ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická
Katedra elektroenergetiky

DIPLOMOVÁ PRÁCE

Adaptivní model chování bateriového systému pro elektromobilitu

Autor práce: **Bc. Ondřej Naxer**
Vedoucí práce: **Ing. Lenka Šroubová, Ph.D.**

2022

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Ondřej NAXER**
Osobní číslo:	**E20N0032P**
Studijní program:	**N0713A060013 Výkonové systémy a elektroenergetika**
Specializace:	**Elektroenergetika**
Téma práce:	**Adaptivní model chování bateriového systému pro elektromobilitu**
Zadávající katedra:	**Katedra elektroenergetiky**

Zásady pro vypracování

1. Prostudujete aktuálně používané technologie baterií v dopravě.
2. Vyhodnoťte data získaná na základě technického listu a statických měření baterie.
3. Vytvořte adaptivní model předpovídající chování baterie v závislosti na profilu jízdy.
4. Vytvořte zařízení, které bude zaznamenávat parametry jízdy.
5. Ověřte navržený model při testovací jízdě.

Rozsah diplomové práce: **40 – 60 stran**
Rozsah grafických prací: **podle doporučení vedoucího**
Forma zpracování diplomové práce: **elektronická**

Seznam doporučené literatury:

1. BERG, Helena. Batteries for electric vehicles: materials and electrochemistry. Cambridge: Cambridge University Press, 2015. ISBN 978-1107085930.
2. Směrnice evropského parlamentu a rady (EU) 2019/1161. Evropský parlament; Rada Evropské unie. In: Úř. věst. L 188, 12.7.2019, s. 116-130.
3. HOSTAŠA, Michal. Testování baterií elektromobilu. Ostrava, 2012. Bakalářská práce. VŠB Technická univerzita Ostrava. Vedoucí práce Ing. Jiří Kulhánek, Ph.D..

Vedoucí diplomové práce: **Ing. Lenka Šroubová, Ph.D.**
Katedra elektrotechniky a počítačového modelování

Datum zadání diplomové práce: **8. října 2021**
Termín odevzdání diplomové práce: **26. května 2022**

Západočeská univerzita v Plzni
Fakulta elektrotechnická

L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Doc. Ing. Karel Noháč, Ph.D.
vedoucí katedry

V Plzni dne 8. října 2021

Abstrakt

Předmětem této diplomové práce je sestavení matematického modelu pro typický bateriový systém pro elektromobilitu, využívající reálné provozní veličiny, zhodnocující aktuální stav baterie, profil jízdy a okolní podmínky prostředí. Výstupem matematického modelu je optimalizovaný kvantitativní údaj predikující dostupný dojezd vozidla. Pro získání a zpracování potřebných podkladových dat je využita vývojová deska Raspberry Pi Pico, samotný model bateriového systému je pak realizován programově v jazyce Python.

Klíčová slova

Elektromobilita, bateriový systém, matematický model, telemetrie, predikce spotřeby, stanovení dojezdu.

Abstract

The aim of this diploma thesis is to set up the mathematical model for a typical battery system used in electromobility, in accordance with the real operating values, actual battery state, driver's profile, and ambient conditions. The output of this mathematical model is the optimized comparative value predicting the remaining vehicle's range. Raspberry Pi Pico microcontroller board is used to obtain and process the necessary operational data. The mathematical model is then implemented to Python language.

Key Words

Electromobility, battery system, mathematical model, telemetry, consumption prediction, optimized range data.

Poděkování

Rád bych v prvé řadě poděkoval své rodině za láskyplnou a finanční podporu během náročných vysokoškolských studií. Dědovi za nasměrování na správný a perspektivní obor. Dále vedoucí diplomové práce Ing. Lence Šroubové, Ph.D. za svěření aktuálního a praktického tématu, poskytnutí cenných odborných rad, tematické připomínky a ochotný přístup při řešení technických problémů. Významný podíl na realizaci celé této práce přísluší konzultantovi Ing. Miroslavu Blohmannovi, který laskavě poskytl elektromobil k testovací jízdě a dále potřebné související technické specifikace a jiné poznatky.

Obsah

Úvod ... - 1 -
1 Bateriové systémy pro elektromobilitu ... - 2 -
1.1 Základní principy bateriových systémů ... - 2 -
1.2 Olověné akumulátory ... - 3 -
1.2.1 Princip olověných akumulátorů ... - 3 -
1.2.2 Klasické olověné akumulátory se zaplavenými elektrodami ... - 4 -
1.2.3 Bezúdržbové olověné akumulátory ... - 6 -
1.2.4 Elektrické parametry a charakteristiky ... - 7 -
1.3 Lithiové akumulátory ... - 14 -
1.3.1 Lithium-iontové akumulátory ... - 16 -
1.3.2 Lithium-polymerové akumulátory ... - 17 -
1.3.3 Akumulátory na bázi lithium-železo-fosfát ... - 18 -
1.3.4 Význačné parametry lithiových akumulátorů ... - 19 -
1.3.5 Charakteristiky lithiových akumulátorů ... - 21 -
1.4 Porovnání akumulátorů ... - 24 -
2 Telemetrické zařízení ... - 26 -
2.1 Bezdrátové měřící zařízení ... - 26 -
2.1.1 Popis zařízení a jeho funkce ... - 29 -
2.1.2 Zapojení jednotlivých modulů ... - 39 -
2.1.3 Řešení programových problémů ... - 40 -
2.2 Geolokační modul ... - 42 -
2.3 Datasběrný software ... - 44 -
3 Subjekt měření ... - 49 -
4 Data z testovacích jízd ... - 52 -
4.1 Testovací jízda #1 ... - 52 -
4.2 Testovací jízda #2 ... - 54 -
4.3 Testovací jízda #3 ... - 57 -
4.4 Testovací jízda #4 ... - 60 -
4.5 Evaluace dat ... - 65 -

4.6 Zhodnocení měření ..... - 66 -
5 Statické měření bateriového systému ..... - 68 -
5.1 Měřící sestava ..... - 68 -
5.2 Zhodnocení statického měření ..... - 73 -
6 Model bateriového systému ..... - 74 -
6.1 Technické parametry použitých článků ..... - 74 -
6.2 Popis chování bateriového systému ..... - 78 -
6.3 Modelové případy ..... - 79 -
6.3.1 Prostá integrace výkonu ..... - 79 -
6.3.2 Sledování napěťové křivky ..... - 79 -
6.3.3 Dopředný model ..... - 80 -
7 Ověření funkce matematického modelu ..... - 81 -
Závěr ..... - 87 -
Literatura ..... - 89 -
Přílohy I

Seznam symbolů a zkratek

Značka	Popisek	Jednotka
C	Kapacita akumulátoru	(A.h), (W.h)
C_{10}	10hodinová kapacita	(A.h)
C_{20}	20hodinová kapacita	(A.h)
c	Měrná tepelná kapacita	($J.kg^{-1}.K^{-1}$)
I	Elektrický proud	(A)
m	Hmotnost	(kg)
p_U	Napěťový převod	(-)
p_I	Proudový převod	(-)
Q	Jouleovo teplo	(J = W.s)
R_V	Vnitřní odpor akumulátoru	(Ω)
T	Termodynamická teplota	(K)
t	Teplota	(°C)

AGM	Absorbent Glass Mat
BMS	Battery Management System
LCO	Lithium Cobalt Oxide
LFP	Lithium Iron Phosphate Oxide
LMO	Lithium Manganese Oxide
MF	Maintenance-Free
NCA	Lithium Nickel Cobalt Aluminium Oxide
NiCd	Nickel-cadmium battery
NiMH	Nickel-metal hydride battery
NMC, NCM	Lithium Nickel Cobalt Manganese Oxide
UPS	Uninterruptible power supply
SLI	Starting, lighting and ignition battery type
VRLA	Valve Regulated Lead Acid Battery
SoC	State of Charge
SoH	State of Health
DoD	Deep of Discharge

PE	Polyethylene
PP	Polypropylene
PPCP	Polypropylene copolymer

AD, A/D	Analog-to-Digital converter
BT	Bluetooth
FLASH	Electronic non-volatile memory
GND	Ground
GPIO	General-Purpose Input/Output
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
OS	Operating System
PC	Personal Computer
RAM	Random Access Memory
ROM	Read-Only Memory
RTC	Real-Time Clock
UART	Universal Asynchronous Receiver-Transmitter

USB	Universal Serial Bus
UTC	Universal Time Coordinated
GPS	Global Positioning System
IMU	Inertial Measurement Unit
NMEA	National Marine Electronics Association
PTC	Thermistor with positive temperature coefficient

Úvod

Elektromobilitu provází neustálý vývoj, jehož hlavním cílem je minimalizovat mnohé limity, potažmo rozdíly, v nasazení elektromobilů oproti konvenčním vozidlům na spalovací pohon. Jedním z kritérií je například maximalizace a dostatečně přesná predikce dojezdu vozidla na elektrický pohon. Ve srovnání s chemickou energií obsaženou v 0.5 kg benzínu, je v případě olověných akumulátorů zapotřebí cca 100násobek hmotnosti v podobě elektrolytu [1]. Hmotnostně, při stejném objemu akumulované energie, pak lépe vycházejí lithiové akumulátory. Váha bateriového systému přitom může negativně ovlivnit jízdní vlastnosti vozidla, především jeho dojezd. Dále je patrný zásadní rozdíl ve formě doplnění energií. Zatímco doba doplnění energie v podobě konvenčních paliv se pohybuje nanejvýš v řádu několika minut, a při níž lze získat maximální dojezd, dobíjení akumulátorů je omezené hned několika faktory – lokální dostupností dobíjecích míst, jejich typem, poměry v nadřazené síti a samozřejmě časem vytyčeným procesu nabíjení, který se pohybuje řádově ve vyšších desítkách minut až hodin. Mezi netechnické ukazatele potom řadíme například pořizovací cenu baterií nebo náklady na dodanou 1 kWh elektrické energie. Je tedy dostatečně zřejmé, že naprosto klíčový bude další vývoj ve zvyšování hustoty akumulované energie, použití produktů recyklace při výrobě nových baterií, a současně nejrůznější optimalizace a predikce v energetické spotřebě vozidla pro jeho efektivní obsluhu a provoz.

V následujících kapitolách budou po teoretické stránce přiblíženy obecné principy bateriových článků, potažmo celých systémů, využívaných pro pohon osobních elektromobilů.

V praktické části budou následně představeny možnosti měření provozních veličin během testovací jízdy, a to prostřednictvím vývojové platformy Raspberry Pi Pico. S využitím běžně dostupných modulů pro měření a bezdrátovou komunikaci bude představeno telemetrické zařízení, a to včetně podpůrného softwaru pro zpracování a vizualizaci výsledků. Dále bude sledováno chování konkrétního typu bateriového systému v laboratorním kontrolovaném prostředí (sestavení měřícího obvodu) pro stanovení technického stavu baterie a případné srovnání s technickými listy výrobce. Cílem bude využití získaných dat z dílčích měření, a aplikace teoretických poznatků, pro sestavení adaptivního modelu daného typu bateriového systému. Výstupem této práce bude algoritmus aplikovatelný do řídicího systému elektromobilu podávající dostatečně přesnou informaci o dostupném dojezdu na baterii.

1 Bateriové systémy pro elektromobilitu

V dnešních elektromobilech je možné se setkat se základními dvěma druhy akumulátorů, tj. olověnými a lithiovými. První olověné akumulátory se přitom objevily již v 19. století, krátce před vynalezením elektrických točivých generátorů a napájely první elektromobily. Od té doby tento typ akumulátorů prošel významným vývojem, a díky svým vlastnostem, především jejich spolehlivosti a cenové relaci, se používá v řadě aplikací, kde jen obtížně lze nalézt adekvátní náhradu. Oproti tomu lithiové baterie jsou poněkud mladší, jejich vznik lze datovat teprve ke konci 20. století. Pro určité aplikace, zejména spotřební elektroniku, se začaly jevit jako vhodnější, zejména z hlediska vlivu na rozměry, výslednou hmotnost zařízení a zlepšení doby nabíjení. Nezávisle na použitou technologii, platí v obou případech obecné principy akumulátorů.

1.1 Základní principy bateriových systémů

Základním funkčním prvkem každého bateriového systému je článek. Ten se obecně skládá z 5 hlavních komponent: elektrod (záporné anody a kladné katody), membrány (separátoru) mezi nimi, elektrolytu, svorek (kladné a záporné) a mechanického pouzdra. Jednotlivé články jsou pak ve výsledku vhodně propojeny pro dosažení žádaných elektrických parametrů. Elektrolyt může být ve formě kapaliny, gelu nebo jako pevný materiál. Tradiční akumulátory, například olověné nebo NiCd, využívají kapalného elektrolytu, který může být buď kyselý, nebo alkalický. V současné době se využívá spíše elektrolytů na bázi gelu, pasty nebo pryskyřice, jimiž typickými představiteli jsou akumulátory olověné (gelové), NiMH a lithium-iontové. Dále se lze setkat s lithium-polymerovými akumulátory na bázi pevného elektrolytu. [1] K základním požadavkům na trakční akumulátory obecně patří:

- vysoká energetická hustota pro maximální možný dojezd na jedno nabití,
- poskytování stabilního výkonu při všech možných provozních stavech,
- dlouhodobá životnost a stálost kapacity při nízkých i vysokých teplotách,
- bezúdržbovost (např. bez nutnosti doplňování elektrolytu),
- bezpečnost během samotného provozu i při procesu nabíjení,
- enviromentální ohledy (netoxicita, recyklovatelnost).

1.2 Olověné akumulátory

Olověné akumulátory představují velice rozšířenou technologii akumulátorů, ať už v automobilovém průmyslu, nebo obecně v dopravě, dále jako zdroje pro napájení a zálohování (UPS) citlivých telekomunikačních zařízení, nebo přímo silových zařízení. V závislosti na konkrétní aplikaci rozeznáváme následující druhy olověných baterií [2]:

- SLI (startovací, pro účely osvětlení a zapalovací) pro konvenční automobily,
- statické baterie používané jako záložní zdroje energie pro telekomunikační a výpočetní systémy, elektrárny,
- trakční a průmyslové baterie pro napájení dopravních prostředků na elektrický pohon, vysokozdvižných vozíků a důlních zařízení,
- speciální baterie pro využití v letectví, ponorkách a v oblasti vojenské techniky,
- akumulační baterie pro obnovitelné zdroje energie.

Dále z hlediska případné údržby rozlišujeme tzv. baterie se zaplavenými elektrodami (flooded batteries) s vysokým obsahem antimonu v mřížce olověné elektrody; pokročilejší technologií jsou pak „bezúdržbové“ baterie, jejichž kladné elektrody jsou tvořeny prvky PbCaSn nebo též malého množství Sb a u záporných elektrod je použito PbCa; v poslední řadě baterie typu VRLA s elektrodovými mřížkami PbSnCa a využívající AGM separátorů (Absorbed Glass Mat) [2].

1.2.1 Princip olověných akumulátorů

Článek olověné baterie se skládá z anody a katody, které jsou v podobě mřížek a ponořeny v elektrolytu. Činnou hmotu na kladné elektrodě tvoří kysličník olovičitý (PbO_2), na záporné elektrodě je to potom houbovité olovo. Mřížky jsou odlity ze slitiny olova legované různými přísadami pro dosažení vyšší mechanické pevnosti (odolnost vůči otřesům a vibracím), chemické odolnosti, elektrické vodivosti, a zlepšení vazby s činnou hmotou [3]. Nejčastějšími přísadami jsou antimon (Sb), vápník (Ca), cín (Sn) a selen (Se). Přidáním antimonu a cínu se dosahuje zlepšení chování trakčních baterií při cyklickém hlubokém vybíjení a nabíjení, na druhou stranu se však zvyšuje spotřeba vody a jsou vyžadována častější vyrovnávací nabíjení. Vápník zase omezuje proces samovybíjení, nicméně u kladných elektrod na bázi olova a vápníku se objevuje další vedlejší efekt, tj. růst rozměrů těchto elektrod vlivem oxidace olověné mřížky při přebíjení. Pro omezení předchozích následujících jevů využívají moderní olověné akumulátory další přídavné látky, např. selen, kadmium, cín či arzen [4].

Mřížky elektrod slouží jako nosiče činných hmot, společně pak tvoří jednotlivé desky. Desky příslušného typu elektrod se pravidelně střídají a tvoří deskové skupiny, navzájem jsou odděleny vložkami (separátory) a ponořeny v elektrolytu. Záporná desková skupina má obvykle o desku navíc [3]. Každá dvojice desek poskytuje v nezatíženém stavu napětí cca 2,1 V (nabitý 12 V akumulátor o šesti elektrodách má tedy napětí cca 12,6 V) [5].

Separátory jsou obvykle tvořeny porézní membránou ze skelných vláken napuštěné v elektrolytu. Základními požadavky na tyto oddělující prvky jsou: schopnost snadného průchodu iontů, odolnost vůči agresivnímu prostředí, mechanická opora soustavy desek a optimální dotyk s deskami, který umožní proudění elektrolytu a vyrovnávání jeho hustoty.

Elektrolyt akumulátoru obsahuje koncentrát kyseliny sírové (H_2SO_4) naředěný destilovanou vodou, přičemž v závislosti na typu podnebí je doporučována jeho určitá hustota (cca 1,26-1,285 g/cm^3 pro mírné podnebí [3]). Hustota elektrolytu ovlivňuje jeho vodivost, s vyšší hodnotou hustoty vodivost klesá, svorkové napětí a kapacita akumulátoru naopak rostou. Při dosažení horní meze hustoty elektrolytu však hrozí nebezpečí většího napadání desek kyselinou sírovou.

Akumulátor jako funkční celek je obvykle uzavřen v polymerní nádobě (PPCP). Jelikož během rychlého nabíjení (nebo vybíjení), dochází k vývinu plynů z rozkladu vody, je potřeba nádobu vybavit i bezpečnostními ventily. Samozřejmostí jsou i inspekční zátky sloužící ke kontrole a doplňování elektrolytu.

1.2.2 Klasické olověné akumulátory se zaplavenými elektrodami

Jedná se o klasické provedení, které bylo vysvětleno již v předcházející *kapitole 1.2.1*. Tyto akumulátory vyžadují pravidelnou údržbu, tzn. kontrolu a případné doplňování elektrolytu (destilované vody). Destilovaná voda se používá právě díky absenci iontů, které by mohly způsobit vnitřní zkrat na akumulátoru. Výhody a nevýhody těchto typů akumulátorů shrnuje následující tabulka:

Tabulka 1 Výhody a nevýhody olověných akumulátorů se zaplavenými elektrodami [5]

Výhody	Nevýhody
velmi malý vnitřní odpor	vysoká citlivost na hluboké vybití (max. výdrž cca 1-3 dny a současně podstatné snížení kapacity) pozn.: vybitý akumulátor může zamrznout (poškození elektrod)
vysoká energetická účinnost (až 85 %)	přísné rozmezí nabíjecího napětí (pro 6 článků rozsah cca 14,0-14,4 V s ohledem na okolní teplotu)
menší citlivost na přebíjení než u gelových a AGM akumulátorů	dlouhá doba nabíjení (max. nabíjecí proud cca 5-10 % z celk. kapacity) pozn.: při velkých nabíjecích i vybíjecích proudech dochází k sulfataci elektrod a snížení kapacity
dobré vlastnosti při nízkých i vysokých teplotách	životnost akumulátoru závislá na účinnosti desulfatace elektrod během procesu nabíjení
odolnost vůči proudovým výkyvům	teplota nabíjení nesmí překročit stanovenou mez (cca 40 °C)
snadná výroba	pravidelná kontrola hladiny elektrolytu a omezené transportní možnosti (obsah se může vylít)
vysoký stupeň recyklovatelnosti	kontrolní měření napětí je potřeba provádět alespoň 2 h po posledním nabíjení či provozu
přijatelný poměr výkon/cena	pomalé samovybíjení, avšak rychlejší než v případě bezúdržbových typů (cca 8-10 mV/den) pozn.: při poklesu napětí pod 12 V, tj. 2 V na článek, je akumulátor vybitý
	nešetrnost k životnímu prostředí

1.2.3 Bezúdržbové olověné akumulátory

Na rozdíl od klasického typu akumulátorů, bezúdržbový typ (MF) nemá elektrody „zaplavené“ elektrolytem, ale místo toho je elektrolyt napuštěn přímo do separátoru ze skelných vláken dotovaných bórem. Takový akumulátor následně označujeme jako AGM (akumulátor s vázaným elektrolytem). Dalším řešením bezúdržbového provedení je tzv. gelový akumulátor, kdy jsou elektrody usazeny do křemičitého gelu. V obou případech je nádoba baterie hermeticky uzavřena a neumožňuje žádný uživatelský zásah do vnitřních funkčních částí, stejně tak neexistuje riziko vylití elektrolytu při náhlé změně pracovní polohy. Pro regulaci tlaku uvnitř nádoby mohou být baterie vybaveny jednocestným přetlakovým ventilem, který je ochrání před natlakováním způsobeným přebíjením. Akumulátory s přetlakovým ventilem dodatečně označujeme jako VRLA (Valve Regulated Lead Accid). [5]

V následujících tabulkách jsou uvedeny výhody a nevýhody jednotlivých typů bezúdržbových typů olověných akumulátorů.

Tabulka 2 Výhody a nevýhody olověných gelových akumulátorů, převzato z [6]

Výhody	Nevýhody
bezúdržbový, lze instalovat šikmo, pomalé samovybíjení	vyšší výrobní náklady oproti typu AGM, ale levnější než klasický olověný akumulátor
dobrý odvod tepla do okolí prodlužuje životnost	citlivý na přebití
vysoký výkon až do konce životnosti, pak kapacita náhle a prudce klesá	měrná hustota energie a zátěžový proud nedosahují příliš vysokých hodnot
sloučením kyslíku a vodíku vyrábí vodu	dochází k uvolňování plynů (nutnost pojistných ventilů)
bezpečná obsluha, vysychá méně než typ AGM	nutnost skladování v nabitém stavu, avšak hůře si vedou akumulátory se zaplavenými elektrodami
vysoký počet cyklů, odolný vůči teplu	
široká nabídka velikostí akumulátorů	

Tabulka 3 Výhody a nevýhody olověných akumulátorů s vázaným elektrolytem (AGM), převzato z [6]

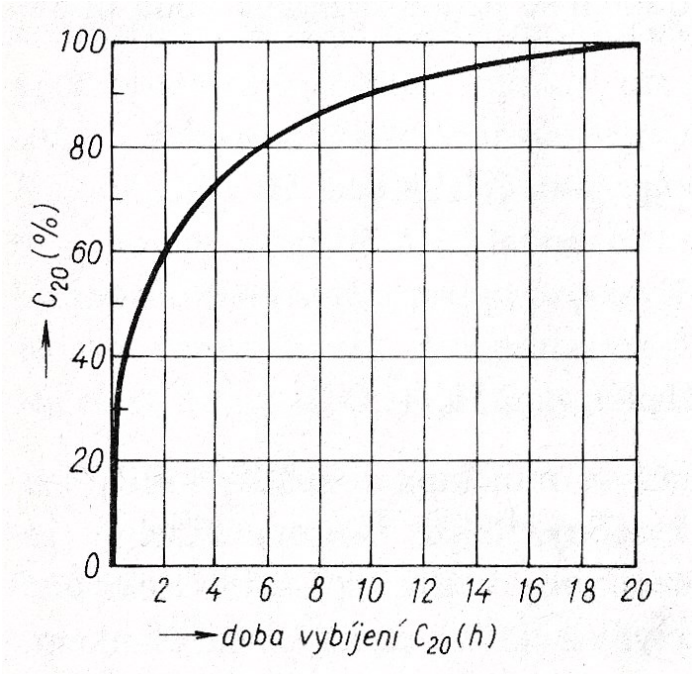
Výhody	Nevýhody
odolnost proti vytečení (elektrolyt absorbovaný v separátoru)	vyšší výrobní náklady, ale levnější než gelový typ
vysoký měrný výkon, nízký vnitřní odpor, reaguje na zatížení	citlivý na přebití, nicméně vyšší tolerance oproti gelu
nabíjení až cca 5× rychlejší oproti typu se zaplavenými elektrodami	kapacita postupně klesá (u gelového typu skokově ke konci životnosti)
lepší živostnost ve srovnání s konvenčními akumulátory	nízká měrná hustota energie
zadržování vody (reakce kyslíku a vodíku za vzniku vody)	nutnost skladování v nabitém stavu, avšak trpí méně než typ se zaplavenými elektrodami
odolnost proti vibracím díky zdvojené konstrukci	
dosahuje dobrých vlastností při nízkých teplotách	
menší sklon k sulfataci v případě, že baterie není pravidelně nabíjena naplno	
obsahuje méně elektrolytu a olova	

1.2.4 Elektrické parametry a charakteristiky

1.2.4.1 Kapacita akumulátoru

Základním parametrem každého akumulátoru je jeho jmenovitá kapacita C v jednotkách (A.h). Kapacita udává množství elektrického náboje, který je akumulátor schopen za určitých podmínek, tj. celkovém stavu akumulátoru, jeho účinnosti, velikosti vybíjecího proudu a teplotě, dodat do vnějšího elektrického obvodu. Obvykle se udává parametr C_{20}, tzv. 20hodinová kapacita, která představuje možnost zatěžování akumulátoru konstantním proudem až po dobu 20 hodin, při respektování teploty elektrolytu 25 °C (normalizovaný údaj). Pro demonstraci, např. olověnou baterii o kapacitě $C_{20} = 60\ A \cdot h$ by mělo být možné zatěžovat proudem $I = 60/20 = 3\ A$ až po dobu 20 hodin. Závislost

kapacity akumulátoru na vybíjecím proudu zachycuje charakteristika na *Obr. 1*. Někdy bývá také uváděna kapacita C_{10}, jíž odpovídá interval zatížení 10 hodin.



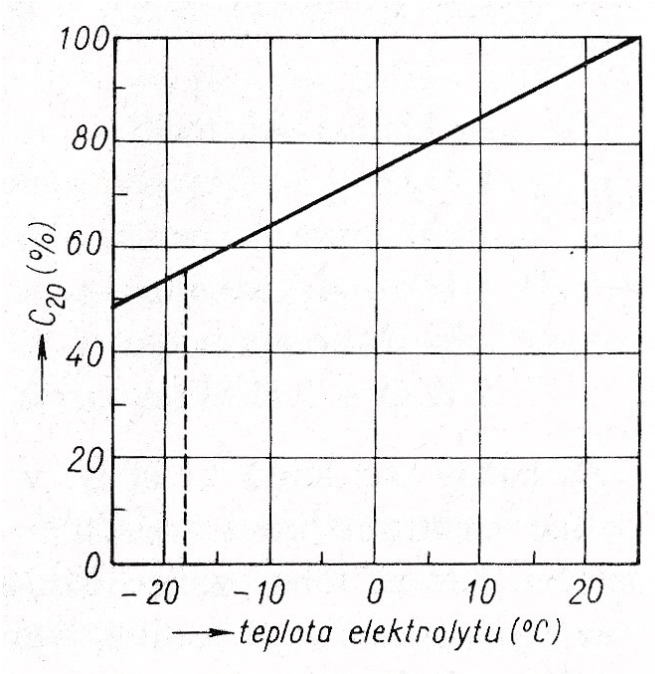
Obr. 1 Závislost kapacity olověného akumulátoru na vybíjecím proudu [3]

Skutečnou 20hodinovou kapacitu lze získat jedině měřením. Pro tyto účely akumulátor zatěžujeme konstantním proudem o velikosti 0,05násobku C_{20}, a to až do okamžiku dosažení napětí mezi vývody cca 10,5 V (platí pro 6článkový akumulátor), jež by měla odpovídat stavu hlubokého vybití. Po celou dobu měření je potřeba zaznamenávat oteplení elektrolytu. Kapacita se posléze určí jako součin vybíjecího proudu a doby vybíjení. Nicméně pro získání parametru hodného případnému srovnání s jmenovitou kapacitou, je ještě potřeba výslednou hodnotu přepočítat na vztažnou teplotu 25 °C, k čemuž lze využít následující vztah [3]:

$$C_{20} = \frac{C'}{1 - 0{,}01 \cdot (t - 25)}, \tag{1.1}$$

kde C_{20} je výsledná 20hodiná kapacita, C' je naměřená hodnota kapacity a t je aritmetický průměr počáteční a koncové teploty elektrolytu.

Teplota, jak již bylo nastíněno, je dalším faktorem ovlivňujícím nasazení akumulátoru. Při velmi nízkých teplotách dochází k významnému útlumu pohyblivosti nosičů náboje – iontů; zároveň se zvyšuje hustota elektrolytu, kyselina obtížněji vniká do pórů a zmenšuje se využití činné hmoty hlouběji pod povrchem desek [3]. Závislost kapacity akumulátoru na teplotě zachycuje charakteristika na *Obr. 2*.



Obr. 2 Závislost kapacity olověného akumulátoru na teplotě [3]

Pro získání přesnější představy o celkové možné době provozu akumulátoru, při konstantním odběru, je zapotřebí do výpočtu, vycházejícího z kapacity C_{20} zahrnout ještě energetickou účinnost akumulátoru. Uvažovaný vybíjecí proud stačí umocnit příslušným koeficientem [5], který se liší typ od typu olověného akumulátoru:

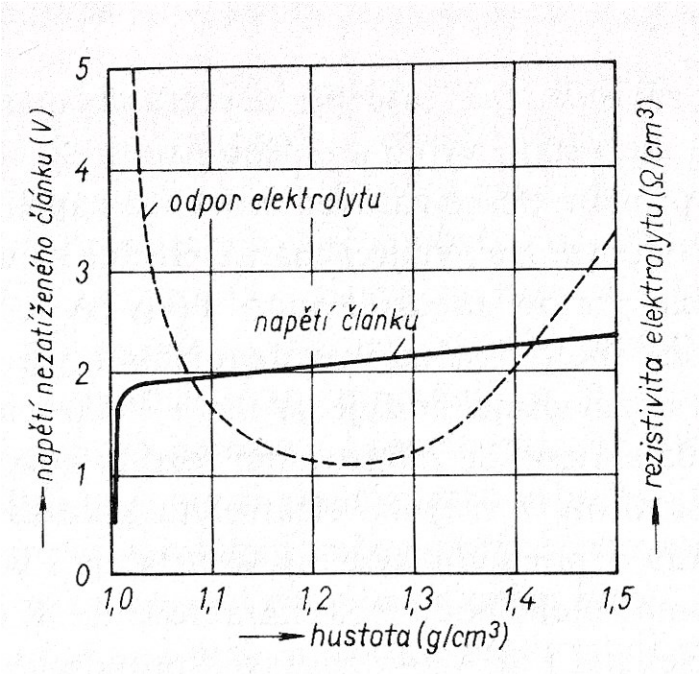
- akumulátor se zaplavenými elektrodami ... 1,20 ÷ 1,60
- gelový akumulátor ... 1,10 ÷ 1,25
- akumulátor s vázaným elektrolytem ... 1,05 ÷ 1,15.

Z výše uvedených koeficientů lze tak předpokládat nejvyšší účinnost provozu u typu s vázaným elektrolytem (AGM). Je dobré poznamenat, že kapacita olověných akumulátorů se postupem času spontánně snižuje vlivem změn ve struktuře činné hmoty, postupnou sulfatací, a z části také zanášením na dně nádoby v podobě kalu. Určitou anomálii představují tradiční akumulátory se zaplavenými elektrodami, u nichž se může v prvních několika cyklech dokonce objevit jistý nárůst kapacity [3].

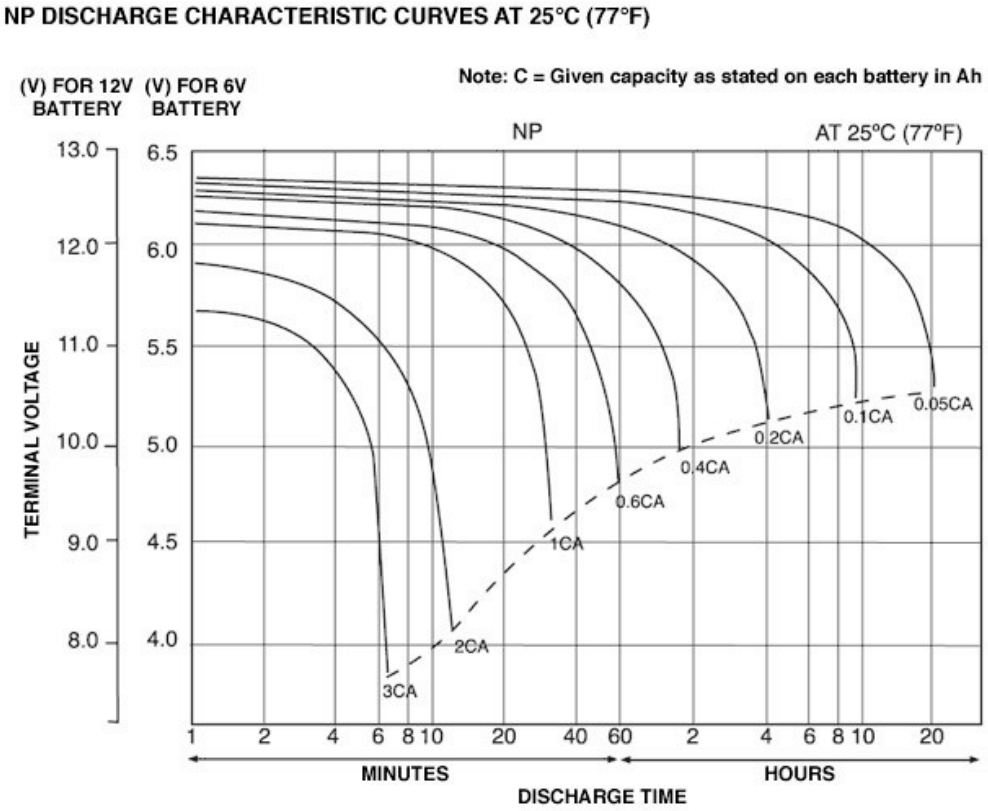
Vhodným nabíjením akumulátorů lze maximalizovat jejich životnost, resp. co možná nejvíce omezit pokles kapacity. Proces nabíjení je několikastupňový a zpravidla se využívá pulsních nabíjecích proudů. K tomuto účelu lze s výhodou použít speciálních inteligentních nabíječek. S ohledem na možné využití olověných akumulátorů pro elektromobilitu, je nutné poznamenat, že nabíjení prostřednictvím rekuperačního brždění, které je charakteristické velkými proudy, se může negativně promítnout do životnosti akumulátorů.

1.2.4.2 Napětí akumulátoru

Z hlediska napětí lze u olověných článků předpokládat nominální hodnotu 2 V na článek. U nezatíženého článku se může napětí, v závislosti na hustotě elektrolytu a teplotě, vyšplhat až na cca 2,15 V. Právě nabitý článek může po určitou dobu vykazovat o něco vyšší napětí; vybitý článek, jehož hustota elektrolytu je nižší, disponuje naopak nižším napětím. Ihned po nabití nebo vybití akumulátoru, je zapotřebí počítat s případným rozdílem mezi hustotou elektrolytu v pórech činné hmoty a vně elektrolytu, ovlivňujícím výsledné napětí článku, a to až do doby, než se hustota v celém objemu ustálí [3]. Napětí může dále významně poklesnout, pokud byl článek vybit hluboko pod přípustnou mez. Obecně lze však tvrdit, že napětí nezatíženého článku je jen velmi málo závislé na jeho úrovni nabití, což vyplývá ze závislosti na *Obr. 3*, kde malá odchylka od jmenovité hodnoty hustoty se jen velice málo odrazí na napětí. Zjišťování disponibilní úrovně kapacity by tedy mělo probíhat pouze za provozu (zatížení) s ohledem na vnitřní odpor zdroje (viz *Obr. 4* – vybíjecí charakteristiky), případně u akumulátoru v klidu s dostatečně dlouhým časovým odstupem, než dojde k ustálení všech elektrochemických jevů. U akumulátorů se zaplavenými elektrodami se tak doporučuje měřit napětí na svorkách nejdříve 2 hodiny po posledním nabíjení nebo jízdě, bezúdržbové akumulátory min. po 24 hodinách [5].



Obr. 3 Závislost napětí článku a vodivosti elektrolytu na hustotě elektrolytu [3]



Obr. 4 Vybíjecí charakteristiky olověného akumulátoru [7]

1.2.4.3 Vnitřní odpor akumulátoru

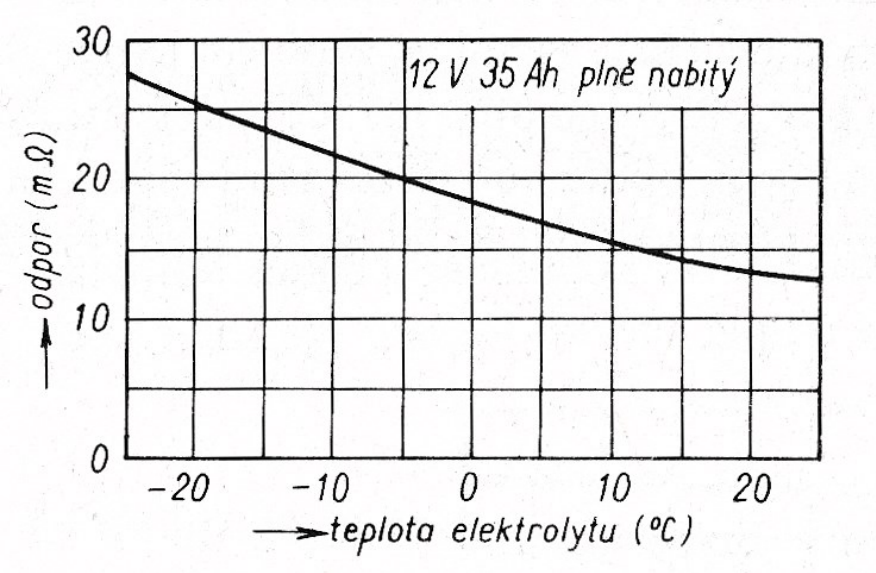
Celkový vnitřní odpor olověného akumulátoru, jakožto zdroje napětí, je součtem dílčích vnitřních odporů jednotlivých článků a propojovacích cest mezi nimi. Závisí na počtu, činném povrchu a stavu elektrod, na jejich konstrukci a přechodových vrstvách, na hustotě a teplotě elektrolytu, druhu separátorů atd. [3]. Vzhledem k těmto souvislostem nelze vnitřní odpor akumulátoru považovat za statický parametr, ale naopak dynamickou veličinu velice závislou na okamžitých podmínkách. Vnitřní odpor se pak následně promítá do vybíjecích charakteristik (viz *Obr. 4*) ve formě určitého úbytku napětí vlivem vybíjecího proudu. Hodnotu vnitřního odporu za definovaných podmínek lze experimentálně získat měřením napětí na vývodech při různých zatěžovacích proudech (viz *Vztah 1.2*).

$$R_V = \frac{u_1 - u_2}{i_2 - i_1}, \tag{1.2}$$

kde R_V je hodnota vnitřního odporu za definovaných podmínek, u_X jsou naměřené ustálené hodnoty napětí při odpovídajících proudech zátěží i_X.

Pozn.: napětí je vhodné odečítat až po ustálení hodnot, tj. cca 5÷10 s.

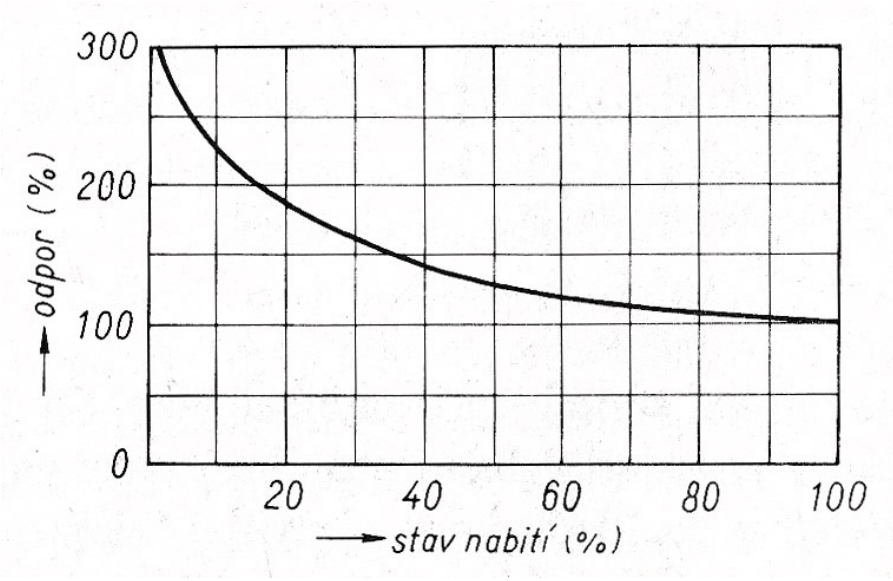
Teplotní závislost vnitřního odporu může hrát poměrně významnou roli, kdy při velmi nízkých teplotách lze očekávat až dvojnásobné hodnoty odporu oproti standardním podmínkám (viz *Obr. 5*). Výkon, který je schopen akumulátor poskytnout se tak sníží spíše z důvodu vyššího vnitřního odporu nežli samotnému snížení kapacity.



Obr. 5 Závislost vnitřního odporu olověného akumulátoru na teplotě [3]

Kromě teplotní závislosti má na vnitřní odpor dále vliv stav akumulátoru. Vybitý nebo výrazně opotřebovaný akumulátor vykazuje několikanásobně větší odpor (viz *Obr. 6*).

Jestliže se odpor plně nabitého akumulátoru zvětší přibližně na dvojnásobek odporu nového akumulátoru téhož typu, pak končí životnost tohoto akumulátoru [3].



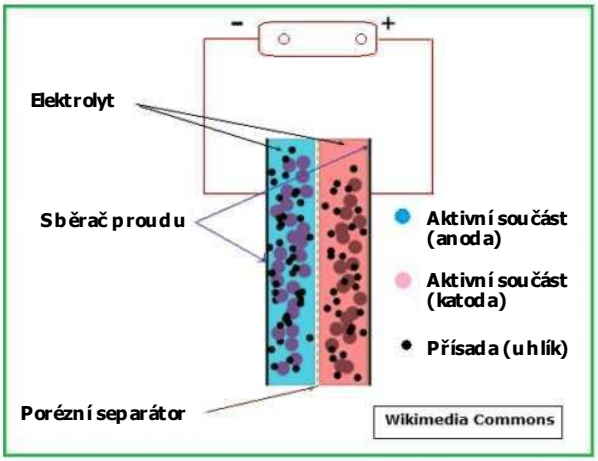
Obr. 6 Závislost vnitřního odporu olověného akumulátoru na poměrném stavu nabití [3]

1.3 Lithiové akumulátory

Aktuálně nejčastěji používanými typy článků pro elektromobilitu jsou právě lithium-iontové, případně lithium-polymerové. Základem lithiových článků je elektrolyt (kapalný či pevný), běžně se používá lithium hexafluorofosfát ($LiPF_6$) v nepolárním organickém rozpouštědle [8]. Během procesu nabíjení dochází na anodě k silnému obohacení lithiem, zatímco katoda zůstává naopak ochuzená. Oproti tomu při vybíjení (provozu), ionty lithia migrují z anody na katodu skrze elektrolyt, současně se uvolňují elektrony, které jsou shromažďovány sběrači proudu (current collectors). Sběrače proudu jsou tvořeny hliníkovou nebo měděnou fólií, v relaci k polaritě příslušné elektrody. V současné době se anoda skládá z grafitové směsi, případné úpravy v jejím složení již nepřinášejí žádná výrazná zlepšení. Oproti olověným akumulátorům, u těchto nedochází k chemickým reakcím při interakci iontů s mřížkou materiálu anody (záporné elektrody), díky čemuž je dosahováno delší životnosti bez výrazných poklesů výkonu [8]. Katoda (kladná elektroda) je tvořena kombinací lithia a dalšího, případně více vybraných kovy, jimiž se dosahuje požadované hustoty energie. Aktuálně se veškerý výzkum soustředí právě na aktivní materiály tvořící katodu, např.:

- NMC (NCM) – Lithium Nickel Cobalt Manganese Oxide ($LiNiCoMnO_2$),
- LCO – Lithium Cobalt Oxide ($LiCoO_2$),
- LFP – Lithium Iron Phosphate Oxide ($LiFePO_4$),
- LMO – Lithium Manganese Oxide ($LiMn_2O_4$),
- NCA – Lithium Nickel Cobalt Aluminium Oxide ($LiNiCoAlO_2$).

Dalším klíčovým prvkem je separátor. Jedná se o tenkou porézní membránu, která odděluje anodu a katodu mezi sebou a zabraňuje tak jejich přímému fyzickému kontaktu. Separátor musí disponovat dostatečnou mechanickou pevností a být schopen propouštět pouze ionty, zatímco pro elektrony se chová jako izolant. Na vhodném druhu separátoru závisí veškerá funkčnost a bezpečnost provozu baterie. Obvykle se k tomuto účelu používá plastových fólií z polypropylenu (PP), polyetylenu (PE), případně také keramických materiálů.



Obr. 7 Princip Li-ion bateriového článku [9]

Mezi výrobci lithiových baterií jsou rozšířeny různé typy struktur bateriových článků (viz *Obr. 8*), jejich použití se odvíjí na konkrétní aplikaci.

Obr. 8 Srovnání jednotlivých struktur Li-ion článků pro elektromobilitu [9]

Základní výhody a nevýhody technologie lithiových baterií jsou zobrazeny v následující tabulce:

Tabulka 4 Výhody a nevýhody nasazení lithiových baterií [9]

Výhody	Nevýhody
vysoká energetická hustota	krátká životnost (několik let)
nabíjení bez projevu tzv. paměťového efektu (není potřeba baterii zcela vybít a nabít pro obnovení plné kapacity)	nebezpečí „umření“ akumulátoru, pokud dojde k jeho vybití pod určitou mez (integrovány ochrany)
velmi dobře drží svůj náboj (ztratí pouze cca 5 % svého náboje měsíčně)	náchylnost k vysokým i nízkým teplotám (pokles kapacity), na druhou stranu rozsah pracovních teplot je široký
nízká hmotnost a „bezúdržbovost“ oproti klasickým olověným bateriím	při poškození separátoru hrozí u určitých typů vzplanutí baterie či dokonce výbuch
snesou velké nabíjecí proudy	potřeba speciální elektroniky (battery management system)

1.3.1 Lithium-iontové akumulátory

Lithium-iontové akumulátory (Li-ion) se obvykle vyskytující v podobě cylindrických článků s kapalným elektrolytem, u nichž jsou elektrody svinuty po obvodu [8]. Články disponují kovovým pláštěm, často normalizovaných rozměrů, s kombinací mechanických a elektronických bezpečnostních prvků. Při překročení stanovené mezní úrovně tlaku uvnitř článku, dojde vlivem separátoru k oddělení elektrod, a tím k přerušení exotermické reakce. Současně se zde nachází také tlakový ventil, který případný přetlak plynu vypustí mimo vnitřní prostředí akumulátoru. K tomu může dojít např. při nadměrné zátěži nebo nesprávné metodice nabíjení. Dále může být jednotlivý akumulátor vybaven např. ochranou proti přehřátí (PTC čidlem). V komerční sféře se články obvykle vyskytují ve známých velikostech „18650“. Jmenovité napětí jednoho článku činí cca 3,6 V, přičemž při procesu nabíjení se na kontakty přikládá napětí vyšší, tj. cca 4,2 V [8]. Energetická hustota dosahuje 150-200 Wh/kg [8]. Bateriové systémy elektromobilů se standardně skládají z nízkokapacitních článků řazených sério-paralelně.

Tabulka 5 Výhody a nevýhody Li-ion akumulátorů [18]

Výhody	Nevýhody
vysoká energetická hustota	akumulátory stárnou bez ohledu na to, zda jsou využívány
samovybíjení při pokojové teplotě do 5 % měsíčně	nebezpečí poškození nebo dokonce vzplanutí při nesprávném používání nebo nabíjení
netrpí paměťovým efektem	akumulátorům nesvědčí úplné vybití
vysoká životnost akumulátorů (cca 600 cyklů), záleží na hloubce vybíjení	akumulátory rychleji stárnou při velkých pracovních teplotách a nadměrné proudové zátěži

1.3.2 Lithium-polymerové akumulátory

Lithium-polymerové akumulátory (Li-pol) obsahují elektrolyt ve formě iontově vodivé polymerní sloučeniny. Články jsou často zapouzdřeny v poměrně pružné hliníkové fólii, což může představovat značné riziko, pokud by například došlo k mechanické deformaci, či výraznějšímu poškození obalu akumulátoru. Potom snadno dochází k vnitřním zkratům a nevratnému poškození článku. S výhodou lze u nich skládat elektrody na sebe, a dosahovat tak nejrůznějších tvarů pro optimální využití prostoru, např. ve spotřební elektronice. Energetická hustota lithium-polymerových článků je mírně vyšší ve srovnání s lithium-iontovými typy [8]. Jmenovité napětí se pohybuje v rozmezí 3,6 až 3,7 V.

Tabulka 6 Výhody a nevýhody Li-pol akumulátorů [18]

Výhody	Nevýhody
nejvyšší energetická hustota z lithiových typů akumulátorů	akumulátory stárnou bez ohledu na to, zda jsou využívány
nízká úroveň samovybíjení (5 % měsíčně) při pokojové teplotě	nebezpečí vzplanutí při mechanickém poškození nebo přímém zkratování
netrpí paměťovým efektem	akumulátorům nesvědčí úplné vybití, hrozí jim „umření“ při dlouhodobém nevyužívání

vysoká životnost akumulátorů (cca 1000 cyklů). záleží na hloubce vybíjení	až dvojnásobná pořizovací cena oproti Li-ion článkům
snesou velké nabíjecí a vybíjecí proudy	
možnost přizpůsobení se prostoru vyhrazeném pro umístění baterie	

1.3.3 Akumulátory na bázi lithium-železo-fosfát

Jedná se o akumulátory označované jako LiFe, $LiFePO_4$, nebo $LiFeYPO_4$. V třetím případě obsahuje materiál kladné elektrody navíc ytrium, které vede ke zlepšení vodivosti a stability struktury [8]. Oproti předešlým typům operují při nižším jmenovitém napětí (3,2 V) a nabíjecím napětí (3,6 V). Mimo to, lze u nich očekávat i nižší energetickou hustotu, cca 90-120 Wh/kg. Mezi výhody patří vyšší proudová zatížitelnost v relaci ke kapacitě, nepatrně vyšší odolnost vůči hlubokému vybití (díky tenkému potahu katody působícího proti její degradaci) a při 4 článcích možnost adekvátní náhrady za 12 V olověné akumulátory. Články LiFe jsou také v současné době považovány za jedny z nejbezpečnějších akumulátorů – nevytékají, nehoří, ani neexplodují [17].

Tabulka 7 Výhody a nevýhody LiFe akumulátorů [17] [18]

Výhody	Nevýhody
vysoká energetická hustota, avšak nižší oproti předchozím typům	nižší svorkové napětí článků
velký počet nabíjecích cyklů (1000 až 3000) závisející na hloubce vybíjení	rychlonabíjení snižuje životnost akumulátorů
velmi stálá kapacita vzhledem k počtu nabíjecích cyklů	možnost předčasného selhání při častém vybíjení pod cca 30 % kapacity
vysoká životnost (až 20 let v případě staničních akumulátorů)	větší míra samovybíjení
netrpí paměťovým efektem	horší výkon při nízkých teplotách
účinnost nabíjení až 95 %	
zpětná kompatibilita s Pb systémy	

nejbezpečnější provoz ze všech typů akumulátorů	

1.3.4 Význačné parametry lithiových akumulátorů

1.3.4.1 Jmenovitá kapacita (Nominal capacity)

Kapacita (A.h) představuje celkový náboj, kterým daná baterie disponuje za stanovených podmínek, tj. velikost vybíjecího proudu nebo teplota. Větší vybíjecí proudy, jakožto určité vyšší násobky kapacity, vedou na nižší hodnoty finální kapacity akumulátoru. Její hodnota se také přímo odvíjí od množství použitého aktivního materiálu.

1.3.4.2 Nominální energie (Nominal energy)

Nominální energie (W.h) vyjadřuje celkové množství energie odebrané baterii během jejího vybíjení definovaným vybíjecím proudem ze stavu plného nabití až do dosažení jejího konečného napětí. Energii lze následně vypočítat prostým vynásobením vybíjecího výkonu a doby do vybití baterie. Podobně jako u jmenovité kapacity, nominální energie je ponížena při vyšších násobcích C.

1.3.4.3 Specifická energie (Specific energy)

Specifická energie ($W.h.kg^{-1}$) udává nominální hodnotu energie vztaženou na celkovou hmotnost baterie. Vyšší hodnota při stejné hmotnosti baterie znamená delší dobu provozu, oproti baterii o nižší specifické hustotě. Cílem je tedy vývoj bateriových článků s co nejvyšší specifickou energií.

1.3.4.4 Specifický výkon (Specific power)

Na rozdíl od specifické energie, specifický výkon ($W. kg^{-1}$) vyjadřuje, jaké je zapotřebí množství aktivní hmoty pro dosažení požadovaného výkonu v daném čase. Stejně jako specifická hustota energie, je i specifický výkon závislý na chemickém složení a způsobu zapouzdření jednotlivých článků.

1.3.4.5 Objemová energetická hustota (Energy density)

Energetická hustota ($Wh.l^{-1}$) definuje kolik místa baterie zabere na místě aplikace [13], pro dosažení požadovaného dojezdu.

1.3.4.6 Výkonová hustota (Power density)

Výkonová hustota ($W.l^{-1}$) udává maximální dostupný výkon vztažený na jednotku objemu a určuje rozměry baterie, na základě nichž lze dosáhnout daných výkonnostních parametrů.

1.3.4.7 Stav nabití (State of charge)

Stav nabití reprezentuje momentálně dostupnou kapacitu baterie vzhledem k maximální možné, vyjádřeno v procentech. Změnu kapacity lze vysledovat na základě integrace proudu, resp. množství odvedeného náboje.

1.3.4.8 Hloubka vybití (Depth of discharge)

Hloubka vybití udává procentuální míru vybití oproti stavu s maximální dostupnou kapacitou, tzn. že pro plně nabitou baterii je hloubka vybití rovna nule. V případě odvedení více než 80 % náboje, hovoříme o tzv. hlubokém vybití neboli *deep discharge*.

1.3.4.9 Jmenovité napětí (Nominal voltage)

Jmenovité napětí se nachází přibližně uprostřed mezi plně nabitým a plně vybitým stavem baterie, přičemž jako vybíjecí proud je uvažován 0,2násobek její kapacity (0,2C). Jednotlivé lithiové články obvykle dosahují typicky jmenovitých napětí 3,2 až 3,8 V.

1.3.4.10 Svorkové napětí (Terminal voltage)

Svorkové napětí se typicky měří u baterie v zatíženém stavu, tedy s připojenou zátěží, a je poměrně závislé na stavu nabití/hloubce vybití.

1.3.4.11 Napětí naprázdno (Open-circuit voltage)

Napětí naprázdno lze naměřit, pokud ke svorkám baterie není připojena zátěž. Opět se jedná o hodnotu závislou na poměrném stavu nabití, která však může snadno přesáhnout jmenovitou hodnotu napětí.

1.3.4.12 Nabíjecí (konečné) napětí (Charge voltage)

Hodnota napětí, do níž je baterie obvykle nabíjena v režimu konstantního nabíjecího proudu (viz *Charge current*). Následně je článek nabíjen konstantním napětím, dokud proud neklesne pod stanovenou hodnotu, tzv. konečný nabíjecí proud, typ. 0,05C nebo 0,1C.

1.3.4.13 Konečné vybíjecí napětí (Cut-off voltage)

Konečné napětí je hodnota, při níž je proces vybíjení považován za dokončený a bylo tak dosaženo maximální užitečné kapacity baterie. Toto napětí se liší typ od typu baterie a také účelu, pro který je baterie využívána.

1.3.4.14 Nabíjecí proud (Charge current)

Výrobcem doporučená hodnota nabíjecího proudu, která je v první fázi nabíjecího cyklu udržována jako konstantní, přičemž takto lze dosáhnout cca 70 % stavu nabití baterie, posléze je udržováno konstantní nabíjecí napětí.

1.3.4.15 Maximální trvalý vybíjecí proud (Maximum continuous discharge current)

Jedná se o mezní hodnotu proudu definovanou výrobcem baterie, kterou může být akumulátor trvale vybíjen, aniž by došlo k poškození baterie či nevratnému snížení její kapacity. Společně se maximálním trvalým výkonem elektromotoru ovlivňuje nejvyšší udržitelnou rychlost vozidla a jeho zrychlení.

1.3.4.16 Krátkodobý 30 s vybíjecí proud (Maximum 30-sec discharge pulse current)

Maximální hodnota proudu, kterou je možné krátkodobě odebírat po dobu až 30 vteřin. Současně se špičkovým výkonem elektromotoru má zásadní vliv na zrychlení vozidla.

1.3.4.17 Vnitřní odpor (Internal resistance)

Velikost vnitřního odporu může být rozdílná během procesu nabíjení a vybíjení, a rovněž jako všechny předchozí parametry závisí na stavu nabití. Čím vyšší je jeho hodnota, s tím nižší účinností baterie pracuje. Vznikají tak dodatečné výkonové, resp. tepelné ztráty a svorkové napětí je pak více či méně ovlivněno úbytky na vnitřním odporu.

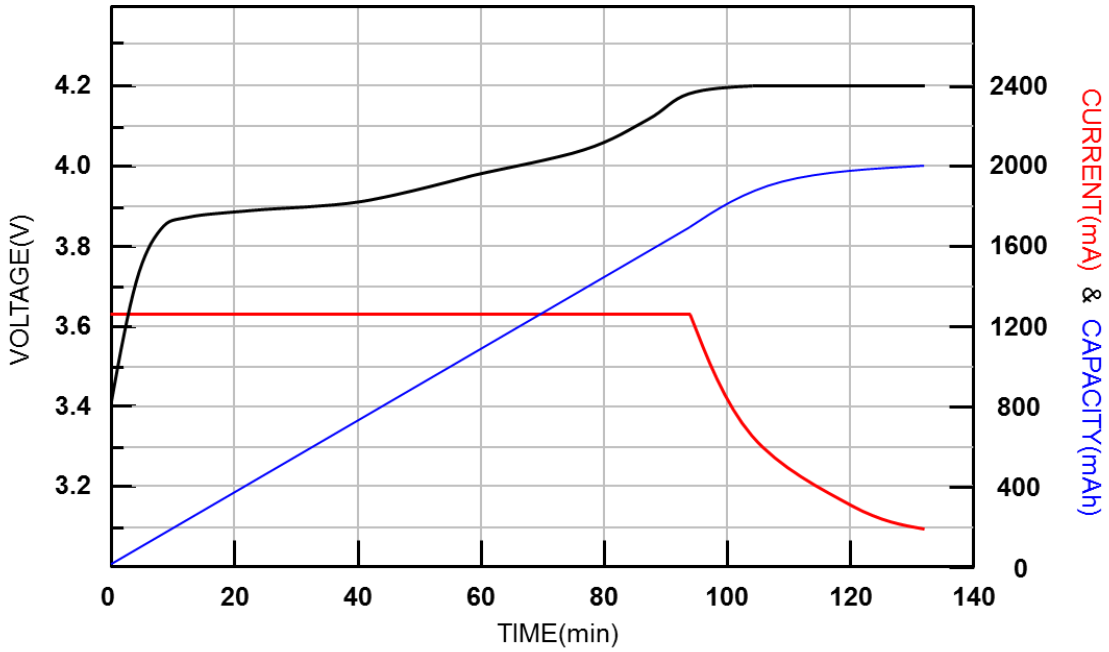
1.3.4.18 Počet nabíjecích cyklů (Cycle-life)

Každá nabíjecí baterie vykazuje určitý počet nabíjecích cyklů, než dojde ke zhoršení jejích provozních parametrů pod jistou přípustnou mez – State of Health (*SoH*). Počet nabíjecích cyklů je velmi závislý na podmínkách nabíjení a vybíjení, tj. hloubce vybití, teplotě, vlhkosti apod. Čím větší se uplatňuje hloubka vybití, tím kratší lze uvažovat životnost.

1.3.5 Charakteristiky lithiových akumulátorů

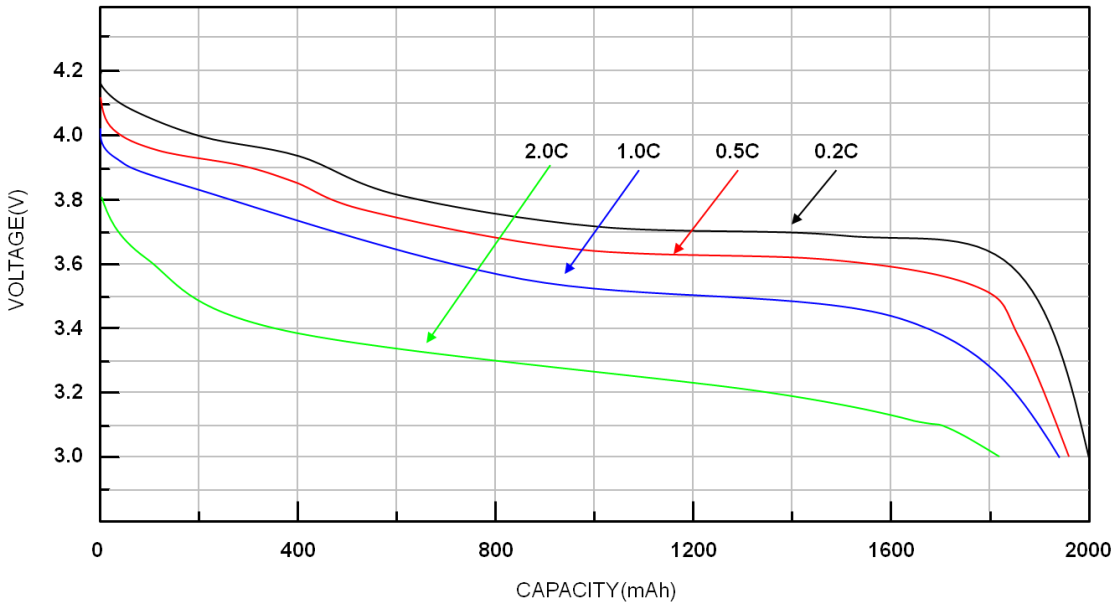
K základním charakteristikám článků řadíme nabíjecí a vybíjecí charakteristiky. Nabíjecí proces vyžaduje v jeho první fázi regulaci nabíjecího proudu, kontrolu napětí článku a teploty [14]. Velikost nabíjecího proudu vychází z výrobcem doporučených násobků

jmenovité kapacity (v A.h). U klasických Li-ion článků by napětí nemělo překročit hodnotu cca 4,2 V, jinak může dojít ke zhoršení vlastností akumulátoru. Po dosažení konečného nabíjecího napětí článek disponuje zhruba 70 % své nominální kapacity. Toto napětí se dále udržuje, nicméně nabíjecí proud postupně klesá. Proces nabíjení je u konce, jakmile se nabíjecí proud sníží na hodnotu odpovídající cca 0,05 až 0,1C.

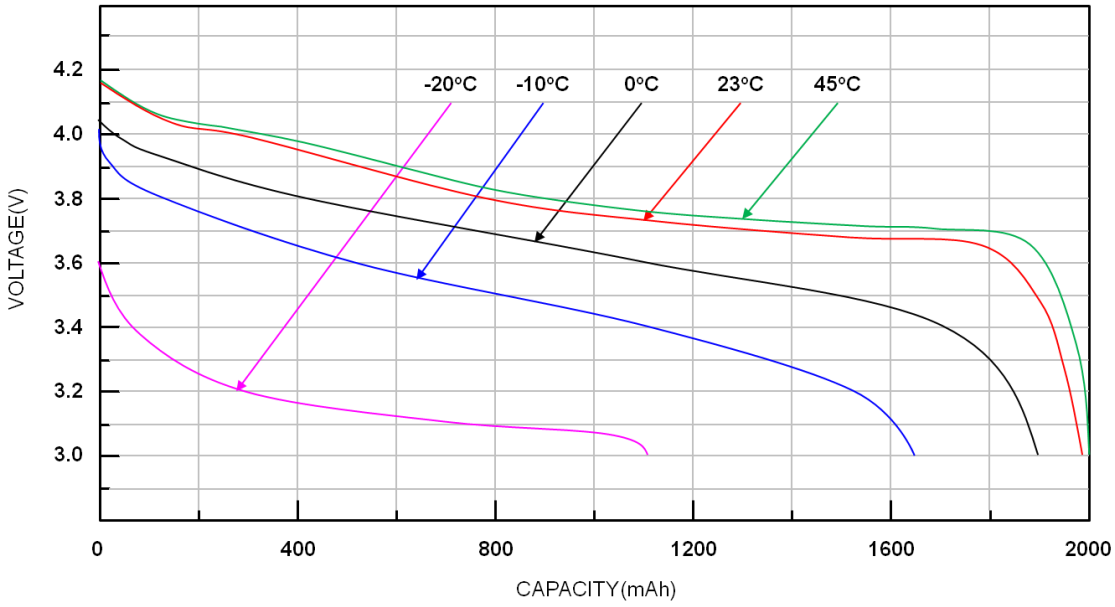


Obr. 9 Průběh napětí a proudu při nabíjení typického Li-ion akumulátoru [15]

Během vybíjení akumulátoru postupně klesá hodnota napětí, přičemž míra propadu napětí se odvíjí od velikosti vybíjecího proudu a stavu nabití. Proces vybíjení by měl být ukončen při dosažení konečného vybíjecího napětí, typ. 3 V pro Li-ion typ článku. Další vybíjení by mohlo eskalovat k degradaci elektrických vlastností.

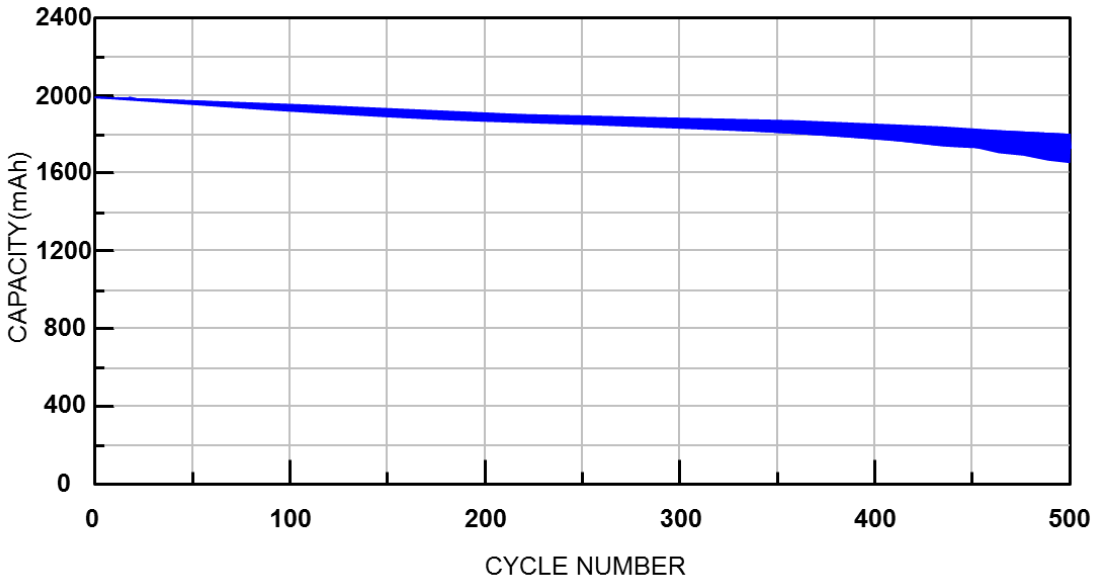


Obr. 10 Typická vybíjecí charakteristika Li-ion akumulátoru pro různé hodnoty vybíjecích proudů [15]

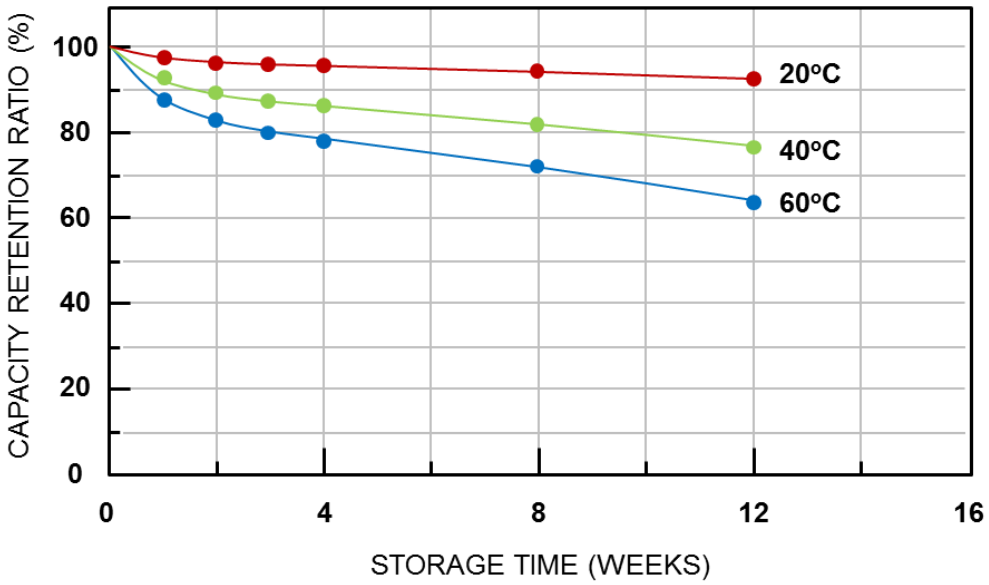


Obr. 11 Typická vybíjecí charakteristika Li-ion akumulátoru pro různé hodnoty teploty [15]

Veškeré potřebné elektrické mezní i doporučené parametry, charakteristiky a pracovní podmínky jsou vždy uvedeny v katalogových listech výrobce konkrétního typu akumulátorů.



Obr. 12 Vliv počtu nabíjecích/vybíjecích cyklů na kapacitu Li-ion akmulátoru [15]



Obr. 13 Vliv teploty skladování na proces samovybíjení [15]

1.4 Porovnání akumulátorů

Lithiové akumulátory se oproti olověným typům ukázaly jako vhodnější variantou, a to z důvodu vyšší hmotnostní a objemové energetické hustoty. Jejich nevýhodou je však pořizovací cena. Klíčovým prvkem u akumulátorů je vždy teplota, která ovlivňuje jejich základní provozní parametry a degradační procesy. Vliv teploty na užitečnou kapacitu, jakožto hlavní parametr, je v obou případech obdobný. Při velmi nízkých teplotách, tj. -20 °C, lze počítat s poklesem až o 40 % nominální hodnoty. Toto úskalí je částečně

kompenzováno ohřevem článků od přítomných výkonových ztrát na nich vznikajících. Bateriím rovněž nesvědčí ani vyšší pracovní teploty, ideální uváděná teplota je proto 25 °C. Dalším žádaným parametrem je počet nabíjecích cyklů, přičemž s jejich navyšováním je potřeba počítat s postupným trvalým snižováním jmenovité kapacity. U Li-ion a gelových akumulátorů (VRLA) lze po cca 400 nabíjecích cyklech stále počítat s min. 80 % nominální kapacity, nicméně velmi také záleží na jejich hloubce vybíjení (DoD). Není proto vhodné akumulátory vybíjet pod více než 20 %. Užitečnou kapacitu mohou také ovlivnit velikosti vybíjecích proudů. U moderních typů akumulátorů a vybíjecích proudech řádově o nižších násobcích *C* lze tento vliv teoreticky zanedbat. Zajímavou alternativou za původní olověné akumulátory lze považovat články $LiFePO_4$, které jsou z hlediska napěťových úrovní případnou adekvátní náhradou, a zároveň vynikají vysokou bezpečností provozu.

2 Telemetrické zařízení

Pro účely měření provozních parametrů za jízdy elektromobilem, byla zkonstruována telemetrická sestava, skládající se z bezdrátového měřícího zařízení a GPS modulu. K ovládání, sběru a zpracování dat, slouží uživatelsky vytvořený ovládací software navržený pro platformu Windows. S pomocí tohoto měřícího aparátu bylo možné získat představu o chování bateriového systému za různých jízdních podmínek, tj. sledovat elektrické i neelektrické parametry, které ve výsledku více či méně ovlivňují dojezd elektromobilu. Konkrétní specifikace jednotlivých zařízení i dílčích modulů bude do detailu rozebrána.

2.1 Bezdrátové měřící zařízení

Pro bezdrátové měřící zařízení byly prvotně uvažovány následující široce rozšířené vývojové platformy:

- Arduino UNO,
- Arduino Mega,
- Raspberry Pi 4B,
- Raspberry Pi Pico.

Níže uvedená tabulka shrnuje některá jejich vybraná klíčová technická specifika.

Tabulka 8 Technické parametry uvažovaných vývojových platforem

	Arduino UNO Rev3	Arduino Mega Rev3	Raspberry Pi 4B (1 GB)	Raspberry Pi Pico
Procesor	Single-core ATmega328P (16 MHz)	Single-core ATmega2560 (16 MHz)	Quad-core Arm Cortex-A72 (1,5 GHz)	Dual-core Arm Cortex-m0+ (133 MHz)
RAM	2 KB	8 KB	1 GB	264 KB
ROM/FLASH	1 KB	4 KB	slot na externí microSD kartu	2 MB
GPIO	14 digitálních	54 digitálních	28 víceúčelových	26 víceúčelových
Logika	5 V	5 V	3,3 V	3,3 V

A/D	6×10-bit	16×10-bit	-	3×12-bit
Konektivita	1×UART, 1×SPI, 1×I2C, 6×PWM, 1×USB	4×UART, 1×SPI, 1×I2C, 15×PWM, 1×USB	1×UART, 1×SPI, 1×I2C, 2×PWM, 4×USB, 2×microHDMI, WiFi, Bluetooth	2×UART, 2×SPI, 2× I2C, 16×PWM, 1×microUSB
Primární programovací jazyk	C++	C++	Python, C/C++, … (závisí na OS)	MicroPython, CircuitPython, C/C++
Napájení	1,8 ÷ 5,5 V DC	1,8 ÷ 5,5 V DC	5 V DC / 3 A	1,8 ÷ 5,5 V DC
Low-power mód	ano	ano	ne	ano
Rozsah teplot	-40 ÷ 85 °C	-40 ÷ 85 °C	0 ÷ 50 °C	-20 ÷ 85 °C
Rozměry	68,6×53,4 mm	101,52×53,3 mm	85,6×56,5 mm	21×51 mm
Pořizovací cena	€20 €7÷10 (klon)	€35 €15÷20 (klon)	€45	€5

Na základě technických parametrů, z toho dále plynoucích výhod a nevýhod, a uživatelskému testování jednotlivých vývojových desek, byla zvolena platforma Raspberry Pi Pico.

Tabulka 9 Výhody a nevýhody vývojových platforem

	Arduino UNO Rev3	**Arduino Mega Rev3**	**Raspberry Pi 4B (1 GB)**	**Raspberry Pi Pico**
Výhody	+ jednoduchost použití	+ jednoduchost použití	+ vysoký výpočetní výkon a možnost použití vláken	+ velký výpočetní výkon a podpora 2 jader

	+ rozsáhlý soubor knihoven a komunita	+ rozsáhlý soubor knihoven a komunita	+ využívá „plnohodnotný“ OS Linux, případně Windows	+ disponuje vnitřním úložištěm pro program a knihovny
	+ relativně kompaktní provedení	+ lepší konektivita oproti UNO	+ nadstandardní konektivita (Wifi, BT)	+ velice kompaktní rozměry
	+ variabilita napájecích napětí	+ variabilita napájecích napětí	+ jazyk Python	+ jazyk MicroPython
				+ cena
Nevýhody	- velice omezený prostor pro hlavní program	- nepříliš kompaktní	- vyšší nároky na napájení (příkon)	- obtížnější dohledávání knihoven
	- úprava programu vyžaduje opětovnou kompilaci	- úprava programu vyžaduje opětovnou kompilaci	- vyšší taktovací frekvence může být zdrojem rušení	- hloubka vnořených volání funkcí omezena na 10
	- pouze 10-bit AD převodník	- pouze 10-bit AD převodník	- starší verze hardwaru nemusí být kompatibilní s novým verzemi Pi OS	- při nastavení automatického běhu hlavního programu, je při jeho případné změně nutné smazat celou paměť a znovu nahrát firmware

	- pouze 1× UART		- cena	

2.1.1 Popis zařízení a jeho funkce

Bezdrátová měřící jednotka sestává ze samotného jednočipového počítače Raspberry Pi Pico, napájecích modulů v podobě lineárních stabilizátorů napětí *AMS1117*, a specializovaných samostatných měřících modulů. K těmto patří externí A/D převodník *ADS1015/ADS1115*, sdružený modul gyroskopu, akcelerometru a barometru *MPU925X*, dále proudové čidlo *HSTS016L*, a dvojice digitálních teplotních čidel *DS18B20*. Pro opatření dat časovým razítkem je použit RTC modul *DS1302*. Bezdrátová komunikace je zajištěna prostřednictvím Bluetooth modulu typu *HC-06*.

Obr. 14 Instalované měřící zařízení

2.1.1.1 Jednočipový počítač Raspberry Pi Pico

Vývojová platforma RPi Pico běží na uživatelsky zvoleném firmwaru založeném na Micropythonu verze 1.18. Ten je možné do zařízení nainstalovat tak, že před samotným připojením Raspberry k PC se podrží na tlačítko *BOOTSEL* na svrchní straně desky,

následně stačí mikrokontrolér připojit k PC přes USB port, vyčkat ještě několik vteřin a poté uvolnit ono tlačítko. Pico se poté připojí jako běžná externí disková jednotka, jež v základu obsahuje 2 soubory se základní nápovědou, která obsahuje odkazy pro stažení potřebného firmwaru. Případně lze požadovanou verzi firmwaru stáhnout přímo z oficiálních stránek https://micropython.org/download/rp2-pico/. Soubor s koncovkou *UF2* pak již stačí jen přetáhnout do hlavního adresáře RPi Pico, které se po chvíli samovolně odpojí a provede instalaci dané verze MicroPythonu na pozadí. K programování mikrokontroléru v jazyce MicroPython lze následně použít např. volně dostupné vývojové prostředí Thonny Python IDE (https://thonny.org/).

Kromě MicroPythonu lze Pico také programovat v jazyce C/C++, nebo použít zjednodušenou verzi Micropythonu, tj. CircuitPython (https://circuitpython.org/). Nespornou výhodou CircuitPythonu je především podpora nativních knihoven pro celou řadu rozšiřujících modulů, od základních až po specializované, tj. např. od výrobce Adafruitu. Nevýhodou naopak pro někoho může být absence obsluhy přerušení (interrupts) nebo nemožnost přístupu k druhému jádru (vláknu) procesoru. Z uživatelského pohledu se CircuitPython dobře osvědčil pro prvotní testování některých modulů, u jiných bohužel nebylo možné dohledat potřebnou knihovnu. Knihovny jsou dostupné jak ve zkompilované verzi, tak i čitelné podobě kódu, záleží pouze na preferenci. Automatické spuštění hlavního programu se zde realizuje jeho uložením pod názvem „*code.py*“, narozdíl od MicroPythonu, kde se používá „*main.py*“. Použití těchto vyhrazených názvů je však doporučováno až pro finální podobu programu, neboť pokud dojde ke spuštění kódu samovolně, tj. bez zásahu uživatele v prostředí Thonny, pak RPi Pico není možné nadále pomocí IDE řídit, ani přistupovat k žádným souborům na něm uložených, nelze se prakticky k mikrokontroléru vůbec připojit! V případě CircuitPythonu se může uživatel pokusit o tzv. měkký restart (soft reboot) v záložce *Run*. Při použití MicroPythonu však nezbude nic jiného než přemazat celou vnitřní flash paměť pomocí „flash_nuke.uf2“ a znovu nahrát firmware, tím však dojde ke ztrátě veškerých uložených dat, tj. včetně programu. Alternativně lze použít *„MicroPython_RenameMainDotPy.uf2“*, který pouze přejmenuje spustitelný soubor a umožní tak přístup k datům, nicméně stále neodpadá nutnost opětovného nahrání výkonného firmwaru.

Pozn. 1: Pro aplikaci souborů s koncovkou *UF2* je vyžadován postup uvedený v 1. odstavci této kapitoly), tj. s pomocí *BOOTSEL*.

Pozn. 2: Ukázkové zdrojové kódy hlavního programu, resp. jeho dvou variant, v jazyce MicroPython jsou dostupné v *Přílohách #1 a #2*.

2.1.1.2 Systém napájení

Jako hlavní napájecí zdroj je použito 6×AA baterií, které napájí externí lineární napěťový stabilizátor *AMS1117*, jež na výstupu produkuje napětí 5 V. Z tohoto napětí je následně napájen druhý stabilizátor *AMS1117* v řadě, tentokrát však o jmenovitém výstupu 3,3 V. Napětí 3,3 V slouží jako tvrdý referenční napěťový zdroj pro externí AD převodník. 5 V stabilizátor je rovněž použit k napájení mikrokontroléru, avšak navíc je do napájecí cesty vřazena Schottkyho dioda *1N5819* (40 V/1 A), která způsobuje mírný úbytek napětí o hodnotě cca 0,2 V. Tento úbytek však nemá žádný vliv na funkčnost jednočipu. Hlavním účelem této diody je eliminace nežádoucích toků vyrovnávacích a nabíjecích proudů, které by se objevily v případě napájení Raspberry Pi Pico z více zdrojů, např. při současném programování skrze micro-USB, kdy je mimo jiné přítomno i napětí z USB sběrnice. Dioda vlastně pomáhá zprostředkovávat „OR" funkci, kdy pro napájení je v danou chvíli vždy použito pouze zdroje o vyšším jmenovitém napětí. Napájecí a zemnící rozvod dílčích modulů je koncipován přímými propojovacími cestami na napájecí, resp. zemnící body, a to prostřednictvím vodičů o průměru 0,25 mm^2. Nicméně pro prvotní testování (na nepájivém poli) mohou dobře posloužit i standardní zemnící cesty, které zajišťuje přímo RPi Pico (celkem 8×GND). Bohužel pak zvýšený počet, a tedy i příkon modulů, může způsobit citelné úbytky napětí. Ukázkovým příkladem zde může být Bluetooth komunikační modul, který v nespárovaném stavu odebírá proud cca 30÷40 mA, což může vést k poklesům napětí o několik setin až desetin voltu.

2.1.1.3 Měření napětí a proudu

Měření elektrických veličin napětí a proudu zprostředkovává modul analogově-digitálního převodníku s označením *ADS1015/ADS1115*. Jedná se o kombinaci 12-bitového a 16-bitového převodníku, z nichž je použit právě 16-bitový, tzn. *ADS1115*. Vyšší rozlišení nalézá uplatnění zejména při měření napěťového výstupu z proudového čidla. Je dobré dodat, že plného 16-bitového rozlišení lze dosáhnout pouze u diferenčního měření napětí, tj. měření rozdílu potenciálů. Tento způsob měření ovšem zabírá rovnou 2 analogové vstupy naráz. Celkový počet analogových vstupů je 4. Při použití vhodné knihovny pro MicroPython není potřeba pracovat s „raw" daty a místo toho rovnou získat hodnotu napětí ve voltech. Rozsah měřených napětí je omezen napájecím napětím modulu, resp. stejně tak je i omezeno max. rozdílové napětí mezi 2 analogovými vstupy. Měření napětí probíhá v tzv. „single-shot" módu, tj. při obdržení programového požadavku na novou hodnotu, kdy je však nutné čekat na dokončení konverze. Alternativou může být i tzv. „continuous" režim

s periodickým vzorkováním vyvolávající požadavek na hardwarové přerušení (reakce na sestupnou hranu). Mimo nastavení měřícího režimu je dostupná i volba případného zesílení vstupního signálu, zisk byl nastaven na 1 (bez zesílení) při vstupním rozsahu ±4096 mV.

Převod vyššího napětí měřené baterie elektromobilu, tj. cca 84 V, je realizován prostřednictvím vysokoohmového napěťového děliče, složeného z rezistorů 428 kΩ a výstupního rezistoru 10 kΩ. Odpovídající velikosti rezistorů byly naměřeny multimetrem a převodový poměr ověřen měřením napětí na výstupu děliče. Navržen byl rovněž i 9,33 MΩ dělič, který byl však z hlediska potenciálně vyššího šumu nahrazen právě 438 Ω, jehož převodový poměr je:

$$p_U = \frac{428 + 10}{10}\ (-) \tag{2.1}$$

Proudové čidlo *HSTS016L* využívá principu Hallova jevu v kombinaci s děleným magnetickým obvodem/jádrem, jímž je možné lépe koncentrovat přítomné magnetické pole od průchozího vodiče, což má pozitivní vliv na obdržené hodnoty. Čidlo je napájeno přímo z výstupu 5 V stabilizátoru napětí. Disponuje celkem 4 piny, tj. 2 napájecími a 1 výstupním a 1 kalibračním pinem. Výrobce dodává čidla s různými jmenovitými proudovými rozsahy, od kterých se odvíjí a je zaručena lineární oblast, resp. převodní charakteristika, těchto čidel (Hallovy sondy). Maximální proudový rozsah může být však až o 50 % vyšší než jmenovitý. Hallův jev lze s výhodou uplatnit jak pro měření střídavých, tak i stejnosměrných proudů, resp. účinků těchto proudů v podobě magnetických polí (magnetické intenzity a indukce). Reakční doba je velice malá, cca 1 μs, a měření je zatíženo chybou max. 1 % z jmenovitého rozsahu. V závislosti na jmenovitém rozsahu se stanoví převodní koeficient, jenž se stanoví následujícím způsobem:

$$p_I = \frac{100}{0{,}625}\ (A/V), \tag{2.2}$$

kde hodnota 100 A odpovídá jmenovitému proudovému rozsahu a konstanta 0,625 V vyjadřuje max. přírůstek napětí vyvolaný měřením jmenovité hodnoty proudu (100 A).

Pozn.: Napětí na výstupním pinu proudového čidla je posunuto o polovinu napájecího napětí, přesnou hodnotu „offsetu“ lze získat měřením napětí na kalibračním pinu. Nicméně jednodušší možností při dostatku analogových vstupních pinů AD převodníku, je měření přímo hodnoty rozdílového napětí mezi těmito dvěma piny, což je i náš případ.

Naměřené rozdílové napětí mezi výstupním a kalibračním pinem stačí vynásobit proudovým převodem. V případě měření střídavých či pulzních proudů, je zapotřebí

periodicky načítat druhé mocniny naměřených napětí, za stanovenou dobu tento součet vydělit celkovým počtem hodnot, následně odmocnit, a nakonec vynásobit převodní konstantou. Tento postup vede k získání tzv. TrueRMS, tj. skutečné efektivní hodnoty proudu, kterou je možné změřit například lepším klešťovým multimetrem. Po nezávislých měření různých velikostí proudů s tímto čidlem, byly obdrženy výsledky velice blízké těm, udávaným komerčním klešťovým multimetrem.

Vedle měření provozních elektrických veličin pomocí telemetrického zařízení, jsou rovněž k dispozici též hodnoty z tzv. *BMS*, neboli *battery management system*. Ty mohou později posloužit alespoň pro orientační srovnání dvou nezávislých systémů měření. Dále z důvodu prvotního ověření funkce měřícího zařízení, proběhlo testovací měření klidového proudu, se zapnutými světlomety pro zvýšení odběru, s využitím klešťového multimetru.

2.1.1.4 Časová synchronizace prostřednictvím RTC modulu

RTC modul s označením *DS1302*, resp. hodiny reálného času, umožňuje časově synchronizovat datové toky z dílčích telemetrických jednotek, tj. měření provozních veličin trakční baterie s GPS daty. Časové razítko navíc představuje primární klíč, kterým jsou rozlišeny jednotlivé řádky v příslušné databázové tabulce. Mikrokontrolér RPi Pico v kombinaci s časovými knihovnami MicroPythonu umožňuje adaptaci aktuálního data a času, tudíž je možné aktuální synchronní UTC časový údaj načíst z externího RTC pouze jednou při inicializaci hlavního programu, a Pico je následně schopné informaci o datu a času udržet a pravidelně aktualizovávat. K výhodám tohoto předání patří rychlejší čtení aktuálního časového razítka a snížení vlivu vybíjení RTC baterie. Synchronní UTC čas uložený v externím RTC modulu, je však potřeba periodicky, nejlépe cca 1× týdně aktualizovávat, aby byla eliminována každá případná odchylka. Nastavení aktuálního data a času lze např. provést jeho načtením z PC (viz *Příloha #3*) při fyzickém propojení RPi Pico s Thonny IDE, a využitím funkce *localtime()* z balíku časové knihovny *time* pro MicroPython. Samotné předání RTC modulu je možné realizovat zavoláním čtecí funkce/metody objektu RTC s parametrem v podobě pole data a času ve správném definovaném formátu/pořadí. Posléze je ještě možné časový údaj manuálně poupravit.

Jelikož interní časové funkce, podobně jako externí RTC modul, nepodporují rozlišení na milisekundy, případně mikrosekundy, je zapotřebí pro bezproblémové ukládání do databáze toto omezení překlenout. Dodatečné rozlišení bylo realizováno počítáním tzv. ms ticků, resp. *time.ticks_ms()* od definovaného okamžiku, tj. bezprostředně po synchronizaci s externími RTC a předání data a času interním. Pomocí nekonečného cyklu program vyčká na

následující změnu času v řádu sekund, a jakmile k ní dojde, uloží si aktuální počet ticks. Během skládání časového razítka je s výhodou použit operátor modulo, kterým lze extrahovat zbylé milisekundy od celých sekund.

$$millis = (now - start)\ \%\ 1000 \tag{2.3}$$

kde *millis* (ms) je výsledný počet milisekund (přídavek k aktuálnímu časovému razítku), *now* (ms) je aktuální počet milisekund od nabootování desky a *start* (ms) je inicializační počet milisekund (vycházející z okamžiku synchronizace na celé sekundy).

Pozn. 1: RTC časové razítko nelze nikdy nastavit naprosto přesně, stále je potřeba počítat s relativní chybou řádově až 1 s. Pro účely ukládání do databáze je vhodné zabezpečit dodatečné rozlišení na milisekundy pro odlišení dat přijatých v rámci 1 s intervalu.

Pozn. 2: Funkce *localtime()* vrací pole hodnot v jiném formátu, než vrací či požaduje knihovna RTC modulu, tj.:

(year, month, month_day, hours, minutes, seconds, week_day, year_day),

oproti tomu vstupní/výstupní formát pro modul RTC je následující:

(year, month, month_day, week_day, hours, minutes, seconds).

Podobný formát využívá i interní RTC podpora z balíku *time*, avšak navíc s rozšířením o tzv. *subseconds*, jehož význam závisí na použitém HW, v případě Pi Pico je tento parametr vracen vždy jako nulový:

(year, month, month_day, week_day, hours, minutes, seconds, subseconds).

2.1.1.5 Čtení údajů z akcelerometru a gyroskopu

Modul *MPU925X* poskytuje informace o poloze, zrychlení a barometrickém tlaku. Jedná se o kombinaci 3-osého gyroskopu, 3-osého magnetometru, 3-osého akcelerometru, tlakového a teplotního senzoru. V poslední verzi telemetrického zařízení je implementováno pouze měření zrychlení a barometrického tlaku, resp. nadmořské výšky, pro jejíž výpočet je dále zapotřebí znát teplotu a lokální barometrický tlak redukovaný na hladinu moře.

Akcelerometr rozkládá zrychlení do celkem tří význačných směrů *(X, Y, Z),* které jsou vyobrazeny na samotném plošném spoji modulu. Pro správnou funkci se též definuje maximální rozsah přetížení, jakožto násobek gravitačního zrychlení. Dle zdroje [10] by přetížení při rozjezdech či intenzivním brždění nemělo za běžných podmínek překročit hranici 1 g, tj. přibližně 9,8 $m.s^{-2}$, konkrétní hodnoty jsou uvedeny v následujících tabulkách. Nejmenší dostupný a současně i nastavený rozsah akcelerometru je proto 2 g. Prostřednictvím údajů z akcelerometru lze při zpětné analýze podchytit případné rozjezdy, brždění, změnu směru jízdy a vyhodnocovat i jejich intenzitu.

Tabulka 10 Hodnoty přetížení u vybraných typů automobilů při intenzivním brždění na suchém rovném povrchu z rychlosti 96,6 kmph až do úplného zastavení [10]

Vozidlo	g	m/s^2	kmph/s
BMW M3	1,00	9,85	35,4
Toyota Celica GT	0,94	9,21	33,2
Lincoln Continental	0,92	9,02	32,5
Nissan Maxima	0,85	8,32	29,9
Chevy Blazer	0,76	7,47	26,9
Dodge Colt GL	0,72	7,07	25,4

Tabulka 11 Hodnoty přetížení během brždění v závislosti na dovednostech řidiče, situaci, nebo technických limitech automobilu [10]

Brždění	g	kmph/s
Bezpečné brždění	0,30÷0,35	10,6÷12,4
Průměrný řidič	0,47	16,6
Zručný řidič	0,62	21,9
Řidič profesionál	0,66	23,0
Technický limit automobilu	0,77÷1,00	24,8÷35,2

Tabulka 12 Hodnoty přetížení při rozjezdu z nulové rychlosti na 96,6 kmph, odpovídající 0,55 g [10]

Čas (s)	0	1	2	3	4	5
Rychlost (kmph)	0	19,3	38,6	57,9	77,2	96,6
Vzdálenost (m)	0	2,74	10,67	24,08	42,98	67,06

Dále byla testována funkce gyroskopu, jejímž úkolem mělo být stanovení úhlu, resp. % stoupání či klesání. K tomu by však bylo zapotřebí kontinuálně shromažďovat data z gyroskopu ve velice krátkých intervalech, tj. řádově ms nebo jejich zlomky, a tato data integrovat v čase pro obdržení informace o úhlu. Avšak vzhledem k podstatně delším dobám konverze např. u teplotních čidel (řádově stovky ms), není prakticky možné v reálném čase tato data sbírat a zpracovávat, aniž by došlo ke krátkodobému výpadku informačních toků. I sebemenší pauza ve sběru dat pak může způsobit poměrně citelnou chybu ve smyslu např. zpětného nenavrácení k nulové hodnotě v naprosté horizontální rovině shodné s tou, při inicializaci senzoru. Je to dáno také určitou nepřesností gyroskopu, jehož hodnoty

v úhlových jednotkách dps (degrees per second) v klidovém stavu kmitají kolem rovnovážné polohy a časovou integrací tak periodicky klesá a roste hodnota úhlu. Tuto chybu lze do jisté míry eliminovat použitím preciznějšího senzoru, např. IMU, které se používají do leteckých modelů. Alternativně lze také počítat okamžitý úhel natočení pomocí akcelerometru, jehož výsledky v klidu jsou poměrně přesné, nicméně za jízdy jsou velice proměnlivé až dokonce náhodné/nestabilní. Pro představu je níže uveden předpis pro výpočet úhlu stoupání (deg) z údajů akcelerometru:

$$pitch = atan2\left(acc_Z, copysign(acc_Y, acc_Y) * sqrt\left((0.01 * acc_X * acc_X) + (acc_Y * acc_Y)\right)\right) * 180/pi, \quad (2.4)$$

kde *pitch* (deg) je výsledný úhel stoupání; acc_X, acc_Y, acc_Z (m.s^{-1}) jsou velikosti zrychlení v příslušných souřadných osách; *atan2, copysign, sqrt* a *pi* jsou součástí matematické knihovny (*math*).

Před samotným měřením jak z akcelerometru, tak gyroskopu, by měla být provedena počáteční kalibrace. Pro měření relativních hodnot zrychlení pomocí akcelerometru je vhodné na úplném začátku naměřit dostatečný objem dat v krátkých časových intervalech, z něhož následně určit střední klidové hodnoty, které pak odečítat od výsledných hodnot. V případě telemetrického zařízení jsou klidové hodnoty z akcelerometru ještě filtrovány, kdy jsou vyloučeny všechny případné omyly, např. pokud by došlo k nečekanému krátkodobému otřesu během kalibrace, čímž by se negativně změnila střední hodnota. Veškerá kalibrační data jsou navíc ukládána do souboru na samotném zařízení. Oproti tomu knihovna pro použití gyroskopu již nativně obsahuje kalibrační funkci, kde jako parametry jsou zadávány počet cyklů a časový krok. Standardně platí, čím větší objem dat, tím přesnější výsledky.

2.1.1.6 Zjišťování nadmořské výšky pomocí barometru

Nadmořskou výšku lze přibližně s přesností ±1 m určit z hodnot teploty a tlaků. K tomu slouží digitální barometr označovaný jako *BMP180*, který je součástí kombinovaného senzoru *MPU925X*. S jeho pomocí lze měřit tlaky v rozsahu 300÷1100 hPa, odpovídající nadmořským výškám -500÷9000 m. K výpočtu je zapotřebí znát lokální barometrický tlak redukovaný na hladinu moře, aktuální výstup z tlakového senzoru a teplotu. Přepočet na nadmořskou výšku v metrech je dán vztahem:

$$altitude = -7990 \times ln(\frac{pressure}{baseline})\ (m), \quad (2.5)$$

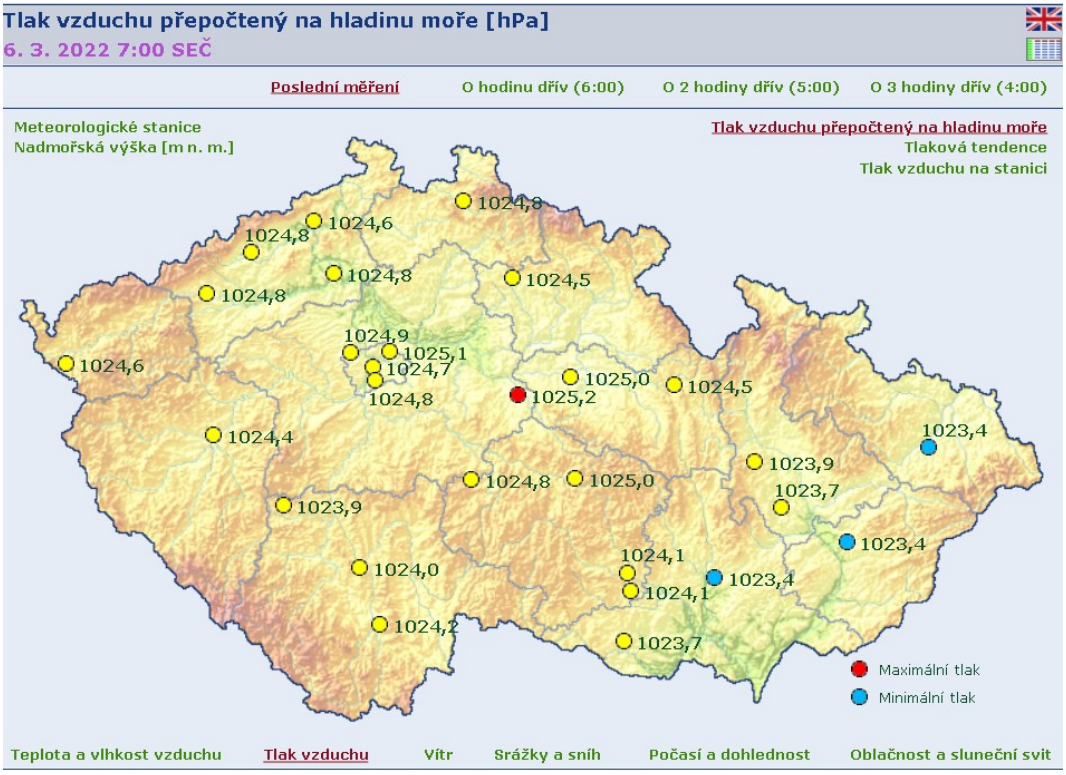
kde *pressure* (mbar) udává naměřený absolutní barometrický tlak, jehož stanovení zahrnuje i konverzi teploty; *baseline* (Pa) představuje lokální hodnotu barometrického tlaku redukovanou na hladinu moře, kterou udává např ČHMÚ.

Pozn.: Uvedený vzorec je vyňatý z knihovny pro modul *BMP180*.

Alternativně lze také snímat relativní změny tlaku, a ty následně přepočítávat na skutečnou nadmořskou výšku, jeli dostatečně přesně známa výška startovního místa/pozice.

Získanou nadmořskou výšku lze posléze porovnat s údajem z GPS přijímače, či online mapovými podklady, jsou-li známy přesné GPS souřadnice místa (viz https://api.mapy.cz/view?page=altitude).

Jak již bylo zmíněno prostřednictvím modulu lze měřit i teplotu, jejíž hodnotu lze brát pouze jako orientační, neboť např. při pokojové teplotě prostředí byla zjištěna teplota o několik stupňů Celsia vyšší než skutečná, navíc během provozu modulu teplota dokonce pomalu rostla, pravděpodobně z důvodu vznikajících ztrát na čipu.



Obr. 15 Mapa tlaků vzduchu přepočtených na hladinu moře [11]

2.1.1.7 Měření teploty digitálními teplotními čidly

Pro měření teploty jsou použita digitální teplotní čidla typu *DS18B20*. K telemetrickému zařízení jsou připojena celkem 2 tato čidla ve vodotěsném provedení a délkou přívodního kabelu 1 m, z něhož jsou vyvedeny celkem 3 piny. Dva piny slouží pro napájení a třetí je

datový. Důležitým specifikem je potřeba připojení externího rezistoru o hodnotě 4,7 kΩ mezi datový (žlutý) a napájecí (červený) pin. Při použití více čidel se uplatňuje jejich paralelní řazení, rezistor však zůstává jediný, společný pro všechny. Čtení teplot se následně realizuje dotazem na konkrétní, unikátní adresu čidla. Teplotní čidlo měří teploty v rozsahu -55÷125 °C a disponuje konfigurovatelným rozlišením 9 až 12 bitů, od kterého se pak odvíjí doba konverze. Dvojice čidel zabezpečuje měření teploty okolního prostředí a teploty baterie s uživatelsky definovaným rozlišením 9 bitů. V následující tabulce je možné pozorovat závislost požadovaného rozlišení na době konverze.

Tabulka 13 Závislost doby konverze na požadovaném rozlišení teplotních čidel DS18B20

Rozlišení (bits)	Max. doba konverze (ms)
9	112
10	196
11	364
12	700

2.1.1.8 Komunikace prostřednictvím Bluetooth

Bezdrátovou komunikaci s ovládacím softwarem zajišťuje externí modul typu *HC-06*, který odesílá, resp. přijímá data prostřednictvím sériové linky (UART). Modul bezproblémově funguje bez jakýchkoliv dodatečných nastavení. Výchozí modulační rychlost činí 9600 baudů, kterou lze případně změnit pomocí tzv. AT příkazů. Stejně tak je možné změnit název zařízení, PIN pro párování, pokročilé parametry sériového spojení, nebo obnovit jeho tovární nastavení. Jedinečnou MAC adresu však z principu editovat nelze. Ta pak slouží pro rychlou a jednoznačnou identifikaci telemetrického zařízení.

2.1.2 Zapojení jednotlivých modulů

Následující tabulka shrnuje způsoby napájení modulů a vazby na komunikační sběrnice.

Tabulka 14 Připojení jednotlivých modulů k mikrokontroléru (wiring)

Modul (sensor)	Typ komunikace (bus)	Piny (module ↔ board)
AMS1117 (3.3 V)	-	VIN ↔ VOUT (AMS 5V) GND ↔ GND VOUT ↔ V (ADS) GND ↔ G (ADS)
AMS1117 (5 V)	-	VIN ↔ BAT (+) GND ↔ BAT (−) VOUT ↔ 1N5819 (A) GND ↔ GND
D 1N5819 **(40 V/1 A)**	-	1N5819 (A) ↔ VIN (AMS 5V) 1N5819 (K) ↔ VSYS (39)
DS1302	-	VCC ↔ 3V3(OUT) GND ↔ GND CLK ↔ GP18 DAT ↔ GP16 RST ↔ GP17
ADS1115	I2C(1)	V ↔ VOUT (AMS 3.3V) G ↔ GND SCL ↔ GP15 SDA ↔ GP14 ADDR ↔ GND A0 ↔ DIV_OUT (+) A1 ↔ DIV_OUT (−) A2 ↔ Vout (HSTS) A3 ↔ Vref (HSTS)
DIV **(R428k ± 1 %,** **R10k ± 1 %)**	-	R428k (1) ↔ DIV_IN (+) R428k (2) ↔ DIV_OUT (+) R10k (1) ↔ DIV_OUT (+) R10k (2) ↔ DIV_IN/OUT (−)

HSTS016L (100 A ± 1 %)	-	+5V ↔ VOUT (AMS 5V) 0V ↔ GND Vout ↔ A2 (ADS) Vref ↔A3 (ADS)
MPU925X	I2C(0)	+3V3 ↔ 3V3(OUT) GND ↔ GND SCL ↔ GP9 SDA ↔GP8
DS18B20	1-Wire	VDD ↔ 3V3(OUT), R4k7 GND ↔ GND DATA ↔ GP22, R4k7
R4k7 ±1 %	-	(1) ↔ VDD (DS18) (2) ↔ DATA (DS18)
HC-06	UART(0)	VCC ↔ 3V3(OUT) GND ↔ GND RXD ↔ GP12 TXD ↔GP13

Situační a obvodové schéma jsou k dispozici v *Přílohách #4* a *#5*.

2.1.3 Řešení programových problémů

Jeden z nejzávažnějších problémů způsobovalo nasazení druhého výpočetního jádra pro účely vypořádání se s delšími konverzními časy u teplotních čidel. Ty by pak v ideálním případě nezpožďovaly práci s ostatními moduly. Bohužel po určité krátké době docházelo k samovolným pádům hlavního programu. V drtivé většině případů nebyly známy ani podrobnosti selhání, jelikož se RPi Pico stačilo odpojit od Thonny dříve, než proběhlo odeslání chybových hlášení. Řešením by mohlo být použití debuggovacích portů mikrokontroléru pro odladění programu, k tomu je zapotřebí např. připojené druhé RPi Pico. Z chyb, které se podařilo zachytit, se jednalo většinou o selhání na straně komunikace s moduly nebo výpočetní chyby v přidružených knihovnách. Podobně bylo vyzkoušeno i přesunutí veškerých operací s *MPU925X* a *ADS1115* do separátního vlákna, výsledky byly však obdobné.

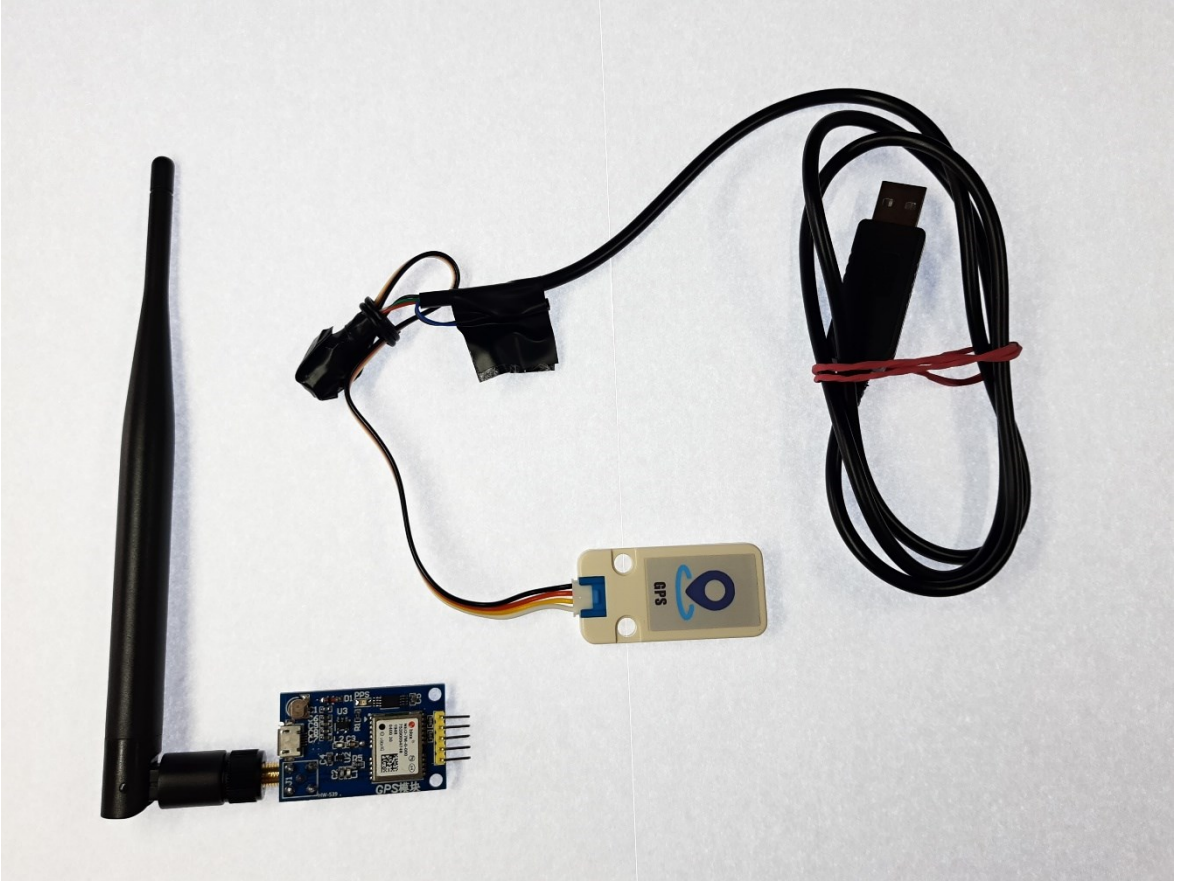
Dále některé z knihoven pro MicroPython využívaly softwarovou implementaci *machine.SoftI2C()*, kterou však nelze kombinovat s defaultní hardwarovou implementací

machine.I2C() od jiných modulů připojených na téže komunikační sběrnici. Řešením je editace inicializační části knihovny, kde je zapotřebí zaremovat řádek volající metodu *start()* odkazované hardwarové instance *I2C*. *machine.I2C()* tuto metodu nepodporuje.

Kromě firmwaru Micropython byl testován i CircuitPython, a to z důvodu rozsáhlého balíku knihoven. Nicméně některé knihovny nebylo možné pro CircuitPython dohledat, tudíž se i vzhledem k absenci podpory přerušení a podpory více vláken, nakonec přešlo přímo na MicroPython. Aby bylo možné stávající, ověřené knihovny nadále využívat, bylo potřeba doinstalovat knihovny zabezpečující kompatibilitu mezi těmito dvěma firmwary, tzv. *Blinka compatibility layer* (https://circuitpython.org/blinka/raspberry_pi_pico/). Tím se však za jistých okolností, tj. vlivem nadměrné hloubky vnořování CircuitPython funkcí, objevil další problém v podobě „maximum recursion depth exceeded“, kdy rekurzivní limit pro MicroPython je definován na 10. Z toho důvodu byly nakonec sjednoceny veškeré knihovny pro použití přímo pro MicroPython a tato chyba tak eliminována.

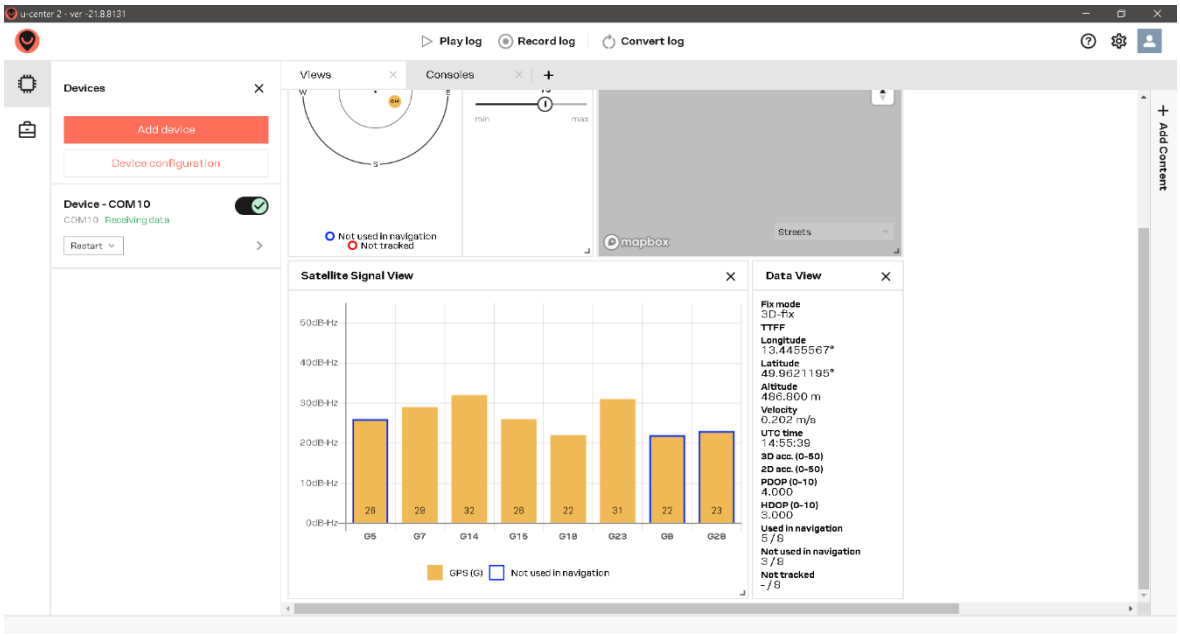
2.2 Geolokační modul

Geolokační nebo také GPS modul zprostředkovává data ze satelitních družic, vyhodnocuje a odesílá přes sériové rozhraní UART. Celkem byly vyzkoušeny 2 navigační čipy, tj. *Neo-7M* a *AT6558*, přičemž *Neo-7M* dokáže komunikovat pouze s *Navstar GPS*. V případě *AT6558*, který je součástí *M5Stack Mini GPS/BDS* modulu, lze komunikovat s dalšími globálními navigačními systémy: *Galileo*, *BeiDou* a *GLONASS*.



Obr. 16 Ukázka GPS modulů – Neo-7M (vlevo), M5Stack GPS Mini (vpravo) + TTL převodník

Z obou zmíněných modulů se nejlépe osvědčil *AT6558*, který bez problému chytal signál i v členité městské zástavbě, přesně podle specifik výrobce, a to s tzv. *Cold startem* do 35 s. Pro připojení k PC přes USB rozhraní bylo potřeba modul doplnit o převodník TTL na USB. Oproti tomu *Neo-7M* bylo možné zafixovat jen mimo vysoké budovy, na volném prostranství méně zatíženém rušením. Pozitivem zde však byla přítomnost microUSB portu přímo na desce modulu a stažitelná aplikace *u-center* výrobce *u-blox*, s pomocí níž lze modul konfigurovat, ale především sledovat v reálném čase data ze satelitů, počet družic, nechat zobrazit pozici na mapě, nastavit logování dat do souboru apod. Aplikace bohužel není kompatibilní s *AT6558*.



Obr. 17 Aplikace u-center pro vizualizaci GPS dat

Jelikož komunikace s moduly probíhá v normalizovaném textovém formátu *NMEA*, lze pro jednoduché ukládání dat použít např. aplikaci *PuTTY*. Každá přijatá zpráva zabírá přesně jeden řádek, jenž začíná znakem „$“ hned následovaným identifikátorem zprávy, např. *GPRMC*, *GPGGA*, *GPGSA* apod. Dále jsou zařazena jednotlivá data příslušející určitému druhu zprávy, navzájem oddělená znakem „čárka“. Zpráva je vždy ukončena znakem „hvězdička“ a dvěma dalšími znaky představující kontrolní součet, díky němuž lze ověřit správnost/integritu přijatých dat. Níže je uveden příklad sériové komunikace:

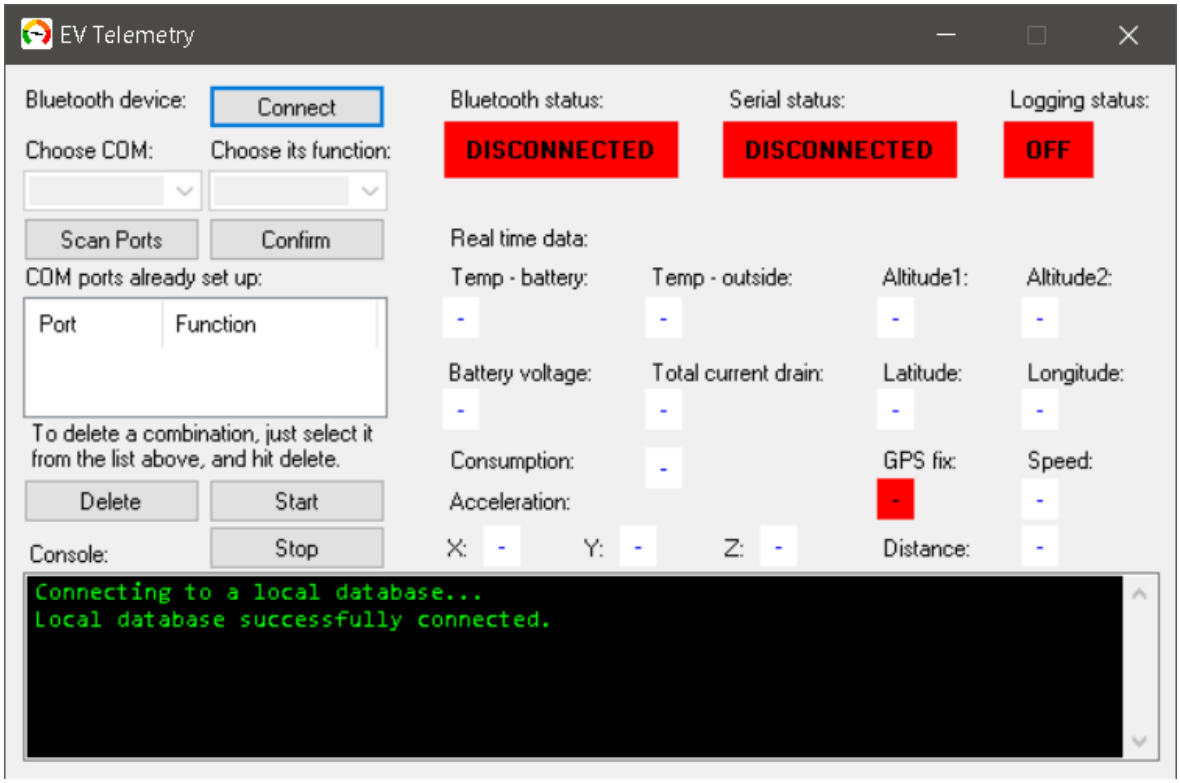
```
$GPRMC,121802.00,A,4943.44938,N,01322.03467,E,0.563,,080322,,,A*7F
$GPVTG,,T,,M,0.563,N,1.042,K,A*24
$GPGGA,121802.00,4943.44938,N,01322.03467,E,1,05,1.62,330.9,M,45.2,M,,*52
$GPGSA,A,3,32,12,02,29,22,,,,,,,,4.61,1.62,4.31*09
$GPGSV,2,1,08,02,39,101,26,11,33,099,26,12,65,071,21,22,31,295,30*70
$GPGSV,2,2,08,25,,,30,29,36,211,15,31,19,310,24,32,32,272,27*41
$GPGLL,4943.44938,N,01322.03467,E,121802.00,A,A*6D
```

Pro účely separování užitečných dat byla rovněž vytvořena konzolová aplikace pro Windows (viz *Příloha #6*), která je schopna převést textový soubor s *NMEA* zprávami do CSV tabulky, obsahující datum a čas, zeměpisnou šířku, délku, nadmořskou výšku, rychlost, stav fixace, ale také průběžně vyhodnocovat ujetou/ušlou vzdálenost.

Obr. 18 Konzolová aplikace pro separaci GPS dat

2.3 Datasběrný software

Pro účely obsluhy bezdrátového měřícího zařízení a GPS modulu byla vytvořena desktopová aplikace pro platformu Windows, která zabezpečuje příjem dat, jejich zpracování, vizualizaci a ukládání do databáze. Aplikace je navržena v jazyce *C#* a pro svůj běh vyžaduje sadu *.NET framework* min. verze 4.7.2.



Obr. 19 Ovládací desktopová aplikace

Po spuštění aplikace dojde nejprve k ověření konektivity s lokální SQL databází, výsledek je následně vypsán do konzole. Ověřuje se platnost konektorů, kdy je uvažováno, že databázové soubory se nachází ve stejné složce jako aktuální spustitelný soubor. Nejsou-

li tyto soubory přítomny, nebo nelze-li do nich zapisovat, není možné později aktivovat funkci logování.

S pomocí aplikace je možné se připojit (*Connect*) ke vzdálenému telemetrickému zařízení, kalibrovat jej, spustit a ukončit měření. Interní Bluetooth klient je nakonfigurován na statickou MAC adresu bezdrátového komunikačního modulu telemetrického zařízení. Kliknutím na tlačítko připojit dojde k vyhledání všech dostupných zařízení v bezprostředním okolí a vypsání jejich MAC adres do konzole. Pokud se mezi nimi nachází i cílová Bluetooth adresa, proces připojení je zahájen. Po úspěšném navázání spojení s cílovým zařízením se zavedou přijímací a odesílací vlákna pro periodickou obsluhu datového streamu. Každé z vláken obsahuje nekonečný cyklus s výkonnou částí a uspáním daného vlákna po stanovenou dobu, vždy po dokončení jednoho jeho cyklu. Uspáním je zajištěno snížení vytížení procesoru a zvolená doba trvání souvisí s měřící periodou, tak aby nedocházelo ke zpoždění sběru dat, potažmo jejich hromadění v přijímacím zásobníku. Informace o úspěšném připojení je uvedena v konzoli a viditelně po celou dobu připojení pod *Bluetooth status*.

Kromě bezdrátové konektivity disponuje aplikace obsluhou fyzického sériového komunikačního rozhraní. Prostřednictvím tlačíka *Scan Ports* se provede vyhledání dostupných USB zařízení, která přidá do seznamu (*Choose ports*). Následně je možné se znalostí čísla virtuálního portu UART převodníku připojeného GPS modulu přiřadit tomuto portu jeho funkci (*Choose its function = UART*). Následně tlačítkem *Confirm* se ověří a potvrdí aktuálně zvolená kombinace čísla portu s jeho funkcí. Seznam platných kombinací je veden pod *COM ports already set up*. Pokud již bylo předtím úspěšně navázáno Bluetooth připojení, po vyhledání portů dojde také k automatickému přidání čísla virtuální portu vyhrazenému Bluetooth do tohoto seznamu.

Jakmile jsou úspěšně nastaveny příslušné komunikační kanály, je možné s pomocí tlačítka *Start* navázat komunikaci přes sériové rozhraní, současně se vyšle povel skrze Bluetooth pro zahájení kalibrace telemetrického zařízení a následnému spuštění měření. Sériová komunikace je pouze jednosměrná, tzn. je uvažováno pouze čtení dat z GPS modulu. Informace o jejím stavu je opět uvedena v konzoli a vizuálně pod *Serial status*. V konzoli se dále zobrazí informace o nastavení času, probíhající kalibraci čidel a zahájení měření. Probíhající měření je mj. patrné z aktualizovaných hodnot sledovaných veličin.

Jsou-li přijímána platná data, je možné zapnout jejich ukládání do databáze, stačí kliknout na *Logging status – OFF*, čímž se zahájí ukládání dat do příslušných databázových tabulek. GPS data a údaje z čidel telemetrického zařízení jsou ukládána do samostatných tabulek.

Zvlášť jsou ještě ukládána statistická data z údajů čidel. Primárním klíčem je ve všech případech časové razítko. Konkrétní konfigurace databázových tabulek je uvedena níže:

	Name	Data Type	Allow Nulls	Default
🔑	cas_razitko	nchar(30)	☐	
	zem_sirka	float	☑	
	zem_delka	float	☑	
	nadm_vyska	float	☑	
	rychlost	float	☑	
	gps_stav	nchar(15)	☐	
	vzdalenost	float	☑	

Obr. 20 Konfigurace datové tabulky *Gps* pro údaje z GPS

	Name	Data Type	Allow Nulls	Default
🔑	cas_razitko	nchar(30)	☐	
	teplota_okoli	float	☑	
	teplota_baterie	float	☑	
	voltu_svorky	float	☑	
	proud	float	☑	
	spotreba	float	☑	
	zrychleni_x	float	☑	
	zrychleni_y	float	☑	
	zrychleni_z	float	☑	
	vyska	float	☑	

Obr. 21 Konfigurace datové tabulky *Sensors1* pro primární údaje z čidel

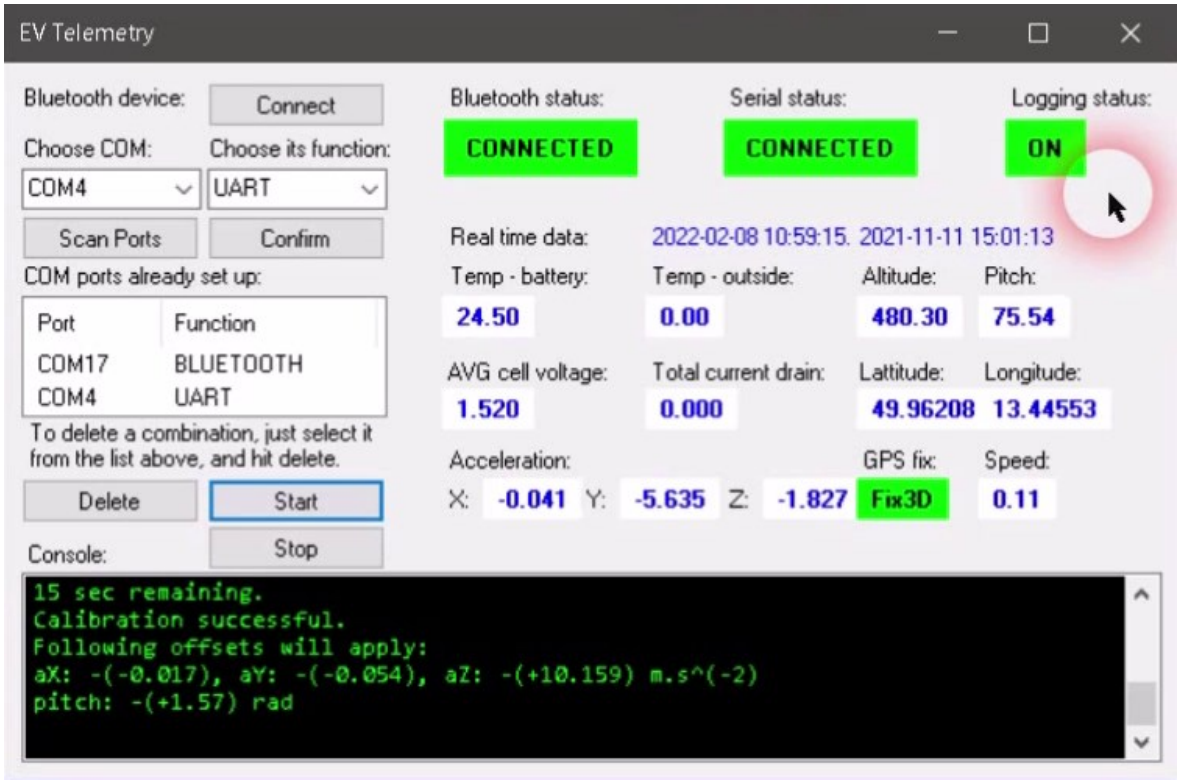
	Name	Data Type	Allow Nulls	Default
🔑	cas_razitko	nchar(30)	☐	
	voltu_min	float	☑	
	voltu_max	float	☑	
	amper_min	float	☑	
	amper_max	float	☑	
	zrych_x_min	float	☑	
	zrych_x_max	float	☑	
	zrych_y_min	float	☑	
	zrych_y_max	float	☑	
	zrych_z_min	float	☑	
	zrych_z_max	float	☑	

Obr. 22 Konfigurace datové tabulky *Sensors2* pro statistické údaje z čidel

Původně byl pro časová razítka používán datový typ *datetime*, bohužel z důvodu výskytu problémů při exportu tabulek, kdy byla vždy zahozena ms část, se nakonec přešlo na čistě textový formát. K tabulkovým datům lze přistupovat pomocí Visual Studia, a to prostřednictvím panelu *Server Explorer*, kde je následně možné přidat spojení (*Connect to Database*) s lokálním databázovým souborem. Podrobný návod je dostupný na oficiálních stránkách nápovědy [19]. Ukončení zápisu dat do databáze lze provést opětovným klinutím

na stav. Po úspěšném propojení databázového souboru s Visual Studiem a připojením k databázovému serveru, je možné procházet jednotlivé tabulky. Pravým kliknutím na zvolenou tabulku se pak nabízí možnost *Show Table Data*. Následně se otevře tabulka s daty, která je však ve výchozím stavu limitována na pouhých 1000 řádků. Pro zobrazení všech řádků stačí v ovládacích prvcích nad tabulkou přepnout *Max Rows* na *All*. Případný export hodnot lze realizovat kliknutím na tlačítko *Script to File*, rovněž těsně nad tabulkou. Takto je vygenerován SQL soubor, a to včetně příslušných příkazů. V běžném textovém editoru lze pak s pomocí opětovného použití elementární funkce *Find and Replace* separovat jednotlivá data od příkazů.

Pro operaci s Bluetooth a separaci GPS dat byly použity dodatečné knihovny v podobě NuGet balíčků, tj. *InTheHand.Net.Bluetooth* a *NmeaParser*. Knihovna *NmeaParser* byla navíc uživatelsky upravena, konkrétně část parsující čas, která vyvolávala výjimky kvůli přítomnému desetinnému formátu času v příslušných NMEA zprávách.



Obr. 23 Příklad stavu (starší verze) aplikace během provozu

Zastavení měření se realizuje pomocí tlačítka *Stop*, které pošle telemetrickému zařízení informaci o ukončení, dále ověří časová razítka aktuálně zpracovávaných dat, jestli nedošlo ke zpoždění jejich čtení ze zásobníku, a finálně přeruší ukládání do databáze. Opětovné případné spuštění již nelze provést pouze stisknutím tlačítka *Start*, ale je potřeba resetovat (odpojit a připojit napájení RPi Pico). Tímto však dojde i k rozvázání Bluetooth spojení.

Po zavření aplikace se automaticky uvolní všechny využívané virtuální porty a zavře se spojení s databázovým serverem.

Zdrojový kód aplikace je dostupný v *Příloze #7*.

3 Subjekt měření

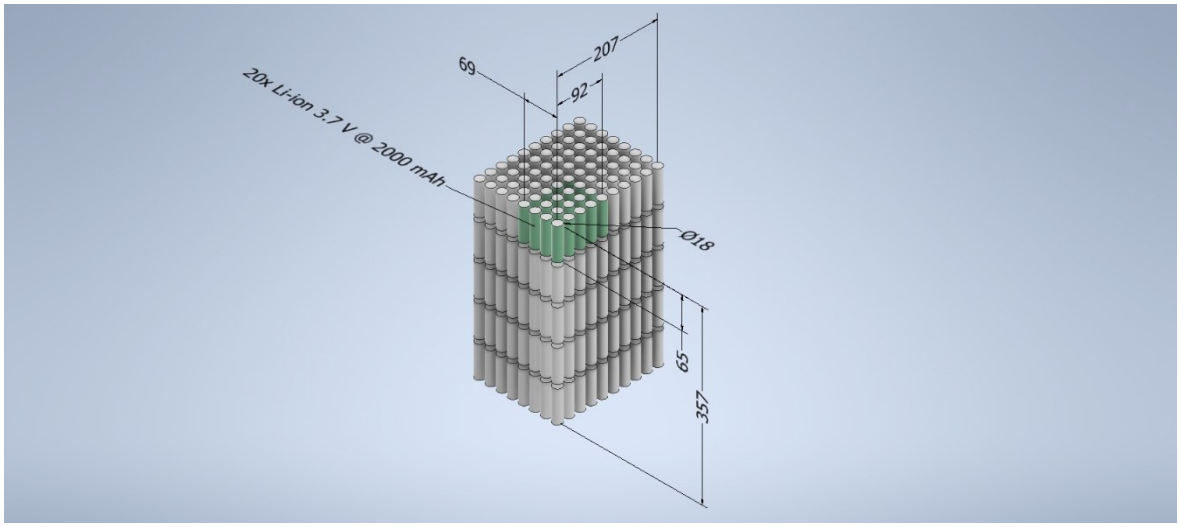
Subjektem měření je elektrické tříkolové vozidlo – tříkolka – od značky *elBlesk*. Jedná se o elektromobil poháněný bezúdržbovým BLDC motorem o jmenovitém výkonu 2,5 kW, jehož prostřednictvím lze dosáhnout rychlosti až cca 50 km/h. Pro účely napájení lze použít několik druhů akumulátorů, např.:

- VRLA olověný akumulátor 6-EVF-45,
- Lithium-iontový LCR18650-2000,
- Lithium-iontový INR18650-30Q,
- nebo LiFePO4 akumulátor.

Obr. 24 Elektrická tříkolka elBlesk

V případě olověného akumulátoru je napájení realizováno sériovým spojením 6×12 V jednotek. U lithiových typů se vždy jedná o sério-paralelní řazení jednotlivých cylindrických akumulátorů v pouzdru 18650 a kapacitě řádově 2000÷3000 mAh. Výsledná baterie pak disponuje kapacitou cca 2,8 nebo 5,6 kWh. Dle [12] by maximální dojezd na jedno plné dobití měl činit 70, resp. 140 km (v případě 5,6 kWh baterie).

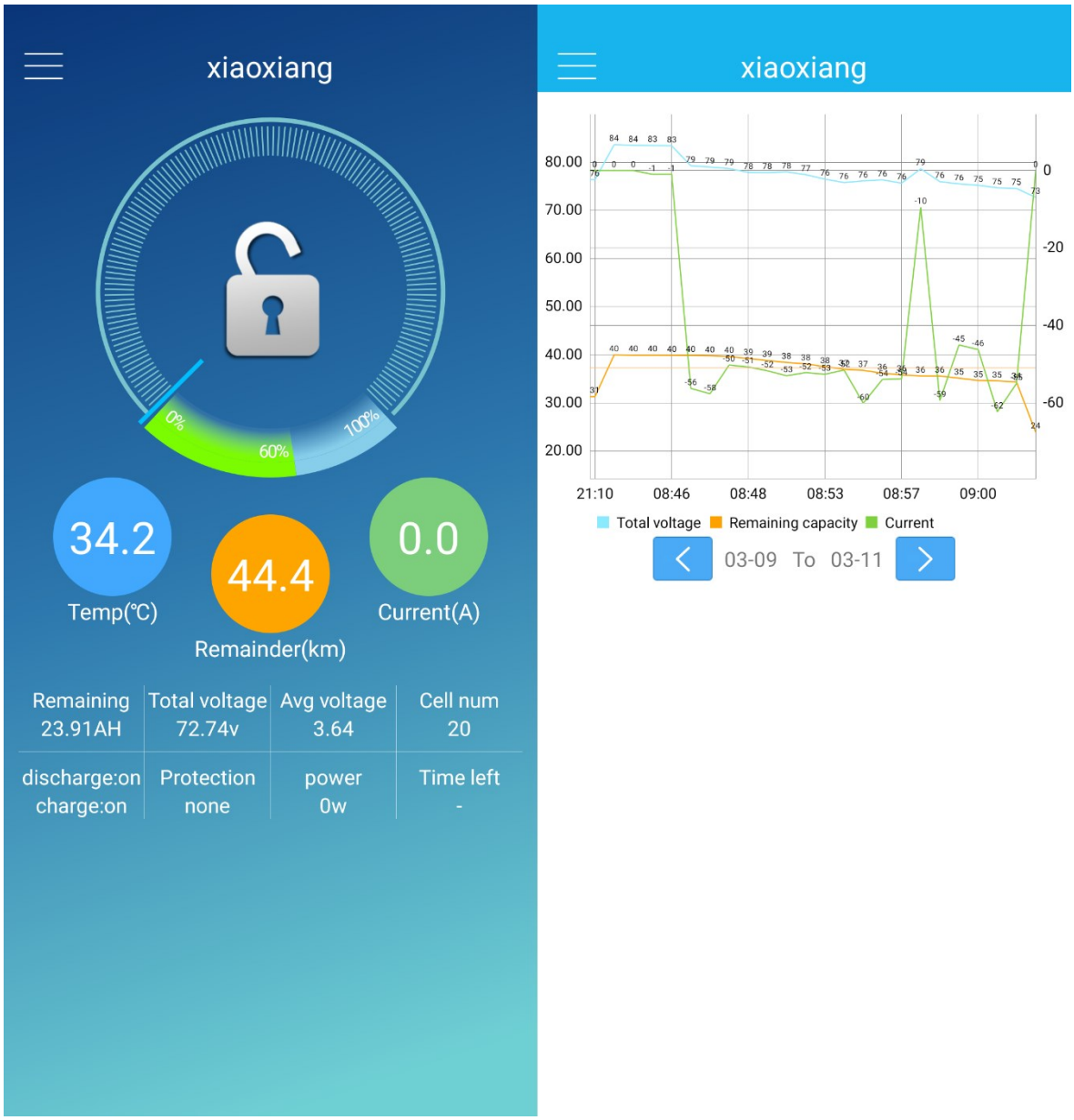
Obr. 25 Boční pohled na box bateriového systému



Obr. 26 Orientační (možné) uskupení baterie, celkem 400×18650 Li-ion článků (40 A.h)

Jelikož jednotlivé články mohou být během provozu různě zatěžovány, je součástí bateriového systému ještě BMS (Battery Management System) modul. BMS je aktivního typu, tzn., že sjednocování napěťových stavů článků se provádí na základě regulace odběru,

nikoliv mařením nadbytečné energie, tzv. shuntováním do odporu, jako u pasivních systémů. Mimo jiné BMS dále zastává funkce nadproudové, zkratové, tepelné ochrany a hlídá stav baterie. Konfigurace systémových parametrů je možná s pomocí BT mobilní aplikace, kde je možné v reálném čase také pozorovat napětí, vybíjecí/nabíjecí proud, teploty apod.



Obr. 27 Mobilní aplikace k BMS, evaluace dat

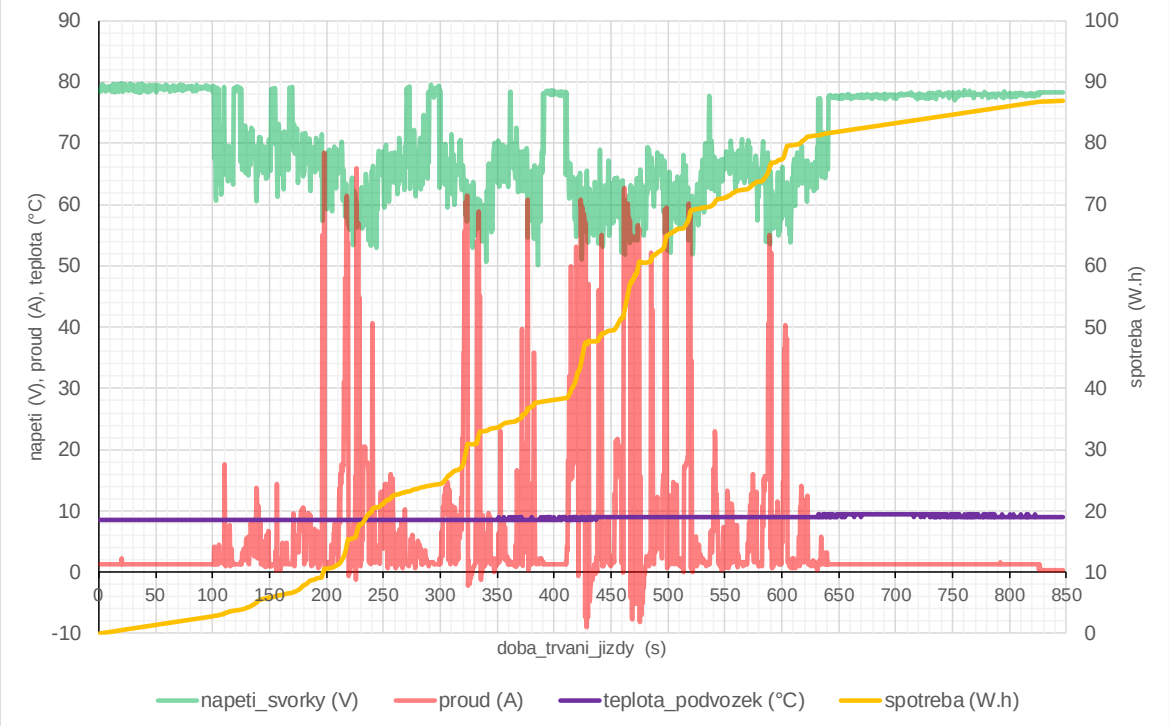
Kromě pohonu způsobují dodatečný odběr např. osvětlení a systém vytápění/větrání. Je potřeba také počítat s určitými (spínacími) ztrátami v řídící jednotce/měniči.

4 Data z testovacích jízd

V průběhu vypracování práce proběhly celkem 4 testovací jízdy, během nichž současně docházelo k opravám a vylepšením funkce telemetrického zařízení a hlavního ovládacího programu. Obdržená data z jednotlivých jízd byla následně zpracována dodatečnými skripty v jazyce Python. Proběhly tak např. úpravy časových razítek, spotřeby, nadmořské výšky a finální propojení dat ze senzorů a GPS modulu, díky čemuž bylo možné vypočítat další charakteristické parametry, např. stoupání.

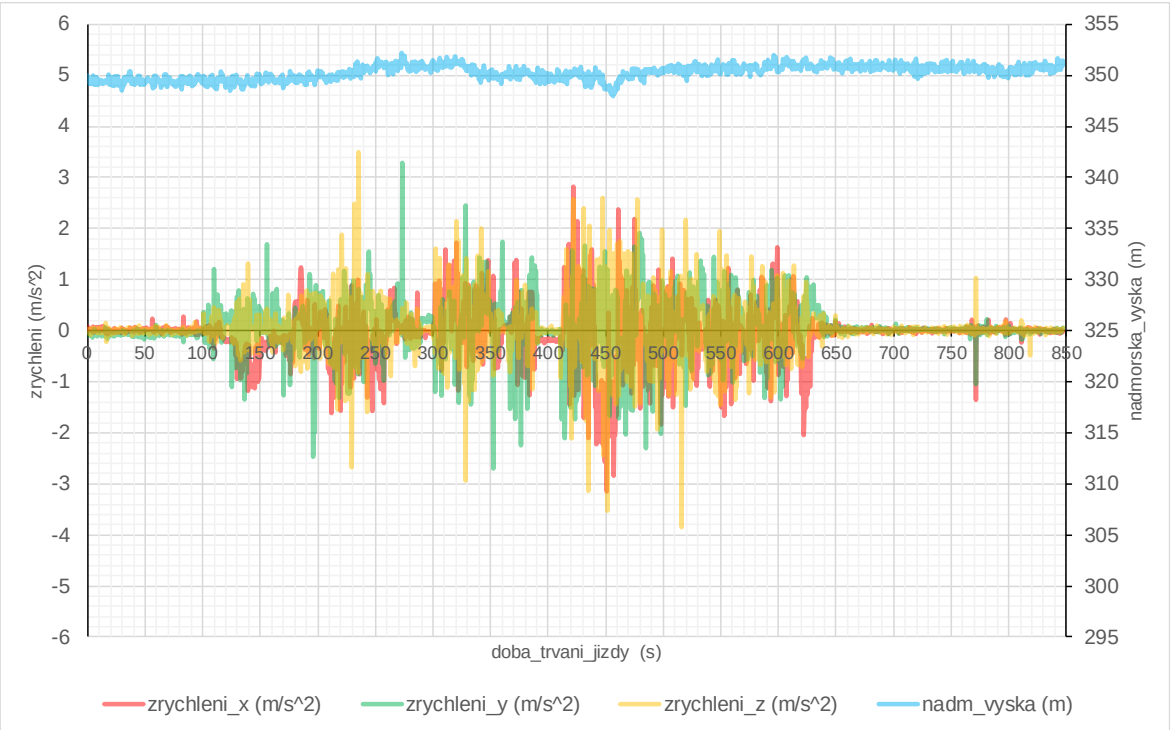
4.1 Testovací jízda #1

První testovací jízda proběhla bez použití GPS modulu (*Neo-7M*), u něhož se v městské zástavbě nepovedlo dosáhnout fixování, z tohoto důvodu byl pro následující jízdy nahrazen modulem *AT6558* s podporou více satelitních systémů. Opatření časovým razítkem probíhalo jeho přímým čtením z externího RTC modulu a umělým doplnění o dodatečné rozlišení na ms. Později se ukázalo, že tento postup vede k nadměrnému vytížení RTC modulu s nepříznivým vlivem na jeho napájení. Dále z původně zamýšleného měření teploty baterie bylo nakonec ustoupeno, a jedinou snímanou hodnotou se stala teplota okolí.



Obr. 28 Vizualizace průběhů napětí, proudu, teploty okolí a spotřeby během testovací jízdy

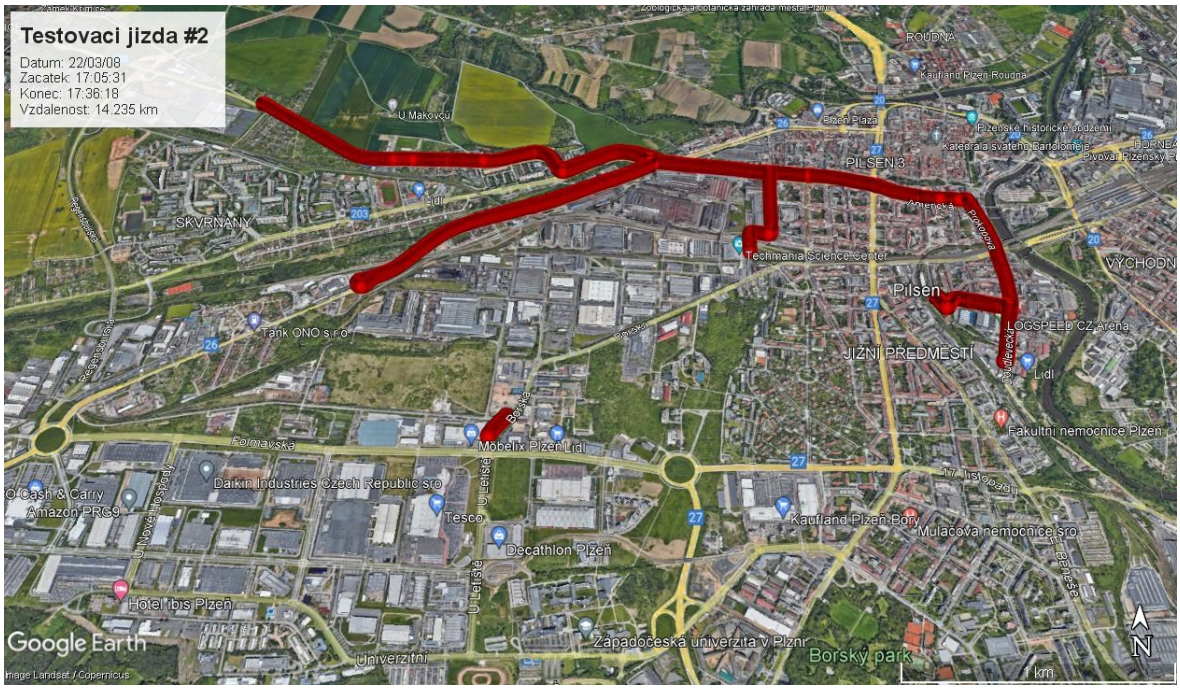
Obr. 29 Vizualizace průběhů napětí a proudu v závislosti na stavu nabití baterie



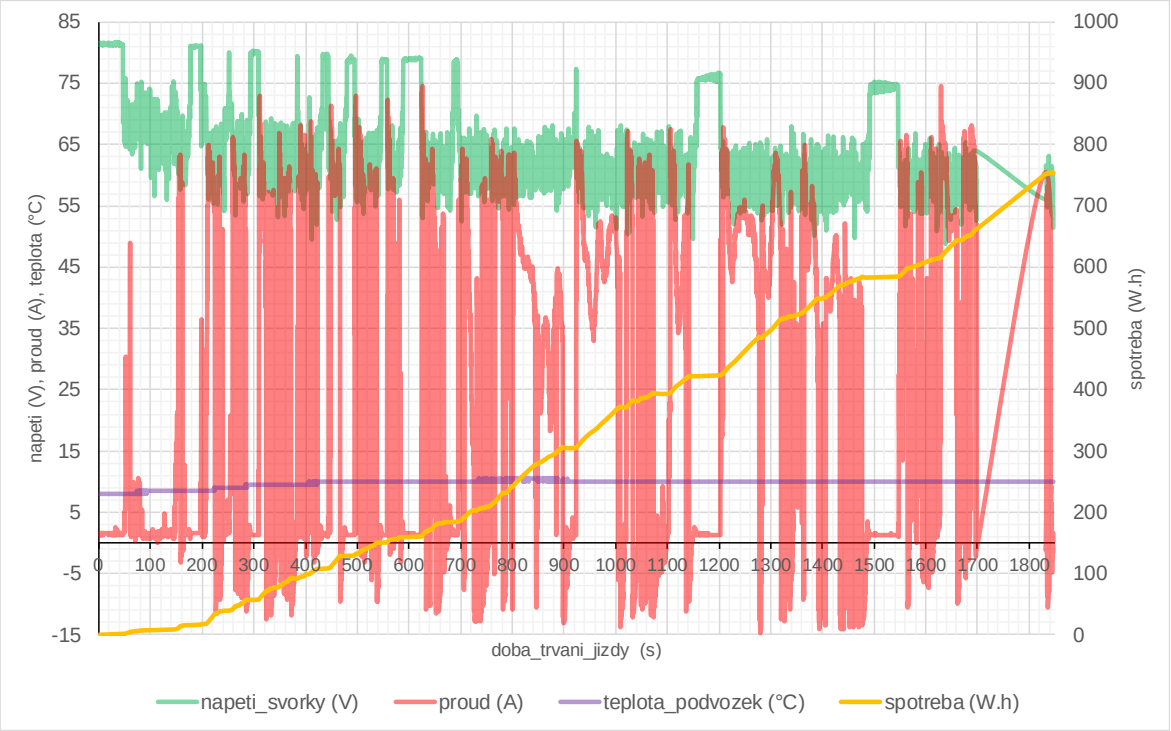
Obr. 30 Vizualizace hodnot zrychlení v jednotlivých osách a nadmořské výšky během testovací jízdy

4.2 Testovací jízda #2

Během druhé testovací jízdy byly podobně jako u první jízdy periodicky shromažďovány okamžité hodnoty provozních veličin. Nicméně časové razítko bylo při inicializaci telemetrického zařízení načteno přímo do paměti mikropočítače. GPS data byla úspěšně zprostředkována s využitím nového geolokačního modulu, avšak vlivem zpožděného čtení ze sériového portu byly některé GPS údaje, tj. v závěru testovacího okruhu, zahozeny (cca posledních 15 minut). S využitím mobilní aplikace bylo možné dodatečně zjistit vnitřní hodnoty teploty baterie a sledovaných elektrických veličin ze záznamu z BMS a porovnat je s externím měřícím systémem.

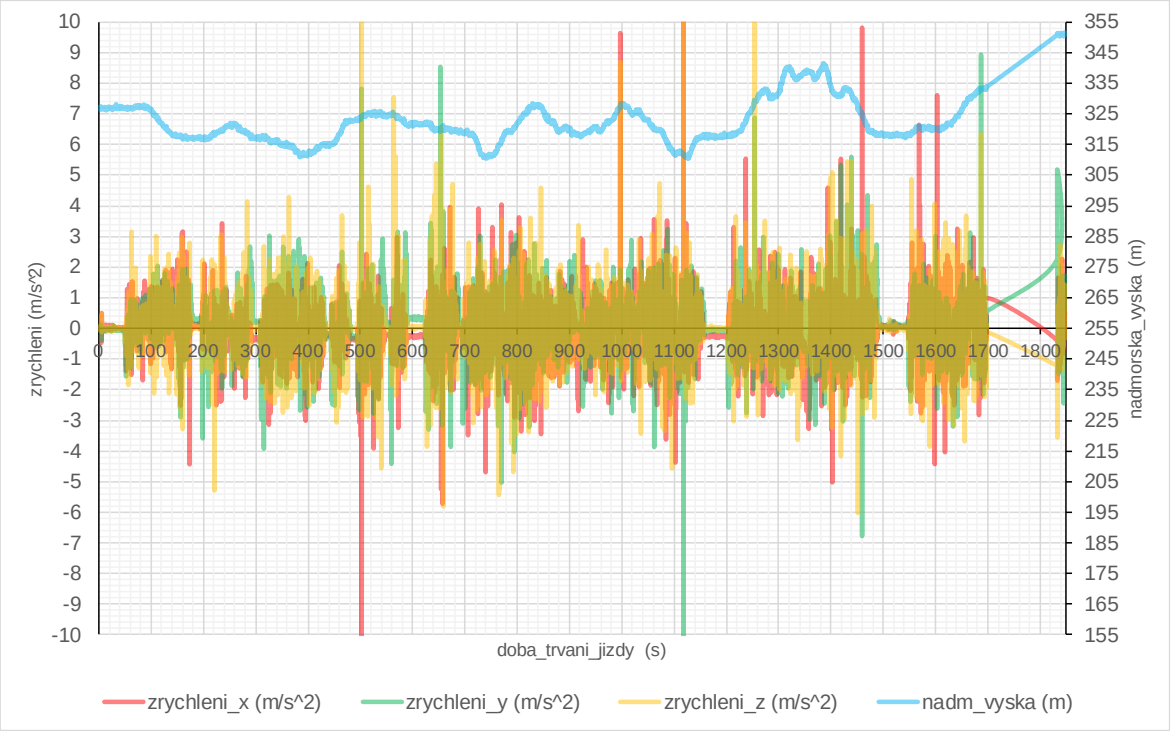


Obr. 31 Trasa testovací jízdy #2

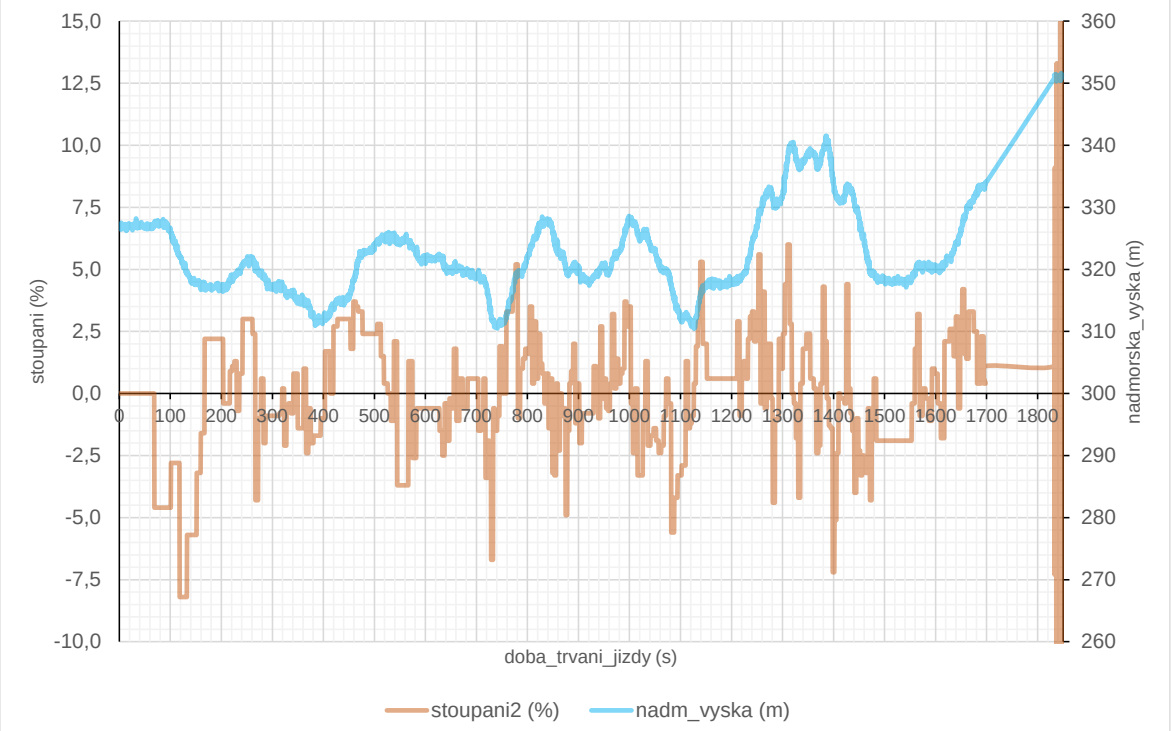


Obr. 32 Vizualizace průběhů napětí, proudu, teploty okolí a spotřeby během testovací jízdy

Obr. 33 Vizualizace průběhů napětí, proudu a rychlosti v závislosti na stavu nabití baterie



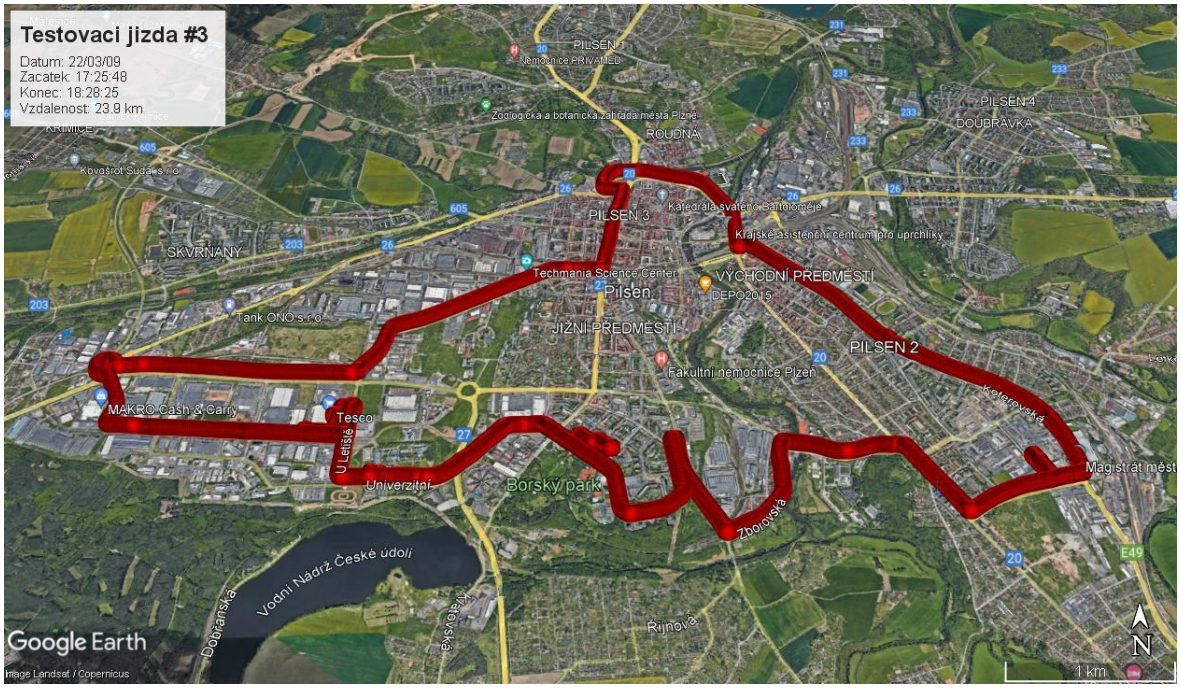
Obr. 34 Vizualizace hodnot zrychlení v jednotlivých osách a nadmořské výšky během testovací jízdy



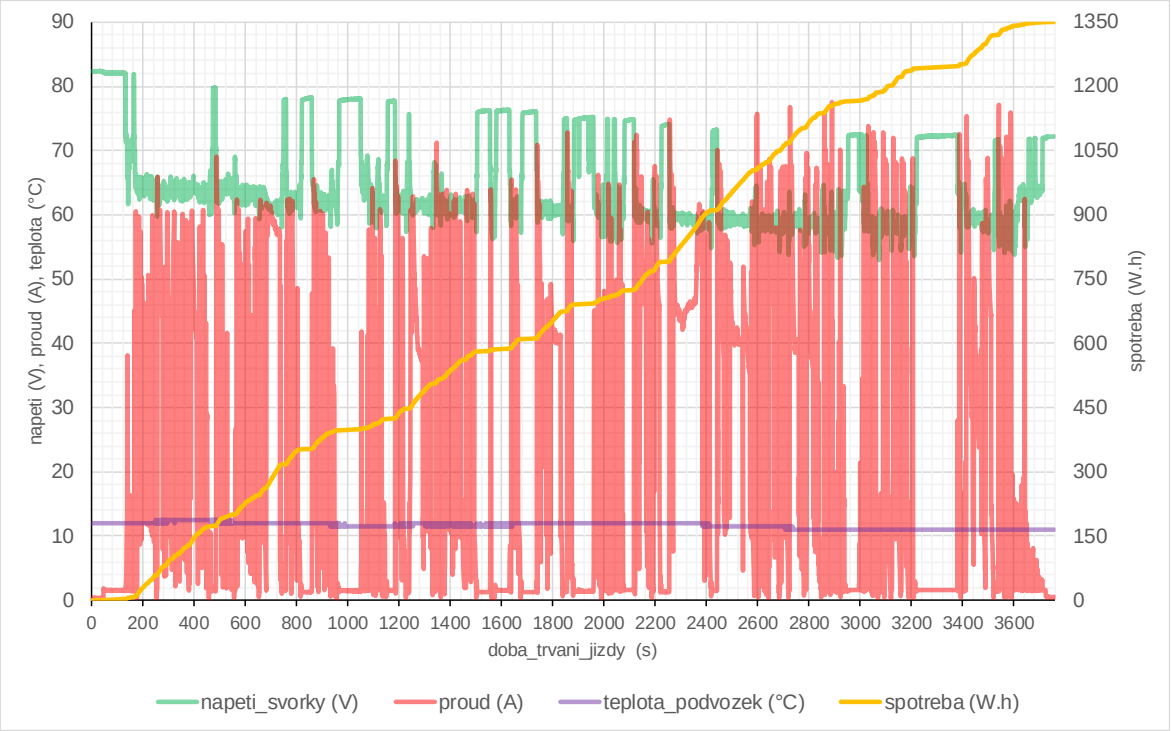
Obr. 35 Vizualizace hodnot stoupani a nadmořské výšky během testovací jízdy

4.3 Testovací jízda #3

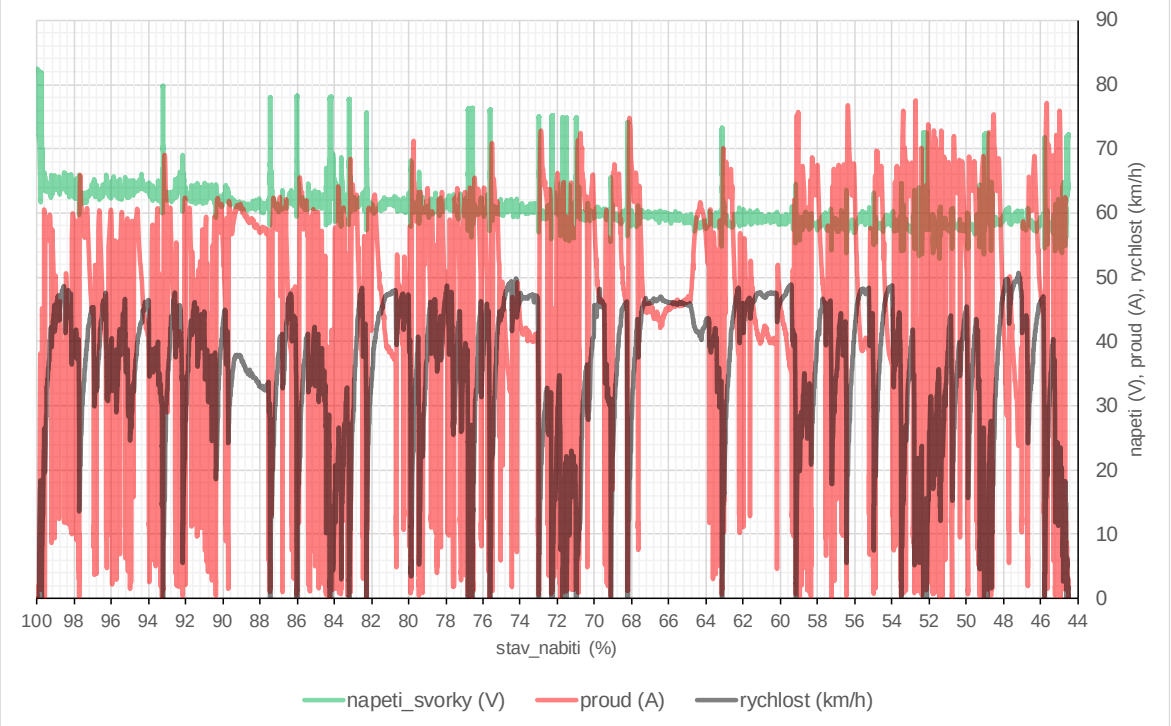
U třetí testovací jízdy byl poprvé nasazen režim sběru efektivních hodnot (TrueRMS) provozních veličin v definovaných časových intervalech, který vedl na lépe vypovídající zachycení chování bateriového systému při jeho zatížení, a to vzhledem k jeho pulsnímu charakteru. Efektivní hodnoty byly vyhodnocovány z ms vzorků provozních veličin za dobu jedné periody měření, tj. cca 500 ms. Měření provozních veličin opět probíhalo v definovaných intervalech, mezi nimiž byl vždy vymezen 100 ms úsek pro odečet hodnoty teploty, během kterého nebylo technicky možné (bez použití jiného výpočetního jádra) sledovat vývoj napětí a proudu. Pro minimalizaci vlivu na výslednou odebranou energii, bylo při jejím výpočtu předpokládáno, že během tohoto relativně krátkého úseku nedojde k žádným výrazným změnám efektivních hodnot sledovaných veličin, a tedy uvažován konstantní výkon multiplikovaný prodlouženou periodou. Mimo jiné bylo opraveno zpoždění ve zpracování GPS sériových dat. Po zběžné analýze přijatých NMEA zpráv bylo navíc potřeba upravit algoritmus pro separování vybraných GPS dat, jelikož oproti původnímu *Neo-7M* modulu bylo zjištěno jiné pořadí NMEA zpráv v rámci 1 s intervalu.



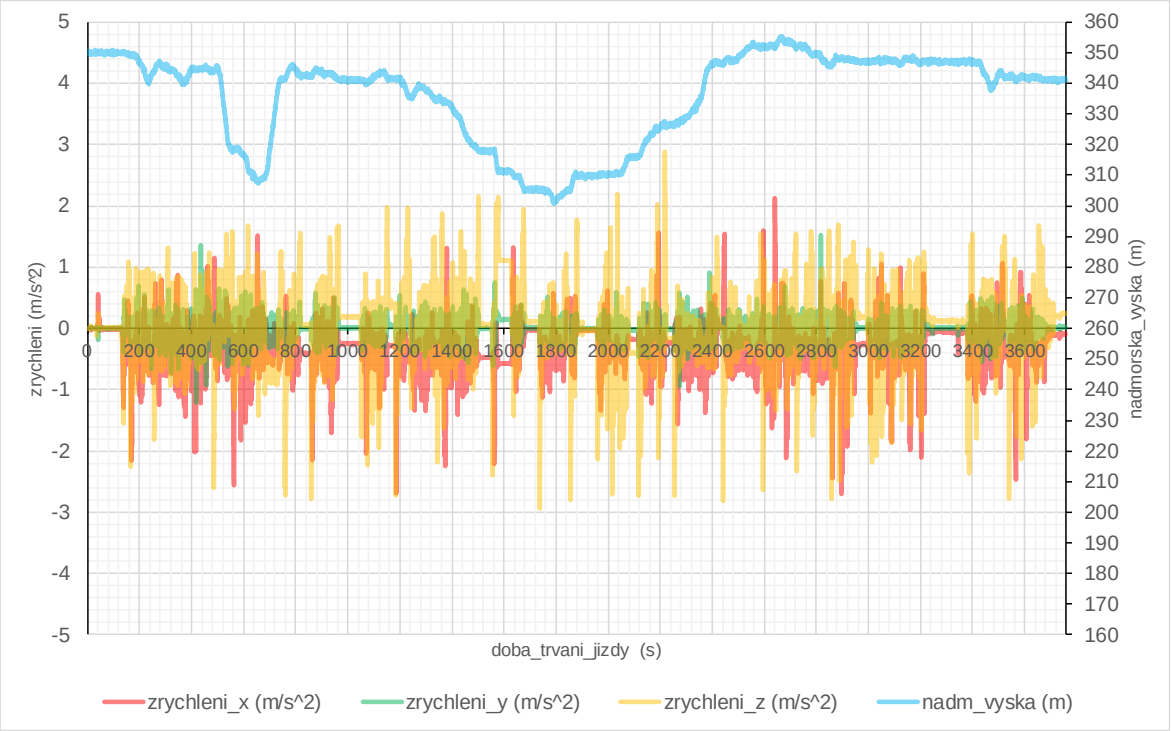
Obr. 36 Trasa testovací jízdy #3



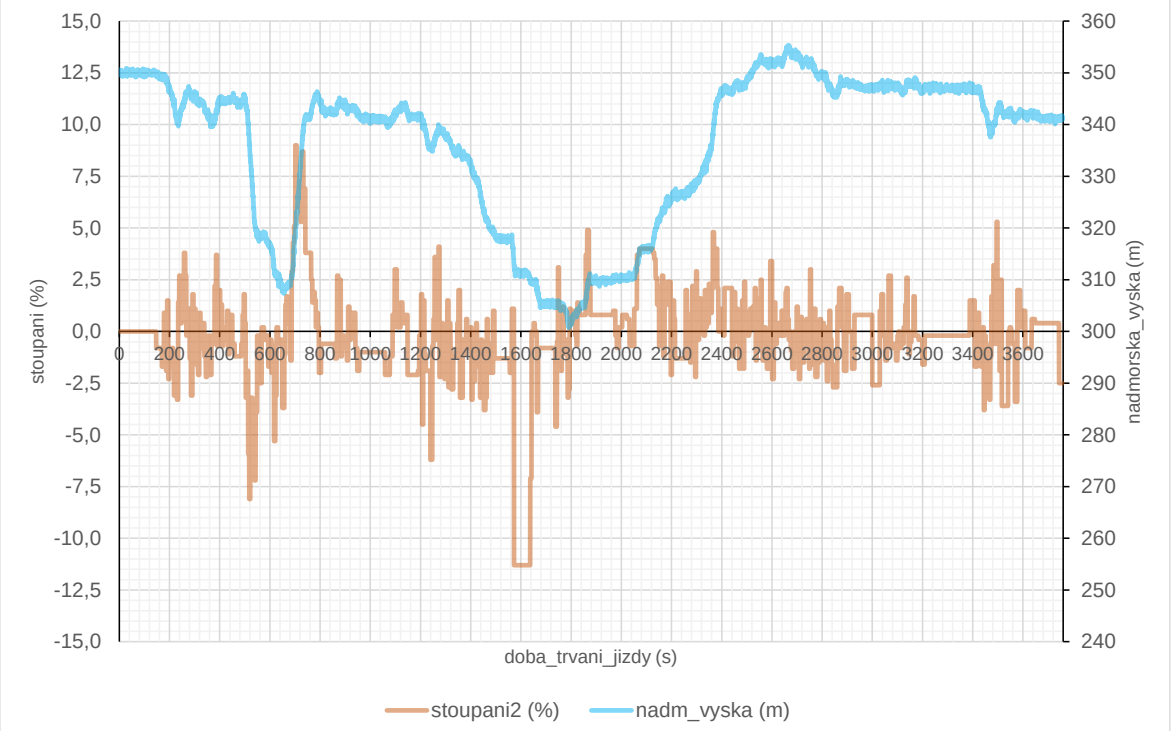
Obr. 37 Vizualizace průběhů napětí, proudu, teploty okolí a spotřeby během testovací jízdy



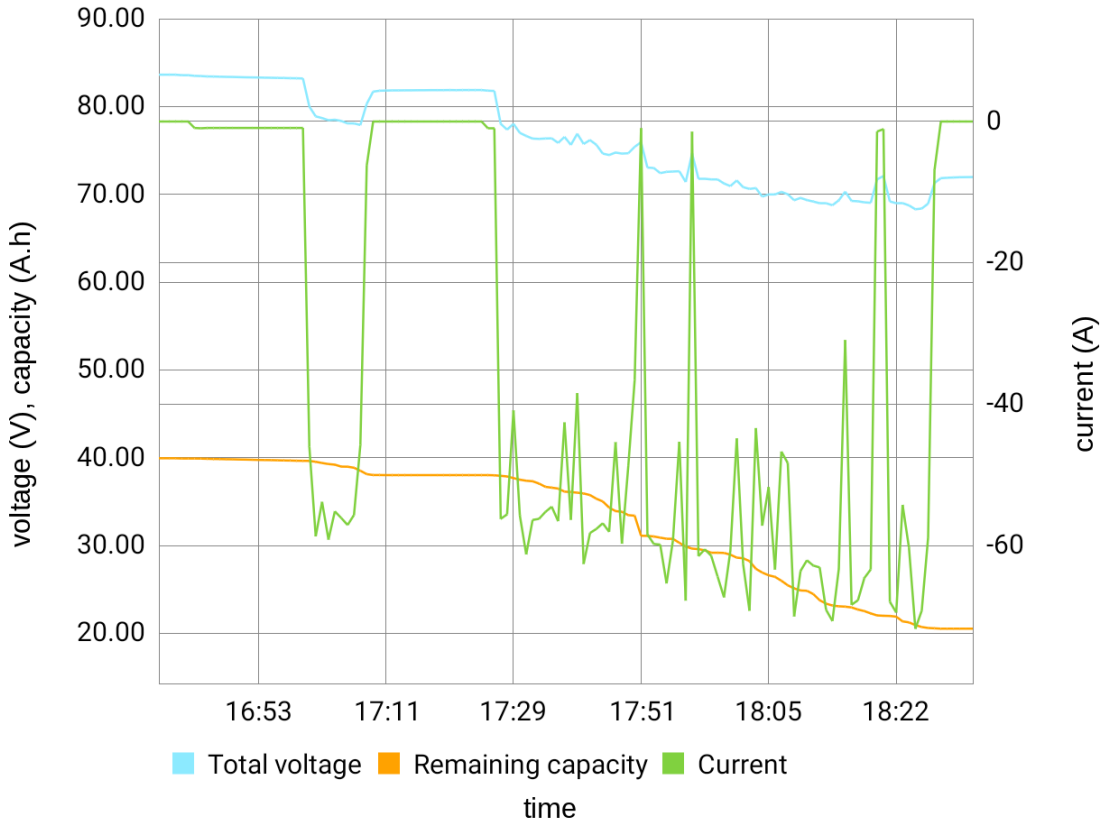
Obr. 38 Vizualizace průběhů napětí, proudu a rychlosti v závislosti na stavu nabití baterie



Obr. 39 Vizualizace hodnot zrychlení v jednotlivých osách a nadmořské výšky během testovací jízdy



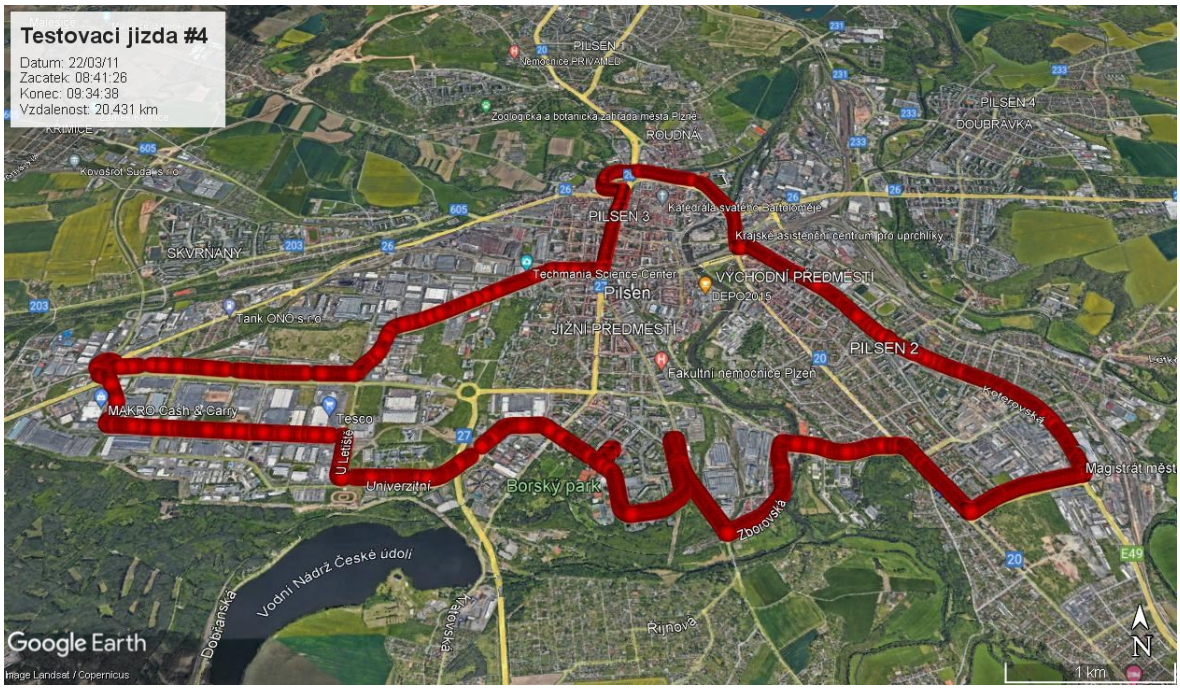
Obr. 40 Vizualizace hodnot stoupani a nadmořské výšky během testovací jízdy



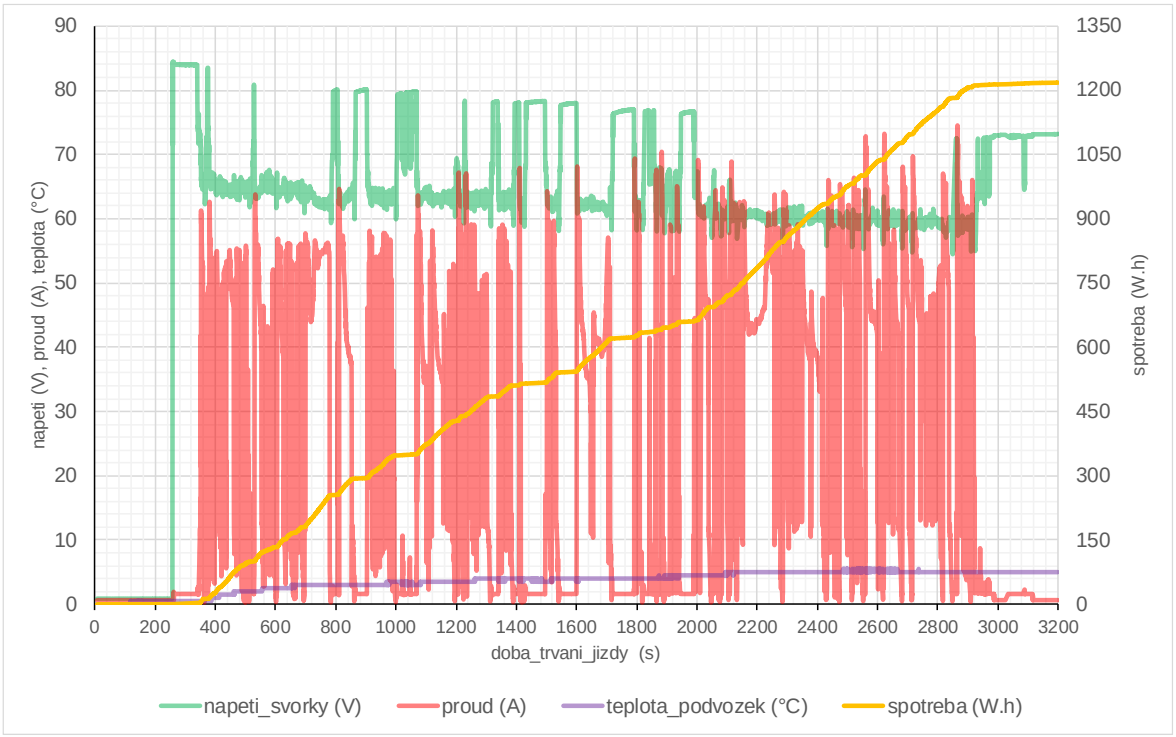
Obr. 41 Data zaznamenaná prostřednictvím BMS aplikace

4.4 Testovací jízda #4

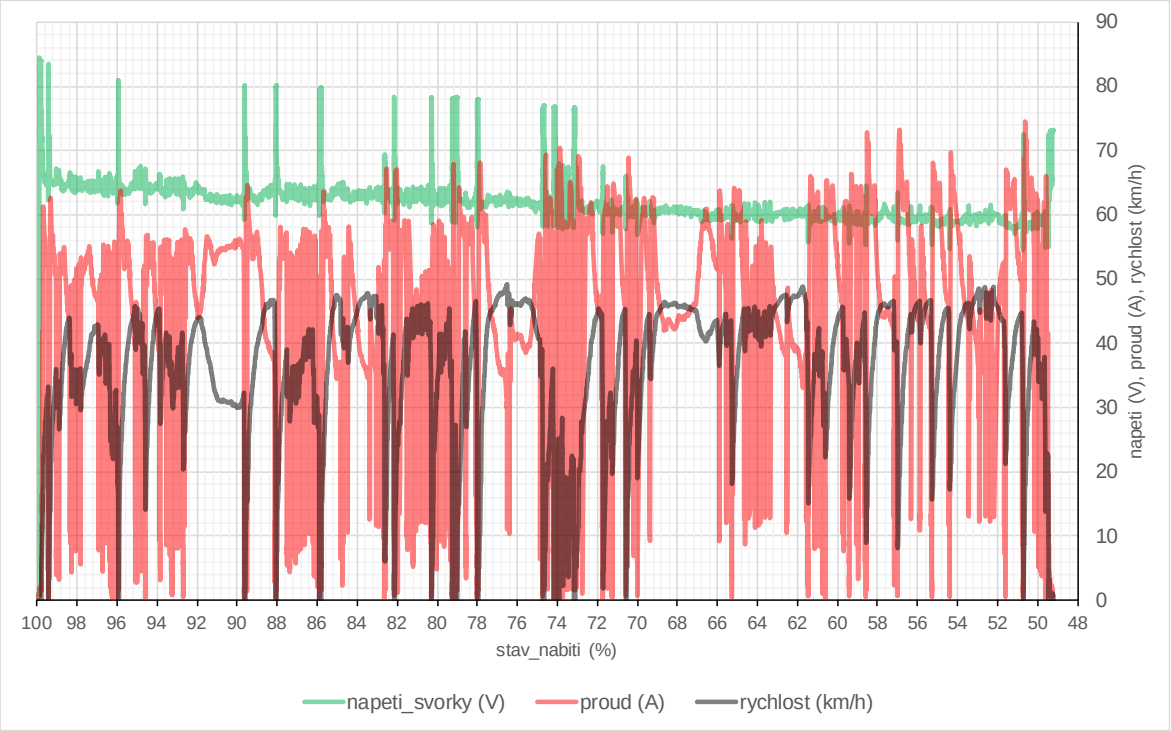
Čtvrtá testovací jízda přinesla po posledních úpravách nejvěrohodnější soubor dat, jak ze senzorů, tak i GPS modulu. Záznam ze senzorů byl doplněn o další statistická data, tj. min, max, střední průměr; napětí, proudu a údajů z akcelerometru. Navíc byla přidána kompenzace nadmořské výšky ze znalosti mapových podkladů startovacího místa, a následného sledování relativních změn barometrického tlaku promítajících se do změn výšky. Vlivem průběžného vývoje tlaku během dne však nelze stále počítat s absolutně přesnými hodnotami.



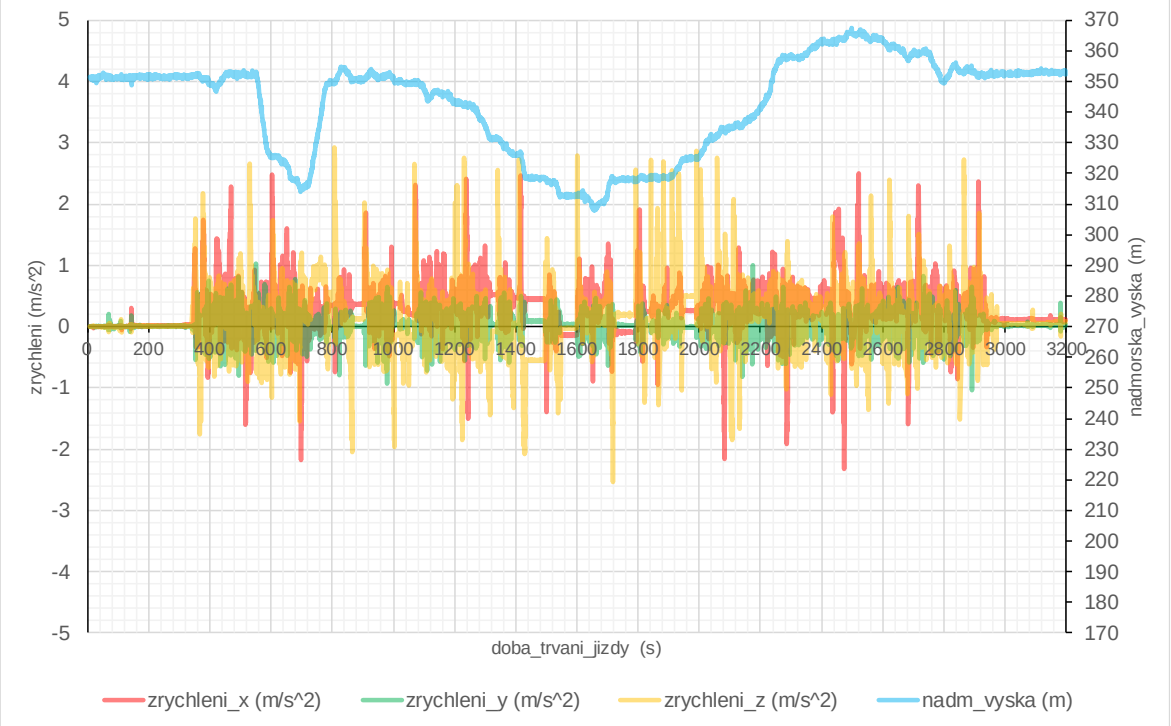
Obr. 42 Trasa testovací jízdy #4



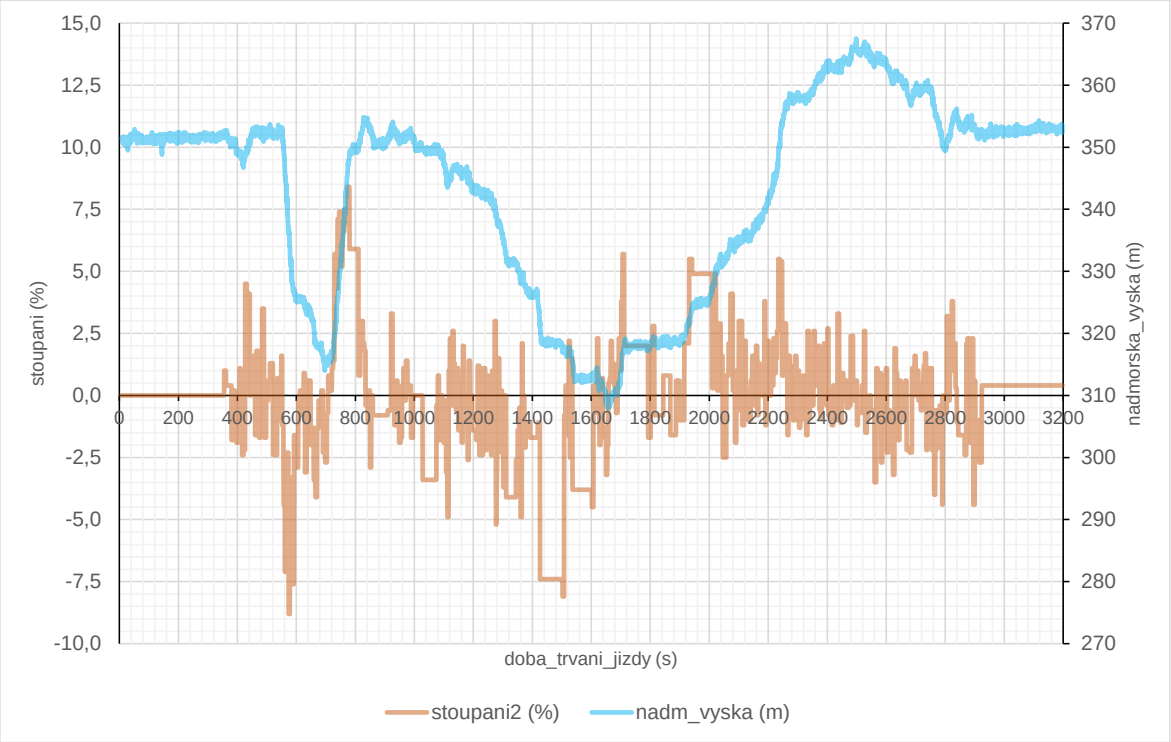
Obr. 43 Vizualizace průběhů napětí, proudu, teploty okolí a spotřeby během testovací jízdy



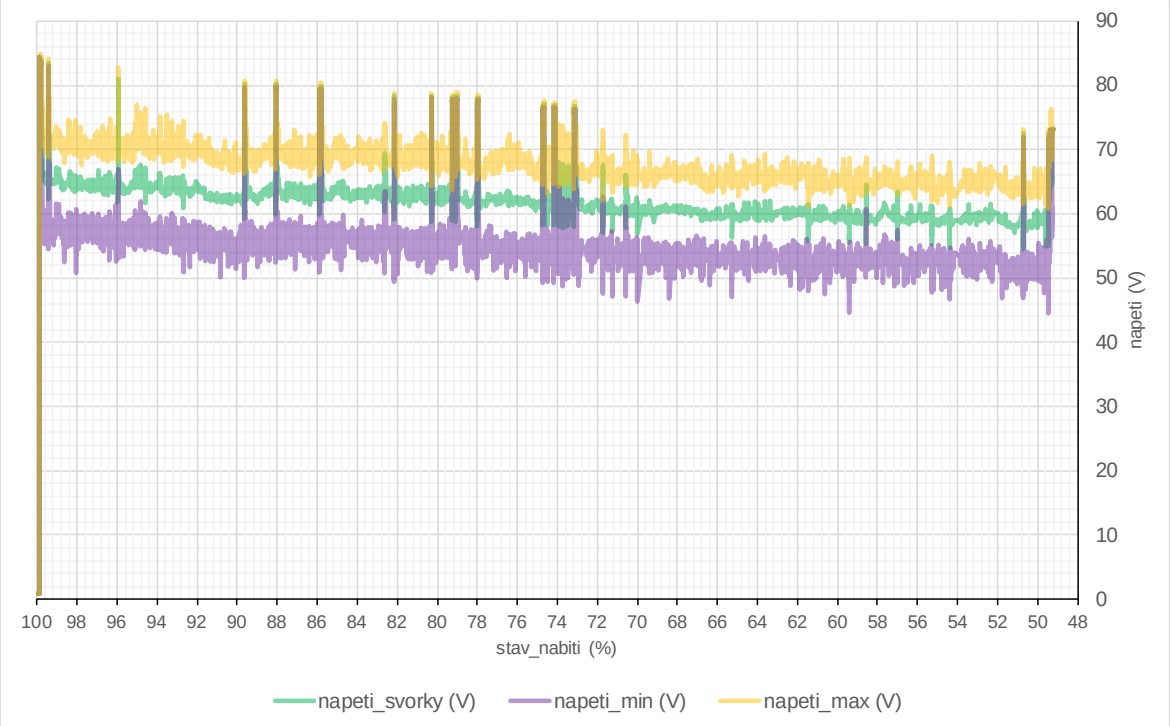
Obr. 44 Vizualizace průběhů napětí, proudu a rychlosti v závislosti na stavu nabití baterie



Obr. 45 Vizualizace hodnot zrychlení v jednotlivých osách a nadmořské výšky během testovací jízdy

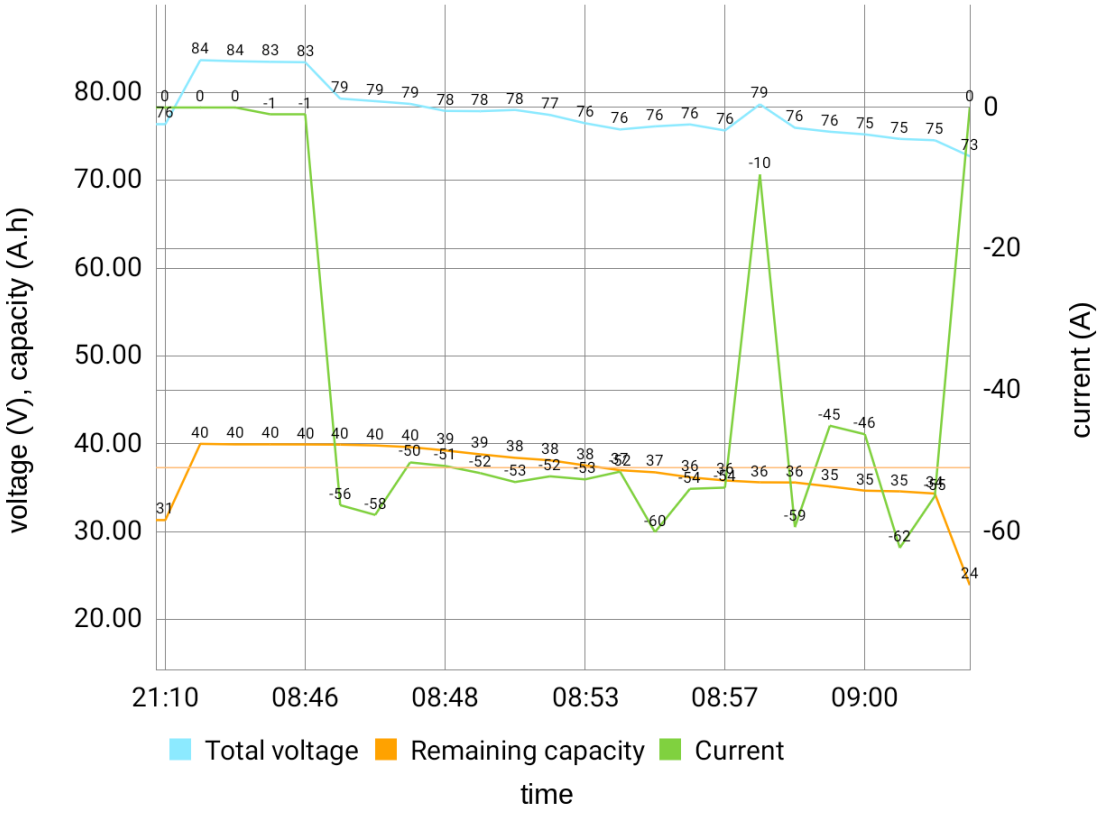


Obr. 46 Vizualizace hodnot stoupani a nadmořské výšky během testovací jízdy



Obr. 47 Vizualizace efektivních hodnot napětí, včetně min. a max. hodnot během měřící periody

Obr. 48 Vizualizace efektivních hodnot proudu, včetně min. a max. hodnot během měřící periody



Obr. 49 Data zaznamenaná prostřednictvím BMS aplikace

4.5 Evaluace dat

V případě testovacích jízd #2 až #4 byla vždy použita plně nabitá baterie, tj. na začátku každé z těchto jízd je předpokládáno 100 % užitečné kapacity. Jednalo se o novou baterii stále ve fázi „zabíhání“.

Teplota „podvozku“ nemusí nutně absolutně přesně reprezentovat teplotu okolí, a může být ovlivněna tepelnými ztrátami od motoru.

V průběhu jízdy #2 jsou patrné naměřené záporné hodnoty odběru, které mohou značit určitý stupeň rekuperace. Je však důležité vzít v potaz, zda tato energie byla skutečně předána jednotlivým bateriovým článkům, či zmařena BMS systémem.

V závěru testovací jízdy #2 došlo k výpadku signálu GPS, a tedy při propojení dat ze senzorů, byla tato nepárová data vyloučena, nicméně návaznost dat o spotřebě byla zachována.

Na charakteristikách napětí je možné pozorovat pozvolně klesající trend, z něhož lze přibližně určit obalovou křivku, resp. v případě jízdy #4 lze vycházet přímo z minimálních hodnot napětí za měřící interval.

Konfigurace akcelerometru byla následující:

- kladný směr osy +X doprava (z pohledu řidiče)
- kladný směr osy +Y dopředu (ve směru jízdy)
- kladný směr osy +Z nahoru (obloha)

Při rozjezdu vozidla vpřed je patrný záporný nárůst hodnot ve směru osy Y, následně dochází k převrácení do kladných hodnot +Y. Průměrné hodnoty zrychlení za měřící interval nepřekročily hodnotu 0,3 g. Nicméně okamžité hodnoty zrychlení se vyšplhaly až ke hranici 2 g.

Výpočet stoupání je pouze orientační, přičemž přepočet probíhal vždy po cca 50 ujetých metrech, hodnoty jsou tak platné vždy pro předchozí úsek.

Skript v jazyce Python k propojení dat ze senzorů a GPS je k dispozici v *Příloze #8*.

4.6 Zhodnocení měření

Měření napětí a proudů bylo z důvodu jejich pulsního charakteru realizováno v podobě skutečné efektivní hodnoty (TrueRMS) ve stanovených intervalech. Okamžité hodnoty napětí a proudů se totiž ukázaly jako nedostatečně vypovídající. Pro zajištění co možná nejpřesnějších hodnot poskytovaných A/D převodníkem, bylo použito zvláštního lineárního stabilizátoru, jakožto referenčního zdroje napětí. Měření teploty mělo původně zahrnovat jak teplotu okolí, tak jednotlivých článků. Bohužel kvůli zapouzdřenému provedení bateriového systému byla nakonec zjišťována pouze okolní teplota. Údaje o vnitřních poměrech baterie bylo však možné sledovat v přidružené mobilní aplikaci od výrobce bateriového systému. Rozlišení teploty bylo nakonec omezeno na pouhých 9 bitů, tj. na ½ °C, kvůli potřebné době konverze – cca 100 ms pro nejnižší možné rozlišení. Doba konverze tak vytvářela „slepá“ časová okna, díky kterým nebylo možné zajistit plnou návaznost dat z A/D převodníku a např. gyroskopu. Proběhly pokusy o přesun měření teploty do druhého výpočetního jádra, bohužel poté docházelo k častým pádům hlavního programu. Údaje z gyroskopu měly posloužit k určení úhlu stoupání, avšak to by vyžadovalo nepřerušené čtení a integraci úhlu ve velice krátkých časových intervalech, tj. řádově ms nebo dokonce jejich zlomky. Alternativně se nabízela možnost výpočtu stoupání z hodnot zrychlení, nicméně během jízdy se takto obdržené hodnoty jevily jako značně nestabilní. Výsledné stoupání bylo nakonec realizováno až při samotném vyhodnocení naměřených dat z ujeté vzdálenosti (ze souřadnic GPS) a nadmořské výšky. Ke stanovení nadmořské výšky byl použit modul barometru, který ze znalosti lokálního „normálního“ tlaku redukovaného na hladinu moře poskytoval hodnoty s přesností cca ±1 m. Výsledná přesnost byla však do jisté míry ovlivněna i neustálým vývojem tlaku v průběhu dne. Proto se následně přistoupilo pouze ke sledování relativních změn nadmořské výšky oproti startovní (známé) pozici, jejíž nadmořská výška byla určena s pomocí mapových podkladů. Opatření časovým razítkem takto získaných dat proběhlo s pomocí RTC modulu. Platný datum a čas byl před zahájením měření nahrán do paměti mikrokontroleru a dále aktualizováván, již bez účasti externího modulu. Kvůli omezenému rozlišení časového razítka, tj. pouze na celé jednotky s, bylo uměle pomocí ticků procesoru dosaženo přesnosti v rámci ms. Datum a čas spravovaný RTC modulem bylo potřeba alespoň 1×týdně manuálně přehrát kvůli jeho zpožďování.

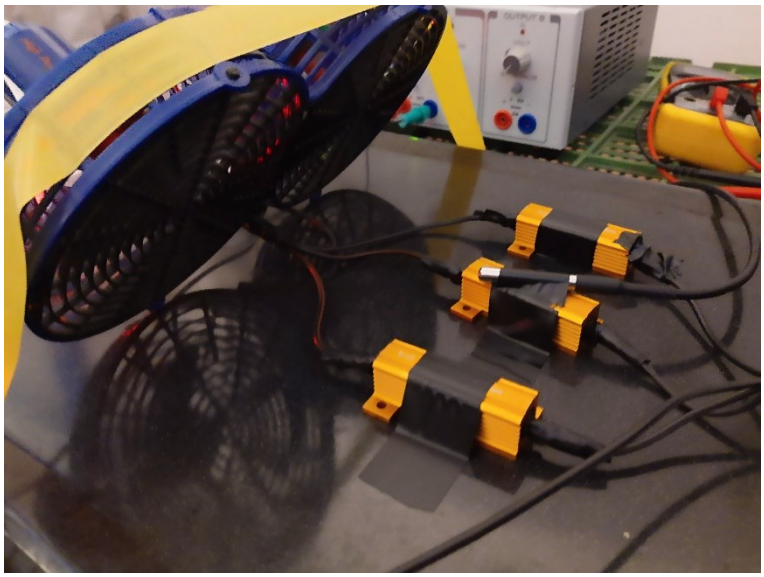
Jako druhá jednotka telemetrické skupiny posloužil GPS modul, který byl fyzicky připojen skrze TTL/USB převodník přímo k PC. Celkem byly otestovány 2 GPS moduly – *Neo-7M* a *M5Stack Mini*, z nichž se nejlépe osvědčil právě *M5Stack Mini*, který bez

problému pracoval i v městské zástavbě, a to díky podpoře více globálních navigačních systémů současně. Komunikace probíhala přes sériovou linku v podobě jednotlivých standardizovaných NMEA zpráv. Pro separaci požadovaných dat byla použita externí knihovna *NmeaParser*, která byla upravena kvůli potížím s parsováním času v desetinném formátu.

Data ze senzorů a GPS modulu byla následně shromažďována, zpracovávána a ukládána do lokální SQL databáze prostřednictvím vlastní uživatelské aplikace v jazyce C# pro Windows desktop. Samozřejmostí bylo i ovládání bezdrátového telemetrického zařízení a vizualizace dat v reálném čase. Bluetooth klient byl realizován s využitím externích knihoven z balíku *InTheHand*. Veškerá data z jízd byla posléze dodatečně zpracována pomocí skriptů v jazyce Python, zejména se jednalo o propojení jednotlivých databázových tabulek na základě časových razítek a s tím související linearizace dat z GPS, dopočet některých dalších parametrů a výsledná vizualizace.

5 Statické měření bateriového systému

Statické měření spočívalo v orientačním zjištění skutečné kapacity baterie s využitím odporové zátěže simulující odběr odpovídající příslušnému násobku její kapacity, konkrétně 0,05C. Jako zátěž byly použity výkonové rezistory *Widap 160092*, o hodnotě 150 Ω a maximálním ztrátovém výkonu 50 W (při teplotě 25 °C). Jednotlivé zátěžové rezistory byly v tomto případě řazeny paralelně, přičemž každý z rezistorů byl protékán proudem cca 0,5 A. Měření probíhalo při teplotě okolí cca 12 °C s aktivním chlazením zátěže ofukem a zabezpečením volné cirkulace vzduchu v místnosti.

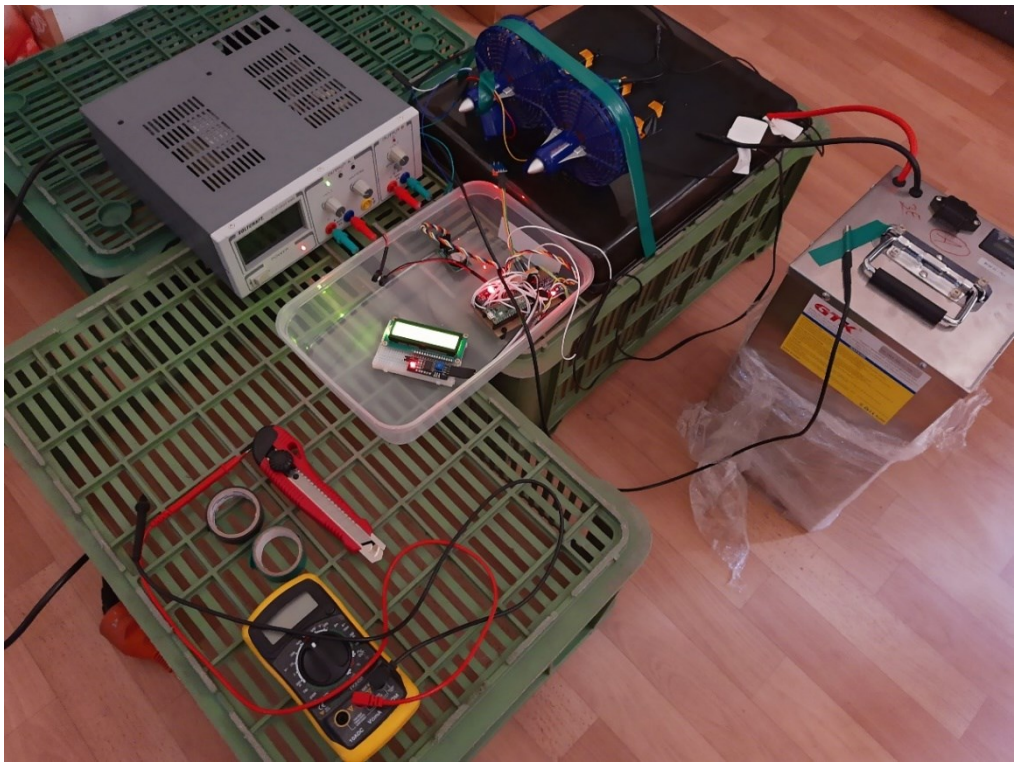


Obr. 50 Výkonové rezistory a chlazení

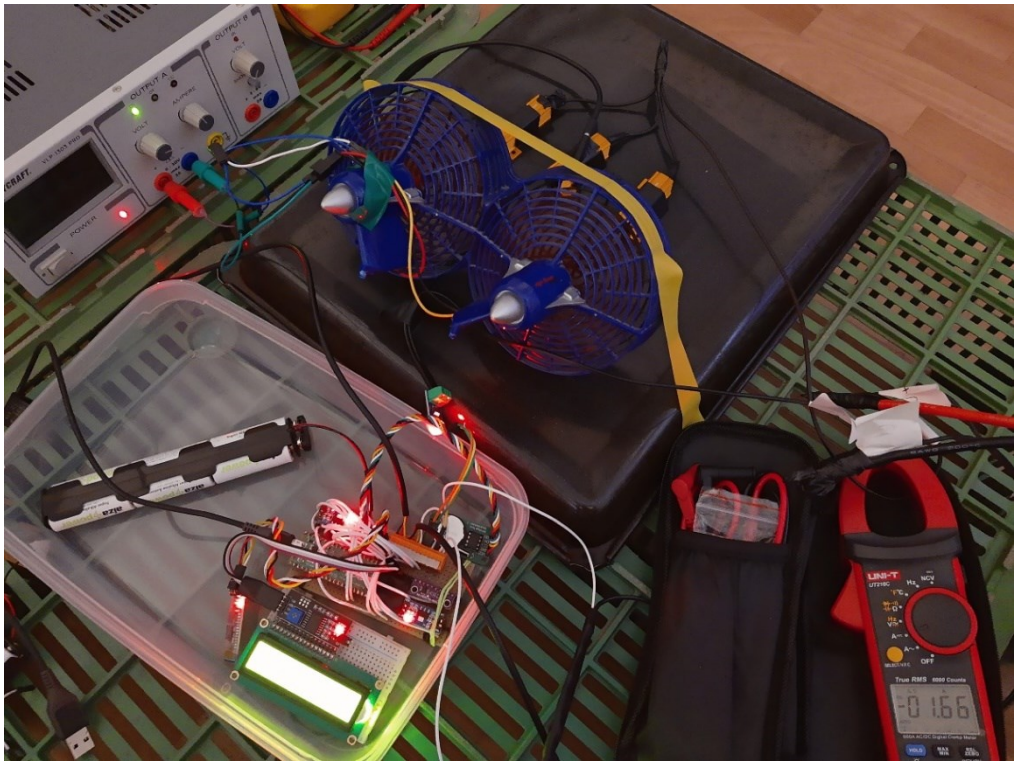
5.1 Měřící sestava

Měřící sestava sestávala z telemetrického zařízení požívaného pro snímaní provozních dat během jízd. Z důvodu nižších měřených proudů bylo však nahrazeno dosavadní proudové čidlo *HSTS016L* za základní *ACS712* o jmenovitém rozsahu 5 A, taktéž na principu Hallova jevu. Nové čidlo disponuje celkem 2 napájecími piny (5 V a GND) a 1 výstupním, u kterého je výstupní napětí opět posunuto přibližně o polovinu napájecího napětí vůči zemi; referenční vývodový pin u tohoto čidla chybí. Měřící SW v jazyce Python bylo tak potřeba doplnit o zjištění a uložení klidové hodnoty napětí (offsetu), která byla následně odečítána od dalších hodnot při reálném zatížení. Ovládání RPi Pico prostřednictvím Bluetooth bylo zachováno, nicméně naměřená data byla v 1 s intervalech přímo ukládána do interní flash paměti desky ve formátu CSV. Pro možnou kontrolu stěžejních naměřených dat byl dále přidán LCD displej, a hodnoty bylo tak možné rovnou porovnávat s údaji v aplikaci

BMS a dále do měřícího obvodu vřazenými multimetry. Ukázka měřící sestavy je uvedena níže.

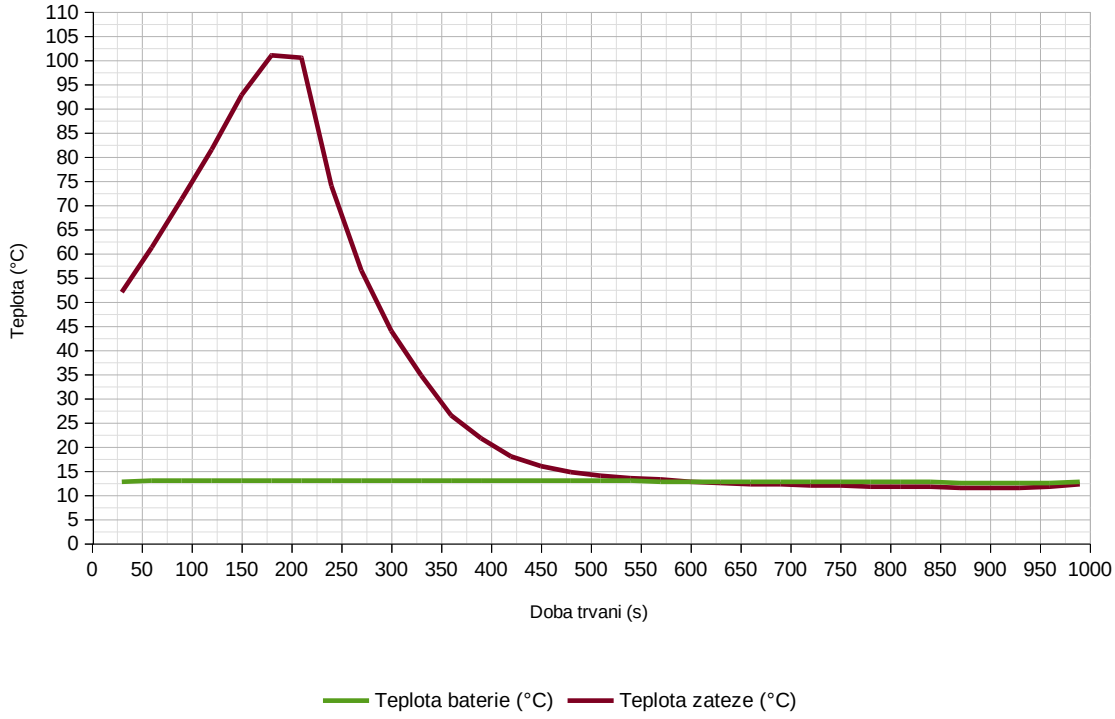


Obr. 51 Statické měření – měřící sestava komplet



Obr. 52 Statické měření – měřící sestava detail

Při sníženém výkonu chlazení docházelo poměrně brzy k enormnímu přehřívání zátěže, a to řádově s růstem 0.3 °C/s, přičemž teplota v jednu chvíli vystoupala až na 100 °C.



Obr. 53 Ryhlý nárůst teploty záteže při nižším chladícím účinku

Po následné optimalizaci směru proudění a zvýšení příkonu ventilátorů (na cca 2x3 W) se podařilo teplotu zátěže stabilizovat těsně pod úrovní 60 °C, avšak z důvodu potenciálního nebezpečí vzniku požáru při výpadku chlazení bylo měření nakonec přerušeno po odebrání energie odpovídající cca 5 % kapacity baterie. Ztrátový výkon na každém jednotlivém odporu činil:

$$P = R \cdot I^2 = 150 \cdot 0{,}55^2 = 45{,}375\ W$$

Tepelné ztráty by bylo následně možné přepočítat na nárůst teploty za 1 sekundu, v případě uvažování hliníkového pouzdra o hmotnosti *m* jen cca 30 g, přičemž měrná tepelná kapacita *c* hliníku činí přibližně 1000 J.kg^{-1}.K^{-1}:

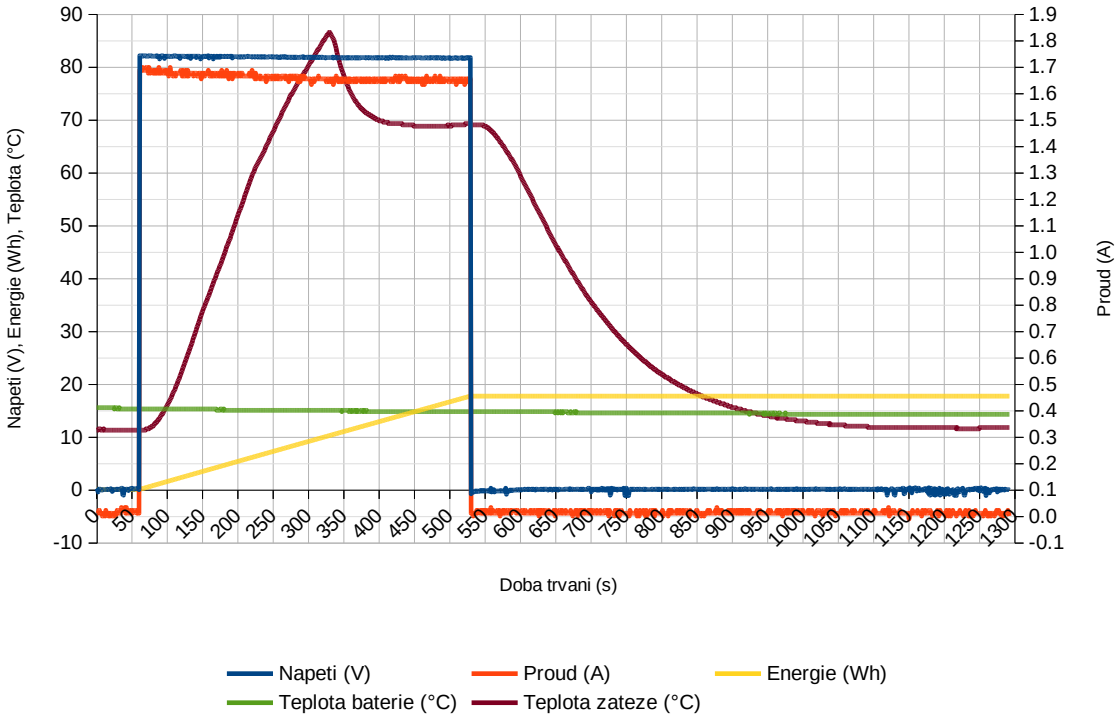
$$Q = m \cdot c \cdot \Delta T \tag{5.1}$$

$$\Delta T = \frac{Q}{m \cdot c} = \frac{45{,}375 \cdot 1}{0{,}03 \cdot 1000} \approx 1{,}5\ °C/s$$

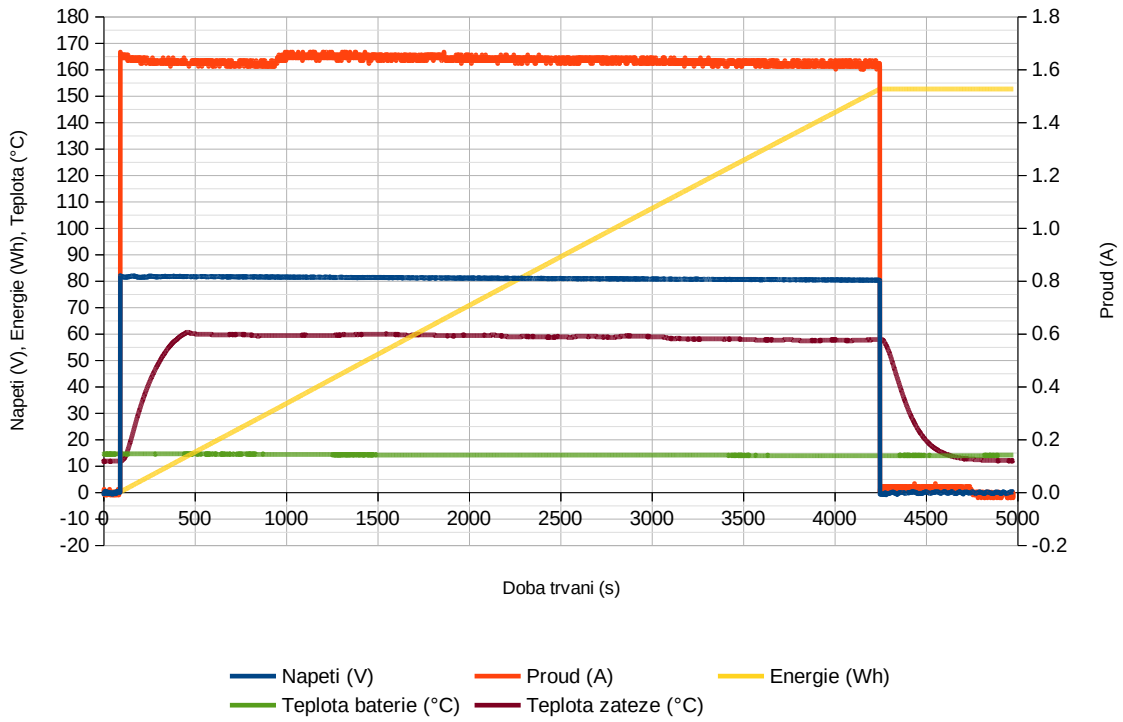
Takovýto nárůst teploty by bylo možné očekávat při úplné absenci chlazení. Při volbě výkonových rezistorů o vyšším ztrátovém výkonu, lze očekávat mnohem pozvolnější nárůst teploty, např. pro 300 W by platilo:

$$\Delta T = \frac{Q}{m \cdot c} = \frac{45{,}375 \cdot 1}{0{,}7 \cdot 1000} \approx 0{,}065\ °C/s$$

V tomto případě by byla větší i teplosměnná plocha, což by umožnilo mnohem lepší odvod ztrátového tepla do okolí a zvýšila se účinnost chlazení ofukem. Použité rezistory lze také díky montážním otvorům připevnit přímo na chladič. Efektivitu chlazení by bylo možné také posoudit sestavením modelu pro výpočet teplotního pole s využitím některého ze simulačních softwarů, např. COMSOL Multiphysics nebo ANSYS, což ale není předmětem této práce.



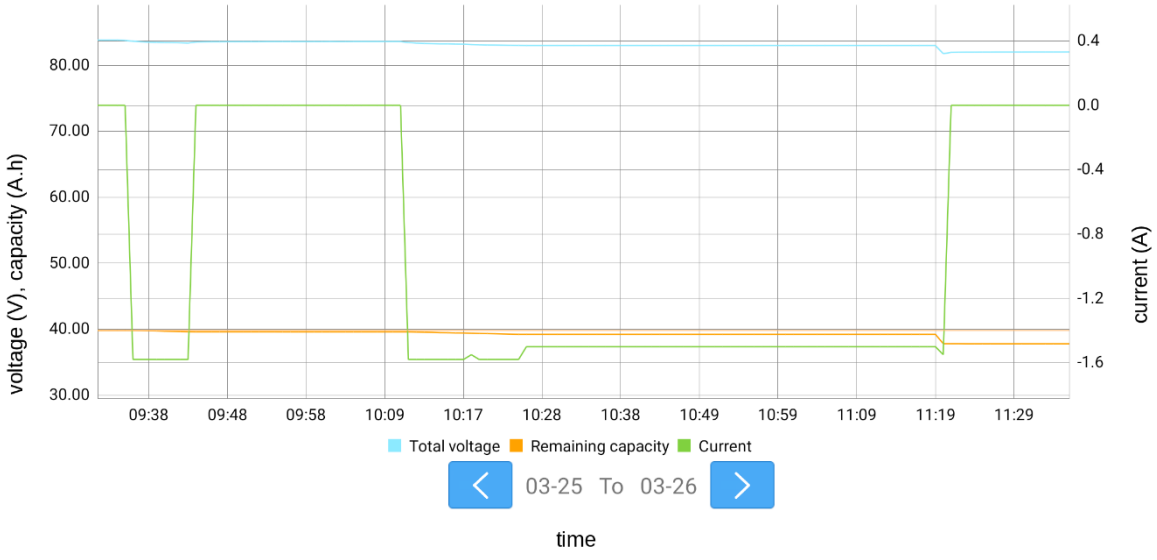
Obr. 54 Testování účinnosti chlazení



Obr. 55 Finální (částečné) měření kapacity

Vzhledem k nekompletnímu měření bude dále v modelu vycházeno ze jmenovitých parametrů kapacity jednotlivých článků udávaných výrobcem.

Níže je dále uveden průběh napětí, proudu a zbývající kapacity naměřených BMS.



Obr. 56 Záznam o vybíjení baterie z aplikace BMS

Při porovnání s hodnotami obdrženými pomocí telemetrického zařízení a multimetru, je u údajů z BMS možné pozorovat menší odchylky proudu, řádově cca desetinu ampéry.

5.2 Zhodnocení statického měření

Telemetrické zařízení bylo rovněž využito při statických měřeních. Pro účely vybíjení baterie s cílem stanovení její kapacity byl použit proud odpovídající $0{,}05C$ protékající odporovou zátěží. Původní proudové čidlo muselo být z důvodu nízkých hodnot proudů nahrazeno jiným s jmenovitým rozsahem ±5 A, opět na principu Hallova jevu. Dále byl přidán LCD displej pro zobrazení okamžitých dat. Měřící program byl upraven, tak aby umožnil zapisování naměřených dat přímo do flash paměti RPi Pico. Měření kapacity baterie se bohužel potýkalo s nadměrným zahříváním zátěže a bylo kvůli zajištění bezpečnosti předčasně ukončeno. Teploty na 50 W výkonových rezistorech v krátkém časovém rozpětí dosáhly dokonce hranice 100 °C. Následně byl optimalizován systém chlazení ofukem a teplotu se nově podařilo dlouhodobě udržet těsně pod úrovní 60 °C. Pro kompletní měření by bylo vhodné použít rezistory dimenzované na vyšší ztrátový výkon a s větší teplosměnnou plochou, případně jejich připevnění na další chladič s využitím dostupných montážních otvorů. Nominální kapacita baterie pro následný model byla tak převzata z datasheetu výrobce.

6 Model bateriového systému

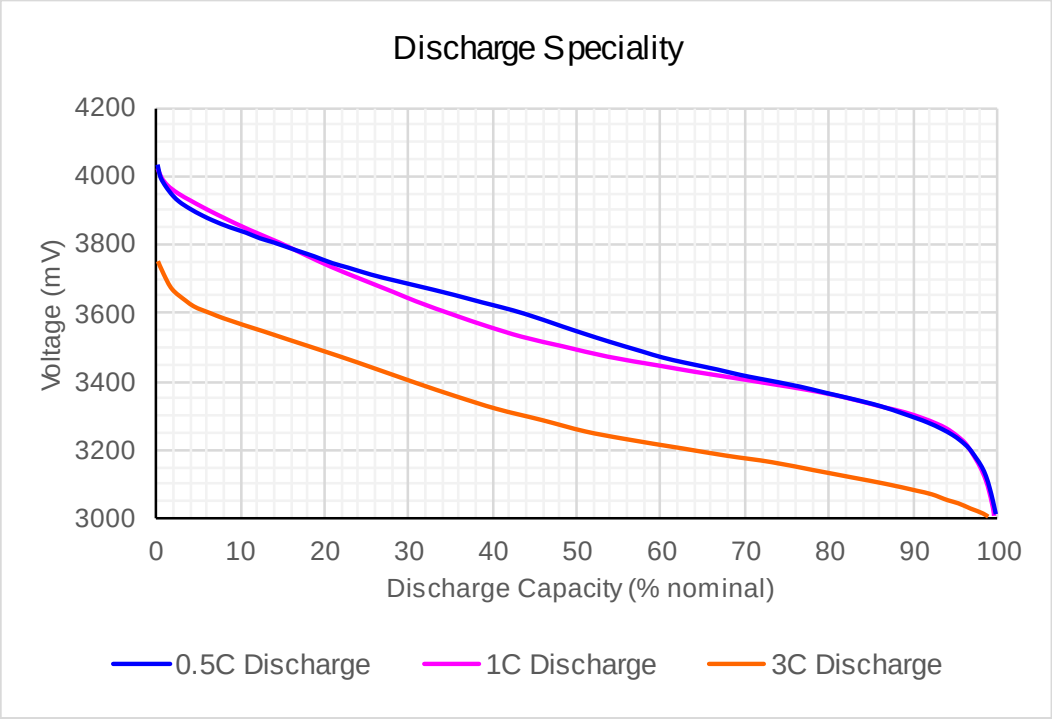
Hlavními předpoklady pro dostatečně vypovídající model bateriového systému jsou technická specifika výrobce akumulátorů a rozsáhlé množství přesných naměřených provozních dat, pokud možno z průběhu celého roku, které reprezentují chování bateriového systému na vnitřní i vnější podněty.

6.1 Technické parametry použitých článků

Bateriový systém je složen z dílčích Li-ion cylindrických článků v pouzdře 18650, výrobce MOTOMA. Základní parametry článků (LCR18650-2000MAH) jsou uvedeny v seznamu níže:

- **Jmenovité napětí** … 3,7 V
- **Jmenovitá kapacita** … 2000 mAh
- **Koncové nabíjecí napětí** … 4,2 V
- **Maximální nabíjecí proud** … 1000 mA (0,5C)
- **Konečný nabíjecí proud** … 20 mA (0,01C)
- **Maximální vybíjecí proud** … 6000 mA (3C)
- **Krátkodobý vybíjecí proud** … 10 000 mA (5C), max. 1 min.
- **Konečné vybíjecí napětí** … 3,0 V
- **Garantovaná min. kapacita po 300 nab./vyb. cyklech ≥ 80 % výchozí kap.**
- **Pracovní rozsah teplot:**
  - ***Nabíjení*** … 0 ÷ 45 °C
  - ***Vybíjení*** … -20 ÷ 60 °C
- **Relativní vlhkost:** … 45 ÷ 85 %

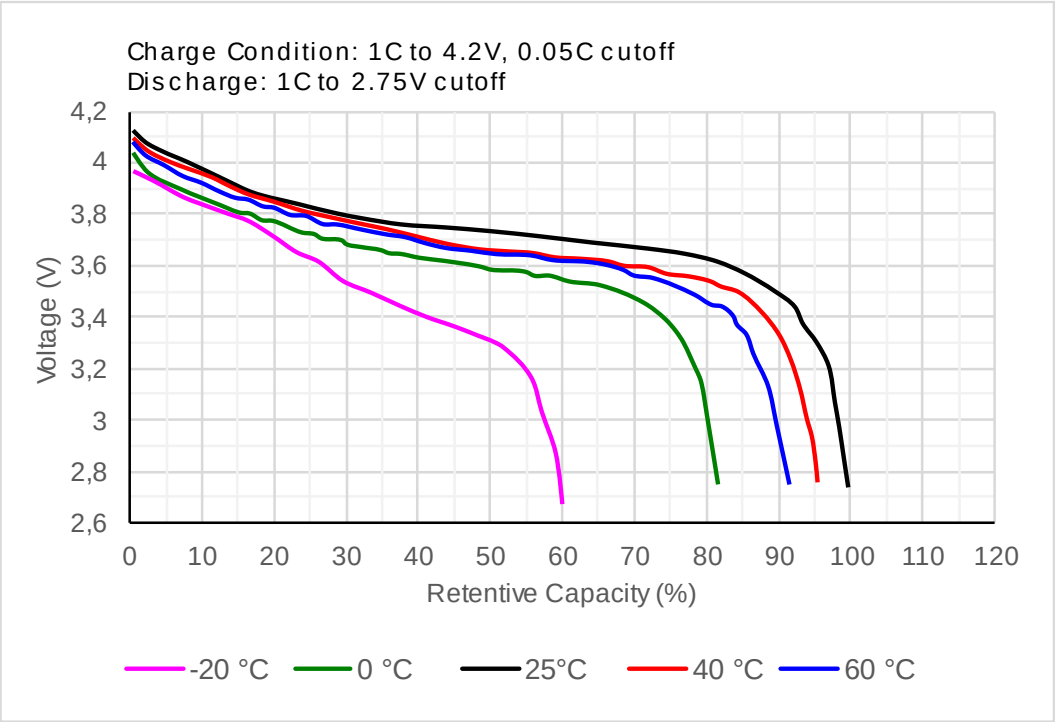
Dále jsou k dispozici charakteristické průběhy hlavních veličin, parametry měřícího vybavení, se kterým byly charakteristiky pořízeny, podmínky prostředí, měřicí postupy a různé bezpečnostní procedury.



Obr. 57 Vybíjecí křivky pro různé hodnoty proudů

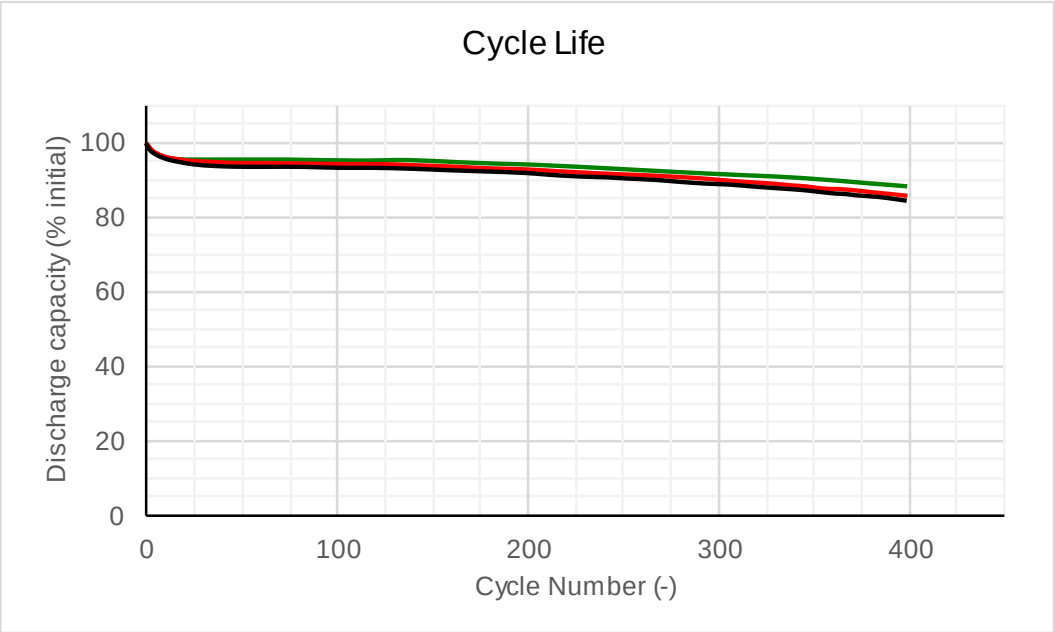
Z výše vyobrazených vybíjecích charakteristik je patrné, že velikost vybíjecího proudu ovlivňuje velikost napětí článků. Běžně pak vyšší vybíjecí proudy mohou snížit užitečnou velikost kapacity, avšak v případě rozsahu proudů 0,5C až 3C k žádnému výraznému ponížení kapacity nedochází. Mimo jiné platí, že sklon charakteristik se odvíjí od velikosti vnitřního odporu článků. Čím nižší je jeho hodnota, tím „plošší" je výsledná charakteristika, výkon je předáván efektivněji a projevují se nižší úbytky napětí, což má pozitivní vliv na celkovou dobu provozu (oddálení cut-off voltage). Typická hodnota pro Li-ion články se pohybuje řádově 10^{-1} Ω [16]. V datasheetu výrobce tento údaj bohužel chybí.

Dalším významným sledovaným parametrem je teplota článků. Opět jsou k dispozici vybíjecí charakteristiky, tentokrát při konstantním vybíjecím proudu 1C. Konec vybíjení (cutoff) je nastaven na 2,75 V. Z jednotlivých vybíjecích křivek je patrné, že maximum energie lze získat při provozu za teploty 25 °C. S rostoucí teplotou se množství užitečné energie snižuje, stejně tak je tomu v případě nízkých teplot, kde je však pokles nejmarkantnější. Vystavením článků vysokým teplotám vede k umocnění vnitřních degradačních procesů. U některých typů akumulátorů může dokonce docházet k mírnému navýšení jejich kapacity při provozu za teplot o něco málo vyšších než standardních 25 °C.



Obr. 58 Vybíjecí křivky (1C) v závislosti na provozní teplotě

Výrobce rovněž udává životnost svých článků v podobě závislosti změny kapacity vlivem narůstajícího počtu nabíjecích cyklů. Dle specifikace by neměla kapacita po 300 nabíjecích cyklech klesnout pod 80 % její nominální hodnoty. Nicméně z přiložených charakteristik vyplývá, že pokles kapacity po 300 cyklech činí jen cca 10 %. Životnost článků ovšem velmi závisí na respektování podmínek nabíjení a vybíjení a dodržování zásad výrobce.



Obr. 59 Závislost kapacity na počtu nabíjecích cyklů

Kromě jiného jsou zmíněny doporučené podmínky pro skladování baterií, kdy lze i při jejich nečinnosti očekávat jistou míru samovybíjení. Vhodným nakládáním s bateriemi lze tento vliv redukovat na minimum. Kompletní technický list je uveden v *Příloze #9*.

6.2 Popis chování bateriového systému

Nejzákladnější verzi modelu si lze snadno představit na principu integrace výkonu. Prostřednictvím telemetrického zařízení je možné poměrně přesně stanovit množství energie odebrané baterii. Podobně funguje i přidružený BMS systém, u něhož však byly za příležitosti statických měření zjištěny jisté odchylky od skutečných hodnot napětí a proudů, a z nich plynoucí případné nepřesnosti modelu. Je-li známa nominální energie baterie, lze následně dopočítat její poměrný stav nabití. Za skutečného provozu je však potřeba pracovat se skutečnou dostupnou energií baterie, proto je vhodné tento údaj držet neustále v paměti, a rovněž brát v úvahu případné dobíjení.

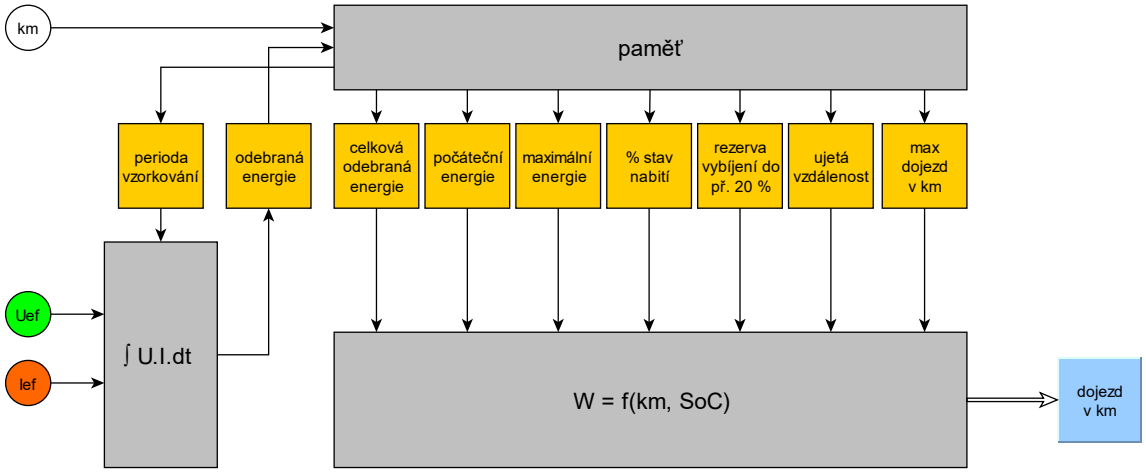
Kapacitu lithiových baterií může dále ovlivnit teplota. Při velmi nízkých teplotách je nutné počítat s výrazným poklesem kapacity, řádově o 20 až 40 % při teplotách pod bodem mrazu. S kapacitou navíc velice úzce souvisí úroveň napětí. Pokud dojde k poklesu napětí článků nebo jejich skupin pod stanovenou mez, tj. bude dosaženo konečného vybíjecího napětí, jsou pak automaticky vyřazeny z provozu systémem BMS, záleží na konkrétní konfiguraci. V případě námi použitého bateriového systému dochází k odstavení při poklesu napětí pod úroveň 60 V delším než 10 s. Minimální hodnota napětí článku je dále nastavena na 2,5 V. V zimních měsících za studeného stavu baterie se mohou dostavit velké proudové špičky, kdy baterie není schopna poskytnout dostatečný výkon, a napětí okamžitě klesá pod mezní úroveň. Nicméně vliv okolní teploty postupně klesá s dobou aktivního provozu baterie, kdy dochází k postupnému oteplování článků od provozních proudů. Z údajů v aplikaci těsně po skončení jízdy byla zaznamenána teplota cca 34 °C, skutečná teplota během jízdy tedy může být pochopitelně ještě o něco málo vyšší. U některých elektromobilů může být problematika studeného stavu řešena dodatečným vytápěním prostoru bateriového systému, což však způsobuje dodatečné ztráty.

6.3 Modelové případy

Dostupný dojezd elektromobilu lze do jisté míry odhadnout na základě předpokládaného (typického) chování bateriového systému, měření provozních veličin a zahrnutím údajů z tachometru, případně GPS. Dále budou rozebrány jednotlivé modelové situace.

6.3.1 Prostá integrace výkonu

Stanovením množství energie odebrané baterii, při uvažování stavu před začátkem jízdy, a maximální jmenovité kapacity, lze stanovit její poměrné vybití. Model je dále dobré doplnit o určitou energetickou rezervu, např. 20 % nominální kapacity. Odhad počtu zbývajících km je možné učinit na základě již ujeté vzdálenosti a jí odpovídající spotřebované energii. Případnou korekci (na počátku jízdy) lze učinit na základě teoretické maximální hodnoty dojezdu.

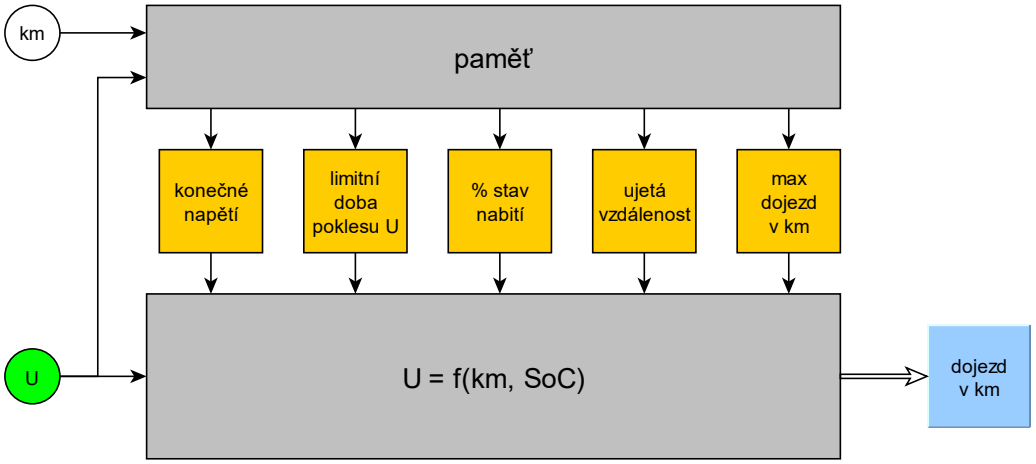


Obr. 60 Prostý integrální model

Model může být potenciálně ovlivněn teplotou baterie ve studeném stavu a počtem kompletních nabíjecích cyklů.

6.3.2 Sledování napěťové křivky

Při najetí minimální vzdálenosti je možné stanovit trend vývoje napětí, opět vzhledem k ujeté vzdálenosti či stavu nabití baterie. Hledáme tak průsečík s hranicí minimální úrovně napětí definovanou BMS, tj. konečné vybíjecí napětí, a jemu odpovídající vzdálenost v km.



Obr. 61 Model sledující trend napětí

6.3.3 Dopředný model

Reálná měření provozních veličin poskytují informaci o aktuálním stavu bateriového systému, jeho dostupné kapacitě. V případě, že by bylo k dispozici větší množství dat reprezentující nejrůznější provozní situace, přesně vydefinované profily tras, napříč ročními obdobími, se zahrnutím jízdních specifik řidičů (jejich jízdních profilů), pak by bylo možné vytvořit tzv. dopředný model, který by na základě znalosti cílového místa, výškového profilu trasy, aktuální dopravní situace, případně také počasí, dokázal velice přesně předpovědět informaci o zbývajícím dojezdu, jeho proveditelnosti, případně navrhnout úpravu trasy s ohledem na rozmístění a dostupnost nabíjecích míst. S výhodou by zde mohly být použity prvky strojového učení, v podobě např. neuronových sítí.

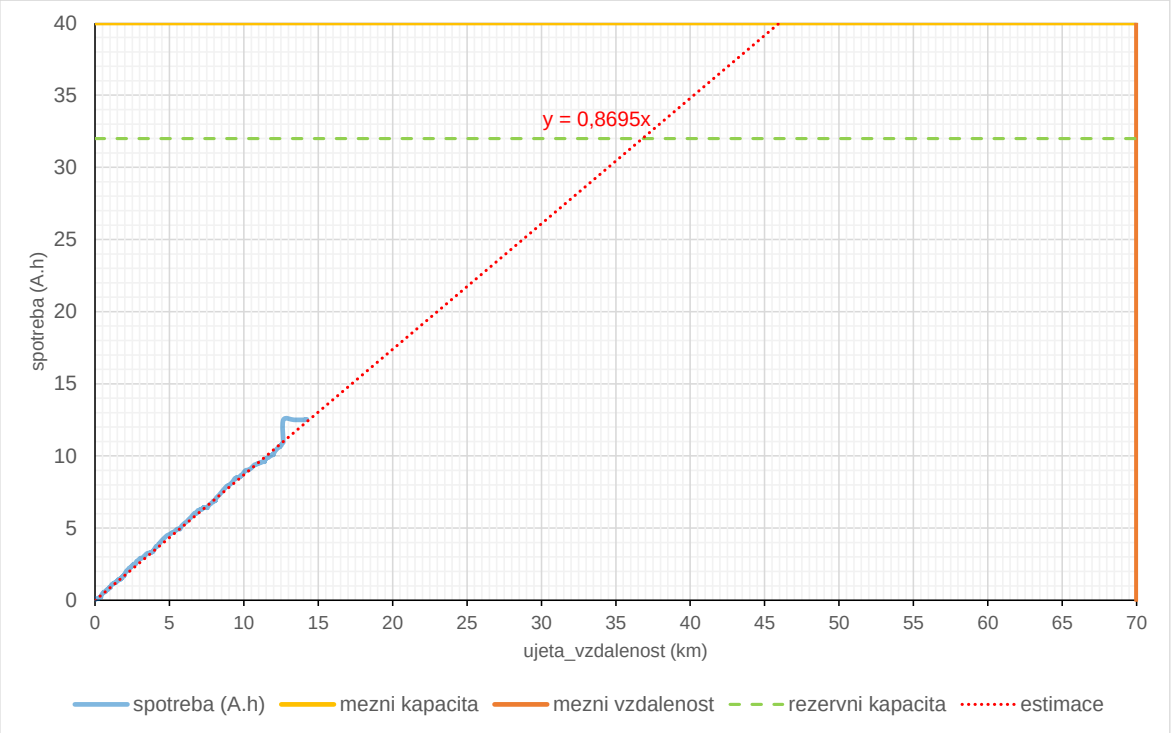
7 Ověření funkce matematického modelu

Ověření matematického modelu proběhlo na datech získaných z jednotlivých jízd, a současně byla získaná data porovnána s údaji z aplikace BMS. Ve vztahu k modelovým případům uvedeným v předešlé kapitole byly stanoveny následující dostupné dojezdy bezprostředně po ukončení jízd #2 až #3…

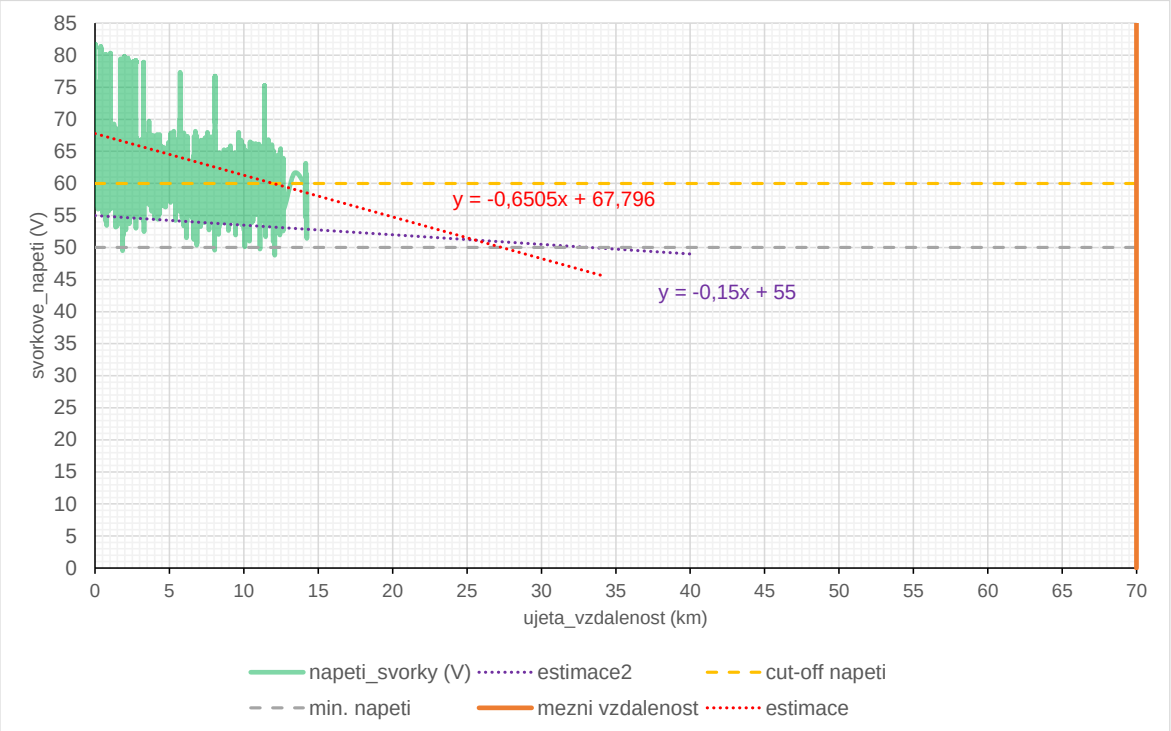
Tabulka 15 Srovnání dostupného dojezdu od různých modelů

Sledovaný parametr	Jízda #2	Jízda #3	Jízda #4
Užitečná kapacita před začátkem jízdy (A.h)	40	40	40
Celková odebraná energie během jízdy (A.h)	12,530	22,206	20,315
Celk. odebraná energie dle aplikace BMS (A.h)	14,430	19,480	16,090
Celková ujetá vzdálenost (km)	14,235	23,900	20,431
Průměrná spotřeba (A.h/km)	0,880	0,929	0,994
Zbývající dojezd při Ø spotřebě (km)	31,2	19,1	19,7
Zb. dojezd při Ø sp. a 20 % kap. rezervě (km)	22,1	10,5	11,7
Zb. dojezd na základě trendu spotřeby (km)	31,7	19,5	19,8
Zb. dojezd na základě trendu sp. s kap. rez. (km)	22,5	10,8	11,7
Zb. dojezd na základě trendu $U_{stř}$ do 60 V (km)	-2,3	4,3	9,4
Zb. dojezd na základě trendu U_{min} do 50 V (km)	19,0	6,1	9,5
Zb. dojezd dle aplikace BMS	-	-	44,4

Další případná měření, resp. provedení testovacích jízd až do okamžiku zhoršení jízdních vlastností pod přípustnou mez, by vedla k zpřesnění modelů a ověření jejich vhodnosti.

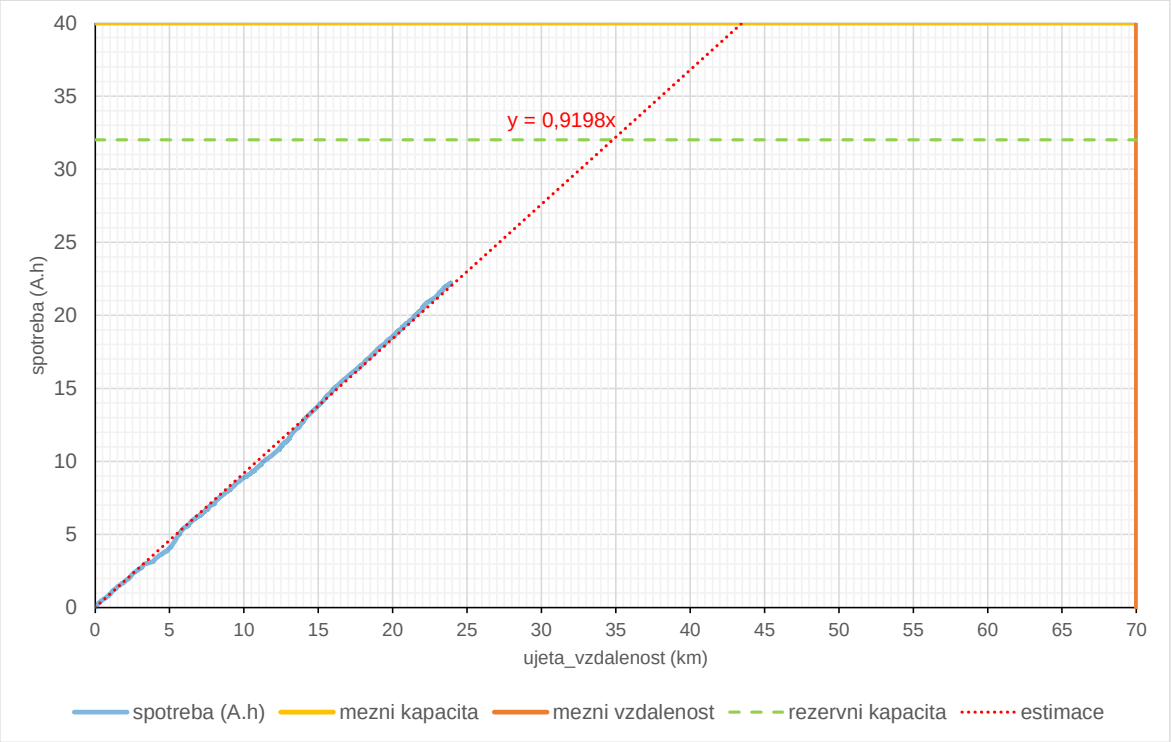


Obr. 62 Odhad zbývajícího dojezdu po jízdě #2 na základě trendu odběru

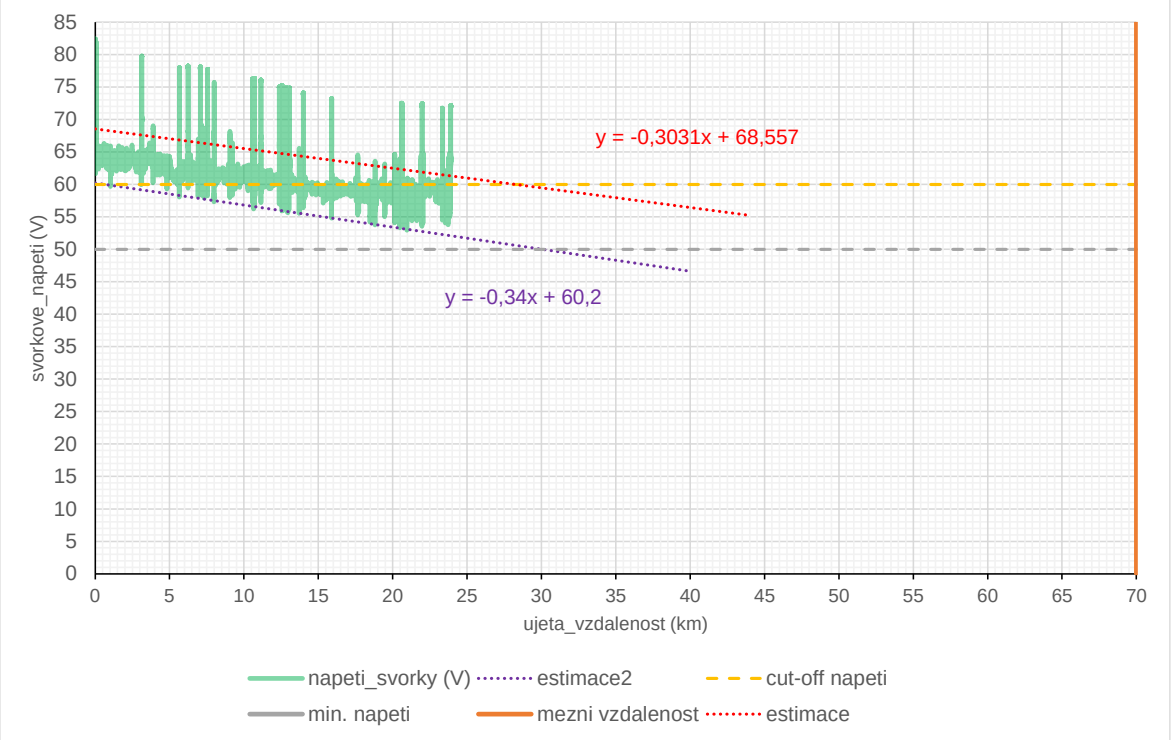


Obr. 63 Odhad zbývajícího dojezdu po jízdě #2 na základě trendů napětí

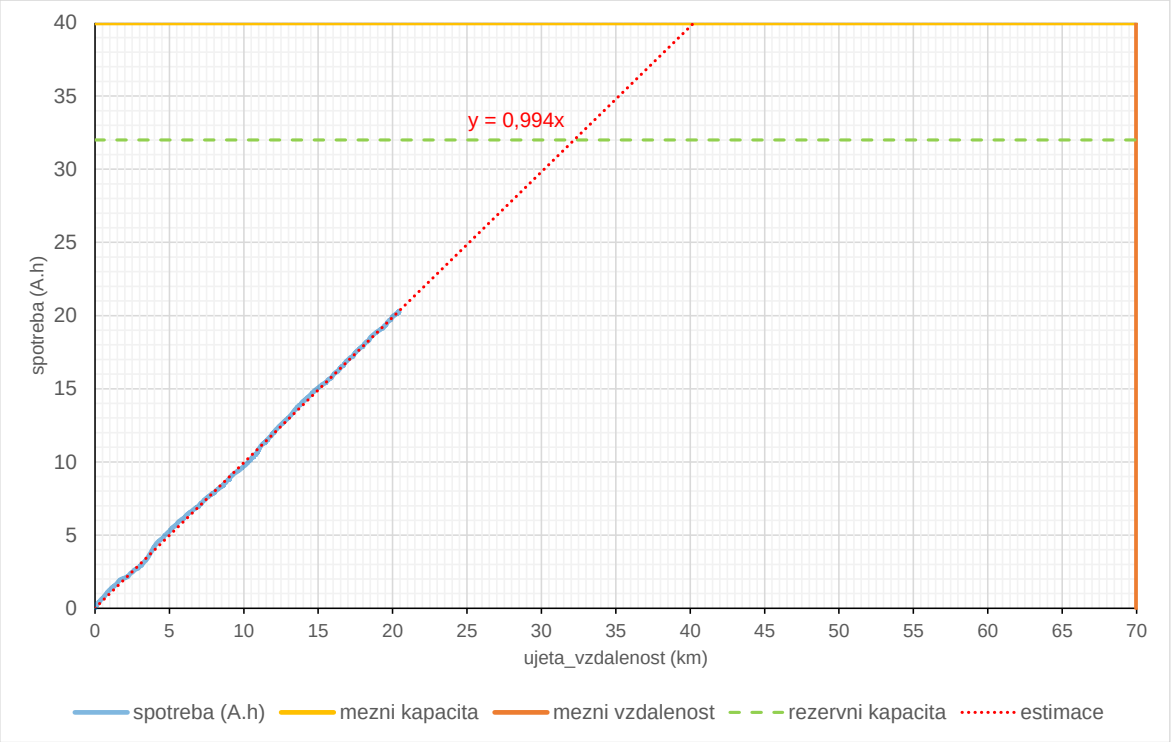
Z výše uvedeného grafu je patrné, že pro zjištění trendu poklesu napětí jsou spíše žádoucí efektivní hodnoty napětí vyhodnocované za danou periodu, namísto okamžitých hodnot.



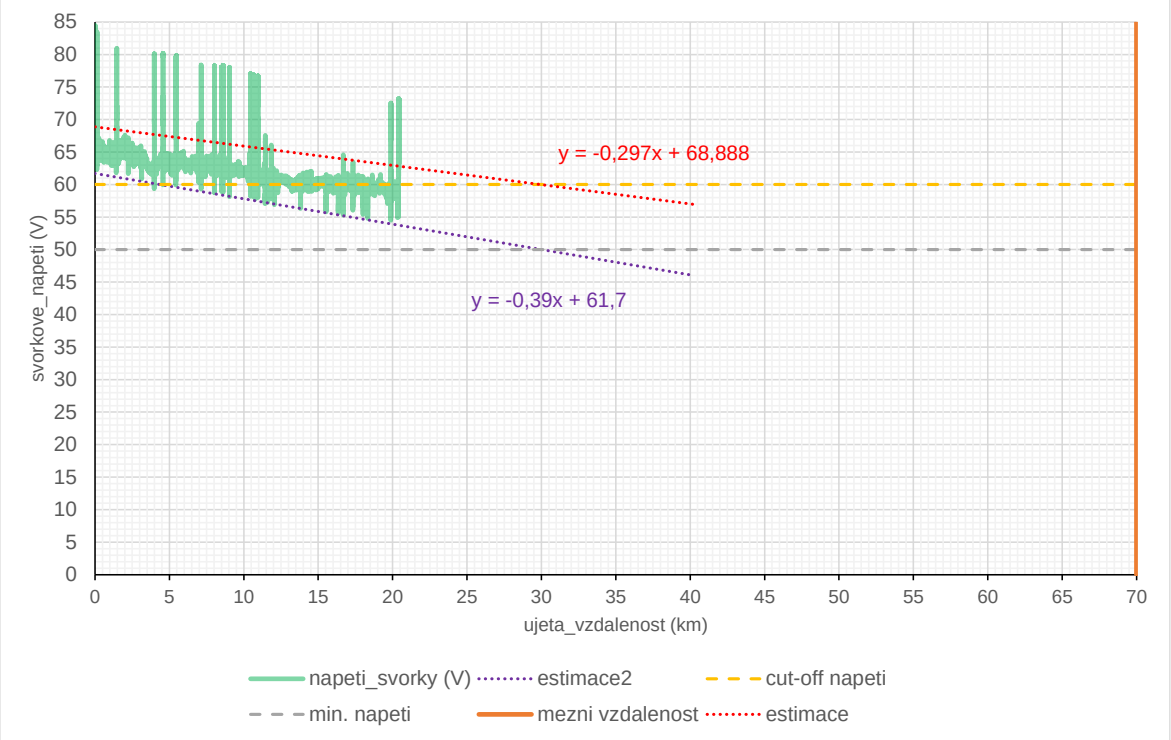
Obr. 64 Odhad zbývajícího dojezdu po jízdě #3 na základě trendu odběru



Obr. 65 Odhad zbývajícího dojezdu po jízdě #3 na základě trendů napětí

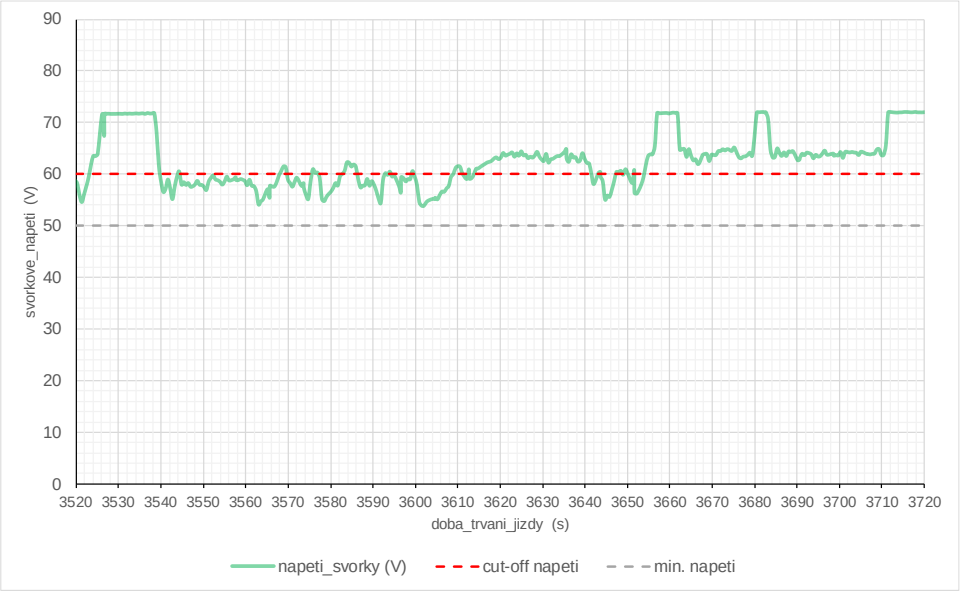


Obr. 66 Odhad zbývajícího dojezdu po jízdě #4 na základě trendu odběru

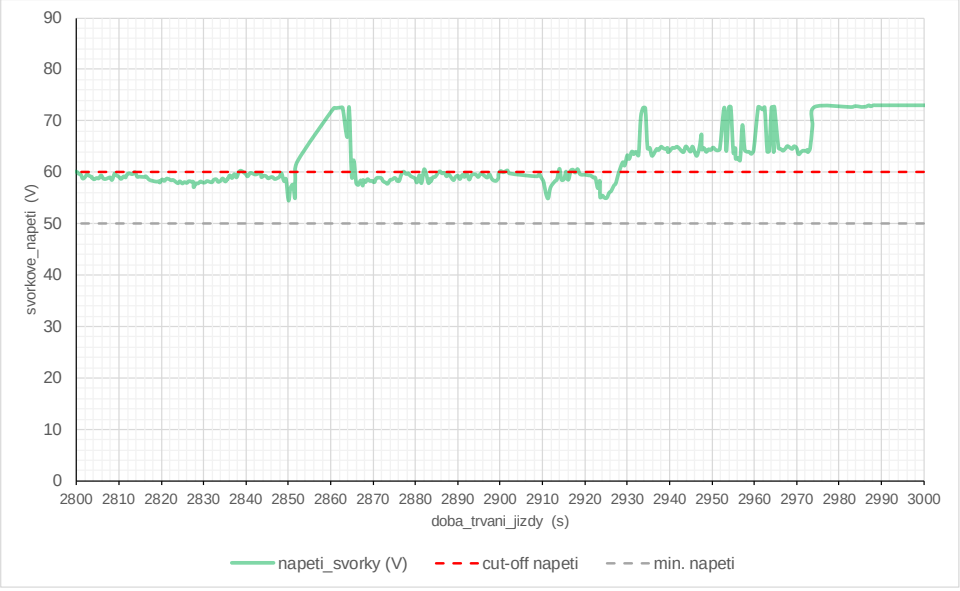


Obr. 67 Odhad zbývajícího dojezdu po jízdě #4 na základě trendů napětí

Grafy níže (*Obr. 68, 69*) zachycují závěrečný 200 s úsek z testovacích jízd #3 a #4, kdy na úplném konci dochází k narovnání napětí vlivem následného konstantního minimálního odběru (cca 1 A) při nastartovaném vozidle. Současně jsou patrné minoritní poklesy pod úroveň konečného vybíjecího napětí (cut-off). V případě doby trvání těchto poklesů překračující nastavených 10 s, by mělo dojít k ukončení procesu vybíjení. Nicméně BMS hlídá poměry přímo na jednotlivých článcích, a v případě dosažení limitního stavu některého z nich vypíná bateriový výstup. Klíčovou roli zde hrají také přesnost a metodika měření provozních veličin systémem BMS.

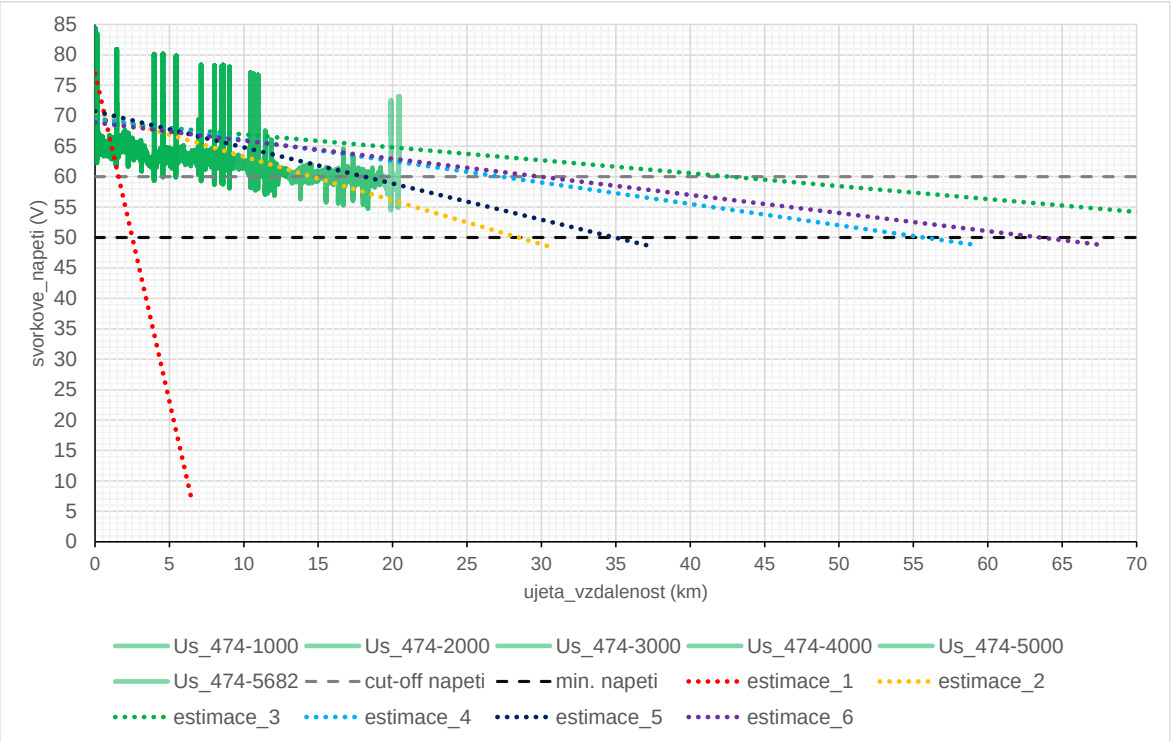


Obr. 68 Závěrečný časový úsek s rozlišením po 10 s testovací jízdy #3 a vyznačením limitních úrovní napětí



Obr. 69 Závěrečný časový úsek s rozlišením po 10 s testovací jízdy #4 a vyznačením limitních úrovní napětí

Poslední graf (*Obr. 70*) zachycuje vliv počtu naměřených hodnot na výsledném stanovení trendové křivky napětí.



Obr. 70 Odhad zbývajícího dojezdu postupně během testovací jízdy #4

Pro zajímavost určením křivek trendu napětí z jízd #3 a #4 bylo dosaženo velmi podobných rovnic. Nicméně s ohledem na *Obr. 70* je zřejmé, že pro dosažení optimální přesnosti modelu při nízkém počtu najetých kilometrů je v nejlepším zájmu doplnit výpočet o alespoň prostou časovou integraci výkonu, na základě níž by bylo možné stanovit i stav nabití (*SoC*) akumulátoru. Zde by se rovněž s výhodou uplatnila i informace o předchozí spotřebě, případně eventuálních částečných dobíjeních, které by byly uloženy v paměti palubního počítače. Ze zkušenosti, výrobce tříkolky udává, že bezpečná úroveň vybití Li-ion baterie (*DoD*) je cca 80 %. Dále již může při větších proudových špičkách docházet k výpadkům dodávky výkonu. 20 % energetická rezerva byla proto využita i při orientačních výpočtech dojezdu při uvažování průměrné spotřeby, která se pohybovala řádově 0,9÷1,0 A.h/km.

Závěr

Předmětem diplomové práce bylo nejprve zhodnotit aktuální nasazení bateriových systémů pro elektromobilitu, co do jednotlivých typů, účelu, a jejich srovnání navzájem. Dále se postup prací zaměřil na vývoj a konstrukci telemetrického zařízení, pomocí něhož následně proběhla řada měření provozních parametrů a stavů v rámci jednotlivých testovacích jízd. Pro ověření katalogových údajů článků použitých v bateriovém systému byla uskutečněna statická měření v definovaném prostředí s využitím příslušné zátěže. Obdržené hodnoty reprezentující chování bateriového systému byly posléze evaluovány a transformovány na užitečný bateriový model.

V úvodní teoretické části byly zmíněny a podrobně vysvětleny principy bateriových systémů založených na olověných a lithiových akumulátorech, a to včetně jejich typických parametrů a charakteristik uváděných v technických listech. Nejčastějším typem olověných akumulátorů jsou tzv. bezúdržbové gelové. V případě lithiových se jedná o technologie Li-ion a $LiFePO_4$. Jako zajímavá náhrada za původní olověné akumulátory jsou pokládány články $LiFePO_4$, které vynikají vysokou bezpečností provozu.

Pro měření a záznam vybraných parametrů a veličin během testovacích jízd bylo vyvinuto vlastní telemetrické zařízení, které sestává celkem ze dvou jednotek. První jednotka obsahuje jednočipový počítač Raspberry Pi Pico programovaný v jazyce MicroPython, externí A/D převodník, sdružený modul akcelerometru, gyroskopu a barometru, digitální teplotní čidla, proudový senzor na principu Hallova jevu, hodiny reálného času, komunikační Bluetooth adaptér a napájecí obvody v podobě stabilizátorů napětí. Tímto bylo možné pokrýt veškeré význačné sledované provozní parametry, opatřit je časovým razítkem a následně je prostřednictvím Bluetooth odeslat obslužnému softwaru pro platformu Windows. Druhou jednotku představoval GPS modul fyzicky propojený s PC přes sériové rozhraní.

Měření napětí a proudů bylo realizováno v podobě skutečných efektivních hodnot. Dále byla zaznamenávána okolní teplota a vnitřní poměry baterie byly sledovány s pomocí mobilní aplikace výrobce BMS. Výpočet stoupání byl uskutečněn při vyhodnocení naměřených dat z ujeté vzdálenosti (ze souřadnic GPS) a nadmořské výšky. Data ze senzorů a GPS modulu byla soustřeďována do lokální SQL databáze prostřednictvím vlastní uživatelské aplikace v jazyce C# pro Windows desktop. Data z jízd byla potom zpracovávána pomocí skriptů v jazyce Python.

Model bateriového systému byl koncipován s využitím naměřených dat z testovacích jízd provedených na tříkolce firmy *elBlesk*, katalogových parametrů a základních charakteristik výrobce použitých Li-ion článků, a teoretických předpokladů.

V rámci testovacích jízd bylo možné pozorovat klesající trend svorkového napětí baterie, případně stanovit jeho obalovou křivku. Vlivem pulsního charakteru proudu docházelo k deformacím průběhu napětí. Maximální zjištěné hodnoty proudu dosahující téměř 80 A způsobovaly značný propad napětí. Právě napětí je v podání systému BMS jednou z hlídaných a strategických veličin, kdy pokles pod stanovenou úroveň – tzv. *cut-off voltage*, nebo-li konečného vybíjecího napětí, po dobu delší než 10 s by mělo přerušit dodávku výkonu. Minimální hodnota napětí pro jednotlivé články či jejich skupiny je však stanovena na 2,5 V, tj. cca 50 V na bateriový systém. Sledováním vývoje napětí by pak mělo být možné s využitím aktuálně ujeté vzdálenosti stanovit zbývající dojezd. Velmi podobných rovnic bylo dosaženo určením křivek trendu napětí z jízd #3 a #4. Orientační dojezdy jsou dostupné v *Tab. 15*. Jelikož zatím neproběhla žádná z testovacích jízd až do úplného zastavení elektromobilu, nebo do výrazného zhoršení jeho jízdních vlastností, nelze 100 % potvrdit dosažené výsledky. Maximálně lze vzít částečně v potaz údaje z mobilní aplikace BMS, ty však mohou být rovněž zatíženy určitou malou relativní chybou (do 10 %), která byla odhalena při statických měřeních, a to v podobě mírně nepřesného měření velikosti proudu, tj. cca 0,1÷0,2 A při testovaném rozsahu do 2 A.

Reálná měření provozních veličin takto poskytla celou řadu poznatků o stavu a chování bateriového systému. Pokud by bylo k dispozici větší množství dat reprezentující nejrůznější provozní situace, přesně vydefinované profily tras, napříč všemi ročními obdobími, se zahrnutím jízdních specifik řidičů (jejich jízdních profilů), pak by bylo možné tato data využít k dalšímu zpřesnění modelu. Na základě znalosti cílového místa, výškového profilu trasy, aktuální dopravní situace, případně také počasí, by bylo možné s dostatečnou přesností předpovědět informaci o zbývajícím dojezdu, jeho proveditelnosti, případně navrhnout úpravu trasy s ohledem na rozmístění a dostupnost nabíjecích míst. S výhodou by zde mohly být použity prvky strojového učení, v podobě např. neuronových sítí.

Literatura

[1] DHAMEJA, Sandeep. *Electric Vehicle Battery Systems*. Elsevier Science. 2001. ISBN: 9780080488769.

[2] PAVLOV, Detchko. *Lead-Acid Batteries: Science and Technology: A Handbook of Lead-Acid Battery Technology and Its Influence on the Product*. Elsevier Science. 2017. ISBN: 9780444595607.

[3] KULBÍN, Pavel. *Elektrická zařízení osobních automobilů*. SNTL - Státní nakladatelství technické literatury. 1985. ISBN: 04-547-85.

[4] *Jak funguje olověný akumulátor*. TZB-info [online]. [cit. 2021-11-17]. Dostupné z: https://oze.tzb-info.cz/akumulace-elektriny/16090-jak-funguje-oloveny-akumulator

[5] *Průvodce světem olověných akumulátorů*. Extol [online]. [cit. 2021-11-27]. Dostupné z: https://www.extol.cz/ke-stazeni/manuals/doc/417302.pdf

[6] *Pokročilé technologie olověných akumulátorů – gel a AGM*. TZB-info [online]. [cit. 2021-11-16]. Dostupné z: https://oze.tzb-info.cz/akumulace-elektriny/16815-pokrocile-technologie-olovenych-akumulatoru-gel-a-agm

[7] *How bad is it to undervoltage a 12-volt lead-acid battery?* [online]. [cit. 2021-11-16]. Dostupné z: https://newbedev.com/how-bad-is-it-to-undervoltage-a-12-volt-lead-acid-battery

[8] *Lithiové akumulátory. Přehled základních typů a jejich vlastností*. TZB-info [online]. [cit. 2022-02-22]. Dostupné z: https://oze.tzb-info.cz/akumulace-elektriny/13612-lithiove-akumulatory

[9] SARKAR, Subodh. *Lithium-Ion Battery: The Power of Electric Vehicles with Basics, Design, Charging technology & Battery Management Systems*. Subodh Sarkar. 2018. ISBN: 9780463160244.

[10] *Acceleration Parameters*. Police Radar Information Center [online]. [cit. 2022-03-05]. Dostupné z: https://copradar.com/chapts/references/acceleration.html

[11] *Tlak vzduchu přepočtený na hladinu moře*. ČHMÚ [online]. [cit. 2022-03-06]. Dostupné online: https://www.chmi.cz/aktualni-situace/aktualni-stav-pocasi/ceska-republika/stanice/profesionalni-stanice/mapy/tlak-vzduchu

[12] *elBlesk tříkolka – elektromobil pro každého*. elBlesk [online]. [cit. 2022-01-22]. Dostupné z: https://www.elblesk.cz/elblesk-trikolka/

[13] KOZUBÍK, Libor. *Technický pokrok v oblasti akumulátorových baterií* [přednáška]. [online]. [cit. 2022-04-06]. Brno, 2. ročník semináře Perspektivy elektromobility,

19. března 2013. Dostupné z: http://www.odbornecasopisy.cz/data-ftp-user/konference/2013/emob/07-amper-2013-pemb-kozubik-ibm.pdf

[14] CETL, Tomáš. *Lithiové akumulátory velkých výkonů a jejich použití*. Časopis ELEKTRO [online]. [cit. 2022-04-06]. Praha, 2005, č. 12/2005. Dostupné z: http://www.odbornecasopisy.cz/elektro/casopis/tema/lithiove-akumulatory-velkych-vykonu-a-jejich-pouziti--13384

[15] HO, Vincent. *Li-ion Battery and Gauge Introduction*. Richtek Technology [online]. [cit. 2022-04-06]. Dostupné z: https://www.richtek.com/Design%20Support/Technical%20Document/AN024

[16] *How does Internal Resistance affect Performance*? Battery University [online]. [cit. 2022-04-12]. Dostupné z: https://batteryuniversity.com/article/how-does-internal-resistance-affect-performance

[17] *Proč používat LiFePO baterie?* ABCTECH [online]. [cit. 2022-04-16]. Dostupné z: https://www.abctech.cz/default.asp?show=wm&wmpart=article&wmaid=87

[18] *Typy baterií a technologie*. Elektrosport [online]. [cit. 2022-04-16]. Dostupné z: https://elektrosport.cz/kontakty-a-informace/vse-o-elektrokolech/baterie/typy-baterii-a-technologie

[19] *Přidání nových připojení - Visual Studio (Windows)*. Microsoft [online]. [cit. 2022-05-22]. Dostupné z: https://docs.microsoft.com/cs-cz/visualstudio/data-tools/add-new-connections

Přílohy

Příloha #1

Varianta 1 – Měřící program v jazyce MicroPython pro Raspberry Pi Pico, sběr okamžitých hodnot sledovaných veličin 1× během intervalu 500 ms

```
import time
from machine import Pin, I2C, UART, RTC, Timer
import onewire
import ds1302
import ds18x20
from ads1115 import *
from mpu9250 import MPU9250
from bmp180 import BMP180
from math import pi, sqrt, tan, atan2, sin, copysign

# I2C0 init ... MPU
i2c0_id = 0
i2c0_scl = Pin(9)
i2c0 sda = Pin(8)
i2c0 = I2C(id=i2c0_id, scl=i2c0_scl, sda=i2c0_sda)

# I2C1 init ... ADC
i2c1_id = 1
i2c1_scl = Pin(15)
i2c1_sda = Pin(14)
i2c1 = I2C(id=i2c1_id, scl=i2c1_scl, sda=i2c1_sda)

# 1-Wire init
ow_pin = Pin(22)
ow_bus = onewire.OneWire(ow_pin)

# UART0 init
uart0_tx = Pin(12)
uart0_rx = Pin(13)
uart0_rate = 9600
uart0 = UART(0, baudrate=uart0_rate, tx=uart0_tx, rx=uart0_rx)

# RTC init
clk_pin = Pin(18)
dio_pin = Pin(16)
cs_pin = Pin(17)
rtc = ds1302.DS1302(clk_pin, dio_pin, cs_pin)

# Onboard RTC init
board_rtc = RTC()
```

```
# Temperature sensors init
temp1_addr = [0x28, 0xD3, 0x1A, 0x63, 0x4B, 0x20, 0x01, 0x35] # Battery temp. (R)
temp2_addr = [0x28, 0x2F, 0xD9, 0x65, 0x4B, 0x20, 0x01, 0x5F] # Ambient temp. (L)
temp1_sens = ds18x20.DS18X20(ow_bus, temp1_addr)
temp2_sens = ds18x20.DS18X20(ow_bus, temp2_addr)
temp1_sens.set_res(9) # 9-bit resolution
temp2_sens.set_res(9)

# ADC init
ads = ADS1115(address=0x48, i2c=i2c1) # Differential mode, 16-bit res.
ads.setVoltageRange_mV(ADS1115_RANGE_4096) # Voltage range, actual limited by VDD
ads.setMeasureMode(ADS1115_SINGLE) # Single-shot mode
vdiv_R1 = 428 # Main resistor (bigger one)
vdiv_R2 = 10 # Output resistor (smaller one)
vdiv_const = (vdiv_R1+vdiv_R2)/vdiv_R2 # Voltage divider ratio
hall_const = 1/(0.625/100.0) # Hall-sensor const, 625 mV ~ 100 A rated current

# MPU9250 init
mpu = MPU9250(i2c0)
acc_off_xyz = [0.0, 0.0, 0.0] # Acc (X, Y, Z) calibration offset in m.s^(-2)

# BMP180 init
bmp = BMP180(i2c0)
bmp.oversample_sett = 2
bmp.baseline = 103400 # Local barometric pressure

# Essential
sampling = 500 # Period in ms
milisecs = 0 # Additional RTC occuracy in ms
energy = 0 # W = P.t in (W.h)
telemetry = False # Indicates user telemetry start request
calibration = False # Indicates successful calibration
calduration = 15 # Calibration duration in sec
bt_out = "" # Data output message
tim = machine.Timer() # Timer for periodic reading sensors data

def get_synced_time():
    ''' Performs date and time sync '''
    utc = rtc.date_time()
    # Sets current time as system time
    board rtc.datetime((utc[0], utc[1], utc[2], utc[3], utc[4], utc[5], utc[6], 0))
    # Output format: [year, month, day, weekday, hours, minutes, seconds, miliseconds]
    return board_rtc.datetime()

def get_timestamp():
    ''' Returns current timestamp '''
    # Always use system time
    utc = board_rtc.datetime()
```

```
    # Output format: [year, month, day, hours, minutes, seconds]
    return (utc[0], utc[1], utc[2], utc[4], utc[5], utc[6])

def get_voltage_chxx(channel):
    ''' Returns voltage between specified channels '''
    ads.setCompareChannels(channel)
    ads.startSingleMeasurement()
    while ads.isBusy():
        pass
    voltage = ads.getResult_V()
    return voltage

def get_mpu_data():
    ''' Returns acceleration data '''
    # Accelerator data
    acc = mpu.acceleration
    acc_x = acc[0]-acc_off_xyz[0]
    acc_y = acc[1]-acc_off_xyz[1]
    acc_z = acc[2]-acc_off_xyz[2]
    return (acc_x, acc_y, acc_z)

def get_altitude():
    ''' Returns altitude from pressure sensor '''
    return bmp.altitude

def get_sensors_data(t):
    ''' Periodically obtains sensors data and prepares them for transfer '''
    if (telemetry == True):
        global milisecs
        mse = milisecs
        milisecs = abs(milisecs - 500)
        ax, ay, az = get_mpu_data()
        # Small pause before next reading from the same I2C bus
        time.sleep(0.02)
        al = get_altitude()
        ye, mo, da, ho, mi, se = get_timestamp()
        u1 = get_voltage_chxx(ADS1115_COMP_0_1)*vdiv_const
        i1 = get_voltage_chxx(ADS1115_COMP_2_3)*hall_const
        global energy
        energy += u1*i1/(sampling/1000.0)/3600.0
        eg = energy
        temp1 sens.convert temp()
        t1 = temp1_sens.read_temp()
        # Small pause before next reading from OW bus
        time.sleep(0.02)
        temp2_sens.convert_temp()
        t2 = temp2_sens.read_temp()
```

```
        bt_out = "$DATA,%04d-%02d-
%02d %02d:%02d:%02d.%03d,%.2f,%.2f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f;" % (ye, mo, da, ho,
mi, se, mse, t1, t2, u1, i1, eg, ax, ay, az, al)
        send_over_bt(bt_out)

def calibrate(cal_time):
    ''' Performs calibration of MPU sensor for specified time '''
    # Calibration of accelerometer
    send_over_bt("$INFO,Calibration of accelerometer in progress...;")
    send_over_bt("$INFO,%d sec remaining.;" % cal_time)
    max_readings = (int)(cal_time/(sampling/10)*1000)
    raccx = [0.0]*max_readings
    raccy = [0.0]*max_readings
    raccz = [0.0]*max_readings
    ye, mo, da, ho, mi, se = get_timestamp()
    file_name = "calib" + str(ye+mo+da+ho+mi+se) + ".txt"
    file = open(file_name, 'w')
    file.write("acc-x\t\tacc-y\t\tacc-z\r\n")
    file.write("----------------------------------------\r\n")
    # Obtain calibration data from continues readings
    for r in range(max_readings):
        rest = get_mpu_data()
        # Fill rest data arrays
        raccx[r] = rest[0]
        raccy[r] = rest[1]
        raccz[r] = rest[2]
        # Write rest data to file
        file.write("%+.3f\t\t%+.3f\t\t%+.3f\r\n" % (rest[0], rest[1], rest[2]))
        if (r % int(1000/(sampling/10)) == 0):
            print("pending: %d / %d sec" % (r/int(1000/(sampling/10)), cal_time))
        time.sleep((sampling/10)/1000)
    # Rest data postprocessing
    # Average values calculation
    raccx_avg = sum(raccx)/max_readings
    raccy_avg = sum(raccy)/max_readings
    raccz_avg = sum(raccz)/max_readings
    # Filter outstanding values
    non0 = [max_readings, max_readings, max_readings] # Non-zero values count
    # First order filter
    acctolL = 0.9 # Accelerometer lower tolerance
    acctolH = 1.1 # Accelerometer upper tolerance
    for p in range(max readings):
        if (abs(raccx[p]) < acctolL*abs(raccx_avg) or abs(raccx[p]) > acctolH*abs(raccx_avg
)):
            raccx[p] = 0.0
            non0[0] -= 1
        if (abs(raccy[p]) < acctolL*abs(raccy_avg) or abs(raccy[p]) > acctolH*abs(raccy_avg
)):
            raccy[p] = 0.0
```

```
            non0[1] -= 1
        if (abs(raccz[p]) < acctolL*abs(raccz_avg) or abs(raccz[p]) > acctolH*abs(raccz_avg
)):
            raccz[p] = 0.0
            non0[2] -= 1
    if (non0[0] > 0):
        raccx_avg = sum(raccx)/non0[0]
    if (non0[1] > 0):
        raccy_avg = sum(raccy)/non0[1]
    if (non0[2] > 0):
        raccz_avg = sum(raccz)/non0[2]
    # Define offsets
    global acc_off_xyz
    acc_off_xyz = [raccx_avg, raccy_avg, raccz_avg]
    file.write("----------------------------------------\r\n")
    file.write("Accelerometer offsets...\r\n")
    file.write("acc-x: -(%+.3f) m.s^(-2)\r\n" % acc_off_xyz[0])
    file.write("acc-y: -(%+.3f) m.s^(-2)\r\n" % acc_off_xyz[1])
    file.write("acc-z: -(%+.3f) m.s^(-2)\r\n" % acc_off_xyz[2])
    file.write("\r\nCalibration took in place: %04d-%02d-
%02d %02d:%02d:%02d\r\n\r\n" % (ye, mo, da, ho, mi, se))
    file.close()
    send_over_bt("$INFO,Calibration successful.;")
    send_over_bt("$INFO,Following offsets will apply:;")
    send_over_bt("$INFO,aX: -(%+.3f), aY: -(%+.3f), aZ: -(%+.3f) m.s^(-
2);" % (acc_off_xyz[0], acc_off_xyz[1], acc_off_xyz[2]))

def check_bt_incoming():
    ''' Scanning for any incoming bluetooth serial data '''
    bt_in = uart0.readline()
    if (bt_in != None):
        print("Data received: %s" % bt_in)
        command = bt_in.decode("utf-8")
        global calibration
        global telemetry
        if (command.find("$CAL;") >= 0 and calibration == False):
            # Calibration user request received
            send_over_bt("$INFO,Obtaining current date and time...;")
            dt = get_synced_time() # Load current date and time from external RTC
            send_over_bt("$INFO,Current date and time: %04d-%02d-
%02d %02d:%02d:%02d.;" % (dt[0], dt[1], dt[2], dt[4], dt[5], dt[6]))
            calibrate(calduration)
            calibration = True
        if (command.find("$START;") >= 0 and telemetry == False):
            # Telemetry user request received
            # Synchronize with RTC to get additional occuracy in ms
            secs = get_timestamp()[5]
            if (secs < 58):
                secs += 2
```

```
            else:
                secs = secs + 2 - 60
            while (get_timestamp()[5] < secs):
                time.sleep(0.02)
            # Setting up timers for periodical data logging
            tim.init(mode=Timer.PERIODIC, period=sampling, callback=get_sensors_data)
            telemetry = True
        if (command.find("$STOP;") >= 0 and telemetry == True):
            # Stop user request received
            telemetry = False
            time.sleep(2*sampling/1000)
            send_over_bt("$INFO,Telemetry service stopped.;")
            tim.deinit()
            return False
    # Unknown or blank user requests
    # True ... Continue scanning
    return True

def send_over_bt(data):
    ''' Sends data over bluetooth '''
    uart0.write((data + '\r\n').encode())
    #print('Data sent: ', data)

# Waiting for user input...
while (check_bt_incoming() == True):
    time.sleep(sampling/1000)
```

Příloha #2

Varianta 2 – Měřící program v jazyce MicroPython pro Raspberry Pi Pico, výpočet TrueRMS hodnot napětí a proudu za interval 500 ms, následováno 100 ms pauzou pro odečet okolní teploty

```
import time
from machine import Pin, I2C, UART, RTC, Timer
import onewire
import ds1302
import ds18x20
from ads1115 import *
from mpu9250 import MPU9250
from bmp180 import BMP180
from math import sqrt

# Local conditions
local_hpa = 1027.3 # Local barometric pressure in hPa
local_alt = 351.0 # Start point altitude in metres, or 0

# I2C0 init ... MPU
i2c0_id = 0
i2c0_scl = Pin(9)
i2c0_sda = Pin(8)
i2c0 = I2C(id=i2c0_id, scl=i2c0_scl, sda=i2c0_sda)

# I2C1 init ... ADC
i2c1_id = 1
i2c1_scl = Pin(15)
i2c1_sda = Pin(14)
i2c1 = I2C(id=i2c1_id, scl=i2c1_scl, sda=i2c1_sda)

# 1-Wire init
ow_pin = Pin(22)
ow_bus = onewire.OneWire(ow_pin)

# UART0 init
uart0_tx = Pin(12)
uart0_rx = Pin(13)
uart0_rate = 9600
uart0 = UART(0, baudrate=uart0_rate, tx=uart0_tx, rx=uart0_rx)

# RTC init
clk_pin = Pin(18)
dio_pin = Pin(16)
cs_pin = Pin(17)
rtc = ds1302.DS1302(clk_pin, dio_pin, cs_pin)
```

```
# Onboard RTC init
board_rtc = RTC()

# Temperature sensors init
temp2_addr = [0x28, 0x2F, 0xD9, 0x65, 0x4B, 0x20, 0x01, 0x5F] # Ambient temp. (L)
temp2_sens = ds18x20.DS18X20(ow_bus, temp2_addr)
temp2_sens.set_res(9) # 9-bit resolution

# ADC init
ads = ADS1115(address=0x48, i2c=i2c1) # Differential mode, 16-bit res.
ads.setVoltageRange_mV(ADS1115_RANGE_4096) # Voltage range, actual limited by VDD
ads.setMeasureMode(ADS1115_SINGLE) # Single-shot mode
vdiv_R1 = 428 # Main resistor (bigger one)
vdiv_R2 = 10 # Output resistor (smaller one)
vdiv_const = (vdiv_R1+vdiv_R2)/vdiv_R2 # Voltage divider ratio
hall_const = 1/(0.625/100.0) # Hall-sensor const, 625 mV ~ 100 A rated current

# MPU9250 init
mpu = MPU9250(i2c0)
acc_off_xyz = [0.0, 0.0, 0.0] # Acc (X, Y, Z) calibration offset in m.s^(-2)

# BMP180 init
bmp = BMP180(i2c0)
bmp.oversample_sett = 2
bmp.baseline = local_hpa*100 # Local barometric pressure in Pa

# Essential
sampling = 500 # Period in ms
millis0 = 0 # Millis synchronized with machine.RTC
volts_rms = 0.0 # TrueRMS value in (V)
volts_min = 0.0 # Min value in (V)
volts_max = 0.0 # Max value in (V)
amps_rms = 0.0 # TrueRMS value in (A)
amps_min = 0.0 # Min value in (A)
amps_max = 0.0 # Max value in (A)
energy = 0.0 # W = P.t in (W.h)
acc_max_xyz = [0.0, 0.0, 0.0] # Max acceleration values during sampling period
acc_min_xyz = [0.0, 0.0, 0.0] # Min acceleration values...
acc_avg_xyz = [0.0, 0.0, 0.0] # Avg acceleration values...
bmp_offset = 0.0
sampled_values = 0 # Total sampled values
telemetry = False # Indicates user telemetry start request
calibration = False # Indicates successful calibration
calduration = 30 # Calibration duration in sec
bt_out = "" # Data output message

def get_synced_time():
    ''' Performs date and time sync '''
    utc = rtc.date_time()
```

```
    # Sets current time as system time
    board_rtc.datetime((utc[0], utc[1], utc[2], utc[3], utc[4], utc[5], utc[6], 0))
    # Output format: [year, month, day, weekday, hours, minutes, seconds, milliseconds]
    return board_rtc.datetime()

def get_timestamp():
    ''' Returns current timestamp '''
    # Always use system time
    utc = board_rtc.datetime()
    # Output format: [year, month, day, hours, minutes, seconds]
    return (utc[0], utc[1], utc[2], utc[4], utc[5], utc[6])

def get_voltage_chxx(channel):
    ''' Returns voltage between specified channels '''
    ads.setCompareChannels(channel)
    ads.startSingleMeasurement()
    while ads.isBusy():
        pass
    voltage = ads.getResult_V()
    return voltage

def get_mpu_data():
    ''' Returns acceleration data '''
    # Accelerator data
    acc = mpu.acceleration
    acc_x = acc[0]-acc_off_xyz[0]
    acc_y = acc[1]-acc_off_xyz[1]
    acc_z = acc[2]-acc_off_xyz[2]
    return (acc_x, acc_y, acc_z)

def get_altitude():
    ''' Returns altitude from pressure sensor '''
    return bmp.altitude

def get_sensors_data(period_start):
    ''' Collects and formats sensors data for transfer '''
    if (telemetry == True):
        ye, mo, da, ho, mi, se = get_timestamp()
        millis_diff = time.ticks_diff(period_start, millis0)
        ms = millis_diff%1000
        t1 = 0.0
        temp2 sens.convert temp()
        t2 = temp2_sens.read_temp()
        u1 = volts_rms
        i1 = amps_rms
        ax, az, ay = acc_avg_xyz
        al = get_altitude()-bmp_offset
        millis2 = time.ticks_ms()
        millis_diff2 = time.ticks_diff(millis2, period_start)
```

```
        global energy
        energy += volts_rms*amps_rms*(millis_diff2/1000.0)/3600.0 # Estimated energy in kWh
, expecting no major changes while reading other sensors data
        eg = energy
        bt_out1 = "$DATA1,%04d-%02d-
%02d %02d:%02d:%02d.%03d,%.2f,%.2f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f,%.3f;" % (ye, mo, da, ho,
mi, se, ms, t1, t2, u1, i1, eg, ax, ay, az, al)
        send_over_bt(bt_out1)
        umi = volts_min
        uma = volts_max
        imi = amps_min
        ima = amps_max
        axmi, aymi, azmi = acc_min_xyz
        axma, ayma, azma = acc_max_xyz
        bt_out2 = "$DATA2,%04d-%02d-
%02d %02d:%02d:%02d.%03d,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f;" % (ye, mo, da, ho, mi, se, ms
, umi, uma, imi, ima, axmi, axma, aymi, ayma, azmi, azma)
        send_over_bt(bt_out2)

def calibrate(cal_time):
    ''' Performs calibration of MPU925X sensor '''
    # Altitude calibration
    global bmp_offset
    if (int(local_alt) != 0):
        bmp_offset = get_altitude()-local_alt
    send_over_bt("$INFO,Current altitude calibrated to: %.02f m.;" % local_alt)
    # Calibration of accelerometer
    send_over_bt("$INFO,Calibration of accelerometer in progress...;")
    send_over_bt("$INFO,%d sec remaining.;" % cal_time)
    max_readings = (int)(cal_time/(sampling/10)*1000)
    raccx = [0.0]*max_readings
    raccy = [0.0]*max_readings
    raccz = [0.0]*max_readings
    ye, mo, da, ho, mi, se = get_timestamp()
    file_name = "calib" + str(ye+mo+da+ho+mi+se) + ".txt"
    file = open(file_name, 'w')
    file.write("acc-x\t\tacc-y\t\tacc-z\r\n")
    file.write("----------------------------------------\r\n")
    # Obtain calibration data from continues readings
    for r in range(max_readings):
        rest = get_mpu_data()
        # Fill rest data arrays
        raccx[r] = rest[0]
        raccy[r] = rest[1]
        raccz[r] = rest[2]
        # Write rest data to file
        file.write("%+.3f\t\t%+.3f\t\t%+.3f\r\n" % (rest[0], rest[1], rest[2]))
        if (r % int(1000/(sampling/10)) == 0):
            print("pending: %d / %d sec" % (r/int(1000/(sampling/10)), cal_time))
```

```
        time.sleep((sampling/10)/1000)
    # Rest data postprocessing
    # Average values calculation
    raccx_avg = sum(raccx)/max_readings
    raccy_avg = sum(raccy)/max_readings
    raccz_avg = sum(raccz)/max_readings
    # Filter outstanding values
    non0 = [max_readings, max_readings, max_readings] # Non-zero values count
    # First order filter
    acctolL = 0.9 # Accelerometer lower tolerance
    acctolH = 1.1 # Accelerometer upper tolerance
    for p in range(max_readings):
        if (abs(raccx[p]) < acctolL*abs(raccx_avg) or abs(raccx[p]) > acctolH*abs(raccx_avg
)):
            raccx[p] = 0.0
            non0[0] -= 1
        if (abs(raccy[p]) < acctolL*abs(raccy_avg) or abs(raccy[p]) > acctolH*abs(raccy_avg
)):
            raccy[p] = 0.0
            non0[1] -= 1
        if (abs(raccz[p]) < acctolL*abs(raccz_avg) or abs(raccz[p]) > acctolH*abs(raccz_avg
)):
            raccz[p] = 0.0
            non0[2] -= 1
    if (non0[0] > 0):
        raccx_avg = sum(raccx)/non0[0]
    if (non0[1] > 0):
        raccy_avg = sum(raccy)/non0[1]
    if (non0[2] > 0):
        raccz_avg = sum(raccz)/non0[2]
    # Define offsets
    global acc_off_xyz
    acc_off_xyz = [raccx_avg, raccy_avg, raccz_avg]
    file.write("----------------------------------------\r\n")
    file.write("Accelerometer offsets...\r\n")
    file.write("acc-x: -(%+.3f) m.s^(-2)\r\n" % acc_off_xyz[0])
    file.write("acc-y: -(%+.3f) m.s^(-2)\r\n" % acc_off_xyz[1])
    file.write("acc-z: -(%+.3f) m.s^(-2)\r\n" % acc_off_xyz[2])
    file.write("\r\nBarometer offset...\r\n")
    file.write("altitude: -(%+.2f) m\r\n" % bmp_offset)
    file.write("\r\nCalibration took in place: %04d-%02d-
%02d %02d:%02d:%02d\r\n\r\n" % (ye, mo, da, ho, mi, se))
    file.close()
    send_over_bt("$INFO,Calibration successful.;")
    send_over_bt("$INFO,Following offsets will apply:;")
    send_over_bt("$INFO,aX: -(%+.3f), aY: -(%+.3f), aZ: -(%+.3f) m.s^(-2), alt: -
(%+.2f) m;" % (acc_off_xyz[0], acc_off_xyz[1], acc_off_xyz[2], bmp_offset))

def check_bt_incoming():
```

```
    ''' Scanning for any incoming bluetooth serial data '''
    bt_in = uart0.readline()
    if (bt_in != None):
        print("Data received: %s" % bt_in)
        command = bt_in.decode("utf-8")
        global calibration
        global telemetry
        if (command.find("$CAL;") >= 0 and calibration == False):
            # Calibration user request received
            send_over_bt("$INFO,Obtaining current date and time...;")
            dt = get_synced_time() # Load current date and time from external RTC
            # Millis sync in next 2 secs
            secs = get_timestamp()[5]
            if (secs < 58):
                secs += 2
            else:
                secs = secs + 2 - 60
            while (get_timestamp()[5] < secs):
                time.sleep(0.001)
            millis0 = time.ticks_ms() # Millis zero point
            send_over_bt("$INFO,Current date and time: %04d-%02d-
%02d %02d:%02d:%02d.;" % (dt[0], dt[1], dt[2], dt[4], dt[5], dt[6]))
            calibrate(calduration)
            calibration = True
        if (command.find("$START;") >= 0 and telemetry == False):
            # Telemetry user request received
            telemetry = True
        if (command.find("$STOP;") >= 0 and telemetry == True):
            # Stop user request received
            telemetry = False
            send_over_bt("$INFO,Telemetry service stopped.;")
            return False
    # Unknown or blank user requests
    # True ... Continue scanning
    return True

def send_over_bt(data):
    ''' Sends data over bluetooth '''
    uart0.write((data + '\r\n').encode())
    #print('Data sent: ', data)

# Loop
while (True):
    millis1 = time.ticks_ms()
    while (time.ticks_ms() - millis1 < sampling):
        ax, ay, az = get_mpu_data()
        acc_avg_xyz[0] += ax
        acc_avg_xyz[1] += ay
        acc_avg_xyz[2] += az
```

```
        if (sampled_values > 0):
            # Min acceleration values
            if(acc_min_xyz[0] > ax):
                acc_min_xyz[0] = ax
            if(acc_min_xyz[1] > ay):
                acc_min_xyz[1] = ay
            if(acc_min_xyz[2] > az):
                acc_min_xyz[2] = az
            # Max acceleration values
            if(acc_max_xyz[0] < ax):
                acc_max_xyz[0] = ax
            if(acc_max_xyz[1] < ay):
                acc_max_xyz[1] = ay
            if(acc_max_xyz[2] < az):
                acc_max_xyz[2] = az
        else:
            # Save first value only
            acc_min_xyz[0] = ax
            acc_min_xyz[1] = ay
            acc_min_xyz[2] = az
            acc_max_xyz[0] = ax
            acc_max_xyz[1] = ay
            acc_max_xyz[2] = az
        u1 = get_voltage_chxx(ADS1115_COMP_0_1)
        volts_rms += (u1*u1)
        if (sampled_values > 0):
            # Min voltage value
            if(volts_min > u1):
                volts_min = u1
            # Max voltage value
            if(volts_max < u1):
                volts_max = u1
        else:
            # Save first value only
            volts_min = u1
            volts_max = u1
        i1 = get_voltage_chxx(ADS1115_COMP_2_3)
        amps_rms += (i1*i1)
        if (sampled_values > 0):
            # Min drain value
            if(amps_min > i1):
                amps min = i1
            # Max drain value
            if(amps_max < i1):
                amps_max = i1
        else:
            # Save first value only
            amps_min = i1
            amps_max = i1
```

```
        #rms continue
        sampled_values += 1
        time.sleep(0.002) # 2 ms reading period
    acc_avg_xyz = [acc_avg_xyz[0]/sampled_values, acc_avg_xyz[1]/sampled_values, acc_avg_xyz[2]/sampled_values]
    volts_rms = sqrt(volts_rms/sampled_values)*vdiv_const
    amps_rms = sqrt(amps_rms/sampled_values)*hall_const
    get_sensors_data(period_start=millis1)
    volts_rms = 0.0
    amps_rms = 0.0
    sampled_values = 0
    check_bt_incoming()
    time.sleep(0.01)
```

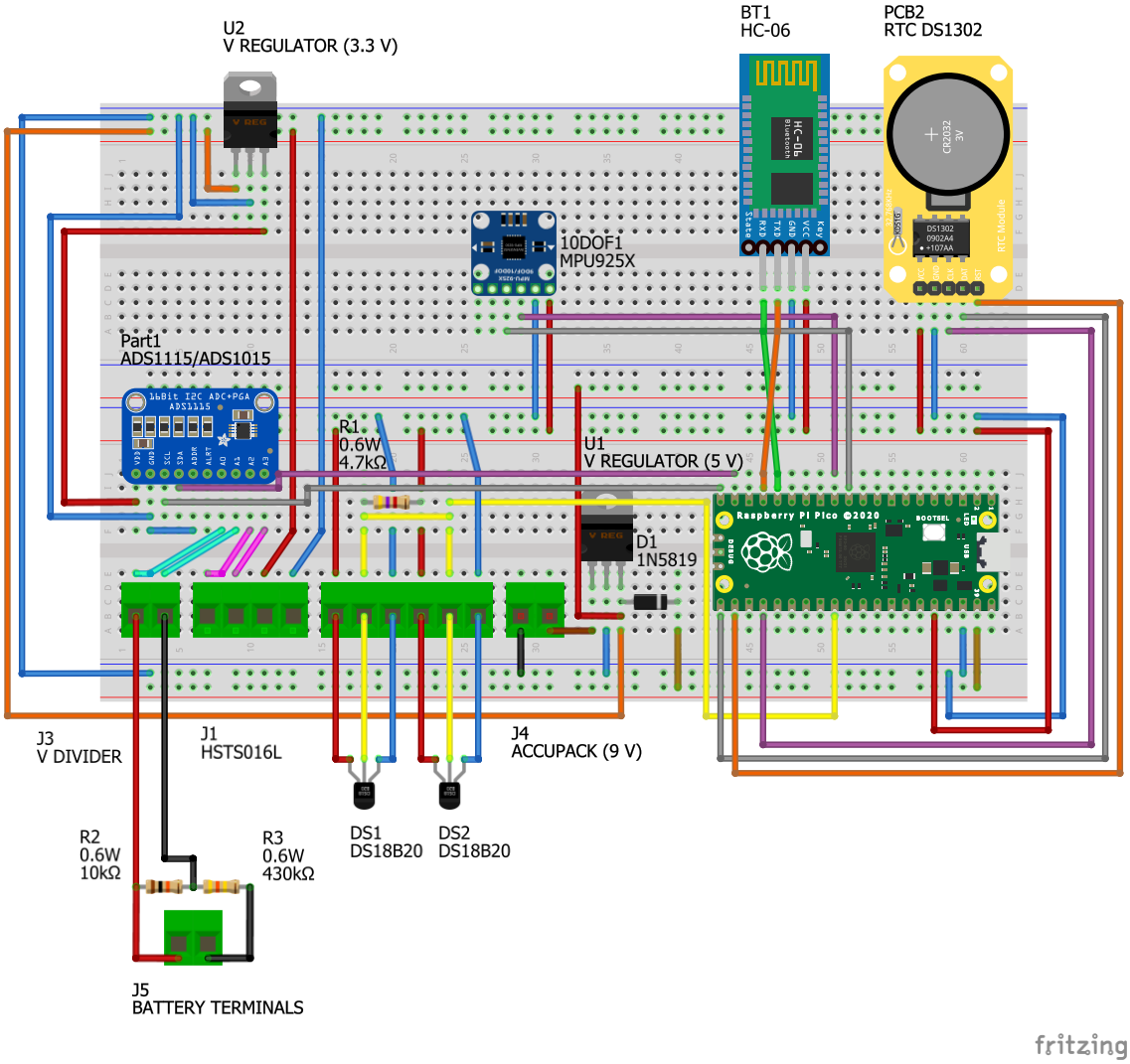
Příloha #3

Skript v jazyce MicroPython pro časovou synchronizaci RTC modulu, spustitelný pomocí Thonny IDE

```
import time
from machine import Pin, RTC
import ds1302

# RTC init
clk_pin = Pin(18)
dio_pin = Pin(16)
cs_pin = Pin(17)
rtc = ds1302.DS1302(clk_pin, dio_pin, cs_pin)

# RPi RTC init
board_rtc = RTC()

# get current datetime from IDE
dt = time.localtime(time.time())
# (year, month, month_day, hours, minutes, seconds, week_day, year_day)
dt_utc = [dt[0], dt[1], dt[2], dt[6], dt[3], dt[4], dt[5]]
# (year, month, month_day, week_day, hours, minutes, seconds)
# sets current (GMT+1) time to UTC
if (dt_utc[4] > 0):
    dt_utc[4] -= 1
else:
    dt_utc[4] = 23
# save datetime to external RTC
rtc.date_time(dt_utc)

# load datetime from external RTC
utc = rtc.date_time()
# and use it as Pico UTC datetime
board_rtc.datetime((utc[0], utc[1], utc[2], utc[3], utc[4], utc[5], utc[6], 0))

# print every sec Pico datetime
while 1:
    print(board_rtc.datetime())
    time.sleep(1)
```

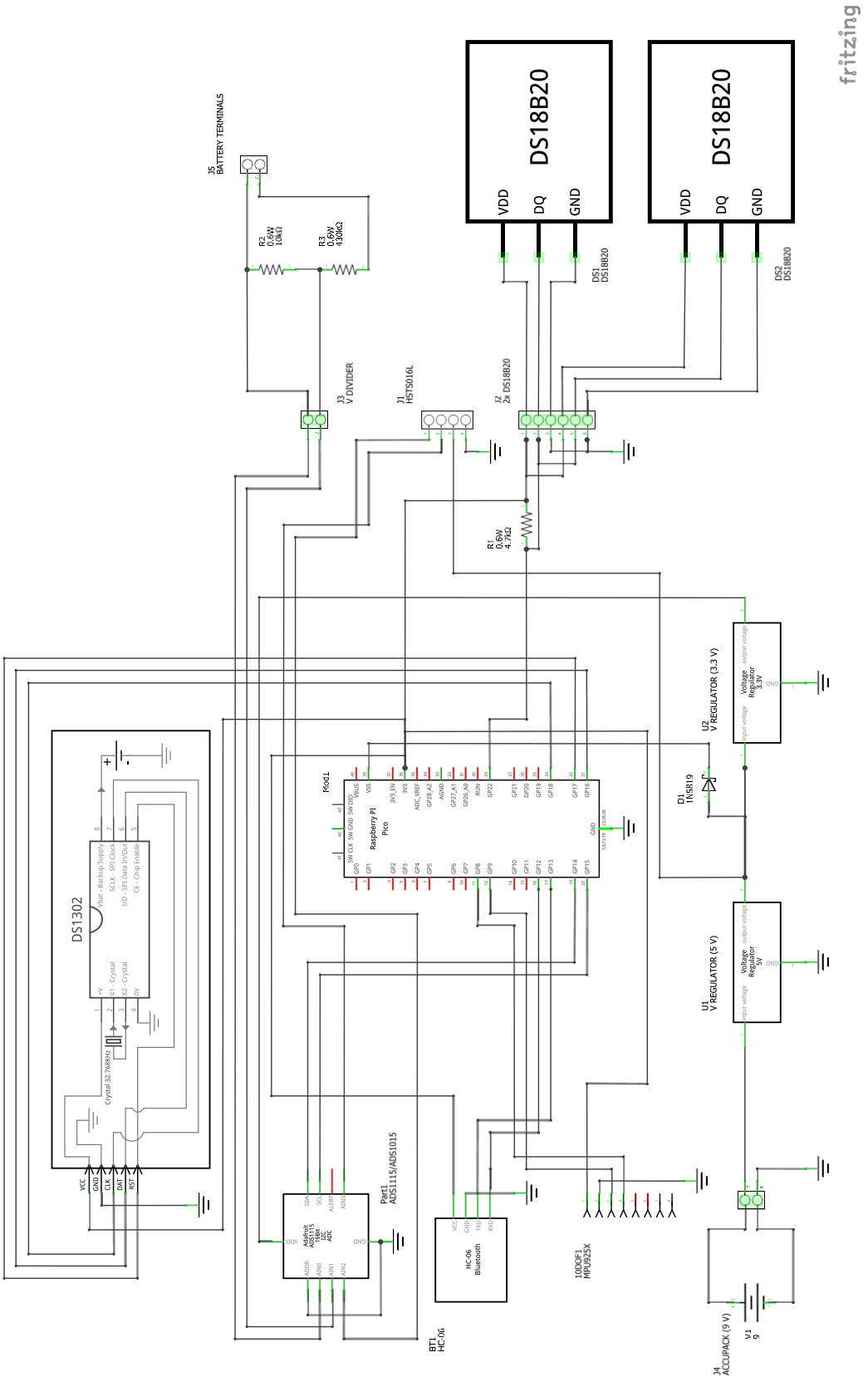
Příloha #4

Situační schéma zapojení bezdrátového telemetrického zařízení



Příloha #5

Obvodové schéma zapojení bezdrátového telemetrického zařízení



Příloha #6

Pomocný program v jazyce C# (Konzolová aplikace pro Windows) ke konverzi NMEA zpráv ve formátu TXT (záznamu sériové komunikace z GPS modulu) do souboru CSV

```
using NmeaParser;
using NmeaParser.NmeaLines;
using System;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;

namespace GpsToCsv
{
    class Program
    {
        string outputFile;
        public static void Main(string[] args)
        {
            // Set locale to en-US
            CultureInfo.DefaultThreadCurrentCulture = new CultureInfo("en-US");
            Console.OutputEncoding = System.Text.Encoding.UTF8;
            var instance = new Program();
            string fileName;
            // Parse source data file name
            do
            {
                Console.WriteLine("Type in data filename and hit enter...");
                fileName = Console.ReadLine().Trim();
            }
            while (fileName.IndexOf('.') < 0);
            // Get current executable's dir
            string appPath = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location
);
            // Input file should be placed there
        string fullFilePath = string.Format("{0}\\{1}", appPath, fileName);
            if (!File.Exists(fullFilePath))
                // Source file doesn't exist -> EXIT
                Console.WriteLine("The source file doesn't exist!");
            else {
                // Output filename using current datetime
                DateTime dt = DateTime.Now;
                instance.outputFile = string.Format("{0}\\out{1}.csv", appPath, (dt.Hour +
dt.Minute + dt.Second).ToString());
                if (!File.Exists(instance.outputFile))
                {
                    // Create a file to write to.
```

```
                    using (StreamWriter sw = File.CreateText(instance.outputFile))
                    {
                        // Add table header
                        sw.WriteLine("cas_razitko,zem_sirka,zem_delka,nadm_vyska,rychlost,g
ps_stav,vzdalenost");
                    }
                }
                Console.WriteLine("Reading the source file...");
                string[] lines = File.ReadAllLines(fullFilePath);
                // Read data file line by line
                for (int i = 0; i < lines.Length; i++)
                {
                    if (i % 100 == 0)
                        Console.WriteLine("{0}/{1}", i, lines.Length);
                    if (lines[i].StartsWith("$"))
                        instance.Parse_GpsData(lines[i]);
                }
                Console.WriteLine("Tabular data succesfully saved to:{0}{1}", Environment.N
ewLine, instance.outputFile);
            }
            // Keep the console window open
            Console.WriteLine("Press any key to exit...");
            Console.ReadKey();
        }

        string[] GPSDataTemp = new string[7]; // date, time, lat, lon, alt, speed, fix
        int[] GPSFullData = new int[7] { 0, 0, 0, 0, 0, 0, 0 }; // 1 per valid single data
        int earthRadius = 6371;
        double last_lat = 0; // not available before first succesful GPS fix
        double last_lon = 0;
        double distanceByGPS = 0; // in km
        public void Parse_GpsData(string NmeaLine)
        {
            try
            {
                if (NmeaLine.Contains("RMC")) // GPRMC, GNRMC
                {
                    var nmeaMsg = new NmeaLineFactory().ParseLine(NmeaLine);
                    var rmcMsg = (RmcLine)nmeaMsg;
                    DateTime GPSDt = rmcMsg.FixTime;
                    string gps_date = GPSDt.ToString("yyyy-MM-dd");
                    string gps time = GPSDt.ToString("HH:mm:ss");
                    string gps_sped = string.Format("{0:0.##}", rmcMsg.GeoCoordinate.Speed)
;
                    // Data with new timestamp? => save previous to file
                    if (GPSDataTemp[1] != gps_time)
                    {
                        Publish_GpsData(rmcMsg.GeoCoordinate.Latitude, rmcMsg.GeoCoordinate
.Longitude);
```

```
                    }
                    GPSFullData[0] = 1;
                    GPSFullData[1] = 1;
                    GPSFullData[5] = 1;
                    GPSDataTemp[0] = gps_date;
                    GPSDataTemp[1] = gps_time;
                    GPSDataTemp[5] = gps_sped;
                }
                else if (NmeaLine.Contains("GGA")) // GPGGA, GNGGA
                {
                    var nmeaMsg = new NmeaLineFactory().ParseLine(NmeaLine);
                    var ggaMsg = (GgaLine)nmeaMsg;
                    string gps_time = ggaMsg.FixTime.ToString();
                    string gps_lati = string.Format("{0:0.#######}", ggaMsg.Position.Latitu
de);
                    string gps_long = string.Format("{0:0.#######}", ggaMsg.Position.Longit
ude);
                    string gps_alti = string.Format("{0:0.##}", ggaMsg.Position.Altitude);
                    // Data with new timestamp? => save previous to file
                    if (GPSDataTemp[1] != gps_time)
                    {
                        Publish_GpsData(ggaMsg.Position.Latitude, ggaMsg.Position.Longitude
);
                    }
                    GPSFullData[1] = 1;
                    GPSFullData[2] = 1;
                    GPSFullData[3] = 1;
                    GPSFullData[4] = 1;
                    GPSDataTemp[1] = gps_time;
                    GPSDataTemp[2] = gps_lati;
                    GPSDataTemp[3] = gps_long;
                    GPSDataTemp[4] = gps_alti;
                }
                else if (NmeaLine.Contains("GSA")) // GPGSA, GNGSA, BDGSA
                {
                    var nmeaMsg = new NmeaLineFactory().ParseLine(NmeaLine);
                    var gsaMsg = (GsaLine)nmeaMsg;
                    string gps_fix = string.Format("{0}", gsaMsg.Fix);
                    if (gps_fix != "NotAvailable")
                        GPSFullData[6] = 1;
                    GPSDataTemp[6] = gps_fix;
                }
            }
            catch (Exception ex)
            {
                // Invalid checksum of NMEA message
                if (ex is System.ArgumentException) ;
                // NMEA message data parsing error
                if (ex is System.FormatException) ;
```

```
                // Unsupported type of NMEA message
                if (ex is System.ArgumentOutOfRangeException) ;
            }
        }

        public bool Publish_GpsData(double new_lat, double new_lon)
        {
            try
            {
                if (GPSFullData.Sum() != 7)
                    return false;
                GPSFullData = new int[] { 0, 0, 0, 0, 0, 0, 0 };
                string gps_dtim = string.Format("{0} {1}", GPSDataTemp[0], GPSDataTemp[1]);
                string gps_dist = string.Format("{0:0.###}", distanceByGPS);
                // Distance calculation
                if (last_lat != 0 && last_lon != 0)
                {
                    double degRadC = Math.PI / 180.0; // deg to rad const
                    double dLat = (new_lat - last_lat) * degRadC;
                    double dLon = (new_lon - last_lon) * degRadC;
                    var a = Math.Pow(Math.Sin(dLat / 2.0), 2.0) + Math.Cos(last_lat * degRa
dC) * Math.Cos(new_lat * degRadC) * Math.Pow(Math.Sin(dLon / 2.0), 2.0);
                    var c = 2.0 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1.0 - a));
                    distanceByGPS += (earthRadius * c); // x km
                    gps_dist = string.Format("{0:0.###}", distanceByGPS);
                }
                last_lat = new_lat;
                last_lon = new_lon;
                // Save to output file
                string dataLine = string.Format("{0},{1},{2},{3},{4},{5},{6}", gps_dtim, GP
SDataTemp[2], GPSDataTemp[3], GPSDataTemp[4], GPSDataTemp[5], GPSDataTemp[6], gps_dist);
                using (StreamWriter sw = File.AppendText(outputFile))
                {
                    sw.WriteLine(dataLine);
                }
                return true;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
                return false;
            }
        }
    }
}
```

Příloha #7

Hlavní program v jazyce C# (desktopová aplikace pro Windows) ke komunikaci s moduly, zpracování a ukládání dat

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Management;
using System.Reflection;
using System.Text.RegularExpressions;
using System.Threading;
using System.Windows.Forms;
using InTheHand.Net;
using InTheHand.Net.Bluetooth;
using InTheHand.Net.Sockets;
using NmeaParser;
using NmeaParser.NmeaLines;

namespace TelemetryApp
{
    public partial class Telemetry : Form
    {
        public Telemetry()
        {
            InitializeComponent();
        }

        private string appPath;
        private SqlConnection dbConn1, dbConn2, dbConn3;
        // WARNING: DATABASE FILE WILL BE REPLACED ON EACH BUILD!!!
        private string dbConnStr = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename={0
}\\Telemetry.mdf;Integrated Security=True";
        // SAFE USE OF CONNECTION STRING
        //dbConnStr = "Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\Users\\ondr
e\\source\\repos\\TelemetryApp\\TelemetryApp\\Telemetry.mdf;Integrated Security=True";
        private List<Label> labels = new List<Label>();
        private void App_OnLoad(object sender, EventArgs e)
        {
            // Set locale to en-US
            CultureInfo.DefaultThreadCurrentCulture = new CultureInfo("en-US");

            // Get running EXE directory
            appPath = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
```

```
            // Set-up ports ComboBox
            ddmCOMPorts.Enabled = false;

            // Set-up function ComboBox
            ddmCOMFunc.Enabled = false;
            ddmCOMFunc.Items.Add("UART");

            // Set-up combinations ListView
            lstCombinations.View = View.Details;
            lstCombinations.Columns.Add("Port", 70);
            lstCombinations.Columns.Add("Function", 110);

            // Add data labels to List
            labels.Add(lblDateTimeRTCVal);
            labels.Add(lblTempBattVal);
            labels.Add(lblTempAmbVal);
            labels.Add(lblVoltageVal);
            labels.Add(lblCurrentVal);
            labels.Add(lblConsumpVal);
            labels.Add(lblAccelXVal);
            labels.Add(lblAccelYVal);
            labels.Add(lblAccelZVal);
            labels.Add(lblAltitude1Val);
            labels.Add(lblDateTimeGPSVal);
            labels.Add(lblLatitudeVal);
            labels.Add(lblLongitudeVal);
            labels.Add(lblAltitude2Val);
            labels.Add(lblSpeedVal);
            labels.Add(lblGPSFixVal);
            labels.Add(lblDistanceVal);

            // Hide timestamps prefix
            labels[0].Text = "";
            labels[10].Text = "";

            // Set-up connection to a local DB
            try
            {
                txbConsole.AppendText("Connecting to a local database..." + Environment.New
Line);

                dbConnStr = string.Format(dbConnStr, appPath);
                dbConn1 = new SqlConnection();
                dbConn1.ConnectionString = dbConnStr;
                dbConn1.Open();
                dbConn2 = new SqlConnection();
                dbConn2.ConnectionString = dbConnStr;
                dbConn2.Open();
                dbConn3 = new SqlConnection();
```

```
                dbConn3.ConnectionString = dbConnStr;
                dbConn3.Open();
                txbConsole.AppendText("Local database successfully connected." + Environmen
t.NewLine);
            }
            catch (System.Data.SqlClient.SqlException ex)
            {
                txbConsole.AppendText("Error: Connection to local database failed!" + Envir
onment.NewLine);
            }
        }

        private void btnScanCOMs_Click(object sender, EventArgs e)
        {
            // Get all available serial ports names
            ManagementObjectSearcher portsSearcher =
                new ManagementObjectSearcher("root\\CIMV2", "SELECT * FROM Win32_PnPEntity
WHERE ClassGuid=\"{4d36e978-e325-11ce-bfc1-08002be10318}\"");

            var ports = from ManagementObject s in portsSearcher.Get()
                        select new { Name = s["Name"], DeviceID = s["DeviceID"], PNPDeviceI
D = s["PNPDeviceID"] };

            // Clear any previous known ports
            ddmCOMPorts.Items.Clear();
            lstCombinations.Items.Clear();

            if (ports.Count() > 0)
            {
                // Display each port name in ComboBox
                foreach (var port in ports)
                {
                    var pnpDeviceId = port.PNPDeviceID.ToString();
                    if (!pnpDeviceId.Contains("BTHENUM"))
                    {
                        Regex rgx = new Regex("COM[0-9]{1,2}");
                        Match comName = rgx.Match(port.Name.ToString());
                        ddmCOMPorts.Items.Add(comName.Value);
                    }
                    else
                    {
                        var bluetoothDeviceAddress = pnpDeviceId.Split('&')[4].Split(' ')[0
];
                        if (bluetoothDeviceAddress.Length == 12
                            && bluetoothDeviceAddress != "000000000000"
                            && bluetoothDeviceAddress == btAddress.ToString()
                            && btIsConn == true)
                        {
                            Regex rgx = new Regex("COM[0-9]{1,2}");
```

```
                    Match comName = rgx.Match(port.Name.ToString());
                    string[] parts = { comName.Value, "BLUETOOTH" };
                    ListViewItem comb_item = new ListViewItem(parts);
                    lstCombinations.Items.Add(comb_item);
                }
            }
        }
        if (ddmCOMPorts.Items.Count > 0)
        {
            // Set first port as visible
            ddmCOMPorts.SelectedIndex = 0;
            ddmCOMPorts.Enabled = true;
            // Enable function ComboBox
            ddmCOMFunc.Enabled = true;
            ddmCOMFunc.SelectedIndex = 0;
        }
        else
            PlaceLog("Error: No active ports available!" + Environment.NewLine);
    }
    else
    {
        // No ports available - show alert
        string alert = "No COM ports available right now!";
        string title = "Error";
        MessageBox.Show(alert, title);
    }
}

private void btnConfirmComb_Click(object sender, EventArgs e)
{
    if (!ddmCOMPorts.Enabled)
    {
        // No scan was performed
        string alert = "You have to scan for COM ports first!";
        string title = "Info";
        MessageBox.Show(alert, title);
        return;
    }
    string[] parts = new string[2];
    parts[0] = ddmCOMPorts.SelectedItem.ToString();
    parts[1] = ddmCOMFunc.SelectedItem.ToString();
    foreach (ListViewItem combination in lstCombinations.Items)
    {
        if (combination.SubItems[0].Text.Contains(parts[0])
        || combination.SubItems[1].Text.Contains(parts[1]))
        {
            // Single port to a single function only
            string alert = "This port or function has already been used!";
            string title = "Warning";
```

```
                    MessageBox.Show(alert, title);
                    return;
                }
            }
            ListViewItem comb_item = new ListViewItem(parts);
            lstCombinations.Items.Add(comb_item);
        }

        private void btnDeleteComb_Click(object sender, EventArgs e)
        {
            int num_del = 0;
            foreach (ListViewItem combination in lstCombinations.SelectedItems)
            {
                lstCombinations.Items.Remove(combination);
                num_del++;
            }
            if (num_del == 0)
            {
                // No items selected or available
                string alert = "No combination(s) selected to delete!";
                string title = "Info";
                MessageBox.Show(alert, title);
            }
        }

        private bool serialPoolInit = false; // Prevent from starting multiple reading pool
s
        private void btnStartTel_Click(object sender, EventArgs e)
        {
            // Scan for current configuration and set ports
            foreach (ListViewItem combination in lstCombinations.Items)
            {
                string curr_port_name = combination.SubItems[0].Text.ToString();
                string curr_port_func = combination.SubItems[1].Text.ToString();
                if (curr_port_func == "UART" && UART.IsOpen == false)
                {
                    try
                    {
                        UART.PortName = curr_port_name;
                        UART.BaudRate = 9600;
                        UART.Open();
                        lblUARTStatus.Text = "CONNECTED";
                        lblUARTStatus.BackColor = System.Drawing.Color.Lime;
                    }
                    catch (System.IO.IOException ex)
                    {
                        if (!UART.IsOpen)
                        {
                            // UART not set/opened properly
```

```
                            string alert = "UART port is not set!";
                            string title = "Error";
                            MessageBox.Show(alert, title);
                        }
                    }
                }
                else if (curr_port_func == "BLUETOOTH")
                {
                    // BT communication port defined by another way
                }
                txbConsole.AppendText("Virtual port ");
                txbConsole.AppendText(curr_port_name);
                txbConsole.AppendText(" configured as ");
                txbConsole.AppendText(curr_port_func);
                txbConsole.AppendText(" port" + Environment.NewLine);
            }
            // Start reading thread
            if (UART.IsOpen == true && serialPoolInit == false)
            {
                ThreadPool.QueueUserWorkItem(UART_DataProcessing);
                serialPoolInit = true;
            }
            // Needs to be tested
            if (btClient != null)
            {
                btToSend = "$CAL;$START;";
            }
        }

        // Toggle data logging status
        private bool logging = false;
        private DateTime loggingEndTime = new DateTime(); // Save data to DB up to this dat
etime
        private void ToggleLogging()
        {
            if (lblDBLogStatus.Text == "OFF" && logging == false)
            {
                logging = true;
                lblDBLogStatus.Text = "ON";
                lblDBLogStatus.BackColor = System.Drawing.Color.Lime;
                PlaceLog("Started saving data into DB." + Environment.NewLine);
            }
            else if (lblDBLogStatus.Text == "ON" && logging == true)
            {
                logging = false;
                loggingEndTime = DateTime.UtcNow;
                PlaceLog("Please check both timestamps before closing the app!" + Environme
nt.NewLine);
```

```
                PlaceLog("Some data may still be in process of saving into DB." + Environme
nt.NewLine);
                lblDBLogStatus.Text = "OFF";
                lblDBLogStatus.BackColor = System.Drawing.Color.Red;
                PlaceLog("Stopped saving new data into DB." + Environment.NewLine);
            }
        }

        private void lblDBLogStatus_Click(object sender, EventArgs e)
        {
            ToggleLogging();
        }

        // Stop receiving new data
        private void btnStop_Click(object sender, EventArgs e)
        {
            btToSend = "$STOP;";
            if (logging == true)
            {
                ToggleLogging();
            }
        }

        // Access graphical elements of UI thread from other threads
        delegate void PlaceLog_Callback(string logg);
        private void PlaceLog(string logg)
        {
            if (this.txbConsole.InvokeRequired)
            {
                PlaceLog_Callback g = new PlaceLog_Callback(PlaceLog);
                this.Invoke(g, new object[] { logg });
            }
            else
            {
                this.txbConsole.AppendText(logg);
            }
        }

        private delegate void SetControlPropertyThreadSafeDelegate(
            Control control,
            string propertyName,
            object propertyValue
            );

        public static void SetControlPropertyThreadSafe(
            Control control,
            string propertyName,
            object propertyValue)
        {
```

```
            if (control.InvokeRequired)
            {
                control.Invoke(new SetControlPropertyThreadSafeDelegate
                    (SetControlPropertyThreadSafe),
                    new object[] { control, propertyName, propertyValue }
                );
            }
            else
            {
                control.GetType().InvokeMember(
                    propertyName,
                    BindingFlags.SetProperty,
                    null,
                    control,
                    new object[] { propertyValue }
                );
            }
        }

        private BluetoothClient btClient;
        private Guid btServiceClass;
        private BluetoothAddress btAddress = BluetoothAddress.Parse("00:21:06:be:88:60"); /
/ HC-06
        //private BluetoothListener btListener;
        private Stream btStream;
        private StreamReader btReader;
        private StreamWriter btWriter;
        private bool btIsConn = false;
        private string btToSend = "";
        private bool listenerPoolInit = false;
        private bool senderPoolInit = false;

        private void btnBTConnect_Click(object sender, EventArgs e)
        {
            if (btClient != null)
            {
                btClient.Close();
            }

            btClient = new BluetoothClient();

            btServiceClass = BluetoothService.SerialPort;

            var bluetoothDeviceInfos = btClient.DiscoverDevices();
            txbConsole.AppendText("Searching for bluetooth devices..." + Environment.NewLin
e);
            var deviceInfos = bluetoothDeviceInfos.ToList();
            BluetoothDeviceInfo device = null;
```

```
            foreach (var bluetoothDeviceInfo in deviceInfos)
            {
                var scannedDeviceAddress = bluetoothDeviceInfo.DeviceAddress;
                txbConsole.AppendText("Device found: <" + scannedDeviceAddress + ">." + Env
ironment.NewLine);

                if (scannedDeviceAddress == btAddress)
                {
                    device = bluetoothDeviceInfo;
                }
            }

            if (device == null)
            {
                return;
            }

            txbConsole.AppendText("Connecting to <" + device.DeviceAddress + ">..." + Envir
onment.NewLine);

            try
            {
                if (!device.Connected)
                {

                    btClient.Connect(btAddress, btServiceClass);

                    Stream btStream = btClient.GetStream();
                    btReader = new StreamReader(btStream);
                    btWriter = new StreamWriter(btStream);

                    if (listenerPoolInit == false)
                    {
                        ThreadPool.QueueUserWorkItem(BT_Listener);
                        listenerPoolInit = true;
                    }
                    if (senderPoolInit == false)
                    {
                        ThreadPool.QueueUserWorkItem(BT_Sender);
                        senderPoolInit = true;
                    }

                    btIsConn = true;
                    // Successfully connected to the Bluetooth device
                    lblBTStatus.Text = "CONNECTED";
                    lblBTStatus.BackColor = System.Drawing.Color.Lime;
                    txbConsole.AppendText("Connected." + Environment.NewLine);
                }
            }
```

```
            catch (System.Net.Sockets.SocketException ex)
            {
                btClient.Close();
                btIsConn = false;
                // Connection to the Bluetooth device failed
                txbConsole.AppendText("Error occured!" + Environment.NewLine);
                return;
            }
        }

        private void BT_Listener(object state)
        {
            while (true)
            {
                // Reading data section
                string line = null;
                try
                {
                    line = btReader.ReadLine();
                    if (line != null)
                    {
                        if (line.Contains("$DATA1,"))
                        {
                            Regex rgx = new Regex("[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-
9]{2}:[0-9]{2}.[0-9]{3}(,[-]{0,1}[0-9]+.[0-9]+)+");
                            Match extr = rgx.Match(line);
                            if (extr.Success)
                            {
                                string[] separated = extr.Value.Split(',');
                                bool waitForSave = false;
                                DateTime dataStamp;
                                if (DateTime.TryParse(separated[0], out dataStamp))
                                {
                                    int dtDiff = DateTime.Compare(loggingEndTime, dataStamp
);
                                    waitForSave = (dtDiff > 0) ? true : false; // end > dat
a
                                }
                                if (logging == true || waitForSave == true)
                                {
                                    SqlCommand command = new SqlCommand("INSERT INTO Sensor
s1 (cas razitko,teplota okoli,teplota baterie,voltu svorky,proud,spotreba,zrychleni x,zrych
leni_y,zrychleni_z,vyska) VALUES(@ts,@t1,@t2,@u1,@i1,@sp,@ax,@ay,@az,@al);", dbConn1);
                                    command.Parameters.AddWithValue("@ts", separated[0]);
                                    command.Parameters.AddWithValue("@t1", separated[1]);
                                    command.Parameters.AddWithValue("@t2", separated[2]);
                                    command.Parameters.AddWithValue("@u1", separated[3]);
                                    command.Parameters.AddWithValue("@i1", separated[4]);
                                    command.Parameters.AddWithValue("@sp", separated[5]);
```

```
                                    command.Parameters.AddWithValue("@ax", separated[6]);
                                    command.Parameters.AddWithValue("@ay", separated[7]);
                                    command.Parameters.AddWithValue("@az", separated[8]);
                                    command.Parameters.AddWithValue("@al", separated[9]);
                                    command.ExecuteNonQuery();
                                }
                                for (int i = 0; i < separated.Length; i++)
                                {
                                    SetControlPropertyThreadSafe(labels[i], "Text", separat
ed[i]);
                                }
                            }
                        }
                        else if (line.Contains("$DATA2,"))
                        {
                            Regex rgx = new Regex("[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-
9]{2}:[0-9]{2}.[0-9]{3}(,[-]{0,1}[0-9]+.[0-9]+)+");
                            Match extr = rgx.Match(line);
                            if (extr.Success)
                            {
                                string[] separated = extr.Value.Split(',');
                                bool waitForSave = false;
                                DateTime dataStamp;
                                if (DateTime.TryParse(separated[0], out dataStamp))
                                {
                                    int dtDiff = DateTime.Compare(loggingEndTime, dataStamp
);
                                    waitForSave = (dtDiff > 0) ? true : false; // end > dat
a
                                }
                                if (logging == true || waitForSave == true)
                                {
                                    SqlCommand command = new SqlCommand("INSERT INTO Sensor
s2 (cas_razitko,voltu_min,voltu_max,amper_min,amper_max,zrych_x_min,zrych_x_max,zrych_y_min
,zrych_y_max,zrych_z_min,zrych_z_max) VALUES(@ts,@umi,@uma,@imi,@ima,@axmi,@axma,@aymi,@aym
a,@azmi,@azma);", dbConn3);
                                    command.Parameters.AddWithValue("@ts", separated[0]);
                                    command.Parameters.AddWithValue("@umi", separated[1]);
                                    command.Parameters.AddWithValue("@uma", separated[2]);
                                    command.Parameters.AddWithValue("@imi", separated[3]);
                                    command.Parameters.AddWithValue("@ima", separated[4]);
                                    command.Parameters.AddWithValue("@axmi", separated[5]);
                                    command.Parameters.AddWithValue("@axma", separated[6]);
                                    command.Parameters.AddWithValue("@aymi", separated[7]);
                                    command.Parameters.AddWithValue("@ayma", separated[8]);
                                    command.Parameters.AddWithValue("@azmi", separated[9]);
                                    command.Parameters.AddWithValue("@azma", separated[10])
;
                                    command.ExecuteNonQuery();
```

```
                                }
                            }
                        }
                        else if (line.Contains("$INFO,"))
                        {
                            Regex rgx = new Regex("[$]INFO,.+;");
                            Match extr = rgx.Match(line);
                            line = extr.Value.Substring(6, line.Length - 7);
                            PlaceLog(line + Environment.NewLine);
                        }
                    }
                }
                catch (Exception ex)
                {
                    if (ex is System.IO.IOException)
                    {
                        PlaceLog("Error: Bluetooth connection was terminated!" + Environmen
t.NewLine);
                        System.Threading.Thread.Sleep(5000); // Additional sleep when BT te
rminated
                    }
                    if (ex is System.Data.SqlClient.SqlException)
                        PlaceLog("Error: Data couldn't be saved to DB!" + Environment.NewLi
ne);
                    if (ex is System.ObjectDisposedException) ;
                }
                System.Threading.Thread.Sleep(200);
            }
        }

        private void BT_Sender(object state)
        {
            while (true)
            {
                // Sending data section
                try
                {
                    if (btToSend.Length > 0)
                    {
                        string message = btToSend;
                        btToSend = "";
                        if (btWriter != null)
                        {
                            btWriter.WriteLine(message);
                            btWriter.Flush();
                            PlaceLog("Sent: " + message + Environment.NewLine);
                        }
                    }
                }
```

```
                catch (Exception ex)
                {
                    if (ex is System.IO.IOException)
                    {
                        PlaceLog("Error: Bluetooth connection was terminated!" + Environmen
t.NewLine);
                        System.Threading.Thread.Sleep(5000); // Additional sleep when BT te
rminated
                    }
                    if (ex is System.ObjectDisposedException) ;
                }
                System.Threading.Thread.Sleep(200);
            }
        }

        private string[] GPSDataTemp = new string[7]; // date, time, lat, lon, alt, speed,
fix
        private int[] GPSFullData = new int[7] { 0, 0, 0, 0, 0, 0, 0 }; // 1 per valid sing
le data
        private int earthRadius = 6371;
        private double last_lat = 0; // not available before first succesful GPS fix
        private double last_lon = 0;
        private double distanceByGPS = 0; // in km
        private void UART_DataProcessing(object state)
        {
            while (true)
            {
                try
                {
                    string sData = UART.ReadLine();
                    if (sData != null)
                    {
                        if (sData.Contains("RMC")) // GPRMC, GNRMC
                        {
                            var nmeaMsg = new NmeaLineFactory().ParseLine(sData);
                            var rmcMsg = (RmcLine)nmeaMsg;
                            DateTime GPSDt = rmcMsg.FixTime;
                            string gps_date = GPSDt.ToString("yyyy-MM-dd");
                            string gps_time = GPSDt.ToString("HH:mm:ss");
                            string gps_sped = string.Format("{0:0.##}", rmcMsg.GeoCoordinat
e.Speed);
                            // Data with new timestamp? => save previous to file
                            if (GPSDataTemp[1] != gps_time)
                            {
                                Publish_GPSdata(rmcMsg.GeoCoordinate.Latitude, rmcMsg.GeoCo
ordinate.Longitude);
                            }
                            GPSFullData[0] = 1;
                            GPSFullData[1] = 1;
```

```
                            GPSFullData[5] = 1;
                            GPSDataTemp[0] = gps_date;
                            GPSDataTemp[1] = gps_time;
                            GPSDataTemp[5] = gps_sped;
                        }
                        else if (sData.Contains("GGA")) // GPGGA, GNGGA
                        {
                            var nmeaMsg = new NmeaLineFactory().ParseLine(sData);
                            var ggaMsg = (GgaLine)nmeaMsg;
                            string gps_time = ggaMsg.FixTime.ToString();
                            string gps_lati = string.Format("{0:0.#######}", ggaMsg.Positio
n.Latitude);
                            string gps_long = string.Format("{0:0.#######}", ggaMsg.Positio
n.Longitude);
                            string gps_alti = string.Format("{0:0.##}", ggaMsg.Position.Alt
itude);
                            // Data with new timestamp? => save previous to file
                            if (GPSDataTemp[1] != gps_time)
                            {
                                Publish_GPSdata(ggaMsg.Position.Latitude, ggaMsg.Position.L
ongitude);
                            }
                            GPSFullData[1] = 1;
                            GPSFullData[2] = 1;
                            GPSFullData[3] = 1;
                            GPSFullData[4] = 1;
                            GPSDataTemp[1] = gps_time;
                            GPSDataTemp[2] = gps_lati;
                            GPSDataTemp[3] = gps_long;
                            GPSDataTemp[4] = gps_alti;
                        }
                        else if (sData.Contains("GSA")) // GPGSA, GNGSA, BDGSA
                        {
                            var nmeaMsg = new NmeaLineFactory().ParseLine(sData);
                            var gsaMsg = (GsaLine)nmeaMsg;
                            string gps_fix = string.Format("{0}", gsaMsg.Fix);
                            if (gps_fix != "NotAvailable")
                                GPSFullData[6] = 1;
                            GPSDataTemp[6] = gps_fix;
                        }
                    }
                }
                catch (Exception ex)
                {
                    // Serial connection was terminated
                    if (ex is System.IO.IOException) ;
                    // Invalid checksum of NMEA message
                    if (ex is System.ArgumentException)
                        PlaceLog("Invalid NMEA message received." + Environment.NewLine);
```

```
                    // NMEA message data parsing error
                    if (ex is System.FormatException)
                        PlaceLog("NMEA message data parsing error" + Environment.NewLine);
                    // Unsupported type of NMEA message
                    if (ex is System.ArgumentOutOfRangeException)
                        PlaceLog("Unsupported NMEA message received." + Environment.NewLine
);
                }
                System.Threading.Thread.Sleep(50);
            }
        }

        private bool Publish_GPSdata(double new_lat, double new_lon)
        {
            try
            {
                if (GPSFullData.Sum() != 7)
                    return false;
                GPSFullData = new int[] { 0, 0, 0, 0, 0, 0, 0 };
                string gps_dtim = string.Format("{0} {1}", GPSDataTemp[0], GPSDataTemp[1]);
                string gps_dist = string.Format("{0:0.###}", distanceByGPS);
                // Distance calculation
                if (last_lat != 0 && last_lon != 0)
                {
                    double degRadC = Math.PI / 180.0; // deg to rad const
                    double dLat = (new_lat - last_lat) * degRadC;
                    double dLon = (new_lon - last_lon) * degRadC;
                    var a = Math.Pow(Math.Sin(dLat / 2.0), 2.0) + Math.Cos(last_lat * degRa
dC) * Math.Cos(new_lat * degRadC) * Math.Pow(Math.Sin(dLon / 2.0), 2.0);
                    var c = 2.0 * Math.Atan2(Math.Sqrt(a), Math.Sqrt(1.0 - a));
                    distanceByGPS += (earthRadius * c); // x km
                    gps_dist = string.Format("{0:0.###}", distanceByGPS);
                }
                last_lat = new_lat;
                last_lon = new_lon;
                // Timespans not reflecting stop time
                bool waitForSave = false;
                DateTime dataStamp;
                if (DateTime.TryParse(gps_dtim, out dataStamp))
                {
                    int dtDiff = DateTime.Compare(loggingEndTime, dataStamp);
                    waitForSave = (dtDiff > 0) ? true : false; // end > data
                    /*if (waitForSave)
                        PlaceLog(loggingEndTime.ToString("yyyy-MM-
dd HH:mm:ss") + " > " + gps_dtim + Environment.NewLine);*/
                }
                if (logging == true || waitForSave == true)
                {
                    // Save to DB
```

```
                    SqlCommand command = new SqlCommand("INSERT INTO Gps (cas_razitko,zem_s
irka,zem_delka,nadm_vyska,rychlost,gps_stav,vzdalenost) VALUES(@ts,@la,@lo,@al,@sp,@gs,@ds)
;", dbConn2);
                    command.Parameters.AddWithValue("@ts", gps_dtim); // Datetime UTC expec
ted
                    command.Parameters.AddWithValue("@la", GPSDataTemp[2]);
                    command.Parameters.AddWithValue("@lo", GPSDataTemp[3]);
                    command.Parameters.AddWithValue("@al", GPSDataTemp[4]);
                    command.Parameters.AddWithValue("@sp", GPSDataTemp[5]);
                    command.Parameters.AddWithValue("@gs", GPSDataTemp[6]);
                    command.Parameters.AddWithValue("@ds", gps_dist);
                    command.ExecuteNonQuery();
                }
                // Print values to screen
                SetControlPropertyThreadSafe(labels[10], "Text", gps_dtim);
                SetControlPropertyThreadSafe(labels[11], "Text", GPSDataTemp[2]);
                SetControlPropertyThreadSafe(labels[12], "Text", GPSDataTemp[3]);
                SetControlPropertyThreadSafe(labels[13], "Text", GPSDataTemp[4]);
                SetControlPropertyThreadSafe(labels[14], "Text", GPSDataTemp[5]);
                if (GPSDataTemp[6] != "NotAvailable")
                {
                    SetControlPropertyThreadSafe(labels[15], "Text", GPSDataTemp[6]);
                    SetControlPropertyThreadSafe(labels[15], "BackColor", System.Drawing.Co
lor.Lime);
                }
                else
                {
                    SetControlPropertyThreadSafe(labels[15], "Text", "None");
                    SetControlPropertyThreadSafe(labels[15], "BackColor", System.Drawing.Co
lor.Red);
                }
                SetControlPropertyThreadSafe(labels[16], "Text", gps_dist);
                return true;
            }
            catch (Exception ex)
            {
                if (ex is System.Data.SqlClient.SqlException)
                    PlaceLog("Error: Data couldn't be saved to DB!" + Environment.NewLine);

                return false;
            }
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            // Close opened serial ports before app closure
            if (UART.IsOpen)
                UART.Close();
            // Performs automatically after app closure??
```

```
            if (btReader != null)
                btReader.Close();
            if (btWriter != null)
                btWriter.Close();
            if (btClient != null)
                btClient.Close();
            // Close DB connections
            if (dbConn1 != null)
                dbConn1.Close();
            if (dbConn2 != null)
                dbConn2.Close();
            if (dbConn3 != null)
                dbConn3.Close();
        }
    }
}
```

Příloha #8

Skript v jazyce Python pro propojení dat ze senzorů a GPS, včetně linearizace a dopočtu některých dalších parametrů

```
import sys
from datetime import datetime, time, timedelta
from math import atan, pi

separator = ','

# zjisteni poctu nenulovych radku v datovem souboru
sfile1 = open("dbo.Sensors1.data.final2.csv", 'r')
scount1 = 0
for line in enumerate(sfile1):
    if (len(line) > 0):
        scount1 += 1
sfile2 = open("dbo.Sensors2.data.final.csv", 'r')
scount2 = 0
for line in enumerate(sfile2):
    if (len(line) > 0):
        scount2 += 1
if (scount1 != scount2):
    print("Nesouhlasi pocet radku mezi Sensors1 a Sensors2!")
    sys.exit()
# GPS data lines
gcount = 0
gfile = open("dbo.Gps.data.final.csv", 'r')
for line in enumerate(gfile):
    if (len(line) > 0):
        gcount += 1
# nalezeni startovni pozice
# prvni radek je hlavicka (preskocit)
si = 1
gi = 1
udate = datetime(1, 1, 1)
sfile1.seek(0)
slines1 = sfile1.readlines()
timstr11 = slines1[si].split(separator)[0]
tim11 = datetime.combine(udate, time.fromisoformat(timstr11))
sfile2.seek(0)
slines2 = sfile2.readlines()
timstr12 = slines2[si].split(separator)[0]
tim12 = datetime.combine(udate, time.fromisoformat(timstr12))
if (tim11-tim12 > timedelta(milliseconds=1)):
    print("Nesouhlasi casova razitka mezi Sensors1 a Sensors2!")
    sys.exit()
gfile.seek(0)
```

```
glines = gfile.readlines()
timstr2 = glines[gi].split(separator)[0]
tim2 = datetime.combine(udate, time.fromisoformat(timstr2))
# sfile1 ma pozdejsi data
if (tim11 > tim2):
    for gi in range(gi, gcount):
        timstr1 = slines1[si].split(separator)[0]
        tim1 = datetime.combine(udate, time.fromisoformat(timstr1))
        timstr2 = glines[gi].split(separator)[0]
        tim2 = datetime.combine(udate, time.fromisoformat(timstr2))
        if (tim1 - tim2 < timedelta(seconds=1) and tim1.second == tim2.second):
            break
elif (tim2 > tim11):
    for si in range(si, scount1):
        timstr1 = slines1[si].split(separator)[0]
        tim1 = datetime.combine(udate, time.fromisoformat(timstr1))
        timstr2 = glines[gi].split(separator)[0]
        tim2 = datetime.combine(udate, time.fromisoformat(timstr2))
        if (tim2 - tim1 < timedelta(seconds=1) and tim1.second == tim2.second):
            break

# kontrolni vypis cisel radku (jako v editoru) a casovych razitek
print("Sensors start line: %d @ %s\r\nGPS start line: %d @ %s" % (si+1, slines1[si].split(s
eparator)[0], gi+1, glines[gi].split(separator)[0]))

# soubor pro zapis nakombinovanych dat
w = open("dbo.Sensors.data.combined32.csv", "w")
header1 = slines1[0].strip().split(separator)
header2 = glines[0].strip().split(separator)
header2[0] = header2[0]+"_gps"
header3 = slines2[0].strip().split(separator)
pop = 0 # pop other timestamps == -1; do not pop == 0
if (abs(pop)):
    header2.pop(0)
    header3.pop(0)
comb_h = header1+header2+header3
comb_h.append("stoupani1 (st.)")
comb_h.append("stoupani2 (pr.)")
comb_h.insert(0, "trvani_jizdy (s)")
joined_headers = separator.join(comb_h)
w.write(joined_headers + '\n')

ride_time = 0.0 # doba jizdy
last_time = tim11 # posledni cas
pitch = 0.0 # vychozi vodorovna parkovaci pozice
grade = 0.0
delta = 50 # zjistovani stoupani po x metrech
milnik = delta # aktualni milnik, bude navysovan vzdy o deltu
last_alt = 0.0 # posledni pozice
```

```
last_dis = 0.0

# kombinace tabulek, pokracuji start radkem
for si in range(si, scount1):
    sens_data1 = slines1[si].strip().split(separator)
    sens_data2 = slines2[si].strip().split(separator)
    timstr1 = sens_data1[0]
    tim1 = datetime.combine(udate, time.fromisoformat(timstr1))
    # jsem o sekundu dal se senzory oproti GPS
    if (tim1 > tim2 and tim1 - tim2 >= timedelta(seconds=1)):
        # hlidam konec souboru s GPS daty
        if (gi+1 < gcount):
            gi += 1
        else:
            break
    timstr2 = glines[gi].split(separator)[0]
    tim2 = datetime.combine(udate, time.fromisoformat(timstr2))
    # kontrola zda neni sekundova (i vice) mezera mezi GPS daty
    if (tim2 > tim1):
        continue
    # sedi mi sekundy
    gps_data1 = glines[gi].strip().split(separator)
    gps_lat1 = float(gps_data1[1])
    gps_lon1 = float(gps_data1[2])
    gps_alt1 = float(gps_data1[3])
    gps_dis1 = float(gps_data1[6])
    # pouze prirazeni GPS dat dane sekunde
    gps_lat_out = gps_lat1
    gps_lon_out = gps_lon1
    gps_alt_out = gps_alt1
    gps_dis_out = gps_dis1
    # hlidam zda jsou dostupna gi+1 data
    if (gi+1 < gcount):
        gps_data2 = glines[gi+1].split(separator)
        gps_lat2 = float(gps_data2[1])
        gps_lon2 = float(gps_data2[2])
        gps_alt2 = float(gps_data2[3])
        gps_dis2 = float(gps_data2[6])
        # linearne proloz
        millis = tim1.microsecond/1000.0
        gps_lat_out = gps_lat1-(gps_lat1-gps_lat2)/1000.0*millis
        gps_lon_out = gps_lon1-(gps_lon1-gps_lon2)/1000.0*millis
        gps_alt_out = gps_alt1-(gps_alt1-gps_alt2)/1000.0*millis
        gps_dis_out = gps_dis1-(gps_dis1-gps_dis2)/1000.0*millis
    comb_data = sens_data1+gps_data1+sens_data2
    comb_data[len(sens_data1)-1] = "%.1f" % (float(comb_data[len(sens_data1)-
1])) # vyska jen na 1 des. misto
    if (abs(pop)):
        # cas razitka vynechat
```

```
        comb_data.pop(len(sens_data1)+len(gps_data1)-1)
        comb_data.pop(len(sens_data1))
    comb_data[len(sens_data1)+pop+1] = "%.7f" % gps_lat_out # koordinaty
    comb_data[len(sens_data1)+pop+2] = "%.7f" % gps_lon_out
    comb_data[len(sens_data1)+pop+3] = "%.1f" % gps_alt_out # vyska
    comb_data[len(sens_data1)+len(gps_data1)+pop-1] = "%.3f" % gps_dis_out # vzdalenost
    # vypocet stoupani kazdych delta metru
    act_dis = gps_dis_out*1000.0
    if (act_dis > 0):
        if (act_dis >= milnik):
            #print("vzdalenost: %d, milnik %d" % (int(gps_dis_out*1000.0), milnik))
            act_alt = float(comb_data[len(sens_data1)-1])
            act_dis = gps_dis_out*1000.0
            if (act_dis-last_dis > 0):
                pitch = atan((act_alt-last_alt)/(act_dis-last_dis))*180.0/pi # ve stupnich
                grade = (act_alt-last_alt)/(act_dis-last_dis)*100.0 # v procentech
            last_alt = act_alt
            last_dis = act_dis
            milnik += delta
    elif (last_alt == 0.0 and last_dis == 0.0):
        last_alt = float(comb_data[len(sens_data1)-1])
        last_dis = act_dis
    comb_data.append("%.1f" % pitch)
    comb_data.append("%.1f" % grade)
    # vypocet casu od zacatku jizdy
    ride_time += (tim1-last_time).total_seconds()
    last_time = tim1
    comb_data.insert(0, "%.3f" % ride_time)
    joined_data = separator.join(comb_data)
    w.write(joined_data + '\n')

w.close()
sfile1.close()
gfile.close()
```

Příloha #9

Technický list výrobce k použitým Li-ion článkům

MOTOMA®

Power into the Future

TECHNICAL SPECIFICATION FOR

LI-ION

LCR18650-2000MAH

FILE NO. : DSE-LCR-LCR18650-2000-V17A

EDITION : V17A

DATE : 2017/03/08

CYLINDRICAL RECHARGEABLE BATTERY

MOTOMA®
Power into the Future

MOTOMA POWER CO., LTD
Http: // www.motoma.com E-mail: info@motoma.com

Prepared By	Checked By	Approved By
QE1	QE	A

MOTOMA POWER CO.,LTD
Http: // www.motoma.com E-mail:info@motoma.com

1. SCOPE

This product specification describes the nominal specification, technical requirement, testing method, warning and caution of the Li-ion rechargeable battery, which is manufactured by MOTOMA POWER CO.,LTD.

2. BATTERY MODEL LCR18650 2000mAh

3. NOMINAL SPECIFICATION

3.1 Nominal voltage 3.7V **3.2 Nominal capacity** 2000mAh **3.3 Minimum capacity** 2000mAh **3.4 Charging**

Constant Current and Constant Voltage (CC/CV)

Charge voltage	4.2V
Standard charge current	400mA (0.2C)
End current	20mA (0.01C)
Max. charge current	1000mA (0.5C)

3.5 Discharging

Standard discharge current	400mA (0.2C)

Max. discharge current	6000mA (3C)
Instant discharge current (5C)	Max one minute
End voltage of discharge	3.0V

3.6 Cycle Life

Discharge capacity (300th Cycle) ≥80% of Initial Capacity(0.2C)

3.7 Operate temperature range (relative humidity: 45%~75%) Standard

charge 0~45℃

Discharge: -20~60℃

3.8 Storage (relative humidity: 45%~75%)

Less than 30 days	-20~45℃
Less than 180 days	-20~35℃

3.9 Internal Impedance ≤45mΩ **3.10 Weight** Approx. 43g

3.11 Dimensions Shown in the page 6

4. APPEARANCE PERFORMANCE

There shall be no practical damage such as conspicuous liquid electrolyte leakage, flow and dirt under conditions of storage or operation as specified herein.

5. BATTERY CHARACTERISTICS

5.1 Testing conditions

Test should be conducted with new batteries within one month after shipment from our factory and the cells shall not be cycled more than five times before the test. Unless otherwise defined, test and measurement shall be done under temperature of 25±2℃ and relative humidity of 45~85%.

5.2 Measurement apparatus

5.2.1 Dimension measuring instrument

The dimension measurement shall be implemented by instruments with equal or more precision scale of 0.02mm.

5.2.2 Voltmeter

Standard class specified in the national standard or more sensitive class having inner impedance not less than 10 KΩ/V.

5.2.3 Ammeter

Standard class specified in the national standard or more sensitive class. Total external resistance including ammeter and wire is less than 0.01Ω.

5.2.4 Impedance meter

Impedance shall be measured by a sinusoidal alternating current method(AC 1kHz).

5.3 Charging procedure for test purpose

The battery shall be charged at an ambient temperature of 20±5℃ at a constant current of 400mA (0.2C) until the battery voltage reaches 4.2V, then charge at constant voltage of 4.2 V while tapering the charge current. Charging shall be terminated when the charging current has tapered to 20mA (0.01C).

5.4 Discharging performance

TABLE I: Discharging performance at 20℃

Constant Discharge Current Rate	Constant Discharge Current	End Point Voltage	Discharge Duration	Available Capacity
0.5C	1000mA	3.0V	120min	100%
1.0C	2000mA	3.0V	60min	100%
3.0C	6000mA	3.0V	20min	100%

※**Note:** All the testing should be done within 1 hour after being standard charged.

5.5 Temperature characteristics

Battery shall meet the discharge capacity requirements at different discharge temperature as showed in the follow table. The capacities are to be measured with constant discharge current on 0.2CmA (3.0V cut-off) after standard charge at 25±2℃.

TABLE 2: Temperature Characteristics

Discharge Temperature	0℃	25℃	60℃
Available Capacity	60%	100%	95%

5.6 Charge retention

TABLE 3: Charge retention

Item	Measuring Procedure	Requirements
Storage Characteristics (25℃)	The capacity on 0.2C discharge shall be measured after standard charge and then to be stored at 25±2℃ for 30 days.	Retention Capacity≥85%
	To measure the Retention Capacity after the battery to be cycled on 1.0C for three times.	Retention Capacity≥90%

Storage Characteristics (60℃)	The capacity on 0.2C discharge shall be measured after standard charge and then to be stored at 60±2℃ for 7 days.	Retention Capacity≥60%
	To measure the Retention Capacity after the battery to be cycled on 1.0C for three times.	Retention Capacity≥80%

5.7 Endurance in cycles

10min rest period after being standard charged, discharge the battery at a current of 0.2C to 3.0V, rest 10min, the capacity shall be measured after 300cycles of standard charge and discharge at 25±2℃.

Discharge capacity (300th Cycle) ≥80% of Initial Capacity

10min rest period after being standard charged, discharge the battery at a current of 1.0C to 3.0V, rest 10min, the capacity shall be measured after 300cycles of standard charge and discharge at 25±2℃.

Discharge capacity (300th Cycle) ≥75% of Initial Capacity

5.8 Mechanical performance

TABLE 4: Mechanical performance

Item	Measuring Procedure	Requirements
Vibration test	After standard charge, the battery is to be tested as following conditions: Amplitude:0.8mm Frequency:10~55Hz(sweep:1Hz/min) Direction: X/Y/Z axis for 90~100min. The battery is to be tested in three mutually perpendicular to each axis.	No fire, no explosion, no smoking is obtained.
Drop Test	Drop the battery in the shipment condition(full-charge) from 1m height onto 5cm or thicker concrete with p-tile on it 1 times each of X, Y, and Z directions at 25±2℃	No fire, no explosion, no smoking is obtained.

5.9 Safety performance

TABLE 5: Safety performance

Item	Measuring Procedure	Requirements
Overcharge Test	After standard charge (Section 4.4), the battery shall be charged at 2C/4.6V for 8.0hrs.	No fire, no explosion, no smoking is obtained.
Short circuiting Test	After standard charge (Section 4.4), the battery shall be subjected to a short-circuit condition with a wire of resistance less than 100mΩ for 1 hour.	No fire, no explosion, no smoking is obtained.
Over discharge Test	After discharged to the cut-off voltage, the battery shall be subjected to a short-circuit condition with a load of resistance less than 30Ω for 24hour.	No fire, no explosion, no smoking is obtained.
Heating Test	A battery is to be heated in a gravity convection or circulating air oven. The temperature of the oven is to be raised at a rate of 5±2℃/min to a temperature of 130±2℃ at which temperature the oven is to remain for 30 minutes before the test is discontinued.	No explosion, no fire.

5.10 Storage characteristics

After standard charged as 5.3, store the testing cells at 20℃±5℃ for 28 days. Then discharge at 0.5C to 3.0V.The discharge capacity≥80% of Initial capacity.

6. ENVIRONMENTAL PROTECTION REQUIREMENT

6.1 The requirement on Hazardous Substances in the materials should comply with MOTOMA standard on HSF (Hazardous Substance Free).

6.2 The requirement on Hazardous Substances in the Products should comply with 2006/66/EC and MOTOMA standard on HSF.

7. SHIPPING

The capacity of delivery battery is approximately at 80% of charging. During transportation, keep the battery from acutely vibration, impacting, solarization, drenching.

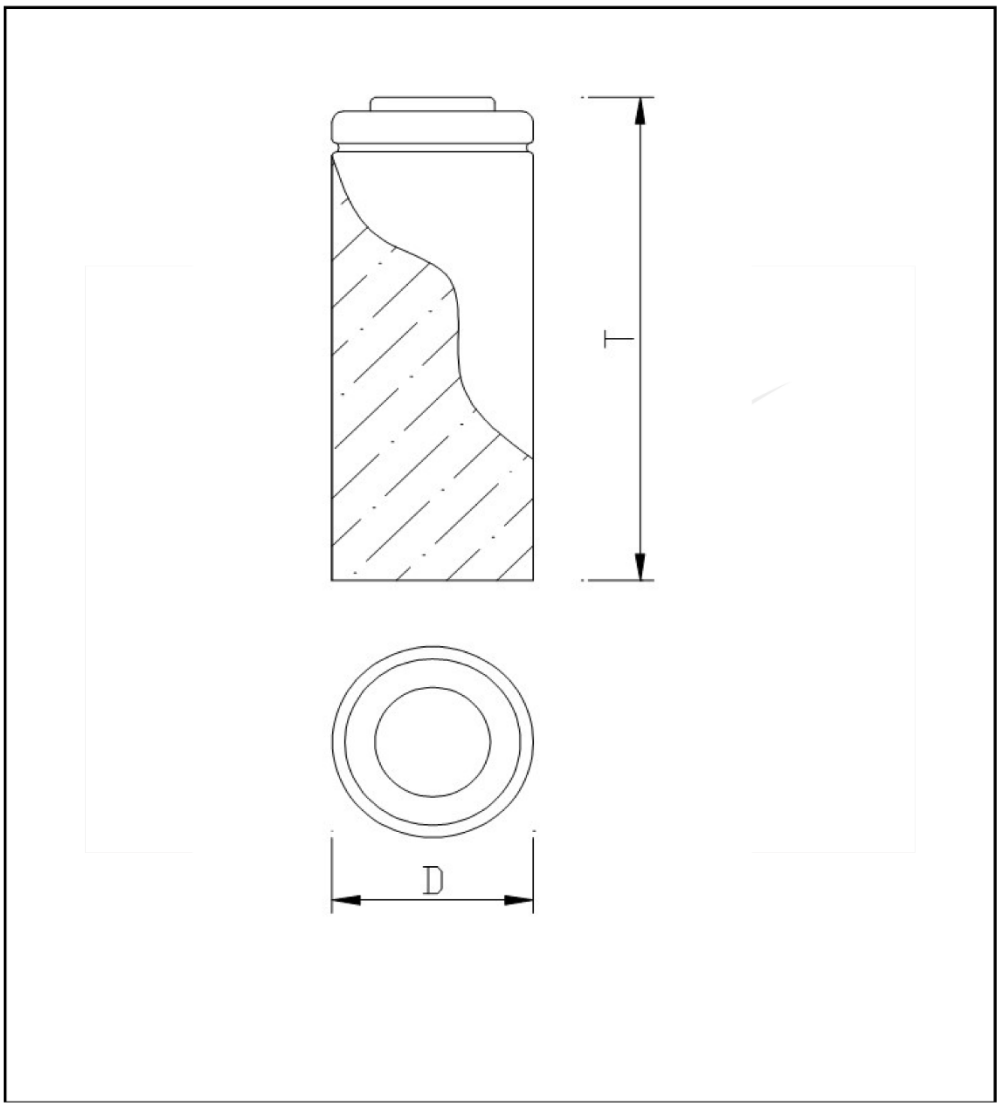
8. OTHERS

Any matters that this specification does not cover should be conferred between the customer and MOTOMA.

MOTOMA®
Power into the Future

MOTOMA POWER CO., LTD
Http: // www.motoma.com E-mail: info@motoma.com

9. ASSEMBLY DRAWING



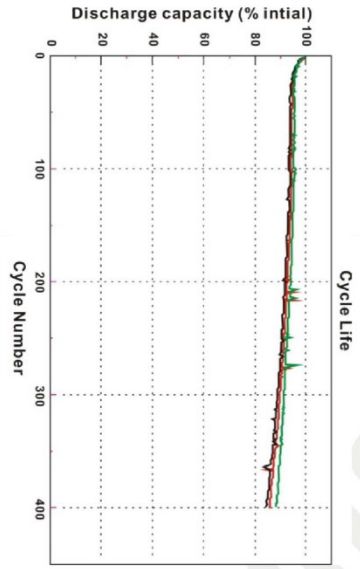
D	18	**H**	64.8	**Unit**	mm

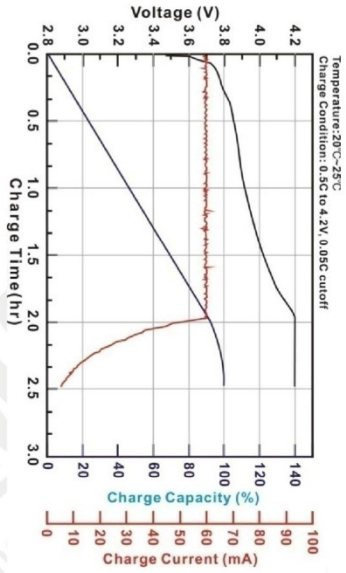
10. PERFORMANCE CURVE

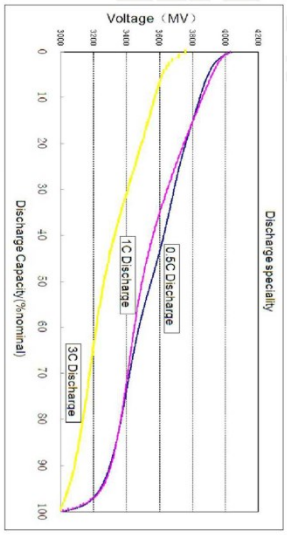
MOTOMA®
Power into the Future

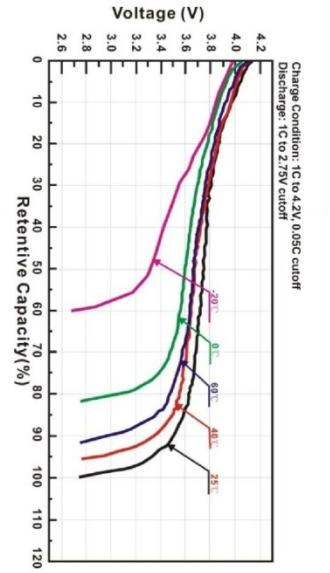
MOTOMA POWER CO., LTD

Http: // www.motoma.com E-mail: info@motoma.com









MOTOMA®
Power into the Future

MOTOMA POWER CO., LTD
Http: // www.motoma.com E-mail: info@motoma.com

Handling Precautions and Guideline For LCR (Lithium-ion Cylindrical) Rechargeable Batteries Preface

This document of 'Handling Precautions and Guideline LCR rechargeable batteries shall be applied to the battery cells manufactured by MOTOMA.

Note (1)

The customer is requested to contact MOTOMA, in advance, if and when the customer needs other applications or operating conditions than those described in this document. Additional experimentation may be required to verify performance and safety under such conditions.

Note (2)

MOTOMA will take no responsibility for any accident when the cell is used under other conditions than those described in this Document.

Note (3)

MOTOMA will inform, in a written form, the customer of improvement regarding proper use and handling of the cell, if it is deemed necessary.

1. Charging

1.1 Charging current

Charging current should be less than maximum charge current specified in the Product Specification. Charging with higher current than recommended value may cause damage to battery electrical, mechanical, and safety performance and could lead to heat generation or leakage.

1.2 Charging voltage

Charging shall be done by voltage less than that specified in the Product Specification (4.2V/cell). Charging beyond 4.30V, which is the absolute maximum voltage, must be strictly prohibited. The charger shall be designed to comply with this condition.

It is very dangerous that charging with higher voltage than maximum voltage may cause damage to the cell electrical, mechanical safety performance and could lead to heat generation or leakage.

1.3 Charging temperature

The cell shall be charged within 0℃~45℃ range in the product specification.

1.4 Prohibition of reverse charging

Reverse charging is prohibited. The cell shall be connected correctly. The polarity has to be confirmed before wiring. In case of the cell is connected improperly, the cell cannot be charged. Simultaneously, the reverse charging may cause damaging to the cell which may lead to degradation of cell performance and damage the cell safety, and could cause heat generation or leakage.

2. Discharging

2.1 Discharging current

The cell shall be discharged at less than the maximum discharge current specified in the product specification. High discharging current may reduce the discharging capacity significantly or cause over-heat.

2.2 Discharging temperature

The cell shall be discharged within 0℃~45℃ range specified in the Product Specification.

2.3 Over-discharging

It should be noted that the cell would be at an over-discharged state by its self-discharge characteristics in case the cell is not used for long time. In order to prevent over-discharging, the cell shall be charged periodically to maintain between 3.7V and 3.9V. Over-discharging may causes loss of cell performance, characteristics, or battery functions.

The charger shall be equipped with a device to prevent further discharging exceeding a cut-off voltage specified in the Product Specification. Also the charger shall be equipped with a device to control the recharging procedures as follows: The cell battery pack shall start with a low current (0.01C) for 15 – 30 minutes, i.e. pre-charging, before rapid charging starts. The rapid charging shall be started after the (individual) cell voltage has been reached above 3.0V within 15 - 30 minutes that can be determined with the use of an appropriate timer for pre-charging. In case the (individual) cell voltage does not rise to 3.0V within the pre-charging time, then the charger shall have functions to stop further charging and display the cell/pack is at abnormal state.

3. Protection Circuit Module

The cell/battery pack shall be with a PCM that can protect cell/battery pack properly. PCM shall have functions of (1) overcharging prevention, (2) over-discharging prevention, (3) over current prevention to maintain safety and prevent significant deterioration of cell performance. The over current can occur by external short circuit

3.1 Overcharging prohibition

Overcharging prohibition function shall stop charging if any one of the cells of the battery pack reaches 4.275±0.020V

3.2 Over-discharging prohibition

Over-discharging prevention function shall work to avoid further drop in cell voltage of 3.0±0.035V or less per cell in any cell of the battery pack. It is recommended that the dissipation current of PCM Shall be minimized to 0.5uA or less with the over-discharging prevention..

The protection function shall monitor each bank of the battery pack and control the current all the time.

4. Storage

The cell shall be stored within -20℃~ 45℃ range environmental condition.

If the cell has to be storied for a long time (over 3 months), the environmental condition should be:

MOTOMA®
Power into the Future
MOTOMA POWER CO., LTD
Http: // www.motoma.com E-mail: info@motoma.com

Temperature: 23±5℃; Humidity: 65±20%RH

The voltage for a long time storage shall be 3.7V~3.9V range.

Handling Instructions

1. WARNING !

- Do not immerse the battery in water or allow it to get wet.
- Do not use or store the battery near sources of heat such as a fire or heater.
- Do not use any chargers other than those recommended by MOTOMA POWER.
- Do not reverse the positive(+) and negative(-) terminals.
- Do not connect the battery directly to wall outlets or car cigarette-lighter sockets.
- Do not put the battery into a fire or apply direct heat to it.
- Do not short-circuit the battery by connecting wires or other metal objects to the positive(+) and negative(-) terminals.
- Do not pierce the battery casing with a nail or other sharp object, break it open with a hammer, or step on it.
- Do not strike, throw or subject the battery to physical shock.
- Do not directly solder the battery terminals.
- Do not attempt to disassemble or modify the battery in any way.
- Do not place the battery in a microwave oven or pressurized container.
- Do not use the battery in combination with primary batteries(such as dry-cell batteries) or batteries of different capacity, type or brand.
- Do not use the battery if it gives off an odor, generates heat, becomes discolored or deformed, or appears abnormal in any way. If the battery is in use or being recharged, remove it from the device or charger immediately and discontinue use.

2.CAUTION !

- Do not use or store the battery where is exposed to extremely hot, such as under window of a car in direct sunlight in a hot day. Otherwise, the battery may be overheated. This can also reduce battery performance and/or shorten service life.
- If the battery leaks and electrolyte gets in your eyes, do not rub them. Instead, rinse them with clean running water and immediately seek medical attention. If left as is, electrolyte can cause eye injury.
- Use the battery only under the following environmental conditions. Failure to do so can result in reduced performance or a shorten service life. Recharging the battery outside of these temperatures can cause the battery to overheat, explode or catch fire.

  Operating environment:

  When charging the battery: 0℃～45℃

  When discharging the battery: -20℃～60℃

When stored up to 30 days: -20℃～45℃

When stored up to 90 days: -20℃～35℃

Příloha #10

Technický list výrobce k alternativním olověným gelovým VRLA bateriím

超威 CHILWEE

A Reliable Power Solution Provider

6-EVF-45 12V 45Ah(3hr) VRLA GEL BATTERY

Chilwee EVF/DZM Series VRLA Gel Battery is specially designed for motive power applications, i.e. electric bicycles, electric tricycles, electric motocycles and other device require DC power source. The EVF/DZM Series adopts international leading technologies to ensure the batteries with features of long cycle life, large current discharge capability, high reliability and safety, and environmental-friendly.

FEATURES

Non-Cadmium Design, Environment-friendly: Chilwee Battery has adopted internationally leading technology - container formation non-cadmium production technology, which is in the leading position in the industry. It helps to save energy 28.5%, save water 90%, and non-discharge of waste water.

Super Long Voyage Ability: The discharge time of Chilwee battery is prolonged. Active additive has been added in postive plates, so as to good consistency of the formated active material. This enables the battery has high charge/discharge efficiency, and the power output is elevated.

Strong Motive Power: Super thin plate design is adopted to increase the area of the plates macroscopic electrochemical reaction, which enables the battery has excellent large current discharge ability. Adopting cast-weld process to reduce the battery's internal resistance, and it improves battery charging acceptance capability. Battery's charge/discharge efficiency is high.

High Durability: The Chilwee battery has excellent cycle life which can reach 600 cycles @ 80% DOD. The battery banks has good consistency. Enhanced ABS material has been adopted on battery container and lid, and the battery is well sealed to prevent leakage.

High Reliability and Safety: Improved negative material prescription and increased micropoles structure at negative helps to improve a lot on low temperature charge/discharge performance. Low water loss rate, high temperature resistance, and the pressure value of Open/Close safety valve is accurate to prevent battery bulging. Safety valve and acid filter can efficiently prevent sparks splashed into battery to ensure safe operation.

SPECIFICATION

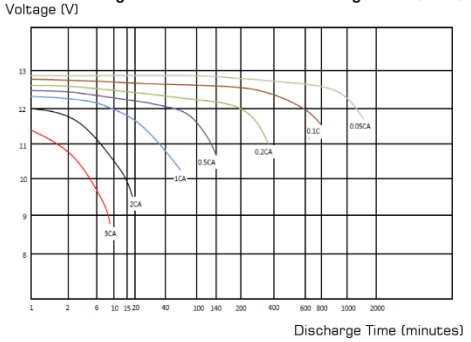
Nominal Voltage (V)		12V
Open Circuit Voltage (V/Block)		13.1V - 13.45V
Number of Cells (Per Block)		6 Cells
Rated Capacity (Ah, 25℃)	2h rate (to 1.75V/Cell)	40Ah
	3h rate (to 1.75V/Cell)	45Ah
	5h rate (to 1.80V/Cell)	48Ah
	20h rate (to 1.85V/Cell)	50Ah
Nominal Weight (Kgs)		Approx. 13.5Kgs
Dimension (L X W X H, Total Height. mm)		(226mm±2) X (120mm±2) X (175mm±2), (175mm±2)
Container Material		Enhanced ABS
Charge Voltage	Float (V/Block)	13.50V - 13.80V
	Cycle (V/Block)	14.50V - 14.80V
Maximum Discharge Current (A)		300A (5s)
Maximum Charge Current (A)		5A

超威 CHILWEE

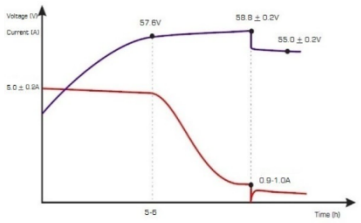
A Reliable Power Solution Provider

6-EVF-45 12V 45Ah(3hr) VRLA GEL BATTERY

Discharge Curves at Different Discharge Rate (25℃)

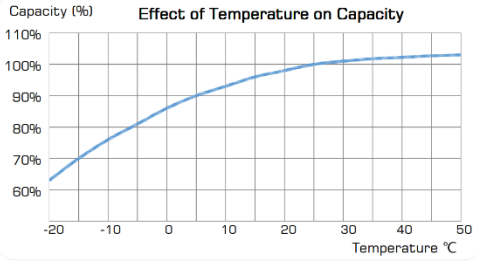


Charge Curve for 6-EVF-45 (4 Blocks/String)

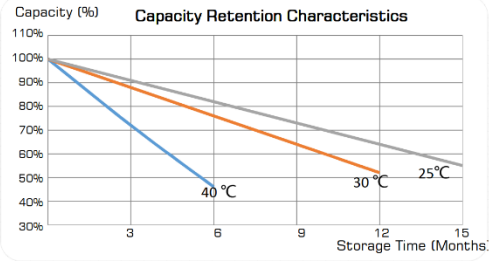


Phase 1: The Max. charge current is 5A, and the charge voltage is gradually risen up to 57.6V;
Phase 2: The charge voltage is gradually risen up to 58.8V ± 0.2V. When the charge current has dropped to 0.9A-1.0A, shifting to float charge.
Phase 3: The constant float charge voltage is 55.0V ± 0.2V.

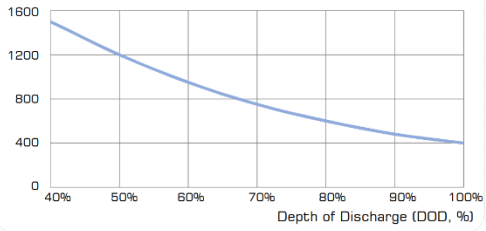
Effect of Temperature on Capacity



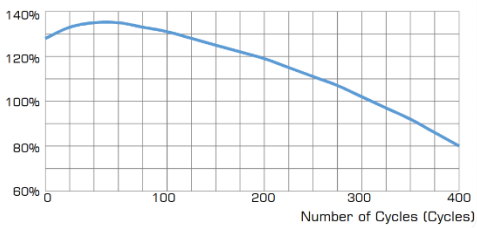
Capacity Retention Characteristics



Cycle Life vs. Depth of Discharge



Number of Cycles vs. Capacity



RECOMMENDED SETTING PARAMETERS

Item		48V Battery Bank	60V Battery Bank	72V Battery Bank
Charger Parameters	Max. Charge Voltage (V)	58.6V-59.0V	73.3V-73.7V	88.0V-88.2V
	Float Charge Voltage (V)	54.8V-55.2V	68.6V-68.9V	82.3V-82.7V
	Max. Charge Current (A)	4.5A-5.0A	4.5A-5.0A	4.5A-5.0A
	Shifting Current (A)	0.9A-1.0A	0.9A-1.0A	0.9A-1.0A
	Temperature Compensation Coefficient (mV/℃/Cell)	2.5~4.0mV/℃/Cell	2.5~4.0mV/℃/Cell	2.5~4.0mV/℃/Cell
Controller Parameters	Under-voltage Protection (V)	42V±0.5V	52.5V±0.5V	63V±0.5V
	Limited Current (A)	≤40A	≤40A	≤40A
	Turn-on Lock Current (A)	≤0.3A	≤0.3A	≤0.3A
Electrical Motor Parameters	Average Current (A)	≤20A	≤20A	≤20A
	Motor Power (W)	≤700W	≤750W	≤800W

** All the data and technical curves are for customer's reference only. This information is subject to change without any prior notice.*

For More Information, please contact:

CHAOWEI POWER CO., LTD.
No. 12 Zhizhou Ave., Zhicheng New Industrial Park, Changxing County, Zhejiang Province, China. 313100
Tel: +86 572 6200283 Fax: +86 572 6054410
Web: www.cnchaowei.com Email: info@chaowei.biz