzor**: tento typ tepelného senzoru sa skladá z dvoch rôznych kovov spojených v dvoch bodoch. Pomocou meniaceho sa napätia medzi týmito dvomi bodmi sa dajú vypočítať zmeny teploty. Tieto články sú nelineárne a na prevod teploty sa častokrát využíva tabuľková kompenzácia hodnôt. Pri tomto spôsobe merania teploty je úplne normálna odchýlka merania 5°C. Ich hlavnou výhodou je schopnosť merať najširšie pásmo teplôt od -200°C až po 2000°C a aj preto sa stále využívajú.
4. **Tepelné senzory na báze polovodičov**: tento teplotný senzor je umiestnený na integrovanom obvode. Tieto senzory sú založené na dvoch diódach s tepelnou senzitivitou. Pomocou ich volt-ampérových charakteristík je možné s nich odčítavať a vypočítať teplotu. Ich výstup je lineárny avšak presnosť nie je veľmi uspokojivá (častokrát majú odchýlky od 1°C až po 5°C). Majú najpomalšie reakcie na zmeny teplôt (5 až 60 sekúnd), ale na druhú stranu sa s nimi veľmi jednoducho pracuje a častokrát (keďže pracujú na integrovanom obvode) dokonca vieme z nich odčítať číselnú hodnotu teploty [12] [13].

Pri analyzovaní a testovaní rôznych typov senzorov sme začínali pri tých najjednoduchších až po tie zložitejšie.

Ako prvý sme využívali teplotný senzor na báze polovodičov DS18B20. Pre jeho pomalé reakcie a nie veľmi vysokú presnosť sme ho zamietli. Práca s ním bola síce veľmi jednoduchá, avšak namerané hodnoty nás nepresvedčili.

Ako druhý sme analyzovali o otestovali termistor KTY81-120 práca s ním bola so začiatku trochu komplikovaná, keďže sme museli zimplementovať funkciu, ktorá vedela jeho výstupy prevádzať do číselných hodnôt, avšak po nájdení podobnej funkcie a patričných úpravách je práca s ním veľmi jednoduchá, presná a spoľahlivá. Zatiaľ ho považujeme za najlepší tepelný senzor, ktorý máme, aj keď jeho pracovné rozpätie činí od -55°C po 150°C. Minimálne môže a bude použitý na zaznamenávanie teploty vnútri v izolovanej sústave (predpokladáme, že teplota dnu by nemala prekročiť hodnoty od -10°C po 20°C).

Aktuálne pracujeme na analýze a testovaní odporového teplotného detektora (RTD) Proffuse PT100-1020. Bol vybratý z dôvodu jeho presnosti, dostupnosti a schopnosti merať teploty v rozmedzí od -70°C po 500°C [14].

Teplotný senzor Dallas DS18B20

Číslicový digitálny senzor Dallas, najčastejšie sa vyskytujúci v púzdre TO-92 s tromi kolíkmi ponúka maximálne rozlíšenie 12 bitov, teda 0,0625°C. Operačný teplotný rozsah je -55°C až +125°C.

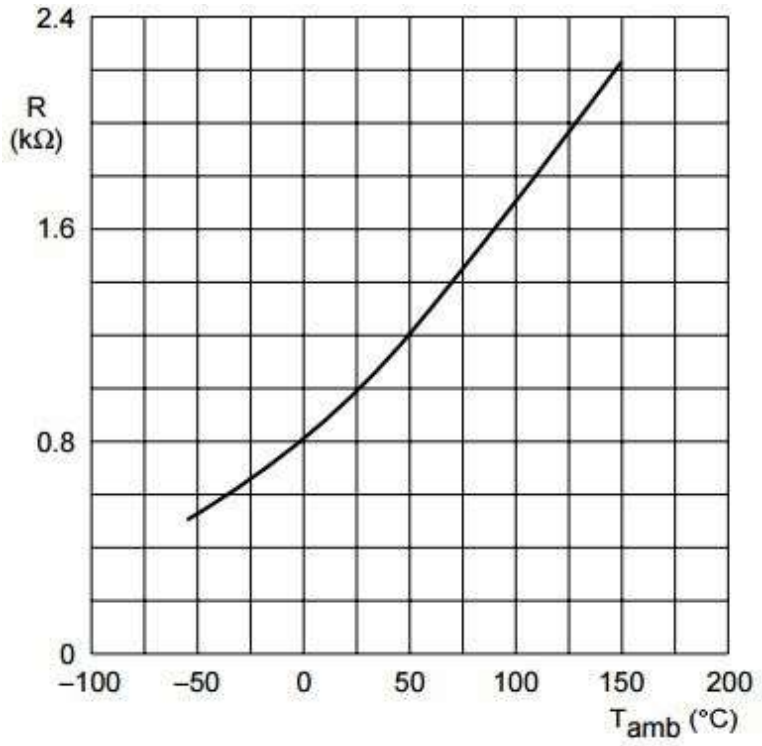
Každý tento senzor má unikátny 64 bitový identifikátor, vďaka ktorému je možné v sérii zapojiť aj viac ako jeden senzor [15].

Základný princíp senzoru je nasledovný. V púzdre sa nachádzajú 2 oscilátory. Jeden s vysokým tepelným koeficientom, druhý s nízkym. Len čo sa na zbernici vyskytne signál pre začatie merania teploty, oscilátory začnú pracovať. Po ich úspešnom splnení úlohy sa hodnota zapíše do 12 bitového registra a čaká na prečítanie hodnoty z nadradeného obvodu [7].

Obrázok 3.6Teplotný senzor DS18B20 [16].

Termistor KTY81-120

Tieto teplotné senzory majú kladný teplotný súčiniteľ odporu a sú vhodné na použitie v meracích a kontrolných systémoch. Senzory sú zapuzdrené v tvrdenom plastovom SOD-70 balíku. Ich tolerancia činí 0,5%. Vyrába ich spoločnosť Philips Semiconductors [17].



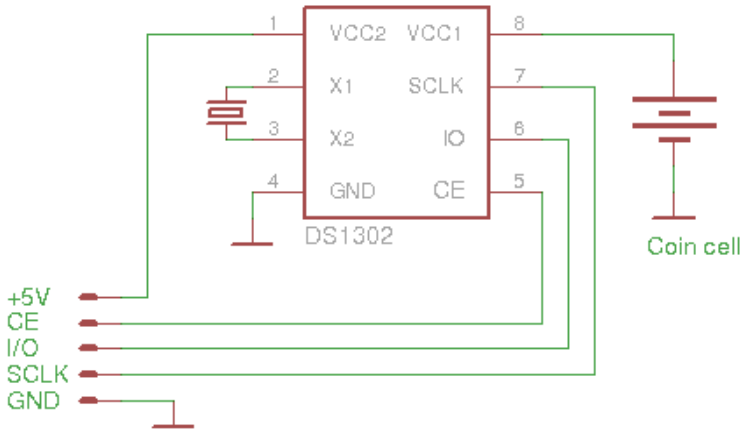
Obrázok 3.7Graf závislosti teploty od odporu (KTY81-120) [17]

3.1.6 Meranie času

Meranie fyzikálnej veličiny času, je pre náš projekt kľúčové. Bez tejto veličiny by sme nevedeli v reálnom čase určiť, kedy a čo sa dialo s naším servisným modulom. Na mikrokontroléri Arduino UNO existuje viacero spôsobov zaznamenávania času.

Najjednoduchším princípom je merať čas pomocou nastavenia istej odozvy. Priebeh merania vyzeral takto: Inicializovali sme program, nastavili sme veličinu času na 0 a spustili sme opakujúcu sa slučku. Na konci každej slučky sme pozastavili činnosť procesora na 0,2 sekundy. A tak sme mohli po 5-tich zopakovaniach prehlásiť, že prešla 1 sekunda, atď... Pri jednoduchých programoch, ktoré nevyťažujú procesor, je táto metóda veľmi jednoduchá a spoľahlivá. Avšak, keď sme pridali do opakujúcej sa slučky viaceré volania metód, výpočet niektorých funkcií a čítania portov, bola táto metóda počítania času irelevantná, keďže bola veľmi nepresná. Pomocou stopiek sme odmerali 10 minút (600 sekúnd) priebehu programu, očakávali sme, že podobné hodnoty nám vráti aj naše Arduino, avšak vrátil hodnotu 7 minút a 23 sekúnd (443 sekúnd). To značilo, že namiesto 3000 cyklov sa vykonalo len 2215. Nepresnosť merania činila 26,17% a to bolo pre nás neprijateľné.

Ďalšou a asi najlepšou metódou merania času, je pridať k Arduinu integrovaný obvod RTC (Real-time clock – Hodiny reálneho času). Základom takmer každého RTC modulu je riadiaci čip (napríklad DS1302), kryštál a nabíjacia batéria alebo superkondenzátor. Schéma zapojenia väčšinou vyzerá asi takto [18].

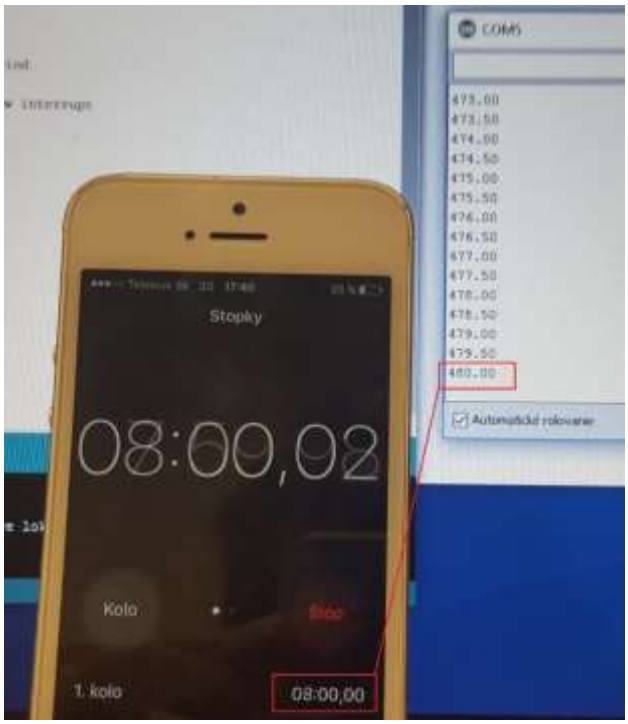


Obrázok 3.8 Schéma zapojenia RTC modulu [19]

Aj keď táto metóda merania času je asi najlepšia a je vysoko pravdepodobné, že sa k nej nakoniec vrátime, zistili sme že jednoduché a presné stopky sa dajú implementovať aj priamo na Arduine s čipom ATmega 328p.

Na mikrokontroléri Arduino UNO je prítomný 16Mhz kryštál, ktorý kooperuje s procesorom ATmega 328p. Pomocou knižnice „TimerOne.h“ je možné jednoducho pracovať so všetkými hardvérovými časovačmi. Sú tri. Len čo sme implementovali jednoduchý program,

otestovali sme ich. Časovače nás svojou vysokou presnosťou veľmi prekvapili. Stopli sme čas 8 minút a výsledky môžete vidieť aj sami v obrázku pod textom [20].



Obrázok 3.9Porovnanie presnosti „Arduino stopiek" a stopiek

3.1.7 Preposielanie údajov zo stratosférického balóna do aplikácie

Vypúšťanie stratosférických balónov by bolo veľmi zložité a takmer zbytočné, pokiaľ by sme nemali k dispozícii aktuálne informácie o polohe balónu, jeho nadmorskej výške, vonkajšej resp. vnútornej teploty, tlaku a ďalších veličín. Vďaka zbieraniu takýchto informácií je potom dohľadanie balónu oveľa jednoduchšie. Existujú spôsoby, ktoré vzhľadom na nadmorskú výšku, v ktorej balón praskol, rýchlosť a smer vetra, hmotnosť balónu dokážu vypočítať, kde približne stratosférický balón dopadne. Jednou z možností je použiť webovú službu na http://predict.habhub.org/ . V našom servisnom module bude použitý GPS modul, ktorý nás v každom čase dokáže informovať o súradniciach, kde sa balón nachádza či už počas letu, alebo po dopade.

Tieto informácie bude samozrejme potrebné nejakým efektívnym a spoľahlivým spôsobom preniesť do našej aplikácie, ktorá bude tieto informácie zbierať a vyhodnocovať, aby boli lepšie čitateľné.

Ako funguje prenos údajov:

Aby mohol prenos údajov fungovať, potrebujeme tzv. tracker, sieť a internetové pripojenie. Tracker je jednoduchý prijímač GPS, ktorý prenáša pomocou rádiového prenosu

prenáša tieto informácie na sieť, ktorá je pripojená k internetu, takže informácie sú dostupné všade na zemi. Sieť môže byť buď pozemná (napríklad amatérske APRS stanice) alebo sieť založená na satelitných systémoch (napr. systém Iridium).

Dostupné možnosti prenosu údajov:

A. Telefónny tracker

Nie veľmi vhodná voľba, ktorá obsahuje množstvo nevýhod. Telefóny nie sú stavané na také nízke teploty, aké sa vyskytujú v takých nadmorských výškach, kam ideme balón vypustiť. Ďalšou nevýhodou je to, že servisný modul s telefónom môže dopadnúť v oblasti, kde nie je pokrytie telefónnej siete, tým pádom sa neprenesú údaje o polohe servisného modulu.

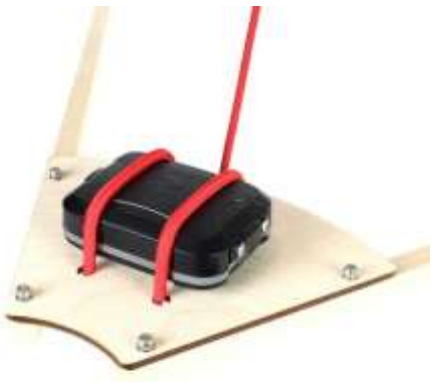
B. Satelitný tracker

Vhodnejšia voľba ako telefónny tracker. Na rozdiel od telefónov, satelitné trackery sa spoliehajú iba na satelitné siete, ktoré potom posielajú informácie ďalej na zem. Nevýhodou je, že anténa tohoto trackeru musí byť vždy nasmerovaná na oblohu.

C. Rádiový tracker

V našom projekte zostavíme vlastný tracker pozostávajúci s GPS shieldu pre Arduino a rádiového prijímača a vysielača.

Jednotlivé modely rádiových vysielačov a prijímačov budú uvedené v neskoršej kapitole.

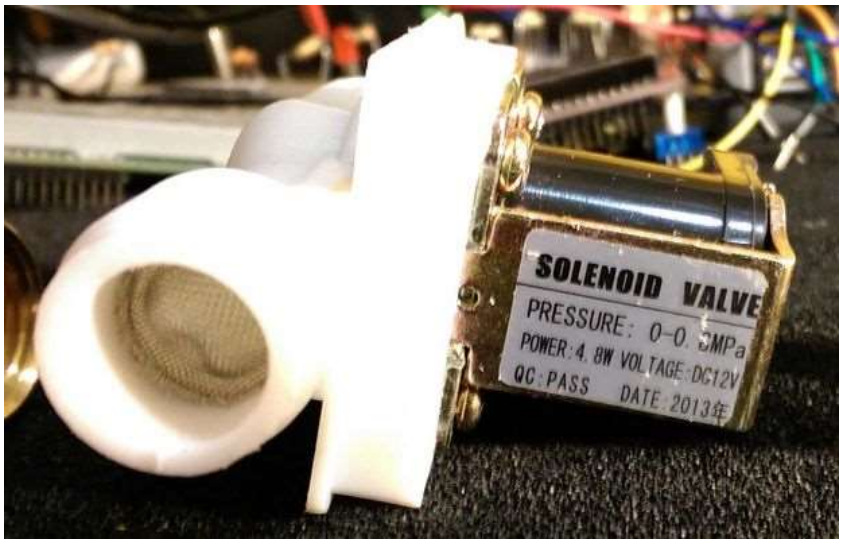


Obrázok 3.10 Rádiový tracker

3.1.8 Vypúšťací ventil na postupné znižovanie tlaku v stratosférickom balóne

Keďže so stúpajúcou výškou klesá tlak vzduchu, plyn v stratosférickom balóne sa rozpína, čím balón zväčšuje svoj objem až do bodu, kedy nepraskne. Aby sme s balónom dosiahli lepšie výsledky čo sa týka najvyššej dosiahnutej nadmorskej výšky, navrhli sme, aby náš stratosférický balón používal systém na postupné vypúšťanie plynu pomocou na diaľku ovládaného ventilu. Tým sa zníží tlak v balóne a spolu s ním aj rýchlosť stúpania.

Takéto ventily môžu byť ovládané jednoduchým elektromagnetom. Bežne sú dostupné v dvoch variantoch – normálne zatvorené a normálne otvorené. Normálne zatvorené ventily sú zatvorené a držia tekutinu resp. vzduch v ventile, až kým elektromagnet nedosiahne požadovaný prúd a napätie, kedy sa ventil otvorí a pustí tekutinu ďalej. Normálne otvorený ventil je opakom normálne zatvoreného. Takýto ventil býva väčšinou ovládaný Arduinom. Pre začiatok použijeme namiesto vypúšťacieho ventilu servomotor.



Obrázok 3.11 Vypúšťací ventil

3.2 Server

Zo špecifikácie nášho projektu vzniká potreba mať miesto, ktoré je dostupné z internetu. Na tomto mieste budú uložené dáta, ku ktorým budú mať prístup používatelia v rôznych roliach a externé časti nášho systému.

K serveru bude pristupovať náš zákazník, ktorý bude mať prístup k dátam, ktoré sú v reálnom čase prijímané zo sondy, odosielané a ukladané v databáze. Na našej webovej stránke bude verejnosti prístupná mapa s trasou letu balóna a základnou telemetriou, ktorá s aktualizuje v reálnom čase. Webová stránka bude slúžiť na propagáciu letu balóna a jeho lokalizáciu.

Dáta bude používať modul/podsystém na zdieľanie informácií n asociálnych sieťach, ktorý bude zverejňovať aktuálne informácie o balóne ako polohu, výšku balóna, teplotu okolia a podobne. Dáta na server bude cez aplikačné rozhranie odosielať riadiaci počítač letu, ktorý komunikuje priamo zo sondou pomocou GSM, rádia a iných technológií.

Je vhodné aby tieto funkcie boli združené na jednom mieste. Pre našu funkcionalitu musí server poskytovať webový server, databázové úložisko a aplikačný server. K tomu nám bude slúžiť server poskytnutý našou fakultou. Webový server zabezpečuje Nginx webserver.

3.2.1 Webserver

Počas tvorby projektu sme sa rozhodli použiť webový server nginx. Pred niekoľkými rokmi sme objavili tento webserver a do dnešného dňa sme presvedčení o tom, že je to najlepší webserver, aký bol kedy vytvorený. Vo veľkej konkurencii webserverov jeho popularita neustále rastie a v súčasnosti je používaný už na takmer 15 % webserveroch na celom internete. Webserver nginx poskytuje vysoký výkon, nízku spotrebu pamäte, vynikajúcu stabilitu a ľahkú konfigurovateľnosť – všetko atribúty, ktoré sú veľmi potrebné na dnešnú spoľahlivú internetovú prevádzku.

Nginx nám umožnil jednoducho nakonfigurovať všetky súčasti servera, ktoré sme chceli prezentovať na internete. Bol takisto nápomocný pri nastavovaní aplikačného servera.

Neoddeliteľnou súčasťou webservera nginx je modul PHP-FPM, ktorý interpretuje PHP skripty. S týmtom modulom máme takisto množstvo skúseností z minulých projektov a jeho použitie bolo jednoduché a modul spĺňal svoj účel.

Obrázok 3.12 Logo webserveru nginx

3.2.2 Služby na správu zdrojového kódu

Server poskytnutý fakultou sme sa rozhodli využiť naplno a okrem webového serveru Apache na ňom prevádzkujeme web tímového projektu a web na propagáciu letu balóna a jeho lokalizáciu. Obe tieto webové stránky pravidelne aktualizujeme, preto sme sa rozhodli používať verziovanie zdrojového kódu pomocou webovej služby GitHub a službu na kontinuálnu integráciu Jenkins.

Vždy, keď do jednej zo zmienených webových stránok pridáme nejakú funkcionalitu, každú verziu zmien ukladáme do určeného verejného repozitára s kódom. Vďaka tomuto postupu máme prehľad o vykonaných zmenách a ich prípadné vrátenie je jednoduchšie. Dokopy používame štyri github repozitáre:

1. Pre webovú stránku tímového projektu
2. Pre webovú stránku s aktuálnymi informáciami o balóne
3. 2 repozitáre pre kód určený pre mikropočítače

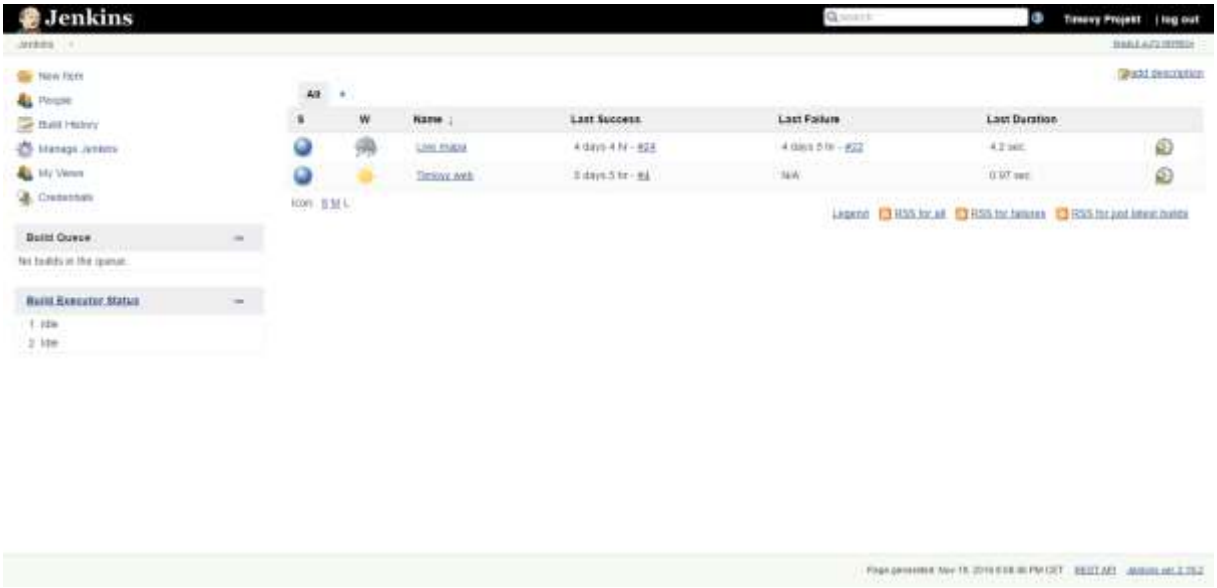
Prácu so zdrojovým kódom uľahčujú intuitívne príkazy služby git ako napríklad:

- git add .

    - označenie aktuálneho priečinka ako „git repozitár“, v tomto priečinku sa budú sledovať zmeny zdrojového kódu
- git commit -m „sprava“
    - pomocou príkazu git commit sa pridajú zmeny do lokálneho repozitáru. Pomocou prepínaču „m“ dokážeme ku commitu pridať tzv. commit message, ktorý by mal všeobecne špecifikovať cieľ resp. obsah aktuálneho commitu
- git push
    - pomocou príkazu git push sa zmeny, ktoré boli vykonané v lokálnom repozitári, pošlú priamo na vzdialený server. K tomuto príkazu sa bežne pripája aj vetva zdrojového kódu, do ktorej ideme zmeny „pushnúť“, zvyčajne ide o hlavnú „master“ vetvu.
- git pull
    - pomocou príkazu git pull je možné do lokálneho repozitáru uložiť zdrojový kód zo vzdialeného repozitáru

Ďalší prístup ku správe zdrojového kódu, ktorý sme sa rozhodli použiť je kontinuálna integrácia. Princípom tohto prístupu je priebežné sledovanie zmien zdrojového kódu, z ktorého je automaticky zostavovaný a testovaný projekt. [21]

Pre implementáciu tohto prístupu sme zvolili existujúci nástroj Jenkins, ktorý považujeme za najvhodnejší pre náš projekt. Na nasledujúcom obrázku vidíme aplikačné rozhranie programu Jenkins. Poskytuje integráciu s už spomínanou službou GitHub, takže dokáže zmeny automaticky sťahovať a odosielať do potrebných repozitárov.



Obrázok 3.13 Grafické rozhranie Jenkins

3.2.3 Aplikačný server

Existuje viacero technológií, ktoré môžeme použiť pre náš server. Existuje viacero možností postavených na rôznych jazykoch. Niektoré frameworky poskytujú robustnú architektúru a komplexnú funkcionalitu, na druhej strane sú rýchle, jednoduché frameworky, ktoré je možné jednoducho upraviť podľa potrieb.

Zoznam uvažovaných frameworkov:

- Java: Spring, Stripes
- Python: Django, Flask, Pyramid
- Ruby: Ruby on Rails
- JavaScript: Node.js
- PHP: TYPO3, Yii, CodeIgniter

Zvolili sme si webový framework Flask postavený na jazyku Python. Tento framework je jednoduchý, nenáročný. Neobsahuje mnoho funkcionality, no je možné ho rozšíriť cez poskytované rozšírenia. Vyberali sme taktiež podľa predchádzajúcich skúseností nášho tímu.

Spolu s aplikačným serverom Flask sme použili modul gunicorn, ktorý nám pomohol pri spracovaní socketov.

3.2.4 Databázové úložisko

V databáze budú uložené údaje, ktoré sú zaznamenávané počas letu balóna. Na základe povahy a štruktúry dát sme zvolil relačnú SQL databázu. V prvotných fázach projektu sme na základe nízkej frekvencie a početnosti dát zvolili databázu SQLite, ktorú bolo jednoducho možné implementovať do našej aplikácie.

V neskorších fázach projektu sa avšak intenzita toku dát zvýšila a boli sme nútení prejsť na výkonnejšie riešenie, ktorým sa po dlhých úvahách stala najpoužívanejší databázový server súčasnosti, MySQL. Hneď od prvého momentu sme cítili veľké zlepšenie, najmä čo sa týka výkonu.

Tieto zmeny z dlhodobejšieho hľadiska vidíme veľmi pozitívne. Tieto riešenia sú navyše škálovateľné, to znamená, že pokiaľ by naša aplikácia mala zobrazovať niekoľko letov balónov naraz, nemala by s tým mať žiaden problém.

V databáze budú uložené údaje, ktoré sú zaznamenávané počas letu balóna. Na základe povahy a štruktúry dát sme zvolil relačnú SQL databázu. Keďže frekvencia a početnosť dát nie je príliš veľká, zvolili sme databázu SQLite, ktorú je možné jednoducho implementovať do našej aplikácie. Pokiaľ sa vyskytne neočakávaná potreba siahnuť po výkonnejšom databázovom systéme, môžeme zvoliť MySQL alebo PostgreSQL, ktoré sú podporované frameworkom Flask. S týmto variantom je vhodné počítať pri návrhu architektúry systému.

3.3 Návrh

3.3.1 Architektúra servera

Architektúra servera využíva princípy viacvrstvovej architektúry. Jednotlivé vrstvy majú definovanú funkcionalitu a sú izolované v možnosti komunikovať len s najbližšími vrstvami systému.

Náš systém obsahuje tri základné vrstvy:

a. Vrstva riadenia

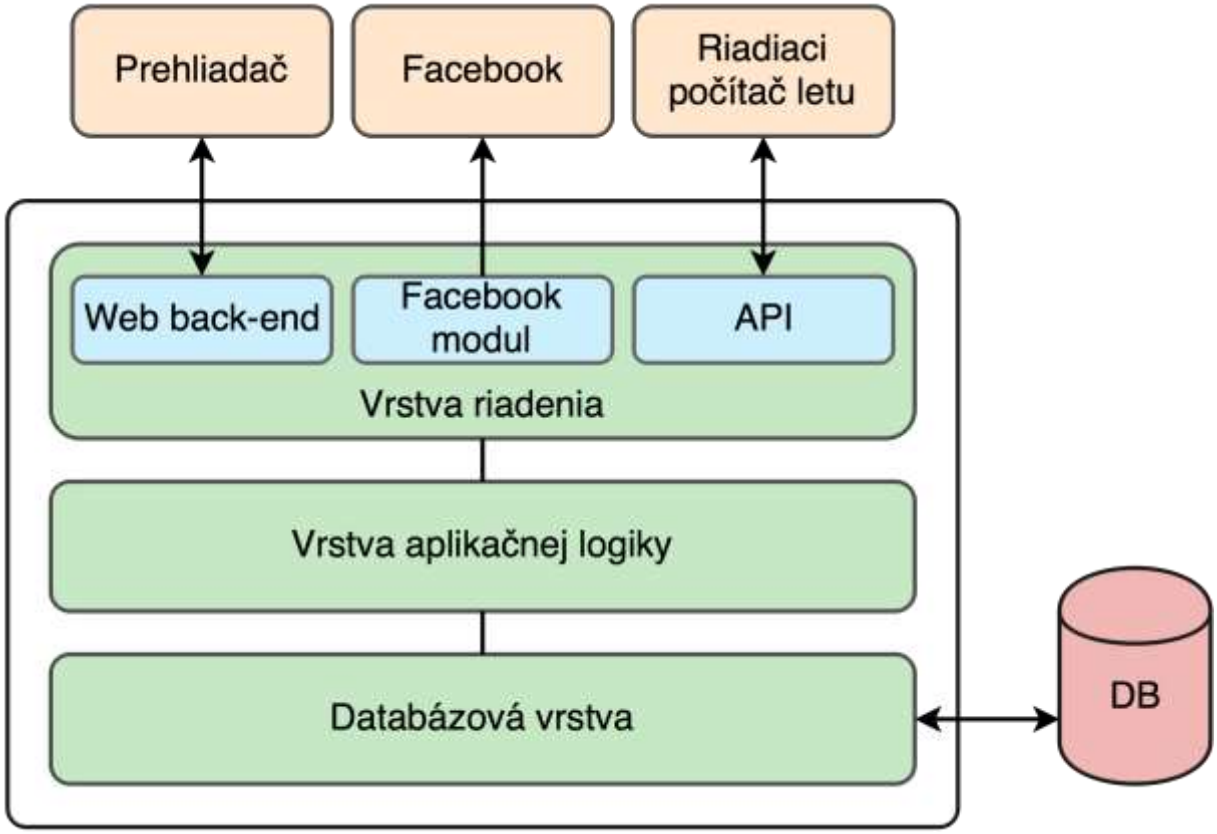
Vrstva riadenia obsahuje logiku, ktorá riadi prístup externých častí nášho systému. Stará sa o zobrazenie webovej stránky s trasou balóna v reálnom čase a odosielanie aktuálne pozície v reálnom čase do klientského prehliadača. Ďalej obsahuje modul, ktorý aktualizuje príspevky na sociálnych sieťach. Poslednou časťou je aplikačné rozhranie, ktoré využíva riadiaci počítač letu pre odosielanie údajov zo sondy. Poslednou časťou je používateľské rozhranie pre zákazníka, v ktorom si môže prezerať všetky údaje prijaté zo letu jeho sondy a možnosť ich exportovať.

b. Vrstva aplikačnej logiky

Vrstva aplikačnej logiky obsahuje hlavnú logiku nášho servera. Zabezpečuje komunikáciu jednotlivých modulov vrstvy riadenia s databázovou vrstvou.

c. Vrstva databázy

Databázová vrstva obsahuje logiku pre prístup k databáze. Databázový server je tak v prípade neočakávanej potreby väčšieho výkonu, spomenutej v časti Analýza, možné jednoducho vymeniť v tejto vrstve.



Obrázok 3.14 Navrhovaná viacvrstvová architektúra servera

3.3.2 Webová stránka

Hlavnou funkciou webovej stránky je zobrazovať na mape trasu letu balóna so základnými informáciami o stave. Webová stránka obsahuje dve hlavné časti: mapa a bočný panel.

Použité sú mapové podklady od spoločnosti Google, ktorá poskytuje možnosť zobraziť rozhranie s mapou na našej webovej stránke. Google takisto poskytuje aplikačné rozhranie, cez ktoré je možné do mapy pridávať vlastné prvky. Do mapy tak môžeme pridať body týkajúce sa letu balóna a čiary znázorňujúce trasu letu balóna.

Pretože pozícia balóna a ostatné údaje sa aktualizujú v reálnom čase, resp. hneď po validovaní hodnôt prijatých zo sondy, je potrebné aby si prehliadač klienta s webovou stránkou udržoval spojenie s naším serverom. Túto komunikáciu zabezpečuje protokol WebSocket, cez ktorý server odosiela prehliadaču aktuálne dáta, ktoré sú následne aktualizované na webovej

stránke.

Obrázok 15 Navrhovaná podoba webovej stránk

3.3.3 Návrh komunikácie – GSM

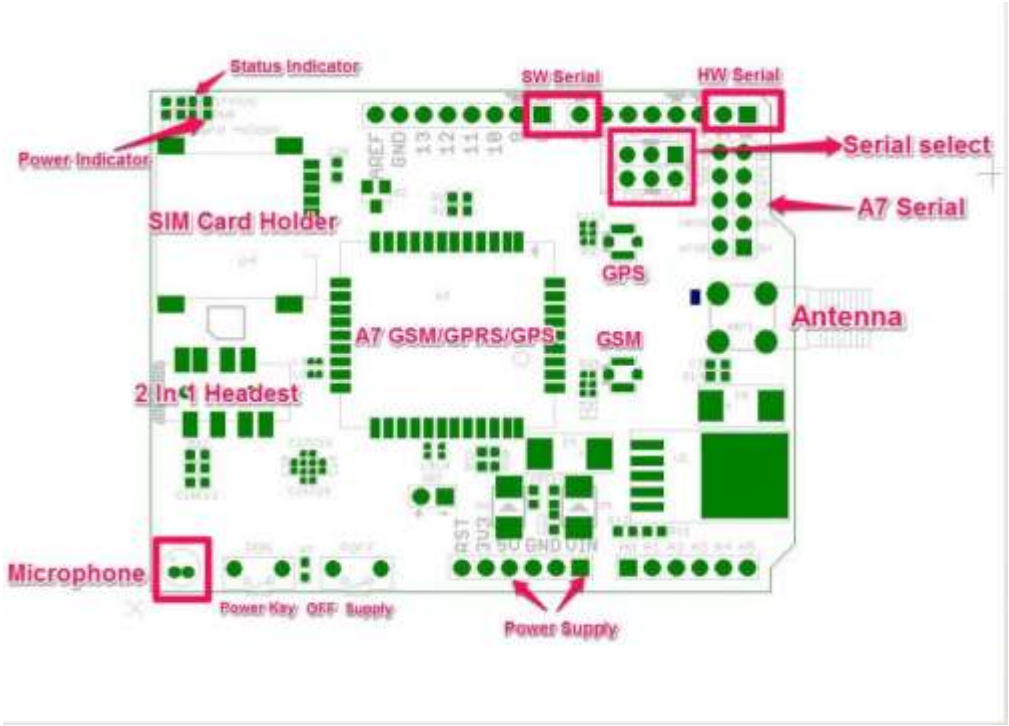
Obvod s názvom A7 GPRS / GSM / GPS, je obvod, ktorý v našom servisnom module používame na komunikáciu s modulom pomocou mobilnej siete. Obvod používa najnovší GPRS / GPS / GSM čip A7. Obvod podporuje siete GSM / GPRS Quad-band (850/900/1800/1900). Tiež podporuje hlasové hovory, SMS, GPRS dátové služby a funkcie GPS. Obvod môže byt použitý na vytvorenie jednoduchého telefónu.

Modul je riadený AT príkazmi cez UART a podporuje 3,3V a 4,2V logické úrovne.

a. Technická špecifikácia

- Operating temperature -30 ? to + 80 ?;
- 1KG peak suction
- Low standby current
- Operating Voltage 3.3V-4.2V;
- Power voltage> 3.4V;
- Standby average current 3ma less;
- Support the GSM / GPRS four bands, including 850,900,1800,1900MHZ;
- Support China Mobile and China Unicom's 2G GSM network worldwide;
- GPRS Class 10;
- Sensitivity <-105;
- Support voice calls;
- ?Support SMS text messaging;
- Support GPRS data traffic, the maximum data rate, download 85.6Kbps, upload 42.8Kbps;
- Supports standard GSM07.07,07.05 AT commands and extended commands Ai-Thinker;
- Supports two serial ports, a serial port to download an AT command port;
- AT command supports the standard AT and TCP / IP command interface;
- Support digital audio and analog audio support for HR, FR, EFR, AMR speech coding;
- Support ROHS, FCC, CE, CTA certification;
- SMT 42PIN

b. Schéma:



Obrázok 3.16 Schéma GSM modulu

Power supply – Napájanie 5-9VDC z externého zdroja
Antenna interface – externá GPS a GSM anténa
Serial port select – Výber HW alebo SW sériového portu pomocou jumperov.
Hardware Serial - D0/D1 na Arduino/Crowduino
Software serial - D7/D8 na Arduino/Crowduino
Microphone – na uskutočnenie hovoru
Speaker - na uskutočnenie hovoru
Power key – zapínacie tlačidlo
OFF key – vypínacie tlačidlo

c. Využitie pinov na pripojenom Arduine

D0 - Nepoužitý v prípade použitia HW sériového portu
D1 – Nepoužitý v prípade použitia HW sériového portu
D2 - Nepoužitý
D3 - Nepoužitý
D4 - Nepoužitý
D5 - Nepoužitý
D6 - Nepoužitý
D7 – SW sériový port
D8 - SW sériový port
D9 – Využitý na SW ovládanie napájanie GSM shieldu
D10 - Nepoužitý

D11 - Nepoužitý
D12 - Nepoužitý
D13 - Nepoužitý

d. AT príkazy

Jednotlivé funkcie telefónu sa ovládajú zadávaním príslušných AT príkazov. Pre sieť GSM, európsky telekomunikačný a štandardizačný inštitút (ETSI) stanovil špecifické profily AT príkazov v norme (GSM 07.07). AT príkazy posielané do telefónu môžu byť zadávané v troch podobách:

a) test AT príkazu, (či telefón príkazu rozumie) je AT+ príkaz =? CR
b) načítanie nastavených hodnôt z telefónu AT+ príkaz ? CR
c) zápis dát alebo hodnôt do telefónu AT+ príkaz = parameter CR

Skratka AT je začiatok príkazu, doplnenie podľa požadovaného povelu, sa zadáva iba v prípade, ak to požaduje príkaz pre nastavenie alebo zápis dát a je potvrdenie príkazu. Pri komunikácii z počítača je to ENTER a pri komunikácii z mikrokontroléra hodnota 0Dh.

Najjednoduchším AT príkazom je samotná dvojica znakov AT. Odpoveď mobilného telefónu na správne zadaný príkaz je OK. Zle zadané príkazy telefón ignoruje. Ak sú však v príkaze zadané nesprávne parametre, telefón odpovie ERROR. Pri tvorbe GSM komunikátora využívame najmä AT príkazy pre prácu s textovými správami sms.

e. Práca s textovými správami SMS

SMS správy v telefóne sa delia do troch skupín. Každej skupine môže byt priradený niektorý z pamäťových priestorov. Počet a kapacita pamäťových priestorov sú však dané typom telefónu. Každej z troch skupín sú priradené určité operácie pre prácu s textovými správami. Prvej skupine sú priradené operácie na čítanie a mazanie správ. Druhá skupina obsahuje operácie zapisovania a odosielania správ. Do tretej skupiny sú ukladané prijaté správy.

f. Odoslanie textovej správy

Odoslanie textovej správy v našom prípade prebieha pomocou sekvencie konkrétnych AT príkazov. Konkrétne ide o príkazy AT+CMGF a AT+CMGS. AT+CMGF slúži na nastavenie módu posielania textových správ a AT+CMGF slúži na zvolenie telefónneho čísla na ktoré sa má správa odoslať. Každý AT príkaz musí byt potvrdený znakom CR, na čo GSM shield odpovedá správou „OK“. Na ukončenie odosielaného textu slúži znak CTRL+Z.

Konkrétna nami zostrojená funkcia na odoslanie SMS teda vyzerá nasledovne:

```
void SendSMS(String str){
        swSerial.print("AT+CMGF=1");    //Odoslanie sms v textovom móde
        delay(100);
        swSerial.print(char(13));        //CR
        delay(100);
        swSerial.print("AT+CMGS=+421902744909;");
        delay(100);
        swSerial.print(char(13));        //CR
        delay(100);
        swSerial.print(str);             //Samotný text správy
        delay(100);
        swSerial.print(char(26));        //CTRL+Z, ukončenie odosielania
        delay(100);
        swSerial.println();
}
```

g. Prehľad niektorých AT príkazov podporovaných GSM shieldom

AT + CGATT = 1	Return OK, attached to the network
AT + CGACT = 1	Activate the network, then you can use the tcp/ip command
AT + CIPSTART = "TCP" "121.41.97.28", 60000	TCPIP server connection
AT + CIPCLOSE	Close TCP/IP connection
AT+CMGF=1	Send a text message
AT+CIPTCFG	Passthrough mode configuration
AT+CIPTMODE	Enter the passthrough mode

AT+GPS=1	Open GPS
AT+GPS=0	Close GPS
AT+AGPS=1	Open AGPS
AT+AGPS=0	Close AGPS

3.4 Implementácia 1. prototypu

3.4.1 Zaznamenávacia časť

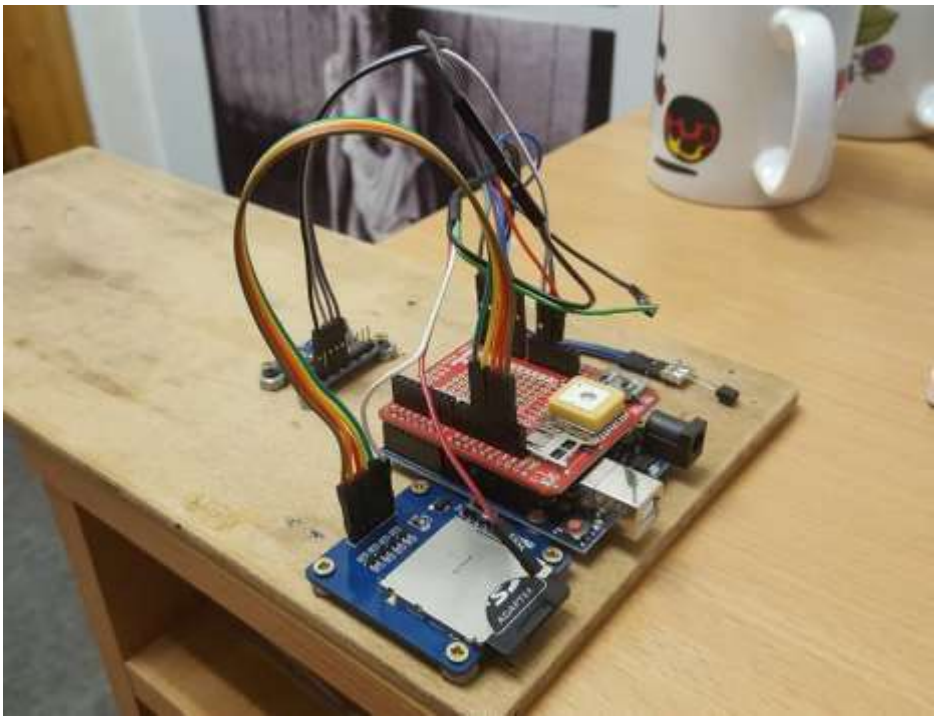
V prvom prototype využívame viacero senzorov pomocou, ktorých meriame/počítame rôzne, pre nás dôležité fyzikálne veličiny. Tieto veličiny sú čas, teplota (2-ma senzormi), tlak a GPS súradnice. Ako sme písali vyššie v analýze na počítanie času aktuálne využívame 16MHz kryštál nachádzajúci sa na prototypovacej platforme Arduino UNO, ktorý kolaboruje s mikročipom ATMega328p. Pomocou neho vieme jednoducho časovať volané procedúry a na tomto časovaní je aj založená architektúra firmvéru (programu), na ktorom stále pracujeme. Časť programu sa nachádza pod odsekom.

```
void Time_function()
{
 timer2++;          // spocitavanie polperiod
}
void setup()
{
 Serial.begin(9600);  // inicializacia seriovej linky
 Timer1.initialize(500000);         // inicializacia casovaca 1 a nastavenie periody
 Timer1.attachInterrupt(Time_function);  // volanie procedury pri preteceni nastaenej hodnoty
 bmp.begin(); // inicializacia bmp senzoru
}
void loop()
{
 timer = timer2 / 2;  // pocitanie celych sekund
 if ((timer != oldTime) || firstTime)            // podmienka vstupu do hlavnej casti programu
 {
  firstTime = false;
  oldTime = timer;
  temp = kty(0); // volanie funkcie na vypocet teploty z analogoveho portu 0
  bmp.getEvent(&event); // odcitanie hodnot zo senzorov
```

```
    bmp.getTemperature(&temp2); //zapisanie teploty do premennej temp2
    Print_function();           // volanie funkcie na vypis
  }
}
```

Na meranie vonkajšej teploty využívame termistor KTY81-120, pretože dokáže pracovať v podmienkach, ktoré sú pre nás kľúčové. Na meranie vnútornej teploty využívame teplotný senzor, ktorý sa nachádza na multi-senzore 10DOF IMU. Ide o čip BMP180, okrem merania teploty dokáže zaznamenávať aj tlak. Po dôkladnom hľadaní senzoru tlaku s vhodným meracím rozpätím sme nenašli žiaden, ktorý by dokázal merať tlak vo výške nad 10km. Avšak na druhú stranu aj hodnoty do 10km môžu byť zaujímavé a preto aj túto fyzikálna veličinu pomocou toho multi-senzoru meriame.

V neposlednom rade meriame polohu pomocou GPS senzoru na Arduino štíte od spoločnosti Sparkfun. Tento senzor nám z analýzy vyšiel ako najlepší avšak v ďalšej iterácií plánujeme do nášho servisného modulu zahrnúť ešte jeden sekundárny GPS modul (kvôli nepredvídateľným situáciám, ktoré by mohli nastať).



Obrázok 3.17 Prvý prototyp na platforme Arduino UNO

3.4.1.1 Backup pamäťový modul

Počas implementácie prototypu sme sa stretli s otázkou zvýšenej spoľahlivosti, ktorá je kľúčovým bodom nášho servisného modulu. Údaje čo meriame sú základom nášho projektu (úspešného letu) a teda je dôležité ich ukladať na čo najviac miest.

Všetky údaje čo pošleme ukladáme do databázy nášho servera, avšak pokiaľ by nastala situácia, že by sa dáta nepodarilo odoslať alebo prijať, nemali by sme čo uložiť. Najjednoduchšou možnosťou ako predchádzať tomuto riziku je dáta po nameraní okamžite ukladať na nejaké médium. Rozhodli sme sa pre pamäťovú kartu, vzhľadom na možnosti, ktoré ponúka (veľká kapacita, nízke rozmery, odolnosť voči podmienkam a iné...).

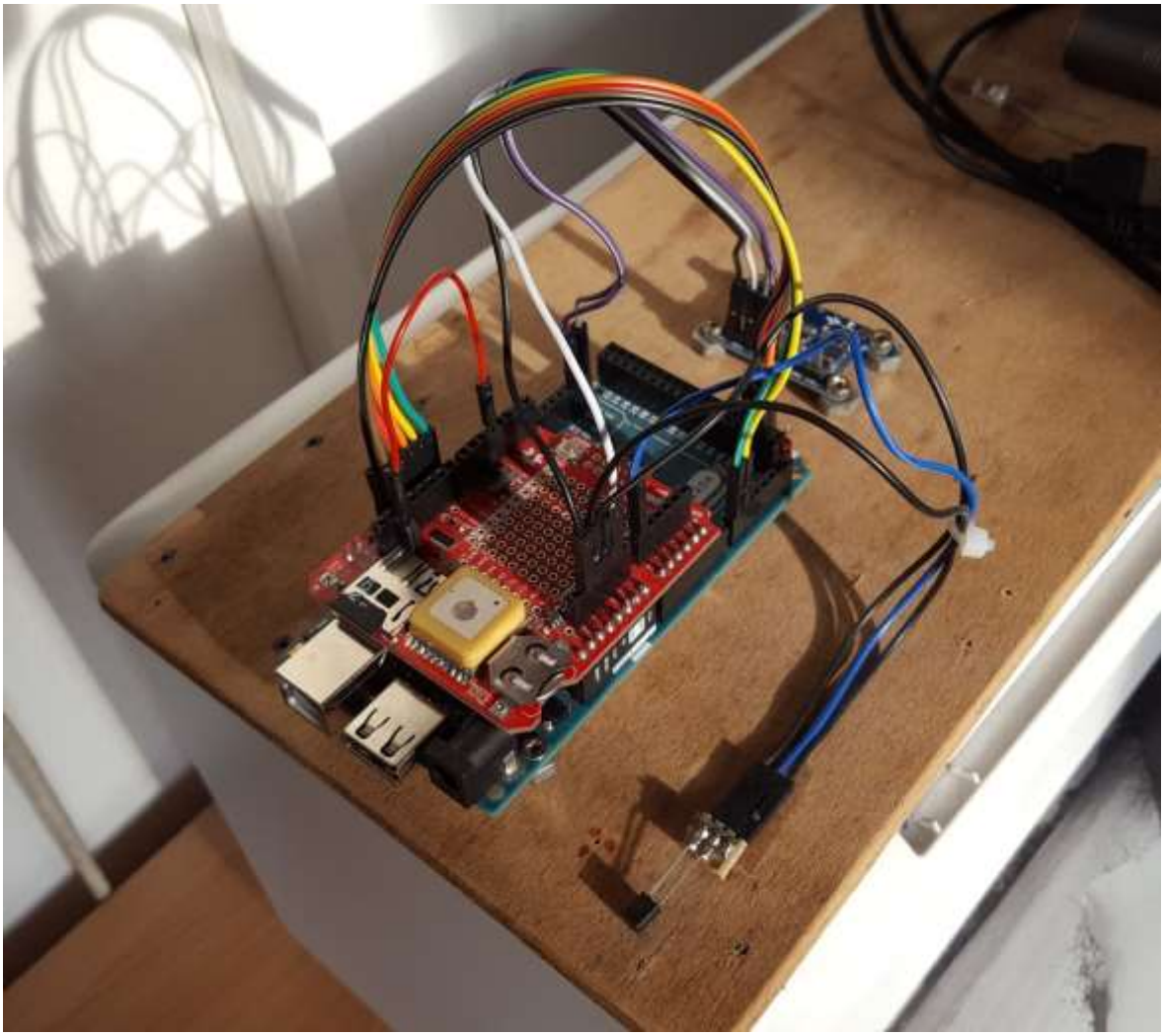
Náš GPS Logger štít od spoločnosti Sparkfun obsahuje slot aj riadiacu jednotku k pamäťovej karte a aj práve preto sme tento štít pri analýze vybrali. Je pripojený pomocou zbernice SPI. Ako som písal v kapitole vyššie, keďže štít je navrhnutý priamo pre prototypovaciu platformu Arduino UNO, aby fungoval na platforme Mega, musel som vytvoriť externé prepojenie (namiesto pinov D10 – 13 využívam D50 – D53).

Skúšali sme viacero riešení ukladania dát, avšak ako najjednoduchší spôsob nám vyšlo zapisovať údaje priamo do textového súboru a nové údaje vždy pripisovať na koniec.

Pred každým odosielaním, sa všetky dáta uložia na pamäťovú kartu, ktorá sa hneď po uložení dát zavrie a najbližšie sa otvorí až pri ďalšom zápise. Toto ošetrenie vykonávame z dôvodu, aby nenastala situácia, že viacero dát sa nekorektne zapíše. Vďaka tejto metóde neprídeme o žiadne dáta ani pri výpadku spojenia medzi modulom a serverom. Pod textom sa nachádza časť programu, ktorá zabezpečuje ukladanie dát na pamäťovú kartu a taktiež obrázok aktuálneho prototypu.

```
void logGPSData()
{
 int connected_card = digitalRead(4);
 if (connected_card)        //testovanie vloženia karty v moduli
 {
 createFile("test.txt");
 writeToFileFloat(timer);
 writeToFileFloat(aktlng);
 writeToFileFloat(aktlat);
 writeToFileFloat(aktalt);
 writeToFileFloat(temp);
 writeToFileFloat(temp2);
 writeToFileFloat(event.pressure);
 writeToFileFloat(urtime);
 writeToFileFloat(aktsat);
 file.println("\n");
 closeFile();
 }
```

```
  else
    initializeSD(); //v prípade, že karta nie je vložená/vložená nesprávne karta sa znovu inicializuje
}
```



Obrázok 3.18 Snímka aktuálneho servisného modulu (Arduino MEGA)

Aby sme zabránili neočakávanému dočasnému odpojeniu pamäťovej karty, pred zápisom na ňu, kontrolujeme pin (CD – card detect, spínač) vloženia karty. Tento pin obsahuje štít Sparkfun. Vzhľadom na jeho funkcionalitu zapája sa cez pullup odpor. Keďže čipi AVR tieto odpory (v rozmedzí 20kΩ až 50kΩ) majú už v sebe implementované, softvérovo (pomocou príkazu - digitalWrite(4, HIGH)) som si odpor na digitálnom pine 4 aktivoval a nemusel teda riešiť externý odpor [22] .

3.4.1.2 Testovanie straty GPS signálu

Počas Testovanie súčiastok a ich vlastností je jedna z najdôležitejších oblastí, ktoré pri projekte nášho typu (jednorazové vypustenie balónu spolu s naším servisným modulom a následná absencia fyzického kontaktu s produktom) je nutné riešiť.

Zaznamenávanie GPS polohy spolu s jej preposielaním na náš server (pomocou rádia/GSM siete) sú najdôležitejšie a najkritickejšie problémy, ktoré riešime. Napríklad: na preposielanie údajov využívame rádiovú komunikáciu, ktorá je celkom spoľahlivá ale ako zálohu v prípade výpadku tohto signálu vieme základné údaje odosielať aj pomocou SMS správ (naraz sa dáta odosielajú cez obe metódy) a tým násobíme spoľahlivosť prenosu.

Čo sa týka zaznamenávania polohy, využívame 2 nezávislé zariadenia, ktoré merajú GPS súradnice, avšak je dobré vedieť ako sa ktoré v krízovej situácií (napr. strata signálu) zachová. Práve aj preto vznikol tento test.

Keďže je bežná situácia, že počas letu nám môže vypadnúť GPS signál, je nutné túto možnosť otestovať aj pred začatím samotného letu. Pri testovaní nás zaujímalo hlavne aké výstupy nám bude GPS senzor generovať, či sa po získaní stavu znovu sám zotaví, či je nutné riešiť nejaké podporné opatrenia pri opätovnom získaní signálu, ako sa zachovajú naše ošetrovacie mechanizmy na validáciu súradníc a iné...

Testovanie č. 1. prebiehalo nasledovne: vonku pred školou som stál s modulom, kým som nechytil patričné GPS satelity (aspoň 4 – vtedy vie modul určiť polohu + čas). Následne som šiel dnu do školy, kde však stále som chytal signál (nie moc dobrý ale predsa). Pár krát som sa previezol vo výťahu hore, dole a signál som stratil (paráda). GPS vypisovalo klasickú „nulovú“ súradnicu ako pri inicializácií (0, 0, 0). Následne som vyšiel von a čakal kým signál znovu nechytím. Ani nie do 30 sekúnd som mal opäť optimálny signál. Celý proces som viac-krát opakoval a s testovania sme zistili, že modul sa pri výpadku a opätovnom signálu vie jednoducho vysporiadať.

Priebeh testu č. 2: Na balkóne svojej izby som sa nachádzal spolu s modulom a čakal som kým GPS shield nezíska minimálny počet satelitov, na základe ktorého je možné pozíciu vypočítať (minimálne 3 satelity, čím viac tým sa poloha spresňuje). Pomocou konzoly a pomocných výpisov „počtu nájdených satelitov“. Behom krátkej chvíle zariadenie zachytilo 6 satelitov. Poloha sa začala vykresľovať na servery (na ktorý sa údaje dostali pomocou postupu poukázaného o odstavec vyššie). Potom som spolu so zariadením išiel dnu (do miestnosti za 3 steny), kde som o malú chvíľu satelity stratil (mal som ich 2 a menej). Tento proces som X krát opakoval (okolo 100-krát) a vždy sa zariadeniu pod priamym nebom podarilo satelity nájsť. Z výpočtov súradníc vychádzali len hodnoty 0 (pre os X,Y a Z). Táto možnosť je softvérovo

ošetrená (takéto súradnice obsahujú hodnotu „fail“ -> server ich nevykreslí). Ako extra ošetrenie zariadenie testuje namerania náhodných (zlých) súradníc. Vychádzame z predpokladu, že náš balón v horizontálnom smere neprekročí rýchlostnú hranicu 100km/h (dosť predimenzované). V prípade, že sa tak stane (vzhľadom na výpočet náhodných súradníc v čase), algoritmus vyhlási súradnice za neplatné, rovnako ako v prípade nedostatku počtu satelitov.

Pod odstavcom sa nachádza obrázok, ktorý monitoruje priebeh (trajektóriu simulovaného letu – strata signálu...) testu uskutočnený na internáte mladosť:

Obrázok 3.19 Simulácia výpadku signálu č. 2

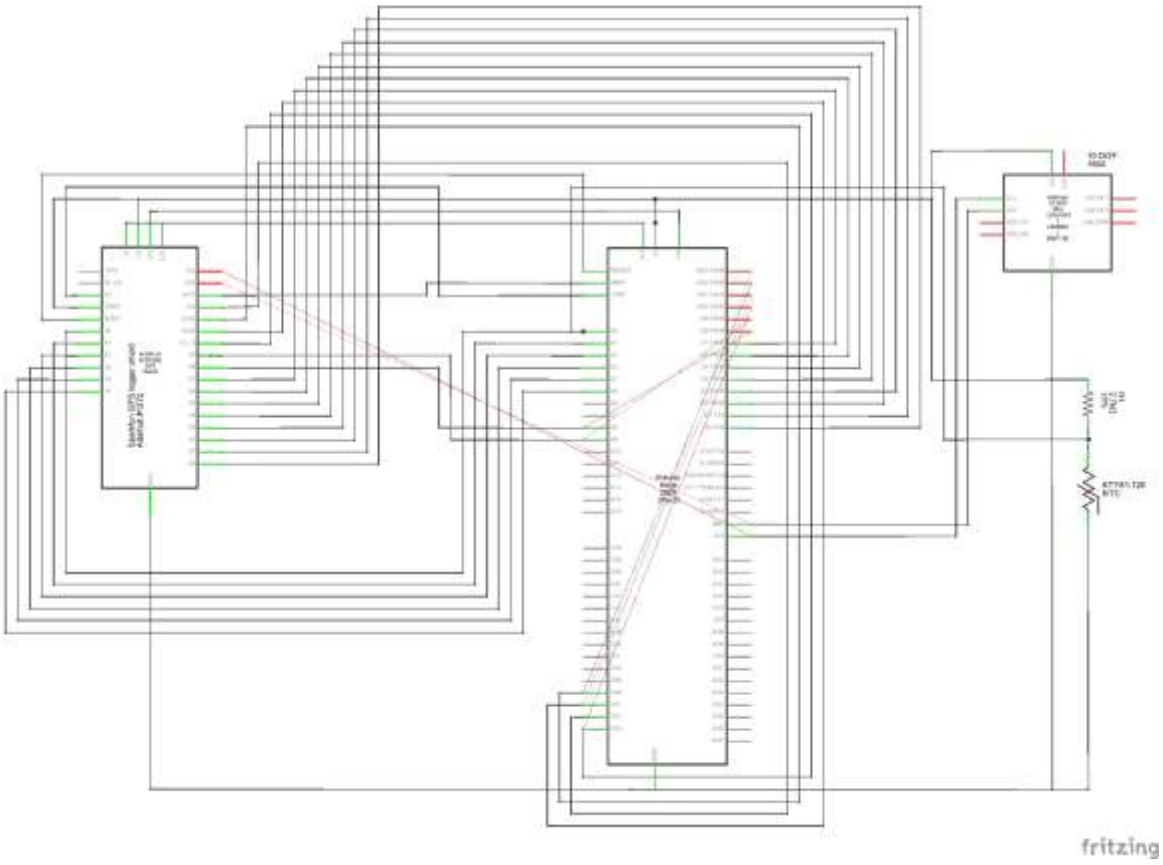
3.4.1.3 Migrácia HW z Arduino UNO na Arduino MEGA + schéma zapojenia

Vzhľadom na obohacovanie modulu o HW, ale aj SW súčasti, sme začali byť limitovaný počtom zberníc, pinov, pamäte na premenné ale aj program. Jediné možné (rýchle) riešenie bolo prejsť na inú prototypovaciu platformu – Arduino MEGA.

Je síce pravda, že štíty (GPS, GSM), ktoré máme sú veľkosťou aj väčšinou pinou kompatibilné avšak napriek tomu sme museli riešiť isté množstvo komplikácií, vzhľadom na rozličné rozmiestnenie istých zberníc.

Pod textom je možné vidieť aktuálne zapojenie servisného modulu. Keďže sme prototypovali na platforme Arduino UNO a nie všetky piny (kolíky), majú na daných platformách rovnaké rozmiestnenie, museli sme ich vhodne poprepájať externými prepojeniami.

Napríklad Arduino UNO má SPI zbernicu na pinoch D10 - D13, ale Mega nie. Preto sme ich na megu pripojili na piny D50 - D53. Taktiež I2C zbernicu má UNO na pinoch A4-A5 a Mega napr. na D20-D21. Tieto zmeny sme vykonávali vzhľadom na kompatibilitu Sparkfun GPS Loger štítu a multisenzoru 10-DOF.



Obrázok 3.20 Schéma zapojenia servisného modulu

3.4.1.4 Zaznamenávanie GPS súradníc a ich ošetrenie

Zaznamenávanie GPS súradníc, je jednou s kľúčových funkcií pri dohľadaní balónu. GPS štít Sparkfun toto dokáže a navyše vykazuje dobré hodnoty, čo sa týka odolnosti a funkcionality aj v podmienkach, do ktorých ho plánujeme vypustiť.

Na prenos údajov zo zariadenia do Arduina sa využíva zbernica UART, ktorá pozostáva z 2-ch pinov: RX a TX. RX je určený pre odosielanie údajov a TX pre ich príjem. Z dôvodu lepšieho pochopenia zbernice využívame v projekte emulovanú zbernicu, ktorú nám dovoľuje, na niektorých pinoch Arduina vytvoriť knižnica: SoftwareSerial.h.

Na prácu so súradnicami, časom a inými veličinami, ktoré nám daný štít/modul poskytuje, využívame knižnicu: TinyGPS++.h. Vďaka nej je práca so súradnicami veľmi jednoduchá.

Po prečítaní viacerých fór, sme zistili, že niektoré GPS moduly môžu niekedy namerať nevalidnú súradnicu (V zmysle: každé HW zariadenie môže obsahovať nejakú chybu, ktorá by však % mala byť veľmi malá. A je teda možné, že počas letu nám GPS senzor nameria súradnicu, ktorá s meraním nemá nič spoločné: napr. 100km úplne vedľa reálnej trasy letu).

V analýze sme zistili, že balón ktorý vypúšťal študent Žilinskej univerzity dosiahol maximálnu rýchlosť 90km/h a to pri páde. Preto po každom nameraní GPS súradníc, zisťujeme ich validnosť na základe podmienky: Ak rýchlosť v hociktorom vodorovnom smere prekročila rýchlosť 100km/h, považuj súradnicu za nesprávnu. Pre ďalšie meranie následne zväčšujeme možný rozptyl. Pod textom sa nachádza funkcia, ktorá vzdialenosti týchto súradníc overuje – následne vyhodnocuje či sú súradnice validné alebo nie.

```
//#define ref 0.0001259 // 14 meters - pre 0,5 sekundove merania
//#define ref 0.005036 // 556 meters (pre 20 sekundove intervaly - 100km/h
#define ref 0.00125 // 139 meters (pre 5 sekundove intervaly - 100km/h

boolean isGpsValid()
{ //treba doriesit osetrenie prvej suradnice - prvu povazujem za spravnu - resp spravna je aj 0.0.0 - osetrenie v pythone
		isOk = false;
		reftimes = ref * times;

		if ((aktlat == 0) && (aktlng == 0))
		{
				isOk = true;
				times = 1;
				return isOk;
		}

		if (((oldlat - reftimes) <= aktlat) && (aktlat <= (oldlat + reftimes)))
		{
				if (((oldlng - reftimes) <= aktlng) && (aktlng <= (oldlng + reftimes)))
				{
						isOk = true;
```

```
                    times = 1;
                }
                else
                {
                    times++;
                    if (times == 13)  //v pripade ze sa nacita odveci
prva suradnica, tak aby aspon po minute sa suradnica poslala
                    {
                        isOk = true;
                        times = 1;
                    }
                }
            }
            return isOk;
}
```

3.4.1.5 Watchdog časovač

Vzhľadom na typ a zameranie projektu je nutné sa stále zaoberať otázkou zvýšenej spoľahlivosti. Keďže Watchdog časovač (WDT) je jedným s prvých a základných mechanizmov na zvýšenie spoľahlivosti, rozhodli sme sa ho do projektu implementovať.

Jeho hlavná podstata je resetovať systém v prípade neočakávaného sa zacyklenia a v našom projekte je implementovaný na základe nasledovných predpokladov:

1. Celý firmvér sa skladá s dvoch hlavných častí: inicializácia a sľučka
2. Inicializácia sa vykoná len raz (nastavenie premenných, senzorov atď...)
3. Slučka sa opakuje stále dookola (gro programu: meranie a odosielanie údajov)
4. Spúšťanie slučky je podmienené HW časovačom na Arduine (každých 5 sekúnd sa vykoná funkcionalita sľučky)

Na začiatku spúšťania slučky sa nastaví príznak „wdt = true“. Následne prebiehajú všetky merania, zaznamenávania údajov, nastavovania premenných, výpisov a podobne (toto všetko trvá približne 1,2 sekundy)... Na konci slučky sa nastaví hodnota príznaku „wdt = false“.

Len čo HW časovač na Arduine znovu zaregistruje priebeh ďalších 5-tich sekúnd, ako prvé sa vykoná porovnanie príznaku: či „wdt = true“ ?, ak áno znamená to, že predchádzajúca slučka nebola korektne ukončená a je teda nutné celý program resetovať. V prípade, že „wdt =

false“, časovač povolí spustenie ďalšej hlavnej výpočtovej sučky. Pod odstavcom vidíme rozhodovaciu časť kódu pre WDT.

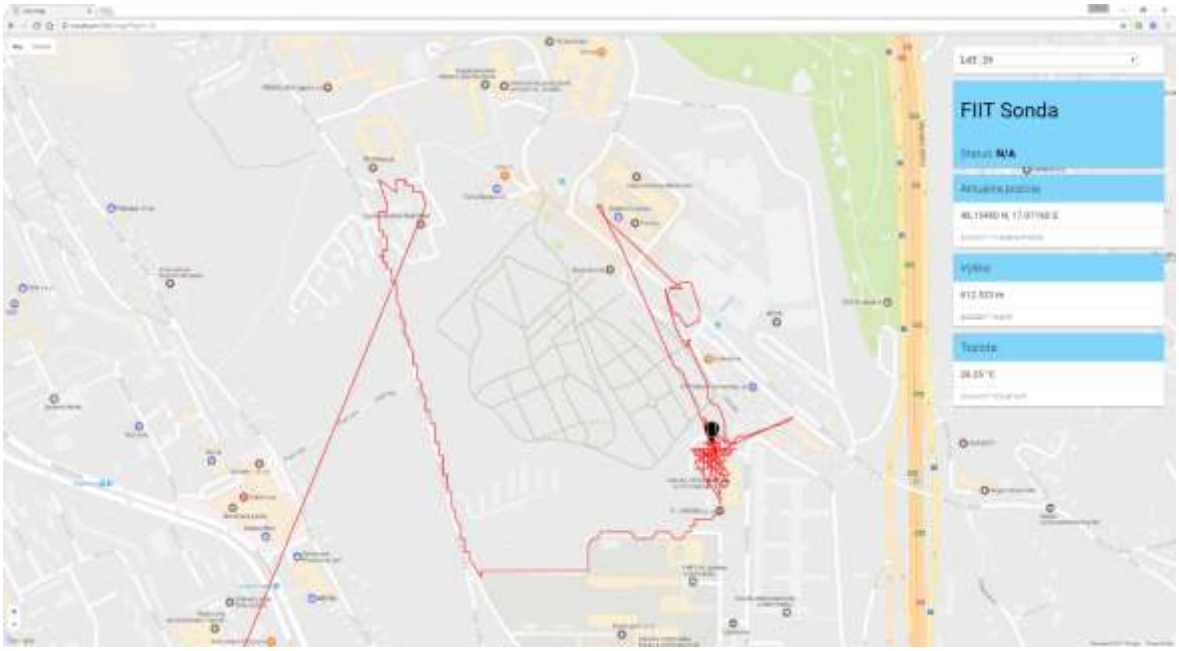
```
boolean wdt = false;   //definovanie premennej wdt

void(* resetFunc) (void) = 0; //funkcia resetu Arduina

void Time_function()
{
 if (wdt)                  //porovnanie korektného ukončenia predchádzajúcej slučky
 {
   resetFunc();//volanie funkcie reset
 }
 timer+=5;
}
```

3.4.1.6 Testovací dataset

Testovací dataset sme vytvárali 15.3.2017 o 15:30. Experimentu predchádzala nasledovná príprava: Na notebooku sme si rozbehali obraz serveru a databázy. Zariadenie sme prepojili s notebookom pomocou dostatočne dlhého USB kábla. Upravili všetky príslušné adresy (kam sa dáta odosielali, vzhľadom na localhost). Upevnili modul na batoh a išli na prechádzku do školy (kde sme otestovali odolnosť modulu na výpadky GPS signálu – kapitola 3.4.1.2 Testovanie straty GPS signálu).

Obrázok 3.21 Testovací tím (Michal Valíček, Jakub Findura, Ján Pánis, Maroš Frkáň – fotograf)

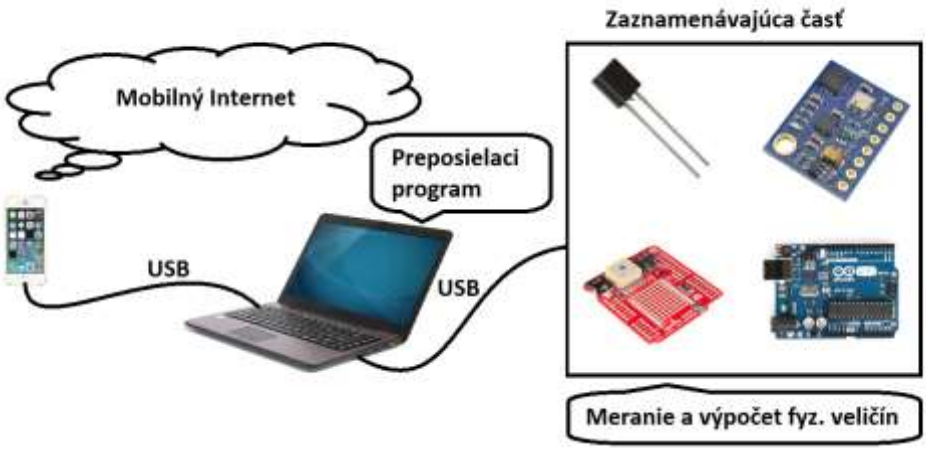
Test po prvý krát dopadol katastrofálne, pretože naše GPS nám generovalo len „nulové súradnice“, avšak na druhý pokus (po doladení firmvéru vzhľadom na zmenu platformy), nás milo prekvapil. Celú cestu sa nám podarilo úspešne zaznamenať aj na server aj na pamäťovú kartu (okrem iného aj časy [RTC a relatívny], teploty, tlak, počet satelitov a iné...). Túto trasu nám taktiež v reálnom čase vykresľovalo v prehliadači čím sme otestovali nie len modul, ale aj server, prenos údajov, databázu, websokety a veľa ďalšieho. Získané testovacie dáta sú zobrazené na mape na obrázku Obrázok 3.22 Testovacia dráha letu 15.3.2017



Obrázok 3.22 Testovacia dráha letu 15.3.2017

3.4.2 Preposielacia časť

V prvom prototype sme si model celého servisného modulu čiastočne zjednodušili, keďže sme vypustili komunikáciu cez rádiové spojenie. V skratke náš model je vyjadrený v obrázku pod odstavcom.



Obrázok 3.23 Zjednodušený model zaznamenávacej a preposielcej časti

Údaje sa namerajú a predspracujú na mikrokontoléry Arduino UNO. Následne sa pošlú na sériovú (USB) linku. Na tejto linke počúva počítač a len čo zachytí odoslané dáta, spracuje ich do vopred dohodnutého formátu JSON. Následne pomocou knižnice ‚requests' využitím metódy HTTP POST dáta odošleme v správnom formáte na server, kde sa následne spracujú, respektíve zobrazia na našej stránke v mapke.

Pod odstavcom sa nachádza časť programu v jazyku Python, ktorý sa nachádza na počítači a prijaté dáta spracováva a odosiela na server:

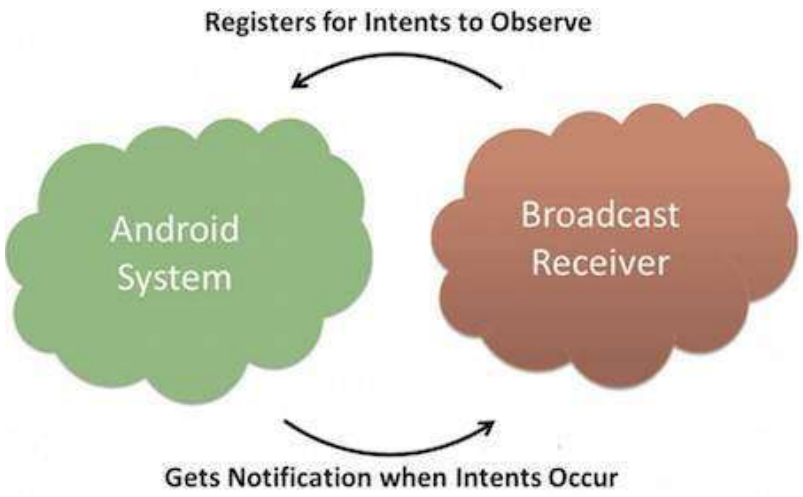
```
while 1: // nekonecny cyklus pocuvania
    try:
        line=(ser.readline().decode("utf8"))        // pocuvanie udajov v spravnom formate
        if (line != ""):
            print(line)        // vypis prijateho riadku
            line_parsed = line.split(",")        // parsovanie prijateho suboru
            update['type'] = "updateMsg"        // vyskladanie struktury vhodnej pre json
            update['data']['time'] = line_parsed[0]
            update['data']['location']['x'] = line_parsed[4]
            update['data']['location']['y'] = line_parsed[5]
            update['data']['location']['z'] = line_parsed[6]
            update['data']['temperature']['in'] = line_parsed[3]
            update['data']['temperature']['out'] = line_parsed[2]
            update['data']['isburst'] = line_parsed[7]
            json_data = json.dumps(update)     // parsovanie struktury do formatu JSON
            response = requests.post("http://posttestserver.com/post.php", json=json_data)  //
odosielanie na server
    except ser.SerialTimeoutException:
        print('Data could not be read')
    time.sleep(0.1)
```

3.4.3 Prijímanie SMS - problém

Počas analýzy sme identifikovali riziko súvisiace s rádiovým spojením servisného modulu s naším prijímačom vo forme antény pripojenej k notebooku. Takéto spojenie nie je úplne spoľahlivé a hrozí výpadok spojenia čo má za následok absenciu dát v ďalších softvérových moduloch v našom projekte. Dopad takéhoto rizika vieme redukovať aplikovaním redundantného spôsobu odosielania GPS súradníc zo servisného modulu určeného pre dohľadanie a sledovanie pohybu balóna v atmosfére. Tento spôsob zahŕňa pravidelné odosielanie SMS správ pomocou pridaného hardvérového GSM modulu do mikropočítača Arduino. Analýzou sme však narazili na problém prijatia SMS správy s GPS súradnicami a ich uloženie do databázy na serveri. Existujú viaceré spôsoby preposlania SMS správy do notebooku avšak nepodarilo sa nám nájsť vhodný softvér, ktorý by tieto SMS jednoduchým spôsob preposlal formou HTTP requestu na náš hlavný server pomocou API implementovaného na serveri. Rozhodli sme sa tento problém vyriešiť vytvorením vlastnej jednoduchej Android aplikácie, ktorá prijatú SMS prepošle formou HTTP requestu na server.

3.4.4 Prijatie SMS správy v Android aplikácií

Pre prijatie SMS správy v aplikácie sme použili Android API, konkrétne triedu BroadcastReceiver.



Obrázok 3.24 Príjanie SMS v android systéme pomocou broadcast príjmačov

```
public class SmsReceiver extends BroadcastReceiver {
    private Bundle bundle;
    private SmsMessage currentSMS;
    private String message;
    final String DELIMITER = ",";

    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) {
            bundle = intent.getExtras();
            if (bundle != null) {
                Object[] pdu_Objects = (Object[]) bundle.get("pdus");
                if (pdu_Objects != null) {
                    for (Object aObject : pdu_Objects) {
                        currentSMS = getIncomingMessage(aObject, bundle);
                        String senderNo = currentSMS.getDisplayOriginatingAddress();
                        message = currentSMS.getDisplayMessageBody();
```

Po prijatí SMS správy je nutné správu rozparsovať z csv formátu do formátu JSON. Pre vytvorenie JSON premennej sme použili json java knižnicu.

3.4.5 Odoslanie HTTP requestu v Android aplikácií

Pre odoslanie HTTP requestov sme použili knižnicu Volley. Táto sieťová knižnica pre Android aplikácie umožňuje odoslanie HTTP requestov jednoduchým spôsobom v čo najkratšom čase.

Výhody Volley knižnice

- Automatické plánovanie sieťových požiadaviek
- Viacero súčasných sieťových spojení
- Určenie priority pre požiadavky
- API prístup pre zrušenie odosielania
- Možnosť ošetrenia v prípade nedostupného servera

Volley je vhodnou knižnicou pre viaceré typy RPC (angl. Remote Procedure Call) operácií. Pri odosielaní je možné odosielať nielen reťazce znakov ale napr. aj obrázky a pre nás dôležité JSON objekty. Volley nie je vhodným riešením pri odosielaní veľkých súborov a pri veľkých prúdových operáciách. Jadro tejto knižnice je vyvíjané v slobodnom repozitári od spoločnosti Google.

Pre pridanie knižnice do projektového gradle súboru pridať závislosť na Volley zdrojové kódy.

```
dependencies {
   ...
   compile 'com.android.volley:volley:1.0.0'
}
```

V našom projekte stačí pomocou Volley knižnice odoslať pripravený JSON objekt na príslušné API na hlavnom serveri. Odpoveďou serveru vieme overiť správne prijatie JSON objektu na serveri. Použitím návrhového vzoru observer je používateľ oboznámený o návratovom kóde serveru formou Toast notifikácie.

```
StringRequest stringRequest = new StringRequest(Request.Method.POST, URL, new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
```

```
        Log.i("VOLLEY", response);
        ObservableObject.getInstance().updateValue(response);
    }
```

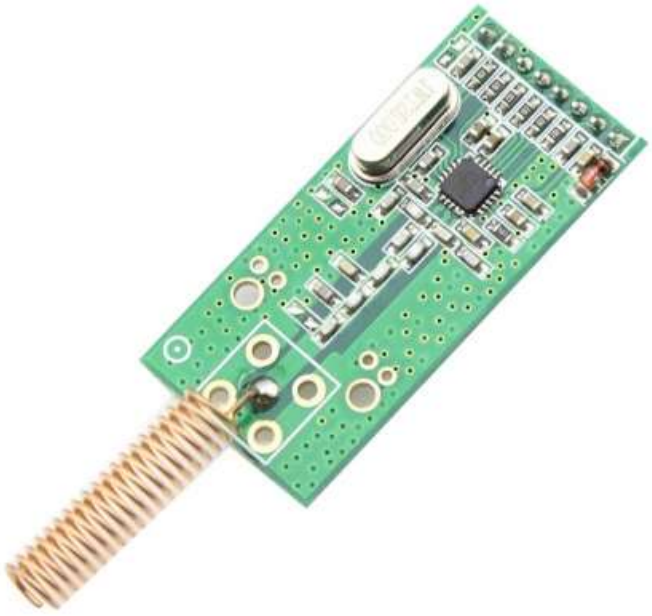
3.4.6 Majáčik na dohľadanie

Z dôvodu nedokonalej presnosti GPS súradníc, sme nútení vyriešiť alternatívne riešenie dohľadania balóna, resp. jeho presnejšie zameranie. Problém môže vzniknúť jednoducho napr. pri páde do hrachového poľa, kde aj pár metrov môže človeka pri hľadaní značne zmiasť. Práve preto v našom servisnom module implementujeme jednoduchý majáčik, ktorý pomocou smerovej antény bude jednoduchšie zamerať.

Pri riešení tejto problematiky vychádzame hlavne z dostupných aktuálnych riešení, vyskytujúcich sa najmä v rádioamatérskych orientačných behoch. Rádioamatérsky orientačný beh je pretekársky šport, ktorý kombinuje určovanie smeru pomocou smerovej antény so schopnosťami orientovať sa v teréne. Cieľom je nájsť postupne 5 rôznych rádiových vysielačov v čo najkratšom čase. Toto by sme chceli využiť v našom servisnom module. K riešeniu teda potrebujeme tak ako v spomenutom športe drobný vysielač, ktorý sa primontuje k servisnému modulu a rádio prijímač, pomocou ktorého náš modul dohľadáme [23].

V našom prvom riešení využívame RF transmitting module 433Mhz spolu s dvomi Arduinami.

Špecifikácia RF modulu [24]:



Obrázok 25 RF transceiver modulu CC1101 433Mhz

- Frekvencia: 433Mhz
- Modulácia: ASK
- Výstup príjmača: High - 1/2 Vcc, Low - 0.7v
- Vstup vysielača: 3-12V (vyššie napätie = lepší dosah)
- Dosah 40m v budove, 90m na otvorenom priestore

HW implementácia:

3.4.6.1 Anténa

Anténa Yagi-Uda, všeobecne známa ako anténa Yagi, je smerová anténa pozostávajúca z viacerých paralelných prvkov v jednej línii, zvyčajne polovičkových dipólov vyrobených z kovových tyčí. Yagi-Uda antény pozostávajú z jediného poháňaného prvku pripojeného k vysielaču alebo prijímaču pomocou prenosovej linky a dodatočných "parazitických prvkov", ktoré nie sú pripojené k vysielaču alebo prijímaču: takzvaný reflektor a jeden alebo viac riaditeľov. To bolo vynájdené v roku 1926 Shintaro Uda Tohoku Imperial University, Japonsko, a (s menšou úlohou jeho kolega) Hidetsugu Yagi.

Návrh antény s využitím VK5DJ Yagi kalkulačkou:

- Yagi konštrukčná frekvencia = 433,00 MHz
- Vlnová dĺžka = 692 m
- Parazitické prvky pripevnené k nekovovému alebo oddelené od výložníka

- Sklopený dipól namontovaný rovnako ako riadiace jednotky a reflektor
- Riaditeľ / reflektor diam = 3 mm
- Radiátor diam = 3 mm

Reflector

Dĺžka 337,8 mm v polohe ramena = 30 mm (IT = 154,0 mm)

Radiator

Jednoduchý dipól s hrotom 326,6 mm na špičke, rozmiestnený 138 mm od reflektora pri výložníku 168 mm (IT = 148,5 mm). Sklopený dipól 333,1 mm špička na špičke, rozmiestnený 138 mm od reflektora na výložníku 168 mm (IT = 151,5 mm)

Riaditelia

Dir	Dĺžka vzdialenosti Zisk (Mm)	Zisk (mm)	Vzdialenosti (mm)	Pozícia výložníka (mm)	Vývoj zisku (dBd)	IT (dBi)
1	305,4	51,9	220,4	137,5	4,8	6,9
2	302,2	124,6	345,0	136,0	6,5	8,6
3	299,3	148,9	493,9	134,5	7,8	9,9

Komentáre

Skratka "IT" znamená "Vložiť do" je konštrukčná vzdialenosť od špičky prvku k okraju ramena pre
Cez cez montáž boomu.

Medzery sa merajú uprostred od predchádzajúceho prvku Tolerancia dĺžok prvkov je +/- 2 mm

Poloha ramena je bod upevnenia pre každý prvok meraný od zadnej časti ramena a zahŕňa 30 mm

Previs. previs. Celková dĺžka ramena je 524 mm vrátane dvoch previsov 30 mm

Odhad svetla lúča 3dB je 65 stupňov

Polovičná vlna 4: 1 balun používa 0,75 rýchlostný faktor RG-6 (penový PE) a je dlhá 260 mm plus vedenie

Zložená dipolová konštrukcia

Merania sa odoberajú zvnútra ohybov

Skosená dĺžka dipólu meraná špičkou na špičke = 333 mm

Celková dĺžka tyče = 696 mm

Stred tyče = 348 mm

Vzdialenosť BC = CD = 149 mm

Vzdialenosť HI = GF = 144 mm

Vzdialenosť HA = GE = 172 mm

Vzdialenosť HB = GD = 199 mm

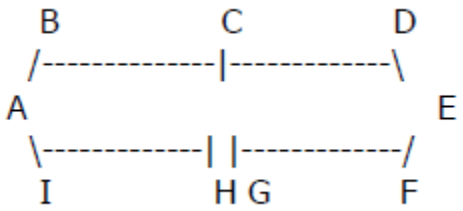
Vzdialenosť HC = GC = 348 mm

Medzera pri HG = 10 mm

Priemer ohybu BI = DF = 35 mm

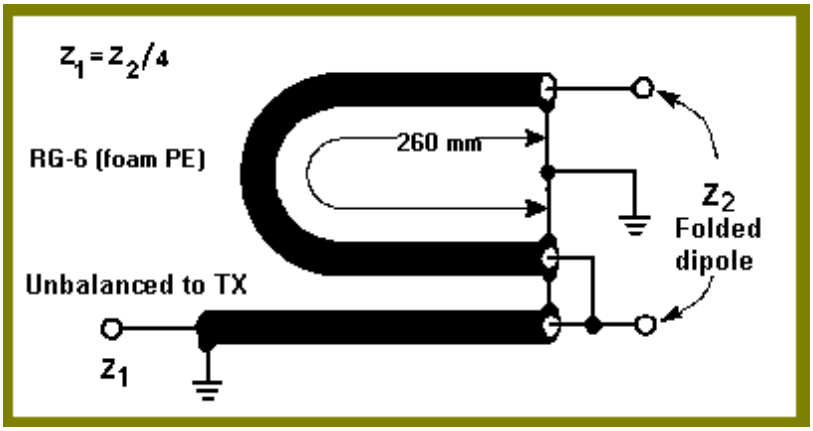
Ak je zložený dipól považovaný za rovinnú rovinu (pozri ARRL Antenna Handbook), jeho rezonančná frekvencia je menšia ako

Rozsah plochého algoritmu je 10: 1

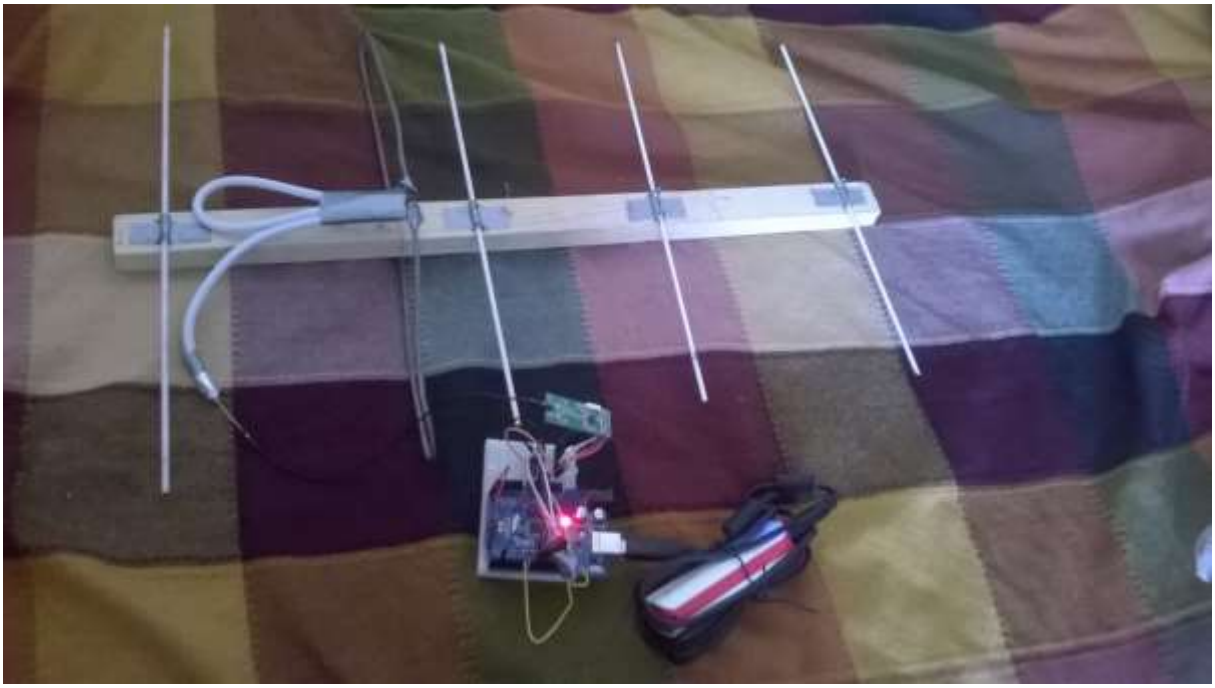


Obrázok 26 Konštrukcia zloženého dipólu

Konštrukcia Balunu:



Obrázok 27 Konštrukcia Balunu



Obrázok 28 Fotografia zrealizovaného dohľadávacieho systému

3.4.7 Spracúvajúca časť

Server je postavený na frameworku Flask, ktorý umožňuje jednoducho vytvárať webové aplikácie. Jednotlivé súbory sú rozdelené v štruktúre podľa pravidiel frameworku. Vrstvy podľa návrhu systému sú tiež v jednotlivých adresároch.

- /balon
  - /templates – HTML súbory, ktoré sa zobrazujú na webovej stránke
  - /static – statické súbory, ktoré sa týkajú webovej stránky ako štýly, JavaScript skripty a knižnice
  - /controllers – vrstva riadenia
  - /service – vrstva aplikačnej logiky
  - /database – vrstva databázy

- /models – modely pre databázu
- main.py – hlavný súbor aplikácie
- config.py – konfiguračný súbor

Pomocou anotácií je možné jednotlivé funkcie mapovať na požadované URL adresy, pri ktorých sa vykonajú.

```
@app.route('/map')
def balloonDashboard():
...
@app.route('/admin/flight/<int:flight_id>/detail')
def flight_detail(flight_id):
...
@app.route('/api/flight/<int:flight_number>/telemetry', methods=['POST'])
def api_telemetry(flight_number):
```

Server prijíma a spracúva dáta, ktoré sú zo sondy posielané viacerými kanálmi, v našom prípade rádiové spojenie a GSM sieť. Aplikácie, ktoré tieto dáta zo servisného modulu prijímajú ich následne posielaju na webový server cez protokol HTTP. Definované sú dva typy správ, ktoré je možné odoslať na rozdielne URL.

Prvou je priebežné odosielanie parametrov počas letu, ktoré sú následne použité na vykresľovanie trasy letu a ďalších grafov v reálnom čase počas letu.

```
/api/flight/<int:flight_number>/telemetry
```

Druhá správa obsahuje udalosť, ktorá sa počas letu vyskytne. Jedná sa o štart letu, roztrhnutie balóna, a pristátie.

```
/api/flight/<int:flight_number>/event/<string:event>
```

HTTP požiadavka obsahuje všetky potrebné dáta vo formáte JSON. Podstatnou časťou sú jednotlivé hodnoty veličín, ktoré boli namerané.

Server umožňuje vytváranie nových letov, ku ktorému sa následnie viažu odosielané údaje. Aby nebolo možné voľne pridávať údaje k danému letu, je vytvorená zjednodušená autentizácia v podobe hash-u, ktorý musí každá správa obsahovať. Tento hash je vytvorený pri vytvorení letu. Po overení sú údaje uložené do databázy servera a zobrazené na webovej stránke.

3.4.7.1 Databázová vrstva

Databázová vrstva je najnižšia časť nášho systému, ktorá komunikuje s databázou. V databázovej vrstve je použitý plugin Flask SQLAlchemy, ktorý poskytuje možnosť ukladať dáta do SQLite databázy za využitia vzoru Active Record. Táto vrstva zjednodušuje prístup

k dátam a poskytuje funkcie pre jednoduché získavanie dát ako posledné parametre alebo filtrovanie parametrov podľa typu.

```
def saveParameter(parameter):
    db.session.add(parameter)
    db.session.commit()
    return parameter.id

def getParametersByFlight(flight_id)

def getParametersByKeyByFlight(key, value, flight_id)

def getParameterLastByFlight(key, value, flight_id)

def getParameterFirstByFlight(key, value, flight_id)
```

Pre jednotlivé dáta sú vytvorené modely, ktoré využívajú spomínaný Active Record a cez ktoré je možné jednoduché ukladanie a získavanie dát v podobe objektov bez ďalšej transformácie a vytvárania objektov pre použitie v aplikácii.

```
class Parameter(db.Model):
    id = db.Column(db.Integer, primary_key=True)

    type = db.Column(db.String(20))
    source = db.Column(db.String(20))

    valid = db.Column(db.Boolean)
    validated = db.Column(db.Boolean)

    time_received = db.Column(db.DateTime)
    time_created = db.Column(db.DateTime)

    flight_id = db.Column(db.Integer, db.ForeignKey('flight.id'))
    values = db.relationship('Value', backref="parameter", lazy='dynamic'
```

3.4.8 Zobrazovacia časť

Primárnou časťou zobrazovania je naša webová stránka, na ktorej sa nachádza mapa s trasou letu balóna. Na stránke sa zobrazujú údaje uložené v databáze servera. V rámci serverovej časti sme implementovali obsluhu požiadaviek klientskej časti na zobrazenie webovej stránky a odosielanie aktuálnych dát cez WebSocket. Na komunikáciu slúži knižnica Flask-Socket.IO, ktorá zabezpečuje spojenie.

Implementovali sme klientsku časť webovej stránky, ktorá zobrazuje mapu a údaje zo sondy, ktoré sa aktualizujú v reálnom čase. Túto aktualizáciu zabezpečuje na klientskej strane JavaScript a protokol WebSocket, cez ktorý server posiela klientom aktuálne dáta. Toto spojenie

zabezpečuje na klientskej strane knižnica Socket.IO. Po prijatí klient, tieto dáta spracuje a na stránke vykreslí aktuálnu trasu letu balóna a aktualizované hodnoty týkajúce sa letu (poloha balóna, výška, okolitá teplota).

```
socket.on('baloon_update', function(data) {
	var message = JSON.parse(data);
	switch(message.type) {
		case "balloonEvent":
			handleBalloonEvent(message);
			break;
		case "balloonPath":
			handlePathUpdate(message);
			break;
		case "balloonTelemetry":
			handleTelemetryUpdate(message);
			break;
	}
});
```

Dáta medzi serverom a klientom sa prenášajú vo formáte JSON. Do tohto formátu je jednoduché prevádzať dáta na strane servera (Python) ako aj na strane klienta (JavaScript).

```
{
	"type": "balloonEvent",
	"created": 1477866660,
	"data": {
		"type": "currentPosition",
		"eventTime": 1477867645,
		"point": {
			"time": 1477867645, // Optional
			"lat": 49.548258,
			"lng": 19.162854
		},
		"info": {
			"title": "Reached 10000 m",
			"text": "We have reached altitude of 10000 meters. YAY!!!"
		}
	}
}
```

Na zobrazenie mapy využívame mapy od spoločnosti Google, ktorý poskytuje taktiež rozsiahle API, ktorým je možné mapy upraviť podľa potreby. S dostupného API využívame objekty pre značku a čiaru, ktoré na mape znázorňujú trasu balóna a udalosti ako vzlet, roztrhnutie balóna alebo pristátie.

3.5 Propagácia na fakulte

Počas šiesteho šprintu sme sa zamerali aj na propagáciu nášho projektu. Kontaktovali sme kompetentnú osobu na našej fakulte, ktorá zodpovedá za zobrazovanie informácií na školských televízoroch. Aby ale aplikácia mohla byť zobrazovaná, bolo potrebné ju prerobiť tak, aby dokázala „prežiť“ dočasný výpadok internetu. Na tento problém sme našli viacero riešení. Buď webovú stránku zobrazujúcu aplikáciu necháme v pravidelných intervaloch obnovovať, alebo túto funkcionalitu implementujeme priamo v aplikácii. Druhé riešenie sa nám zdalo lepšie, nakoľko pri prvom riešení by bolo potrebné zaobstarať rozšírenie tretej strany.

3.6 Propagácia na Facebooku

3.6.1 Prehľad

Stále viac a viac podnikov a spoločností si v dnešnej dobe vytvára vlastné stránky na Facebooku. Takéto stránky slúžia najmä na propagáciu svojich výrobkov a služieb. Taktiež môžu slúžiť na komunikáciu so zákazníkmi alebo na podávanie čerstvých informácií o novinkách týkajúcich sa fungovania spoločnosti alebo inštitúcie. Ak požadujeme vždy aktuálny obsah, veľké množstvo webových portálov a aplikácií sa spolieha na automatické odosielanie obsahu na ich príslušné Facebook stránky. Každý, kto niekedy hral hru na Facebooku alebo dokonca použil mobilnú aplikáciu Facebooku sa už stretol s používateľskými prístupovými tokenmi (angl. user-access tokens). Takéto tokeny umožňujú aplikáciám odosielať obsah pod používateľským účtom daného používateľa. Ale ako sa takýto obsah posiela pomocou aplikácie na konkrétnu facebook stránku? Na tento problém sa používajú prístupové tokeny.

3.6.2 Prístupové tokeny

Prístupové tokeny stránky umožňujú aplikácií vykonávať rôzne akcie na stránke ako keby ju vykonávala samotná stránka a nie ako bežný používateľ facebooku. Správcovia stránky majú možnosť zmeniť kontext účtu pod ktorým pridávajú obsah na vlastnú stránku. Pri následnom prezeraní stránky je daný obsah ktorý bol vytvorený (napr. status, komentár, pripomienka, udalosť...) odoslaný pod menom stránky a nie pod „osobným“ používateľským účtom správcu stránky. Prístupové tokeny stránky umožňujú ľubovoľnej aplikácií vytvárať obsah na stránke.

Tieto tokeny taktiež umožňujú aplikáciám publikovanie obsahu na Facebook stránku automaticky. To znamená že používateľ nemusí nijak zasahovať do procesu publikovania obsahu. Takúto automatizáciu dosiahneme využitím Facebook Graph API (angl. application programming interface). V prípade odosielania obsahu s tokenom pre používateľský účet a nie prístupovým tokenom stránky bude obsah odoslaný pod používateľským účtom používateľa a

taktiež vzniká riziko úniku informácií z takéhoto osobného používateľského účtu. Takýto únik môže nastať v prípade ak ma aplikácia nezabezpečený prístup do databázy. Uložením prístupových tokenov sa znižuje pravdepodobnosť, že nastane takáto situácia. Ak by aj takáto situácia nastlala tokeny sú generované vždy len pre činnosť na stránke a na obmedzenú dobu platnosti.

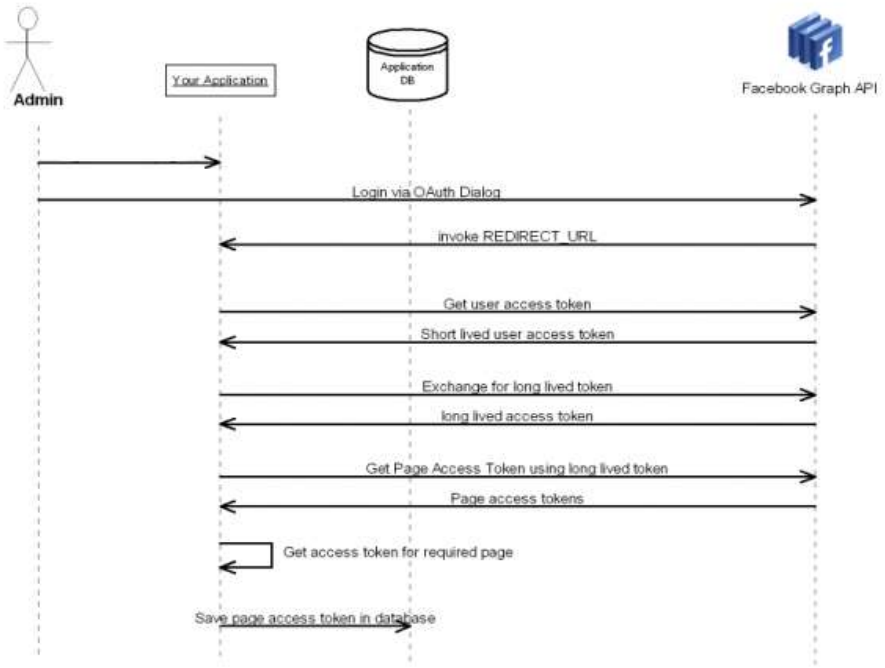
3.6.3 Získanie prístupových Facebook tokenov

Získanie Facebook prístupových tokenov sa skladá zo 4 krokov, ktoré musia byť vykonané v takomto poradí:

1. Zaregistrovanie Facebook aplikácie.
2. Získanie používateľského tokenu Facebook účtu, ktorý ma vhodné práva pre editovanie povolení pre Facebook stránku.
3. Výmena krátko žijúceho tokenu (angl. short lived token) za dlho žijúci token (angl.long lived token).
4. Použitie dlho žijúceho tokenu na získanie prístupového tokenu stránky.

3.6.4 Integrácia v aplikácií

Existuje viacero spôsobov ako implementovať tieto 4 kroky pre získanie prístupového tokenu stránky. Pre začiatok je nutné aby mala aplikácia prístup k Facebook aplikačnému ID (angl. Facebook application ID), aplikačnému tajomstvu (angl. application secret) a ID stránky pre ktorú požadujeme token.



Obrázok 3.29 Diagram znázorňujúci interakciu medzi adminom, aplikáciou a Facebook Graph API.

Názov grafové API (angl. graph API) je odvodený z idei sociálnych grafov. Takéto grafy reprezentujú rôzne informácie na Facebooku podľa toho z čoho sú zložené. Sú to:

- Uzly (angl. nodes) - jednoducho povedané sú to časti Facebooku ako napr. používateľ, fotka, stránka, komentár
- Hrany (angl. edges) - sú to spojenia medzi uzlami, môžu to byť napr. fotky na stránke alebo komentáre k fotke
- Polia (angl. fields) - informácie o uzloch, napr. narodeniny používateľa, názov stránky

Celé grafové API je založené na HTTP, takže s týmto rozhraním dokážeme pracovať v ktoromkoľvek programovacom jazyku pre ktorý je dostupná HTTP knižnica (napr. knižnice cURL, urllib...). Väčšina požiadaviek na Facebook Graph API vyžaduje použitie prístupových tokenov od aplikácie.

3.6.5 Grafový API prieskumník

Najjednoduchší spôsob ako pochopiť funkcionalitu a otestovať Facebook Graph API je použiť Graph API prieskumníka (angl. explorer). Tento nízkoúrovňový nástroj pomáha pri vytváraní požiadaviek, či už na zobrazenie, pridanie alebo odstránenie dát z Facebook databáz.

Obrázok 3.30 Grafový API priezkumník

3.6.6 Fanpage

Na Facebooku sme vytvorili fanpage, na ktorý sme priebežne pridávali aktuálne informácie o priebehu projektu. Fanpage mala dokopy dvoch fanúšikov, člena tímu Jána a jeho priateľku.

3.7 Propagácia na Twitteri

3.7.1 Prehľad

Twitter je sociálna platforma pre komunikáciu medzi ľudmi pomocou odosielania krátkych textových správ. Tieto správy sa nazývajú tweety (angl. tweets) a sú limitované na

maximálnu dĺžku 140 znakov. Táto vlastnosť umožňuje efektívne zobrazenie tweetoch aj na mobilných zariadeniach, ktoré nemusia byť považované chytré (angl. smartphones). Taktiež tweety môžu byť odosielané z webových aplikácií, desktopových aplikácií a iných mobilných zariadení. Používatelia väčšinou odosielajú správy o vlastných aktivitách ako napr. kde sa nachádzajú, čo robia, nad čím práve premýšľajú, čo pozerajú atď. V tweetoch je taktiež možné odoslať odkazy na stránky, obrázky alebo iné tweety. Používatelia Twitteru môžu sledovať obsah iných používateľov a tým si zobrazovať ich obsah na svojej Twitter stránke.

3.7.2 Význam Twitteru

Twitter sa v súčasnosti rýchlo rozrástol na efektívnu komunikáciu ako komunikovať s ostatnými a podávať najčerstvejšie informácie v čo najkratšom čase. Je jasné že Twitter účty niesú určené len pre jednotlivcov ale aj pre organizácie, firmy, skupiny atď. Efektívnosť a dosiahnutie používateľov je pre organizáciu primárnym cieľom. Čím väčší počet používateľov sleduje Twitter účet organizácie tým viac tweetov sa dostáva k používateľom.

Twitter však nie je len o odosielaní tweetov. Monitorovaním zmienok a haštagov (angl. hashtag) môže organizácia zistiť, čo ľudia hovoria o svojich skúsenostiach súvisiacich s organizáciou. Reakciou na ľudí podľa ich Twitter používateľských mien docielime účinný spôsob, ako zaujať divákov a vyvolať v nich pocit, že ich názor je akceptovaný. Často sa nájdu ľudia ktorých obsah môže byť považovaný nevhodný a preto je nutné podať sťažnosť príslušnému správcovskému oddeleniu Twitteru.

3.7.3 Prístup aplikácie pomocou REST API

Resttovské rozhranie poskytuje programový prístup k čítaniu a zapisovaniu dát k službe Twitter. Vytvorenie nového Twitter účtu, čítanie používateľského profilu a dát sledujúcich ľudí, a mnohé ďalšie. REST API identifikuje Twitter aplikácie a používateľa pomocou OAuth. Odpovede zo servera sú vo formáte JSON. JSON poskytuje štandard pre výmenu dát ktorý sa skladá z atribútov typu kľúč – hodnota.

Pre prístup k Twitter API je, podobne ako ku Facebook Graph API, potrebná autentifikácia. Aplikácie musia autorizovať všetky požiadavky pomocou protokolu OAuth 1.0a prípade je možné použiť autentifikáciu len pre aplikáciu (angl. Application only authentication). Autorizácia dovoľuje Twitteru chrániť dáta používateľov a zabraňuje podozrivému správaniu. Taktiež autorizácia poskytuje informácie pre vývojárske tímu Twitteru pre lepšiu analýzu používania API vývojarmi. Použitím týchto informácií vie Twitter lepšie optimalizovať ich API a rozširovať funkcie samotnej platformy.

Pre vykonávanie oprávnených volaní na Twitter API, aplikácia musí najprv získať prístupový token OAuth s menom používateľa služby Twitter. OAuth je autorizačný protokol, ktorý umožňuje používateľom schválenie požiadavky aplikácie pre žiadosť konať v ich mene, bez nutnosti zdieľať svoje heslo. Spôsob, akým aplikácie získavajú tokeny a pristupujú k API bude závisieť na riešenom prípade použitia aplikáciou. Najčastejšie prípady použitia sú znázornené v nasledujúcej tabuľke.

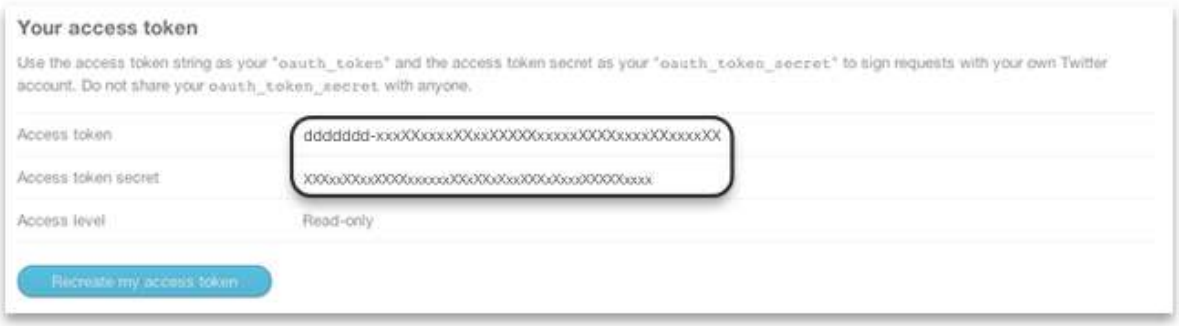
Prípad použitia	Názov prístupového rozhrania
Chceme poskytnúť tlačidlo "Prihlásiť pomocou Twitter účtu" na našej webovej stránke...	Sign in with Twitter
Chceme čítať a posielať Twitter dáta pod menom používateľov na našej webovej stránke...	3-legged OAuth
Máme mobilnú, desktopovú alebo vnorenú aplikáciu ktorá požaduje prístup k webovému prehliadaču...	PIN-based OAuth
Požadujeme prístup k API z nášho vlastného účtu...	Tokens from dev.twitter.com
Potrebujeme prístup k výmene používateľských mien/hesiel a naša aplikácia bola schválená pre použitie procesom xAuth...	xAuth
Poskytujeme API ktoré posiela dáta pod menami Twitter používateľov...	OAuth Echo
Chceme posielať autorizované požiadavky pod menom samotnej aplikácie...	Application-only authentication

3.7.4 Vytvorenie aplikácie pre Twitter API

Vytvorenie novej aplikácie je možné na stránke apps.twitter.com. Po vytvorení prístupových tokenov a tokenov tajomstiev je možný prístup k Twitter API. Tokeny poskytované Twitterom sú generované ako tokeny bez exspirácie. Token sa však môže stať neplatným ak používateľ explicitne odmietne aplikáciu vo svojich nastaveniach prípadne ak ju Twitter administrátor označí ako suspendovanú z aplikácie. V prípade že je suspendovaná aplikácia, správca aplikácie bude oboznámený poznámkou na svojej aplikačnej správcovskej stránke.

Pri návrhu aplikácie je vhodné myslieť na situáciu, ktorá môže nastať v prípade, že sa prístupový token stane neprístupný. Takáto situácia môže nastať kedykoľvek a používateľ je nútený sa znovu autorizovať. Správne riešenie tohto problému značne odbremeňuje používateľa a zvyšuje používateľský komfort.

Mnoho používateľov verí aplikácii ktorá číta informácie o používateľovi, ale nie vždy títo používatelia nevyhnutne súhlasia s odsúhlasením práv pre posielanie nových statusov v ich mene. Aktualizácia informácií cez Twitter API - či už je to meno, pozícia alebo pridanie nového tweetu - vyžaduje odoslanie HTTP POST požiadavky. Každá metóda API, ktorá vyžaduje HTTP POST je považovaná za spôsob zápisu a vyžaduje prístup k čítaniu a zápisu.



Obrázok 3.31 Generovanie prístupových tokenov na dev.twitter.com

V prípade ak proces OAuth, znie nad rámec integrácie aplikačného prepojenia pri návrhu aplikácie, je možné využitie webových zámerov (angl. web intents), ktoré nepotrebujú používať prístupové tokeny pre interakciu s Twitter API.

3.7.5 Limitovanie prístupu k Twitter API

Limitovanie množstva volaní Twitter API je primárne kontrolované pre jednotlivých používateľov aplikácie. Špecifickejšie sa jedná o limitovanie pre používateľský token. Znamená to, že ak volaná metóda dovoľuje 15 volaní pre používateľa v určitom časovom okne tak pre príslušný prístupový token je dovolených rovnako 15 volaní. Pri používaní autentizácie určenej len pre aplikáciu, limity pre volania API sa tykajú všetkých volaní v aplikácii spoločne. Toto limitovanie je považované ako separátne vzhľadom na volania používateľov aplikácie.

Limity pre volania Twitter API sú nastavené na znovuobnovenie po 15 minútových intervaloch. Všetky volania vyžadujú autentizáciu takže neexistuje koncept neautorizovaných volaní a limitov. Volania, ktoré využívajú GET požiadavky sú limitovane na 15 volaní každých 15 minút, pripadne 180 volaní každých 180 minút.

3.7.6 Knižnice pre prístup k Twitter API

Dostupná knižnica Python Oauth2 zabezpečuje implementačne náročnú časť podpisovania požiadaviek OAuth2 procesom. V nasledujúcom zdrojovom kóde je znázornená hlavná časť napojenia na Twitter API.

```
def oauth_req(url, key, secret, http_method="GET", post_body="",
http_headers=None):
     consumer = oauth2.Consumer(key=CONSUMER_KEY,
     secret=CONSUMER_SECRET)
     token = oauth2.Token(key=key, secret=secret)
     client = oauth2.Client(consumer, token)
     resp, content = client.request( url, method=http_method,
     body=post_body, headers=http_headers )
     return content

home_timeline = oauth_req(
'https://api.twitter.com/1.1/statuses/home_timeline.json', 'abcdefg',
'hijklmnop' )
```

3.8 Zobrazovanie zaujímavých informácií o stave balónu

Pre zatraktívnenie používateľského zážitku na webovej stránke s mapou aktuálnej pozície balónu sme sa rozhodli informovať používateľa o známych pripadne zaujímavých výškových míľnikoch, ktoré balón prekoná pri stúpaní. Myslime si, že priebežným informovaním používateľa zaujímavými správami prilákame väčšie množstvo používateľov na stránku a zvýšime čas strávený používateľmi na stránke.

3.8.1 Princíp a myšlienka

Po pripojení používateľa na web stránku s mapou informujúcou o stave balónu, sa okrem ostatných knižníc sa pre používateľa načíta aj Javascript knižnica, ktorá bude zobrazovať vyskakovacie notifikácie so zaujímavou informáciou. Táto notifikácia by sa nemala na webe zobraziť na dobu dlhšiu ako radovo pár sekúnd. Takisto je dôležité správne umiestnenie notifikácie tak aby nezakrývala ostatné informácie na stránke. Informácie kedy bude notifikácia zobrazená bude riadiť samostatný modul na webovom serveri. Modul po spustení serveru načíta dáta s informáciami, ktoré uloží do premennej aby s nimi mohol následne pracovať. Pre jednoduchosť pridávania nových informácií sme sa rozhodli použiť textový súbor vo formáte CSV.

Odosielanie dát pre Javascript knižnicu bude zabezpečovať komunikácia pomocou socketov. Po prijatí dát z balóna sa oznámi aktualizácia pre modul použitým návrhovým vzorom observer. Po oznámení sa skontroluje, ktorá zaujímavá správa je nasledujúca pre odoslanie a či

balón prekročil výšku súvisiacu s touto informáciou. Ak áno správa je odoslaná socketom do prehliadača používateľa, ktorý pomocou použitej Javascript knižnice notifikáciu zobrazí.

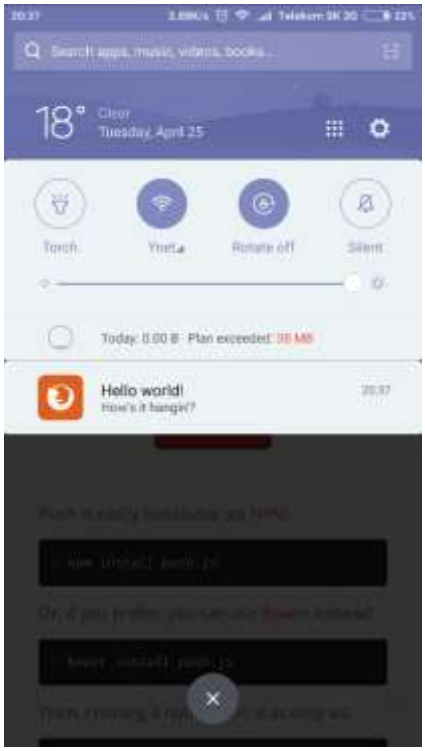
3.8.2 Notifikačné knižnice

Existuje veľké množstvo voľne dostupných knižníc ktoré môžeme použiť. Takmer všetky knižnice je možné ľahko pridať do webových stránok. Ďalšou výhodou je ich pamäťová veľkosť, ktorá zriedka presiahne hranicu 2-3kB.

3.8.2.1 Push.js

Push.js je veľmi často používaná knižnica pre zobrazovanie notifikácií na strane používateľa. Notifikácie sú podporované na všetkých moderných prehliadačoch. Veľkou výhodou tejto knižnice je fakt, že notifikácie sa zobrazujú v notifikačných lištách v prípade navštívenia takejto stránky z telefónu. Pri zobrazovaní na desktopových zariadeniach je možné k textu informácie pridať vlastný obrázok alebo nastaviť správanie po kliknutí myšou na notifikáciu.

Obrázok 32 Push.js notifikácia vo webovom prehliadači na desktopovom zariadení.



Obrázok 3.33Push.js notifikácia zobrazená v notifikačnej lište na mobilnom zariadení.

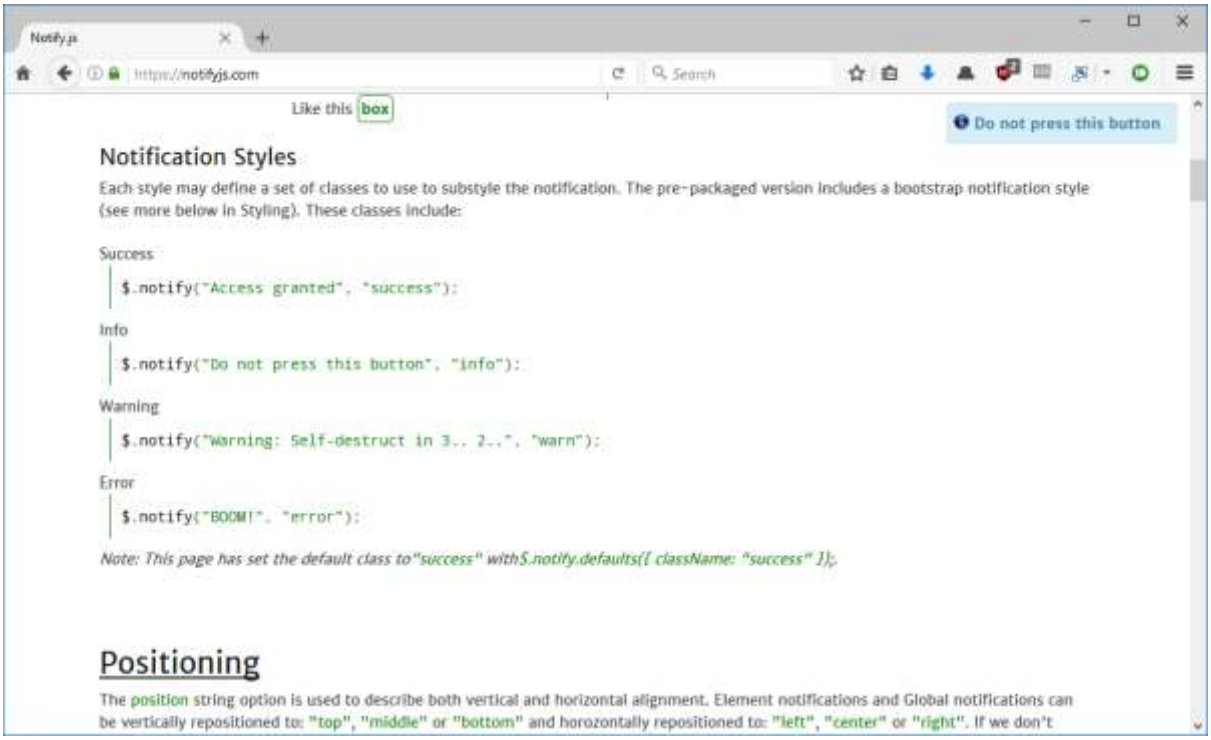
3.8.2.2 Notify.js

Notify.js je jQuery plugin, ktorý poskytuje jednoduché a plne nastaviteľné zobrazovanie notifikácií. Tento plugin umožňuje zobrazenie notifikácií pri ľubovoľnom HTML elemente na stránke. Plugin takisto obsahuje viacero preddefinovaných štýlov, ktoré je možné použiť (napr. informačné , výstražné, chybné notifikácie atď.). Okrem preddefinovaných štýlov je možné vytvoriť vlastný CSS štýl. Nevýhodou tejto knižnice je však pomerne zastaralý dizajn preddefinovaných štýlov.

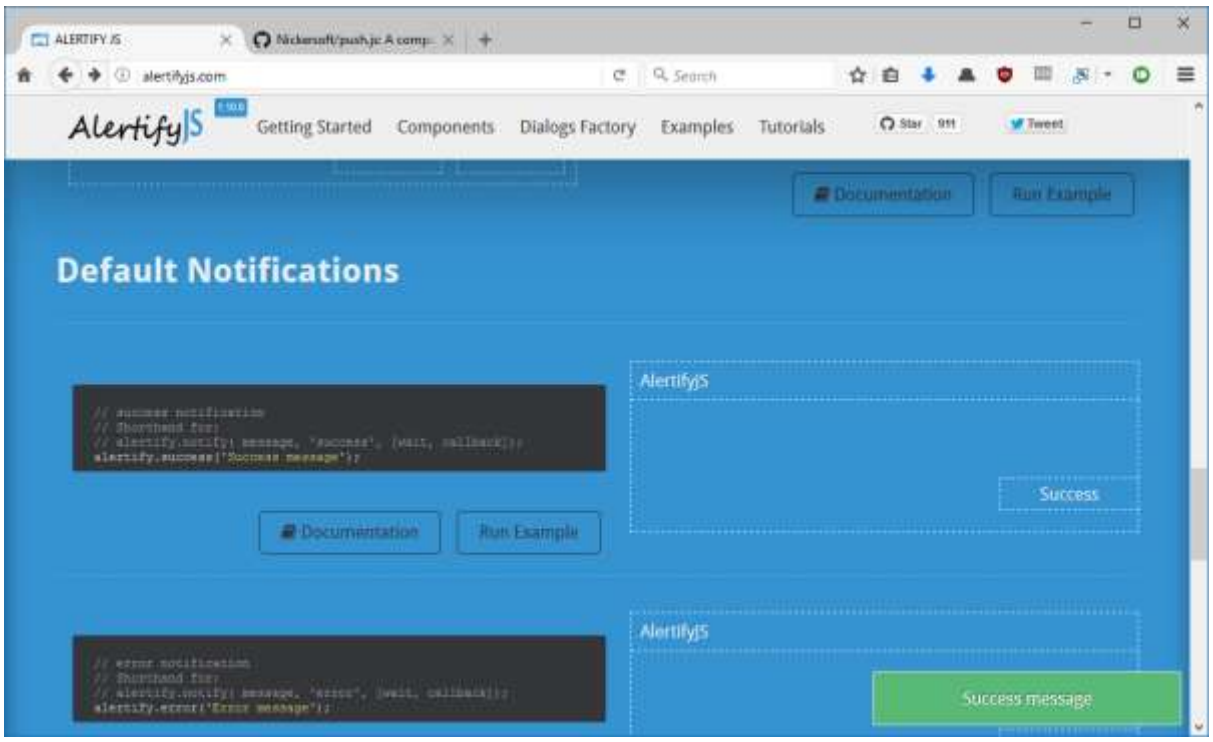
3.8.2.3 AleryifyJS

AleryifyJS je Javascript framework pre vývoj dialógových okien a notifikácií, ktoré používajú moderný dizajn. Zobrazené notifikácie a okná sú responzívne ich dizajn sa preto prispôsobí zariadeniu na ktorom sú zobrazované.

AlertifyJS poskytuje veľké množstvo modulov pre vytváranie vlastných dialogových okien a notifikácií. Výhodou tohto nástroja je aj podpora i18n lokalizácie v prípade použítia viacerých jazykov v notifikáciach.

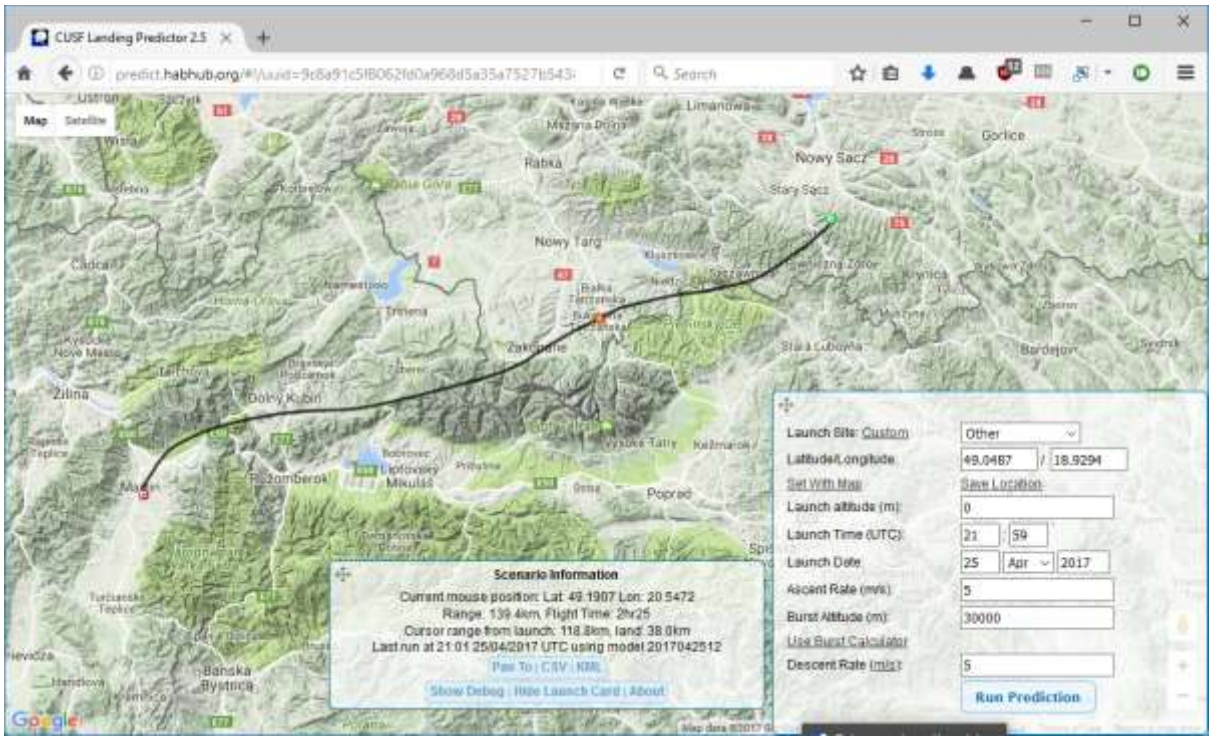


Obrázok 34 Zobrazenie notifikácie pomocou Notify.js (pravý horný roh)



Obrázok 35 Zobrazenie úspešnej správy pomocou AlertifyJS (dole vpravo).

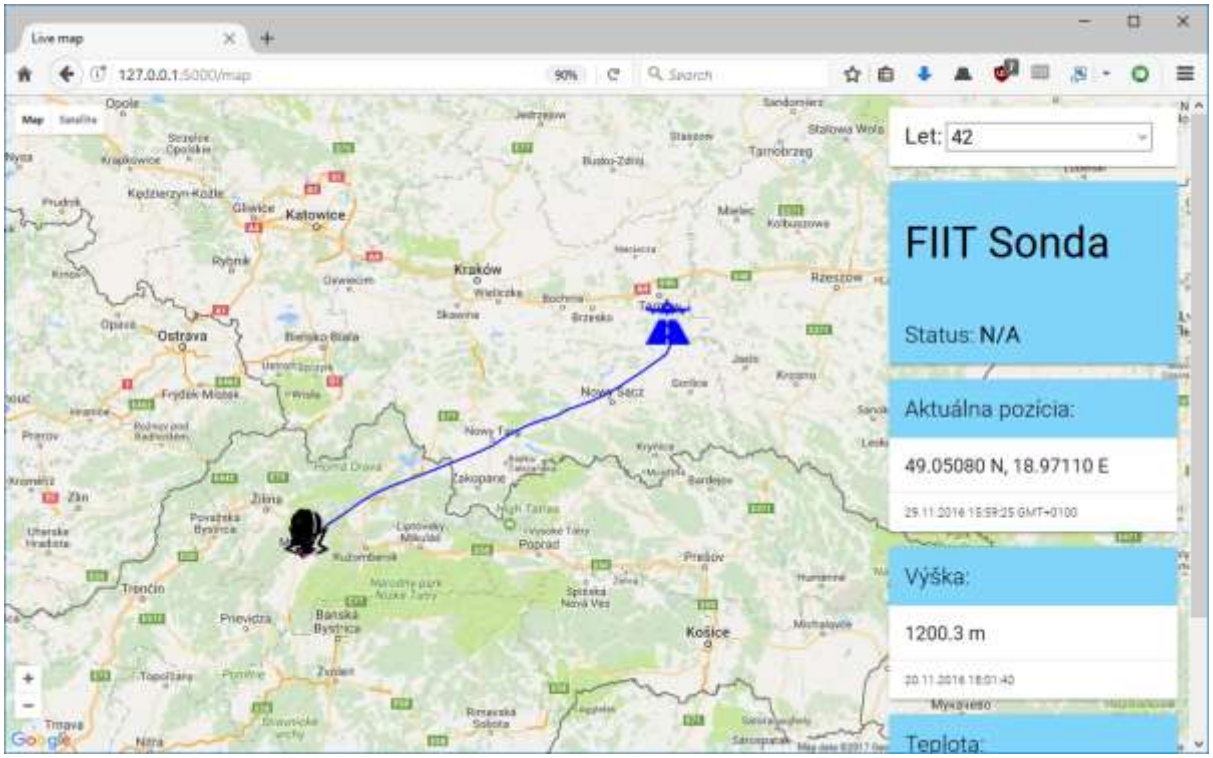
3.9 Zobrazovanie predikovanej trasy balónu



Obrázok 36 Ukážka z anglického predikčného servera http://predict.habhub.org.

Jedným z ďalších zaujímavých informácií, ktoré sú vhodné na zobrazenie na webovej stránke mapy, je aj predikovaná trasa balónu a miesto kde balón pristane. Takáto informácia značne pomôže pri lokalizovaní servisného modulu spoločne s iným nákladom vyslaným do atmosféry. Na webe existuje niekoľko webových portálov, ktoré poskytujú predikciu trasy a dopadu balónu na základe aktuálnych informácií o stave a vetroch v atmosfére.

Pre predpoveď nám stačí odoslať základné informácie o vlastnostiach balónu, mieste a čase jeho vypustenia na predikčný webový server. Nami používaný webový server predict.habhub.org poskytuje okrem zobrazenia predikovanej trasy na mape portálu aj možnosť stiahnutia CSV súboru so súradnicami predikovanej trasy. Tieto súradnice vieme prostredníctvom Google Maps API zapracovať do našej webovej stránky zobrazujúcej mapu s polohou balóna. Celý proces komunikácie je automatizovaný samostatným Python skriptom, ktorý zabezpečuje získanie súradníc z predikčného servera. V mape je predikovaná trasa zobrazená odlišnou farbou pre jednoduché odlíšenie od skutočnej trasy balóna.



Obrázok 3.37 Predikovaná trasa na našej web stránke s mapou.

3.10 Propagácia na IIT.SRC a TP CUP 2017

S naším projektom sme sa zúčastnili konferencie IIT.SRC 2017, ktorá sa konala na našej fakulte. V súvislosti s touto konferenciou bolo potrebné vytvoriť poster, ktorý bude zobrazovať doterajší progres na projekte a zároveň bude slúžiť ako pomôcka pri prezentácii. Samotný poster môžeme vidieť na obrázku 32. Na obrázku 33 môžeme vidieť tímové tričká, ktoré sme si dali vyrobiť, aby sme na konferencii ukázali tímový duch.

Obrázok 3.38 Poster na konferenciu IIT.SRC 2017

Obrázok 3.39 Návrh tričiek na konferenciu IIT.SRC 2017

3.11 Testovanie

3.11.1 Zobrazovanie údajov na stránke v reálnom čase

Pri testovaní odosielania údajov pomocou Web-socketov na stránku, kde sa trasa balónu vykresľuje v reálnom čase, sme zistili, že náš používaný Web server na serveri tímového projektu nepodporuje použitie socketov. Použitý server Apache spolu s pluginom mod_wsgi pre beh python aplikácie totiž neumožňuje vytvoriť a udržovať ďalšie asynchrónne spojenie pre sockety.

Je preto nutné použiť iný web server, ktorý nahradí Apache prípadne Apache musí tieto spojenia presmerovať na druhý webový server udržujúci web-socket spojenia.

3.11.2 Spracovanie údajov na serveri

V rámci testovanie sme odosielali údaje na server po dobu približne dvoch hodín. Frekvencia odosielania bola 1 sekunda, čo je síce extrémna hodnota, ktorá počas letu nebude realizovaná, no cieľom bolo vykonať záťažový test. Spracovanie dát a ich ukladanie bolo bez výraznej záťaže, no načítavanie údajov z databázy pre použitie na stránke sa ukázalo ako príliš pomalé. Načítavanie údajov trvá v priemere dve až tri minúty, čo je pre webovú stránku nepoužiteľné. Toto spomalenie je zapríčinené použitím SQLite databázy, ktorá je uložená v jednom súbore a pri každej požiadavke na databázu je súbor otváraný a postupne čítaný. Pri veľkom množstve dát (po teste mal súbor databázy približne 5MB) je tak prehľadávanie väčšieho množstva dát príliš pomalé. V ďalšej verzii bude potrebné nájsť a implementovať vhodnejší databázový systém.

4 Bibliografia

[1] Letové prevádzkové služby Slovenskej republiky, š. p., „Letecká informačná príručka," 2016.

[2] Európska komisia, „EUR-Lex, nariadenie č. 923/2012," 26 September 2012. [Online]. Available: http://eur-lex.europa.eu/legal-content/SK/TXT/PDF/?uri=CELEX:32012R0923&qid=1479157226138&from=EN.

[3] Letecká informačná služba Slovenskej republiky, „Civilné predpisy, L2 Pravidlá lietania," Apríl 2009. [Online]. Available: http://www.fabryatc.net/civilnepredpisy/L2.pdf.

[4] Zawin, „Svetelektro," 15 3 2011. [Online]. Available: https://svetelektro.com/clanky/nasiel-sa-balon-universum-394.html. [Cit. 10 2016].

[5] Zawin, „Svetelektro," 14 5 2011. [Online]. Available: http://svetelektro.com/clanky/balon-universum-2-zhodnotenie-399.html. [Cit. 10 2016].

[6] „What is Arduino," [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Cit. 3 10 2016].

[7] E. s. r. o., „Digitálne teplotné čidlo DS18B20 po zbernici 1-Wire," [Online]. Available: http://www.elektrobazeny.sk/titulka/arduino/zakladne-info-18b20/. [Cit. 3 10 2016].

[8] Farnell, „Arduino Uno," [Online]. Available: http://www.farnell.com/datasheets/1682209.pdf. [Cit. 3 10 2016].

[9] Farnell, „Arduino Mega2560," [Online]. Available: http://www.farnell.com/datasheets/810077.pdf?_ga=1.6948594.1701222862.1475489180. [Cit. 3 10 2016].

[10] SparkFun, „SparkFun GPS Logger Shield," [Online]. Available: https://www.sparkfun.com/products/13750. [Cit. 3 10 2016].

[11] ADH-tech, „GP3906-TLP DataSheet," [Online]. Available: https://cdn.sparkfun.com/assets/learn_tutorials/4/6/8/GP3906-TLP_DataSheet_Rev_A01.pdf. [Cit. 3 10 2016].

[12] E. Tutorials, „Temperature Sensors,“ [Online]. Available: http://www.electronics-tutorials.ws/io/io_3.html. [Cit. 1 11 2016].

[13] A. C. P. Thermistors, „4 Most Common Types of Temperature Sensors,“ 12 2015. [Online]. Available: http://www.ametherm.com/blog/temperature-sensor-types. [Cit. 1 11 2016].

[14] T. E. components, „Proffuse PT100-1020,“ [Online]. Available: http://www.tme.eu/sk/details/pt100-1020/teplotne-senzory-odporove/proffuse/. [Cit. 1 11 2016].

[15] D. semiconductor, „Ds18B20 Programmable Resolution 1-Wire Digital Thermometer,“ [Online]. Available: http://www.gme.sk/img/cache/doc/530/067/ds18b20-datasheet-1.pdf. [Cit. 1 11 2016].

[16] GMElectronic, „DS18B20,“ [Online]. Available: http://www.gme.sk/ds18b20-p530-067. [Cit. 1 11 2016].

[17] D. semiconductors, „KTY81-1 series Silicon temperature sensors,“ 8 2000. [Online]. Available: http://pdf.datasheetcatalog.com/datasheet/philips/KTY81-1SERIES_3.pdf/. [Cit. 1 11 2016].

[18] MUO, „How and Why to Add a Real Time Clock to Arduino,“ 2 2014. [Online]. Available: http://www.makeuseof.com/tag/how-and-why-to-add-a-real-time-clock-to-arduino/. [Cit. 29 10 2016].

[19] Arduino, „DS1302 Real Time Clock,“ [Online]. Available: http://playground.arduino.cc/Main/DS1302. [Cit. 30 10 2016].

[20] Arduino, „Timer1,“ [Online]. Available: http://playground.arduino.cc/Code/Timer1. [Cit. 31 10 2016].

[21] M. T. Karol Rástočný, „Softvérové nástroje pre vývoj softvéru v tíme,“ 2016.

[22] „Digital Pins,“ [Online]. Available: https://www.arduino.cc/en/Tutorial/DigitalPins. [Cit. 22 4 2017].

[23] M. Petrinec a P. Májová, „Orienteering,“ 12 1 2016. [Online]. Available: http://www.orienteering.sk/page/co-je-to-orientacny-beh. [Cit. 10 2016].

[24] „RLX,“ [Online]. Available: http://rlx.sk/sk/433mhz-868mhz-915mhz-24ghz/3544-433mhz-rf-transmitting-module-er-wrf43301r-transmitter-receiver.html. [Cit. 11 2016].