UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Katedra informačních technologií

Návrh a realizace systému lokalizace automobilu s využitím systému GPS a GSM

Bakalářská práce

Autor: Jan Dian

Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Josef Horálek, Ph.D.

Hradec Králové 2015

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne Jan Dian

Poděkování:

Děkuji vedoucímu bakalářské práce Mgr. Josefu Horálkovi, Ph.D. za veškerou podporu a pomoc při zpracování práce.

Anotace

Tato bakalářská práce si klade za cíl navrhnout a vytvořit modul využívající GPS pro určení aktuální pozice vozidla s jejím následným zobrazením v programu Google Earth.

Čtenář je postupně seznámen se základními principy lokalizace pomocí GPS, s fungováním GSM sítě a s datovými přenosy v této síti.

Práce také popisuje samotný návrh řešení, včetně zapojení elektronických obvodů a programového vybavení, realizovaného v programovacích jazycích Java a Wiring.

Abstract

This bachelor thesis is setting a target to design and build a GPS tracking module to automatically track and record the position of a vehicle. The module can then be connected to PC and can display route data in Google Earth.

The reader is gradually introduced to basic principles of GPS localization, functionality of GPS network and data transmission in this network.

The thesis further describes the proposed solution consisting of the electronic circuit design and the software package. The software package is implemented in the Java and Wiring programming languages.

Obsah

Úvod

Cílem práce je popsat základní principy systémů GPS a GSM. Dalším cílem je navrhnout a vytvořit modul využívající GPS pro určení aktuální pozice vozidla, její následný přenos na server pomocí GSM sítě a následné zobrazení pozice vozidla s využitím programu Google Earth. V první části jsou popsány principy určování polohy systémem GPS, princip fungování GSM sítě a princip přenosu dat po této síti. Jsou zde zmíněny podsystémy GPS a komunikační princip mezi kosmickým segmentem GPS a uživatelskou stanicí (přijímačem). Dále jsou v práci zmíněna již existující komerční i nekomerční řešení tohoto problému a jejich výhody či nevýhody. Jelikož cílem práce je vytvořit modul pro lokalizaci vozidla a zobrazit jeho pozici programem Google Earth, je zde vysvětlen algoritmus pro konverzi získaných souřadnic do formátu pro Google Earth pomocí programu v jazyce Java.

Druhá část práce je zaměřena na popis samotné realizace. Aplikace je řešena s využitím vývojového kitu Arduino, GPS a GSM modulů firem Itrax a Quectel. GPS modul zjišťuje polohu vozidla a informace o této poloze odesílá pomocí NMEA vět do mikroprocesoru. Ten kontroluje jejich validitu, a pokud je vše v pořádku, odesílá informaci pomocí GSM modulu jako UDP paket na vzdálený server. Server tvoří počítač s veřejnou IP adresou, na kterém je spuštěna Java aplikace zajišťující konverzi dat do příslušných formátů a jejich uložení na Cloud úložiště, ze kterého jsou následně zobrazena pomocí programu Google Earth. Aplikace na serveru ukládá nejen informace o aktuální poloze vozidla, ale loguje také jednotlivé trasy mezi zapnutím a vypnutím zapalování ve vozidle. Návrh řešení programu pro mikroprocesor, Java aplikaci a návrh elektronického propojení obvodů umístěných ve vozidle jsou součástí této práce.

1. Systém GPS

Global Positioning System (GPS) je systém pro určování polohy provozovaný Ministerstvem obrany Spojených států amerických. Část tohoto systému s omezenou přesností je uvolněna i pro civilní uživatele, jak uvádí (Hofmann - Wellenhof, Bernhard, 2001, s. 11). Díky GPS můžeme určit čas a polohu na zemi nebo nad zemí, s přesností do deseti metrů. Ke zvýšení přesnosti až na jednotky centimetrů lze použít metodu průměrování nebo metodu diferenčních GPS.

1.1. Struktura systému

GPS se dělí do tří základních segmentů:

- Kosmický
- řídící,
- uživatelský.

1.1.1. Kosmický segment

Kosmický segment v současné době využívá 32 družic NAVSTAR (původně byl projektován na 24 družic), které obíhají na střední oběžné dráze ve výšce 20 350 km nad povrchem Země na šesti kruhových drahách se sklonem 55°. Dráhy jsou vzájemně posunuty o 60° a družice se po nich pohybují rychlostí 3,8km/s, s dobou oběhu kolem Země 11h 58m.

Zásadní části družic NAVSTAR:

- 3 až 4 velmi přesné atomové hodiny (10^{-13}s),
- 12 antén pro vysílání kódů v pásmu L (2000 – 1000 MHz),
- antény pro komunikaci s pozemními kontrolními stanicemi v pásmu S (2204,4 MHz),
- antény pro vzájemnou komunikaci družic v pásmu UHF,
- optické, rentgenové a pulzní - elektromagnetické detektory senzory pro detekci startů balistických raket a jaderných výbuchů,
- solární panely a baterie jako zdroj energie.

V České republice se pohybuje viditelnost od šesti do dvanácti družic. Více uvádí (Hofmann - Wellenhof, Bernhard, 2001, s. 12-18).

1.1.2. Řídící a kontrolní segment

Řídící a kontrolní segment se skládá z několika částí:

- **Velitelství** – Navstar Headquarters na letecké základně Los Angeles v Californii v USA.
- **Řídicí středisko** (MSC, Master Control Station), na Schrieverově letecké základně USAF v Colorado Springs.
- **Záložní řídící středisko** (BMCS, Backup Master Control Station) umístěné v Gaithersburg (Meryland, USA) přebírá cvičně 4× do roka řízení systému, v nouzi je připraveno do 24 hodin.
- **3 povelové stanice** (Ground Antenna), které jsou umístěny na základnách USAF: Kwajalein, Diego Garcia, Ascension Island, případně i Cape Canaveral.
- **18 monitorovacích stanic** (Monitor Stations), které jsou umístěny na základnách USAF: Havaj, Colorado Springs, Cape Canaveral, Ascension Island, Diego Garcia, Kwajalein a dále stanice spravující NGA: Fairbanks (Aljaška), Papeete (Tahiti), Washington DC (USA), Quitto (Ekvádor), Buenos Aires (Argentina), Hermitage (Anglie), Pretoria (Jižní Afrika), Manama (Bahrain), Osan (Jižní Korea), Adelaide (Austrálie) a Wellington (Nový Zéland).

Pokud by došlo k výpadku řízení, například z důvodu zničení pozemních vojenských stanic řídícího a kontrolního segmentu, přejdou družice do automatického režimu AUTONAV (Autonomous Navigation Mode). V tomto režimu jsou schopny pracovat dalších 6 měsíců. Více uvádí (Hofmann-Wellenhof, Bernhard, 2001, s. 18-20)

1.1.3. Uživatelský segment

Uživatelé pomocí GPS přijímačů přijímají signál z družic, které jsou právě nad obzorem. Podle přijatých dat přijímač vypočítá a zobrazí polohu, nadmořskou výšku uživatele a také zobrazí přesný čas. Komunikace je jednosměrná, tedy pouze od družice směrem k přijímači.

Přijímače se dělí podle:

- **přijímaných pásem**
  - ✓ Jednofrekvenční,
  - ✓ dvoufrekvenční,
  - ✓ vícefrekvenční (připravují se pro pásmo L5).
- **kanálů**
  - ✓ Jednokanálové,
  - ✓ vícekanálové.
- **principu výpočtů**
  - ✓ Kódový,
  - ✓ fázový a kódový.

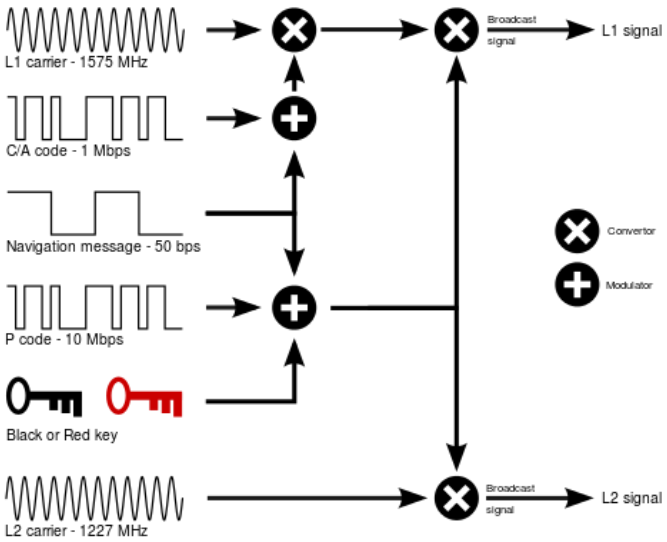
Běžné přijímače (pro negeodetické a nevojenské použití), se vyrábí jako jednofrekvenční, vícekanálové a kódové. Takový GPS přijímač se skládá z:

- Antény
- předzesilovače,
- procesoru,
- časové základny s přesností 10^{-6}s,
- komunikačního rozhraní.

1.2. Přenos signálu

Systém GPS využívá k přenosu informace od družice k přijímači rádiové signály. Každý signál odeslaný družicí je složen z nosné vlny, dálkoměrného kódu a navigační zprávy. Družice vysílají na dvou nosných frekvencích, **L1** (1575.42 MHz, vlnová délka 19cm) a **L2** (1227.60 MHz, vlnová délka 24cm). Frekvence **L1** je modulována dvěma dálkoměrnými kódy, pseudonáhodnou posloupností kódů s hodnotami +1 nebo −1 (Pseudo Random Noise – PRN).

Je to přesný P-kód (Precision nebo P-code), který může být pro vojenské účely šifrovaný, pak se mu říká Y-code a hrubý/dostupný C/A kód (Coarse/Acquisition), který se nešifruje. Frekvence **L2** je modulována pouze P-kódem, nebo šifrovaným Y-kódem (Obr. 1.). Jak uvádí (Hofmann-Wellenhof, Bernhard, 2001, s. 20-24).



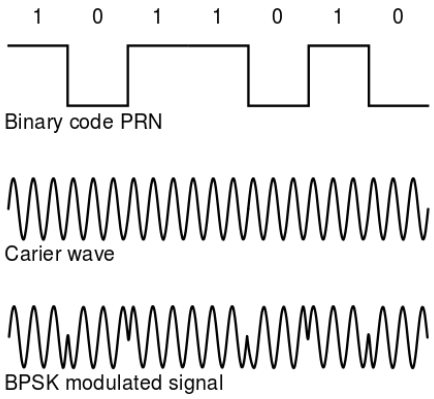
Obr. 1 Schéma modulace signálu vysílaného družicí GPS [10]

1.2.1. Modulace a demodulace

Nejběžnějším způsob, jak přenášet informace mezi body A a B, je pomocí amplitudy napětí v závislosti na informaci, po metalickém vodiči. Takto lze přenášet pouze jednu informaci v daném čase a to jen na určitou vzdálenost, protože přenosové kanály mají elektrické vlastnosti, které neumožňují přenášet informace na dlouhé vzdálenosti. Toto platí u bezdrátového přenosu pomocí elektromagnetických vln dvojnásob. Z tohoto důvodu se používají signály, které mají na daném přenosovém kanále nejlepší vlastnosti. Tyto signály se vytvářejí v modulátorech, které signály v základním pásmu převádí na signály v přeloženém pásmu. Na nosné vlně je tedy namodulována vlastní informace.

Pro přenos GPS informace se používá fázová modulace s binárním klíčováním (BPSK, Binary Phase Shift Keying) (Obr. 2.). Jedna nosná vlna nese jeden bit, který je modulován změnou fáze nosné vlny o 180 stupňů. Pokud jsou na jednu frekvenci modulovány 2 signály, je druhá modulace BPSK posunuta na nosné vlně cca o 80–100 stupňů. Protože všechny družice vysílají na stejném kmitočtu, používá se k oddělení jejich signálů kódový multiplex (CDMA, Code Division Multiple Access). Každá družice má svůj jedinečný PRN kód. Díky této modulaci vypadá vysílaný signál jako šum. Perioda nosné vlny je řádově 1 ns (10^{-9}s) a je tedy výrazně menší než doba trvání datového bitu 1 – 10 μs (1-10×10^{-6}s) PRN kódu.

Anténa přijímače zachycuje signál, jehož součástí je také pseudonáhodný generátor, který je stejný jako generátory na družicích. Pomocí DSP (Digitální Signálový Procesor) přijímač provádí synchronizaci generátoru v přijímači s generátorem na družici (korelace). V přijímači se nejprve vynásobí přijímaný signál kódem družice, ze které chceme signál demodulovat. Poté proběhne demodulace BPSK. Vynásobením se signál požadované družice obnoví, protože 1 × 1 = 1 a (−1) × (−1) = 1. Druhým násobením se úplně zruší vliv prvního násobení na družici. Signál z jiných družic se díky nekorelovanosti neobnoví, a protože vypadá jako šum, filtry ho jako šum odfiltrují. Více uvádí (Elliott D Kaplan, Christopher J Hegarty, 2006, s. 113-115).



Obr. 2 Fázová modulace metodou BPSK [11]

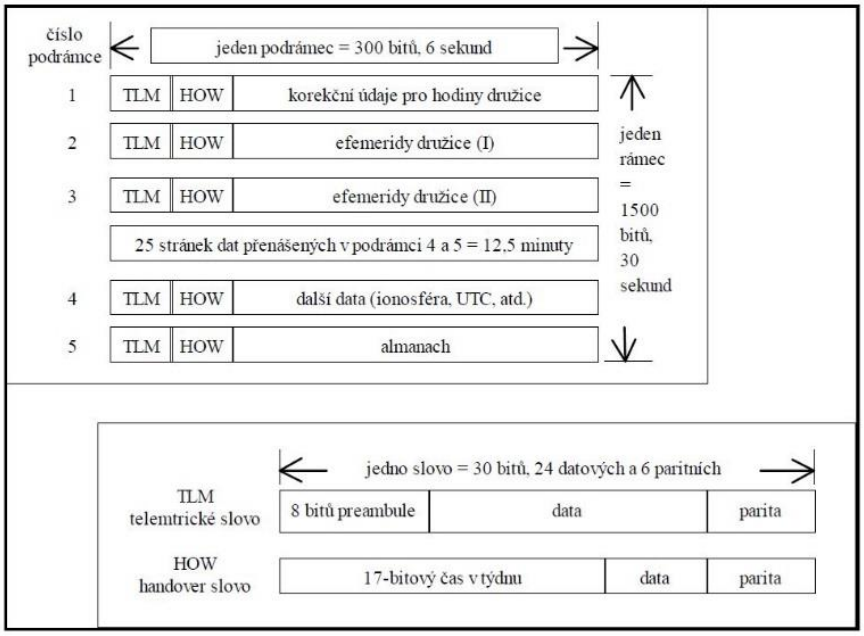
1.2.2. Navigační zpráva

Pro určení polohy GPS přijímače je potřeba znát přesnou polohu vysílající družice v době, kdy odeslala dálkoměrný kód. Poloha družice se vypočítá na základě parametrů její dráhy, které vysílá v navigační zprávě. Navigační zpráva obsahuje kromě parametrů oběžné dráhy dané družice také další údaje:

- Čas vysílání počátku zprávy,
- přesnou polohu družice v prostoru (efemeridy družice),
- údaje umožňující přesně korigovat čas vysílání družice,
- almanach (Obsahuje méně přesné parametry oběžných drah všech družic kosmického segmentu a údaje o stavu těchto družic. Přijímač je díky almanachu schopen vyhledat aktuálně viditelné družice v dané oblasti a zrychlit svůj start,
- koeficienty ionosférického modelu,
- stav družice.

Pomocí těchto údajů lze spočítat přesnou polohu družice a přesný čas odeslání přijaté sekvence dálkoměrného kódu.

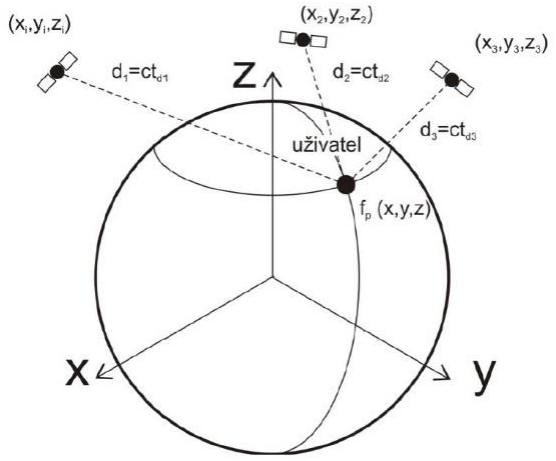
Navigační zpráva je modulována zvlášť na signál a skládá se z rámce (1500 bitů) a pěti podrámců (300 bitů) (Obr. 3.). Podrámce jsou složeny ze slov o délce 30 bitů, 24 bitů se využívá pro přenos informace a zbylých šest pro zabezpečení přenosu proti chybám. Více uvádí (Elliott D Kaplan, Christopher J Hegarty, 2006, s. 142-145).



Obr. 3 Struktura navigační zprávy GPS [2]

1.3. Určení polohy

Ke zjištění polohy přijímače je nutné znát vzdálenosti alespoň od tří družic (Obr. 4.). Vzdálenost se vypočítá z naměřeného času τ_{di}, potřebnému k přenosu signálu z družice do přijímače, jak uvádí (Elliott D Kaplan, Christopher J Hegarty, 2006, s. 54-58).



Obr. 4 Princip dálkoměrné metody [2]

Pokud známe polohu družic určenou souřadnicemi x_i, y_i, $z_{i,}$ můžeme dopočítat polohu přijímače pomocí rovnice (1). GPS lze označit jako družicový dálkoměrný rádiový systém. Existují dva dálkoměrné systémy, aktivní a pasivní. U aktivního systému je komunikace obousměrná. Zařízení vyšle požadavek na družici, ta jej identifikuje a zašle zpět odpověď pro zařízení. Následuje výpočet zpoždění mezi odesláním a přijetím zprávy. Zařízení musí znát polohu družice. Tato metoda má nevýhodu ve velké zátěži na vysílacím zařízení. V případech kdy nelze využít pozemní radiovou komunikaci, například kvůli utajení zařízení, nelze polohu zjistit.

$$\tau_{di} = \sqrt{(x_i - x) + (y_i - y) + (z_i - z)} \tag{1}$$

Komerčně používaná zařízení využívají pasivní systém, kde signál vysílá pouze družice. Vzdálenost d_i mezi přijímačem a družicí se určí pomocí výpočtu τ_{di} času mezi odesláním a přijetím signálu (rov. 2.), díky známé poloze družice, která ji buď sama vysílá, nebo je odvozena z vysílaných parametrů dráhy družice. Časové základny družice a přijímače nejsou synchronní, proto je nutné zajistit její dopočítání o rozdíl Δ_t. Ten je možné přepočítat na vzdálenost $b = c\,.\Delta_t$.

Pro výpočet se rovnice rozšíří o další neznámou a je nutné vypočítat čtyři rovnice o čtyřech neznámých. Proměnná D_i označuje takzvanou pseudovzdálenost, což je přepočet časového posunu τ_{mi} mezi časovou základnou přijímače a přijímačem generovanou kopií signálu s přijatým signálem.

$$\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} = d_i = (\tau_{mi} + \Delta_t).c = D_i + b \tag{2}$$

1.3.1. Metody zpřesňování určení polohy a času

Jak bylo řečeno v kapitole 2., část systému GPS s omezenou přesností je uvolněna i pro civilní uživatele, s přesností do deseti metrů. Některé aplikace však vyžadují vyšší přesnost, které lze dosáhnout pomocí podpůrných metod. Ke zvýšení přesnosti až na jednotky centimetrů lze použít metodu průměrování nebo metodu diferenčních GPS.

- **Průměrování**
  - ✓ Jednoduchá metoda založená na několikahodinovém měření v bodě, jehož souřadnice chceme určit. Naměřené hodnoty se zprůměrují. Praktické zkušenosti ukázaly, že po osmi hodinách měření se už přesnost nezvyšuje. Nevýhodou této metody je velká časová náročnost.

- **Diferenční GPS**
  - ✓ Tato metoda je známá pod zkratkou DGPS (Differential GPS). Je založena na relativním určování polohy, což znamená, že systém GPS používá k zpřesnění signálu pozemní referenční stanice. Tyto referenční stanice jsou umístěny na bodech s přesně známými souřadnicemi a jedná se vlastně o běžné GPS přijímače (referenční přijímače). Referenční stanice provádí měření a má k dispozici odchylku mezi právě naměřenou polohou a polohou skutečnou. Tyto odchylky jsou přenášeny jako tzv. korekce do druhého přijímače někde v terénu a jsou použity pro opravu výsledků měření. Tento postup relativního určování polohy lze použít jak v reálném čase přímo v terénu, tak i při postprocessingu (následné zpracování naměřených dat v počítači).

Tyto kapitoly obsahují pouze základní informace potřebné k pochopení problematiky. Další podrobné informace o systému GPS lze nalézt v odborné literatuře (Elliott D Kaplan, Christopher J Hegarty, 2006)

2. Systém GSM

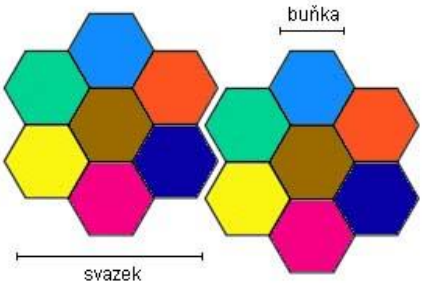
2.1. Buňkový (celulární) princip

Mobilní sítě využívají ke svému fungování rádiové vlny. Frekvence přidělované jednotlivým operátorům jsou striktně omezené. U GSM 900MHz to je 890MHz – 915MHz pro Uplink a 935MHz – 960MHz pro Downlink. V tomto frekvenčním rozsahu 25 MHz lze vytvořit pouze 125 kanálů s odstupem 200 KHz. Kanál č. 0 slouží jako oddělovací a nepoužívá se pro přenos hovorů, využitelných je tedy 124 duplexních (obousměrných) kanálů.

U GSM 1800 MHz je to 1710MHz – 1785 MHz pro Uplink a 1805MHz – 1880MHz pro Downlink. Pro tento frekvenční rozsah je kapacita 375 kanálů.

Pro každý kanál je vytvořeno pomocí metody TDMA (Time Division Multiple Access) 8 časových (time) slotů, každý časový interval představuje 1 uživatelský kanál. Celkem je tedy u systému GSM 900 k dispozici 8 x 124 = 992 duplexních kanálů, u GSM 1800 to je 3000 duplexních kanálů.

Rozsahy frekvencí nikdy nemohou postačovat na to, aby operátor mohl přidělit každému probíhajícímu hovoru ve své síti samostatný komunikační kanál. Řešením problému s nedostatkem frekvencí je vícenásobné použití stejných frekvencí za předpokladu, že se hovory využívající stejnou frekvenci nebudou vzájemně ovlivňovat. K tomu se používá tzv. buňkový (angl. celulární) princip. Jeho principem je rozdělení oblasti na jednotlivé části (buňky), s ohledem na to, aby žádné dvě bezprostředně sousedící buňky nepoužívaly stejnou frekvenci, jak uvádí Peterka ve svém článku (Peterka, 2000).



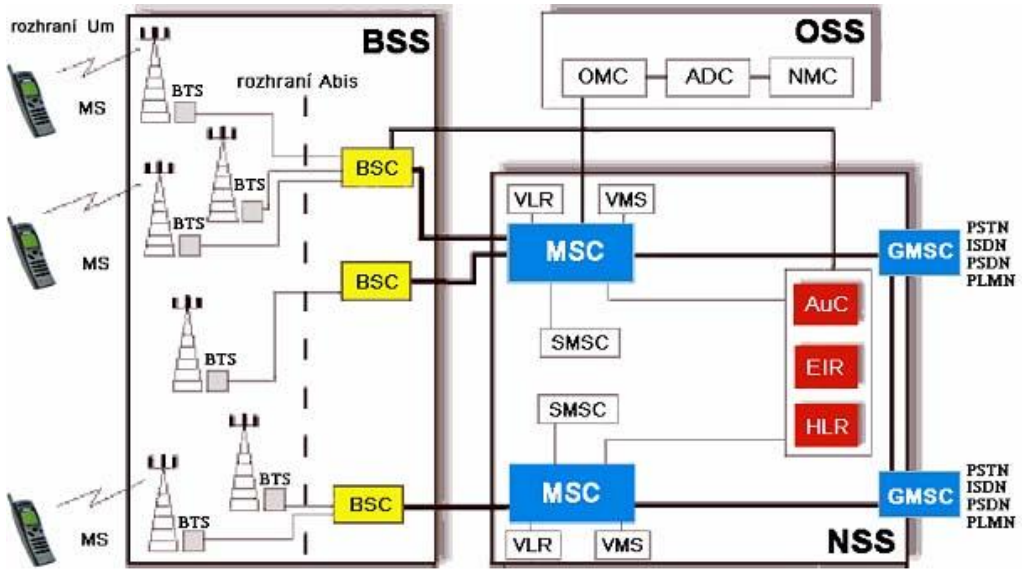
Obr. 5 Představa celulární sítě [12]

V praxi se nejčastěji setkáme s uspořádáním do šestihranných buněk připomínajících plástve medu (Obr. 5.). Na obrázku jsou jednotlivé frekvence označeny různými barvami. Seskupením buněk do svazků, které se opakují, pak lze pokrýt signálem libovolně velké území.

2.2. Architektura sítě GSM

V centru každé buňky se nachází základnová stanice (BS – base station), častěji označovaná jako BTS (Base Transceiver Station). Ta má za úkol komunikovat s jednotlivými mobilními telefony či jinými mobilními zařízeními uvnitř této buňky. Pokud se zařízení pohybuje a přemístí se z jedné buňky do druhé, BTS to poznají a dojde k předání komunikace (tzv. handover-u).

Uživatel mobilního telefonu by handover neměl vůbec poznat. Jednotlivé BTS musí být mezi sebou propojeny a mají společné řízení. Ve většině případů několik z nich sdílí společnou řídící stanici BSC (Base Station Controller).



Obr. 6 Architektura sítě GSM [12]

Základnové stanice jsou přes svoje BSC připojeny na centrální ústřednu MSC (Mobile Services Switching Centre), která slouží ke směrování jednotlivých hovorů k jejich příjemcům, sestavuje jednotlivá spojení i směrem do ostatních sítí (pokud má funkci „brány“ tj. GMSC). Kontroluje přidělení kanálů, eviduje všechny uživatele a účtují se zde hovory. K evidenci toho, ve které buňce se nachází konkrétní uživatel, slouží HLR (Home Location Register), kde jsou shromážděny údaje o všech registrovaných účastnících. Jsou zde uložena čísla IMSI (identifikační čísla SIM karty), údaje o dostupných službách a údaj o lokalitě účastníka. Každý operátor má vždy minimálně jeden registr HLR, ale může jich být i více. K evidenci vlastních uživatelů a také návštěvníků (při roamingu) slouží EIR (Equipment Identity Register), AuC (Authentication Centre) a VLR (Visitor Location Register).

EIR obsahuje tři databáze:

- White list: obsahuje známá a platná IMEI čísla,
- Black list: obsahuje IMEI čísla patřící neplatným nebo ukradeným mobilním telefonům,
- Grey list: obsahuje IMEI čísla, která je potřeba sítí sledovat.

Provoz a údržbu celého systému zajišťuje Operační a podpůrný subsystém - OSS (Operation and Support Subsystem), který se skládá z následujících částí:

- **Provozní a servisní centrum OMC (Operations and Maintenance Centre)**
  - ✓ Řídí chod ostatních subsystémů (BSS, NSS), je odpovědné za ovládání a údržbu MSC, BSC a BTS.

- **Centrum managementu sítě NMC (Network Management Centre)**
  - ✓ Podílí se na správě mobilních stanic - tyto stanice monitoruje. Zajišťuje celkové řízení toku informací v síti.

- **Administrativní centrum ADC (Administrative Centre)**
  - ✓ Podílí se na správě a managementu účastníků sítě GSM. (tarifikace účastníků, registrace (aktivace), placení účtů apod.

Více uvádí ve svém článku Jiří Peterka (Peterka, 2000).

2.3. Mobilní stanice MS

Podle specifikací GSM je mobilní stanicí (MS) jednak vlastní přijímač/vysílač (mobilní telefon) a jednak modul (karta) SIM (Subscriber Identification Module), pomocí které je účastník identifikován v síti. Každý mobilní telefon je v síti jednoznačně identifikován číslem IMEI (International Mobile Equipment). Každá SIM karta má jedinečné identifikační číslo IMSI (International Mobile Subscriber Identity), podle kterého síť pozná, kam má směrovat hovor, či které číslo potřebuje navázat telefonní spojení.

2.4. Určování polohy v GSM síti

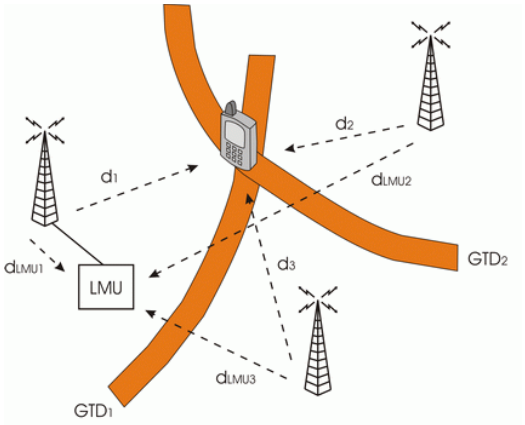
I když si tato práce neklade za cíl zjišťovat polohu v GSM síti a využívá ji pouze k přenosu dat na server, je zde vhodné metody určování polohy v GSM síti uvést. Podrobnější informace a odkazy v článku M. Orlicha (Orlich, 2006).

- **Cell ID**
  - ✓ Cell ID je identifikační číslo buňky v GSM síti, které buňce přiřazuje mobilní operátor a slouží pro určení přístupového bodu pro mobilní zařízení. Mobilní operátor zná polohu všech základových stanic s přesností přibližně 30m.
  - ✓ Přesnost zjištění polohy uživatele pomocí metody Cell ID závisí na velikosti dané buňky, která se ve městech pohybuje od 100m do 500m, mimo město i v desítkách km.

Ke zpřesnění lze využít skutečnosti, že mobilní telefon přijímá signál od více základnových stanic. Díky tomu lze vypočítat průnik buněk, které tyto stanice vytvářejí. Touto metodou se přesnost určení polohy zvýší na hodnoty okolo 300m.

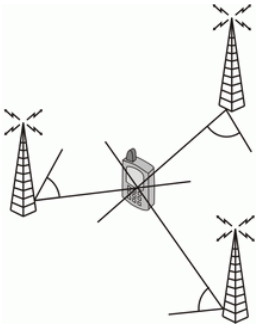
- **Timing advance**
  - ✓ Použitím parametru Timing advance (TA) může být přesnost lokalizace ještě zvýšena. TA představuje čas šíření signálu mezi mobilním zařízením a sítí a díky tomu může být zjištěna přibližná vzdálenost mezi mobilním zařízením a základnovou stanicí s přesností okolo 550m. Díky faktu, že mobilní zařízení přijímá signál od několika základnových stanic, může být poloha uživatele stanovena na základě jednoduché triangulace s přesností několika desítek metrů.

- **Enhanced Observed Time Difference**
  - ✓ Metoda Enhanced Observed Time Difference (E-OTD) sleduje časové rozdíly mezi příchody signálů od tří a více základnových stanic za předpokladu, že je zajištěna synchronnost základnových stanic v síti, což v praxi nemusí být dodrženo. Při použití metody E-OTD je síť vybavena zařízením LMU (Location Measurement Unit), které provádí měření reálných časových rozdílů vysílání signálu základnových stanic (Real Time Difference - RTD). Pokud by byla zajištěna synchronnost základnových stanic, hodnota RTD by byla nulová. Jakmile jsou stanoveny časové rozdíly v příjmu signálu, je mezi dvojicemi základnových stanic určena oblast, od které mají stejnou vzdálenost (viz. Obr. 7). V této oblasti jsou časové rozdíly příchodu signálu označované GTD (Geometric Time Difference), pro které platí vztah GTD = (d2 - d1)/c, kde c je rychlost světla a d je vzdálenost. Pro časové rozdíly zjištěné v mobilním zařízení OTD (Observed Time Difference) a GTD platí převodní vztah GTD = OTD -RTD. Pozice mobilního zařízení je pak vypočtena průnikem oblastí GTDi s přesností 30 - 300m.



Obr. 7 Oblasti konstantních časových rozdílů u metody E-OTD [7]

- **Angle of Arrival**
  - ✓ Metoda Angle of Arrival (AOA) vyžaduje použití směrových antén a znalosti vyzařovacích charakteristik. Měření úhlu, pod kterým je signál přijímán, se provádí v základnové stanici nebo mobilním zařízení. Výsledkem jsou přímky, které stanicí a mobilním zařízením procházejí. Poloha je poté stanovena jako průnik těchto přímek (Obr. 8) s přesností kolem 300m.



Obr. 8 Metoda Angle of Arrival [7]

- **Enhanced Cell Global Identity**
  - ✓ Tato metoda (E-CGI) je rozšířením metod Cell ID a Timing advance o měření úrovní signálů. E-CGI používá pro výpočet vzdálenosti mobilního zařízení od základnové stanice model šíření signálů. Podle naměřených úrovní signálů v místě mobilního zařízení a znalosti vysílacích výkonů základnových stanic jsou predikovány oblasti s nejpravděpodobnějším výskytem uživatele. Poloha uživatele bývá obvykle stanovena jako těžiště této oblasti. Přesnost metody E-CGI je kolem 50-550m pro městské oblasti a 250m-8km pro venkovské oblasti.

2.5. Datové přenosy v GSM síti

Jak uvádí Peterka ve svém článku (Peterka, 2000), v síti GSM lze při využití jednoho slotu (hovorového kanálu) přenášet data maximální rychlostí 22,8 kbps. Tuto rychlost nelze zdaleka využít k samotnému přenosu dat, protože i zde je nutná režie na opravy chyb, na navazování spojení atd. Reálná rychlost dosahuje 9,6 kbps, po oslabení mechanismů ošetřujících chyby (viz. níže) bylo dosaženo rychlosti 14,4 kbps. Každý z kanálů je s využitím časového multiplexu rozdělen na 8 slotů. Každý slot má délku 15/26 milisekund, což je přibližně 0,557 ms.

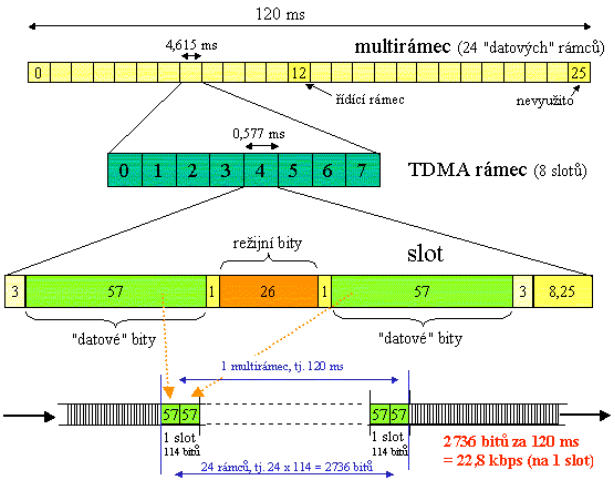
Každých 8 slotů, které lze přiřadit různým hovorům, tvoří dohromady TDMA rámec o délce 120/26 ms, což je přibližně 4,615 milisekundy. Jednotlivé rámce jsou sdružovány do tzv. multirámců, skupin po 26. Každý z multirámců trvá 120 milisekund, z toho je odvozena délka jednoho slotu $\left(\frac{120ms/26}{8}\right)$ (Obr. 9.)

Multirámce jsou mezi sebou odděleny časovou prodlevou, která odpovídá třem slotům. Z celkového počtu 26 rámců v multirámci je 24 využíváno pro vlastní přenos, 12. rámec je řídící a 25. rámec je vyhrazen.

Nejobvyklejší formát jednoho slotu je 156,25 bitů (Obr. 9.). Na konci každého rámce se nachází speciální zakončovací posloupnost o délce $8\frac{1}{4}$ bitu, z toho také vyplívá necelý počet bitů. Každý rámec je tvořen ze dvou bloků o délce 57 bitů, které jsou oddělené 1 a 3 oddělovacími bity. Mezi těmito datovými bloky se nachází další blok velikosti 26 bitů, sloužící k tomu, aby se rozpoznal užitečný signál od nejrůznějších odrazů rádiových vln, ke kterým v běžném provozu dochází.

Tyto rámce a multirámce se dají využít jako přenosový kanál pro přenos dat. Každý slot přenáší dva bloky o délce 57 bitů (pouze datové bity, bez režijních). V jednom rámci je 8 slotů a v multirámci jich je 24 (využitelných). Z toho vyplívá 2736 bitů na multirámec (2 x 57 x 24). Přenos jednoho multirámce trvá 120 milisekund. Jeden slot přenese 2736 bitů za 120 ms, což je 22800 bitů za jednu sekundu - 22,8 kbps (Obr. 9.).

Ke zvýšení rychlosti datových přenosů provozovatelé GSM sítí nejprve zjednodušili (oslabili) mechanismy ošetřující chyby během přenosu. Snížila se režie a rychlost se zvýšila na úkor spolehlivosti. Následovaly nové technologie:



Obr. 9 Rámce a multirámce v síti GSM [5]

- **HSCSD (High Speed Circuit Switched Data)**
  - ✓ HSCSD je řešení, které pro rychlejší datové přenosy v rámci GSM využívá více slotů současně. Technologie HSCSD zachovává přenos dat na principu přepojování okruhů a přidává několikanásobné zrychlení. Pracuje také s asymetrickými přenosy: rychlost směrem k uživateli může být větší než rychlost směrem od uživatele. HSCSD komunikujícím stranám přidělí více slotů současně, které zůstanou přiděleny po celou dobu spojení. Přenosová rychlost není součtem přiděleného počtu slotů. Jednotlivé GSM sloty jsou pouze jednosměrné, pro obousměrnou komunikaci musí tedy být vždy použity dva. Při použití osmi slotů to je 8 x 14,4 = 115,2 kbps duplexně. To je maximální rychlost technologie HSCSD. Další zvyšování rychlostí je limitováno omezenou velikostí rámce, který má pouze 8 slotů.

- **GPRS (General Packet Radio System)**
  - ✓ Technologie GPRS na rozdíl od přenosů dat v GSM dynamicky využívá neobsazené kanály (časové sloty). Ty potom může sdílet více uživatelů. GPRS lépe využívá dostupné prostředky, ale rychlost nebo prostupnost sítě se u této technologie obtížně garantuje. GPRS je mobilní datová síť doplněná k síti GSM. Se sítí GSM sdílí radiovou část sítě a i některé prvky páteřní sítě. GPRS nabízí teoreticky rychlost až 85,2 (down) + 42,6 kbit/s (up). Ve skutečnosti se rychlost pohybuje okolo 50 + 25 kbit/s. Pokrytí sítě GPRS v ČR je všude, kde je dostupné pokrytí signálem sítě GSM.

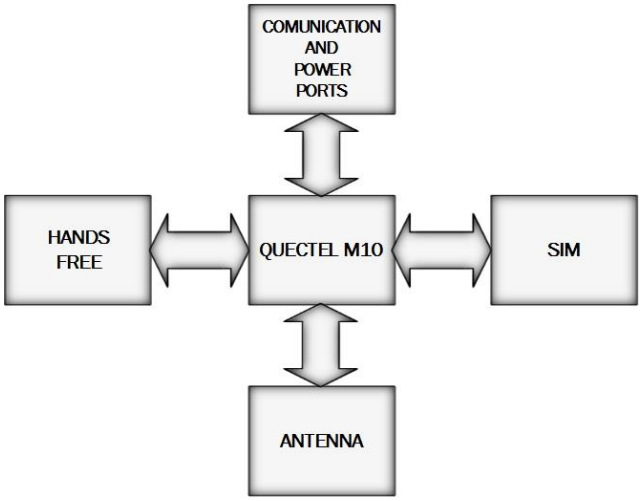
- **EDGE (Enhanced Data Rates for GSM Evolution)**
  - ✓ Technologie EDGE nabízí několik vylepšení pro dosažení efektivnějšího přenosu dat v buňkovém systému. Hlavní vylepšení spočívá v použití modulace 8-PSK (osmistavová fázová modulace). Ta dovoluje přenést tři informační bity pomocí jednoho symbolu na rádiové vrstvě. GPRS používá modulaci GMSK, která dovoluje přenést pouze jeden informační bit na jeden symbol na rádiové vrstvě. EDGE proto teoreticky nabízí rychlost až 238,6 kbit/s + 119,3 kbit/s. Ve skutečnosti lze dosáhnout rychlosti okolo 100 až 150 kbit/s + 50 kbit/s.

- **3G (UMTS, HSDPA, HSUPA, HSPA+, LTE)**
  - ✓ 3G jsou sítě třetí generace. Sítě UMTS byly poprvé definovány standardizačním orgánem 3GPP v roce 1999, ale od té doby jejich vývoj zaznamenal velký skok. Tzv. Release 5 zavedl technologii HSDPA (High Speed Downlink Packet Access), která výrazně zvýšila rychlost datových přenosů směrem k uživateli z původních 384 kbit/s na 1,8 a poté na 3,6 Mbit/s. Release 6 pak přinesl další úpravy, především zrychlení datových přenosů směrem od uživatele do sítě označované HSUPA (někdy také HSPA). Release 7 přinesl technologií HSPA+, která navyšuje přenosové rychlosti na dvojnásobek oproti HSPA (28 Mbit/s na downlinku (na buňku!) a 11,5 Mbit/s na uplinku). Zatím posledním schváleným standardem UMTS je Release 8, označovaný jako LTE (Long Term Evolution). LTE je také označováno jako mobilní síť 4. generace.

2.5.1. Řešení přenosu dat v GSM síti

Pro využití GSM sítě k datovým přenosům je nutné předávat do sítě data přímo v digitální podobě, síť se následně postará o další přenos. K přenosu dat jsou zapotřebí „GSM modemy". Jsou to různá zařízení, která fungují jako koncová zařízení sítě GSM, tedy různé mobilní telefony, GSM moduly, terminály apod. Mohou mít různá rozhraní pro připojování dalších zařízení, např. PC, mikroprocesor apod.

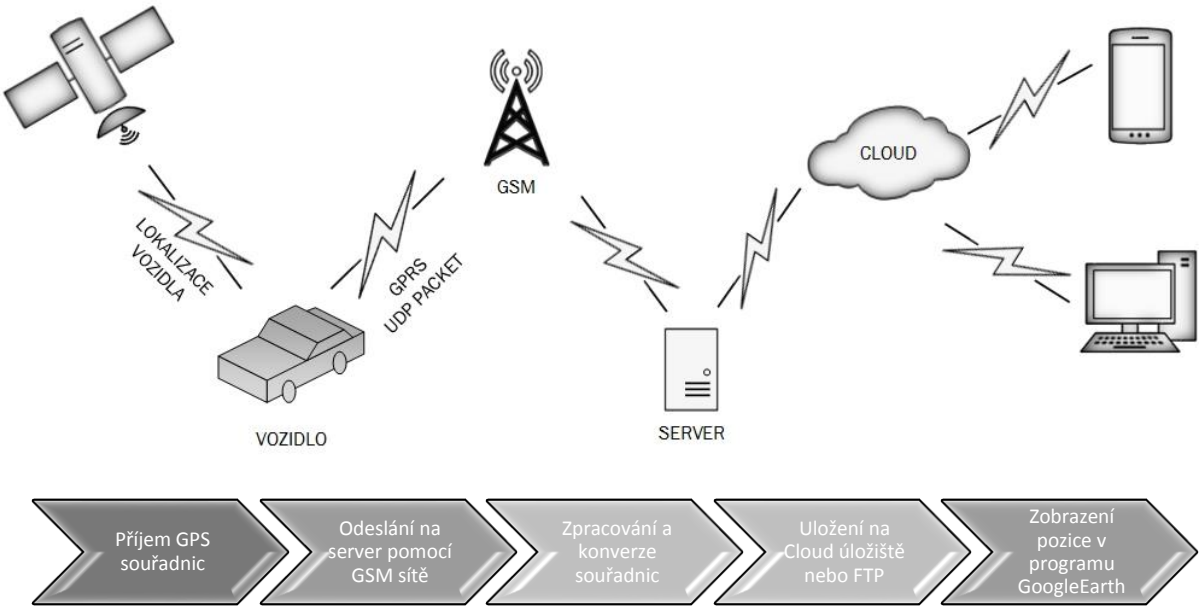
Data mohou být přenášena mezi dvěma koncovými zařízeními, nebo z koncového zařízení na server dostupný ze sítě internet, což je i v případě této práce. Pro toto řešení je využit GSM modul firmy Quectel M10 (Obr. 10.), který podporuje 850/ 900/ 1800/ 1900 MHz pásma sítě GSM a datové přenosy pomocí technologie GPRS. Modul má integrované datové služby: TCP/IP, HTTP, SMTP, FTP, SMS, MMS, WAP i fax.



Obr. 10 Blokové schéma GSM komunikátoru s modulem Quectel M10 [zdroj: autor]

3. Lokalizace vozidla s využitím GPS a přenosem dat po GSM síti.

Tato práce si klade za cíl realizovat řešení lokalizace vozidla s využitím GPS přijímače, následný přenos pozice (souřadnic) po GSM síti na server a její následné zobrazení v programu Google Earth (Obr. 11).



Obr. 11 Postup při lokalizaci vozidla [zdroj: autor]

Komunikace GPS přijímače s počítačem (mikroprocesorem) probíhá na základě NMEA (National Marine Electronics Association) protokolu. V NMEA-0183 formátu jsou data posílána po řádcích. Na začátku každého řádku je znak '$', následuje dvojpísmenná zkratka zařízení (GP = GPS) a dále kód složený ze tří písmen, který určuje formát zprávy. Na konci každého řádku je symbol hvězdičky a hexadecimálně zapsaný kontrolní součet (XOR všech znaků na řádku mezi '$' a '*'). Délka řádku je omezena na maximálně 80 znaků a jednotlivé položky jsou od sebe odděleny čárkami. Ze všech formátů NMEA vět jsou nejvyužívanější:

- RMC (základní informace o pozici),
- GGA (rozšířené informace o pozici včetně nadmořské výšky, odhadované chyby či počtu viditelných satelitů),
- GSA a GSV (informace o satelitech).

3.1. Formát RMC věty

Věta začínající RMC (Recommended minimum specific GPS/Transit data) je doporučené minimum, které by měla poskytovat většina GPS zařízení. Věta obsahuje informace o poloze (zeměpisná délka a zeměpisná šířka), dále pak informace o datu a čase naměření souřadnice a rychlost, kterou se přijímač pohybuje. Tyto informace plně postačují k řešení problému, kterým se zabývá tato práce. Význam jednotlivých částí RMC věty popisuje následující tabulka (Tab. 1).

Příklad věty může být např.:

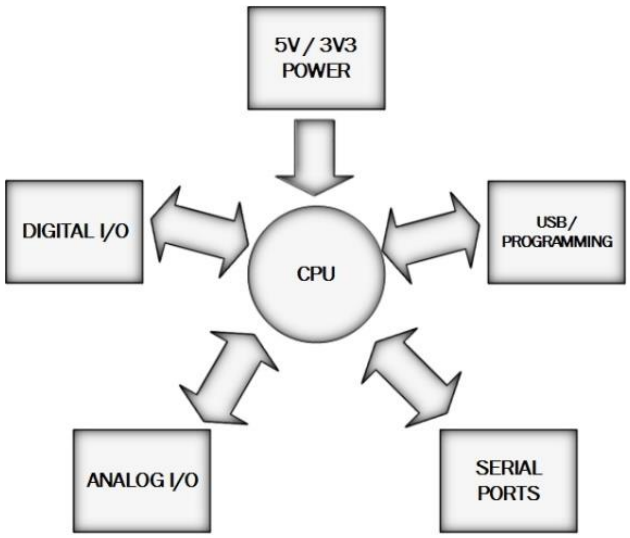
$GPRMC,170138.615,A,4912.2525,N,01635.0378,E,0.04,16.43,280705,,*32

#	Formát	příklad	komentář
1	hhmmss.sss	170138.615	Čas (UTC)
2	c	A	Status (A=OK, V=varování)
3	ddmm.mmmm	4912.2525	Zeměpisná šířka
4	c	N	Indikátor sever/jih (N=sever, S=jih)
5	ddmm.mmmm	01635.0378	Zeměpisná délka
6	c	E	Indikátor východ/západ (E=východ, W=západ)
7	d.d	0.04	Vodorovná rychlost (Speed Over Ground, v uzlech)
8	d.d	16.43	Kurz pohybu ve stupních
9	ddmmyy	280705	Datum ddmmyy
10	d.d	N.A.	Magnetická deklinace ve stupních
11	c	N.A.	Indikátor východ/západ (E=východ, W=západ)
12	*xx	32	Kontrolní součet

Tab. 1 Význam jednotlivých částí RMC věty

3.2. Zpracování RMC věty

GPS přijímač je propojen s vývojovým kitem Arduino Mega 2560 (Obr. 12.), který využívá mikroprocesor ATmega2560. Mikroprocesor neustále přijímá RMC věty z GPS přijímače a každou novou validní (Status A=OK) uloží do paměti (přepíše poslední validní větu). V paměti tedy má vždy poslední validní souřadnice a v určených časových intervalech je odesílá ve formě UDP paketu, pomocí GSM modulu Quectel M10 a prostřednictvím technologie GPRS na server (počítač s veřejnou IP adresou). Po přijetí jsou data na serveru pomocí Java aplikace konvertována do kml formátu (Google Earth placemark) a dále můžou být uložena na lokálním disku serveru, cloud úložišti, nebo odeslána na FTP server. Program Google Earth je nastaven na sledování obsahu tohoto souboru, ať už lokálně nebo na FTP serveru a při každé změně pozice (obsahu souboru) zobrazí aktuální pozici na mapě a to včetně data, času a rychlosti v okamžiku uložení souřadnice.



Obr. 12 Blokové schéma kitu Arduino Mega 2560 [zdroj: autor]

3.2.1. KML formát

Jak bylo uvedeno v předchozím oddíle, RMC věta je na serveru konvertována do kml formátu pro Google Earth. Zkonvertovaný soubor má následující formát:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns=http://www.opengis.net/kml/2.2
xmlns:gx="http://www.google.com/kml/ext/2.2"
xmlns:kml="http://www.opengis.net/kml/2.2"
xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
<Placemark>
<name>28.11.13 / 13:01 / 92 km/h</name>
<description>point description</description>
<Point>
<coordinates>16.136950198,50.324256912,0</coordinates>
</Point>
</Placemark>
</Document>
</kml>
```

Jedná se o popis konkrétního místa pro Google Earth. Důležité informace se nacházejí mezi tagy <Placemark> a </Placemark>.

- Mezi tagy <name> je název označeného místa. V tomto případě je zde uložen datum, čas a rychlost vozidla v okamžiku uložení souřadnice.
- Mezi tagy <coordinates> jsou uloženy souřadnice vozidla

Výsledné zobrazení v programu Google Earth je na předchozím obrázku (Obr. 13.). Podrobnější technické informace k řešení přenosů dat a konverze formátů jsou uvedeny ve druhé části práce.

Obr. 13 Zobrazení pozice vozidla v programu Google Earth [zdroj: Google Earth]

4. Analýza stávajících řešení

Problematikou lokalizace vozu se v České republice zabývají také firmy, ať už formou prodeje lokalizačních zařízení, nebo komplexního servisu včetně lokalizačních služeb na serverech těchto firem. V Královehradeckém kraji se jedná především o firmu LEVEL systems s.r.o. Náchod, která provozuje službu Positrex (www.positrex.cz). Lokalizační jednotka umístěná ve vozidle získává z GPS přijímače informaci o poloze. Dále pak s využitím dalších čidel připojených k jednotce, informaci o řidiči, informaci o tom, zda je vozidlo v pohybu, jakou rychlostí se právě pohybuje, údaje o tankování paliva apod. Tyto informace jsou pomocí GSM modulu posílány jako UDP pakety na server, kde jsou softwarově zpracovány. Samotné zobrazení polohy probíhá online na vlastních mapách ČR a SR, dražší varianty využívají mapy od společnosti Google. Realizace popisovaná v této práci využívá stejných metod, ale vzhledem k tomu, že si tato práce neklade za cíl vytvoření kompletní knihy jízd, jsou na server odesílány pouze NMEA věty, které obsahují, jak bylo popsáno výše, informaci o datu, čase, poloze a rychlosti.

Přibližná pořizovací cena jednotek firmy LEVEL systems s.r.o. se pohybuje od 7000,- Kč. Dále je nutné připočítat registrační poplatky 1500,- Kč a měsíční poplatek za online službu, který startuje na 200,- Kč (ceny jsou uvedeny bez DPH). Další firmou, která se zabývá touto problematikou, je například firma Secar Bohemia a. s., která nabízí službu SHERLOG Trace (www.sherlogtrace.cz). Rozsah nabízených služeb je víceméně stejný, stejně jako ceny. Knihu jízd a online sledování vozidel nabízejí také firmy DHO s.r.o. nebo Princip a.s. Jednou z dalších možností zjišťování polohy jsou GPS loggery, ať už amatérské konstrukce, jako například GPS data logger. Návod na jeho výrobu je k dispozici na stránkách elektronického magazínu Pandatron.cz. Jedná se o návod na výrobu loggeru řízeného mikroprocesorem, který sbírá data o poloze a ukládá je do externí paměti typu flash, po připojení k počítači se pomocí rozhraní RS232 dají data přesunout na pevný disk a dále zpracovat například programem Google Earth (http://pandatron.cz/?1174&gps_data_logger), nebo komerční loggery, například od výrobců MIO, HOLUX, Garmin, Canmore apod. Jejich cena startuje zhruba na 1500,- Kč. Tyto GPS data loggery také zaznamenávají trasu do paměti, ve většině případů na paměťovou kartu, ale není možné jejich polohu sledovat online. Některé z nich, pokud jsou vybaveny i GSM modulem, dokážou jednorázově odeslat svoji pozici pomocí SMS zprávy, ale v tomto případě jejich cena pochopitelně roste. Z předchozích informací je jasné, že online sledování vozidel je spojeno s nákupem sledovacího zařízení a platbou měsíčního poplatku za sledování a i když ceny za služby vzhledem ke konkurenčnímu boji neustále klesají, stále se jedná o měsíční platby minimálně 200,- Kč.

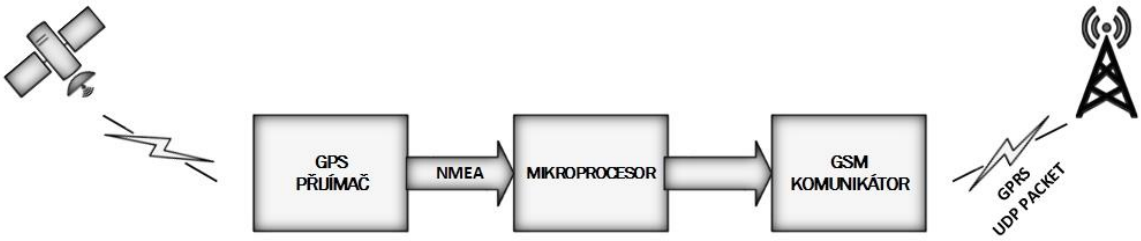
GPS data loggery, ať už komerčně prodávané, nebo amatérsky vyrobené, ve většině případů neumožňují online sledování. Řešení popisované v této práci vyžaduje pouze počítač připojený k internetu s veřejnou IP adresou, kterou většina poskytovatelů internetu ke svým tarifům nabízí zdarma. Pořizovací cena hardware pro realizaci této práce se pohybuje okolo 4000,- Kč, ale jen proto, že byl použit kit Arduino z důvodu lepšího ladění samotné aplikace. Při použití samotného procesoru by cena klesla zhruba na 2000,- Kč. Další náklady na sledování jsou už pouze poplatky za datové přenosy v síti GSM (u komerčního řešení je nutno tyto poplatky také hradit). Například při použití předplacené SIM karty od virtuálního operátora Mobil.CZ, kde je mobilní internet poskytován zdarma, tvoří roční náklady 400,- Kč za povinné dobití minimální částkou 200,- Kč v šesti měsíčních intervalech. Po přepočtu $400,-Kč/12\ měsíců$ vyjdou měsíční náklady na 33,- Kč. Orientační porovnání nákladů a parametrů je zobrazeno v následující tabulce (Tab. 2).

	GPS logger	GPS lokátor	Vlastní řešení
Klady	+ nižší pořizovací náklady + bez měsíčních poplatků	+ online sledování + bez nutnosti vlastního mapového SW	+ online sledování + přizpůsobení funkcí vlastním potřebám + nižší pořizovací náklady + nižší měsíční poplatky
Zápory	- chybí možnost online sledování - nutnost vlastního mapového sw	- vyšší pořizovací náklady - vyšší měsíční poplatky	- nutnost vlastního mapového SW
Pořizovací cena	1500,- Kč	8500,- Kč	2000,- Kč
Měsíční náklady	0,- Kč	200,- Kč	33,- Kč

Tab. 2 Orientační porovnání nákladů a parametrů komerčních a vlastního řešení

5. Návrh hardwarového řešení

Vlastní řešení popisované v této práci se skládá z HW a SW části. Tento blok popisuje HW část, tedy propojení jednotlivých elektronických obvodů a obslužný program pro mikroprocesor. Zapojení použité v této práci se skládá ze tří samostatných komponent:



Obr. 14 Jednotlivé části HW řešení [zdroj: autor]

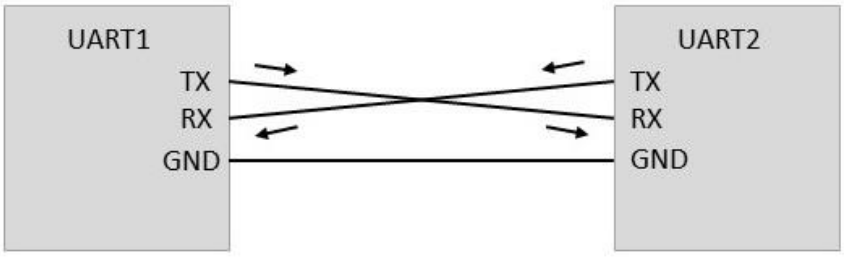
- **GPS přijímač**
  - ✓ Zajišťuje příjem informace poloze z jednotlivých družic a pomocí NMEA vět je ve vteřinových intervalech odesílá na sériový port.

- **Mikroprocesor**
  - ✓ Přijímá na sériovém portu data z GPS přijímače, kontroluje jejich validitu a v nastavených intervalech je pomocí GSM komunikátoru odesílá v podobě UDP paketů na vzdálený server.

- **GSM komunikátor**
  - ✓ Odesílá data přijatá od mikroprocesoru pomocí GPRS přenosu.

5.1. Sériová linka UART

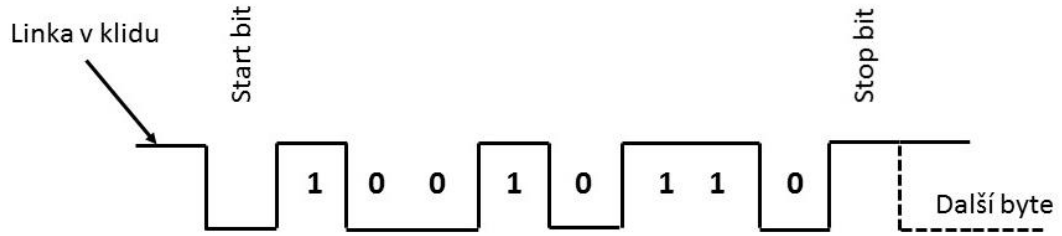
Sériová linka UART (Universal Asynchronous Receiver and Transmitter) se používá k přenosu dat mezi dvěma zařízeními. UART vysílá data na pinu označovaném jako TX (transmit), přijímá na pinu RX (receive). Jak naznačuje následující obrázek (Obr. 15).



Obr. 15 Propojení dvou zařízení pomocí UART rozhraní [zdroj: autor]

Mikroprocesor použitý v této práci využívá 5V logiku, což znamená, že log. 0 odpovídá napěťová úroveň 0V – 0,8V, log. 1 pak 2V – 5V. I přesto, že GPS modul využívá napájení pouze 3,3V, napěťové úrovně pro jednotlivé logické stavy jsou zajištěny.

Jestliže v daném okamžiku neprobíhá komunikace, je klidová úroveň signálu log. 1. Vysílání je zahájeno změnou hodnoty signálu na log. 0 po dobu jednoho bitu (tzv. start-bit). Komunikace pokračuje od nejnižšího datového bitu k poslednímu nejvýznamnějšímu datovému bitu. Následuje stop bit, který má opět úroveň log. 1. Po odvysílání stop-bitu může začít přenos dalšího bajtu. Jak je znázorněno na následujícím obrázku (Obr. 16).

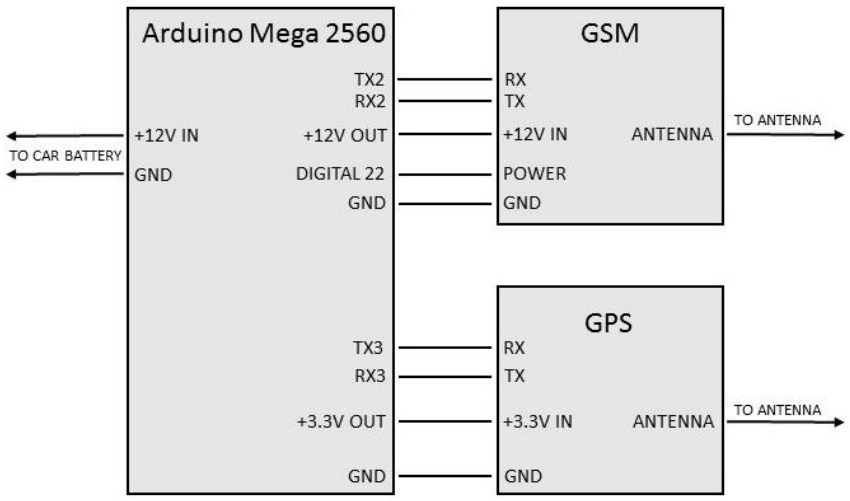


Obr. 16 Průběh komunikace na rozhraní UART [zdroj: autor]

Obě komunikující zařízení musí mít nastavenou stejnou komunikační rychlost. Standardní rychlosti jsou 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bd (bitů za sekundu). Podrobnější informace lze nalézt v knize Stanislava Pechala (Pechal, 2006, s. 53-59).

5.2. Návrh propojení jednotlivých komponent

Jak už bylo uvedeno výše, kompletní řízení celého zapojení obstarává mikroprocesor. V této práci byl použit osmi bitový mikroprocesor ATMEGA 2560 osazený na vývojovém kitu Arduino Mega 2560. Tato práce zdaleka nevyužije všech možností, které vývojový kit nabízí. Byl použit z důvodu dostupnosti kitu, který byl v minulosti využit na ladění podobných aplikací. Procesor disponuje celkem čtyřmi sériovými porty (UART), dále pak 54 digitálními a 16 analogovými vstupy/výstupy. Veškerá schémata jsou v příloze (Příloha C).



Obr. 17 Schéma propojení jednotlivých komponent [zdroj: autor]

V práci jsou využity dva sériové porty k propojení procesoru s GPS a GSM modulem. Vzhledem k tomu, že veškeré obvody jsou napájeny z baterie automobilu, je práce koncipována tak, aby při vypnutém zapalování nevybíjela baterii automobilu. Z tohoto důvodu se při vypnutí zapalování odpojí napájení všech obvodů.

Po připojení napájení (zapalování vozu v poloze ON) se obvody spustí automaticky, pouze GSM modul zůstane ve StandBy módu, a proto musí být zapnut programově, pomocí mikroprocesoru, s využitím jednoho digitálního výstupu, odesláním log. 1 v trvání 2,5s na vstup POWER GSM modulu. Propojení jednotlivých komponent je znázorněno na schématu výše (Obr. 17). Podrobná schémata jednotlivých komponent jsou přiložena v přílohách této práce. Pro správnou funkci musí být na sériových kanálech nastaveny správné komunikační rychlosti, což zajistí program procesoru popsaný v následujících kapitolách.

5.3. Obsluha GPS modulu

Pro tuto práci byl použit GPS přijímač vlastní výroby osazený GPS modulem Fastrax IT300. Modul byl vybrán na základě předchozích zkušeností autora a díky dobrému poměru cena / výkon. GPS přijímač obsahuje stabilizaci napětí na potřebných 3,3V, konektor pro připojení antény, piny pro připojení externího napájení a sériového portu. Parametry modulu IT300:

- SiRFstarIII čipová sada GSC3e / LPX,
- nízká spotřeba energie: 75mW @ 3,0 V,
- velmi vysoká sensitivita -159 dBm (Tracking),
- NMEA a SiRF binární protokoly,
- dva sériové porty,
- 1PPS výstup.
- GPIO k dispozici pro vlastní účely.

Obsluha GPS nevyžaduje žádné zvláštní požadavky, pouze je třeba zajistit správnou komunikační rychlost na sériovém portu procesoru, což je v tomto případě 4800bd. Po připojení napájení začne GPS modul automaticky odesílat na sériový port jednotlivé NMEA věty popsané v kapitole 4.1. Mikroprocesor zajistí příjem a zpracování dat (bude popsáno dále).

5.4. Obsluha GSM modulu

Pro tuto práci byl použit vývojový kit PGSM firmy Pandatron, osazený GSM modulem M10 od firmy Quectel. PGSM obsahuje všechny základní podpůrné obvody pro práci s GSM modulem (slot na SIM kartu, SMA konektor pro připojení antény, konektor Jack 3,5mm pro připojení hands-free). Připojení do zařízení je možné prostřednictvím dvou standardních pinových lišt s roztečí 2,54mm. Parametry osazeného modulu M10 jsou:

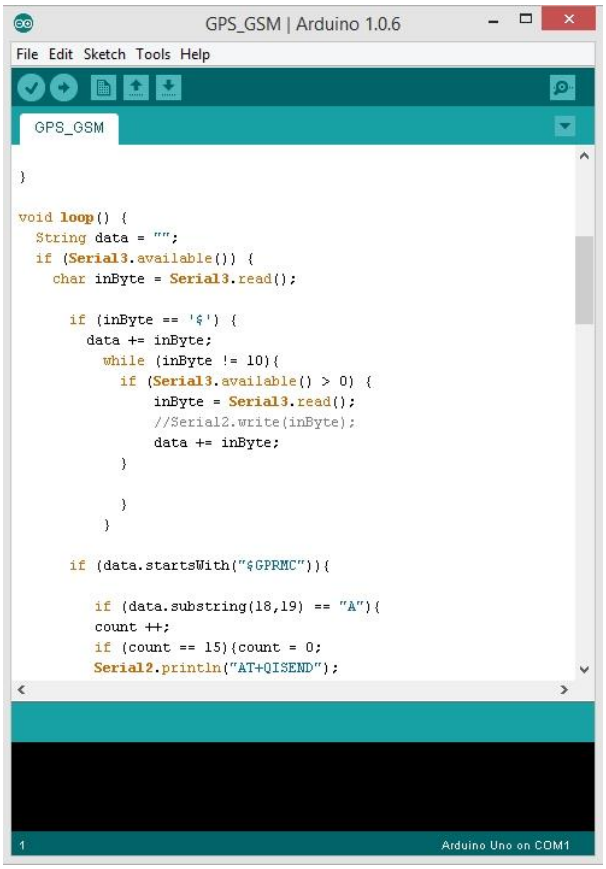
- Čtyřpásmové provedení: 850/ 900/ 1800/ 1900 MHz,
- GPRS multi-slot: Class 12/10/8,
- GPRS mobile station: Class B,
- kompatibilní s GSM: Class 4 (2W @850/ 900 MHz) a Class 1 (1W @1800/1900MHz),
- podpora hlasových služeb s odděleným výstupem pro hands-free,
- integrované datové služby: TCP/IP, HTTP, SMTP, FTP a další,
- SMS, MMS, WAP i fax,
- podpora pro LCD displej, maticovou klávesnici,
- možnost dobíjení napájecí baterie,
- telefonní seznam s funkcí vyhledávání,
- integrovaná vyzváněcí melodie,
- rozsah napájecího napětí: 3,4 až 4,5 V (4,0V typ),
- nízkopříkonový provoz: 1,1mA @ DRX=5 : 0,7mA @ DRX=9,
- rozsah provozních teplot: -45 °C až +85 °C.

GSM modul je ovládán pomocí AT příkazů odesílaných do modulu prostřednictvím sériového portu, jehož komunikační rychlost je 9600bd. AT příkazy můžou buď nastavovat parametry modulu, spouštět různé akce (vytoč hovor, pošli zprávu, navaž spojení, pošli data apod.), nebo se jimi můžeme dotazovat na různé stavy modulu (kontrola přihlášení k síti, úroveň signálu, kontrola nepřečtených zpráv a zmeškaných hovorů apod.). Následující seznam stručně popisuje AT příkazy využité v této práci. Kompletní popis všech AT příkazů modulu Qectel M10 je možno stáhnout ze stránek výrobce (http://www.quectel.com/support/downcenter.aspx).

- **AT+CGREG?**
  - ✓ Dotaz na stav registrace do GSM sítě. Odpověď modulu na tento příkaz je následující: +CGREG: <n>,<**stat**>[,<lac>,<ci>]. Důležitý je parametr **stat**, který může nabývat hodnot 0 – 5 a signalizuje stav přihlášení do GSM sítě.
  - ✓ **(0)** SIM není registrována do žádné sítě a modul aktuálně neprohledává operátory k přihlášení,
  - ✓ **(1)** SIM je registrována do domácí sítě,
  - ✓ **(2)** SIM není registrována do žádné sítě a modul aktuálně prohledává operátory k přihlášení,
  - ✓ **(3)** Registrace byla odepřena,
  - ✓ **(4)** Neznámý stav,
  - ✓ **(5)** SIM je registrována do cizí sítě – roaming.
- **AT+QIFGCNT**
  - ✓ Úvodní inicializace spojení, nastavení nosiče dat na CSD nebo GPRS.
- **AT+QICSPG=1,“název APN“**
  - ✓ Zvolí GPRS mód a nastaví název přístupového bodu pro připojení k internetu.
- **AT+QIMUX=0**
  - ✓ Zakáže funkci MUXIP – spojení bude navázáno pouze s jedním serverem, nikoli s několika najednou.
- **AT+QIMODE=0**
  - ✓ Nastaví netransparentní mód přenosu dat, zapne další protokol pro opravu chyb.
- **AT+QIREGAPP**
  - ✓ Zaregistruje TCP/IP architekturu. Implementace TCP/IP architektury.
- **AT+QIACT**
  - ✓ Aktivuje zvolený kontext.
- **AT+QIOPEN="typ spojení",IP adresa,port**
  - ✓ Naváže spojení se vzdáleným serverem
  - ✓ Typ spojení – TCP / UDP,
  - ✓ IP adresa – IP adresa vzdáleného serveru,
  - ✓ port – port, na kterém vzdálený server „naslouchá“.
- **AT+QISEND**
  - ✓ Vyšle data na vzdálený server.

5.5. Arduino IDE

Vývojové kity arduino využívají programovací prostředí Arduino IDE založené na multiplatformním open-source frameworku Wiring. Arduino IDE je psáno v jazyce Java a díky tomu je multiplatformní. Lze tedy spustit pod libovolným operačním systémem, pro který existuje Java Virtual Machine. Stručně řečeno to znamená, že lze Arduino programovat ze systémů jako GNU/Linux, Mac OS X i MS Windows. Programovacím jazykem je již zmíněný Wiring, který má téměř stejnou syntaxi jako C/C++. Kompletní dokumentaci včetně popisu všech instrukcí lze nalézt na oficiálních stránkách (http://wiring.org.co/). Arduino IDE slouží jak k napsání programu, tak i k naprogramování samotného procesoru. Prostředí lze stáhnout z oficiálních stránek (http://arduino.cc). Po instalaci stačí připojit příslušnou Arduino desku pomocí USB portu k počítači a vše je připraveno k práci.



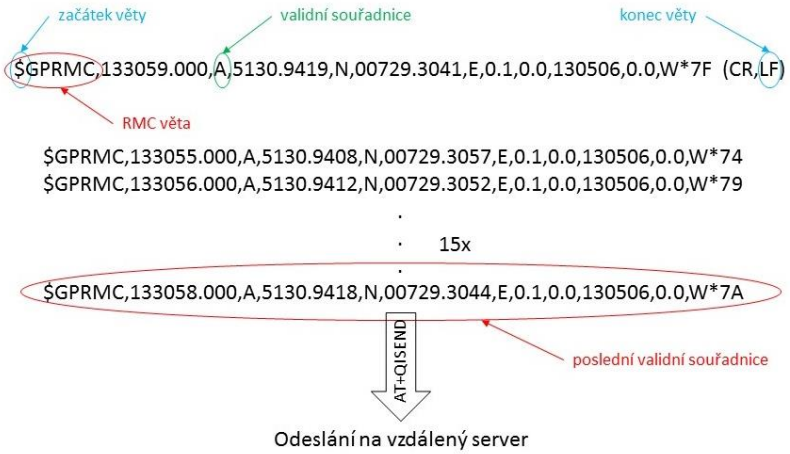
Obr. 18 Programovací prostředí Arduino IDE [zdroj: Arduino IDE]

5.6. Řídící program mikroprocesoru

Tato kapitola popisuje jednotlivé metody řídícího programu mikroprocesoru, které jsou, jak bylo uvedeno výše, psány v programovacím jazyce Wiring. Program zajišťuje příjem souřadnic z GPS modulu, kontrolu validity souřadnic a následné odeslání na vzdálený server k dalšímu zpracování. Níže jsou popsány některé zásadní metody, kompletní výpis programu je v příloze (Příloha A).

- **setup()**
  - ✓ Uvnitř této metody se provádějí veškerá prvotní nastavení. Přenosová rychlost sériového portu č. 2, ke kterému je připojen GSM modul, je nastavena na 9600bd, rychlost sériového portu č. 3, ke kterému je připojen GPS přijímač, je nastavena na 4800bd. Dále je zde nastaven pin *PwrOn* (digital pin č. 22), který slouží k zapnutí GSM modulu, jak bylo popsáno v kapitole č. 6.2., jako výstupní. Dále jsou volány metody *Pwr(), SetConection() a SendStart().*

- **loop()**
  - ✓ Instrukce uvnitř této metody se provádějí opakovaně a jedná se vlastně o hlavní program, ze kterého jsou volány další metody. Pokud jsou dostupná data v bufferu sériového portu č. 3 (pokud nedojde k poruše GPS modulu, měly by být dostupné vždy - viz. Kapitola 6.3.), tak metoda kontroluje každý přijatý znak uložený v proměnné *inByte*, zda se nejedná o „$“ (začátek věty). Poté čte další jednotlivé znaky až po „LF“ (konec věty), které postupně připojuje do proměnné *data*. Jakmile je věta kompletní, provede kontrolu, zda se nejedná o RMC větu. Pokud ano, zkontroluje validitu souřadnice, a když je souřadnice validní, přičte k počítadlu RMC vět (proměnná *count*) jedničku a pokračuje znovu od začátku. Jakmile proměnná count dosáhne hodnoty 15, odešle poslední validní RMC větu, uloženou v proměnné *data*, pomocí příkazu AT+QISEND na vzdálený server (Obr. 19).



Obr. 19 Zpracování RMC vět mikroprocesorem [zdroj: autor]

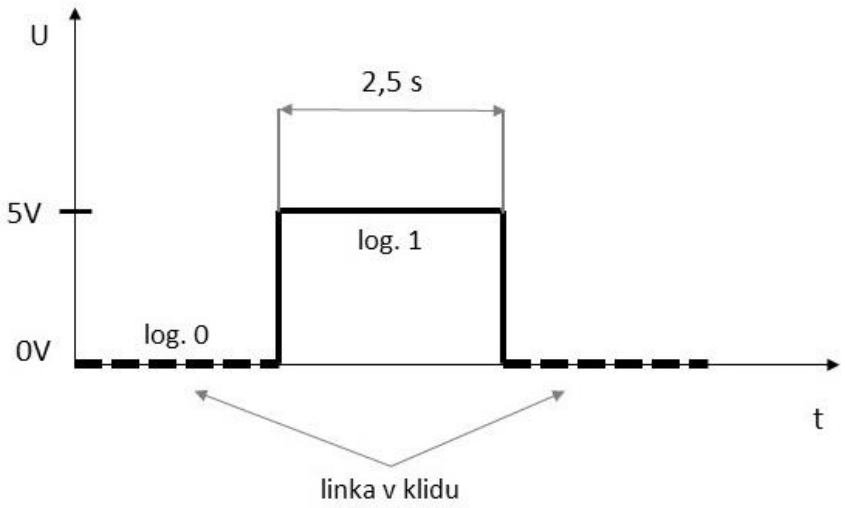
```
if (inByte == '$') {
  data += inByte;
    while (inByte != 10){
      if (Serial3.available() > 0) {
          inByte = Serial3.read();
          data += inByte;
      }
    }
 }
```

Čte celou větu od „$“ po „LF“ a ukládá do proměnné data

```
if (data.startsWith("$GPRMC")){
    Serial1.println(data);
    if (data.substring(18,19) == "A"){
          count ++;
          if (count == 15){count = 0;
                SendData();
                SendChck();
   }
          }

}
```

Pokud věta začíná „$GPRMC“ a obsahuje validní data inkrementuje počítadlo a každou patnáctou větu odešle na server

- **Pwr()**
  - ✓ Prvotní spuštění GSM modulu a test přihlášení do GSM sítě. Metoda nastaví digitální pin č. 22 do log. 1 po dobu 2,5s, jak znázorňuje následující obrázek (Obr. 20).



Obr. 20 Průběh spouštěcího pulsu GSM modulu, na vstupu POWER [zdroj: autor]

Tento puls přivedený na vstup POWER, GSM modulu, zajistí jeho zapnutí. Poté procesor čeká 15s na přihlášení SIM karty do GSM sítě. Následně odešle do GSM modulu, pomocí sériového portu č. 2 příkaz AT+CGREG?, popsaný v kapitole 6.4. a čte odpověď GSM modulu. Toto se opakuje ve 150ms intervalech, dokud se SIM karta nepřihlásí do domácí nebo cizí sítě (n =1, nebo n=5).

- **SetConection()**
  - ✓ Metoda naváže GPRS spojení se vzdáleným serverem. Po odeslání sekvence AT příkazů, popsaných v kapitole 6.4., metoda ověří, zda bylo spojení úspěšné. Pokud ne, příkazy se opakují až do úspěšného navázání spojení se serverem. Jakmile je spojení navázáno, následuje odeslání slova „Start“ na server, což signalizuje začátek nové trasy (předcházelo mu vypnutí zapalování do polohy OFF).

```
void SetConection(){
cntData = false;
while(cntData == false){
	Serial2.println("AT+QIFGCNT");
	Read();

	Serial2.println("AT+QICSPG=1,\"internet.t-mobile.cz\"");
	Read();

	Serial2.println("AT+QIMUX=0");
	Read();

	Serial2.println("AT+QIMODE=0");
	Read();

	Serial2.println("AT+QIREGAPP");
	Read();

	Serial2.println("AT+QIACT");
	Read();

	Serial2.println("AT+QIOPEN=\"UDP\",\"IP adresa serveru\",17");
	Read();
	ChckConectData();
}

}
```

- **Read()**
  - ✓ Tato metoda čte odpovědi z GSM modulu. Dokud jsou dostupná data v bufferu sériového portu, čte ze sériového portu jednotlivé znaky, které spojuje do výsledné proměnné *rdData* a do proměnné `l` ukládá délku načtených dat.

```
void Read(){
rdData = "";
char inByte;
while (Serial2.available() <= 0){}
	while (Serial2.available() > 0) {
	inByte = Serial2.read();
	rdData += inByte;
	delay(150);
}
l = rdData.length();
Serial1.println(rdData);
}
```

6. Návrh softwarového řešení

Tato část práce se zabývá SW řešením popisované problematiky. Souřadnice odeslané modulem umístěným ve vozidle, jak bylo popsáno v předcházejících kapitolách, je nutné zpracovat a připravit pro zobrazení v programu GoogleEarth. Pro uložení souborů je použito, s přihlédnutím na dostupnost dat z více zařízení, cloudové úložiště Dropbox.

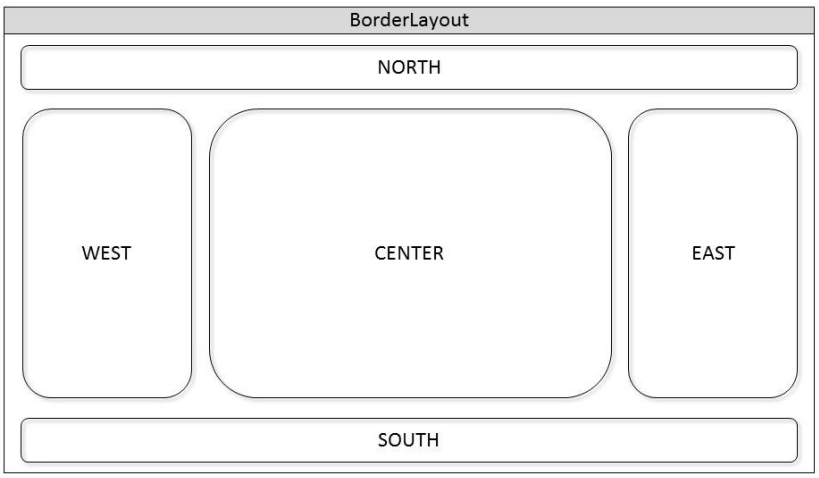
6.1. Popis programu

Pro příjem a zpracování dat odeslaných z vozidla slouží JAVA aplikace, která je spuštěna na počítači s veřejnou IP adresou, fungujícím jako server pro zpracování dat. Programovací jazyk JAVA byl vybrán na základě předchozích zkušeností autora a z důvodu jeho multiplatformního použití.

Program se skládá ze dvou částí. Z grafického uživatelského rozhraní (GUI) a samotného zpracování dat. GUI slouží k uživatelskému zadání parametrů důležitých pro běh aplikace. Druhá část programu zajišťuje samotný příjem dat, jejich konverzi do *.kml formátu a uložení do Dropboxu.

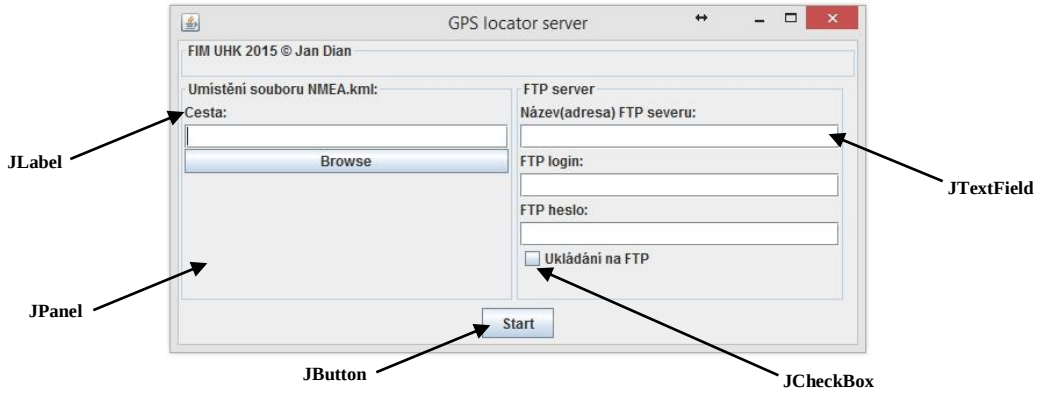
6.1.1. Grafické uživatelské rozhraní (GUI)

Grafické uživatelské rozhraní je vytvořeno pomocí knihovny Swing. Pomocí Swingu je možno v Javě vytvářet okna, dialogy, tlačítka, rámečky, rozbalovací seznamy, atd. Hierarchie tříd grafických komponent Swingu je založena na AWT (Abstract Window Toolkit) a je nedílnou součástí Java SE od verze 1.2. Do té doby byl Swing dostupný pouze jako knihovna k samostatnému stažení. Samotná aplikace využívá třídu JFrame, což je základní kontejner Swingu pro tvorbu aplikačních oken s titulkem, dále pak JPanel, na který lze umisťovat další komponenty. Pro rozvržení komponent je využit BorderLayout, který rozděluje kontejner na pět oblastí, jak ukazuje následující obrázek (Obr. 21).



Obr. 21 BorderLayout - rozvržení okna [zdroj: autor]

Aplikace využívá oblast NORTH pro zobrazení informací o autorovi, oblast WEST pro nastavení cesty k uložení souborů, oblast CENTER pro nastavení parametrů připojení k FTP serveru a v oblasti SOUTH je umístěno tlačítko pro spuštění serveru. V každé oblasti je umístěn JPanel, na něm pak další komponenty (Obr 22).



Obr. 22 Grafické uživatelské rozhraní programu a jednotlivé komponenty [zdroj: autor]

- **JLabel**
  - ✓ Slouží k zobrazení statického textu v okně, doplňuje text ke komponentám bez vlastního popisu.
- **JTextField**
  - ✓ Textová komponenta sloužící jako jednořádkové vstupní pole.
- **JCheckBox**
  - ✓ Zaškrtávací pole.
- **JButton**
  - ✓ Tlačítko, ke kterému lze nastavit tzv. listener pro spouštění metod při kliknutí.

Cestu umístění souboru s aktuální souřadnicí lze zadat buď ručně, nebo s využitím komponenty JFileChooser, po kliknutí na tlačítko Browse. JFileChooser poskytuje GUI pro pohyb v souborovém systému a následný výběr souboru, nebo adresáře.

Pro spuštění aplikace je nutné nastavit cestu k souboru `nmea.kml`, ve kterém je uložena informace o aktuální pozici vozidla. Při požadavku na ukládání souboru také na FTP server je nutné zaškrtnout příslušný checkbox a vyplnit parametry připojení. Aplikace se následně spustí kliknutím na tlačítko Start. Další podrobné informace o knihovně SWING a popis jednotlivých komponent lze nalézt v dokumentaci na stránkách společnosti Oracle (http://docs.oracle.com/javase/6/docs/api/javax/swing/package-summary.html).

6.1.2. Příjem a zpracování dat

Vlastní program je rozdělen do několika metod zajišťujících jednotlivé části, příjem UDP paketu, konverze přijatých dat do souboru `*.kml` a jeho uložení, úprava a uložení souboru s trasou jízdy, uložení souborů na FTP, je-li povoleno.

- **udpComn()**
  - ✓ Metoda zajišťuje příjem UDP paketů, vyslaných GSM modulem z vozidla. Metoda využívá třídu DatagramChannel, která umí odesílat datagramy, přijímat je a naslouchat na určitém UDP portu. Po přijetí jsou data uložena v bufferu a následně čtena po znacích a ukládána do proměnné `gps`.

```
try {
udpChannel = DatagramChannel.open(); //otevřít UDP kanál
udpChannel.socket().bind(new InetSocketAddress("0.0.0.0", 17));
//začít naslouchat

      buffer = ByteBuffer.allocateDirect(512); //alokovat
       buffer
       udpChannel.receive(buffer);
       buffer.flip();

      while (buffer.hasRemaining( )) {

       int i = buffer.get( );
       char znk = (char) i;
       gps = gps + znk;

       }

           buffer.clear(); //vymazat obsah bufferu
}
```

Zpracování dat po jejich přijetí

- **gpsConv()**
  - ✓ Data přijatá pomocí předchozí metody tvoří NMEA větu, ve které jsou jednotlivé informace odděleny čárkami. Tato metoda načítá data mezi oddělovači a upravuje je pro uložení do `nmea.kml` souboru. Přepočítá údaj o poloze z formátu „Deg° Min“ na „Degrees“, dále přepočítá údaj o rychlosti z uzlů na kilometry za hodinu, doplní oddělovače do informace o datu a čase.

```
double gps1 = Double.valueOf(minn).doubleValue();
gps1 = gps1 * 0.01666; // přepočet minut na desetiny stupně
double gps2 = Double.valueOf(mine).doubleValue();
gps2 = gps2 * 0.01666; // přepočet minut na desetiny stupně
gps1 = Double.valueOf(stn) + gps1;
gps2 = Double.valueOf(ste) + gps2;

------------------------------------------------------------

double speed1 = Double.valueOf(speed);
speed1 = speed1 * 1.852; // přepočet rychlosti z uzlů na kilometry
```

- **nmeaRewrite()**
  - ✓ Tato metoda opravuje data v souboru `nmea.kml`, po jejich konverzi metodou gpsConv(). Pomocí třídy FileInputStream načte postupně obsah souboru `nmea.kml` a uloží ho do proměnné typu String.

```
FileInputStream fIn = null;
try {
	fIn = new FileInputStream(inpName);
	nmea = "";
	int b;
	while ((b=fIn.read())!=-1) {
		char znak = (char)b;
		nmea = nmea + znak;

	}
```

Následně vyhledá pozici tagů `<coordinates></coordinates>`, mezi které uloží aktuální GPS souřadnice vozidla. Mezi tagy `<name></name>` uloží informaci o datu, čase získání souřadnice a aktuální rychlost vozidla v čase měření.

```
first = nmea.lastIndexOf("<coordinates>") + 12;
last = nmea.lastIndexOf("</coordinates>");
firstn = nmea.lastIndexOf("<name>") + 5;
lastn = nmea.lastIndexOf("</name>");
lenght = nmea.length();
nmea1 = "";
nmea2 = "";
nmea3 = "";
for (int z=0; z<=firstn;z++){
nmea1 = nmea1 + nmea.charAt(z);
}
for (int z=lastn; z<=first;z++){
nmea2 = nmea2 + nmea.charAt(z);
}
for (int z=last; z<lenght;z++){
nmea3 = nmea3 + nmea.charAt(z);
}

nmea_klm = nmea1 + info + nmea2 + gps + nmea3;
```

Po úpravě obsahu souboru metoda pomocí třídy FileOutputStream přepíše obsah souboru `nmea.kml`.

```
fOut = new FileOutputStream(outpName);
for (int z=0;z<nmea_klm.length();z++){
	Integer zn = Integer.valueOf(nmea_klm.charAt(z));
	fOut.write(zn);
}
```

- **roadRewrite()**
  - ✓ Aplikace zaznamenává nejen aktuální pozici vozidla, ale ukládá také jeho trasu. Při zapnutí zapalování odešle GSM modul umístěný ve vozidle slovo „Start“. Pokud metoda runServer() zjistí přítomnost těchto dat v bufferu, založí nový soubor, jehož název je vytvořen z aktuálního data a času a jeho první souřadnici opatří popiskem Start, aby bylo zřejmé, kde trasa začala. Do příchodu dalšího slova „Start“ ukládá každou přijatou souřadnici do tohoto souboru. Metoda obdobným způsobem jako nemeaRewrite() edituje soubor `road.kml`, do kterého postupně zapisuje souřadnice trasy vozidla.

```
FileInputStream fIn = null;
try {
	fIn = new FileInputStream(rinpName);
	nmea = "";
	int b;
	while ((b=fIn.read())!=-1) {
		char znak = (char)b;
		nmea = nmea + znak;
	}
	first = nmea.indexOf("<coordinates>") + 12;
	last = nmea.indexOf("</coordinates>");
	lenght = nmea.length();
	nmea1 = "";
	nmea2 = "";
	for (int z=0; z<last;z++){
		nmea1 = nmea1 + nmea.charAt(z);
	}

	for (int z=last; z<lenght;z++){
		nmea2 = nmea2 + nmea.charAt(z);
	}
	String nmeaStart = "</coordinates>"+ "\r\n"
	+"</LineString></Placemark>"+ "\r\n" +
	"<Placemark>"+ "\r\n"
	+"<name>Start</name><Point><coordinates>"+ "\r\n";

	String nmeaStart1 = "\r\n" + "</coordinates>" + "\r\n" +
	"</Point></Placemark>"+ "\r\n" +
	"</Document></kml>";

	fOut = new FileOutputStream(routpName);
	if (start){start = false; nmea_klm = nmea1 +
		lastgps + "\r\n" + gps + "\r\n" + nmeaStart + lastgps +
		nmeaStart1;}

	else {nmea_klm = nmea1 + gps + "\r\n" + nmea2;}

		for (int z=0;z<nmea_klm.length();z++){
			Integer zn = Integer.valueOf(nmea_klm.charAt(z));
			fOut.write(zn);
	}
	lastgps = gps;

}
```

- **sendFtp()**
  - ✓ Pokud je požadováno uložení souboru `nmea.kml` na FTP server, metoda pomocí třídy FTPClient naváže spojení s příslušným serverem a uloží soubor.

```
FTPClient client = new FTPClient();
FileInputStream fis = null;


    try {
        client.connect(ftpHost);
        System.out.println(ftpHost);
        client.login(ftpLog, ftpPsw);
        System.out.println(ftpLog + ", " + ftpPsw);
        String filename = nmeaPth + "nmea.kml";
        fis = new FileInputStream(filename);
        client.storeFile("nmea.kml", fis);
        fis.close();
        client.logout();
    }
```

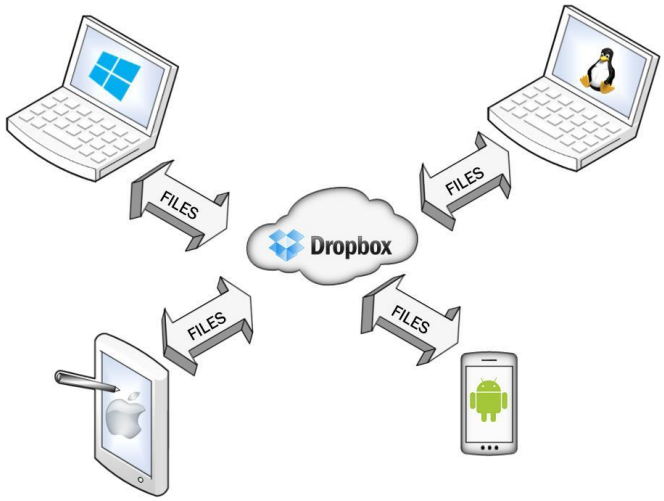
- **runServer()**
  - ✓ Aplikace zaznamenává nejen aktuální pozici vozidla, ale ukládá také jeho trasu. Při zapnutí zapalování odešle GSM modul umístěný ve vozidle slovo „Start“. Pokud metoda zjistí přítomnost těchto dat v bufferu, založí nový soubor, jehož název je vytvořen z aktuálního data a času. Do příchodu dalšího slova „Start“ ukládá, voláním metod popsaných výše, každou přijatou souřadnici do tohoto souboru.

```
while (true){
       udpComn();
       gpsConv();
       now = new Date();
       SimpleDateFormat DATE_FORMAT1 = new SimpleDateFormat("yy-MM-dd-
       HH-mm");
       dateForFile = DATE_FORMAT1.format(now);
       String cesta = nmeaPth;
       String file = cesta + dateForFile + ".kml";

       if (start == true){rinpName = nmeaPth + "road.kml";files =
       file;}

       else {file = files; rinpName = file;}
       String routpName = file;
       nmeaRewrite(inpName,outpName);
       roadRewrite(rinpName,routpName);
       System.out.println(ftpOnOff);
       if (ftpOnOff){sendFtp();}
}
```

Kompletní výpis programu Java aplikace je přiložen v příloze (Příloha B).

6.2. Dropbox

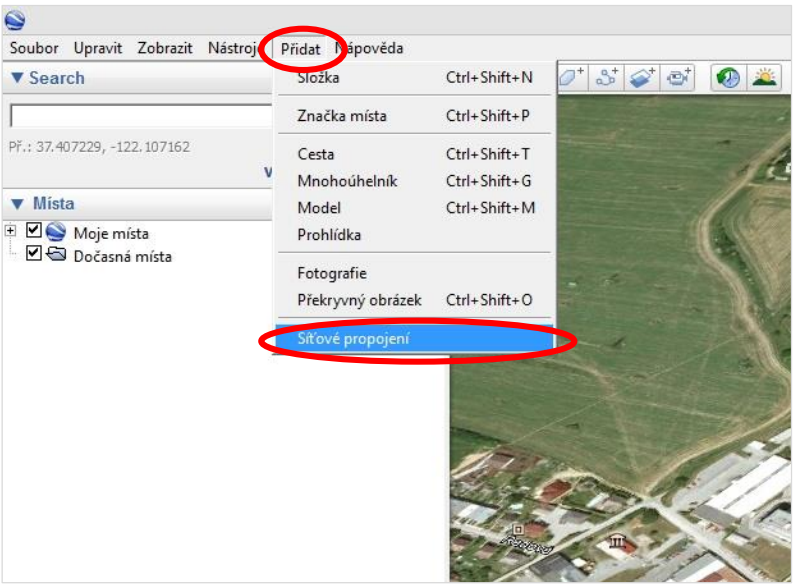
Z důvodu požadavku na dostupnost souboru se souřadnicemi z různých platforem a z více zařízení najednou, je pro ukládání zvoleno cloud úložiště. Pro účely této práce byly testovány úložiště GoogleDrive, Copy.com a Dropbox. Poslední zmiňovaný dosahoval nejvyšší rychlosti synchronizace, a proto byl pro tyto účely vybrán. Ostatní zmíněné lze však také použít. Dropbox v současnosti podporuje zhruba 10 platofrem: Windows, Mac, několik verzí Linuxu a hlavní mobilní platformy Android, iPhone, iPad a BlackBerry. Existují také neoficiální klienti na okrajové operační systémy jako Maemo, Symbian. Dropbox klient umožňuje vložit jakýkoliv soubor do určené synchronizované složky Dropbox ve správci souborů. Dojde tak k nahrání tohoto souboru do webového úložiště a rozšíření na všechna synchronizovaná zařízení, jak ukazuje následující obrázek (Obr. 23). Uživatelé také mohou odeslat data ručně přímo přes webové rozhraní a webový prohlížeč.



Obr. 23 Dropbox - synchronizace souborů [zdroj: autor]

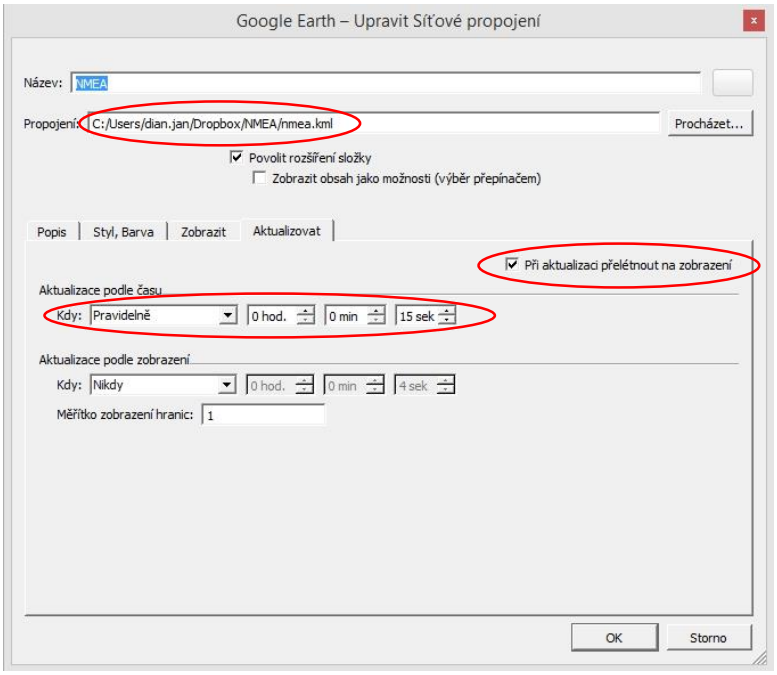
6.3. Google Earth

Pro zobrazování aktuální pozice vozidla a uložených je pro tuto práci využíván software Google Earh. Tento program byl vybrán z důvodu kompletního mapového pokrytí celého světa, navíc ve freeware verzi. Program zvládá i online zpracování souboru se souřadnicemi, dále pak automatickou aktualizaci polohy na základě změny obsahu sledovaného souboru. Další funkcí je zobrazení jednotlivých tras, uložených do souboru v `*.kml` formátu. Google Earth je dostupný pro různé platformy, v současné době je to Windows, Linux a Mac OS. Podporovány jsou i mobilní platformy Android a iOS. Po instalaci programu Google Earth na stolní počítač je nutné nastavit sledování souboru a četnost aktualizace jeho obsahu. Vzhledem k tomu, že Dropbox klient vytvoří složku ve správci souborů, jak bylo uvedeno v kapitole 7.2., cesta k souboru bude směřovat právě tam. Dále je nutné nastavit pravidelnou kontrolu obsahu souboru a přelétnutí mapy na aktuální pozici při jeho změně. Postup při nastavení programu Google Earth je na obrázcích (Obr. 24) a (Obr. 25).



Obr. 24 Google Earth - přidání síťového propojení [zdroj: Google Earth]

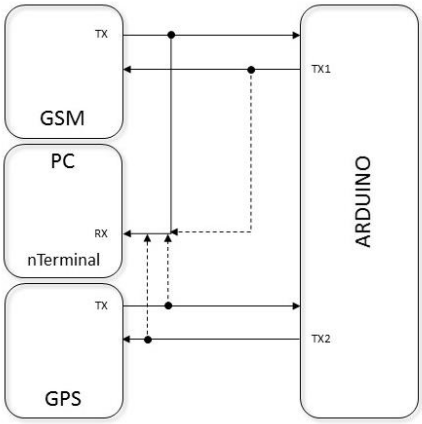
S prodlevou mezi aktualizacemi obsahu lze experimentovat, po několika pokusech se osvědčil čas 15s. Po tomto nastavení mapa automaticky přelétne na aktuální pozici pokaždé, když jí Java aplikace změní v souboru `nmea.kml`. Verze programu pro Android síťové propojení nepodporuje, ale aktuální pozici lze jednoduše zobrazit kliknutím na soubor `nmea.kml` v mobilním Dropbox klientovi.



Obr. 25 Google Earth - nastavení síťového propojení [zdroj: Google Earth]

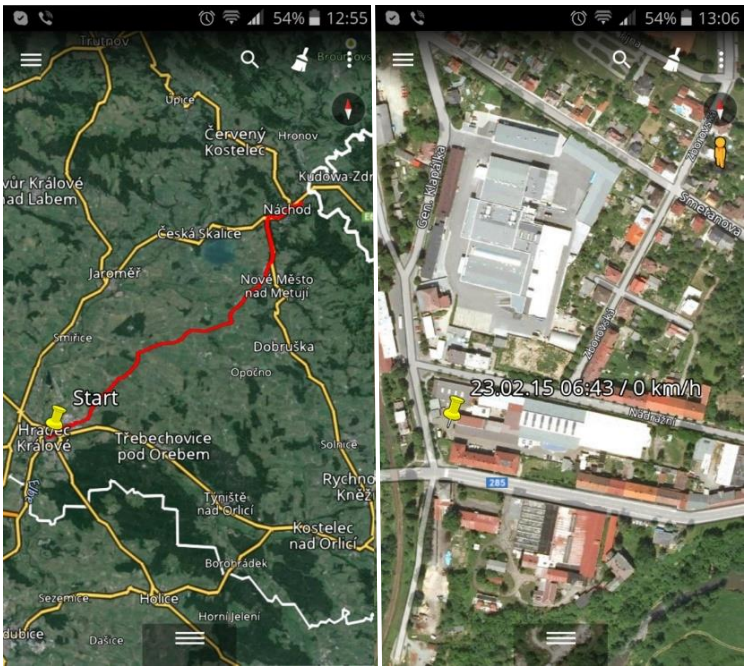
7. Testování aplikace

Pro účely testování HW části aplikace byl vývojový kit Arduino připojen k sériovému portu počítače a pomocí programu nTerminal byla sledována komunikace mezi mikroprocesorem a jednotlivými moduly. Při ladění programu pro mikroprocesor byl do programu vložen kód pro odesílání některých proměnných do sériového portu počítače, kde byl zobrazen pomocí již zmíněného programu nTerminal (Obr. 26).



Obr. 26 Sledování komunikace mezi moduly při ladění aplikace [zdroj: autor]

Při testování SW části aplikace, byl HW modul nahrazen programy Packet Sender pro PC, a mobilní aplikací UDP sender, pro platformu Android. Oba programy umožňují odesílat UDP pakety na konkrétní IP adresu a port, při ladění aplikace tak simulovaly odesílání UDP paketů GSM modulem.



Obr. 27 Zobrazení trasy a aktuální polohy v mobilní aplikaci [zdroj: Google Earth]

Pro účely testování aplikace byla jednotka s GPS přijímačem a GSM modulem umístěna do vozu autora a přibližně během čtyř měsíců byla sledována poloha a zaznamenávány trasy vozidla. Zobrazení polohy a trasy v mobilním telefonu je vidět na obrázku (Obr. 27).

Po analýze nasbíraných dat byly zjištěny výpadky, které způsobilo přerušení spojení se serverem z důvodu slabého, nebo chybějícího signálu GSM sítě. Na základě těchto zjištění byl řídící program mikroprocesoru upraven tak, aby bylo při takovémto výpadku, po obnovení GSM signálu, spojení opět navázáno. Po těchto úpravách k výpadkům již nedocházelo.

8. Závěr

Cílem práce bylo navrhnout a realizovat modul pro lokalizaci vozidla pomocí GPS s následným přenosem získané informace prostřednictvím GSM sítě, dále pak serverovou aplikaci pro zpracování přijatých dat z modulu a její zobrazení v programu Google Earth. Proto je tato práce rozdělena do dvou částí.

V první části práce jsou popsány základní principy lokalizace pomocí GPS. Z velice obsáhlé literatury bylo nutné vybrat jen ty nejdůležitější kapitoly důležité pro pochopení problematiky GPS lokalizace a formátu NMEA vět, ve kterých jsou uloženy informace o poloze, rychlosti, datu a času získání souřadnice. Tato část práce dále popisuje fungování GSM sítě, základní principy přenosu dat v této síti. Následuje také analýza stávající řešení této problematiky a porovnání výhod a nevýhod vlastního řešení.

Druhá část práce popisuje konkrétní návrh elektronického propojení jednotlivých modulů, GPS, GSM a řídící desky. Dále pak komunikaci s jednotlivými moduly, ovládací program mikroprocesoru realizovaný v programovacím jazyce Wiring, serverovou aplikaci pro zpracování přijatých dat realizovanou v programovacím jazyce Java. Vzhledem k potřebě dostupnosti dat z více zařízení různých platforem najednou, je v této části práce také popsáno cloudové úložiště Dropbox využívané k ukládání výstupních dat a jejich propojení s programem Google Earth.

Aplikace byla několik měsíců testována autorem a po odladění dosahovala výsledky srovnatelné s komerčním řešením. Aplikaci by do budoucna bylo vhodné rozšířit například o zobrazení polohy přímo v internetovém prohlížeči pomocí pluginu Google Earth, dále pak o informace o délce logované trasy, rychlosti v jednotlivých úsecích trasy, aby se dala plnohodnotně využít jako komerčně nabízené elektronické knihy jízd.

9. Seznam použité literatury

[1]*Understanding GPS: principles and applications*. 2nd ed. Editor Elliott D Kaplan, Christopher J Hegarty. Boston: Artech House, c2006, xvii, 703 s. ISBN 15-805-3894-0.

[2]RAPANT, Petr. *Družicové polohové systémy*. Vyd. 1. Ostrava: Vysoká škola báňská - Technická univerzita, 2002, 197 s. ISBN 80-248-0124-8. Dostupné z: http://gis.vsb.cz/dokumenty/dns-gps/view

[3]HOFMANN-WELLENHOF, Bernhard. *Global positioning system: theory and practice*. 5th, rev. ed. Wien: Springer, 2001, xxii, 382 s. ISBN 32-118-3534-2.

[4]*GSM, GPRS, and edge performance: evolution towards 3G/UMTS*. 2nd ed. Editor Timo Halonen, Javier Romero, Juan Melero. Chichester: John Wiley, 2003, xxxvii, 615 s. ISBN 04-708-6694-2.

[5]PETERKA, Jiří. Data v mobilních sítích. *Softwarové noviny* [online]. Praha: Softwarové noviny, 2000, č. 8, s. 10 [cit. 2014-06-02]. Dostupné z: http://www.earchiv.cz/a008s200/a008s200.php3

[6]Princip fungování GSM sítě. CELKOM PRAHA, spol. s r.o. *Zesilovače GSM signálu mobilních telefonů* [online]. 2009 [cit. 2014-06-02]. Dostupné z: http://www.zesilovac-signalu.cz/cs/princip-fungovani-gsm-site/

[7]Základní lokalizační metody v GSM. ORLICH, M. *Základní lokalizační metody v GSM* [online]. 2006 [cit. 2014-11-17]. Dostupné z: http://access.fel.cvut.cz/view.php?cisloclanku=2006022801

[8]PECHAL, Stanislav. *Monolitické mikropočítače*. 2., aktualiz. vyd. Praha: BEN - technická literatura, 1998, 269 s. ISBN 80-860-5630-9.

[9]MAJER, Martin. Seriál Síťování v Javě. *Seriál Síťování v Javě - Root.cz* [online]. 2006 [cit. 2015-02-17]. Dostupné z: http://www.root.cz/serialy/sitovani-v-jave/

10. Zdroje online obrázků

[10]AUTOR NEUVEDEN. *Wikipedia.org* [online]. [cit. 4.6.2014]. Dostupný na WWW: http://cs.wikipedia.org/w/index.php?title=R%C3%A1diov%C3%A9_sign%C3%A1ly_GPS&oldid=9938797#mediaviewer/Soubor:GPS_signal_modulation_scheme2.svg

[11]AUTOR NEUVEDEN. *Wikipedia.org* [online]. [cit. 4.6.2014]. Dostupný na WWW: http://cs.wikipedia.org/wiki/R%C3%A1diov%C3%A9_sign%C3%A1ly_GPS#mediaviewer/Soubor:Phase_modulation_BPSK_GPS.svg

[12]AUTOR NEUVEDEN. *http://www.zesilovac-signalu.cz/* [online]. [cit. 4.6.2014]. Dostupný na WWW: http://www.zesilovac-signalu.cz/cs/princip-fungovani-gsm-site/

11. Přílohy

11.1. Příloha A – výpis programu mikroprocesoru

```
/*
  GPS lokátor s přenosem dat po GSM pomocí UDP
 */

int count = 5;
int PwrOn = 22;
int LED = 13;
int conCount = 0;
int l;
String rdData = "";
String data =  "";
boolean conect = false;
boolean cntData = false;
boolean err = false;


void setup() {
  // initialize both serial ports:
  Serial1.begin(9600);
  Serial2.begin(9600);
  Serial3.begin(4800);
  pinMode(PwrOn, OUTPUT);
  pinMode(LED, OUTPUT);
  digitalWrite(PwrOn, LOW);
  digitalWrite(LED, LOW);
  String data = "";
  Pwr(1);
  ChckConect();
  if (conect == false){Restart();}
  SetConection();
  digitalWrite(LED, HIGH);
  SendStart();


}

void loop() {
  data = "";
  if (Serial3.available()) {
    char inByte = Serial3.read();

      if (inByte == '$') {
        data += inByte;
          while (inByte != 10){
            if (Serial3.available() > 0) {
                inByte = Serial3.read();
                data += inByte;
            }

            }
          }
```

```
if (data.startsWith("$GPRMC")){
    Serial1.println(data);
          if (data.substring(18,19) == "A"){
                count ++;
                if (count == 15){count = 0;
                SendData();
                SendChck();
            }
      }

      }
    }
}

void SendChck(){
    String conOK = rdData.substring(1-4,1-2);
    if (conOK.equals("OK") == false){
      ChckConect();
        if(conect == false){
          Restart();
          SetConection();
        }
    }
}

void SendData(){
   Serial2.println("AT+QISEND");
         Read();
         Serial2.print(data);
         Read();
         Serial2.print('\032');
         Read();
}

void SetConection(){
  cntData = false;
  while(cntData == false){
  Serial2.println("AT+QIFGCNT");
  Read();
  Serial2.println("AT+QICSPG=1,\"internet.t-mobile.cz\"");
  Read();
  Serial2.println("AT+QIMUX=0");
  Read();
  Serial2.println("AT+QIMODE=0");
  Read();
  Serial2.println("AT+QIREGAPP");
  Read();
  Serial2.println("AT+QIACT");
  Read();
  Serial2.println("AT+QIOPEN=\"UDP\",\"IP adresa serveru\",17");
  Read();
  ChckConectData();
  }

}
```

```
void SendStart(){
  Serial2.println("AT+QISEND");
  Read();
  Serial2.print("Start");
  Serial2.print('\032');
  Read();
}

void ChckConectData(){
  cntData = false;
  Serial1.print(rdData.substring(l-4,l-2));
  String conOK = rdData.substring(l-4,l-2);
  if(conOK.equals("OK") == true){cntData = true;}
}

void Read(){
  rdData = "";
  char inByte;
  while (Serial2.available() <= 0){}
  while (Serial2.available() > 0) {
    inByte = Serial2.read();
    rdData += inByte;
    delay(150);
    }
 l = rdData.length();
 Serial1.println(rdData);
 }

void Restart(){
  while (conect == false){
    Pwr(0);
    Read();
    Pwr(1);
    ChckConect();
  }
}

void Pwr(int onoff){
  if (onoff == 1) Serial1.println("PowerOn");
  if (onoff == 0) Serial1.println("PowerOff");
  String PwrData = "";
  digitalWrite(PwrOn, HIGH);
  delay(2500);
  digitalWrite(PwrOn, LOW);
  delay(15000);

 }
```

```
void ChckConect(){
  conCount = 0;
  conect = false;
  while(conect == false){
  Serial2.println("AT+CGREG?");
    Read();
    Serial1.println(rdData);
    Serial1.println("--------");
    int l = rdData.length();
    Serial1.print(rdData.substring(l-9,l-8));
    String logOK = rdData.substring(l-9,l-8);
    if(logOK.equals("1") == true){conect = true;}
    if(logOK.equals("5") == true){conect = true;}
    if(conect == false){delay(4000);}
    conCount++;
    if (conCount > 3){break;}
  }
}
```

11.2. Příloha B – výpis programu JAVA aplikace

11.2.1.GUI

```
package cz.uhk.pro25.file;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.io.File;
import java.io.FilenameFilter;
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;

public class locatorGui extends JFrame {
    private JPanel panTop = new JPanel();
    private JPanel panBottom = new JPanel();
    private JPanel panLeft = new JPanel();
    private JPanel panRight = new JPanel();
    private JPanel panCenter = new JPanel();
    private JLabel lbNmeaFile = new JLabel("Cesta:");
    private JTextField tfNmeaFile = new JTextField(25);
    private JLabel lbFtpCon = new JLabel("Název(adresa) FTP severu:");
    private JTextField tfFtpCon = new JTextField(20);
    private JLabel lbFtpLogin = new JLabel("FTP login:");
    private JTextField tfFtpLogin = new JTextField(20);
    private JLabel lbFtpPswrd = new JLabel("FTP heslo:");
    private JTextField tfFtpPswrd = new JTextField(20);
    private JCheckBox chbFtpOn = new JCheckBox("Ukládání na FTP");
    private JButton btnStart = new JButton("Start");
    private JButton btnBrowse = new JButton("Browse");
    private boolean ftpOn = false;
    private FileCopy fCopy = new FileCopy();
```

```
Action
            actBrowseFile,
            actRunServer;

public locatorGui(){
      super("GPS locator server");
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setSize(600, 300);
      createActions();
      initGui();
      add(panTop,BorderLayout.NORTH);
      add(panBottom,BorderLayout.SOUTH);
      add(panRight,BorderLayout.EAST);
      add(panLeft,BorderLayout.WEST);
      add(panCenter,BorderLayout.CENTER);

      setVisible(true);
}



public void initGui(){
      panLeft.setLayout(new GridLayout(8,1));
      panCenter.setLayout(new GridLayout(8,1));
      panLeft.setBorder(BorderFactory.createTitledBorder("Umístění souboru
      NMEA.kml:"));
      panCenter.setBorder(BorderFactory.createTitledBorder("FTP server"));
      panTop.setBorder(BorderFactory.createTitledBorder("FIM UHK 2015 © Jan
      Dian"));
      panLeft.add(lbNmeaFile);
      panLeft.add(tfNmeaFile);
      panCenter.add(lbFtpCon);
      panCenter.add(tfFtpCon);
      panCenter.add(lbFtpLogin);
      panCenter.add(tfFtpLogin);
      panCenter.add(lbFtpPswrd);
      panCenter.add(tfFtpPswrd);
      panCenter.add(chbFtpOn);
      panBottom.add(btnStart);
      panLeft.add(btnBrowse);
      btnStart.addActionListener(actRunServer);
      btnBrowse.addActionListener(actBrowseFile);


      chbFtpOn.addItemListener(new ItemListener() {
         public void itemStateChanged(ItemEvent e) {
             if (e.getStateChange()==1) ftpOn = true;
          }
       });

}
```

```
public void browseFile(){
	JFileChooser fc = new JFileChooser();
	FileFilter filter = new FileNameExtensionFilter("KML file", "kml");
	fc.addChoosableFileFilter(filter);


	if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
	File f = fc.getCurrentDirectory();
	tfNmeaFile.setText(f.getAbsolutePath());

	}

}


public void createActions() {
	actRunServer = new AbstractAction("Start") {

		@Override
		public void actionPerformed(ActionEvent e) {

			String nmeaPath = tfNmeaFile.getText();
			String ftpCon = tfFtpCon.getText();
			String ftpLogin = tfFtpLogin.getText();
			String ftpPswrd = tfFtpPswrd.getText();


			fCopy.runServer(nmeaPath, ftpCon, ftpLogin, ftpPswrd,
			ftpOn);

		}
	};

	actBrowseFile = new AbstractAction("Browse") {

		@Override
		public void actionPerformed(ActionEvent e) {
			browseFile();

		}
	};
	}
}
```

11.2.2.Server

```
package cz.uhk.pro25.file;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.DatagramChannel;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.apache.commons.net.ftp.FTPClient;

public class FileCopy {
    private static String nmea;
    private static String nmea1;
    private static String nmea2;
    private static String nmea3;
    private static String gps;
    private static String lastgps = "";
    private static String nmea_klm;
    private static String info;
    private static int first;
    private static int last;
    private static int firstn;
    private static int lastn;
    private static int lenght;
    private static DatagramChannel udpChannel;
    private static ByteBuffer buffer;
    private static boolean wrong = false;
    private static boolean start = false;
    private static int datx = 0;
    private static Date now;
    private static String date;
    private static String dateForFile;
    private static String rinpName;
    private static String files = "";
    private static String ftpHost = "";
    private static String ftpLog = "";
    private static String ftpPsw = "";
    private static String nmeaPth;
    private static boolean ftpOnOff;
```

```
/**
 * @param ta
 * @param args
 */
public FileCopy(){
}

public void  runServer(String nmeaPath, String ftpCon, String ftpLogin,
String ftpPswrd, boolean ftpOn){

	nmeaPth = nmeaPath;
	ftpHost = ftpCon;
	ftpLog = ftpLogin;
	ftpPsw = ftpPswrd;
	ftpOnOff = ftpOn;
	System.out.println("Server spuštěn");



	wrong = false;
	nmeaPth = nmeaPth + "/";
	String inpName = nmeaPth + "nmea.kml";
	String outpName = nmeaPth + "nmea.kml";
	while (true){
		udpComn();
		gpsConv();
		now = new Date();
		SimpleDateFormat DATE_FORMAT1 = new SimpleDateFormat("yy-MM-dd-
		HH-mm");
		dateForFile = DATE_FORMAT1.format(now);
		String cesta = nmeaPth;
		String file = cesta + dateForFile + ".kml";

		if (start == true){rinpName = nmeaPth + "road.kml";
			; files = file;}
		else {file = files; rinpName = file;}
		String routpName = file;
		nmeaRewrite(inpName,outpName);
		roadRewrite(rinpName,routpName);
		System.out.println(ftpOnOff);
		if (ftpOnOff){sendFtp();}
		System.out.println("Hotovo!");
	}
}
```

```
private static void nmeaRewrite(String inpName, String outpName) {
	if(wrong){return;}
	FileOutputStream fOut = null;
	FileInputStream fIn = null;
	try {
		fIn = new FileInputStream(inpName);
		nmea = "";
		int b;
		while ((b=fIn.read())!=-1) {
			char znak = (char)b;
			nmea = nmea + znak;

		}

		first = nmea.lastIndexOf("<coordinates>") + 12;
		last = nmea.lastIndexOf("</coordinates>");
		firstn = nmea.lastIndexOf("<name>") + 5;
		lastn = nmea.lastIndexOf("</name>");
		lenght = nmea.length();
		nmea1 = "";
		nmea2 = "";
		nmea3 = "";
		for (int z=0; z<=firstn;z++){
			nmea1 = nmea1 + nmea.charAt(z);
		}
		for (int z=lastn; z<=first;z++){
			nmea2 = nmea2 + nmea.charAt(z);
		}
		for (int z=last; z<lenght;z++){
			nmea3 = nmea3 + nmea.charAt(z);
		}

		nmea_klm = nmea1 + info + nmea2 + gps + nmea3;
		fOut = new FileOutputStream(outpName);
		for (int z=0;z<nmea_klm.length();z++){
			Integer zn = Integer.valueOf(nmea_klm.charAt(z));
			fOut.write(zn);

		}
	} catch (FileNotFoundException e) {
		e.printStackTrace();
	} catch (IOException e) {
		e.printStackTrace();
	} finally {
		//zavirame, pokud to jde
		try {
			if (fIn!=null) fIn.close();
			if (fOut!=null) fOut.close();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}


}
```

```
private static void roadRewrite(String rinpName, String routpName) {
	if(wrong){if(!ftpOnOff)wrong = false;
	System.out.println("Neplatná data!!!!!");return;}

	System.out.println("Zapisuju!!!");
	FileOutputStream fOut = null;
	FileInputStream fIn = null;
	try {

		fIn = new FileInputStream(rinpName);
		nmea = "";
		int b;
		while ((b=fIn.read())!=-1) {
			char znak = (char)b;
			nmea = nmea + znak;

		}

		first = nmea.indexOf("<coordinates>") + 12;
		last = nmea.indexOf("</coordinates>");
		lenght = nmea.length();
		nmea1 = "";
		nmea2 = "";
		for (int z=0; z<last;z++){
			nmea1 = nmea1 + nmea.charAt(z);
		}

		for (int z=last; z<lenght;z++){
			nmea2 = nmea2 + nmea.charAt(z);
		}

		String nmeaStart = "</coordinates>"+ "\r\n"
		+"</LineString></Placemark>" + "\r\n" +"<Placemark>"+ "\r\n"
		+"<name>Start</name><Point><coordinates>"+ "\r\n";

		String nmeaStart1 = "\r\n" + "</coordinates>" + "\r\n" +
		"</Point></Placemark>"+ "\r\n" + "</Document></kml>";

		fOut = new FileOutputStream(routpName);
			if (start){start = false; nmea_klm = nmea1 + lastgps +
			"\r\n" + gps + "\r\n" + nmeaStart + lastgps +
			nmeaStart1;}

			else {nmea_klm = nmea1 + gps + "\r\n" + nmea2;}
			for (int z=0;z<nmea_klm.length();z++){
			Integer zn = Integer.valueOf(nmea_klm.charAt(z));
			fOut.write(zn);
		}
		lastgps = gps;

	} catch (FileNotFoundException e) {
		e.printStackTrace();
	} catch (IOException e) {
		e.printStackTrace();
	}
```

```
        finally {
                //zavirame, pokud to jde
                try {
                        if (fIn!=null) fIn.close();
                        if (fOut!=null) fOut.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }


}

public static void udpComn(){
                gps = "";
                now = new Date();
                SimpleDateFormat DATE_FORMAT = new SimpleDateFormat("dd.MM.yy
                HH:mm");

                date = DATE_FORMAT.format(now);

         try {
              udpChannel = DatagramChannel.open(); //otevřít UDP kanál

                udpChannel.socket().bind(new InetSocketAddress("0.0.0.0", 17));
                //začít naslouchat

                buffer = ByteBuffer.allocateDirect(512); //alokovat buffer
                udpChannel.receive(buffer);
                buffer.flip();
                while (buffer.hasRemaining( )) {

                int i = buffer.get( );
                char znk = (char) i;
                gps = gps + znk;

                }
                    buffer.clear(); //vymazat obsah bufferu
          }
          catch(IOException e) {
              e.printStackTrace(System.err);
          }
          finally {
              try {
                  if(udpChannel != null) udpChannel.close();
              }
              catch(IOException e) {}
          }

         System.out.println(gps);
         if (gps.equals("Start") == true){wrong = true; start = true; }
         if (gps.length() < 66){wrong = true; return;}
         if (gps.substring(0, 6).equals("$GPRMC")== false){
               wrong = true; return;}
         int l = gps.length();
         gps = gps.substring(7, l-1);
         System.out.println(gps);

 }
```

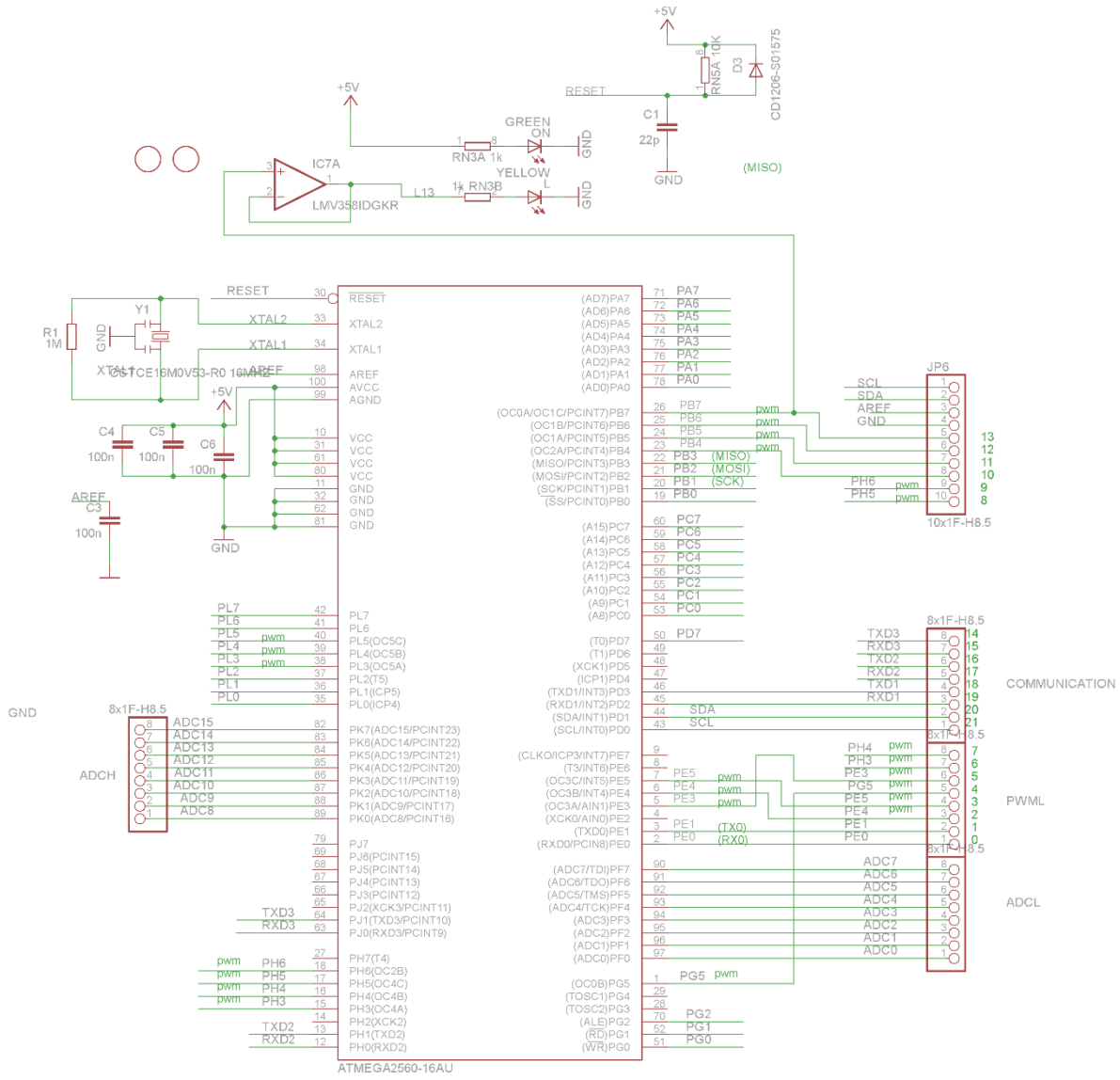
```
public static void gpsConv (){
	if(wrong){return;}
	String  stn = "";
	String  ste = "";
	String  minn = "";
	String  mine = "";
	String  speed = "";
	String  tcas = "";
	String  sj = "";
	String  vz = "";
	int carka = 0;
	int i = 0;
	int j = 0;
	int k = 0;
	int d = 0;
	int odd = (int) ',';
	String cas = "";
	String datum = "";
	String tdatum = "";
	while (i < gps.length()) {
		char temp = gps.charAt(i);
		//--------------------počítání oddělovače "," -------------
		if ((int) temp == odd) {
			carka++;
			i++;
			temp = gps.charAt(i);
		}
		//-----------------------čas-----------------------
		if (carka == 0) {
			if (d<4) {
				cas = cas + gps.charAt(i);
				d++;
			}
		}
		//---------------------souřadnice----------------------------
		if (carka == 2) {
			if (j < 2) {
				stn = stn + gps.charAt(i);
				j++;
			}
			if (j == 2) {
				j++;
				i++;
			}

			if (j > 2) {
				minn = minn + gps.charAt(i);
				j++;
			}
		}
```

```
		if (carka == 3) sj = sj + gps.charAt(i); //sever-jih

		if (carka == 4) {
			if (k < 3) {
				ste = ste + gps.charAt(i);
				k++;
			}
			if (k == 3) {
				k++;
				i++;
			}
			if (k > 3) {
				mine = mine + gps.charAt(i);
				k++;
			}
		}
		if (carka == 5) vz = vz + gps.charAt(i); //východ - západ

		//------------------------------------------------------------

		if (carka == 6) speed = speed + gps.charAt(i); //rychlost
		if (carka == 8)    datum = datum + gps.charAt(i); //datum
		i++;
	}


	double gps1 = Double.valueOf(minn).doubleValue();
	gps1 = gps1 * 0.01666; // přepočet minut na desetiny stupně
	double gps2 = Double.valueOf(mine).doubleValue();
	gps2 = gps2 * 0.01666; // přepočet minut na desetiny stupně
	gps1 = Double.valueOf(stn) + gps1;
	gps2 = Double.valueOf(ste) + gps2;
	minn = String.valueOf(gps1);
	mine = String.valueOf(gps2);
	double speed1 = Double.valueOf(speed);
speed1 = speed1 * 1.852; // přepočet rychlosti z uzlů na kilometry
if (vz.equalsIgnoreCase("W")) mine = "-" + mine;
if (sj.equalsIgnoreCase("S")) minn = "-" + minn;
	speed = String.valueOf(Integer.valueOf((int)speed1)) + " km/h";
	gps = mine + "," + minn + ",0";
	if (lastgps.equals("")){lastgps = gps;
	System.out.println("Poslední pozice byla prázdná!!!!");}

	for (i=0;i < cas.length();i++) { //doplnění oddělovačů do data a času

		if (i==2) tcas = tcas + ':';
		tcas = tcas + cas.charAt(i);
			if (i==1){int hod = Integer.parseInt(tcas); hod=hod+2;
			if (hod >= 24){hod = hod - 24; datx = 1;}
			String hod1 = Integer.toString(hod);tcas = hod1;

			}
	}
```

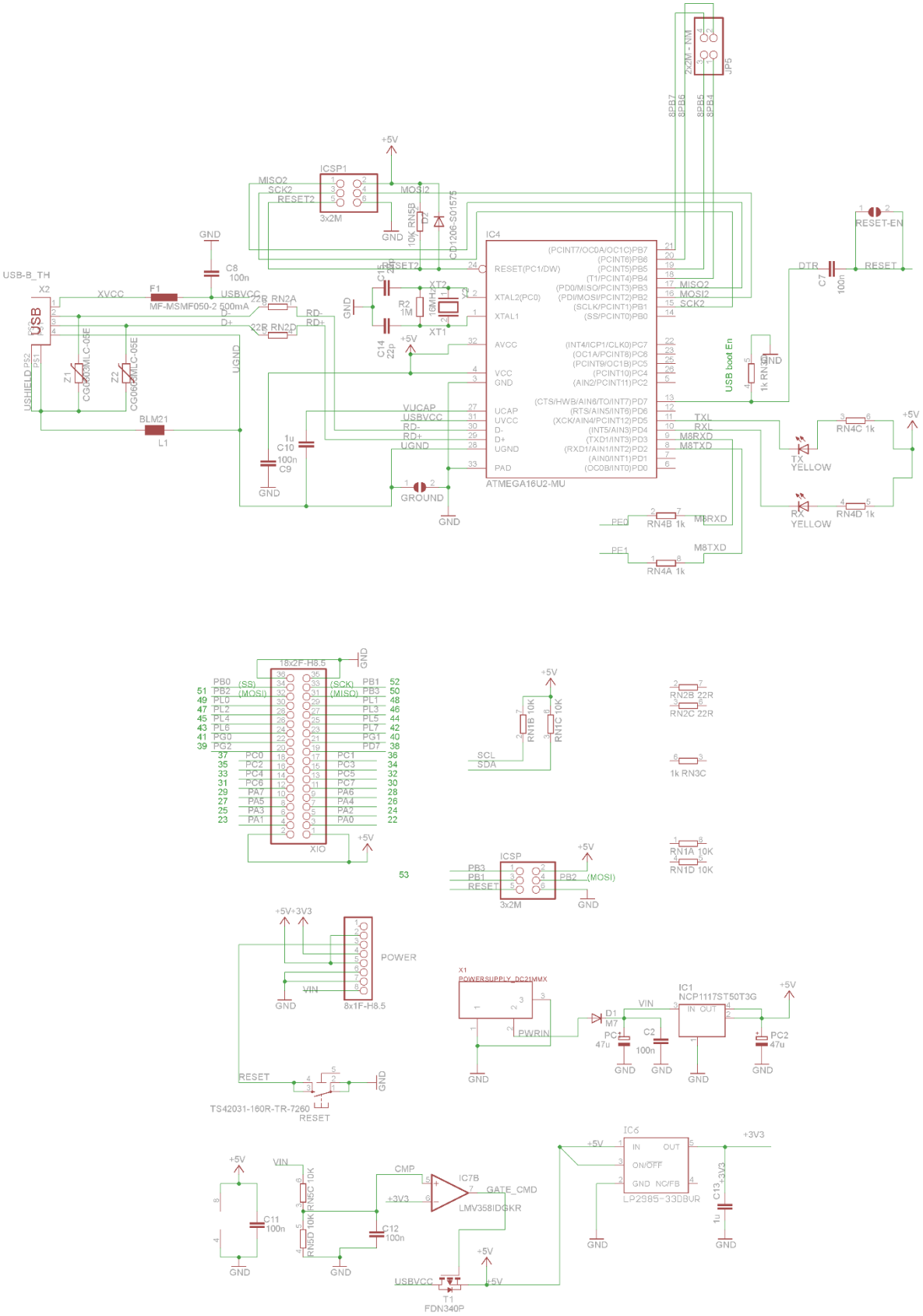
```
            for (i=0;i < datum.length();i++) {
                  if (i==2) {int dat = Integer.parseInt(tdatum);dat = dat + datx;
                        tdatum = Integer.toString(dat); tdatum = tdatum + '.';}
                  if (i==4) {tdatum = tdatum + '.';}
                  tdatum = tdatum + datum.charAt(i);
            }
            datx = 0;
            cas = tcas;
            datum = tdatum;
            info = date + " / " + speed;
      }

      public static void sendFtp() {//nakopírování souboru na web pomocí FTP
            if(wrong){wrong = false; System.out.println("Neplatná data!!!!!");
            return;}

            if(start){start = false; return;}
            System.out.println("Provádím zápis na FTP");
            wrong = false;
            FTPClient client = new FTPClient();
            FileInputStream fis = null;


          try {
              client.connect(ftpHost);
              System.out.println(ftpHost);
              client.login(ftpLog, ftpPsw);
              System.out.println(ftpLog + ", " + ftpPsw);
              String filename = nmeaPth + "nmea.kml";
              fis = new FileInputStream(filename);
              System.out.println(filename);
              client.storeFile("nmea.kml", fis);
              fis.close();
              client.logout();
          }
             catch (IOException e) {
              e.printStackTrace();
          }
      }

}
```

11.3. Příloha C – elektrotechnická schémata

11.3.1. Arduino Mega2560

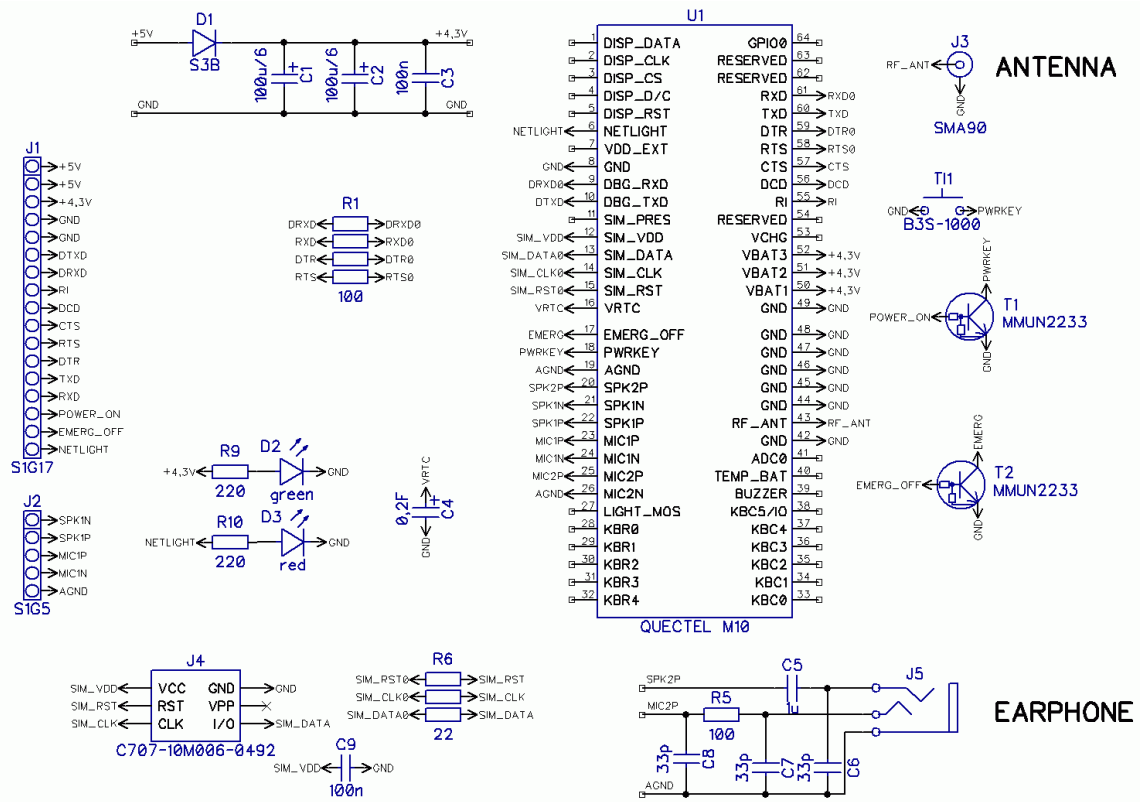


Obr. 28 Arduino Mega 2560 - schéma zapojení [zdroj: www.arduino.cc]



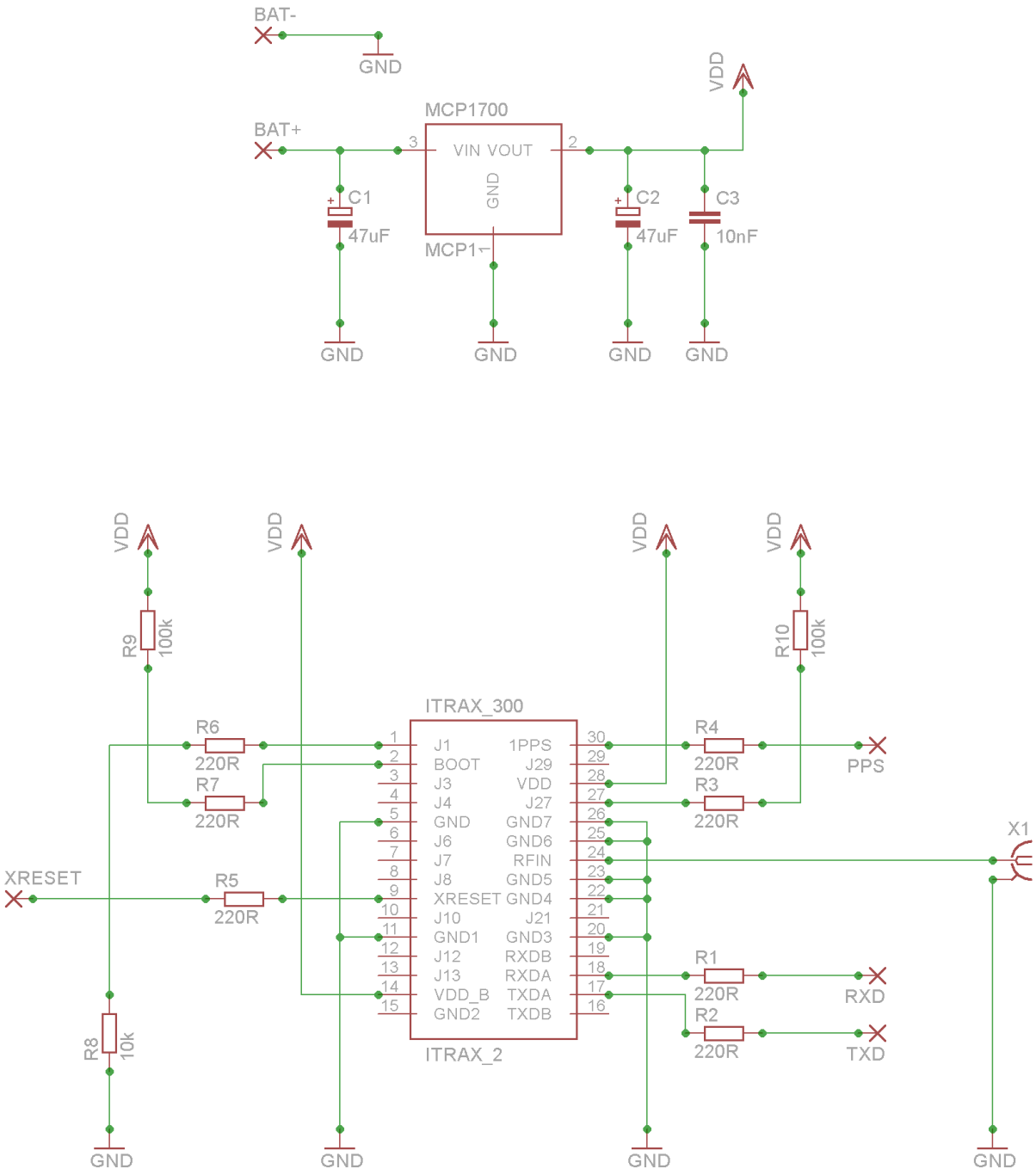
Obr. 29 Arduino Mega 2560 - schéma zapojení [zdroj: www.arduino.cc]

11.3.2.GSM modul PGSM s Quectel M10



Obr. 30 PGSM modul - schéma zapojení [zdroj: www.pandatron.cz]

11.3.3.GPS modul s Fastrax IT300



Obr. 31 GPS přijímač s modulem Itrax 300 - schéma zapojení [zdroj: autor]