

UNIVERZITA OBRANY V BRNĚ

FAKULTA VOJENSKÝCH TECHNOLOGIÍ

Studijní program: Technologie pro obranu a bezpečnost

Studijní obor: Technologie pro ochranu majetku a osob

Ev. číslo: 3101/19



BAKALÁŘSKÁ PRÁCE

Název: Systém pro záznam pohybu vozidla

Zpracoval:

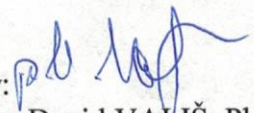
Šimon Brázda

Vedoucí závěrečné práce:

mjr. Ing. Radek Bystřický, Ph.D.

BRNO 2019

Schvaluji:

Vedoucí katedry: 
plk. gšt. prof. Ing. David VALIŠ, Ph.D. et Ph.D.

Studijní program: Technologie pro obranu
a bezpečnost

Studijní obor: Technologie pro ochranu
majetku a osob

Katedra: Bojových a speciálních vozidel

V Brně dne: 30. 10. 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Šimon Brázda

Téma: Systém pro záznam pohybu vozidla

Vedoucí BP: mjr. Ing. Radek Bystřický, Ph.D.

Konzultanti: mjr. Ing. Josef Bajer, Ph.D.

kpt. Ing. Přemysl Janů, Ph.D.

Začátek práce dne: 31. 1. 2019

Ukončení práce dne: 31. 5. 2019

Zadání přijal dne: 30. 10.



Podpis studenta

Pokyny pro zpracování bakalářské práce

I. Obsah práce (výčet úkolů, které je nutno zpracovat a obhájit) a základní údaje pro BP

- Principy měření polohy pomocí systémů satelitní navigace.
- Stručný popis komunikačního protokolu (NMEA/uBlox).
- Podrobný popis komunikačních zpráv využitých v navrženém systému.
- Blokovaný návrh zařízení měřícího polohu vozidla, ukládajícího změřenou polohu na SD kartu nebo jiné paměťové médium a předávajícího informace o aktuální poloze pomocí zvolené bezdrátové komunikace (WiFi, Bluetooth, GSM, apod.)
- Podrobný popis a zdůvodnění použitého způsobu měření, včetně schématu a popisu jednotlivých elektronických součástek.
- Student uvede v práci vývojový diagram navrženého programu a důležité části kódu. Celý program pak přiloží do přílohy a na CD.

Zveřejnění BP je stanoveno zákonem č.111/1998 Sb. Zákon o VŠ a v podmínkách Univerzity obrany postupujte podle Studijního a zkušebního řádu Univerzity obrany v Brně, čl. 24 a 36, přílohy č. 2 „Podstatné požadavky na závěrečné práce“.

II. Doporučená literatura

- [1] E. Kaplan and C. J. Hegarty, *Understanding GPS/GNSS: Principles and Applications*, Third Edition (Gnss Technology and Applications Series). Artech House, 2017. [Online]. Available: <https://www.amazon.com/Understanding-GPS-GNSS-Principles-Applications/dp/1630810584>
- [2] [online]. Copyright © [cit. 29.10.2018]. Dostupné z: https://www.u-blox.com/sites/default/files/products/documents/u-blox6_ReceiverDescrProtSpec_%28GPS.G6-SW-10018%29_Public.pdf
- [3] Standard NMEA-0183 sentences description. *Free Online NMEA Tools for converting, creating and analysing NMEA logs* [online]. Dostupné z: <http://freenmea.net/docs>
- [4] Arduino GPS modul Neo-6M | Arduino návody. Webový magazín o ARDUINU | Arduino návody [online]. Dostupné z: <https://navody.arduino-shop.cz/navody-k-produktum/arduino-gps-modul-neo-6m.html>
- [5] Lekce 13 - Arduino a SD karta. itnetwork.cz - *Ajtácká sociální síť a materiállová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další*. [online]. Copyright © 2018 itnetwork.cz. Veškerý obsah webu [cit. 29.10.2018]. Dostupné z: <https://www.itnetwork.cz/hardware-pc/arduino/arduino-sd-karta>
- [6] Arduino Bluetooth modul HC-05 | Arduino návody. Webový magazín o ARDUINU | Arduino návody [online]. Dostupné z: <https://navody.arduino-shop.cz/navody-k-produktum/arduino-bluetooth-modul-hc-05.html>
- [7] Arduino GSM Shield SIM900 | Arduino návody. Webový magazín o ARDUINU | Arduino návody [online]. Dostupné z: <https://navody.arduino-shop.cz/navody-k-produktum/arduino-gsm-shield-sim900.html>

V Brně dne 29.10.2018

.....
Vedoucí BP (podpis)

BP převzata vedoucím dne

BP odevzdána oponentovi dne

Poděkování

Chtěl bych poděkovat Ing. Radku Brázdovi, mjr. Ing. Radku Bystřickému, Ph.D. a kpt. Ing. Přemyslu Janů, Ph.D. za pomoc a rady při zpracování této bakalářské práce.

Čestné prohlášení

Prohlašuji, že jsem bakalářskou práci na téma „Systém pro záznam pohybu vozidla“ vypracoval samostatně, pod odborným vedením mjr. Ing. Radka Bystřického, Ph.D. a použil jsem pouze literární zdroje uvedené v práci.

V Brně dne.....

.....

Abstrakt

Bakalářská práce se zabývá návrhem a sestrojením systému pro záznam polohy dopravních prostředků pomocí Globálního družicového polohovacího systému (GNSS). V první části této bakalářské práce je definován účel systému a požadavky na něj. Na základě těchto požadavků a účelu je navržen samotný systém. V návrhu se práce zabývá výběrem ovládacího a zpracujícího prvku (počítače), paměťového média a druhu bezdrátové komunikace. Následně jsou objasněny principy funkce GNSS a jejich přijímačů. Dále se práce věnuje popisu jednotlivých komponent použitých v systému. V další části je sestrojeno a naprogramováno samotné zařízení ukládající měřenou polohu na paměťové médium a posílající tyto informace bezdrátovou komunikací GSM (Global System for Mobile Communication) na server. Server přijatá data ukládá a zobrazuje poslední polohu. Na závěr je provedena zkouška systému a jeho zhodnocení.

Klíčová slova

GPS Tracker, Global Navigation Satellite System, NMEA, Arduino, sledování polohy.

Abstract

The bachelor thesis deals with the design and construction of a system for recording position of transport vehicles using the Global Navigation Satellite System (GNSS). The first part of this bachelor thesis defines the purpose of the system and its requirements. Based on these requirements and purpose, the system itself is designed. The design deals with the selection of control and processing unit (computer), storage medium and type of wireless communication. Subsequently, the principles of GNSS and their receivers are explained. Furthermore, the thesis describes the components used in the system. In the next section, the device itself is build and programmed in a way that it stores the measured position on the storage medium and sends it via GSM (Global System for Mobile Communication) wireless communication to the server. The server stores the received data and displays the last location. In the end, the system is tested and evaluated.

Key words

GPS Tracker, Global Navigation Satellite System, NMEA, Arduino, location tracking.

Obsah

Úvod.....	14
1 Účel systému, požadavky na něj a jeho návrh	15
2 Globální družicový polohový systém (GNSS).....	18
2.1 Navigační zpráva.....	18
2.2 Metody určování polohy	19
2.3 Souřadnicový systém	19
2.4 Princip určování polohy pomocí kódových měření	21
2.5 Princip výpočtu polohy přijímače	23
3 Komunikační protokoly NMEA, UBX	28
3.1 Struktura protokolu NMEA.....	28
3.2 Výpočet kontrolní sumy NMEA zpráv	29
4 Převod souřadnic z NMEA vět do souboru GPX.....	30
5 Popis jednotlivých komponent.....	33
5.1 Arduino Nano.....	33
5.2 GPS modul u-blox NEO-7M-0-000	34
5.3 GSM modul SIM800L	37
5.4 MicroSD Card Reader/Writer Adapter v1.0 od CATALEX.....	38
5.5 Napájecí blok	39
6 Sestrojení systému.....	40
7 Vývojový diagram a důležité části kódu.....	44
7.1 Funkce processGPS.....	45
7.2 Funkce calcChecksum.....	49
7.3 Funkce decodeLine	52
7.4 Funkce sendOnSD.....	55
7.5 Funkce gsmSendHttp	58
8 Zpracování dat serverem a zobrazení poslední polohy	61

9	Odzkoušení systému a jeho zhodnocení.....	62
	Závěr	64
	Seznam literatury	66
	Seznam příloh	67

Seznam zkratek

ASCII	American Standard Code for Information Interchange (Americký standardní kód pro výměnu informací)
DDC	Display Data Channel
DOP	Dilution of Precision
EEPROM	Unipolární paměť PROM
FLASH	Elektricky programovatelná paměť s libovolným přístupem
GPX	GPS Exchange Format
GPIO	General-purpose input/output
GPRS	General Packet Radio Service (služba pro uživatele GSM pro přenos dat)
GND	Ground
GNSS	Global Navigation Satellite System (Globální družicový polohový systém)
GPS	Global Positioning System (Globální polohový systém)
GSM	Global System for Mobile Communication (Globální systém pro mobilní komunikaci)
GLONASS	Globalnaja navigacionnaja sputnikovaja sistema (Ruský GNSS)
I2C	Inter-Integrated Circuit (Multi-masterová sériová sběrnice)
IRNSS	Indian Regional Navigation Satellite System (Indický regionální družicový navigační systém)
IMEI	International Mobile Equipment Identity (unikátní číslo přidělené SIM kartě)
KML	Keyhole Markup Language (jazyk pro distribuci geografických dat)
LiPo	Lithium-polymerový akumulátor
LED	Light-Emitting Diode (Elektroluminiscenční dioda)
LNA	Low Noise Amplifier (Nízkošumový zesilovač)
MOSI	Master Out, Slave In (Master výstup, Slave vstup)
MISO	Master In, Slave Out (Master vstup, Slave výstup)
NMEA	National Marine Electronics Association (Americké národní sdružení pro loďní elektroniku)

PWM	Pulse Width Modulation (Pulzně šířková modulace)
PMU	Power Management Unit
PSR	Pseudorange (zdánlivá vzdálenost)
QZSS	Quasi-Zenith Satellite System (Japonský regionální družicový navigační systém)
RS-232	sériová linka
RTCM	Radio Technical Commission for Maritime Services
RX	Receiver (přijímač)
RF	Radio Frequency
RTC	Real-time clock (Hodiny reálného času)
ROM	unipolární paměť
RAM	Random Access Memory (polovodičová paměť s přímým přístupem)
SD	Secure Digital (paměťová karta)
SIM	Subscriber Identity Module (Identifikační karta účastníka mobilní sítě)
SPI	Serial Peripheral Interface (Sériové periferní rozhraní)
SS	Slave Select (adresace zařízení někdy též CS - Chip Select)
SCK	System Clock (hodiny systému)
SRAM	Static Random Access Memory (statická paměť RAM)
TX	Transmitter (vysílač)
UTC	Coordinated Universal Time (Koordinovaný světový čas)
UBX	u-blox protokol pro přenos GPS dat
UART	Universal Synchronous/Asynchronous Receiver and Transmitter (sériová komunikace s nastavitelným asynchronní režimem)
USB	Universal Serial Bus (Univerzální sériová sběrnice)
VCC	Common Collector Voltage (kladné napájecí napětí)
WGS 84	World Geodetic System 1984 (Světový geodetický systém 1984)
XML	Extensible Markup Language - rozšiřitelný značkovací jazyk

Seznam obrázků

Obrázek 1 - Blokové schéma systému	17
Obrázek 2 - Kartézský souřadnicový systém [4]	20
Obrázek 3 - Elipsoidní souřadnicový systém [4]	20
Obrázek 4 - Určení polohy pomocí tří kulových ploch [4]	21
Obrázek 5 - Čtyři satelity jsou potřeba k určení polohy v prostoru [4]	22
Obrázek 6 - Minimálně signály ze čtyř satelitů musí být přijaty [4]	23
Obrázek 7 - Určení polohy v prostoru [4]	24
Obrázek 8 - Odhadovaná pozice vůči skutečné poloze [4]	25
Obrázek 9 - GPS module NEO-7M [11]	36
Obrázek 10 - GSM modul SIM800L schéma [12]	37
Obrázek 11- MicroSD Card Reader/Writer [13]	39
Obrázek 12 – GPS Tracker, pohled zevnitř	41
Obrázek 13 – Schéma zapojení systému	42
Obrázek 14 – GPS Tracker, pohled zepředu	43
Obrázek 15 – GPS Tracker, pohled z boku	43
Obrázek 16 - Vývojový diagram programu GPS_Tracker	44
Obrázek 17 - Vývojový diagram funkce processGPS	47
Obrázek 18 - Vývojový diagram funkce CalcChecksum	50
Obrázek 19 - Vývojový diagram funkce decodeLine	53
Obrázek 20 - Vývojový diagram funkce sendOnSD	56
Obrázek 21 - Vývojový diagram funkce gsmSendHttp	59
Obrázek 22 - Výběr sledovaného zařízení	61
Obrázek 23 - změřená data, garáž	62
Obrázek 24 - Detail cesty	62

Seznam tabulek

Tabulka 1 - Struktura NMEA zpráv	29
Tabulka 2 – Názorný výpočet kontrolní sumy	30
Tabulka 3 - Technická data Arduino Nano	34
Tabulka 4 - Průměrná spotřeba elektrického proudu modulu NEO-7M	35
Tabulka 5 - Provozní režimy GSM modulu SIM800L	38
Tabulka 6 - Struktura GPRMC věty	46

Úvod

V posledních letech dochází k velkému rozvoji a popularizaci takzvaných GPS (Global Positioning System) trackerů/lokátorů a služeb s nimi spojených. Tyto zařízení slouží k sledování polohy zejména dopravních prostředků, ale i osob, zásilek a zvířat. Sledování je dostupné ve třech variantách. První varianta je použití zařízení jako data logger, který data o pozici ukládá na paměťové médium (nejčastěji microSD kartu) a umožňuje jejich zpětné zobrazení. Druhá varianta je posílání pozice pomocí bezdrátové komunikace na server, kde se ukládá a zobrazuje v reálném čase na mapových podkladech. Třetí varianta je kombinace obou předešlých, kdy jsou data ukládána na paměťové médium a odesílána na server současně. Pořizovací cena těchto zařízení se pohybuje v rozmezí 500 až 20 000 Kč v závislosti na množství poskytovaných funkcí, plus cena za provoz okolo 100 Kč měsíčně [1].

Na základě těchto zjištění jsem usoudil, že by bylo vhodné sestavit podobné zařízení a poznat tak, jakým způsobem funguje. Rozpočet pro sestavení systému jsem si stanovil ve výši 1000 korun. Při výběru komponent jsem se snažil zejména zohlednit jejich cenu, dostupnost, velikost, kompatibilitu, aplikovatelnost, volnou dostupnost knihoven a vývojového prostředí pro zvolený ovládací prvek a dostupnost datasheetů. K zpracování dat a ovládání celého zařízení jsem využil otevřenou elektronickou platformu Arduino, založenou na jednoduché počítačové desce (hardware) a vývojovém prostředí, které slouží k tvorbě softwaru. Arduino jsem vybral pro jeho nízkonákladovost, multiplatformnost, jednoduché a čisté programovací prostředí, open source a rozšiřitelný software a hardware [2].

1 Účel systému, požadavky na něj a jeho návrh

Pro kvalitní návrh a sestavení systému je nezbytné nejprve stanovit jeho účel a požadavky na něj. Systém má sloužit k sledování pozice dopravních prostředků v reálném čase a umožnit zpětné zobrazení trasy. Systém je cílen pro vlastníky dopravních prostředků, a to zejména silničních vozidel, kteří chtějí mít možnost lokalizovat své dopravní prostředky v případě jejich odcizení, či jen monitorovat jejich pohyb a kontrolovat oprávněnost jejich cest. Systém lze však také využít při připojení externího akumulátoru k sledování (špionáži) cizích vozidel, zvěře a zásilek.

Obecné požadavky:

- co nejmenší velikost (nechceme, aby pachatel zařízení zpozoroval a chceme, aby zařízení zabíralo co nejméně místa),
- cenová dostupnost (cena samotného zařízení, ale i jeho provoz a služby),
- nízká spotřeba elektrické energie (výdrž baterie),
- jednoduchost (konstrukce, ovládání, provoz, instalace),
- spolehlivost,
- robustnost.

Specifické požadavky:

- bezdrátové sledování polohy v reálném čase,
- ukládání polohy na paměťové médium,
- bezdrátové ukládání dat na server,
- při selhání bezdrátové komunikace pokračovat v ukládání dat na paměťové médium,
- možnost napájení z akumulátoru dopravního prostředku nebo i připojením externího akumulátoru,
- zobrazení poslední polohy serverem,
- rychlé nalezení polohy,
- vypnutí zařízení při vyjmutí klíče ze zapalování nebo odpojením napájení,
- zobrazení změřené trasy v reálném čase,
- šifrování posílaných dat,
- autorizace ke sledování vymezených zařízení na základě identifikace přihlašovacím jménem a autentizace heslem.

Návrh systému:

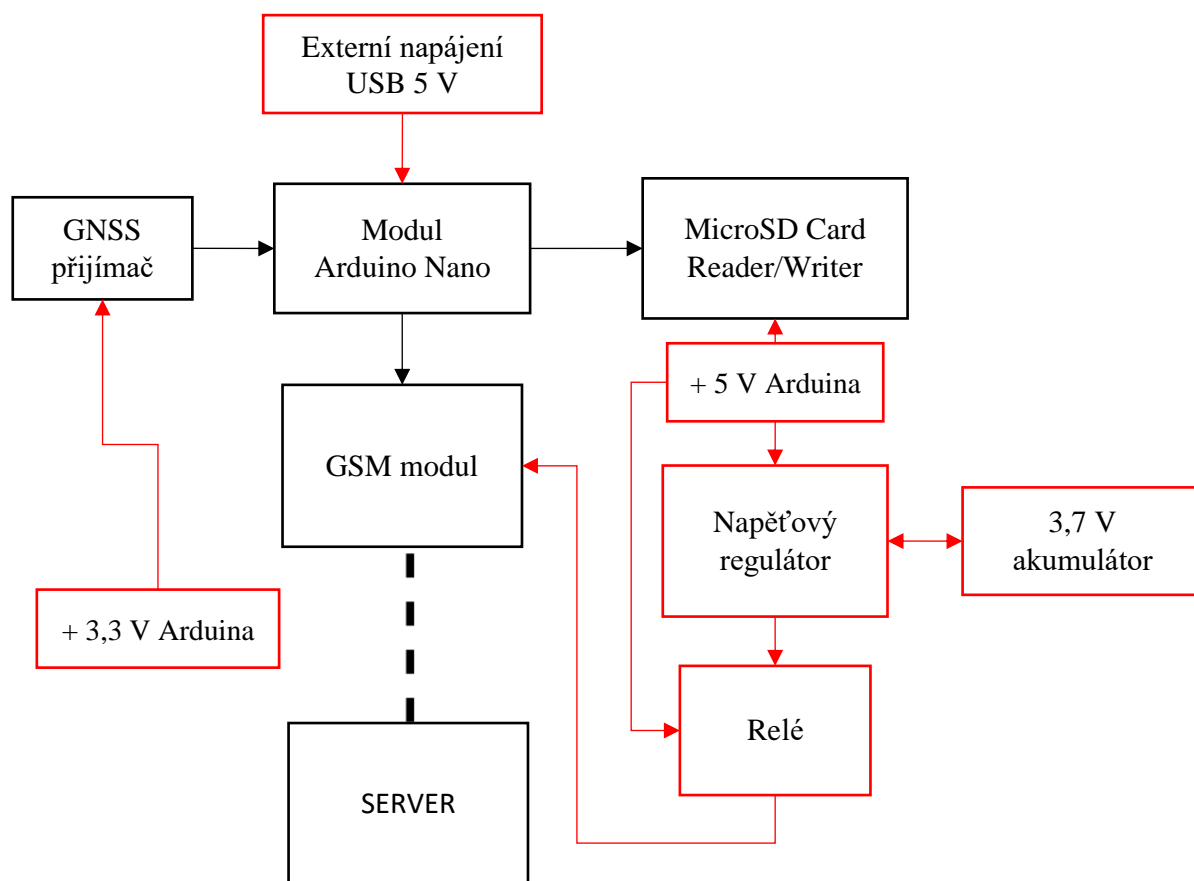
Po zvážení účelu a všech požadavků jsem došel k závěru, že nejvhodnějším způsobem bezdrátové komunikace bude GSM. Bluetooth a WiFi jsou energeticky náročnější, navíc Bluetooth má velice omezený dosah a vyžaduje párování. Ve vozidle by neustále musel být telefon spárovaný se zařízením, který by nakonec též data posílal přes GSM, čímž by se značně zvýšila cena a složitost systému.

Dalším důležitým rozhodnutím bylo zvolení ovládacího prvku. Vybíral jsem mezi dvěma variantami, a to Arduino a Raspberry Pi. Vybral jsem Arduino, jelikož nebudeme zpracovávat velké množství dat (obraz, zvuk), ale pouze krátké textové zprávy, k čemuž se spíše hodí mikrokontrolér nežli mikroprocesor. Co se týče konkrétního typu Arduina, tak nejvhodnější se mi jevil Arduino Nano, díky své malé velikosti, dostatku paměti a množství digitálních pinů.

Způsob napájení zařízení jsem zvolil přes USB, protože v dnešní době již převážná většina motorových vozidel má USB výstup nebo jej po drobné investici může mít. Toto řešení mi umožní komunikovat se systémem, ladit ho a užívat ho napříč různými dopravními prostředky. Zařízení tak bude možné napájet z externího akumulátoru a použít jej nezávisle pro sledování nejen dopravních prostředků ale i zásilek, osob a zvěře.

Přijímač GNSS bude přijímat a zpracovávat signály z družic a následně je pak posílat Arduino. Arduino data zformátuje do potřebné podoby, zapíše na microSD kartu přes modul pro microSD karty a odešle přes GSM na Server. Arduino bude napájeno přes USB z autobaterie. GNSS modul bude napájen 3,3 V z Arduina. MicroSD modul bude napájen 5 V též z Arduina. Z Arduina též bude dobíjena baterie přes napěťový regulátor, který sníží napětí na potřebnou úroveň. Z napěťového regulátoru bude také napájen GSM modul přes spínací relé. Spínání relé bude ovládáno (napájeno) Arduinem. Blokové schéma systému je na obrázku 1.

S ohledem na definované požadavky jsem pro sestavení zařízení využil tyto komponenty: Arduino Nano, GPS modul u-blox NEO-7M-0-000, GSM modul SIM800L, MicroSD Card Reader/Writer Adapter v1.0 od CATALEX, napájecí blok tvořený napěťovým regulátorem, spínacím relé HKE HRS1H-S-DC5V a LiPo akumulátorem 3,7 V. Popis jednotlivých komponent je v kapitole Popis jednotlivých komponent.



Obrázek 1 - Blokové schéma systému

Pro další práci bylo nezbytné nastudovat a popsat, co to je Globální družicový polohový systém (GNSS) a jaké jsou druhy. Vybrat ten nejvhodnější pro mé účely a popsat ho.

2 Globální družicový polohový systém (GNSS)

„Globální družicový polohový systém (Global Navigation Satellite System, zkratkou GNSS) je služba umožňující za pomoci družic autonomní prostorové určování polohy s celosvětovým pokrytím. Uživatelé této služby používají malé elektronické rádiové přijímače, které na základě odeslaných signálů z družic umožňují vypočítat jejich polohu s přesností na desítky až jednotky metrů.“ [3] Každá družice nese atomové hodiny pro určování přesného času a vysílač. Tyto hodiny jsou synchronizovány ze země na jeden systémový čas. Přijímače jsou z důvodů dostupnosti a velikosti vybaveny jen krystalem řízenými hodinami. Tyto hodiny jsou přijímači synchronizovány s časem družic výpočty z navigačních zpráv posílaných družicemi [4], [5].

„V současnosti (rok 2018) je plně funkční globální systém pouze americký GPS a ruský GLONASS. Vývoj probíhá na evropském Galileo a BeiDou-3, s jejich uvedením do plné operační schopnosti se počítá v roce 2019 a 2020. Mimo GNSS existují i regionální autonomní družicové polohové systémy jako je čínský Beidou 1 a 2, japonský QZSS a vyvíjený indický IRNSS.“ [3]

Systém GPS má v současnosti celkem 32 družic. Družice obíhají ve výšce 20 350 km nad povrchem Země na šesti kruhových drahách se sklonem 55°. Dráhy jsou vzájemně posunuty o 60° a na každé dráze je jich 5 až 6 nepravidelně rozmístěno. Na střední oběžné dráze se pohybují rychlostí 3,8 km/s, s dobou oběhu kolem Země 11h 58min [6]. Pro svůj systém jsem se rozhodl využít systém GPS, jelikož je nejrozšířenější a snadno dostupný.

2.1 Navigační zpráva

Aby přijímače byly schopné nalézt své pozice vysílají družice globálního polohovacího systému navigační zprávy. Navigační zprávy se skládají z času vysílání počátku zprávy, stavu družice, efemeridů družic, údajů umožňujících přesně korigovat čas vysílání družic, almanachu, a koeficientů ionosférického modelu [5], [6].

Stav družice informuje uživatele o závadách na družici a o tom, zda a v jakém rozsahu ji použít pro určování polohy [5].

Almanach obsahuje parametry oběžných drah všech družic a údaje o jejich stavu. To umožňuje přijímači, aby při znalosti aktuálního almanachu byl schopen začít vyhledávat družice aktuálně viditelné v dané oblasti a mohl tak značně snížit dobu nastartování přijímače a získání signálu [5].

Efemeridy jsou predikované polohy družic na oběžných drahách [5].

Koeficienty ionosférického modelu používá přijímač pro přibližný odhad vlivu ionosféry na signály pro kterýkoliv místo a kterýkoliv čas [5].

2.2 Metody určování polohy

Přijímač určuje svoji polohu výpočty z dat přijatých navigačních zpráv. K určení polohy používá nejčastěji jednu z metod:

- kódových měření,
- fázových měření,
- dopplerovských měření,
- úhloměrných měření [5].

Fázové měření vychází z možnosti měřit jednotlivé fáze harmonických vln vysílaných družicemi a jejich změn. Na počátku měření přijímač spočítá počet vlnových délek signálu, nacházejících se mezi přijímačem a družicí. Jakmile jednou přijímač počáteční hodnotu nejednoznačnosti počtu vln určí, je již schopen průběžně sledovat změny fázového posunu a počtu celých vln a tím i vlastní polohu [3], [5].

Dopplerovská měření využívají pro určení Dopplerův posun. V důsledku relativního pohybu družice vůči přijímači se průběžně mění frekvence přijímaného signálu. Tento frekvenční posun je po určitou dobu měřen a pak je na základě získaných údajů vypočtena změna radiální vzdálenosti mezi družicí a přijímačem. Z těchto rozdílů vzdáleností pak může být vypočtena poloha přijímače. Tento způsob měření se spíše využívá k určování rychlosti, jakou se přijímač pohybuje [3], [5].

„Úhloměrná měření vychází z možnosti zaměřovat zdroj signálu (družici) pomocí směrových antén a určit úhly vzhledem k vodorovné rovině. Provádí se k více družicím zároveň, nebo k jedné družici v různém čase. Tato metoda se však z důvodu komplikovaného řešení a malé přesnosti nepoužívá.“ [3]

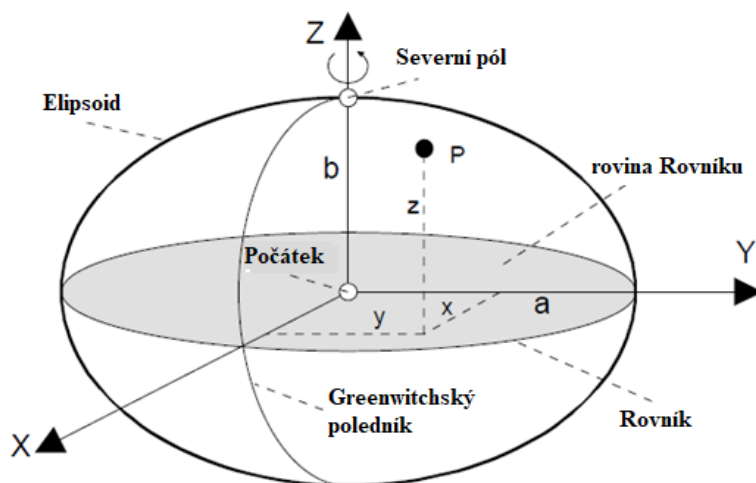
„Kódová měření jsou spolehlivá a přijímači nejčastěji používaná.“ [3] Základním principem kódových měření je určování vzdáleností mezi přijímačem a družicemi [3], [5]. Podrobný popis Kódových měření je v podkapitole Princip určování polohy pomocí kódových měření.

Pro objasnění principu určování polohy pomocí kódových měření a popisu výpočtu polohy je nezbytné si nejdříve ujasnit, jak zeměkouli popisujeme a co to jsou souřadnice.

2.3 Souřadnicový systém

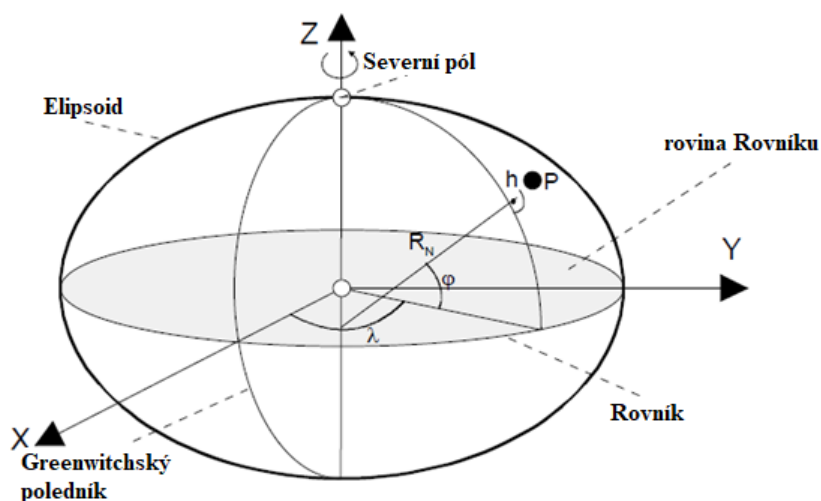
Globální navigační systém pracuje v souřadnicovém systému WGS 84 (World Geodetic System 1984). WGS 84 je geocentrický pravoúhlý pravotočivý systém pevně spojený se Zemí čili se jedná o Kartézský souřadnicový systém s počátkem ve středu hmoty (Geocentrický) elipsoidu, který aproximuje totální hmotu Země [4].

Kladná osa X elipsoidu leží na rovině rovníku (obrázek 2). Prochází středem hmoty a bodem, kde se protíná rovník s Greenwichským meridiánem (nultý poledník). Osa Y také leží na rovníkové rovině, ale s natočením o 90° východně vůči ose X. Osa Z leží kolmo vůči osám X, Y a prochází geografickým Severním pólem (obrázek 2) [4].



Obrázek 2 - Kartézský souřadnicový systém [4]

V praxi jsou používány Elipsoidní souřadnice (ϕ , λ , h) (obrázek 3) místo Kartézských (X , Y , Z). ϕ odpovídá zeměpisné šířce, λ odpovídá zeměpisné délce a h odpovídá elipsoidní výšce (délka čáry P ke středu elipsoidu) [4].



Obrázek 3 - Elipsoidní souřadnicový systém [4]

2.4 Princip určování polohy pomocí kódových měření

K určení vzdálenosti mezi přijímačem a družicemi se využívá časových značek, které obsahují čas vyslání navigační zprávy a polohu každé družice v prostoru, tzv. efemeridy. Přijímač pracuje tak, že ve vstupním signálu identifikuje dálkoměrný kód příslušné družice, zjistí čas odeslání a přijetí jedné sekvence kódu a ze zjištěného časového rozdílu Δt určí vzdálenost mezi přijímačem a družicí r dle vztahu:

$$r = c * \Delta t, \quad (1)$$

$$\Delta t = t_r - t_t, \quad (2)$$

kde:

r je vzdálenost přijímače od satelitu, m,

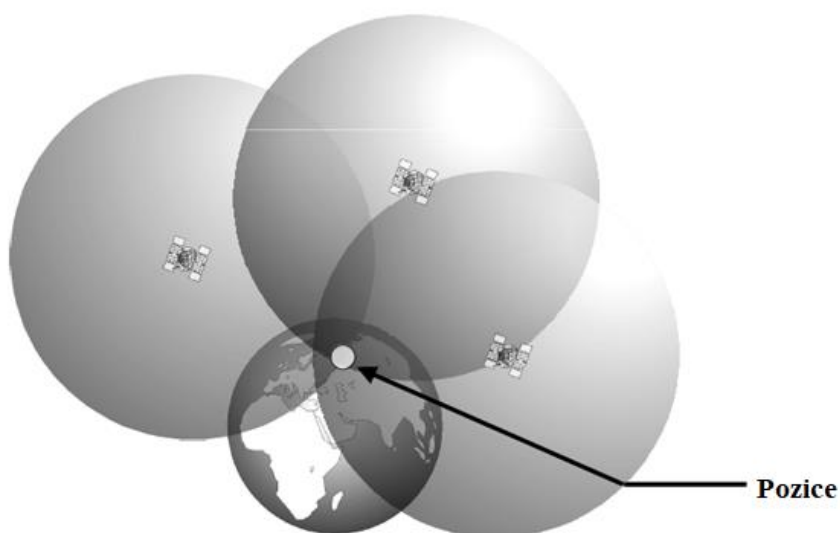
c je přibližná rychlost šíření světla, m.s⁻¹,

t_r je aktuální čas přijímače, s,

t_t je čas poslaný ze satelitu, s,

Δt je doba letu signálu, s [3], [4], [6].

Ze znalosti času přijímače a vyslaného času signálu satelitu je tedy přijímač schopen určit vzdálenost od satelitu. To samotné nestačí k určení přesné polohy, jelikož zatím pouze zná vzdálenost od družice. Tato vzdálenost udává poloměr kulové plochy kolem družice, na které se nachází přijímač. Aby přijímač získal svoji polohu v prostoru potřebuje tedy signál z alespoň tří družic, jelikož ani průnik dvou kulových ploch mu nedá jeden bod. Hledá tedy průnik minimálně tří kulových ploch (Obrázek 4) [4].

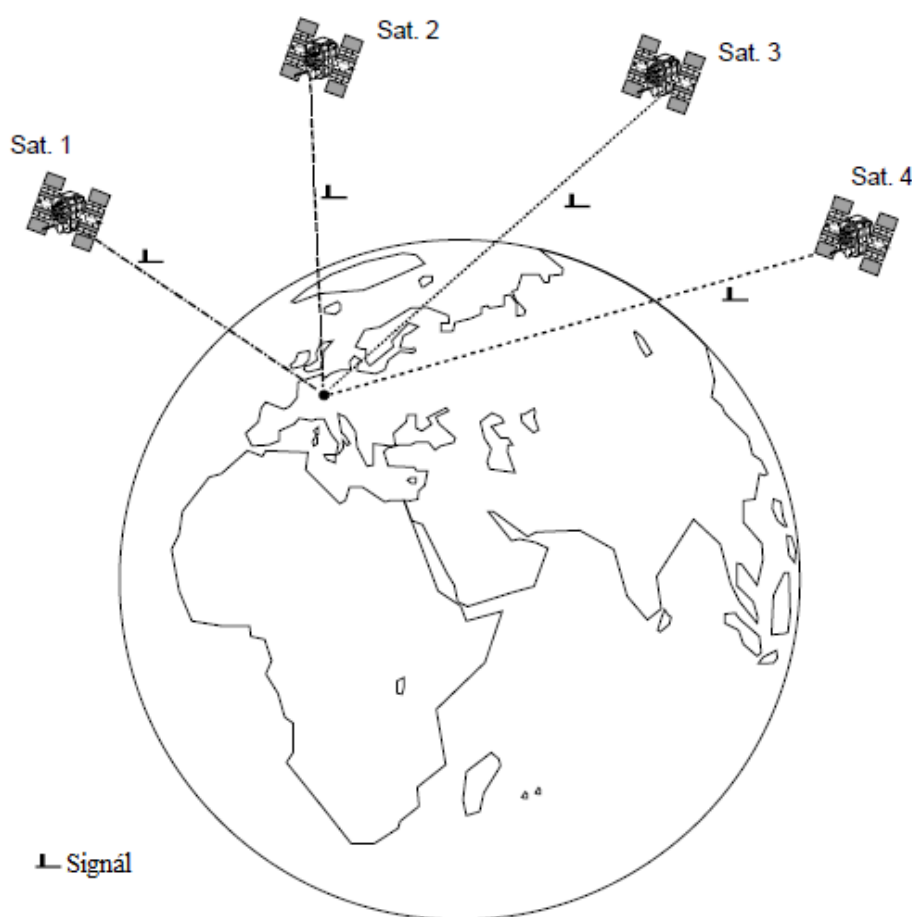


Obrázek 4 - Určení polohy pomocí tří kulových ploch [4]

Nastává zde však další problém. Vzhledem k tomu, že hodiny přijímače nejsou zcela synchronní se systémovým časem družicového navigačního systému, je časový rozdíl Δt zatížen určitou chybou hodin přijímače. Při výpočtu vzdálenosti r proto neurčí skutečnou vzdálenost přijímače od družice, ale jen tzv. zdánlivou vzdálenost (pseudorange). Řešení tohoto problému je popsáno v kapitole Princip výpočtu polohy přijímače [4].

Chce-li přijímač určit pozici v prostoru potřebuje znát zeměpisnou šířku, zeměpisnou délku, nadmořskou výšku a časovou chybu. Má tedy čtyři neznámé. Z matematiky víme, že pro výpočet čtyř neznámých potřebuje minimálně stejný počet rovnic a z toho vyplývá, že ve skutečnosti bude potřebovat signál z nejméně čtyř satelitů [4].

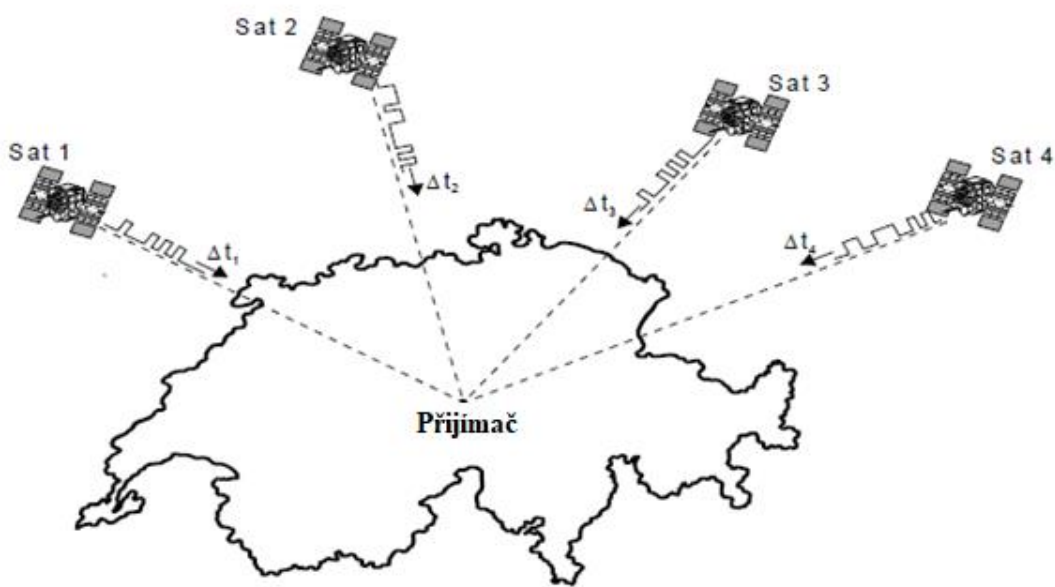
GNSS jsou zkonstruovány tak, aby na každém místě planety byl signál alespoň ze čtyř družic čili, aby bylo možné zjistit polohu na kterémkoliv místě planety (Obrázek 5) [4].



Obrázek 5 - Čtyři satelity jsou potřeba k určení polohy v prostoru [4]

2.5 Princip výpočtu polohy přijímače

Nyní už známe vše potřebné k finálnímu odvození výpočtu pozice. Víme, že musíme obdržet navigační zprávu z minimálně čtyř satelitů, abychom vypočítali časy cest signálů (obrázek 6). Také víme, že výpočty probíhají v Kartézském třídimenzionálním geocentrickém souřadném systému [4].



Obrázek 6 - Minimálně signály ze čtyř satelitů musí být přijaty [4]

Vzdálenost uživatele od satelitů R_1 , R_2 , R_3 a R_4 můžeme určit pomocí znalosti doby cesty Δt_1 , Δt_2 , Δt_3 a Δt_4 mezi satelity a uživatelem. Také známe polohy čtyř satelitů X_{Sat_i} , Y_{Sat_i} , Z_{Sat_i} [4].

Hodiny satelitů jsou synchronizovány na UTC (Universal time coordinated – Koordinovaný světový čas). Oproti tomu hodiny přijímače nejsou synchronizovány, což znamená, že čas v přijímači je buď opožděn nebo předchází. Tuto odchylku označujeme Δt_0 se zápornou hodnotou za předpokladu, že hodiny jsou pozadu a kladnou v případě že hodiny jdou napřed. Tato časová odchylka způsobuje nepřesnosti v měření doby cesty signálu a vzdálenosti. Výslednou nesprávnou hodnotu nazýváme zdánlivou vzdáleností PSR (Pseudorange) [4].

$$\Delta t_{měřená} = \Delta t + \Delta t_0, \quad (3)$$

$$PSR = \Delta t_{měřená} * c = (\Delta t + \Delta t_0) * c, \quad (4)$$

$$PSR = R + \Delta t_0 * c, \quad (5)$$

kde:

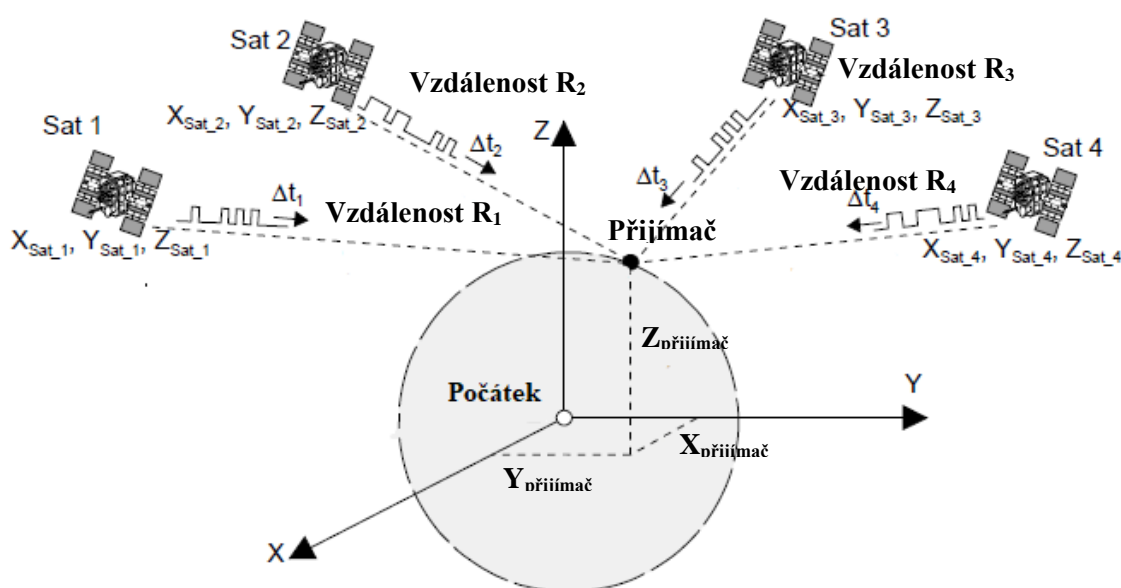
R je pravá vzdálenost přijímače od satelitu, m,

c je rychlost světla, m.s^{-1} ,

Δt je doba cesty signálu od satelitu k přijímači, s,

Δt_0 je rozdíl času satelitu a přijímače, s,

PSR je zdánlivá vzdálenost, m [4].



Obrázek 7 - Určení polohy v prostoru [4]

Vzdálenost R v Kartézském souřadném systému (obrázek 8) můžeme spočítat pomocí rovnice:

$$R = \sqrt{(X_{sat} - X_{user})^2 + (Y_{sat} - Y_{user})^2 + (Z_{sat} - Z_{user})^2} \quad [4]. \quad (6)$$

Po dosazení do rovnice (5) dostaneme:

$$PSR = \sqrt{(X_{sat} - X_{user})^2 + (Y_{sat} - Y_{user})^2 + (Z_{sat} - Z_{user})^2} + c * \Delta t_0 \quad [4]. \quad (7)$$

Pro určení čtyř neznámých (Δt_0 , X_{user} , Y_{user} a Z_{user}) potřebujeme soustavu čtyř rovnic čili platí, že pro čtyři satelity ($i = 1 \dots 4$) dostaneme matici rovnic:

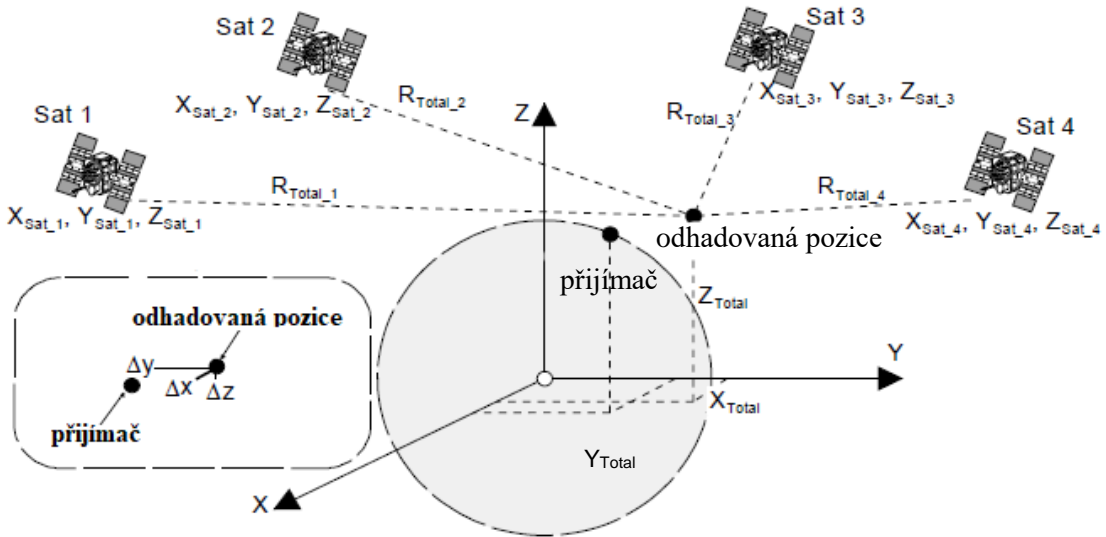
$$PSR = \sqrt{(X_{sat_i} - X_{user})^2 + (Y_{sat_i} - Y_{user})^2 + (Z_{sat_i} - Z_{user})^2} + c * \Delta t_0 \text{ [4]}. \quad (8)$$

Soustava čtyř rovnic (8) je soustavou nelineární. Abychom ji vyřešili, musíme ji nejprve zlinearizovat podle Taylerova modelu (9) a použít pouze první řád (10).

$$f(x) = f(x_0) + \frac{f'}{1!}(x_0) * \Delta x + \frac{f''}{2!}(x_0)^2 * \Delta x + \frac{f'''}{3!}(x_0)^3 * \Delta x + \dots \text{ [4]}. \quad (9)$$

$$f(x) = f(x_0) + f'(x_0) * \Delta x \text{ [4]}. \quad (10)$$

Za účelem linearizace daných čtyř rovnic musíme libovolně odhadnout hodnotu x_0 v blízkosti x . To znamená, že místo přímého výpočtu X_{user} , Y_{user} a Z_{user} budeme nejprve počítat odhadovanou hodnotu X_{Total} , Y_{Total} a Z_{Total} (obrázek 8).



Obrázek 8 - Odhadovaná pozice vůči skutečné poloze [4]

Odhadovaná hodnota zahrnuje chybu způsobenou neznámými proměnnými Δx , Δy a Δz podle rovnic:

$$X_{user} = X_{Total} + \Delta x,$$

$$Y_{user} = Y_{Total} + \Delta y,$$

$$Z_{user} = Z_{Total} + \Delta z \text{ [4]}. \quad (11)$$

Vzdálenost R_{Total} od satelitů k odhadované poloze přijímače můžeme spočítat podobným způsobem jako v rovnici (6), kdy:

$$R_{Total_i} = \sqrt{(X_{Sat_i} - X_{Total})^2 + (Y_{Sat_i} - Y_{Total})^2 + (Z_{Sat_i} - Z_{Total})^2} [4]. \quad (12)$$

Po zkombinování rovnice (12) s rovnicemi (7) a (8) dostaneme rovnici:

$$PSR_i = R_{Total_i} + \frac{\partial(R_{Total_i})}{\partial x} * \Delta x + \frac{\partial(R_{Total_i})}{\partial y} * \Delta y + \frac{\partial(R_{Total_i})}{\partial z} * \Delta z + c * \Delta t_0 [4]. \quad (13)$$

Provedením partiálních derivací dostaneme rovnici:

$$PSR_i = R_{Total_i} + \frac{X_{Total} - X_{Sat_i}}{R_{Total_i}} * \Delta x + \frac{Y_{Total} - Y_{Sat_i}}{R_{Total_i}} * \Delta y + \frac{Z_{Total} - Z_{Sat_i}}{R_{Total_i}} * \Delta z + c * \Delta t_0 [4]. \quad (14)$$

Po transponování čtyř rovnic pro $i = 1$ až 4 do matice rovnic můžeme vyřešit neznámé Δx , Δy , Δz a Δt_0 rovnicemi (13) a (14).

$$\begin{bmatrix} PSR_1 - R_{Total_1} \\ PSR_2 - R_{Total_2} \\ PSR_3 - R_{Total_3} \\ PSR_4 - R_{Total_4} \end{bmatrix} = \begin{bmatrix} \frac{X_{Total} - X_{Sat_1}}{R_{Total_1}} & \frac{Y_{Total} - Y_{Sat_1}}{R_{Total_1}} & \frac{Z_{Total} - Z_{Sat_1}}{R_{Total_1}} & c \\ \frac{X_{Total} - X_{Sat_2}}{R_{Total_2}} & \frac{Y_{Total} - Y_{Sat_2}}{R_{Total_2}} & \frac{Z_{Total} - Z_{Sat_2}}{R_{Total_2}} & c \\ \frac{X_{Total} - X_{Sat_3}}{R_{Total_3}} & \frac{Y_{Total} - Y_{Sat_3}}{R_{Total_3}} & \frac{Z_{Total} - Z_{Sat_3}}{R_{Total_3}} & c \\ \frac{X_{Total} - X_{Sat_4}}{R_{Total_4}} & \frac{Y_{Total} - Y_{Sat_4}}{R_{Total_4}} & \frac{Z_{Total} - Z_{Sat_4}}{R_{Total_4}} & c \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta t_0 \end{bmatrix} \quad [4]. \quad (13)$$

Po úpravě rovnice (13) dostaneme rovnici (14).

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta t_0 \end{bmatrix} = \begin{bmatrix} \frac{X_{Total} - X_{Sat_1}}{R_{Total_1}} & \frac{Y_{Total} - Y_{Sat_1}}{R_{Total_1}} & \frac{Z_{Total} - Z_{Sat_1}}{R_{Total_1}} & c \\ \frac{X_{Total} - X_{Sat_2}}{R_{Total_2}} & \frac{Y_{Total} - Y_{Sat_2}}{R_{Total_2}} & \frac{Z_{Total} - Z_{Sat_2}}{R_{Total_2}} & c \\ \frac{X_{Total} - X_{Sat_3}}{R_{Total_3}} & \frac{Y_{Total} - Y_{Sat_3}}{R_{Total_3}} & \frac{Z_{Total} - Z_{Sat_3}}{R_{Total_3}} & c \\ \frac{X_{Total} - X_{Sat_4}}{R_{Total_4}} & \frac{Y_{Total} - Y_{Sat_4}}{R_{Total_4}} & \frac{Z_{Total} - Z_{Sat_4}}{R_{Total_4}} & c \end{bmatrix}^{-1} \cdot \begin{bmatrix} PSR_1 - R_{Total_1} \\ PSR_2 - R_{Total_2} \\ PSR_3 - R_{Total_3} \\ PSR_4 - R_{Total_4} \end{bmatrix} \quad [4]. \quad (14)$$

Vypočtené Δx , Δy a Δz použijeme pro znovu vypočítání odhadované pozice X_{Total} , Y_{Total} , Z_{Total} podle rovnice (11).

$$X_{Total_New} = X_{Total_Old} + \Delta x,$$

$$Y_{Total_New} = Y_{Total_Old} + \Delta y,$$

$$Z_{Total_New} = Z_{Total_Old} + \Delta z \quad [4]. \quad (14)$$

Odhadované hodnoty X_{Total_New} , Y_{Total_New} a Z_{Total_New} znovu ošetříme soustavou rovnic (14). Opakovaným ošetřením zmenšujeme odchylky Δx , Δy a Δz . Proces se opakuje po celou dobu spuštění přijímače autonomně na pozadí. Výstupem přijímače jsou zprávy podle NMEA protokolu [4].

3 Komunikační protokoly NMEA, UBX

Aby mohly být proměnné jako např. pozice, rychlost, čas atd. přenášeny na periferní zařízení (např. počítač, obrazovku, vysílač) mají GNSS moduly sériové rozhraní úrovně TTL nebo RS-232. Informace jsou posílány přes rozhraní pomocí jednoho z protokolů NMEA, UBX nebo RTCM [7].

Většina přijímačů v základním nastavení pracuje v protokolu NMEA (National Marine Electronics Association). Tento protokol je standardizován Národním sdružením námořní elektroniky (NMEA). V současnosti jsou data předávána podle specifikace NMEA-0183. NMEA specifikovala sady dat pro různé GNSS. Protokol UBX je proprietární protokol firmy u-blox pracující na binární úrovni. Protokol RTCM (Radio Technical Commission for Maritime Services) je jednosměrný protokol, který se používá k napájení přijímače [7].

3.1 Struktura protokolu NMEA

Většina GNSS přijímačů v základním nastavení posílá sedm NMEA datových zpráv, kde:

- GPGGA (GPS Fix Data) obsahuje čas, zeměpisnou šířku, zeměpisnou délku, počet satelitů, nadmořskou výšku, výšku nad elipsoidem, horizontální nejistotu.
- GPGGL (Geographic Position) obsahuje čas, zeměpisnou šířku, zeměpisnou délku.
- GPGSA (GPS DOP and Active Satellites) obsahuje měřicí mód (2D nebo 3D), počet satelitů použitých k určení polohy a přesnosti měření DOP (Dilution of Precision).
- GPGSV (GNSS Satellites in View) obsahuje počet viditelných satelitů, jejich identifikace, jejich elevace a azimut, a poměr úrovně signálu k šumu (signal to noise ratio – S/N).
- GPRMC (Recommended Minimum Specific GNSS Data) obsahuje čas, zeměpisnou šířku, zeměpisnou délku, status systému, rychlost přijímače, směr a datum.
- GPVTG (Course over Ground and Ground Speed) obsahuje směr a rychlost.
- GPZDA (Time and Date) obsahuje UTC čas, datum a místní čas [4].

NMEA protokol je přenášen rychlostí 9600 Baudů pomocí tisknutelných osmibitových znaků ASCII. Přenos začíná počátečním bitem (logická nula), následovaným osmi datovými bity a koncovým bitem (logická jednička), který je přidán na konci zprávy [4].

Každá NMEA zpráva je formátována do struktury:

\$GPDTS,Inf_1,Inf_2,Inf_3,Inf_4,Inf_5,Inf_6,Inf_n*CS<CR><LF>.

Tabulka 1 - Struktura NMEA zpráv

Znak	Popis
\$	Počátek NMEA zprávy
GP	Řetězec charakteristický pro informaci z GNSS
DTS	Identifikátor druhu NMEA zprávy (např.: RMC, GGA, ...)
Inf_1 – Inf_n	Vlastní posílaná data např.: 4717.1115 pro ZŠ
,	Oddělovač jednotlivých informací
*	Klíčový znak označující kontrolní součet zprávy
CS	Kontrolní součet sloužící ke kontrole, zda nebyla zpráva při přenosu nijak porušena
<CR><LF>	<CR> posune kurzor na začátek řádku a (<LF>) posune kurzor na nový řádek

[4]

Maximální počet znaků v jedné NMEA zprávě nesmí být více než 79. Znak \$ a konečné <CR><LF> se nezapočítávají [4]. Skutečná NMEA věta může vypadat například takto: \$GPRMC,182501.0,A,4609.105,N,00638.715,E,000.03,043.4,080918,01.3,W*7D<CR><LF>

3.2 Výpočet kontrolní sumy NMEA zpráv

Kontrolní součet je určen samostatnou operací zahrnující všech osm datových bitů (mimo počáteční a koncový bit) ze všech posílaných znaků včetně oddělovačů. Operace začíná po spuštění datové sady (znakem \$) a končí separátorem kontrolního součtu (*). Osmibitový výsledek je rozdělen do dvou čtyřbitových sad (nibbles) a každá sada je převedena na odpovídající hexadecimální hodnotu (0 až 9 a A až F). Kontrolní součet se skládá ze dvou hexadecimálních hodnot převedených do znaků ASCII [4].

Princip výpočtu kontrolního součtu lze vysvětlit pomocí stručného příkladu. Byla přijata NMEA zpráva \$GPZDA,130305.2,20,06,2001,,*57<CR><LF> (57 je její kontrolní součet). Abychom ověřili, že zpráva nebyla přenosem porušena, vypočítáme její kontrolní součet a ověříme ho s kontrolním součtem v dané zprávě.

Výpočet kontrolního součtu je předveden v tabulce 2. Postup výpočtu je popsán v následujících krocích:

1. Do analýzy jsou zahrnuty pouze znaky mezi \$ a * čili jen GPZDA,130305.2,20,06,2001,,.
2. Každý znak zprávy je převed do osmi bitové hodnoty podle standardu ASCII.
3. Spočítá se počet jedniček v každém sloupci tabulky 2 a je-li počet jedniček lichý, výsledná hodnota sloupce je jedna, je-li sudý, výsledná hodnota je nula.
4. Výsledek je rozdělen do dvou čtyř bitových sad.
5. Obě sady se převedou na hexadecimální hodnotu.
6. Tyto hodnoty jsou převedeny do znaků ASCII.

7. Odpovídají-li znaky vypočítaného kontrolního součtu znakům kontrolnímu součtu v dané zprávě, přenos proběhl v pořádku [4].

Tabulka 2 – Národní výpočet kontrolní sumy

Znak	ASCII (osmibitová hodnota)							
G	0	1	0	0	0	1	1	1
P	0	1	0	1	0	0	0	0
Z	0	1	0	1	1	0	1	0
D	0	1	0	0	0	1	0	0
A	0	1	0	0	0	0	0	1
,	0	0	1	0	1	1	0	0
1	0	0	1	1	0	0	0	1
3	0	0	1	1	0	0	1	1
0	0	0	1	1	0	0	0	0
3	0	0	1	1	0	0	1	1
0	0	0	1	1	0	0	0	0
5	0	0	1	1	0	1	0	1
.	0	0	1	0	1	1	1	0
2	0	0	1	1	0	0	1	0
,	0	0	1	0	1	1	0	0
2	0	0	1	1	0	0	1	0
0	0	0	1	1	0	0	0	0
,	0	0	1	0	1	1	0	0
0	0	0	1	1	0	0	0	0
6	0	0	1	1	0	1	1	0
,	0	0	1	0	1	1	0	0
2	0	0	1	1	0	0	1	0
0	0	0	1	1	0	0	0	0
0	0	0	1	1	0	0	0	0
1	0	0	1	1	0	0	0	1
,	0	0	1	0	1	1	0	0
,	0	0	1	0	1	1	0	0
Exkluzivní disjunkce	0	1	0	1	0	1	1	1
Sady	0101				0111			
Hexadecimální hodnota	5				7			
SCII CS znaky	5				7			

[4]

4 Převod souřadnic z NMEA vět do souboru GPX

Abychom mohli zobrazit polohu či dráhu přijímače, musíme přijatá data nejprve upravit do vhodné podoby a následně převést do formátů podporovaných mapovými systémy. Většina mapových serverů (Google Maps, Google Earth, mapy.cz, atd.) podporují formáty KML (Keyhole Markup Language) a GPX [4].

KML soubor má formát XML. Primárně je určen pro publikaci a distribuci geografických dat. KML má strukturu založenou na tagu s názvy a atributy (bod, linie, plocha, barva atd.) pro speciální grafické zobrazení [4].

GPX formát je schéma XML navržené jako běžný datový formát GPS pro softwarové aplikace. Může být použit k popisu trasových bodů a celých tras. Údaje o poloze (a případně údaje o výšce, čase a dalších informacích) jsou ukládány podobným způsobem jako formát KML, a to pod tagy. GPX formát jsem zvolil, protože jej podporuje převážná většina mapových portálů, a navíc umožňuje zobrazení celých tras s časovými údaji [8].

NMEA protokol obsahuje zeměpisnou šířku ve formátu dddd.dddd,h (d = digit = číslice, h = hemisphere = polokoule ve vertikální rovině, sever nebo jih) a délku ve formátu dddd.dddd,h (d = digit = číslice, h = hemisphere = polokoule v horizontální rovině, západ nebo východ). První dvě/tři číslice reprezentují hodnotu ve stupních, zbylých šest reprezentuje minuty. Tečka značí desetinnou čárku čili 4832.5421 znamená 48° 32,5421'. Abychom tuto hodnotu byli schopní zapsat do GPX/KML formátu a zobrazit v online mapách musíme ji převést do stupňů podělením minut šedesáti: $48^{\circ} 32,5421' = (48 + 32,5421 / 60)^{\circ} = 48,542368^{\circ}$ [4].

Jelikož chceme zobrazit celou zaznamenanou trasu, a ne jen jeden bod, budeme muset upravená data zabalit tak, aby je byly mapové portály schopné zobrazit. K zabalení využijeme XML formát Garmin Connect, který vypadá následovně:

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.1" creator="Garmin Connect"
xsi:schemaLocation="http://www.topografix.com/GPX/1/1
http://www.topografix.com/GPX/1/1/gpx.xsd
http://www.garmin.com/xmlschemas/GpxExtensions/v3
http://www.garmin.com/xmlschemas/GpxExtensionsv3.xsd
http://www.garmin.com/xmlschemas/TrackPointExtension/v1
http://www.garmin.com/xmlschemas/TrackPointExtensionv1.xsd"
xmlns="http://www.topografix.com/GPX/1/1"
xmlns:gpstpx="http://www.garmin.com/xmlschemas/TrackPointExtension/v1"
xmlns:gpxx="http://www.garmin.com/xmlschemas/GpxExtensions/v3"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<metadata>
  <link href="connect.garmin.com">
    <text>Garmin Connect</text>
  </link>
  <time>2019-01-01T23:22:51.000Z</time>
</metadata>
<trk>
  <name>Untitled</name>
  <trkseg>
    <trkpt lon="16.013914" lat="49.454183">
      <time>2019-01-21T08:40:57.000Z</time>
    </trkpt>
    <trkpt lon="16.013685" lat="49.452851">
      <time>2019-01-21T08:41:10.000Z</time>
    </trkpt>
  </trkseg>
</trk>
</gpx>
```

Čili budeme na microSD kartu ukládat souřadnice, datum a čas následujícím způsobem. Zapišeme tag ukazující na bod trasy trkpt a v něm nastavíme atribut lon, který obsahuje

zeměpisnou délku a `lat` v němž je odpovídající zeměpisná šířka. Vnořený element `time` obsahuje datum a čas ve formátu YYYY-MM-DDTHH:MM:SS.SSSZ. Sekundy jsou zapsány na tři desetinná místa a zakončená velkým písmenem Z datum a čas jsou odděleny znakem T. Data zapsaná na microSD by měla vypadat následovně:

```
<trkpt lon="16.537914" lat="49.194183">  
<time>2019-01-21T08:40:57.000Z</time></trkpt>  
<trkpt lon="16.537832" lat="49.194225">  
<time>2019-01-21T08:41:10.000Z</time></trkpt>  
<trkpt lon="16.537859" lat="49.194237">  
<time>2019-01-21T08:41:17.000Z</time></trkpt>  
<trkpt lon="16.537910" lat="49.194241">  
<time>2019-01-21T08:41:24.000Z</time></trkpt>
```

Když budeme chtít data zobrazit v mapovém portále, manuálně je vykopírujeme z microSD karty a vložíme do souboru GPSTracker mezi tagy `<trkseg></trkseg>`. Soubor je převede do GPX formátu výše popsaným způsobem. Soubor uložíme a nainportujeme do mapového portálu.

5 Popis jednotlivých komponent

5.1 Arduino Nano

“Arduino Nano je jednodeskový počítač založený na mikrokontroléru ATmega od firmy Atmel.” [9] Má stejnou funkčnost jako rozšířenější Arduino Uno, ale je menší. Postrádá DC napájecí konektor a pracuje s Mini-B USB kabelem namísto standardního. Mikrokontrolér lze programovat pomocí softwaru Arduino.ino z počítače přes USB rozhraní. Přibližná cena za kopii je 50 Kč [10].

Arduino Nano lze napájet přes mini-USB konektor, přes pin Vin napětím 6-20 V nebo 5 V ostatními digitálními piny. Jako zdroj energie je automaticky vybráno nejvyšší napětí [10].

Deska má celkem 14 digitálních pinů. Jednotlivé piny lze použít jako vstupní nebo výstupní. Pracovní napětí je 5 V. Každý pin může poskytovat nebo přijímat maximálně 40 mA. Kromě toho některé piny mají specializované funkce:

- Sériový kanál na pinech 0 (RX) a 1 (TX). Používá se pro příjem (RX) a přenos (TX) TTL dat.
- PWM (Pulse Width Modulation) na pinech 3, 5, 6, 9, 10 a 11 poskytují 8 - bitový PWM výstup s funkcí analogWrite.
- Komunikace SPI na pinech 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).
- Signalizační LED připojenou k digitálnímu pinu 13. Když je pin HIGH, LED svítí, ale když je pin LOW, je vypnut [10].

Arduino Nano má 8 analogových vstupů, z nichž každý poskytuje 10 bitů rozlišení (tj. 1024 různých hodnot). Ve výchozím nastavení měří od země na 5 voltů, ačkoli je možné změnit horní hranici jejich rozsahu pomocí funkce analogReference [10].

Arduino Nano pro komunikaci s počítačem a jinými zařízeními používá sériovou komunikaci UART TTL (5V), která je k dispozici na digitálních pinech 0 (RX) a 1 (TX). FTDI FT232RL na desce vysílá tuto sériovou komunikaci přes USB a ovladače FTDI (dodávané s programem Arduino) poskytující virtuální počítačový port v počítači. Software Arduino obsahuje sériový monitor, který umožňuje zasílání jednoduchých textových zpráv do a z desky Arduino. LED diody RX a TX na desce indikují přenos dat přes čip FTDI a USB připojení k počítači (ale ne pro sériovou komunikaci na pinech 0 a 1). Knihovna SoftwareSerial umožňuje sériovou komunikaci na libovolném digitálním pinu Arduina. ATmega328 podporuje také komunikaci I2C a SPI. Software Arduino obsahuje knihovnu Wire pro zjednodušení používání sběrnice I2C. Pro použití komunikace SPI musíme použít knihovnu SPI.h [10].

Tabulka 3 - Technická data Arduino Nano

Mikrokontrolér	ATmega328
Provozní napětí	5 V
Vstupní napětí (doporučeno)	7 – 12 V
Počet digitálních I / O pinů	14 (z toho 6 poskytuje PWM výstup)
Počet analogových vstupů	8
Proudové zatížení na 1 pin	40 mA
Flash paměť	32 KB
SRAM	2 KB
EEPROM	1 KB
Bootloader	2 KB
Rychlost hodin	16 MHz
Velikost desky	18 x 45 mm
Hmotnost	7 g

[10]

5.2 GPS modul u-blox NEO-7M-0-000

NEO-7M je GNSS zařízení podporující systémy GPS, GLONASS, QZSS a po spuštění a aktualizaci firmwaru i systém Galileo. Modul se skládá z antény, RF bloku, digitálního bloku, napájecího bloku a komunikačního rozhraní (obrázek 13). RF blok obsahuje pásmovou propust, LNA zesilovač, oscilátor a frekvenční směšovač. Digitální blok se skládá z procesoru GNSS u-blox 7, krystalu RTC, ROM a RAM paměti, digitálního filtru a volitelné FLASH paměti pro lepší programovatelnost a flexibilitu. RTC zprostředkovává referenční čas pro přijímač pomocí 32 kHz oscilátoru [11].

Principem funkce zařízení je příjem signálů (čas a datum) zaslaných satelity. Tyto signály přijímač zpracovává a následně z nich pomocí funkcí popsaných v podkapitole Princip výpočtu polohy přijímače vypočítává parametry NMEA protokolu (souřadnice, nadmořská výška atd.). V základním nastavení vypočtená data zformátuje do NMEA protokolu a odešle pinem TX. Mimo protokol NMEA modul také podporuje speciální binární protokol firmy u-blox zvaný UBX, a protokol organizace RTMC. Všechny protokoly jsou dostupné v rozhraních UART, USB, DDC a SPI [11].

Řízení napájení obstarává Power Management Unit, která přepíná mezi třemi provozními režimy – hledání pozice, Nepřetržitý režim a Úsporný režim. Při hledání pozice zařízení pracuje v plném výkonu, aby byl proces nalezení co nejrychlejší a s vysokou citlivostí. Po nalezení pozice systém přepne do Nepřetržitého režimu, který zajišťuje nižší spotřebu při sledování pozice. Úsporný režim zajišťuje optimální spotřebu elektrické energie na úkor výkonu a lze jej aktivovat pomocí odpovídajících UBX. Nelze jej však využít pro systém GLONASS. Napájecí blok PMU

obsahuje stejnosměrný měnič napětí, který snižuje napájecí napětí převyšující 2,5 V. Doporučené napětí je 1,65 – 3,6 V [11].

Teploty prostředí by neměly jít pod -40 °C a nad 80 °C. Průměrná doba do nalezení polohy ze studeného startu je 30 s. Maximální četnost posílání zpráv je 10 Hz. Rychlost komunikace v základním nastavení je 9600 baudů. Modul má zabudovanou LED, která bliká frekvencí 1 Hz při zjištění času. Průměrné spotřeby jsou v tabulce 4. Cena modulu se pohybuje okolo 100 Kč. Rozměry modulu jsou 35 mm na délku a 25 mm na šířku. Anténa měří 25x25 mm [11].

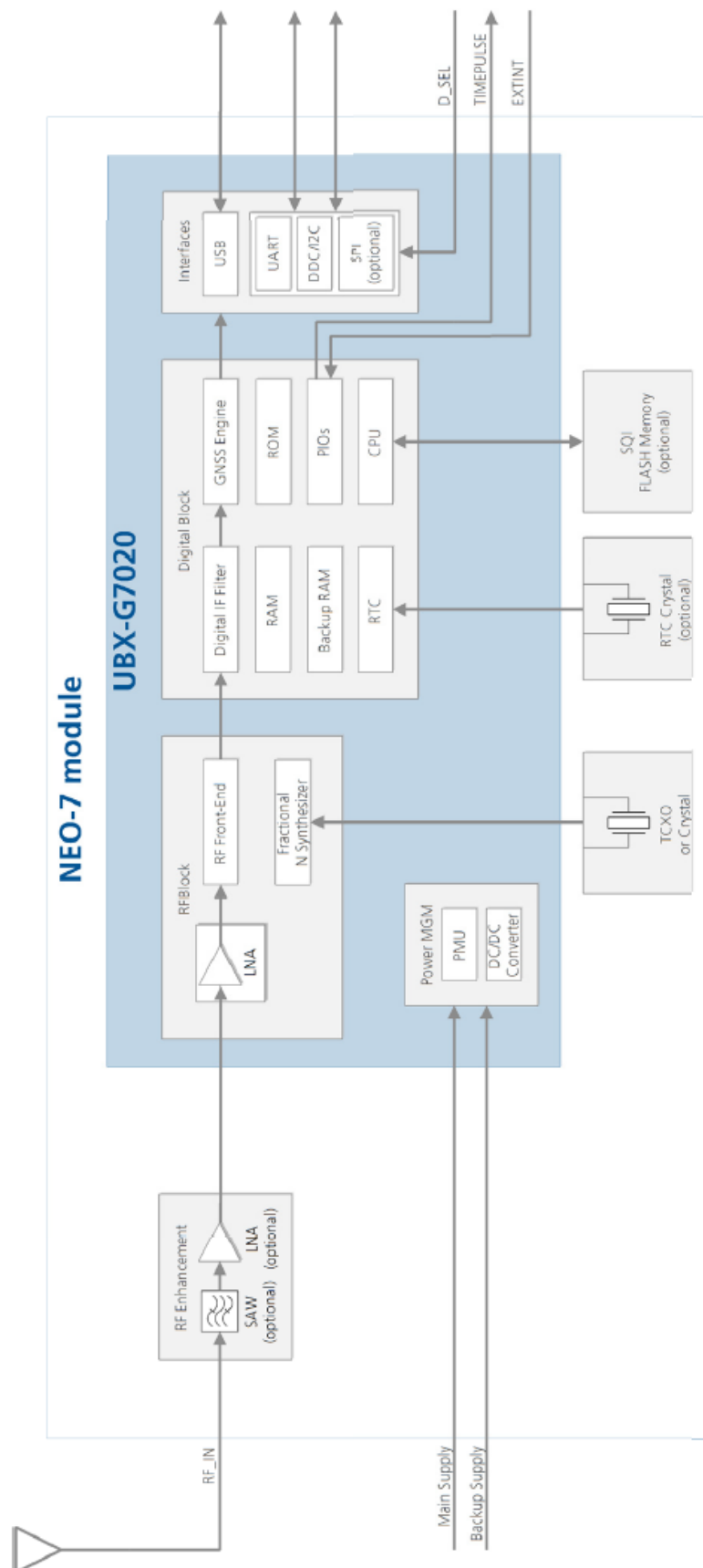
Tabulka 4 - Průměrná spotřeba elektrického proudu modulu NEO-7M

	Režim	Proud v mA (při 3 V)
Průměrná spotřeba	Při hledání pozice	22
	Při sledování pozice (Nepřetržitý režim)	17
	Při sledování pozice (Úsporný režim)	5
Maximální spotřeba		67

[11]

Antény lze použít dvě, a to buď pasivní nebo aktivní. Využil jsem anténu aktivní. Dle dokumentace anténa má minimální zisk 15 dB, maximální zisk 50 dB a maximální S/N poměr 1.5 dB [11].

Tento modul jsem vybral zejména pro jeho velmi příznivou cenu a faktu, že nalézá pozici rychleji než starší verze NEO-6M. Navíc dokáže nalézt pozici i za zhoršených viditelnostních podmínek.



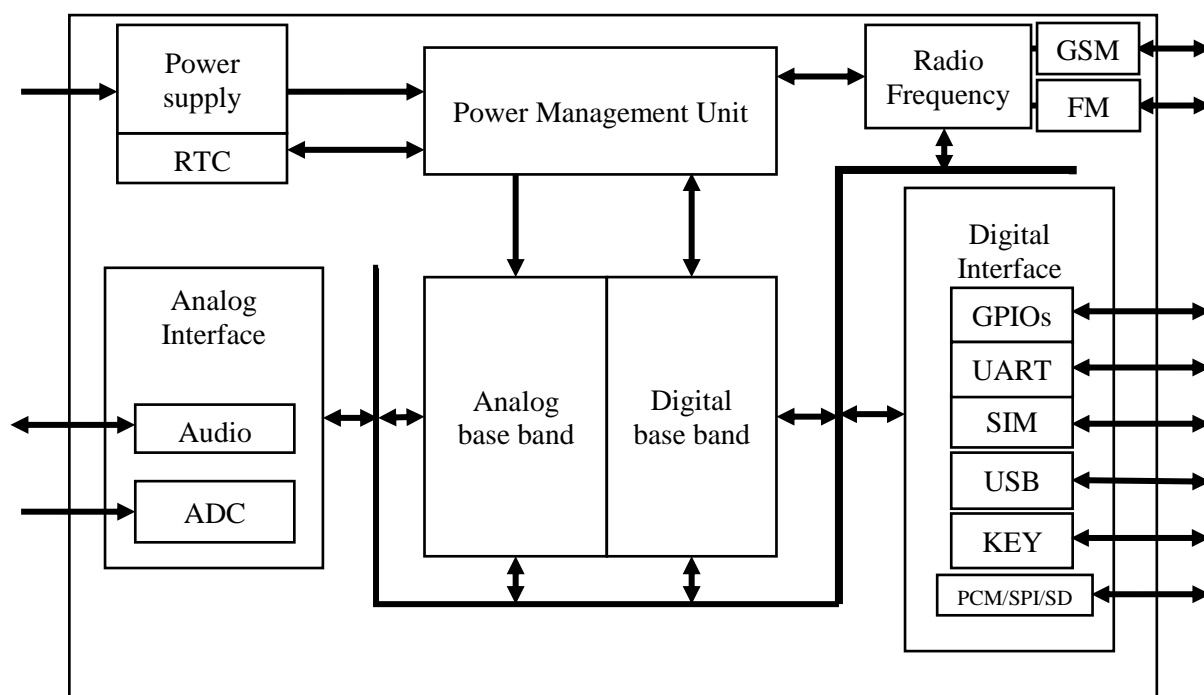
Obrázek 9 - GPS module NEO-7M [11]

5.3 GSM modul SIM800L

SIM800L je čtyř pásmový GSM/GPRS modul, který pracuje ve frekvencích 850 MHz, 900 MHz, 1800 MHz a 1900 MHz. Modul podporuje GPRS komunikaci. Doporučené napájecí napětí je mezi 3,4 V a 4,4 V. Rozhraní pro SIM kartu pracuje v napětích 1,8 V nebo 3 V. Teplota prostředí by neměla překročit 85 °C a klesnout pod -40 °C. Provozní režimy jsou popsány v tabulce 5 [12].

Modul se skládá z napájecí jednotky, rozhraní pro audio s deseti bitovým A/D převodníkem, sériových portů pro přenos dat do jiných zařízení (protokol UART), rozhraním SIM karty, antény, signalizační LED, programovatelných vstupů a výstupů GPIO a mikrokontroléru (obrázek 14). S modulem komunikujeme přes rozhraní na pinech RX a TX pomocí AT příkazů. Rx pin slouží pro příjem zpráv a Tx pro jejich odesílání [12].

SIM (Subscriber Identity Module) kartu jsem použil od firmy SAZKA mobil, pro její výhodnou cenu. Pořízení této SIM karty vyšlo na 75 Kč s počátečním kreditem 150 Kč. 50 MB dat na jeden měsíc vyšlo na 50 Kč.



Obrázek 10 - GSM modul SIM800L schéma [12]

Tabulka 5 - Provozní režimy GSM modulu SIM800L

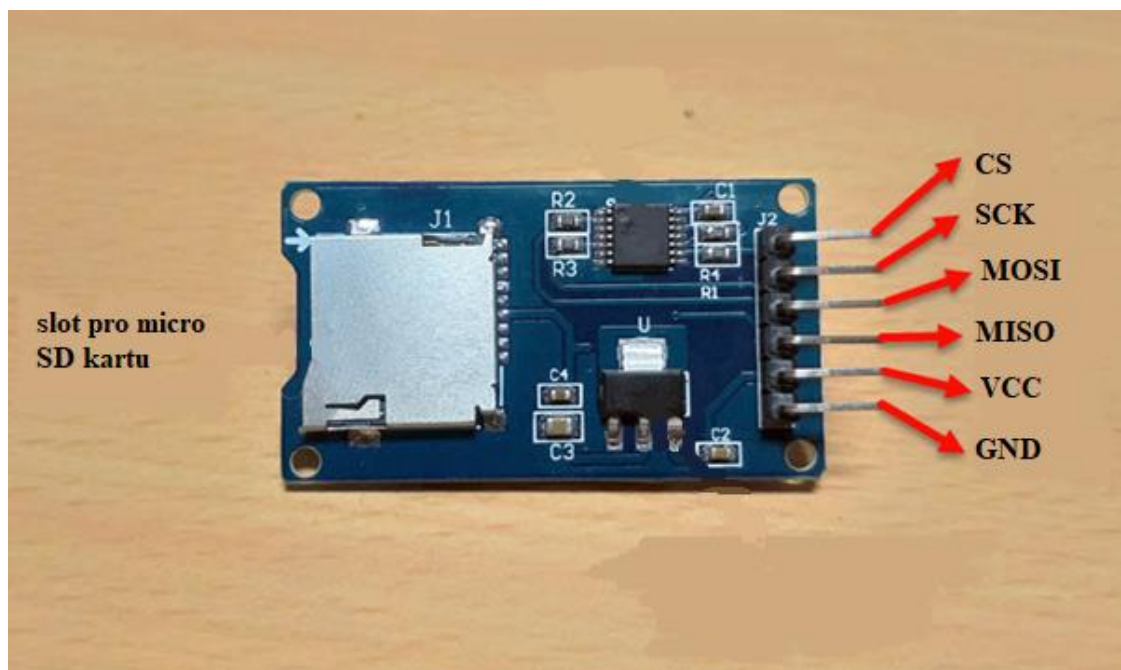
Režim	funkce	Maximální spotřeba [mA]
GSM/GPRS spánek	Modul přejde automaticky do režimu spánku, pokud mu to podmínky pro tento režim dovolí a nejsou žádná data na sériovém portu. Modul může přijímat zprávy SMS.	1,88
GSM nečinný	Software je aktivní. Modul je registrován v síti GSM a připraven komunikovat.	18,7
GSM hovor	Probíhá spojení mezi dvěma uživateli.	216.12
GPRS připraveno	Modul je připraven pro přenos dat přes GPRS, ale žádná data nejsou momentálně přijímána nebo odesílána.	
GPRS data	Probíhá přenos dat přes GPRS (PPP, TCP nebo UDP). Spotřeba závisí na druhu přenosu.	453,57
GSM připojování	Připojování do GSM sítě	2000

[12]

5.4 MicroSD Card Reader/Writer Adapter v1.0 od CATALEX

Modul slouží k zápisu dat na microSD kartu a jejich čtení. Modul komunikuje pomocí protokolu SPI. Zařízení je v pozici Slave. Modul má celkem 6 pinů: napájecí pin VCC na 5 V, GND pin (zem), MOSI – Master OUT Slave IN (vstup), MISO – master IN Slave OUT, SCK – SPI hodiny a CS – chip select (input, slouží k adresaci zařízení). Jelikož microSD karty pracují na napětí 3,3 V je v modulu zabudován 3,3 V napěťový regulátor [13].

Tento modul jsem zvolil pro jeho velice příznivou cenu 30 Kč. Lze však použít modul i bez napěťového regulátoru, jelikož Arduino má i výstup pro 3,3 V. Varianta bez regulátoru by byla vhodnější, protože díky absenci regulátoru se značně zredukuje jeho velikost. Modul je na obrázku 15.



Obrázek 11- MicroSD Card Reader/Writer [13]

5.5 Napájecí blok

Jelikož doporučená napětí pro GSM modul je mezi 3,6 V a 4,4 V, a Arduino má výstupy pouze pro 5 V a 3,3 V, je nutné do systému zahrnout prvek regulující napětí na požadovanou úroveň. Dalším problémem je, že modul vyžaduje ve špičkách proudy až 2 A, přičemž Arduino je schopné poskytnout maximálně 40 mA na pin. Problém jsem vyřešil připojením lithiového článku o napětí 3,7 V, který sníží napětí a pokryje proudové smyčky. Tímto řešením však vznikne další potíž. Když odpojíme systém od napájení, GSM poběží dál, dokud nevybije akumulátor, což je pro nás nežádoucí, proto do obvodu připojíme spínací relé, které při odpojení napájení rozepne napájení GSM modulu.

Regulátor napětí se skládá ze vstupní části, na kterou se přivede 5 V z Arduina, části pro nabíjení baterie, která sníží napětí na 4,4 V a výstupu na kterém můžeme regulovat napětí pomocí proměnného rezistoru. Spínací relé jsem použil od firmy HKE HRS1H-S-DC5V pro operační napětí 5 V a maximální proud 3 A, což je pro naše účely ideální.

6 Sestrojení systému

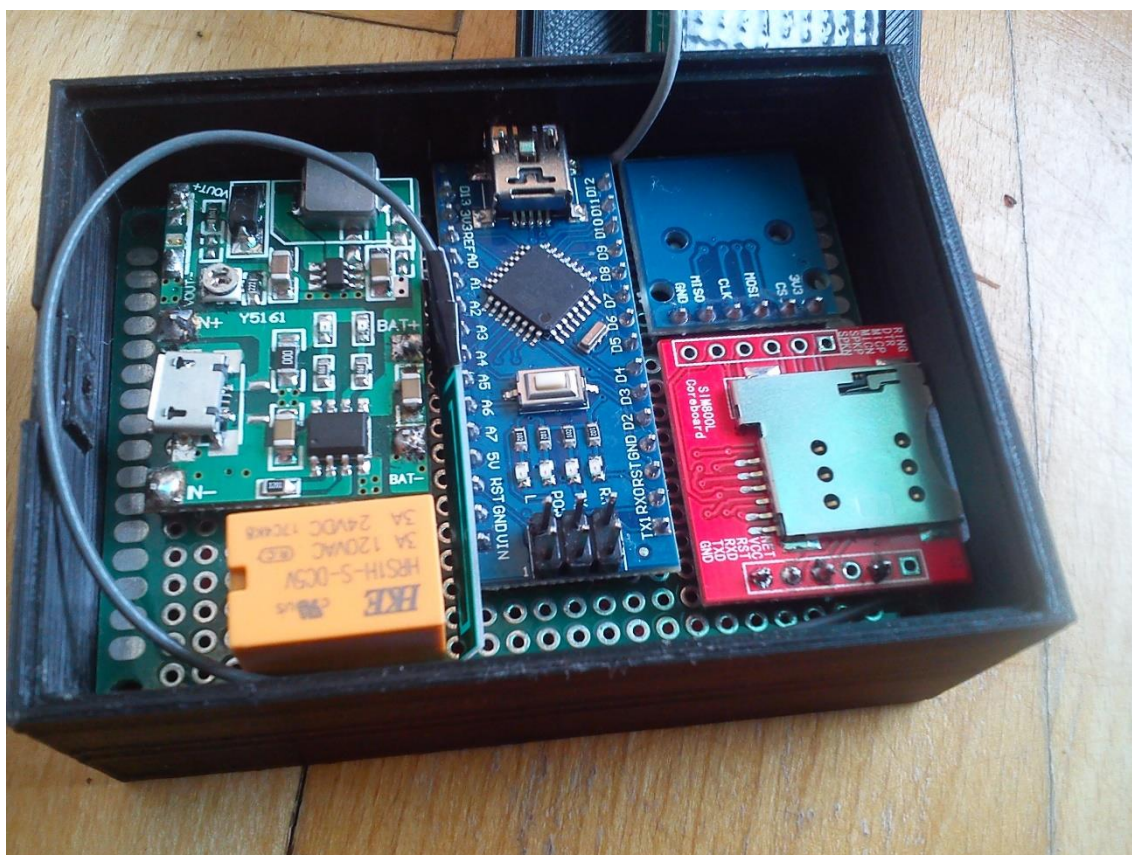
Prvním krokem sestrojení systému bylo zprovoznění jednotlivých komponent. K tomuto účelu jsem využil nepájivého pole, propojovacích vodičů, stolního počítače, vývojového prostředí Arduino.ino a výše zmíněných komponent.

Zprovoznění GNSS modulu spočívalo v zobrazení posílaných NMEA zpráv a jejich zpracování. Ke zpracování NMEA zpráv jsem se snažil využít na internetu dostupných knihoven, narazil jsem však na několik problémů. Hlavním problémem bylo, že knihovny nefungovaly, či fungovaly jen částečně s mým GNSS modulem. Dalšími problémy byla jejich značná velikost a složitost, kdy mnohdy ukládaly do proměnných pro mé účely zbytečné proměnné a zabíraly tak cennou paměť Arduina. Problémy jsem vyřešil sestrojením vlastních funkcí pro zpracování přijímaných NMEA zpráv. Funkce jsou popsány v kapitole Vývojový diagram a důležité části kódu a podkapitolách Funkce processGPS, Funkce calcChecksum a Funkce decodeLine.

Zprovoznění modulu pro microSD kartu spočívalo v otestování funkce jednoduchého řetězce na microSD kartu pomocí knihovny SD.h. Po úspěšném otestování následovalo propojení a naprogramování systému s Arduinem a GNSS modulem tak, aby se při spuštění a získání souřadnic vytvořil adresář s datumem daného měření na microSD kartě a do tohoto adresáře se vytvořil adresář času počátku měření, do kterého se zapisují potřebná GPS data. Toto řešení má za úkol, aby se po vypnutí a znovu zapnutí systému v tentýž den data ukládala do adresáře stejného datumu a vytvořil se adresář nový s časem počátku nové cesty (měření) čili aby každá nová cesta byla zaznamenána zvlášť. Funkce je popsána v podkapitole Funkce sendOnSD.

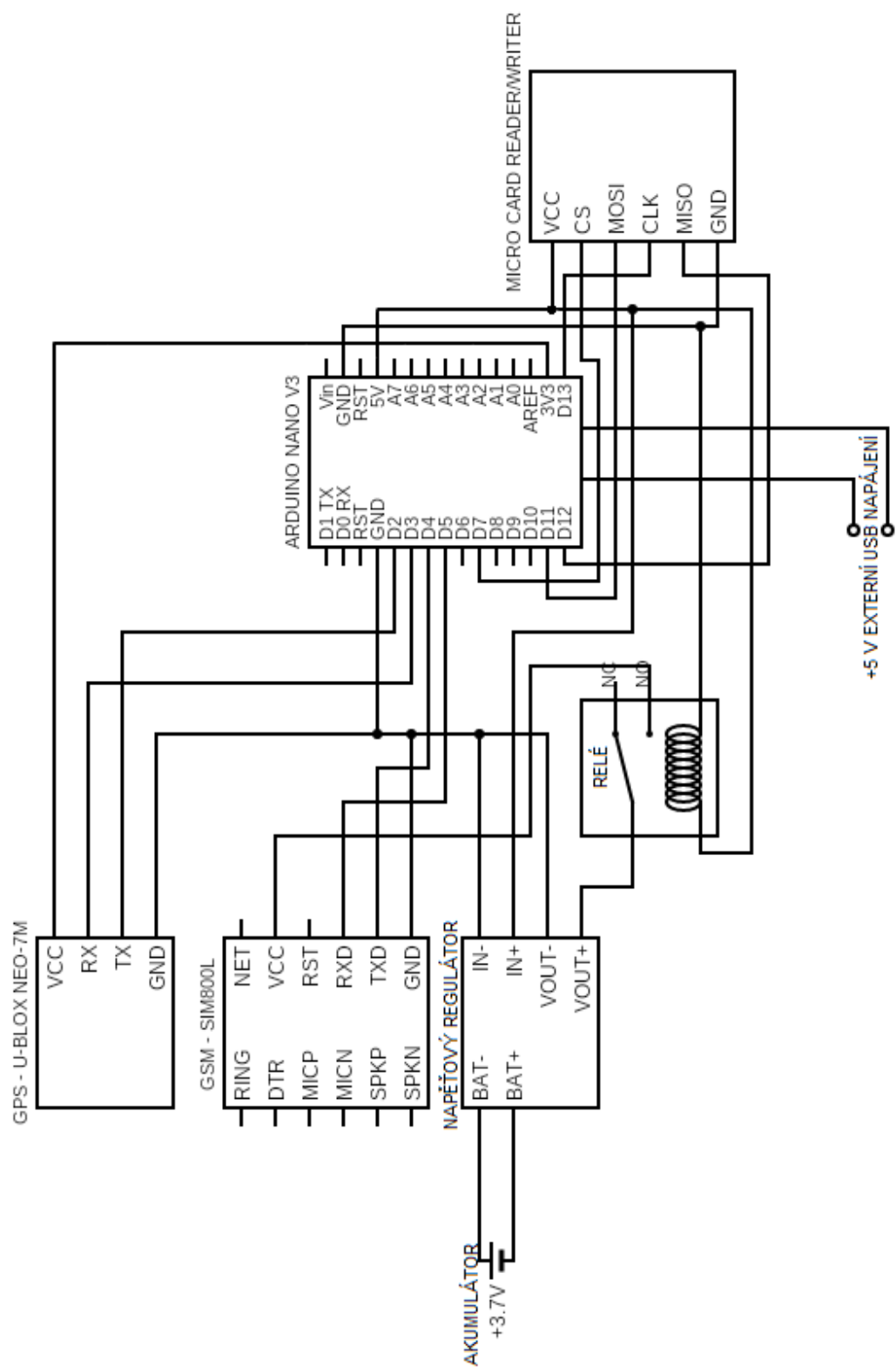
Zprovoznění modulu GSM doprovázelo hned několik problémů. Prvním problémem bylo, že modul by měl být napájen napětím v rozmezí 3,4 a 4,4 V. Arduino však poskytuje napětí buď 3,3 nebo 5 V. Druhým problémem bylo, že se SIM karta nemohla připojit do sítě GSM. Důvodem byla velká spotřeba proudu při navazování spojení s vysílači GSM, kdy Arduino nebylo schopné pokrýt proudové špičky až 2 A. Oba problémy jsem vyřešil napájecím blokem, který poskytl napětí přibližně 4 V a pokryl proudové špičky. Pro komunikaci se serverem jsem zkusil použít na internetu dostupných knihoven, nicméně jsem se potkal s podobnými problémy jako u knihoven pro GNSS modul, kdy s mým modulem nefungovaly vůbec nebo jen částečně. Problém jsem opět vyřešil napsáním vlastní funkce pro odesílání dat na server. Funkce je popsána v podkapitole Funkce gsmSendHttp. O zpracování dat serverem se postaraly na něm běžící programy z příloh D, E, F a G. Zpracování dat serverem je popsáno v kapitole Zpracování dat serverem a zobrazení poslední polohy.

Po úspěšném otestování všech komponent následovalo jejich vzájemné propojení podle obrázku 13 pomocí nepájivého pole a doprogramování celého systému. Po úspěšném sestrojení a odzkoušení zkušební verze následovalo sestrojení finální verze napájením jednotlivých komponent na univerzální desku plošných spojů též podle obrázku 13. Sestrojení bylo završeno navrhnutím a vytisknutím schránky pro finální zařízení. Vývojový diagram a popis důležitých částí kódu naleznete v kapitole Vývojový diagram a důležité části kódu, kód celého programu pak v příloze B.



Obrázek 12 – GPS Tracker, pohled zevnitř

Pro navrhnutí schránky pro sestrojené zařízení jsem využil program Autodesk Fusion 360 ve studentské licenci. V návrhu jsem se snažil zohlednit účel a požadavky z kapitoly Účel systému, požadavky na něj a jeho návrh. Navrženou schránku jsem vytisknul na 3D tiskárně a následně vyvrtal čtyři otvory ve víku tak, aby bylo možné přišroubovat anténu do místa vybrání. K přišroubování antény jsou potřeba čtyři šrouby průměru jeden milimetr. Víko je zajištěno v uzavřené poloze jedním šroubem průměru dva milimetry a zaklesnutím do vybrání v těle schránky. Technické výkresy schránky jsou v příloze A. Pohled dovnitř zařízení je na obrázku 12. Vnější pohledy jsou na obrázcích 14 a 15.



Obrázek 13 – Schéma zapojení systému



Obrázek 14 – GPS Tracker, pohled zepředu

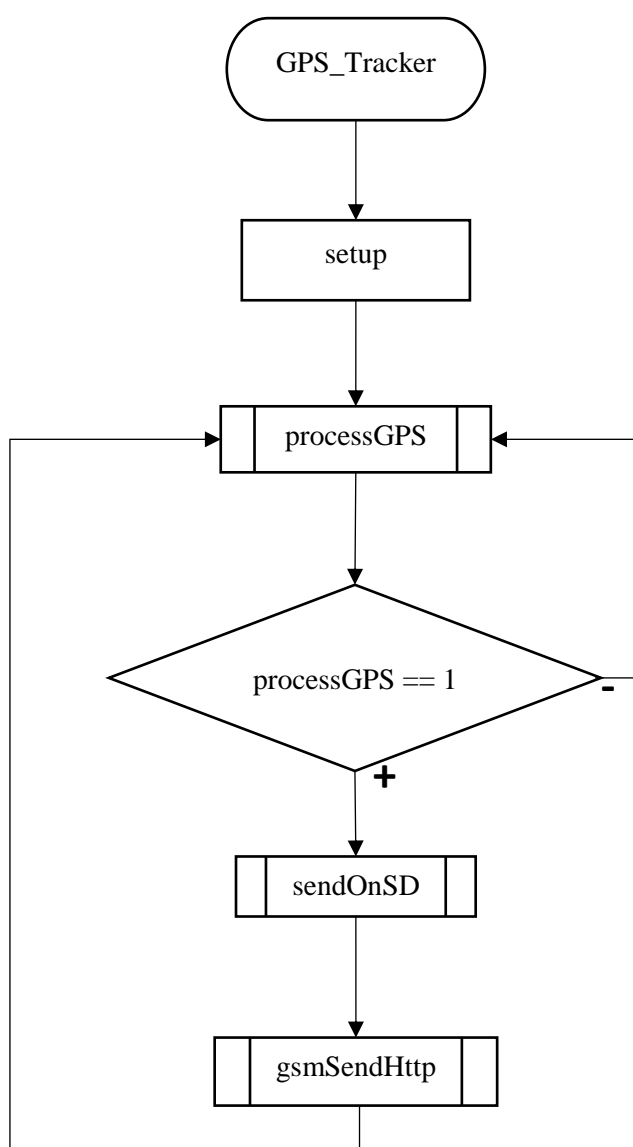


Obrázek 15 – GPS Tracker, pohled z boku

7 Vývojový diagram a důležité části kódu

Program jsem pojmenoval GPS_Tracker. Po spuštění a vytvoření globálních proměnných funkce `setup` zahájí sériovou komunikaci s microSD modulem a GPS přijímačem. Funkce `loop` je nekonečná smyčka, ve které se provádí funkce `processGPS`, `sendOnSD` a `gsmSendHttp`. Funkce `processGPS` se stará o zpracování dat poslaných GPS přijímačem. Pokud funkce `processGPS` úspěšně zpracuje data, funkce `sendOnSD` je zapíše na microSD kartu a funkce `gsmSendHttp` je odešle na server.

Vývojový diagram programu je na obrázku 16. Kód programu GPS_Tracker je součástí přílohy B.



Obrázek 16 - Vývojový diagram programu GPS_Tracker

V programu jsem využil knihovny `SoftwareSerial.h`, `SD.h`, `SPI.h` a `CStringBuilder`. Knihovnu `SoftwareSerial.h` jsem použil k sériové komunikaci mezi Arduinem a GPS modulem a mezi Arduinem a GSM modulem. Pro čtení a zapisování dat na SD kartu jsem využil knihovnu `SD.h`. Knihovna `SPI` mi umožnila komunikovat s SD kartou pomocí `SPI` protokolu. Dále jsem použil knihovnu `CStringBuilder.h`, pomocí níž se skládá řetězec pro odeslání dat přes GSM tak, aby zabíraly co nejméně místa v paměti.

Pro práci s GNSS přijímačem jsem se snažil využít již existující knihovny, narazil jsem však na problémy s kompatibilitou s různými GNSS přijímači. Sestrojil jsem tedy vlastní program pro zpracování dat posílaných GNSS přijímačem. Program je tvořen třemi funkcemi: `processGPS` a v něm vnořených `calcChecksum` a `decodeLine`.

7.1 Funkce `processGPS`

Funkce `processGPS` se stará o zpracování dat, která posílá GPS přijímač. Vývojový diagram funkce je na obrázku 17. Klíčové proměnné jsou `position`, `NMEA_HEADER`, `c` a `line`. Proměnná `position` udává v jakém místě NMEA věty se nachází. Do proměnné `c` se čtou jednotlivé znaky věty. Proměnná `NMEA_HEADER` se používá k porovnání s načtenou hlavičkou, jestli se jedná o námi čtenou větu. Do proměnné `line` ukládáme požadovanou NMEA větu.

Rozhodl jsem se zpracovávat NMEA větu `GPRMC`, jelikož obsahuje všechny potřebná data. `GPRMC` věta může vypadat následovně:

```
$GPRMC,182501.0,A,4609.105,N,00638.715,E,000.03,043.4,080918,01.3,W*7D<CR><LF>.
```

Struktura věty je popsána v tabulce 6.

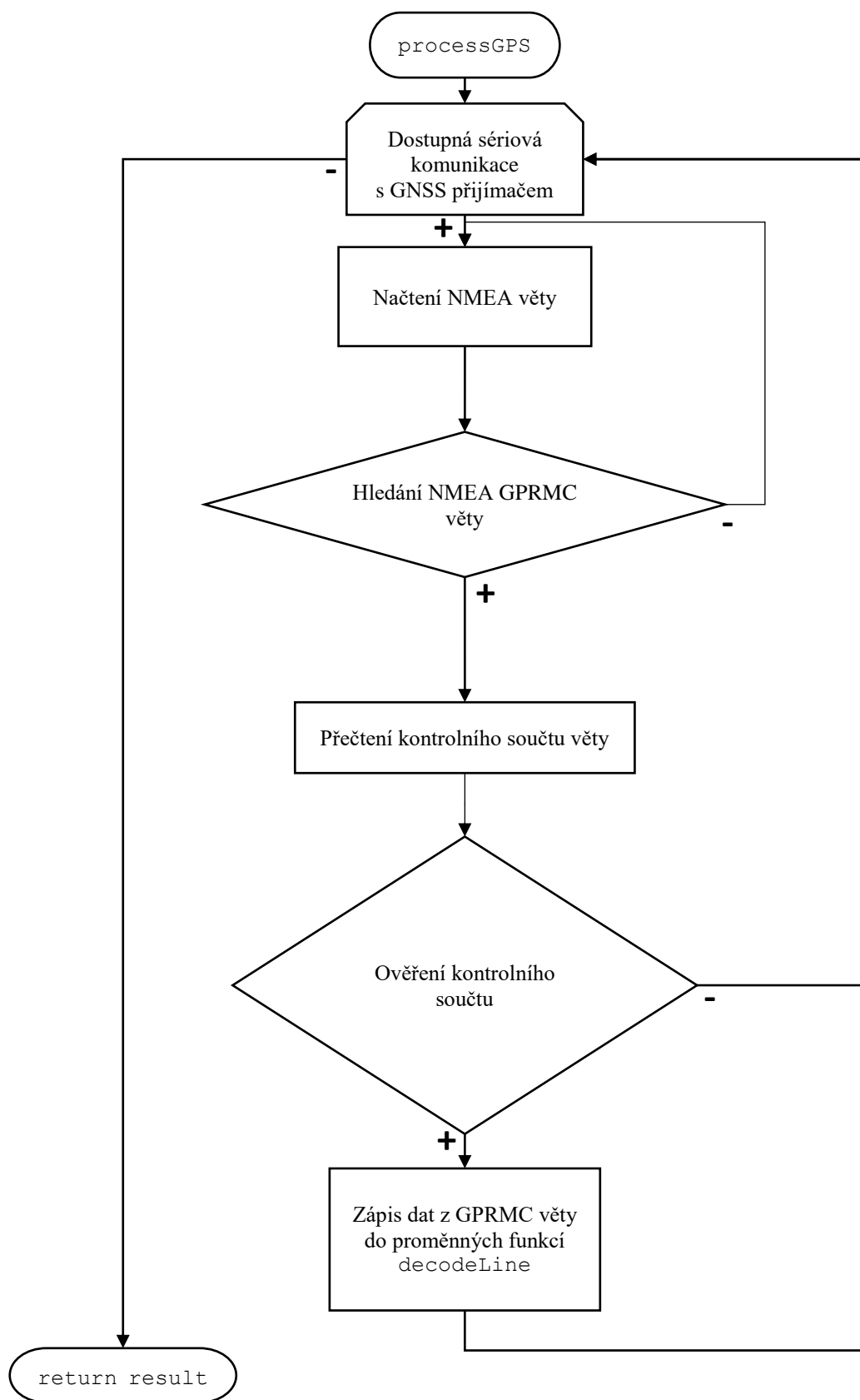
Tabulka 6 - Struktura GPRMC věty

Řetězec	Popis
\$	Počátek NMEA zprávy
GP	Řetězec charakteristický pro informaci z GNSS
RMC	Identifikátor druhu NMEA zprávy (Recommended Minimum Specific GNSS Data)
,	Oddělovač jednotlivých informací
182501.0	UTC čas přijetí, 18h 25m 01,0s
A	Kvalita dat, A jsou platná data, V jsou neplatná data
4609.105	Zeměpisná šířka, 46° 9,105'
N	Severní nebo jižní zeměpisná šířka, N je severní, S je jižní
00638.715	Zeměpisná délka, 6° 38,715'
E	Západní nebo východní zeměpisná délka, W je západní, E je východní
000.03	Rychlost v uzlech
043.4	Kurz, 43,4°
080918	Datum, 8. 9. 2018
01.3	Upravená deklinace, 1,3°
W	Směr deklinace, W je západní, E je východní
*	Klíčový znak označující kontrolní součet zprávy
CS	Kontrolní součet sloužící ke kontrole, zda nebyla zpráva při přenosu nijak porušena
<CR><LF>	<CR> posune kurzor na začátek řádku a (<LF>) posune kurzor na nový řádek

[4]

Funkce začíná smyčkou, která běží, dokud GPS přijímač posílá data. Přijímač posílá packety frekvencí jeden hertz čili funkce se ukončí po načtení prvního packetu a následně se provádí další částí programu. Pro jednoduchost program používá pouze jeden typ NMEA vět a to \$GPRMC, jelikož obsahuje všechny potřebná data (čas, datum, zeměpisná šířka a zeměpisná délka). Do proměnné `c` funkce načte znak z přijatého packetu a pomocí proměnné `position` otestuje, v jakém místě packetu se nachází. Je-li pozice menší než sedm, znamená to, že čte znaky hlavičky. Jednotlivé znaky hlavičky funkce porovná s odpovídajícími znaky předdefinované hlavičky NMEA_HEADER. Pokud daný znak souhlasí, zapíše ho do proměnné `line` a navýší pozici o jedna. Pokud nesouhlasí, nastaví pozici na nulu a začne od začátku.

Po úspěšném načtení hlavičky funkce přejde ke čtení vlastního obsahu věty ukončené kontrolní sumou. Pokud je velikost pozice `position` menší než velikost řádku `line`, funkce zapisuje obsah věty do `line`. Funkce každý znak testuje, zdali je hvězdička. Hvězdička znamená, že se nachází před kontrolní sumou (konec věty). Poté do prvního znaku proměnné `line` zapíše nulu a `readChecksum` též vynuluje. Následně program navýší pozici `position` a znaky kontrolní sumy zapíše do proměnné `checksum`. Dále se pak provede funkce výpočtu kontrolní sumy `calcChecksum` (podkapitola Funkce `calcChecksum`) z přijatých dat a funkce `decodeLine` (podkapitola Funkce `decodeLine`), která rozkóduje načtenou větu a data zapíše do proměnných.



Obrázek 17 - Vývojový diagram funkce processGPS

Kód funkce processGPS:

```
bool processGPS() {
    static int position = 0;
    int readChecksum = -1;
    char checksum[2];
    boolean result = false;
    char line[80];

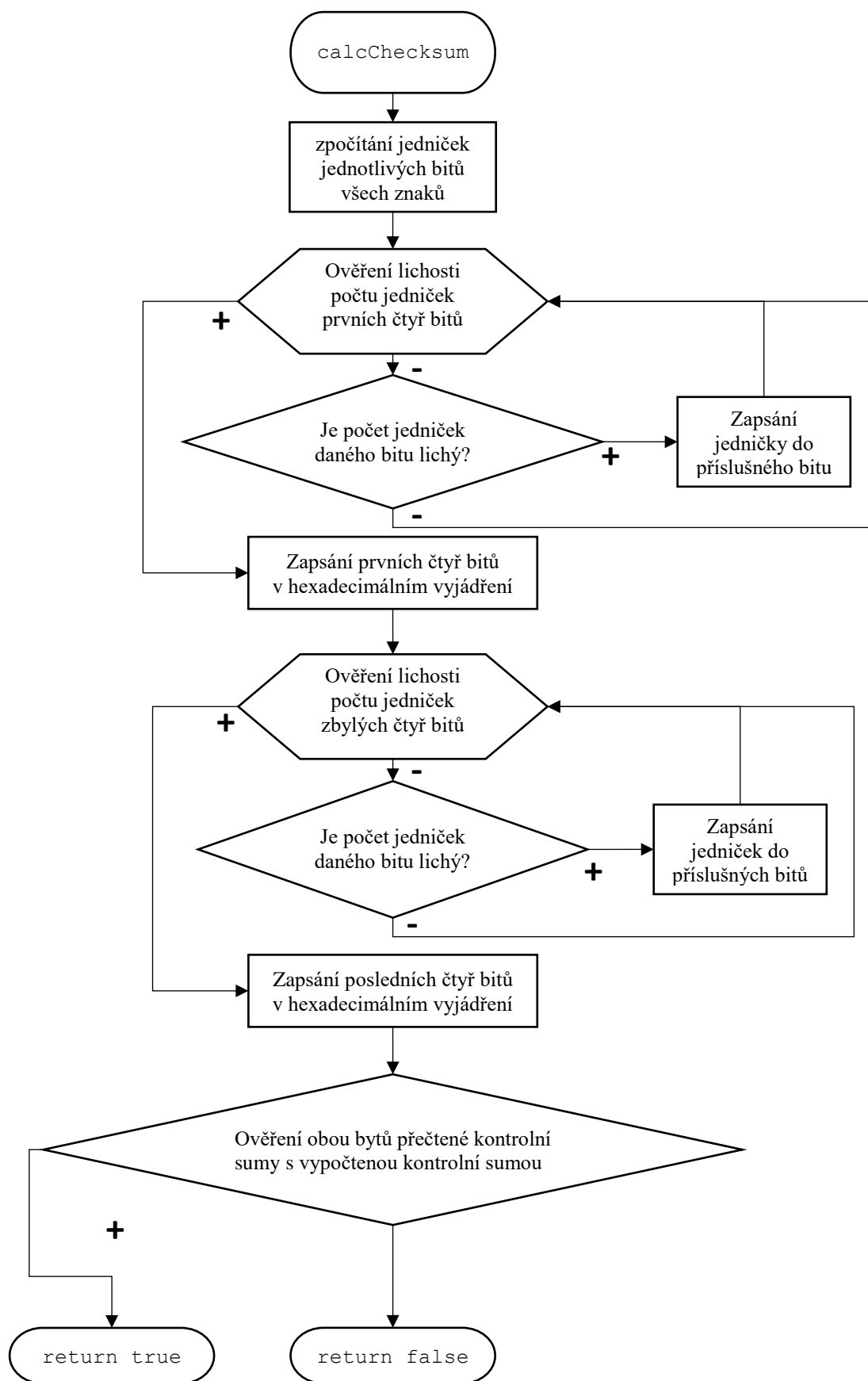
    while ( gpsSerial.available() ) {
        byte c = gpsSerial.read();
        if ( position < 7 ) {
            if ( char(c) == NMEA_HEADER[position] ) {
                if ( position>0 ) {
                    line[position-1] = c;
                }
                position++;
            }
            else
                position = 0;
        }
        else {
            if(readChecksum == -1) {
                if ( position < sizeof(line) ) {
                    line[position-1] = c;
                }
                if ( c=='*' ) {
                    line[position-1] = 0;
                    Serial.println("line: "+String(line));
                    readChecksum = 0;
                }
                position++;
            }
            else {
                checksum[readChecksum++] = c;
                if (readChecksum > 1) {
                    if (calcChecksum(line, checksum)) {
                        decodeLine(String(line));
                        result = isDecoded;
                    }
                    position = 0;
                    readChecksum = -1;
                }
            }
        }
    }
    return result;
}
```


7.2 Funkce calcChecksum

Funkce `calcChecksum` provádí kontrolní součet NMEA GPRMC věty a jeho porovnání s kontrolním součtem poslaným v dané větě. Podrobné vysvětlení výpočtu kontrolního součtu je v podkapitole Princip výpočtu kontrolní sumy.

Funkce započne cyklem `for`, který běží do doby, než zkontroluje veškeré bity všech znaků řádku `line`. Test spočívá v kontrole, zdali se v konkrétním bitu nachází jednička, jestliže ano, zvýší se hodnota příslušného pole `val` o jedna a přejde se na další bit. Tímto způsobem cyklus otestuje všech osm bitů všech znaků řádku `line`. Po provedení cyklu má funkce pole osmi hodnot reprezentující množství jedniček v jednotlivých pořadích bitů.

Zjištěné hodnoty program rozdělí do dvou skupin po čtyřech, protože kontrolní suma musí být ve tvaru dvou hexadecimálních hodnot. Otestuje lichost prvních čtyř hodnot. Principem testu je podělení a znovu vynásobení dvěma. Jelikož systém pracuje v celých číslech, vydělí-li a znovu vynásobí lichou hodnotu dvěma, dostane hodnotu odlišnou, ale naopak je-li hodnota sudá, dostane hodnotu totožnou. V případě, že hodnota je lichá (není sudá), funkce vloží do prvního příslušného bitu proměnné `result1` jedničku operátorem `or`. Tímto postupem otestuje první čtyři hodnoty a přiřadí odpovídající hodnoty do `result1`. Hodnotu `result1` převede do hexadecimální podoby a uloží do proměnné `ptr1`. Stejný proces zopakuje pro zbylé čtyři hodnoty, jejichž výsledky zapíše do `result2`, převede a uloží do `ptr2`. Na závěr parametry `ptr1` a `ptr2` (vypočtený kontrolní součet) porovná s kontrolním součtem přečteným z RMC věty `checksum[0]`, `checksum[1]`. Pokud se kontrolní součty rovnají došlo k úspěšnému přečtení RMC věty.



Obrázek 18 - Vývojový diagram funkce CalcChecksum

Kód funkce calcChecksum:

```
bool calcChecksum (char* line, char checksum[]) {
    int val[] = {0,0,0,0,0,0,0,0};
    unsigned char result1 = 0;
    unsigned char result2 = 0;
    char ptr1[2], ptr2[2];

    for (int i = 0; line[i] != 0; i++) {
        if ( line[i] & 128 ) val[0]++;
        if ( line[i] & 64 ) val[1]++;
        if ( line[i] & 32 ) val[2]++;
        if ( line[i] & 16 ) val[3]++;
        if ( line[i] & 8 ) val[4]++;
        if ( line[i] & 4 ) val[5]++;
        if ( line[i] & 2 ) val[6]++;
        if ( line[i] & 1 ) val[7]++;
    }

    if ( val[0]/2*2 != val[0] ) result1 |= 8;
    if ( val[1]/2*2 != val[1] ) result1 |= 4;
    if ( val[2]/2*2 != val[2] ) result1 |= 2;
    if ( val[3]/2*2 != val[3] ) result1 |= 1;
    sprintf(ptr1,"%X",result1);

    if ( val[4]/2*2 != val[4] ) result2 |= 8;
    if ( val[5]/2*2 != val[5] ) result2 |= 4;
    if ( val[6]/2*2 != val[6] ) result2 |= 2;
    if ( val[7]/2*2 != val[7] ) result2 |= 1;
    sprintf(ptr2,"%X",result2);

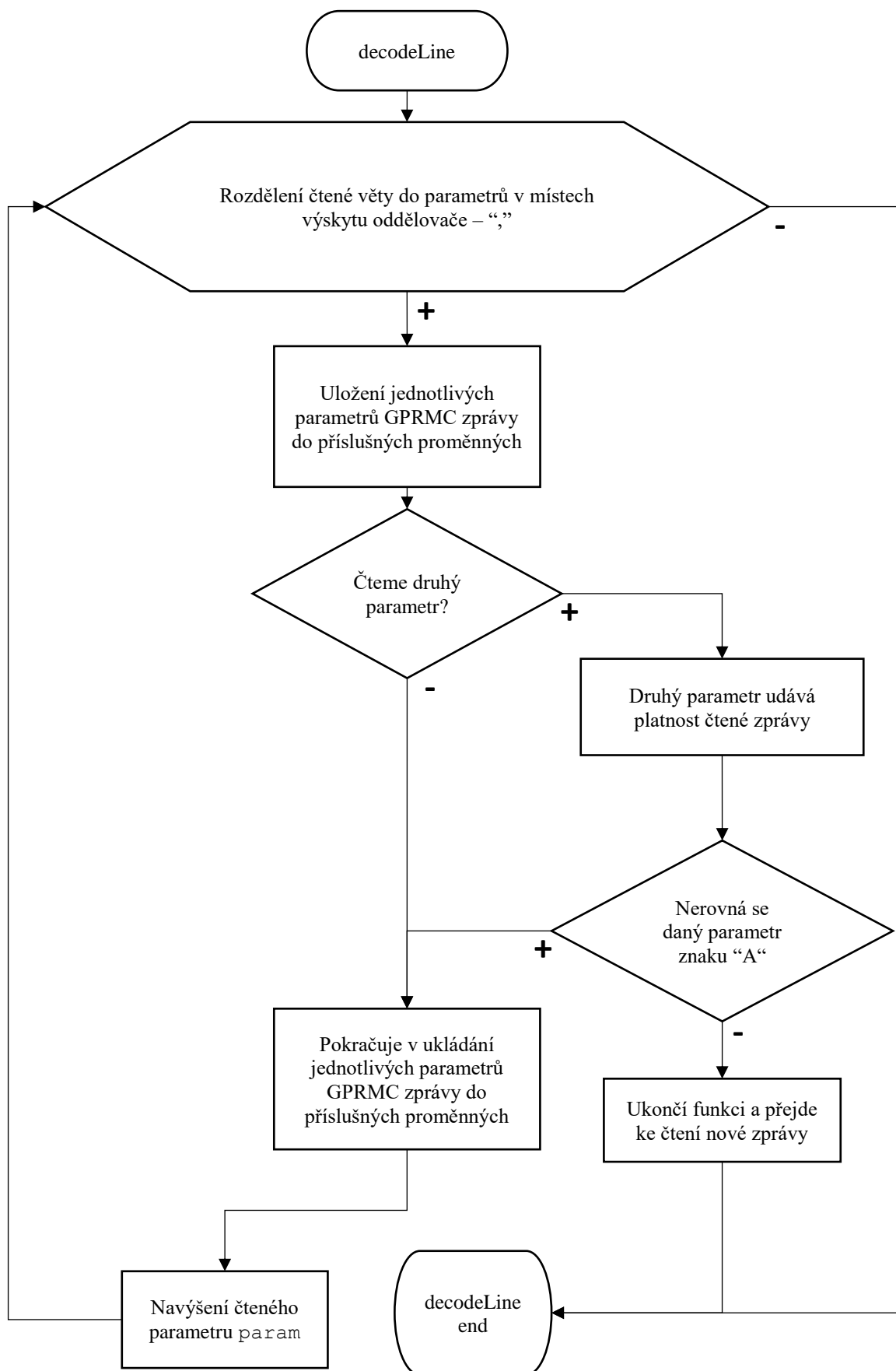
    if ( checksum[0] == ptr1[0] && checksum[1] == ptr2[0] ) {
        return true;
    }
    return false;
}
```

7.3 Funkce decodeLine

Funkce `decodeLine` zajišťuje načtení dat z přijaté GPRMC věty do příslušných proměnných. Do proměnných uloží pouze námi potřebné parametry. Celkem využije deset proměnných, a to jsou `hour`, `min`, `sec`, `lat`, `verHem`, `lon`, `horHem`, `day`, `month` a `year`.

Funkce obsahuje cyklus `for`, který slouží k rozdělení načtené NMEA věty do jednotlivých parametrů a ty pak dekoduje do příslušných proměnných. Proměnná `from` indikuje pozici v řádku, odkud se parametr čte. Do proměnné `to` se ukládá pozice následující čárky. Mezi proměnnou `from` a `to` se pomocí `substring` načte do proměnné `parameter` aktuální parametr. Na konci cyklu se do proměnné `from` uloží předešlá pozice z `to` zvětšená o jedna. `For` cyklus běží do doby, dokud proměnná `param` je menší jak deset a zároveň `from` je menší jak délka řádku `line`.

První dekodovaný parametr obsahuje čas, který funkce `sscanf` rozdělí do příslušných proměnných `hour`, `min` a `sec`. Druhý parametr udává, jestli věta obsahuje platná (A), či neplatná (V) data. Tento parametr se ověří a nemá-li platná data, větu zahodí, ukončí funkci a pokračuje v programu. Má-li platná data, pokračuje v cyklu. Třetí parametr je zeměpisná šířka ve formátu NMEA. Funkce jej rozloží, znovu poskládá a upraví tak, aby výsledek byl ve stupních. Složení a výpočet spočívá v uložení prvních dvou znaků, přeskočení tečky, vydělení zbylých znaků šesti milióny a jejich připojení k uloženým dvěma znakům. Podrobné vysvětlení převodu zeměpisné šířky a zeměpisné délky z NMEA formátu je v kapitole Převod souřadnic z NMEA vět do souboru GPX/KML. Čtvrtý parametr udává, v jaké polokouli se daná zeměpisná šířka nachází (sever nebo jih). Parametr se uloží do proměnné `verHem`. Pokud je tato proměnná S (jih), změní se znaménko proměnné `lat` na záporné. Pátý parametr vyjadřuje zeměpisnou délku (`lon`) a ošetří se stejným způsobem jako zeměpisná šířka. Šestý parametr udává, v jaké polokouli se daná zeměpisná délka nachází (západ nebo východ). Parametr se uloží do proměnné `horHem`. Pokud je tato proměnná W (západ), změní se znaménko proměnné `lon` na záporné. Poslední parametr devět obsahuje datum. Parametr se rozdělí a uloží po dvou znacích do proměnných `day`, `month` a `year`.



Obrázek 19 - Vývojový diagram funkce decodeLine

Kód funkce decodeLine:

```
void decodeLine(String line) {
    int param = 0;
    unsigned int to;
    isDecoded = true;

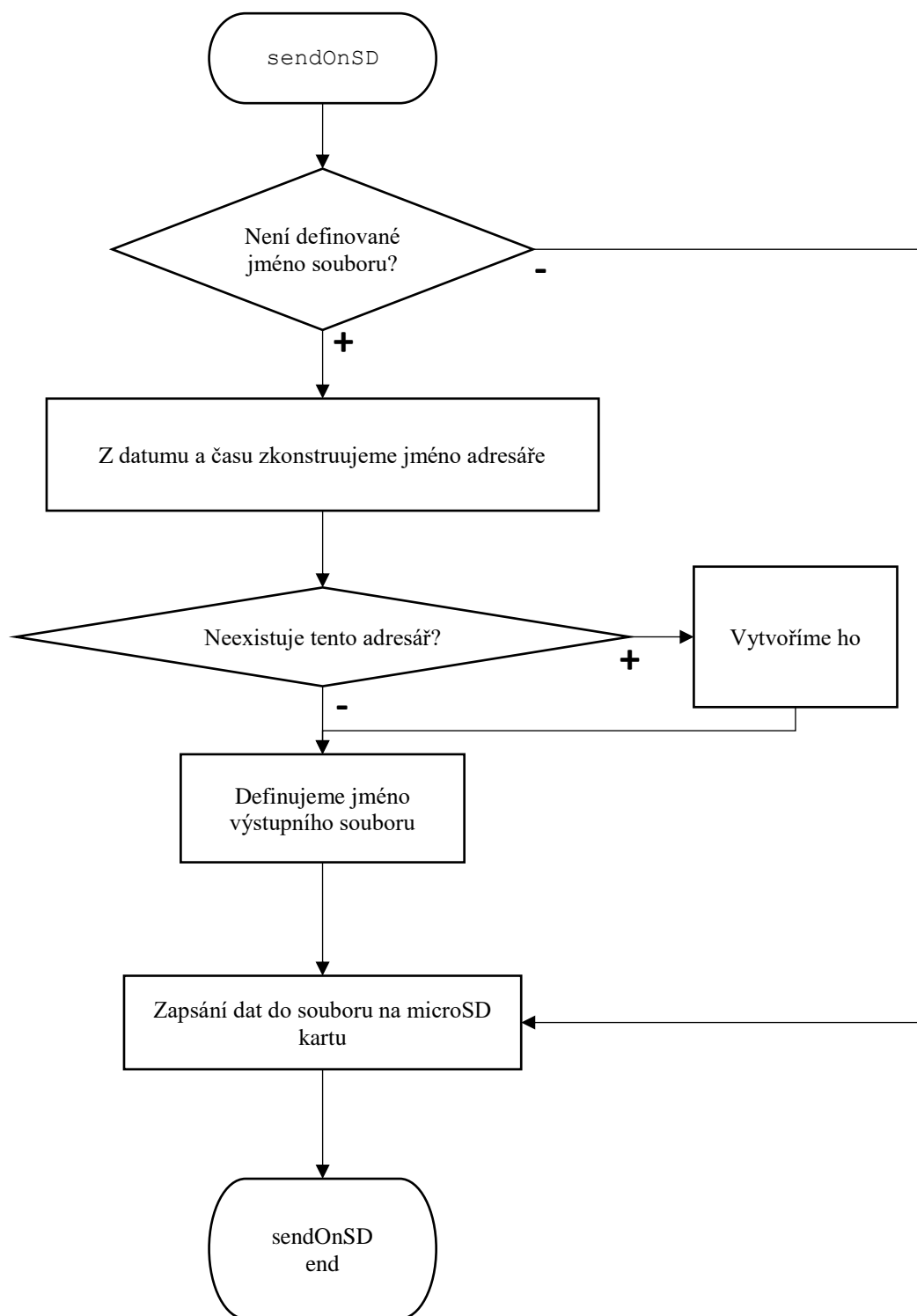
    for ( unsigned int from = 0;
          param<10 && from<line.length();
          from= to+1 ) {

        to= line.indexOf(',', from);
        String parameter= line.substring(from,to);
        char* c= parameter.c_str();

        if ( param==1 )
            sscanf(c,"%2d%2d%2d",&hour,&min,&sec);
        }
        else if ( param==2 ) {
            String validity = String(c);
            if (validity != "A") {
                isDecoded = false;
                break;
            }
        }
        else if ( param==3 ) {
            lat= String(parameter.substring(0,2)).toDouble() +
                  String(parameter.substring(2,4) +
                  parameter.substring(5,10)).toDouble()/6000000;
        }
        else if ( param==4 ) {
            verHem= String(c);
            if (verHem.equals("S")) {
                lat = -lat;
            }
        }
        }
        else if ( param==5 ) {
            lon= String(parameter.substring(0,3)).toDouble() +
                  String(parameter.substring(3,5) +
                  parameter.substring(6,11)).toDouble()/6000000;
        }
        else if ( param==6 ) {
            horHem= String(c);
            if (horHem.equals("W")) {
                lon = -lon;
            }
        }
        }
        else if ( param==9 ) {
            sscanf(c,"%2d%2d%2d",&day,&month,&year);
        }
        param++;
    }
}
```

7.4 Funkce sendOnSD

Funkce `sendOnSD` zajišťuje ukládání zpracovaných GPS dat na microSD kartu. Funkce započne kontrolou, jestli již byl vytvořen název souboru. V případě, že ještě nebyl vytvořen, program pošle informaci o čase a datumu SD kartě, aby mohla tyto údaje uvést u nově vytvořených souborů a adresářů. Naplní proměnou `path` rokem, měsícem a dnem ve formátu YYMMDD. Následně zjistí, zdali existuje adresář se stejným názvem jako proměnná `path`. Pokud neexistuje, vytvoří adresář nový s tímto názvem. Dále naplní proměnou `fileName`, kterou složí z `path`, `hour`, `min` a `sec`. Vytvoří soubor `fileName`, do kterého zapíšeme data ve formátu pro protokol GPX popsaným v kapitole Převod souřadnic z NMEA vět do souboru GPX/KML. Pokud soubor již existuje, zapíše data do něj. Po zapsání soubor uzavře a ukončí funkci.



Obrázek 20 - Vývojový diagram funkce sendOnSD

Kód funkce sendOnSD:

```
void sendOnSD() {
    delay(1000);
    if ( String(fileName).length()==0 ) {
        char path[7];
        SdFile::dateTimeCallback(dateTime);
        sprintf(path,"%02d%02d%02d",year,month,day);
        if ( !SD.exists(String(path)) ) {
            delay(1000);
            SD.mkdir(path);
        }
        sprintf(fileName,"%s/%02d%02d%02d.txt",path,hour,min,sec);

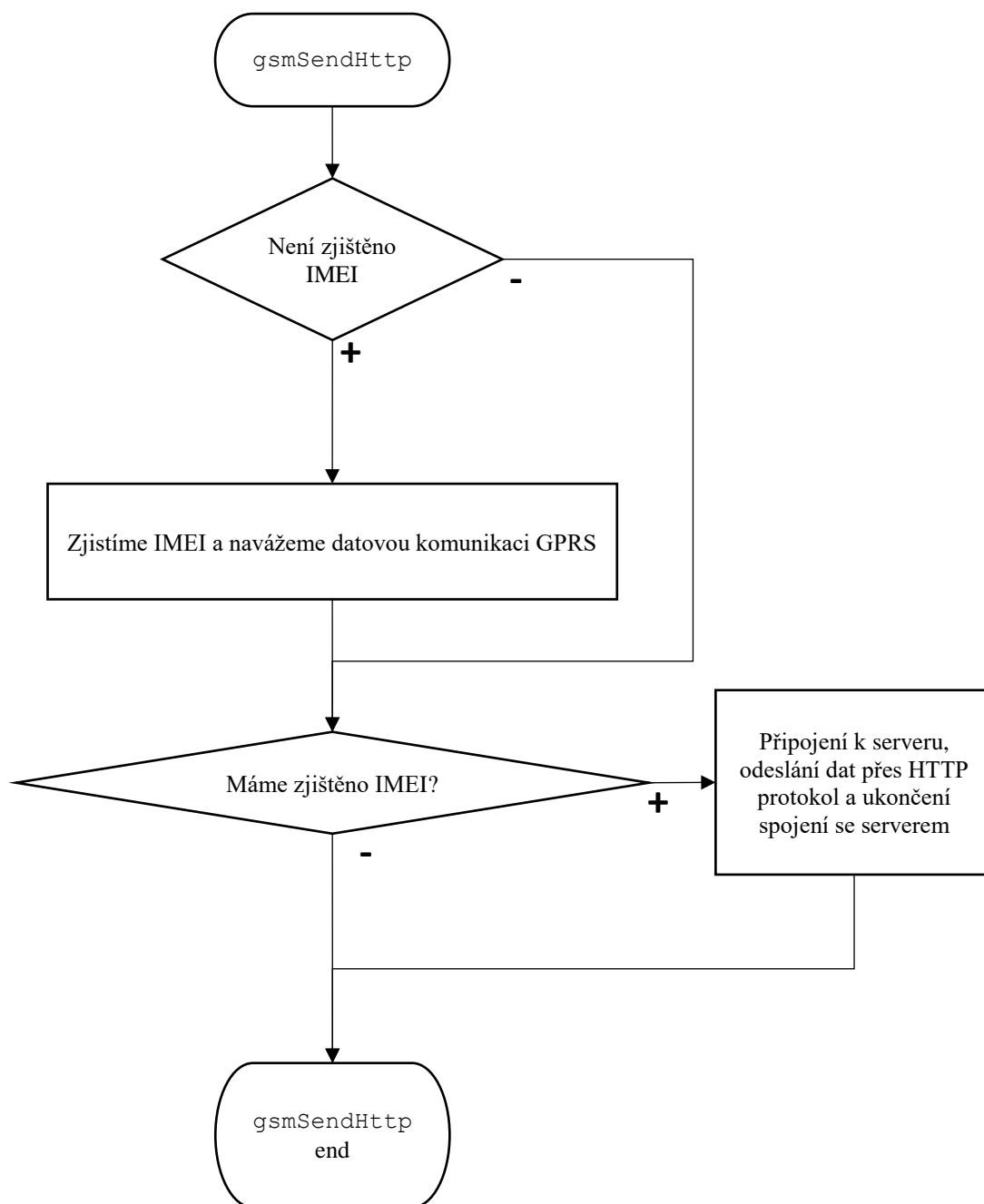
        sprintf(dt,"%4d%02d%02d%02d%02d", (year+2000),month,day,hour,min,sec);
    }

    File myFile = SD.open(fileName,FILE_WRITE);
    myFile.print(F("<trkpt lon=\""));
    myFile.print(String(lon,6));
    myFile.print(F("\\" lat=\""));
    myFile.print(String(lat,6));
    myFile.println(F("\\">"));
    myFile.print(F("<time>"));
    myFile.print(String(year+2000));
    myFile.print(F("-"));
    myFile.print(doubleDigitFix(month));
    myFile.print(F("-"));
    myFile.print(doubleDigitFix(day));
    myFile.print(F("T"));
    myFile.print(doubleDigitFix(hour));
    myFile.print(F(":"));
    myFile.print(doubleDigitFix(min));
    myFile.print(F(":"));
    myFile.print(doubleDigitFix(sec));
    myFile.println(F(".000Z</time></trkpt>"));
    myFile.close();
}
```

7.5 Funkce gsmSendHttp

Funkce `gsmSendHttp` obstarává posílání zpracovaných GPS dat na aplikační server (Apache Tomcat), kde běží servlet (funkce `WriteGpsPosition`), který je zpracovává a ukládá na serverové úložiště.

Na začátku funkce ukončí sériovou komunikaci s GPS, jelikož hardware neumožňuje sériovou komunikaci s více zařízeními současně. Prvním krokem je zjištění IMEI čísla SIM karty. Pokud systém již má IMEI, znamená to, že už je připojen do internetu a přejde rovnou k odeslání dat. V případě že IMEI nemá, vyžádá si ho příkazem `AT+GSN` a připojí se do internetu. Připojení je prováděno sekvencí AT příkazů `AT+CIPMUX=0` (jednokanálové připojení k IP), `AT+CSTT="přístupový bod operátora (APN)"`, `"uživatelské jméno operátora"`, `"heslo operátora"`, `AT+CIICR` (připojení bezdrátového připojení GPRS) a `AT+CIFSR` (vyžádání IP adresy zařízení). Následně program vytvoří řetězec `httpData` z IMEI, zeměpisné délky, zeměpisné šířky, roku, měsíce, dne, hodiny, minuty, sekundy a připojí se k serveru příkazem `AT+CIPSTART="typ připojení (TCP nebo UDP)"`, `"doména"`, `"port"`. Pomocí knihovny `CStringBuilder` vytvoří řetězec `buff` obsahující informace o cílovém http, délce posílaných dat a samotná data. Následně pošle informaci o délce celého packetu `buff`, který bude posílat příkazem `AT+CIPSEND="délka posílaného packetu"` a pošle ho. Ukončí spojení se serverem příkazem `AT+CIPCLOSE` a pokračuje v kódu.



Obrázek 21 - Vývojový diagram funkce *gsmSendHttp*

Kód funkce gsmSendHttp:

```
void gsmSendHttp() {
    gpsSerial.end();
    gsmSerial.begin(9600);
    delay();

    if ( GSMId.length()==0 ) {
        gsmSerial.println(F("AT+GSN"));
        readIMEI();

        gsmSerial.println(F("AT+CIPMUX=0"));
        delay();

        gsmSerial.println(F("AT+CSTT=\"sazkamobil\", \"\", \"\""));
        delay();

        gsmSerial.println(F("AT+CIICR"));
        delay();

        gsmSerial.println(F("AT+CIFSR"));
        delay();
    }

    if( GSMId.length()>0 ) {
        gsmSerial.println(F("AT+CIPSTART=\"TCP\", \"brazda.eu\", 80));
        delay();

        char httpData[78];
        sprintf(httpData,
            "id=%s&d=%s&data=%s;%s;%04d%02d%02d;%02d%02d%02d;",
            GSMId.c_str(), dt, String(lon,6).c_str(),
            String(lat,6).c_str(),
            year+2000, month, day, hour, min, sec);

        char buff[226];
        CStringBuilder sb(buff, sizeof(buff));
        sb.print(F("POST /GpsServlet/WriteGpsPosition HTTP/1.0\r\n"));
        sb.print(F("Host: brazda.eu\r\n"));
        sb.print(F("Accept: *"));
        sb.print(F("/"));
        sb.print(F("*\r\n"));
        sb.print(F("Content-Length: "));
        sb.printf("%d", strlen(httpData));
        sb.print(F("\r\n"));
        sb.print(F("Content-Type: application/x-www-form-urlencoded\r\n"));
        sb.print(F("\r\n"));
        sb.printf("%s", httpData);

        gsmSerial.print(F("AT+CIPSEND="));
        gsmSerial.println(sb.length());
        delay();

        gsmSerial.print(buff);
        delay();

        gsmSerial.println(F("AT+CIPCLOSE"));
        delay(4000);
    }
    gpsSerial.end();
    gpsSerial.begin(9600);
}
```

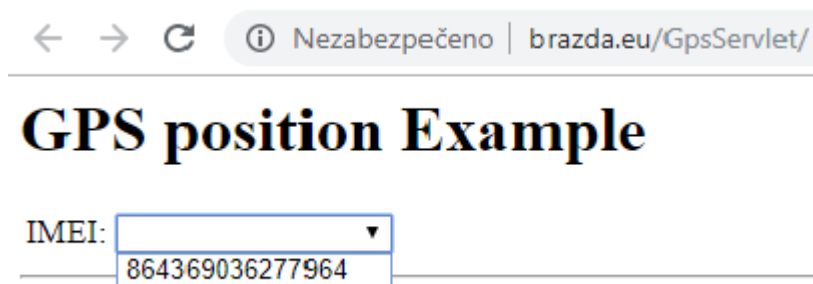
8 Zpracování dat serverem a zobrazení poslední polohy

Pro příjem a zpracování posílaných dat jsem použil aplikační server (Apache Tomcat) a program WriteGpsPosition. Program WriteGpsPosition zpracovává a ukládá přijatá GPS data na serverové úložiště do csv souboru v adresářové struktuře, která obsahuje IMEI, datum a čas. WriteGpsPosition má parametry *id*, *d* a *data*. *Id* obsahuje IMEI, *d* obsahuje datum a čas, parametr *data* obsahuje zapisovaný řádek csv souboru. Kód programu WriteGpsPosition je v příloze C [14].

Ukázka zpracovaných dat serverem:

```
16.537798;49.194225;20181227;111110;  
16.537798;49.194225;20181227;111123;  
16.537827;49.194221;20181227;111131;  
16.537817;49.194233;20181227;111139;  
16.537800;49.194244;20181227;111147;  
16.537771;49.194256;20181227;111155;  
16.537752;49.194260;20181227;111203;  
16.537802;49.194244;20181227;111211;  
16.537823;49.194237;20181227;111219;  
16.537813;49.194233;20181227;111227;
```

Pro zobrazování poslední polohy jsem použil programy: index.html, GetIMEIposition.java a ajax.js. Index.html vytváří webovou stránku <http://brazda.eu/GpsServlet>, na které lze vybrat IMEI zařízení, které chceme sledovat (obrázek 22). Po vybrání IMEI (zařízení) se otevře nové okno v Google mapách zobrazující poslední pozici. Obnova stránky se provádí každých pět sekund [14].

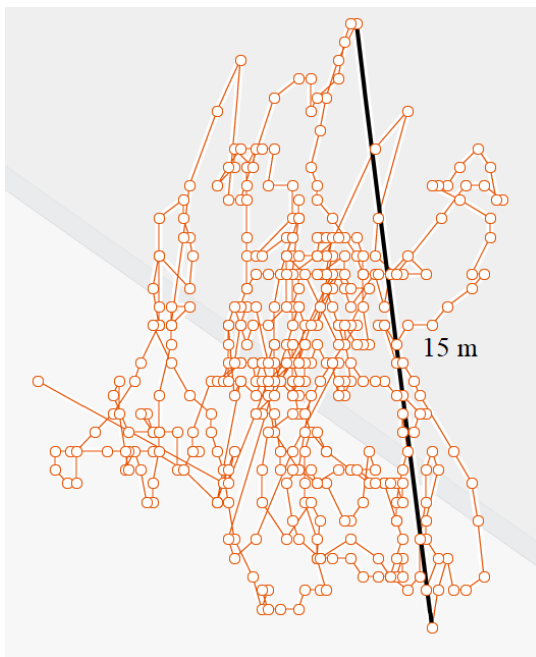


Obrázek 22 - Výběr sledovaného zařízení

Servlet GetIMEIposition vrací odkaz do Google maps s poslední polohou daného IMEI, které získá z dat uložených na serverovém úložišti. Příklad odkazu: <https://www.google.cz/maps/place/49.258468,16.669767>. Zobrazení poslední polohy názorného odkazu je v příloze D. Kódy zmíněných programů jsou součástí příloh E, F a G.

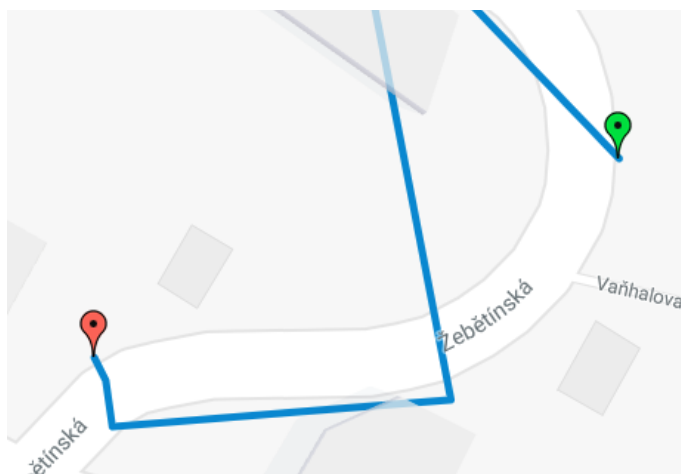
9 Odzkoušení systému a jeho zhodnocení

Pro odzkoušení systému jsem provedl řadu měření. První měření spočívalo v ponechání zapnutého zařízení v garáži s betonovým stropem bez pohybu po jednu hodinu. Změřená data jsou na obrázku 23. Maximální rozptyl změřených bodů byl 15 m.



Obrázek 23 - změřená data, garáž

Dále bylo zařízení podrobena zkušební jízdě osobním automobilem. Zaznamenanou trasu můžete vidět v příloze H zobrazenou v Google mapách. Z detailu počátku cesty (obrázek 24) můžeme vidět prodlevu nalezení polohy (zelená značka) vůči počátku cesty (červená značka).



Obrázek 24 - Detail cesty

Dalším poznatkem, který lze vyčíst z mapy v příloze H je, že cesta byla rozdělena do dvou samostatných tras. Důvodem je, že v místě rozdělení byla provedena přestávka a při opětovném nastartování motoru se odpojily všechny spotřebiče čili se restartovalo i naše zařízení. Při levém kliknutí myši na trasu v Google mapách se zobrazí všechny její změřené body a její parametry obsahující datum a čas počátku cesty, její vzdálenost, délku v čase a průměrnou rychlost.

Zhodnocení systému

Úspěšně se mi podařilo splnit požadavky na bezdrátové sledování polohy v reálném čase, ukládání polohy na paměťové médium, bezdrátové ukládání dat na server, při selhání bezdrátové komunikace pokračovat v ukládání dat na paměťové médium, možnost napájení z akumulátoru dopravního prostředku nebo i připojením externího akumulátoru, zobrazení poslední polohy serverem, vypnutí zařízení při vyjmutí klíče ze zapalování nebo odpojením napájení. Zobrazením změřené trasy v reálném čase, šifrováním posílaných dat a autorizací ke sledování vymezených zařízení na základě identifikace přihlašovacím jménem a autentizace heslem jsem se nezabýval z důvodu nedostatku času.

Posílání dat přes GSM se SIM kartou SAZKA mobil probíhalo v pořádku. Se SIM kartou T-MOBILE jsem narazil na problém, kdy se vždy komunikace přerušila po přibližně jedné minutě. Důvodem bylo automatické přerušování komunikace tímto operátorem. Problém jsem neřešil z důvodu nedostatku času. Dal by se však vyřešit naprogramováním automatického obnovení komunikace.

Sestrojené zařízení má velikost 77 x 55 x 30 mm, dalo by se však ještě zmenšit. Hlavní slabou stránku systému vidím v jeho neschopnosti bez restartu obnovit bezdrátovou komunikaci se serverem při jejím přerušení (ztráta signálu, ukončení komunikace operátorem apod.). Řešení napájení přes USB kabel je protichůdné s požadavkem na skrytost zařízení, bylo však potřebné pro usnadnění ladění systému a snadnou manipulaci, nicméně by bylo vhodnější zařízení napájet přímo z autobaterie a stanovit místo pro jeho skrytí.

Závěr

V bakalářské práci se mi úspěšně povedlo navrhnout, sestavit, naprogramovat a zprovoznit systém pro záznam pohybu vozidel. Systém umožňuje zpětné zobrazení změřených cest na mapových portálech podporujících GPX formát. Dále systém umožňuje zobrazení poslední zaznamenané polohy a sledování pozice v reálném čase.

Je zde však mnoho prostoru pro zdokonalení systému jak v oblasti hardwaru, tak softwaru. Zdokonalení v oblasti hardwaru by mohlo spočívat v užití součástek v SMD provedení, vyrobení desky plošných spojů na míru a lepším řešením napájení, což by zmenšilo celkovou velikost. Nákup dalšího hardwaru by však přesáhl můj rozpočet stanovený pro tuto práci.

Další práci na softwaru vidím v úpravě způsobu ukládání GPS dat na paměťové médium i server tak, aby se ukládala automaticky v GPX formátu a tím umožnila okamžité zobrazení změřených cest na mapových portálech, případně jejich sledování v reálném čase. Hlavním zlepšením systému by bylo naprogramování obnovení komunikace se serverem při jejím přerušení, což by zajistilo spolehlivost systému a jeho kompatibilitu s různými SIM kartami.

Dále by bylo vhodné provést rozsáhlé testování různých GNSS přijímačů, zejména jejich rychlosti nalezení polohy a schopnosti sledovat polohu za zhoršených viditelnostních podmínek, jelikož v mnoha aplikacích zařízení není zaručena. Zrychlení nalezení polohy by mohlo být umožněno stažením almanachu družic z internetu. Také by bylo vhodné otestovat SIM karty od různých poskytovatelů a provést rozsáhlejší testování systému v běžném provozu pro určení jeho spolehlivosti a nedostatků.

V práci jsem naznačil způsob odesílání dat na server a jejich zobrazení. Dalším krokem by bylo zvýšení bezpečnosti zajištěním šifrování odesílaných dat a aplikování přístupového rozhraní do systému, aby do něj měly přístup pouze autorizované osoby, a to pouze ke sledování svých zařízení.

Komponenty byly pořízeny ze zahraničního e-shopu Aliexpress pro jejich výhodnou cenu, lze je však pořídit i u nás v ČR nebo na e-shopech jako je Ebay, Amazon apod. Finální cena zařízení byla 445 Kč. Do ceny jsem nezahrnul komponenty, které jsem již měl k dispozici a dodatečné GPS, GSM moduly, které jsem využil v průběhu sestavování a testování. Po zahrnutí všech využitých součástek v průběhu bakalářské práce se celková cena pohybuje kolem 1000 Kč.

Shrnutí ceny:

• GPS modul	103 Kč,
• Micro SD card reader/writer	18 Kč,
• GSM modul	57 Kč,
• regulátor napětí	51 Kč,
• Arduino Nano	46 Kč,
• SIM karta s daty na 3 měsíce	75 Kč,
• microSD karta 16 GB	95 Kč.

Seznam literatury

- [1] Heureka. *Heureka* [online]. Česká republika: Heureka Shopping, 2019 [cit. 2019-05-24]. Dostupné z: <https://www.heureka.cz/?h%5Bfraz%5D=gps+tracker>.
- [2] Arduino. *Arduino* [online]. Česká republika, 2019 [cit. 2019-05-24]. Dostupné z: <https://arduino.cz/co-je-to-arduino/>.
- [3] *Globální družicový polohový systém* [online]. [cit. 2019-03-15]. Dostupné z: https://www.wikiwand.com/cs/Glob%C3%A1ln%C3%AD_dru%C5%BEicov%C3%BD_polohov%C3%BD_syst%C3%A9m.
- [4] Jean-Marie Zogg. *GPS: Essentials of Satellite Navigation*. u-blox, 2009. ISBN 978-3-033-02139-6.
- [5] RAPANT, Petr. *Družicové polohové systémy*. Ostrava: VŠB - Technická univerzita Ostravy, 2002. ISBN 80-248-0124-8.
- [6] Global Positioning System. *Global Positioning System* [online]. [cit. 2019-03-16]. Dostupné z: https://www.wikiwand.com/cs/Global_Positioning_System.
- [7] U-BLOX. *U-blox 7 Receiver Description: Including Protocol Specification V14*. V14. u-blox. GPS.G7-SW-12001-B1.
- [8] *GPX: the GPS Exchange Format* [online]. 24 Kirkland Dr, Stow, MA: TopoGrafix, c1998 [cit. 2019-02-26]. Dostupné z: <https://www.topografix.com/gpx.asp>.
- [9] Arduino. *Arduino* [online]. San Francisco: Wikipedia [cit. 2019-02-06]. Dostupné z: <https://cs.wikipedia.org/wiki/Arduino>.
- [10] *ARDUINO NANO* [online]. ARDUINO [cit. 2019-02-06]. Dostupné z: <https://store.arduino.cc/arduino-nano>.
- [11] *NEO-7: NEO-7 u-blox 7 GNSS modules Data Sheet*. R07. u-blox. UBX-13003830.
- [12] *SIM800L Hardware Design*. V1.00. SIMCom, 2013. ISBN SIM800L_Hardware_Design_V1.00.
- [13] *Arduino SD Card Module Interface – Hook-up Guide and Data Logging* [online]. ElectronicsHub, 2018 [cit. 2019-02-26]. Dostupné z: <https://www.electronicshub.org/arduino-sd-card-module/>.
- [14] BRÁZDA, Radek. *WriteGpsPosition, Index, GetIMEIposition a ajax*.

Seznam příloh

Příloha A – Technické výkresy schránky

Příloha B – GPS_Tracker

Příloha C – WriteGpsPostition.java

Příloha D – Zobrazení poslední polohy serverem

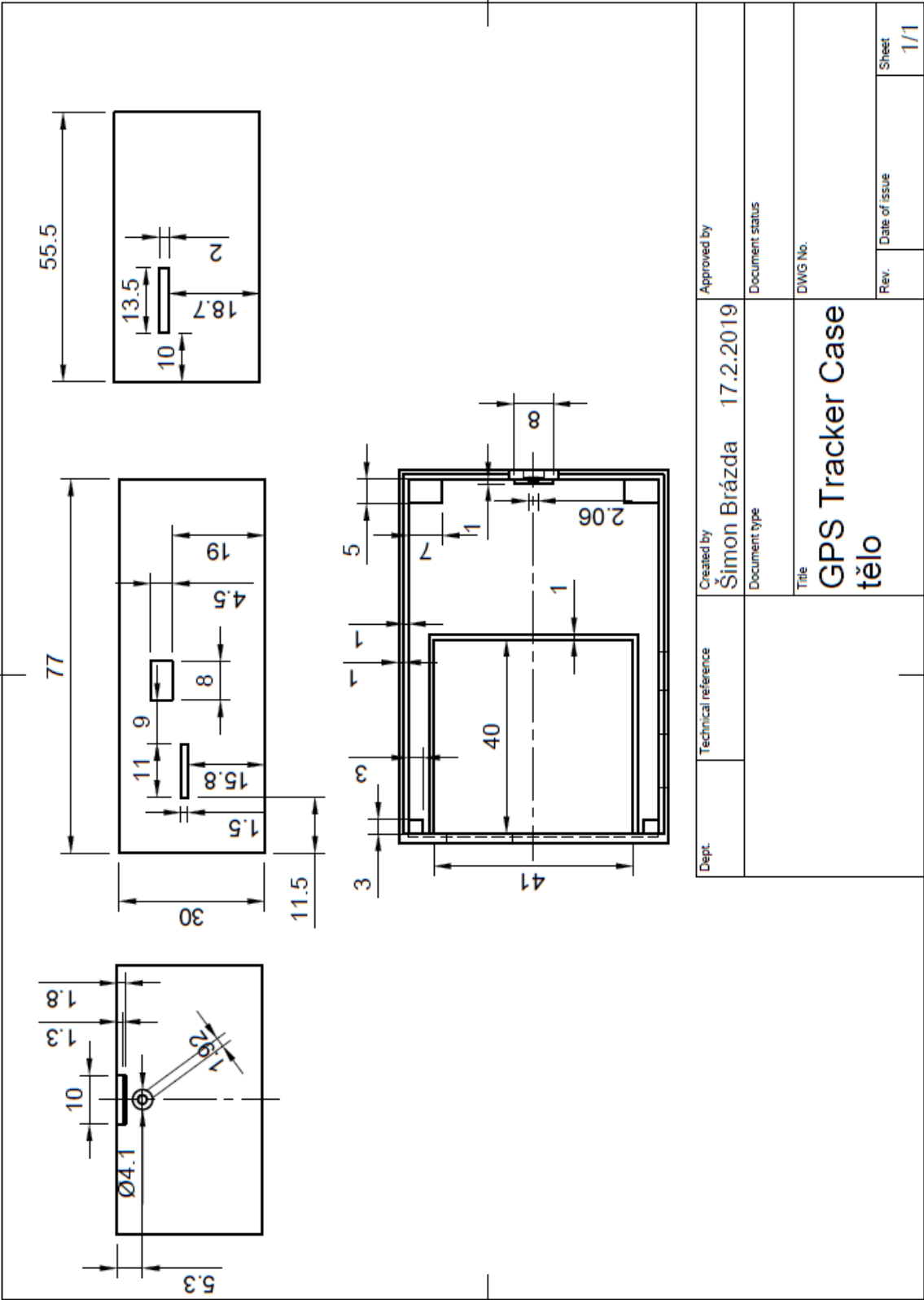
Příloha E – index.html

Příloha F – GetIMEIposition.java

Příloha G – ajax.js

Příloha H – Změřená trasa zkušební jízdy

Příloha A – Technické výkresy schránky



Příloha B – GPS_Tracker

```
#include <SoftwareSerial.h>
#include <SD.h>
#include <SPI.h>
#include <CStringBuilder.h>

SoftwareSerial gpsSerial = SoftwareSerial(2,3);
SoftwareSerial gsmSerial = SoftwareSerial(4,5);

String GSMId = "";

static char fileName[18];
char dt[15];
#define chipSelect 8

const unsigned char NMEA_HEADER[] = { '$' , 'G', 'P', 'R', 'M', 'C', ',', ' ' };

unsigned int hour;
unsigned int min;
unsigned int sec;
double lat;
double lon;
String verHem;
String horHem;
unsigned int day;
unsigned int month;
unsigned int year;
bool isDecoded;

void decodeLine(String line) {
    int param = 0;
    unsigned int to;
    isDecoded = true;

    for ( unsigned int from = 0;
          param<10 && from<line.length();
          from= to+1 ) {

        to= line.indexOf(',', from);
        String parameter= line.substring(from,to);
        char* c= parameter.c_str();

        if ( param==1 )
            sscanf(c,"%2d%2d%2d.",&hour,&min,&sec);
        }
        else if ( param==2 ) {
            String validity = String(c);
            if (validity != "A") {
                isDecoded = false;
                break;
            }
        }
        else if ( param==3 ) {
            lat= String(parameter.substring(0,2)).toDouble() +
                  String(parameter.substring(2,4) +
                  parameter.substring(5,10)).toDouble()/6000000;
        }
        else if ( param==4 ) {
            verHem= String(c);
            if (verHem.equals("S")) {
                lat = -lat;
            }
        }
    }
}
```

```

        else if ( param==5 ) {
            lon= String(parameter.substring(0,3)).toDouble() +
                String(parameter.substring(3,5) +
                    parameter.substring(6,11)).toDouble()/6000000;
        }
        else if ( param==6 ) {
            horHem= String(c);
            if (horHem.equals("W")) {
                lon = -lon;
            }
        }
        else if ( param==9 ) {
            sscanf(c,"%2d%2d%2d",&day,&month,&year);
        }
        param++;
    }
}

bool calcChecksum (char* line, char checksum[]) {

    int val[] = {0,0,0,0,0,0,0,0};
    unsigned char result1 = 0;
    unsigned char result2 = 0;
    char ptr1[2], ptr2[2];

    for (int i = 0; line[i] != 0; i++) {
        if ( line[i] & 128 ) val[0]++;
        if ( line[i] & 64 ) val[1]++;
        if ( line[i] & 32 ) val[2]++;
        if ( line[i] & 16 ) val[3]++;
        if ( line[i] & 8 ) val[4]++;
        if ( line[i] & 4 ) val[5]++;
        if ( line[i] & 2 ) val[6]++;
        if ( line[i] & 1 ) val[7]++;
    }

    if ( val[0]/2*2 != val[0] ) result1 |= 8;
    if ( val[1]/2*2 != val[1] ) result1 |= 4;
    if ( val[2]/2*2 != val[2] ) result1 |= 2;
    if ( val[3]/2*2 != val[3] ) result1 |= 1;
    sprintf(ptr1,"%X",result1);

    if ( val[4]/2*2 != val[4] ) result2 |= 8;
    if ( val[5]/2*2 != val[5] ) result2 |= 4;
    if ( val[6]/2*2 != val[6] ) result2 |= 2;
    if ( val[7]/2*2 != val[7] ) result2 |= 1;
    sprintf(ptr2,"%X",result2);

    if ( checksum[0] == ptr1[0] && checksum[1] == ptr2[0] ) {
        return true;
    }
    return false;
}

bool processGPS() {

    static int position = 0;
    int readCheckSum = -1;
    char checksum[2];
    boolean result = false;
    char line[80];

    while ( gpsSerial.available() ) {
        byte c = gpsSerial.read();
        if ( position < 7 ) {
            if ( char(c) == NMEA_HEADER[position] ) {

```

```

        if ( position>0 ) {
            line[position-1] = c;
        }
        position++;
    }
    else
        position = 0;
}
else {
    if(readChecksum == -1) {
        if ( position < sizeof(line) ) {
            line[position-1] = c;
        }
        if ( c=='*' ) {
            line[position-1] = 0;
            Serial.println("line: "+String(line));
            readChecksum = 0;
        }
        position++;
    }
    else {
        checksum[readChecksum++] = c;
        if (readChecksum > 1) {
            if (calcChecksum(line, checksum)) {
                decodeLine(String(line));
                result = isDecoded;
            }
            position = 0;
            readChecksum = -1;
        }
    }
}
}
return result;
}

void dateTime(uint16_t* date, uint16_t* time) {
    *date = FAT_DATE(year, month, day);
    *time = FAT_TIME(hour, min, sec);
}

void delay() {
    delay(500);
}

void readIMEI() {
    delay(1000);
    while (gsmSerial.available()) {
        char c = gsmSerial.read();
        if ( isdigit(c)!=0 ) {
            GSMId += String(c);
        }
    }
}

String doubleDigitFix(int i) {
    if(i < 10) {
        return String("0" + String(i));
    }
    return String(i);
}

void gsmSendHttp() {
    gpsSerial.end();
    gsmSerial.begin(9600);
}

```



```

delay();

if ( GSMId.length()==0 ) {
    gsmSerial.println(F("AT+GSN"));
    readIMEI();

    gsmSerial.println(F("AT+CIPMUX=0"));
    delay();

    gsmSerial.println(F("AT+CSTT=\"sazkamobil\", \"\", \"\""));
    delay();

    gsmSerial.println(F("AT+CIICR"));
    delay();

    gsmSerial.println(F("AT+CIFSR"));
    delay();
}

if( GSMId.length()>0 ) {
    gsmSerial.println(F("AT+CIPSTART=\"TCP\", \"brazda.eu\", 80));
    delay();

    char httpData[78];
    sprintf(httpData,
        "id=%s&d=%s&data=%s;%s;%04d%02d%02d;%02d%02d%02d;",
        GSMId.c_str(), dt, String(lon,6).c_str(),
        String(lat,6).c_str(),
        year+2000, month, day, hour, min, sec);

    char buff[226];
    CStringBuilder sb(buff, sizeof(buff));
    sb.print(F("POST /GpsServlet/WriteGpsPosition HTTP/1.0\r\n"));
    sb.print(F("Host: brazda.eu\r\n"));
    sb.print(F("Accept: *"));
    sb.print(F("/"));
    sb.print(F("*\r\n"));
    sb.print(F("Content-Length: "));
    sb.printf("%d", strlen(httpData));
    sb.print(F("\r\n"));
    sb.print(F("Content-Type: application/x-www-form-urlencoded\r\n"));
    sb.print(F("\r\n"));
    sb.printf("%s", httpData);

    gsmSerial.print(F("AT+CIPSEND="));
    gsmSerial.println(sb.length());
    delay();

    gsmSerial.print(buff);
    delay();

    gsmSerial.println(F("AT+CIPCLOSE"));
    delay(4000);
}
gsmSerial.end();
gpsSerial.begin(9600);
}

void setup() {
    fileName[0]= 0;
    Serial.begin(9600);
    pinMode(10, OUTPUT);
    SD.begin(chipSelect);
    gpsSerial.begin(9600);
    delay(1000);
}

```

```

void sendOnSD() {
    delay(1000);
    if ( String(fileName).length()==0 ) {
        char path[7];
        SdFile::dateTimeCallback(dateTime);
        sprintf(path,"%02d%02d%02d",year,month,day);
        if ( !SD.exists(String(path)) ) {
            delay(1000);
            SD.mkdir(path);
        }
        sprintf(fileName,"%s/%02d%02d%02d.txt",path,hour,min,sec);

        sprintf(dt,"%4d%02d%02d%02d%02d", (year+2000),month,day,hour,min,sec);
    }

    File myFile = SD.open(fileName,FILE_WRITE);
    myFile.print(F("<trkpt lon=\""));
    myFile.print(String(lon,6));
    myFile.print(F("\" lat=\""));
    myFile.print(String(lat,6));
    myFile.println(F(">"));
    myFile.print(F("<time>"));
    myFile.print(String(year+2000));
    myFile.print(F("-"));
    myFile.print(doubleDigitFix(month));
    myFile.print(F("-"));
    myFile.print(doubleDigitFix(day));
    myFile.print(F("T"));
    myFile.print(doubleDigitFix(hour));
    myFile.print(F(":"));
    myFile.print(doubleDigitFix(min));
    myFile.print(F(":"));
    myFile.print(doubleDigitFix(sec));
    myFile.println(F(".000Z</time></trkpt>"));
    myFile.close();
}

void loop() {

    processGPS();

    if (processGPS()) {
        sendOnSD();
        gsmSendHttp();
    }
}

```

Příloha C – WriteGpsPostition.java

```
package eu.brazda.gps;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.StringTokenizer;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/WriteGpsPosition")
public class WriteGpsPosition extends HttpServlet {

    private static final long serialVersionUID = 1L;
    public static String OUTPUT_FOLDER = "GPS_DATA";

    public WriteGpsPosition() {
        super();
    }

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        int result = 2;
        if ( request.getParameter("id")!=null &&
            request.getParameter("d")!=null &&
            request.getParameter("data")!=null ) {
            String id= request.getParameter("id");
            String data= request.getParameter("data");
            StringTokenizer s= new StringTokenizer(data, ";");
            String par1= s.hasMoreTokens() ? s.nextToken() : "";
            String par2= s.hasMoreTokens() ? s.nextToken() : "";
            String date= request.getParameter("d");
            String path = OUTPUT_FOLDER + "/" + id + "/" + date;
            String filename = id + "_" + date + ".csv";
            result = saveFile(path, filename, data, true);
            if ( result==0 ) {
                path = OUTPUT_FOLDER;
                filename = id + ".txt";
                result = saveFile(path, filename, par2 + "," + par1,
                    false);
            }
        }
        response.getWriter().append(String.valueOf(result));
    }

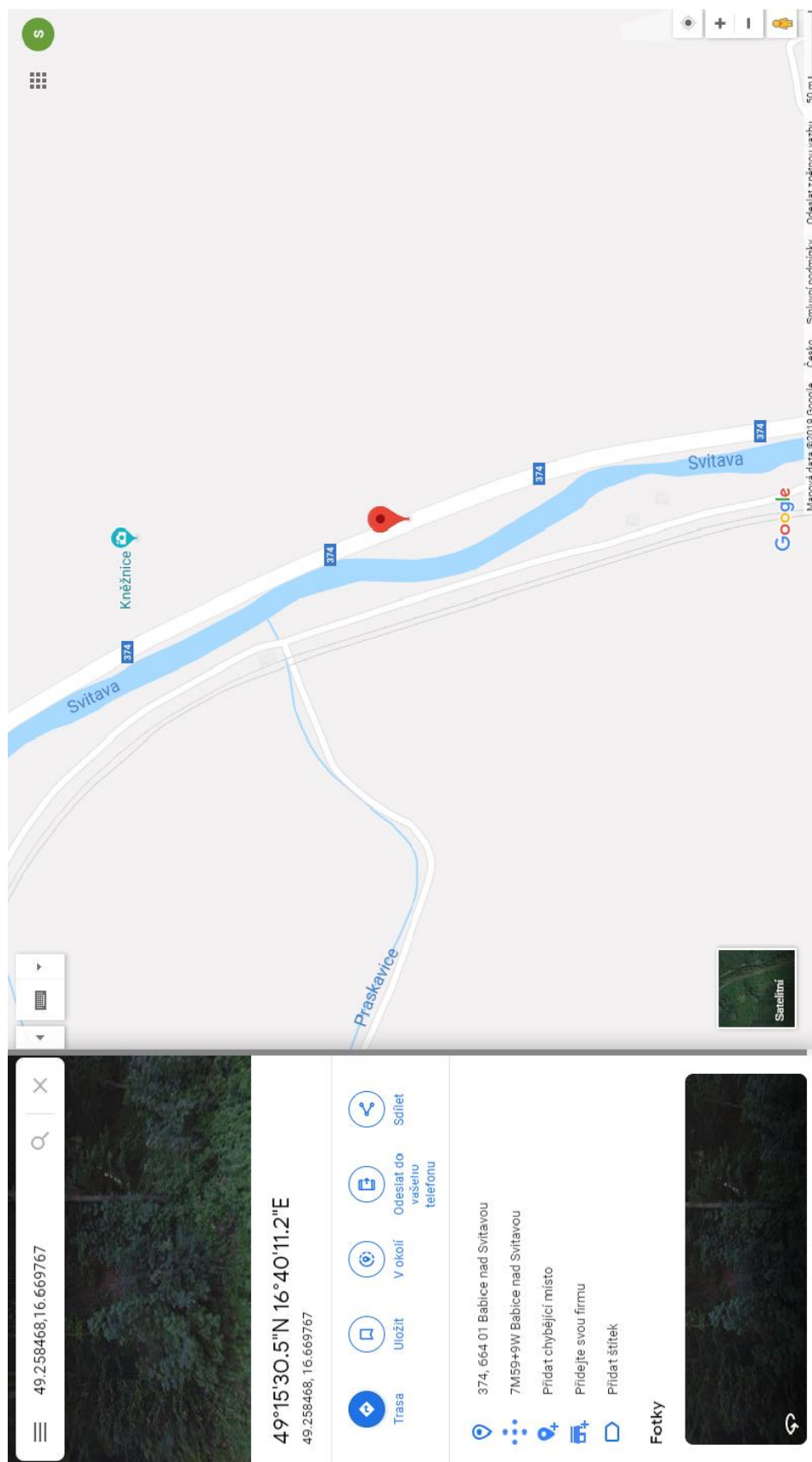
    private int saveFile(String path, String fileName, String content,
        boolean append) {
        int retCode = -1;
        File uploadDir = new File(path);
        File outputFile = new File(uploadDir.getAbsolutePath() + "/" +
            fileName);
        try {
```

```

        if (!uploadDir.exists()) {
            uploadDir.mkdirs();
        }
        FileWriter fw = new FileWriter(outputFile, append);
        fw.write(content+(append ? "\n" : ""));
        fw.close();
        retCode = 0;
    } catch (IOException ioe) {
        retCode = 1;
    }
    return retCode;
}
}

```

Příloha D – Zobrazení poslední polohy serverem



Příloha E – index.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
      charset=ISO-8859-1"></meta>
    <title>GPS position Examples</title>
    <script type="text/javascript" src="js/ajax.js"></script>
  </head>

  <body>
    <h1>GPS position Example</h1>
    <table border="0">
      <tr>
        <td>
          IMEI:
        </td>
        <td>
          <select id="imeis"
            onchange="showLocation(this.value)">
          </select>
        </td>
      </tr></table>
    <hr/>
    <div id="imeiLocation"></div>

    <script>

      incHtmlElementSet(document.location+'GetIMEIposition','imeis');

      function showLocation(id) {
        window.setInterval("window.refreshMap("+id+")",5000);
      }

      function refreshMap(id) {
        incHtmlElementSet(document.location+
          'GetIMEIposition?id='+id,'imeiLocation');
        window.open(document.getElementById
          ('imeiLocation').innerHTML, "positionOnMap",
          "width=1024, height=768");
      }
    </script>

  </body>
</html>
```

Příloha F – GetIMEIposition.java

```
package eu.brazda.gps;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/GetIMEIposition")
public class GetIMEIposition extends HttpServlet {

    private static final long serialVersionUID = 1L;
    public static String OUTPUT_FOLDER = "GPS_DATA";

    public GetIMEIposition() {
        super();
    }

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        String ret= "";
        String id = request.getParameter("id");
        if ( id!=null ) {
            ret = "https://www.google.cz/maps/place/"+
                readDataFromFile(OUTPUT_FOLDER,id+".txt");
        }
        else {
            StringBuilder sb= new StringBuilder();
            File uploadDir = new File(OUTPUT_FOLDER);
            File source= new File(uploadDir.getAbsolutePath());
            String files_list[]= source.list();
            for ( int i=0; files_list!=null &&
                i<files_list.length; i++ ) {
                if ( files_list[i].length()==19 &&
                    files_list[i].endsWith(".txt") ) {
                    sb.append("<option id='"+
                        files_list[i].substring(0,15)+
                        "'value='"+files_list[i].substring(0,15)+
                        "'>"+files_list[i].substring(0,15)+
                        "</option>");
                }
            }
            ret= sb.toString();
        }
        response.getWriter().append(ret);
    }

    private String readDataFromFile(String path, String fileName) {
        String ret= "";
    }
}
```

```

File uploadDir = new File(OUTPUT_FOLDER);
File inputFile = new File(uploadDir.getAbsolutePath()+
    "/" + fileName);
BufferedReader br = null;
try {
    br = new BufferedReader(new FileReader(inputFile));
} catch (FileNotFoundException ex1) {
}
try {
    while (br!=null && br.ready()) {
        ret = br.readLine();
        break;
    }
} catch (IOException ex2) {
}
finally {
    try {
        if ( br!=null ) {
            br.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
return ret;
}

```


Příloha G – ajax.js

```
var xmlhttp= null;
var elementIDName = null;

function initAjax() {
    if ( xmlhttp==null ) {
        if ( window.ActiveXObject ) {
            xmlhttp= new ActiveXObject("Microsoft.XMLHTTP");
        }
        else if ( window.XMLHttpRequest ) {
            xmlhttp= new XMLHttpRequest();
        }
    }
}

function incHtmlElementValueSet() {
    if ( elementIDName!=null ) {
        if ( xmlhttp.readyState==4 ) {
            if ( xmlhttp.status==200 ) {
                var el= document.getElementById(elementIDName);
                if ( !el )
                    el= elementIDName;
                if ( el ) {
                    el.value= xmlhttp.responseText;
                }
            }
        }
    }
}

function incHtmlElementSetContent() {
    if ( elementIDName!=null ) {
        if ( xmlhttp.readyState==4 ) {
            if ( xmlhttp.status==200 ) {
                var el= document.getElementById(elementIDName);
                if ( !el )
                    el= elementIDName;
                if ( el ) {
                    el.innerHTML= xmlhttp.responseText;
                }
            }
        }
    }
}

function incHtmlElementSet(url,id) {
    var el= document.getElementById(id);
    if ( !el )
        el= id;
    if ( el ) {
        initAjax();
        if ( xmlhttp!=null ) {
            elementIDName= id;
            xmlhttp.open("GET",url,false);
            xmlhttp.onreadystatechange= incHtmlElementSetContent;
            xmlhttp.send(null);
            if ( !window.ActiveXObject ) {
                incHtmlElementValueSet();
            }
            elementIDName= null;
        }
    }
}
```

Příloha H – Změřená trasa zkušební jízdy

