FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Vývoj komunikační jednotky do elektrické motorky

Studijní program: B0714A270001 – Mechatronika
Studijní obor: Mechatronika
Autor práce: **Martin Tomek**
Vedoucí práce: Ing. Petr Bílek, Ph.D.
Ústav mechatroniky a technické informatiky

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Vývoj komunikační jednotky do elektrické motorky

Jméno a příjmení:	**Martin Tomek**
Osobní číslo:	M20000094
Studijní program:	B0714A270001 – Mechatronika
Studijní obor (specializace):	Mechatronika
Zadávající katedra:	Ústav mechatroniky a technické informatiky
Akademický rok:	2023/2024

Zásady pro vypracování:

1. Proveďte rešerši komunikačních jednotek používaných ve vozidlech. Seznamte se s jejich hardwarem a funkcemi.
2. Navrhněte vlastní prototyp komunikační jednotky. Vytvořte jeho blokové a elektrické schéma.
3. Zajistěte sběr potřebných dat dostupných na páteřní sběrnici CAN a zpřístupněte je na vzdáleném serveru v podobě databáze.
4. Volitelně doplňte zařízení o GPS modul, akcelerometr a realizujte hardwarovou přípravu pro wifi/bluetooth modul.
5. Realizujte modulární zařízení, navrhněte a vytvořte desky plošných spojů.
6. Vytvořte vývojový diagram a naprogramujte a odlaďte zařízení ve vybraném softwarovém nástroji.
7. Otestujte zařízení v laboratorních nebo i reálných podmínkách a zhodnoťte dosažené parametry.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30 až 40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	čeština

Seznam odborné literatury:

[1] ŽĎÁNSKÝ, BRONISLAV. Elektrotechnika motorových vozidel II. Brno: Avid, 2006. 212 s. ISBN 978-80-87143-14-8.

[2] NORRIS, DONALD. Programming with STM32: Getting Started with the Nucleo Board and C/C++. 2018. McGraw-Hill Education TAB. p. 304. ISBN: 9781260031324.

[3] Muhammad Ali Mazidi, Shujen Chen, Eshragh Ghaemi. STM32 Arm Programming for Embedded Systems (Volume 6). 2018. MicroDigitalEd. p. 378. ISBN: 978-0997925944.

[4] Datasheety vybraných obvodů.

Vedoucí práce:	Ing. Petr Bílek, Ph.D. Ústav mechatroniky a technické informatiky
Datum zadání práce:	12. října 2023
Předpokládaný termín odevzdání:	14. května 2024

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Černohorský, Ph.D.
vedoucí ústavu

V Liberci dne 12. října 2023

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval/a samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce.

Jsem si vědom/a toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom/a povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom/a následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

13. května 2024 Martin Tomek

Vývoj komunikační jednotky do elektrické motorky

Abstrakt

Práce se zabývá teoretickým návrhem a následnou realizací modulárního prototypu jednotky pro sběr dat na motorovém vozidle. Součástí práce je i průzkum historických a současně používaných zařízení pro záznam dat v dopravních prostředcích. Pří návrhu jednotky je zdokumentováno jak elektrické zapojení jednotlivých modulů, tak i vývoj firmware. Pro řízení se využívá vývojový kit Nucleo osazený procesorem ARM Cortex řady M4 spolu s vývojovým prostředím STM32CubeIDE. Do vzdálené databáze se ukládají data ze sběrnice CAN, dále ještě informace z 6-ti osého pohybového snímače a poloha GPS. Externí komunikace probíhá přes mobilní síť za využití MQTT protokolu.

Klíčová slova

sběr dat, CAN sběrnice, mobilní sítě a datový přenos, protokol MQTT, STM32, InfluxDB

Development of a communication unit for an electric motorbike

Abstract

Thesis deals with the theoretical design and subsequent implementation of a modular prototype of a data acquisition unit on a motor vehicle. Thesis also includes survey of historical and currently used data logging devices on vehicles. In the design of the unit, both the electrical connection of the individual modules and the firmware development are documented. The Nucleo development kit with an ARM Cortex M4 series processor and the STM32CubeIDE development environment are used for control. The remote database stores data from the CAN bus, as well as information from the 6-axis motion sensor and GPS position. External communication is via a mobile network using the MQTT protocol.

Keywords

data collection, CAN bus, mobile network and datatransfer, MQTT protocol, STM32, InfluxDB

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Petru Bílkovi, Ph.D. za pomoc při návrhu a financování této jednotky. Chci vyjádřit velké díky i svému dobrému příteli Lukáši Jenšíkovi za užitečné podněty a poskytnuté rady při realizaci. V neposlední řadě také děkuji všem členům své rodiny za psychickou pomoc a oporu při práci

Obsah

Seznam obrázků

Seznam grafů

Seznam tabulek

Seznam zdrojových kódů

Seznam zkratek

2G	označení 2. generace technologií pro přenos dat v mobilních sítích (zastaralá)
ASCII	Kódovací tabulka pro převod znaků anglické abecedy na 8bit čísla
CAN	„Control Area Network“ – typ datové sběrnice
CC/CV	„Constant Current / Constant Voltage“ – označení napájecího zdroje schopného regulovat svůj výstup jako stabilní zdroj proudu nebo stabilní zdroj napětí
DPS	Deska Plošných Spojů
EDA	„Electronic Design Automation“ – nástroj podporující tvorbu elektrických návrhů
GNSS	„Global Navigation Satellite System“ – obecný družicový navigační systém
GPS	„Global Positioning System“ – americký družicový navigační systém
HAL	„Hardware Abstraction Layer“ – sada rutin pro snazší a bezpečnější přístup k hardwarovým prostředkům zařízení
IAM	Produktová řada senzorů pro sledování pohybu od společnosti TDK InvenSense
IDE	„Integrated Development Environment“ – vývojové prostředí podporující tvorbu softwaru
IoT	„Internet of Things“ – obecný název obvykle jednoúčelového zařízení připojeného k internetu
LTE	„Long Term Evolution“ – technologie pro vysokorychlostní přenos dat v mobilních sítích
MCU	„Microcontroller unit“ – jednočipový počítač
MQTT	„Message Queuing Telemetry Transport“ – protokol pro výměnu dat
RTC	„Real Time Clock“ – modul pro udržování informace o reálném čase
SLA	„Service Level Agreements“ – dohoda o procentuální dostupnosti online služby
SMD	„Surface-Mounted Device“ – součástka osazená a pájená z jedné strany plošného spoje
TCU	Zkrácené označení této Telemetrické Komunikační Jednotky
UPS	„Uninterruptible Power Supply“ – zdroj napájení se záložním uložištěm energie
USA	„United States of America“ – Spojené státy americké

Úvod

V současné době probíhá na Technické univerzitě v Liberci vývoj čistě elektrického motocyklu. Tato bakalářská práce si klade za cíl vytvořit plně funkční prototyp jednotky pro sběr provozních dat z její páteřní sběrnice CAN. Pro reálné nasazení by jednotka měla ještě zaznamenávat informace o pohybu a poloze vozidla.

Na základě rešerše o používaných jednotkách ve vozidlech a z dosavadních zkušeností je vytvořen návrh celého zařízení. Aby bylo možné provádět úpravy a opravovat potenciální hardwarové chyby v designu, jsou jednotlivé funkční bloky vyrobeny jako samostatné moduly, které se propojují přes jednotný konektor. Vývoj jednotlivých modulů proběhne vždy standartní cestou od návrhu elektrického schéma, přes zhotovení a osazení DPS, oživení, naprogramování a začlenění do celku.

Oddělenou součástí jednotky je i server pro příjem dat společně s databází, připojený k pevné internetové síti. V práci je popsán datový protokol MQTT pro komunikaci a realizace databázového serveru za pomoci open-source projektů Eclipse Mosquitto a InfluxDB. Ty jsou nainstalované v linuxovém prostředí na mikropočítači Raspberry Pi.

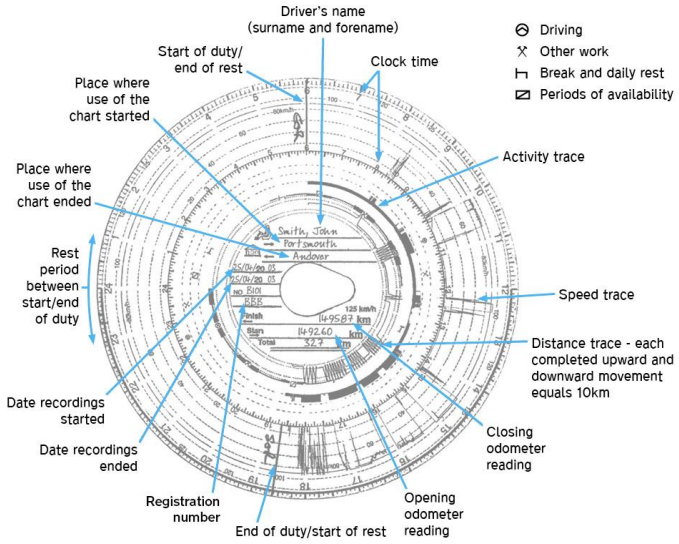
Datové spojení mezi jednotkou a vzdáleným serverem zajišťuje modul pro mobilní připojení LTE od čínské společnosti SIMCom Wireless Solutions Co. Ltd.. Záměrně byl zvolen typ SIM7600CE-T, který disponuje i GNSS rozhraním.

Přestože jednotka nemůže být kvůli svému modulárnímu řešení natrvalo začleněna do systému motocyklu, proběhne test jejího připojení na stacionárním stroji. Odděleně bude jednotka také testována v běžném silničním provozu.

1 Rešerše

1.1 Jednoduchá záznamová zařízení

Mezi první rozšířené záznamové zařízení ve vozidlech patří tachografy. Jedná se o zařízení ukládající základní údaje o rychlosti a ujeté vzdálenosti vozidla. Počátky sahají až do poloviny 19. století [1], tehdy se využíval pro železniční dopravu. V novodobé historii svůj primární užitek přináší stále hlavně v nákladní dopravě. Od papírových pásek nebo kroužků s analogovým zápisem, viz Obrázek 1.1, až po moderní digitální systémy často i připojené k internetu. Od roku 2006 musí být všechna vozidla nad 3,5 t vybavena tímto zařízením.



Obrázek 1.1: Analogové kolečko tachografu [2]

Event data recorder, případně accident data recorder, někdy také označovaný jako „černá skříňka“ pro automobily, je obvykle záznamové zařízení bez přístupu k internetu a slouží hlavně pro sběr dat během závady nebo havárie vozidla. U těchto jednotek ke kladen velký důraz na jejich odolnost a dobrou schopnost uchovávat data. Zaznamenávají obvykle jen po krátký časový interval, zato ale mají krátkou periodu vyčítání dat. Do vozidel jsou montovány jako dodatečné příslušenství, zpravidla tak činí firmy u svých služebních automobilů, nebo mohou být instalovány i na základě výhodnější smlouvy s pojišťovnou.

Pro delší monitoring se používají záznamníky dat, angl. „dataloggers“. Jedná se povětšinou o pasivní odposlechové zařízení připojené k hlavní datové sběrnici vozidla. Data jsou ukládána do vnitřní paměti a následně mohou být vyčtena do externího zařízení. Moderní záznamníky už disponují i bezdrátovým rozhraním, nejčastěji Wi-Fi nebo Bluetooth a mohou se připojit k mobilnímu telefonu nebo notebooku a data zobrazovat v reálném čase. Plánovaný projekt komunikační jednotky se bude do značné míry podobat tomuto typu zařízení. Mezi přímé alternativy se tak řadí například společnost CSS Electronics, která vyvíjí dataloggery pro CAN sběrnici [3]. Základní model CL1000 s cenou 170 € (cca 4200 Kč) dokáže zaznamenávat data na 8GB SD kartu nebo je zasílat živě přes USB. Verze CANedge1 s GNSS a snímačem pohybu stojí 430 € (cca 10 600 Kč). Nejvyšší konfigurace CANedge3, která nejblíže odpovídá požadavkům zamýšlené jednotky, disponuje určováním polohy přes GNSS, snímačem pohybu a připojením k mobilní síti LTE. Jeho cena činí 630 € (15 600 Kč) a postrádá možnosti pro bezdrátové

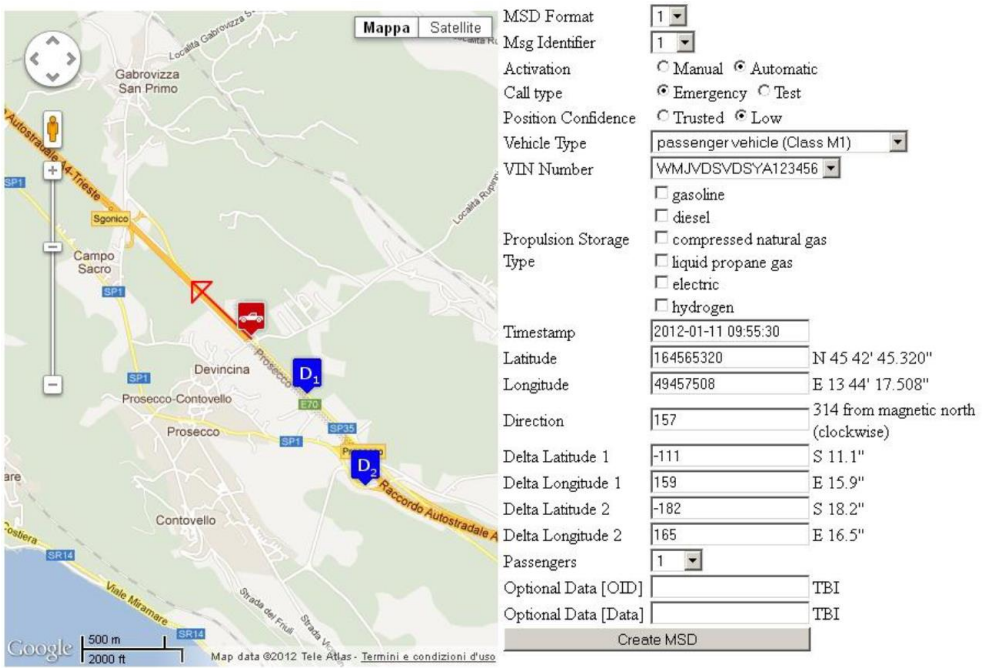
připojení na krátké vzdálenosti. V osobních automobilech se často používají i čínské bezdrátové diagnostiky OBD, jako záznamníky údajů z vozidla. Ceny sice začínají od nižších stovek korun [4], jejich případná použitelnost v univerzitním motocyklu je bohužel zamezena absencí vyšších vrstev CAN protokolu (ze strany motocyklu). Plánovaný rozpočet pro vývoj a tvorbu jednotky je určen na 7 000 Kč.

1.2 Pokročilé interaktivní systémy

S postupným vývojem elektroniky se na začátku tisíciletí začaly objevovat telemetrické jednotky. Do jisté míry přebírají funkce po svém předchůdci tachografu, zpravidla už jsou ale pouze elektronické a vždy umožňují přístup k internetu pro vzájemnou výměnu dat. Kromě zaznamenávání provozních dat tak slouží i pro příjem informací. Opětovně byl hlavním průkopníkem technologie těžký dopravní průmysl. Rozšířené informace o vozidlech poskytovaly více možností pro plánování vytížení a údržbu daných vozidel. V době vzniku měl však každý zúčastněný výrobce svůj vlastní typ. Díky nárůstu dostupného výpočetního výkonu pro levné čipy přešel vývoj na telematické jednotky. Ty už dokáží nejen zaznamenávat, ale i zpracovávat a vyhodnocovat nasbíraná data. Dále se systémy začaly integrovat i do osobních automobilů ve formě prémiových služeb spojených obvykle s vestavěnou navigací a rozšířeným infotainmentem. Ve snaze o standardizaci na poli těžkých strojů byl v roce 2010 ustanoven AEMP Telematics Data Standard, na kterém se shodla většina velkých výrobců těžké techniky jako je Volvo Construction Equipment, Caterpillar Inc nebo Komatsu.

Svou podstatou jednoduché telemetrické jednotky se ale stále používají. Jejich využití však změnilo cílovou skupinu. Nyní se používají pro velmi rychlé předávání informací a případně i pro vzdálenou konfiguraci u závodních vozidel, dronů nebo RC modelů. Mobilního operátora v tomto případě častěji nahradily signálové přijímače na trati.

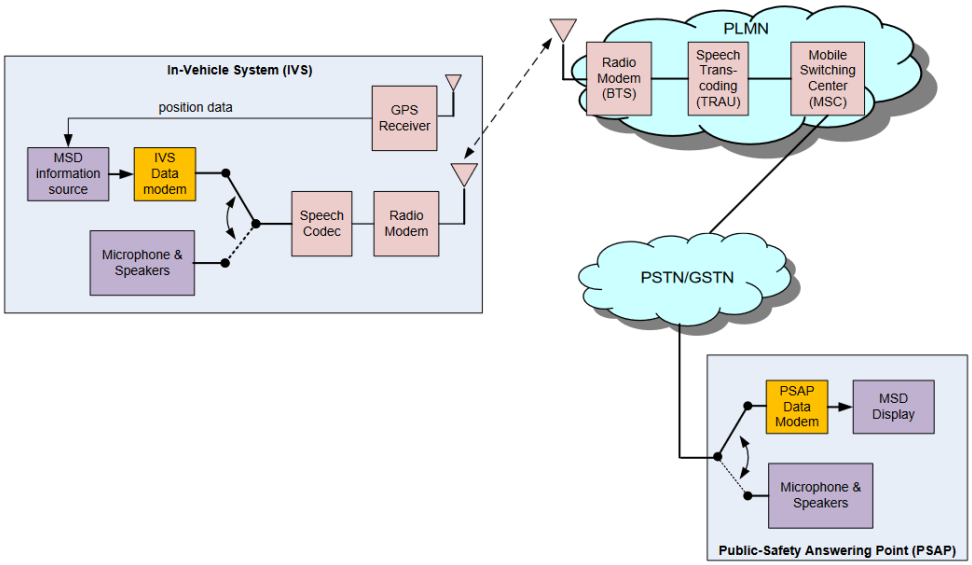
V Evropě prošel evolucí i záznamník při nehodě, který se přetransformoval do moderního systému eCall. Ten je připojený k datové sběrnici automobilu a pokud senzory vozidla zaznamenají vážnou dopravní nehodu, odešle přes mobilního operátora na krizovou linku 112 informace o přesné poloze z GPS společně se směrem jízdy, aktuálním časem a označením vozidla [5].



Obrázek 1.2: Vizualizace nouzové zprávy MSD v systému eCall [6]

Pasažéry vozidla také následně spojí s dispečerem ze složek IZS. Koncept systému byl představen společně se startem projektu Galileo, evropského družicového navigačního systému. Po několika odkladech bylo v roce 2018 nařízeno, že všechny nové typy automobilů do 3,5 t musí mít v sobě tento systém implementovaný. Ještě před tímto nařízením však už jednotlivé automobilky přicházely buďto s vlastními systémy o pomoc v nouzi, nebo s přímou implementací systému eCall. Vozidla značky Volvo, známé svou vynikající bezpečností, měla od roku 2006 příplatkový systém „Volvo On Call“ [7], který plnil stejné funkce jako pozdější eCall. Aby mohl výrobce automobilu využívat systém eCall, musí si zakoupit licence jeho patentu. Společnost Avanti nabízí licence pro certifikaci eCall za cenu $3 za každý vyrobený automobil. Pokud by výrobce od této společnosti chtěl ještě licence pro mobilní síť 4G, celková cena se vyšplhá až na $20 [8]. Alternativou k evropskému systému eCall je v Rusku systém ERA-GLONAS, napojený na stejnojmenný družicový navigační systém GLONASS.

Velká část výrobců, zabývajících se moduly pro připojení k mobilním sítím, přišli s vlastní implementací systému eCall. Od společnosti u-blox se jedná například o GSM brány SARA-G3, LISA-U2 nebo modul, který má í s vestavěný GNSS receiver, MAX-7 [9].



Obrázek 1.3: Znázornění funkce eCall systému společností Telit [6]

Obrázek 1.3 nastiňuje základní strukturu systému eCall. Z GPS modulu jsou získávány informace o aktuální poloze a směru vozidla (MSD). Ty jsou neustále ukládány do ovládací jednotky. Poté co dojde k havárii systém vyšle přes mobilního operátora nouzovou zprávu s MSD informacemi na vzdálený server a přepne komunikaci na hlasový hovor. Vzdálený server předá lokalizační data dispečerovi IZS a připojí ho k hlasovému hovoru pro další pomoc.

1.3 Aktuální vývoj a budoucnost

V současnosti dochází po celém světě k vypínaní zastaralých mobilních sítí. To sebou nese řadu problémů pro starší vozidla vybavené telematickou či telemetrickou jednotkou. Například ve Velké Británii došlo k vypnutí části 2G sítě [10]. Majitelé elektrických vozidel Nissan Leaf vyrobených před rokem 2016 se kvůli tomu ocitli bez možnosti ovládání svého vozu

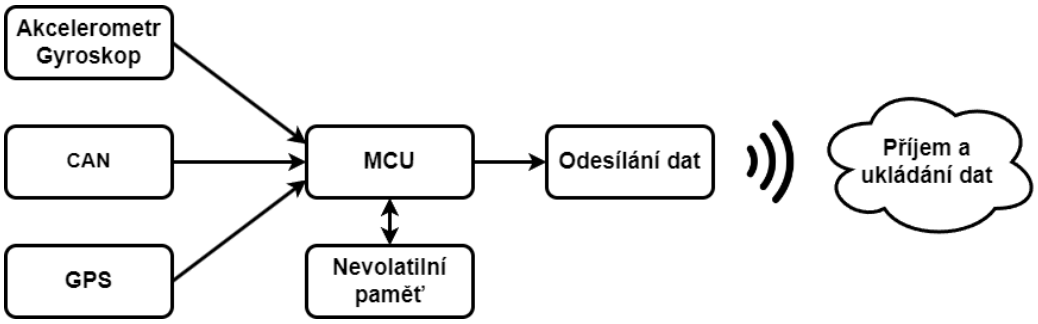
přes mobilní aplikaci [11]. Z této politováníhodné situace by mělo do budoucna vzejít alespoň pár zásad vývoj komunikačních technologií. Předně to, že není vhodné při novém vývoji setrvávat u zastaralých technologií. Na druhou stranu by ale technologický pokrok měl brát ohled na současné technologie a snažit se co nejvíce zachovávat alespoň jejich základní podporu i do budoucna pro zabránění tvorby nadbytečného elektro-odpadu.

Nyní zpět k novým technologiím. Ve spolupráci pražského ČVUT a brněnského VUT vzniká přímo na českém území nový bezpečnostní systém, který se v budoucnosti možná dočká i zákonného nasazení. Jedná se o ELORYKS, systém, který by měl zabránit ujíždění jedinců, kteří nechtějí zastavit na výzvu policisty. V běžném automobilu by měla být buď na páteřní sběrnici, nebo přímo na palivové čerpadlo připojena přijímací jednotka systému ELORYKS s rozhraním ITS G5 [12]. Jedná se o komunikační standard na krátké vzdálenosti využívající frekvenční pásmo okolo 5,9 GHz. Jeho primární využití má být v systému V2V (vehicle to vehicle), případně rozšířeno do V2X (vehicle to anything). V případě systému ELORYKS by se měla jednotka spojit s policejním vozidlem a po ověření přes vzdálený server napojený na policejní databázi bude možno u konkrétního vozidla vypnout palivový přívod. Bohužel se jedná o tzv. Bezpečnostní výzkum, proto není možné získat o tomto projektu více informací.

2 Realizace jednotky

2.1 Jádro jednotky

Z požadavků na výslednou komunikační jednotku lze okamžitě určit její základní strukturu, znázorněnou na Obrázek 2.1. Ústředním bodem celé jednotky je mikrokontrolér. Od něj se dále odvíjí i volba ostatních komponent.



Obrázek 2.1: Diagram komunikační jednotky

Na základě používání mikrokontrolérů řady STM32 v laboratoři, kde vznikl prototyp elektrického motocyklu byl vybrán podobný procesor pro tuto jednotku ze stejné řady. Čip STM32L476RG [13] je 32-bit mikrokontrolér s procesorovým jádrem ARM Cortex M4, 1MB Flash a 128kB SRAM pamětí. Mezi jeho periferie patří mimo jiné i RTC obvod nebo několik kanálů UART, SPI a I2C sběrnic. Také umožnuje velmi užitečnou obsluhu CAN sběrnice. Aby se vývoj TCU zjednodušil a také urychlil, použije se již hotový vývojový kit Nucleo-L476RG se stejnojmenným MCU. Tento kit je osazený Morpho konektorem, ze kterého budou vycházet i všechny ostatní moduly.

Z prvotního záměru programování v prostředí Arduino IDE se vývoj přesměroval do komplexnějšího STM32CubeIDE od STMicroelectronics. To proto, že na univerzitě je to standartně používaný program a bude tak ulehčena přenositelnost zdrojového kódu a jeho další výzkum.

Součástí tohoto prostředí je i nástroj IOC, který umožnuje snadný výpočet a následné nastavení hodinového rozvodu v mikrokontroléru. Také v něm lze nakonfigurovat použité sběrnice v příjemném uživatelském prostředí. Generování částí kódu pak probíhá automaticky. Grafická vizualizace použitých pinů MCU značně ulehčuje orientaci při dalším návrhu schémat a desek plošných spojů.

Když jsou známy celkové možnosti a dostupné sběrnice hlavního MCU, mohou se začít navrhovat moduly pro obsluhu jednotlivých požadavků ze zadání jednotky.

2.2 Snímač pohybu

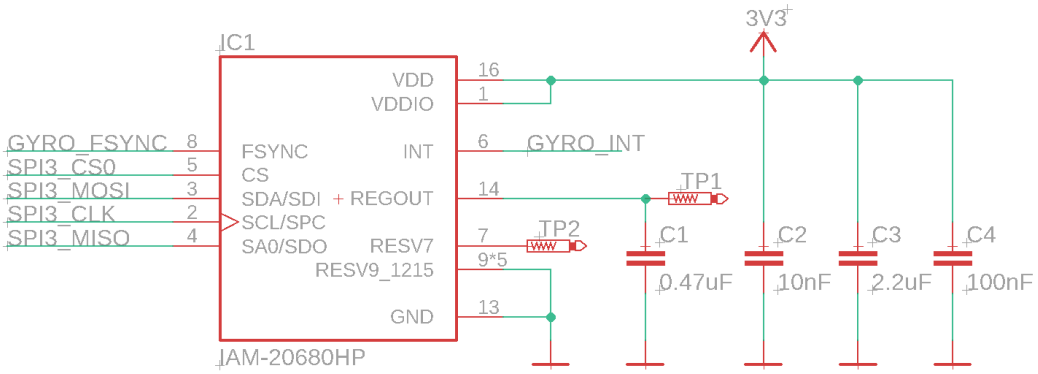
2.2.1 Požadavky na modul

Tuhé těleso, reprezentující pevně zabudovanou komunikační jednotku v motocyklu, má v trojrozměrném prostoru 6 stupňů volnosti – translace ve třech na sebe kolmých směrech a rotace okolo nich. Pro přesné určení míry změny těchto pohybů je zapotřebí využít 6ti osý snímač pohybu. Americká společnost TDK InvenSense je na tomto poli jedním ze světových leaderů s velkým portfoliem snímačů. Pro jednotku je vybraný IAM-20680HP, 6ti osý snímač s volitelným rozsahem a 16bit rozlišením. Může komunikovat přes rozhraní I2C nebo i SPI. Také splňuje normu AEC-Q100 třídy 2, tedy je certifikovaný pro nasazení v automobilovém průmyslu s operační teplotou od -40 °C do 105 °C.

2.2.2 Elektrické schéma a DPS

Zapojení IAM senzoru je převzaté z oficiálního doporučení výrobce TDK InvenSense [14].

Kondenzátory C2 a C4 slouží hlavně jako filtrace AC rušení na přívodním vodiči. Kondenzátor C3 díky své větší kapacitě vytváří miniaturní zdroj napětí, který zamezuje výkyvům napájecího napětí při náhlé změně spotřeby snímače. Snímač v sobě ještě obsahuje další regulátor napětí používaný jako reference pro fyzické senzory uvnitř čipu. Aby jeho hodnota byla co nejvíce stabilní, musí ke svému výstupu mít připojený kondenzátor C1, plnící obdobnou funkci jako výše zmíněný kondenzátor C3.



Obrázek 2.2: Elektrické schéma okolo čipu IAM-20680

Testovací body TP1 a TP2 přivedené od dvou pinů pouzdra LGA16, viditelné na Obrázek 2.2, slouží pro ověření správného připájení komponenty. Ostatní piny jsou vyvedeny na hlavní konektor a lze tak jednoduše ověřit, zdali nedošlo k nadměrnému rozlití pájky a zkratu dvou sousedních pinů. Pouzdro LGA16 má totiž rozteč pinů pouze 0,5 mm. Piny jsou také umístěny jen zespodu součástky. Pájení v domácích podmínkách je tak poměrně náročný proces a protože optická kontrola je v tomto případě nemožná, musí se spoje zkontrolovat elektricky pomocí multimetru.

V době zadávání výroby DPS však stále doznívala tzv. „čipová krize“. Jedná se o období všeobecného nedostatku polovodičových součástek způsobené hlavně pandemii koronaviru Covid-19. Krátkodobý výpadek zasáhl i vybraný snímač, který byl v obchodě Mouser.com několik dní nedostupný. Proto byl do návrhu jednotky přidán ještě druhý snímač MPU-6050 od stejné společnosti, který se mohl okamžitě vypreparovat z čínského vývojového modulku

GY-521. Po odložené objednávce už byl původní snímač opět naskladněn, na finální DPS tak jsou osazeny oba dva typy. Během přenášení podobvodu z celkového elektrického schéma na modulovou desku se však přehlédnuly pull-up rezistory pro I2C sběrnici. Při případném použití této sběrnice je tedy potřeba rezistory v rozsahu 2 kΩ až 10 kΩ připájet od obou linek k napájecímu napětí.

2.2.3 Programování

Díky certifikaci AEC má snímač IAM-20680HP vyšší užitnou hodnotu a dále se bude vyvíjet pouze on. Aby se dalo lépe pracovat s jinými částmi jednotky ve smyslu generování reálných dat, je tento modul programován jako první. Ve snaze dalšího zrychlení práce je využita volně šiřitelná knihovna [15] pro obsluhu podobného snímače ICM-20689. Bohužel je tato knihovna psána v programovacím jazyce Wiring, určeném pro platformu Arduino. Ten přesto staví za základech jazyka C/C++ a dá se proto upravit. Základní syntaxe stejně jako práce s funkcemi, proměnnými nebo třeba bitové operace jsou v obou jazycích shodné a mohou zůstat bez větších zásahů. Komplikace nastávají při používání specializovaných knihoven a s nimi spojených funkcí. Určité funkce mají v obou prostředích ekvivalentní náhradu. Jako příklad poslouží funkce časového pozastavení toku programu `delay()` z Wiringu přenesená jako `HAL_Delay()` do prostředí STM32CubeIDE. Jiné funkce, obvykle pracující se sběrnicemi, potažmo tak i s hardwarem, jsou si natolik odlišné, že je potřeba přepsat celou funkci, která je využívá. Zdrojový kód 2.1 zobrazuje jednu z nově napsaných funkcí, tato konkrétně slouží k zápisu registru do snímače po datové sběrnici SPI.

```
int IAM20680::writeRegister (uint8_t Address, uint8_t Data) {
   HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_RESET);
   HAL_SPI_Transmit(_spi, (uint8_t *)&Address, 1, 100);
   HAL_SPI_Transmit(_spi, (uint8_t *)&Data, 1, 100);
   HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_SET);
   HAL_Delay(10);
   readRegisters(Address, 1, _buffer);
   if (_buffer[0] == Data) {
      return 1;
   } else {
      return -1;
   }
}
```

Zdrojový kód 2.1: Přepsaná funkce knihovny IAM

Používání již hotových knihoven má své výhody, v tomto případě její autor, Inhwan Wee MSc., odvedl spoustu práce při psaní obslužného kódu pro nastavení rozsahů, kalibraci senzoru nebo také transformaci a přepočet ze surových dat na reálné hodnoty. Na druhou stranu ale stejně knihovna potřebuje další úpravy a její již hotové části snižují hloubku programátorova náhledu do dané problematiky. V dalších částech vývoje jednotky jsou proto dodatečné knihovny vytvářeny od nuly.

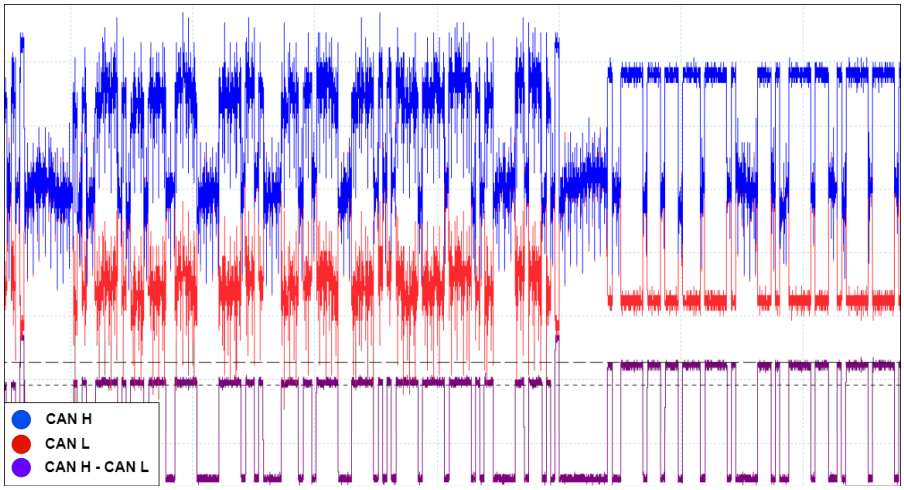
2.3 CAN

2.3.1 Základy sběrnice a datového rámce

Vývoj sériové datové sběrnice CAN započal roku 1983 ve společnosti Robert Bosch GmbH. V roce 1993 byl zaveden standard ISO 11898 určující dvě spodní vrstvy datové komunikace v rámci referenčního modelu ISO/OSI. Kvůli odlišným požadavkům v jednotlivých odvětvích bylo vytvořeno více standardů pro protokoly vyšších vrstev.

Univerzitní motocykl v současnosti pracuje pouze se základními protokoly CAN sběrnice a proto bude dále popsána jen použitá fyzická a linková vrstva, společná pro většinu vyšších CAN protokolů.

Sběrnice využívá pro fyzický přenos dat diferenciální signalizaci. To znamená, že logický stav odesílaného bitu není určen hodnotou napětí signálové linky vůči společnému bodu, obvykle zápornému pólu napájení. Na místo toho se určuje z rozdílu napětí mezi dvěma datovými vodiči sběrnice. Budič sběrnice v klidovém stavu udržuje na obou linkách sběrnice zhruba polovinu pracovního napětí (obecně 2,5 V). CAN protokol tento stav definuje jako logickou „1“ a jedná se o recesivní stav sběrnice, tedy že ho může kterýkoliv další vysílající budič změnit. Dominantní stav logické „0“ se provede zvýšením napětí linky CAN H blíže k vrcholnému napětí a snížením CAN L k nule. Tím vznikne mezi jednotlivými vodiči diferenciální napětí. Tento přenos má mnoho výhod. Mezi jeho největší přednosti patří relativně velká odolnost proti rušení. Proto jsou běžně vodiče sběrnice realizovány pouze kroucenou dvoulinkou bez dalšího stínění. Ze zdroje rušení se obvykle naindukuje stejná úroveň napětí na obou vodičích, tudíž hodnota jejich diferenciálního napětí zůstává stejná. V jednotném rozvodu, například u automobilu, kdy jsou všechny jednotky připojené ke společnému potenciálu – obvykle k rámu vozidla a zápornému pólu akumulátoru, mohou neizolované budiče sběrnice pracovat i s jedním přerušeným datovým vodičem. Systém ale ztrácí svou odolnost vůči rušení. Z převzatého Obrázek 2.3 je vidět velmi dobrá rozlišitelnost bitů jednoduchým odečtením dvou signálů i na velmi zarušené CAN sběrnici.



Obrázek 2.3: Vliv šumu na přenos dat po CAN sběrnici [16]

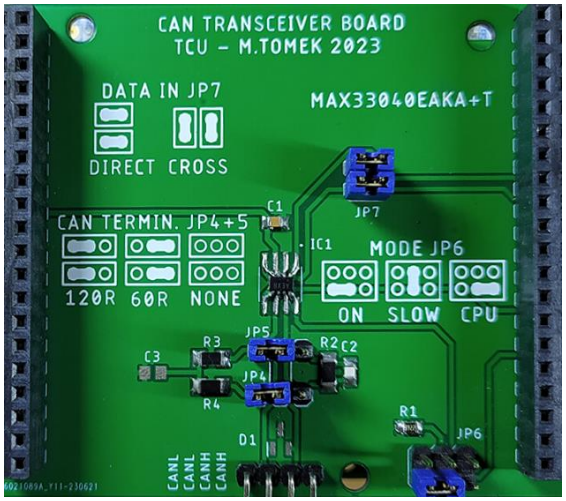
Kolizní detekce pro zabránění vysílání dvou jednotek ve stejnou dobu spolu s určováním priorit je chytře vyřešena pomocí identifikátorů jednotek. Čím je identifikátor menší, tím více dominantních „0“ daná jednotka vysílá a tak má větší prioritu. Pokud by ve stejný okamžik začala

vysílat ještě další jednotka s vyšším identifikátorem, v okamžiku odeslání prvního nestejného bitu, který bude zákonitě recesivní „1“, by měla druhá jednotka okamžitě přerušit své vysílání a náhodný čas počkat. K dalším částem datového rámce patří mimo samotné zprávy i informace o její délce. Pro kontrolu správného obsahu zprávy je zde obsažen cyklický redundantní součet (CRC). Ke svému konci obsahuje datový rámec ještě potvrzovací (acknowledge) bit. Během této doby aktivní jednotka vysílá recesivní bit a očekává příjem dominantního bitu od jakékoliv jednotky. Když tato informace přijde, znamená to, že minimálně jedna další jednotka zprávu zaslechla a správně přijala. Pokud se tak nestane, obvykle se jednotka pokusí vyslat zprávu znovu, což může v krajním případě zahltit budič a uvést ho do chybového stavu.

2.3.2 Elektrické schéma a DPS

Běžné CAN budiče pracují s operačním napětím 5 V. Pro snížení nákladů a komplexnosti napájecího modulu je ale vybrán čip MAX33040EAKA+T od společnosti Analog Devices. Tento budič pracuje s napájecím napětím 3,3 V. Nemá v sobě žádnou nábojovou pumpu ani DC/DC měnič, jím generované maximální napětí na sběrnici tak je zhruba rovné jeho napájecímu napětí. I přes to dokáže pracovat na 5V CAN sběrnici, pro rozlišení logické „1“ bitu totiž dle standardu CAN stačí diferenciální rozdíl napětí 1,5 V. První dvojice hřebínků s propojkami slouží jako prohazovač vstupního signálů, kdyby snad došlo při návrhu k záměně linek. Na výstup budiče je přímo připojen signálový konektor (hřebínek) a pomocí dalšího páru propojek lze ke sběrnici paralelně připojit 60Ω nebo 120Ω rezistory sloužící jako terminátor/y sběrnice.

Špatným přečtením technické dokumentace k budiči byl na DPS modulu vybrán půdorys součástky SOIC namísto odpovídajícího SOT23. Rozvržení jednotlivých pinů zůstává stejné, mění se ale rozteče a celková velikost pouzdra. Při osazování je tedy nutné lehce improvizovat a s využitím pomocných drátků budič správně připájet. Úprava je vidět na Obrázek 2.4.



Obrázek 2.4: Osazená deska CAN modulu

2.3.3 Programování

Z dříve zmíněného datového rámce vyplývá, že pro test komunikace je potřeba mít na sběrnici alespoň dvě aktivní zařízení, jinak nedojde k odezvě a potvrzení ACK bitu. Proto je ke CAN sběrnici jednotky připojena ještě CAN/USB brána od společnosti Ixxat Automation GmbH. K ní se využívá obslužný program „canAnalyser3 Mini“, který poskytuje jednoduchý přehled

o zprávách probíhajících po sběrnici. Brána také umí zprávy posílat a to dokonce i cyklicky, s nastaveným časovým intervalem.

Vybraný mikrokontrolér je pro práci s CAN sběrnicí velice dobře připraven. Dokáže pracovat s přenosovou rychlostí až 1 Mbit/s a zvládá přijímat rámce se standardním 11 bitů dlouhým identifikátorem nebo i prodlouženým identifikátorem na 29 bitů. S pokročilou HAL knihovnou jsou přijímané zprávy automaticky dekódovány a uloženy do jednoho ze dvou FIFO registrů. Odpadá tak problém s přesnou synchronizací a nutnosti použití přerušení pro náhodný příjem dat. Pro vyzkoušení horních mezí mikrokontroléru je nakonfigurována nejvyšší povolená přenosová rychlost se vzorkovacím bodem v 81,25 % délky bitu.

Díky své jednoduchosti není v projektu ani vytvořena vlastní knihovna, jako tomu je u jiných modulů, ale celý obslužný kód je napsaný v jedné sekci hlavní větvě programu. Pokud funkce `HAL_CAN_GetRxFifoFillLevel()` zjistí, že se v zásobníku objeví nová zpráva, je okamžitě přečtena a na základě identifikátoru odesílatele jsou data za pomocí funkce `switch()` uložena do předdefinované části paměťového pole `MC[]` pro budoucí odeslání nebo uložení.

```
uint8_t can_RX_buffer[8];
…
if (HAL_CAN_GetRxFifoFillLevel(&hcan1, CAN_RX_FIFO0) > 0) {
   HAL_CAN_GetRxMessage(&hcan1, CAN_RX_FIFO0, &can_rx_cfg, can_RX_buffer);
   switch (can_rx_cfg.StdId) {
      case 0x201:
         for (int i = 0 ; i < 1 ; i++) {
            MC[124 + i] = can_RX_buffer[0 - i];
         }
         break;
      case 0x202:
         for (int i = 0 ; i < 4 ; i++) {
            MC[125 + i] = can_RX_buffer[3 - i];
         }
         break;
      case 0x301:
         for (int i = 0 ; i < 2 ; i++) {
            MC[129 + i] = can_RX_buffer[1 - i];
         }
         break;
      …
   }
```

Zdrojový kód 2.2: Roztřídění a uložení přijaté zprávy z CAN

Takové řešení má určitou nevýhodu ve své malé flexibilitě, pokud by se v budoucnu do motocyklu přidávaly další jednotky s novým identifikátorem nebo jen jinou délkou zprávy, musí se zde upravované údaje pozměnit.

2.4 Paměť

2.4.1 Výběr paměti

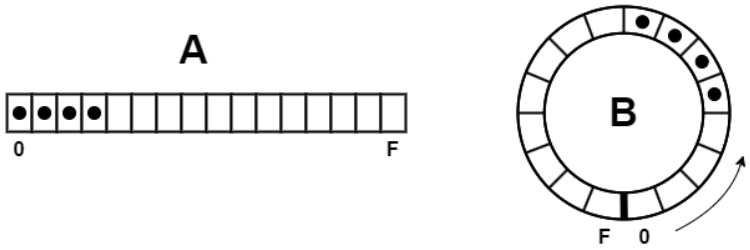
Jedním z cílů komunikační jednotky je co možná nejstabilněji zaznamenávat data. Pokud by byla použita pouze mobilní internetová brána, v případě výpadku sítě by se odesílaná data okamžitě ztratila, nebo by se mohla uložit do provozní proměnné mikrokontroléru. Jeho paměť je však malá a co hůře, je také volatilní. Dokáže tedy udržet informaci pouze pokud je napájena. Proto je do jednotky zařazen i modul s externí nevolatilní pamětí.

Prvotně zvažovaná paměť byla typu FRAM. Mezi její výhody patří velká zápisová rychlost, lepší přístupnost k jednotlivým bytům a také se nemusí před zápisem mazat. Kvůli své nedostupnosti, opět způsobené „čipovou krizí“, společně s malou kapacitou cenově dostupných čipů byl upřednostněn typ paměti Flash. Tato paměť se dá pořídit buďto ve formě SMD součástky, nebo také jako přenosné médium SD/MicroSD karty. Protože je jednotka navrhována s ohledem na možnou implementaci do uzavřeného boxu pro montáž na motocyklu, byla by přenositelnost média značně ztížena a také by mohlo během provozu dojít k uvolnění karty. Jako použitá paměť je vybrána sériová paměť o velikosti 1 Gb řady W25N01GV od společnosti Winbond.

2.4.2 Programování

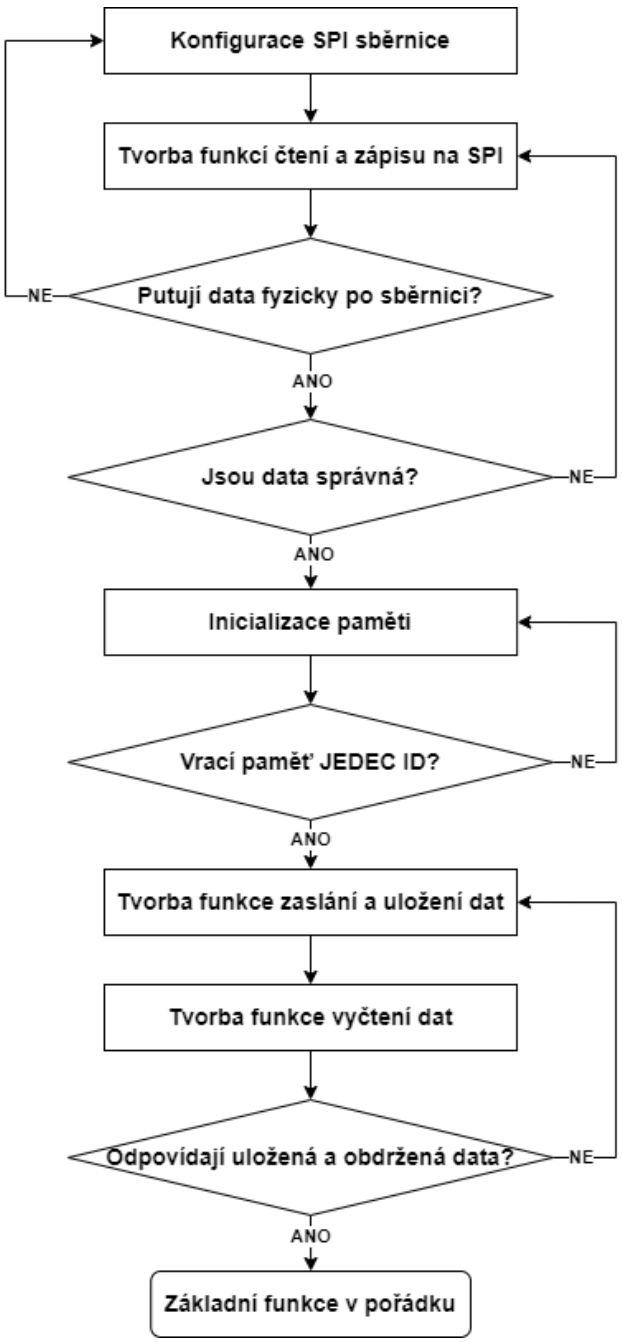
Ještě před tvorbou samotného kódu je potřeba zhodnotit možnosti a případné slabiny vybrané paměti. K tomu poslouží datový list výrobce [17]. Vybraná paměť je rozdělena na 65 536 stránek, z nichž každá obsahuje 2 048 bytů. Paměť obsahuje kromě datového pole ještě vstupní buffer o velikosti jedné stránky. Pro uložení informací se musí data nahrát do tohoto bufferu a z něj pak separátním příkazem zapsat na specifikovanou stránku v poli. Ukládání dat se prování vždy do jedné stránky a před zápisem musí být stránka vymazána deklarovaným příkazem. Bohužel stránky nejdou mazat samostatně, ale musí se vždy brát po předem určených blocích o velikosti 128 KB, což odpovídá 64 stranám. Další nevýhodou flash paměti je relativně malý počet zápisů do jednotlivých buněk, výrobcem uváděných je 100 000. Paměť na konci každé stránky alespoň obsahuje ještě část pro zápis hodnot samoopravného kódu, ten dokáže opravit až 4 bitové chyby.

Rozdíly v přístupu k paměti jsou zobrazené ve zjednodušeném Obrázek 2.5. Když by se paměť používala vždy od počáteční adresy jako zásobník (A), docházelo by k nadměrnému používání této části paměti a hrozilo by její předčasné poškození. Proto bude využit princip fronty s plovoucím začátkem a koncem společně s kruhovým uspořádáním paměti (B). Tím se docílí naprosto rovnoměrného opotřebení všech buněk v paměti.



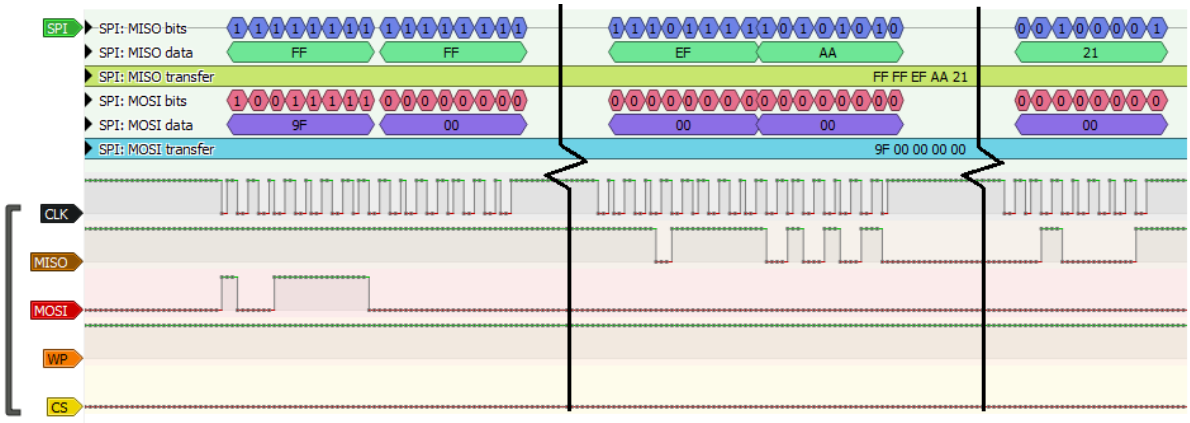
Obrázek 2.5: Lineární a kruhový přístup k paměti

Během psaní programu je nejprve potřeba zajistit funkční a správnou komunikaci mezi mikrokontrolérem a čipem paměti. Pro ujasnění postupu je vytvořen vývojový diagram, viz Obrázek 2.6.



Obrázek 2.6: Vývojový diagram pro zprovoznění externí paměti

Ověřování správnosti dat putujících po SPI sběrnici by se s dostupným osciloskopem Agilent DSO3062A provádělo jen velmi obtížně kvůli jeho malé paměti. Namísto něj je použit logický analyzér s klonem čipu Cypress FX2 a open-source firmwarem fx2lafw. Pro vizualizaci slouží program PulseView, který zároveň může sloužit i jako dekodér dat a ještě více tak usnadní práci. Vyčtené JEDEC ID (EF AA 21) je na Obrázek 2.7.



Obrázek 2.7: Datový přenos SPI sběrnice při výpisu JEDEC ID

Základy obslužného programu byly inspirovány knihovnou WinbondW25N [18] od uživatele squaresausage na GitHubu. Protože se ale opět jednalo o knihovnu pro vývojové prostředí Arduino IDE a některé funkce by se špatně přetvářely pro potřeby kruhového přístupu k paměti, bylo od jejího použití upuštěno. Nicméně díky jednoduchému a srozumitelnému strukturování původní knihovny se tyto principy propsaly i do vlastního programu.

Už během programování základních procedur byly do kódu vloženy užitečné funkční prvky pro snadnější budoucí práci s paměťovým polem. Například při skládání datové zprávy ve Zdrojový kód 2.3 je do paměti ukládán příznak o obsazení dané stránky (odpověď 0x2A) společně s celkovou délkou ukládaných informací, jedná se o části pole `send[3-4]`.

```
int W25N::uploadToBuffer(uint8_t* Data, uint8_t Length){
   while(isBusy()) {}
   writeEnable();
   uint8_t send[3+Length];
   send[0] = RAN_PROG_DATA_LOAD;
   send[1] = 0x00;
   send[2] = 0x00;
   send[3] = 0x2A;
   send[4] = Length;
   for (int i = 5 ; i < 3+Length ; i++) {
      send[i] = Data[i-3];
   }
   dataTransmit(send, 3+Length);
   return 1;
}
```

Zdrojový kód 2.3: Funkce pro odeslání dat do vstupního bufferu pamětí

Aby správně fungoval výše zmíněný kruhový přístup k paměti, musí si program stále udržovat povědomí o začátku a konci uložené fronty. Ukládáním těchto informací do předdefinovaného prostoru paměti by se zničila podstata rovnoměrného opotřebení buněk. Po startu mikrokontroléru je proto nutné celý paměťový prostor projít a najít začátek a konec fronty. Hledání začíná od nulté adresy paměti. Pokud je tato adresa obsazená, znamená to, že se program nachází uvnitř uložené fronty, pro nalezení vrchní adresy se tedy prohledávají stránky v kladného směru, dokud se nenarazí na první neobsazenou stránku. Její adresa je uložena jako vrchol fronty. Pokud v nulté adrese paměti není stránka obsazena, proces hledání probíhá v záporném směru do prvního nalezení obsazené stránky. Hledání spodního prvku fronty probíhá opačnými směry. Během hledání fronty se hlídají ještě stavy prázdné nebo naopak plně obsazené paměti.

Při běžné práci s frontou se musí hlídat celkové naplnění paměti a zároveň se nesmí odstranit poslední člen fronty při úplném vyprázdnění, aby se zachovala informace o její pozici. Jinak by docházelo k pravidelnému startu od počátku datového prostoru. Zápis nového prvku probíhá standardně na nejvyšší pozici fronty, pokud je ale odebrán její nejnižší prvek, nelze ho smazat ihned, ale až když je určen k vymazání celý blok, viz úskalí při programování zmíněné výše. Při běžném použití se nejedná o žádný problém, při restartu jednotky se ale běžně stává, že zprávy z minulé relace jsou již jednou odeslané ale z paměti fyzicky ještě nesmazané. Jednotka je tak na server pošle znovu. Server je na tuto možnost však připraven a nedochází k žádné chybě nebo kolizi.

Aby se zpracovávaná data nemusela vícekrát transformovat, jsou je většině případů uložena ve stejném formátu, jako byla přečtena. Jednotlivé položky přijímané z CAN sběrnice mohou zabírat vždy maximálně 1 byte, údaje z GPS jsou přijaty jako jednotlivé znaky, které mají v ASCII také hodnotu do 1 byte. Výjimku tvoří data ze snímače pohybu. Kvůli snazšímu zpracování informací jsou tato data ukládána do proměnné typu `double`. Pro mikrokontrolery STM32 to znamená, že zabírají 64 bitů operační paměti. Aby je bylo možné uložit do 8-bitových buněk, musí se proměnná rozdělit a její části separátně zapsat do externí paměti.

Jedno z možných řešení by využívalo bitové operace `<<` a `&`. Nejdříve by se provedlo maskování a poté posun bitů pro uložení vždy části jedné proměnné do paměťové buňky. V již používané knihovně `string.h` se ale nachází také funkce `memcpy`. Jak již její název napovídá, funkce dokáže kopírovat zadaný počet bytů z jedné části paměťového prostoru do jiného, určeného pouze ukazateli. Díky tomu je možné přehlédnout datové typy a elegantně tak ukládat mezi sebou nesourodé proměnné.

Pro rychlé otestování poslouží pole `accel[]` deklarované jako `double` a druhé pole `MC[]` tvořené `uint8_t`. Náhodné desetinné číslo v `accel[0]` se uloží do části pole `MC[]` a zpět zase vyčte do druhého prvku pole – `accel[1]`. Krátkým výpisem na sériový terminál je ověřeno správné chování programu. Přesnou podobu testovací procedury zobrazuje Zdrojový kód 2.4.

```
double accel[2];    // datove pole zrychleni
uint8_t MC[20];     // datove pole pro ulozeni do pameti
…
accel[0] = 315.7415;
memcpy(&MC[3], &accel[0], sizeof(accel[0]));
memcpy(&accel[1], &MC[3], sizeof(accel[1]));

sprintf(buffer," Double stored in array: %lf\r\n", accel[1]);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 100);
_____________________________________________________________________________
> Double stored in array: 315.741500
```

Zdrojový kód 2.4: Test ukládání rozdělené proměnné „double“

2.5 Internetová brána a GPS

2.5.1 Výběr modulu

Populárním a dostupným řešením pro připojení k mobilním sítím jsou moduly od SIMCom Wireless Solutions Co.. Ta sice byla v roce 2017 akvizována švýcarskou společností u-blox Holding AG, známou především výrobou GPS/GNSS modulů, své portfolio si však stále drží v nízkých cenových hladinách. Pro jednotku byl vybrán modul SIM7600CE-T dostupný v open-source vývojovém kitu od DFRobot. Ten podporuje kromě klasického standardu GSM také novější LTE-FDD nebo rychlé LTE-CAT 4. Má v sobě rovněž zabudovaný i samostatný GNSS obvod podporující standardy GPS, Galileo, BeiDou a GLONASS.

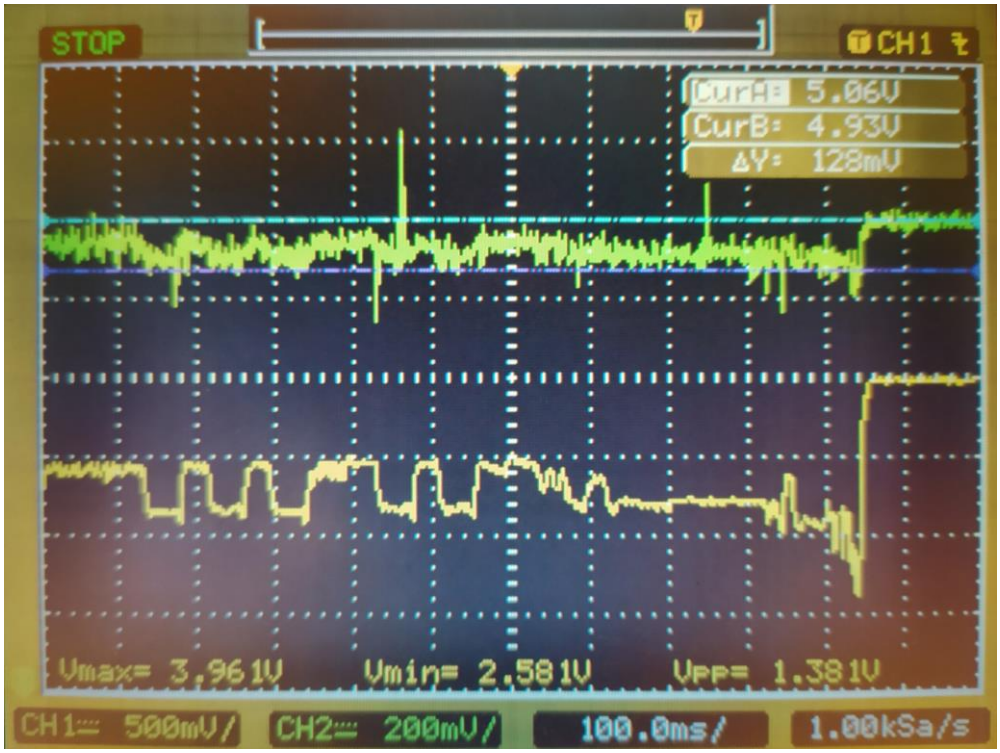
Modul obsahuje vlastní firmware a při správném zapojení může pracovat dokonce jako plnohodnotný telefon, obsahuje totiž výstup na displej, klávesnici, reproduktor a mikrofon. Pro použití v jednotce bude ale plně dostačovat pouze sériová linka, po které lze modul ovládat standardizovanými AT příkazy.

2.5.2 DPS a úpravy

Protože je modul původně určen jako shield pro vývojový kit Arduino, používá mimo jiného konektoru také 5V napěťovou hladinu pro své vstupy a výstupy. Pro komunikaci s mikrokontrolérem je tedy potřeba využít měnič napěťové úrovně. 4-bitový překladač napěťové úrovně TXU0204-Q1 od Texas Instruments je ideálním kandidátem. 4 bity reprezentují 4 linky, dvě zvyšující a dvě snižující, je tak možné překládat dvě UART sběrnice současně. Modul SIM7600 má totiž v základní konfiguraci zvlášť vyvedenou datovou komunikaci pro hlavní GSM čip a pro GNSS obvod. V konečném řešení je ale datová komunikace z GNSS vedena také přes hlavní čip SIM modulu a dvě linky tak zůstávají nevyužité.

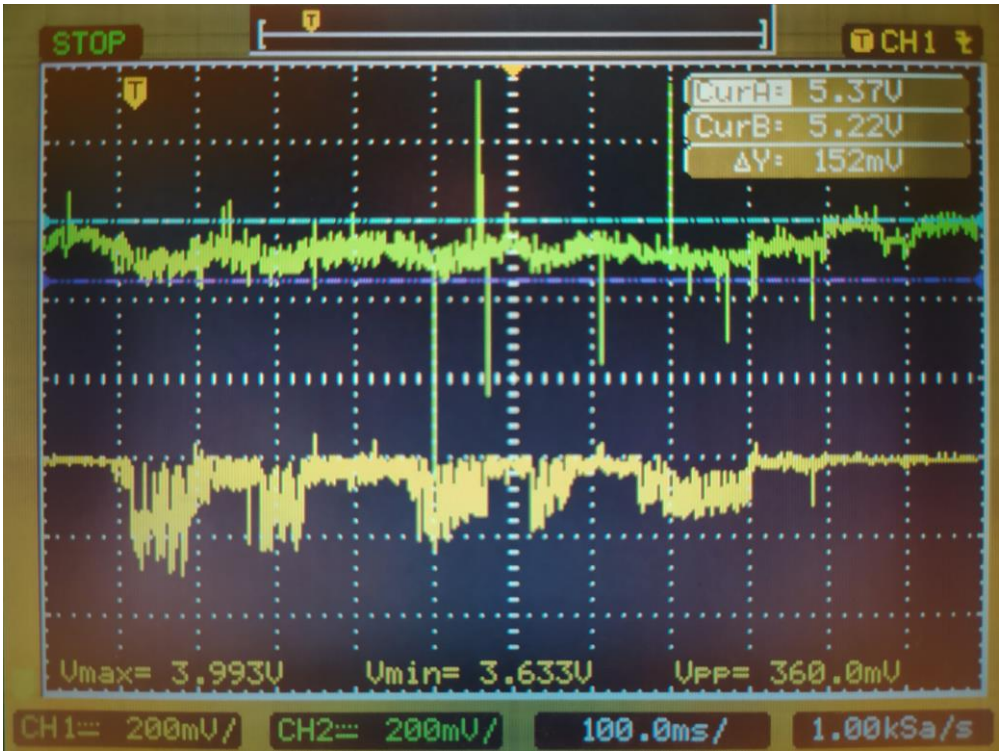
Na desce se ještě vyskytuje vícero konfiguračních hřebínků, převážně pro určování napětí nebo pro opravu zaměněných linek RX a TX u sériové sběrnice. Při návrhu byl vytvořen pouze obvod pro reset modulu, jeho spouštění se ale prování na jiném jeho vstupu. Na DPS se tak musí udělat externí propojka od volného pinu z Morpho konektoru na BOOT pin.

Tímto zásahem bohužel úpravy modulů nekončí. Během zprovozňování modulu se objevila jeho velká nestabilita. Při svém spouštění měl modul od DFRobot neustálé tendence se resetovat. Z odebíraného proudu na laboratorním zdroji bylo patrné, že se modul vypne v okamžiku největšího proudového nárazu. Ten probíhal při přihlašování modulu do sítě operátora, s odejmutou SIM kartou se tento problém totiž nevyskytoval. Protože poslední kondenzátor na 5V větvi se nachází až u zdroje, několik modulů pod tímto, byl zde zespod desky připájen ještě 1000µF elektrolytický kondenzátor. Na SIM modul to mělo však jen nepatrné zlepšení, z osmi restartů se pouze jednou podařilo přihlásit k operátorovi.



Obrázek 2.8: Měření napájecích napětí modulu DFRobot před opravou

Na diagnostiku byl proto použit osciloskop. Jedním kanálem byl připojený na vstupní napětí 5 V (horní zelený průběh) a druhým kanálem na napětí 3,8 V za vnitřním DC/DC měničem na modulu od DFRobot (spodní žlutý průběh). Z průběhu vnitřního napájecího napětí, viz Obrázek 2.8, je patrné, že měnič nedokáže dostatečně reagovat na proudové nárazy od SIM modulu. Jeho napájecí napětí se propadne až na hodnotu V_{min} = 2,58 V která už je pro modul neúnosná, dojde proto k jeho vypnutí a odlehčení zátěže, tím se měnič opět dostane na správnou provozní hodnotu V_{max} = 3,96 V. Navrhnutým řešením je zvýšení kapacity vyrovnávacího kondenzátoru. Po testovacím přiložení dalšího kondenzátoru paralelně ke stávajícímu jsou okamžitě viditelné pozitivní výsledky, modul se pokaždé nastartoval a přihlásil k mobilnímu operátorovi. Původní kondenzátor s jmenovitou kapacitou 470 µF na modulu od DFRobot tak byl nahrazen větším, 1 000µF kondenzátorem. Při pohledu na osciloskop, viz Obrázek 2.9 jsou stále patrné drobné propady napětí, s minimální hodnotou V_{min} = 3,63 V se ale stále jedná o akceptovatelné napětí.



Obrázek 2.9: Měření napájecích napětí modulu DFRobot po opravě

2.5.3 Programování

Fyzická komunikace s modulem je vcelku jednoduchá, pro vysílání dat po sériové sběrnici slouží funkce `HAL_UART_Transmit()`. Protože není známá přesná doba nebo délka přijímaných dat, používá se příjem po jednom znaku za pomocí přerušení, jak znázorňuje Zdrojový kód 2.5.

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
   if(huart == &huart1){
      if(SIM.rx_length < SIM.rx_max_length) {
         SIM.rx_message[SIM.rx_length] = SIM.rx_byte;
         SIM.rx_length = SIM.rx_length + 1;
      }
      HAL_UART_Receive_IT(&huart1, &SIM.rx_byte, 1);
   }
}
```

Zdrojový kód 2.5: Příjem po UART sběrnici

Před samotným programováním AT příkazů jsou z modulu úplně odstraněny propojky určující křížení datových linek. Na modul poté lze připojit externí převodník z UART na USB s čipem FTDI. V terminálovém prostředí Hercules je tak možné zkoušet jednotlivé příkazy, jejich syntaxi a formát zpětné odpovědi.

Jako první jsou vytvořeny funkce pro odeslání konfiguračních příkazů `AT` (nastavení formy komunikace) a `ATE0` (vypnutí zpětného zasílání přijímaných znaků). Během spouštěcí procedury se nejdříve příkazem `AT` zkontroluje, jestli už není modul zapnutý, v případě resetu mikrokontroléru totiž nedochází k vypnutí napájení a může se tak běžně stát, že modul SIM7600 zůstane zapnutý. Pokud je vykonání příkazu neúspěšné, proběhne puls na BOOT pinu a po krátké prodlevě se opět zkontroluje funkčnost `AT` příkazu.

S úspěšně inicializovaným modulem lze vytvářet a testovat další funkce.

Na obsluhu GPS postačí tři příkazy. Zapnutí GPS, kontrolu stavu a samotný požadavek na informace o aktuální poloze. Obecná struktura funkcí pro vykonání AT příkazů je vždy stejná. Příkladem je funkce ve Zdrojový kód 2.6.

```
int SIMCOM::AT_CGPS_Q() {
   rx_length = 0;
   if (writeUART ((uint8_t*)"AT+CGPS?\r", 9) < 0) {
      return -1;   //chyba zapisu
   }
   receiveTime = HAL_GetTick();
   while (HAL_GetTick() - receiveTime < 200) {
      if (rx_length > 10) {
         if (rx_message[rx_length-11] == '1') {
            return 2;     // GPS zapnuta
         } else if (rx_message[rx_length-11] == '0') {
            return 1;     // GPS vypnuta
         }
      }
   }
   return -2;
}
```

Zdrojový kód 2.6: Funkce pro kontrolu stavu GPS

Zde je v první části funkce zaslán příkaz po sběrnici a poté se čeká až do konce předdefinované doby. V této době je neustále kontrolována délka přijímaných dat. Pokud jsou splněny požadavky na délku odpovědi, zkontroluje se její obsah. Když odpovídá předpokládanému výstupu, funkce vrátí kladnou hodnotu, reprezentující, že vše proběhlo v pořádku. Selhání procedury v kterémkoliv bodě vyhodnocování vrátí z funkce zápornou hodnotu, na kterou může nadřazený program dále reagovat.

Přístup ke vzdálenému brokeru vyžaduje několik úkonů. Nejprve se musí v SIM7600 zapnout služba MQTT příkazem `AT+CMQTTSTART`. Následuje přiřazení klienta a až poté se lze připojit pomocí `AT+CMQTTCONNECT=0,"tcp://address:port",ka_time,1,"user","password"`. Zaslání poslední vůle není nutné.

Pokud je jednotka připojená k brokeru, mohou se data přetransformovat do pomocných číselných proměnných. Pro poskládání celé zprávy je využita funkce `sprintf()`, viz Zdrojový kód 2.7. Ta do jednoho bufferu, realizovaného `char` polem, uloží směs běžných znaků a naformátovaných proměnných.

```
char slovo[] = {´5´,´0´,´.´,´2´, ´\0´};
uint16_t cislo = 4268;
double plovouci = 20.785;
sprintf(buffer, “ 1 , 2 , 3 = %s , %d , %lf“, slovo, číslo, plovouci);
HAL_UART_Transmit(&huart2, (uint8_t*)buffer, strlen(buffer), 100);
_____________________________________________________________________
> 1 , 2 , 3 = 50.2 , 4268 , 20.785
```

Zdrojový kód 2.7: Ukázka transformace proměnných na text

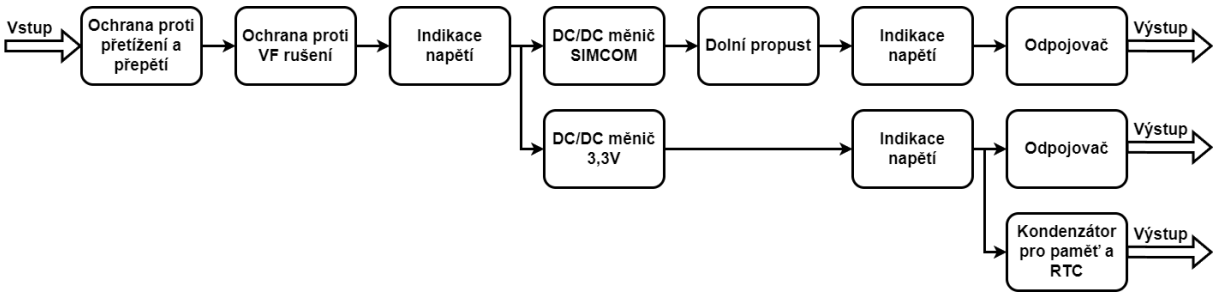
Data jsou po tomto procesu uložena jako znaky ASCII tabulky a ztrácejí tak parametry proměnné. Po odeslání na broker jsou ale podle předem uložených pravidel opětovně roztříděna a uložena do databáze jako fyzická čísla. Ještě před odesláním zprávy se určí její téma (ang. topic), aby mohla být zpráva předána do databáze.

Vývoj se samozřejmě neobešel bez problémů. Funkce pro odesílání dat s krátkými, ručně psanými informacemi pracovaly správně. Když však přišlo na odesílání textových řetězců

s reálnými daty, docházelo k náhodnému selhávání jiných částí programu. Nejčastěji se program zastavoval při pokusu o navázání komunikace po SPI sběrnicí. Kvůli zdánlivě nesouvisejícímu problému bylo věnováno značné množství času hledání chyby v jiných knihovnách. Až díky zpětnému pohledu do jedné ze záloh programu byla objevena chyba při deklaraci datového bufferu. Ten byl pro testovací účely deklarován podstatně menší, než kolik se do něj ukládalo informací. Pokud by se taková závada vyskytla v běžném programu spouštěném v nadřazeném operačním systému, byla by detekována jako neoprávněná manipulace s nepřiřazeným prostorem a její operace by se zamítnula. V případě vestavné aplikace žádný nadřazený systém není a jediné místo, kde by se závada mohla detekovat a včasně odstranit je kompilátor ve vývojovém prostředí. To se bohužel nestalo a tak funkce `sprintf()` a `for()` přepisovaly i části paměti, kde se pravděpodobně nacházela konfigurační data nebo pomocné proměnné HAL knihovny pro SPI sběrnici. Během snahy o komunikaci se tato zkorumpovaná data chybně přečetla a došlo k zablokování chodu mikrokontroléru. Jednoduchá oprava spočívala ve zvětšení pomocného bufferu.

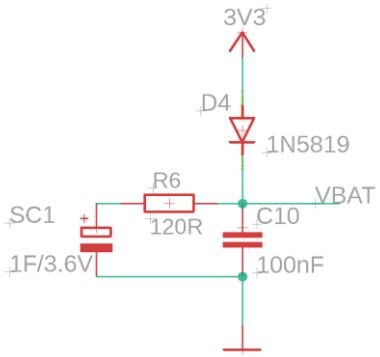
2.6 Napájení

Na elektrickém motocyklu se nachází kromě akumulátorového napětí, které se pohybuje okolo 100 VDC, ještě standardní rozvod 12 VDC pro napájení jednotlivých jednotek. Cílem napájecího modulu je připojit se na toto nižší napětí, které dále přemění na požadované napětí jednotlivých modulů. Pro obsluhu všech modulů je zapotřebí vytvořit dvě napětí. Prvním je standardních 3,3 V pro mikrokontrolér, paměť a senzory. Druhé napětí je lehce vyšší, přesněji 3,8 V nahrazující napětí lithiového článku pro čip SIM7600. Ještě před návrhem elektrického schéma je vytvořeno schéma blokové, viz Obrázek 2.10, které dále slouží jako vodítko při realizaci jednotlivých částí.



Obrázek 2.10: Blokové schéma napájecího modulu

Aby se zabránilo úplné destrukci napájecího modulu a zároveň i ochránil externí zdroj, je ihned za vstupním konektorem zařazená vratná pojistka, tzv. polyfuse. Jedná se v podstatě o PTC rezistor, čím vyšší proud jím protéká, tím více se zahřívá a roste i jeho odpor. Se vzrůstajícím odporem při stálém vnějším napětí musí dle pravidel Ohmova zákona klesat protékající proud. Po odstranění zkratu nebo přetížení (vlivem klesajícího napětí přestanou pracovat polovodičové součástky) pojistka zchladne, zmenší se její odpor a obvod je opět funkční. Tento typ pojistky je sice povětšinou o dost pomalejší než běžná tavná pojistka, na druhou stranu má značnou výhodu ve své resetovatelnosti bez nutnosti ručního zásahu nebo dokonce výměny komponenty. Vybraná obousměrná zenerova dioda s provozním napětím 12 V chrání obvod před napěťovými špičkami. Pokud by se na vstupu do modulu objevilo vyšší, než je deklarované průrazné napětí, dioda vytvoří zkrat a tím toto napětí potlačí. Následuje odrušení vstupního napětí za pomoci feritové perličky a jednoduchého LC článku typu L. Napájecí napětí 3,3 V je běžným standardem a může tak být použitý snižující DC/DC měnič s pevným výstupem. Na něj je ještě připojen superkondenzátor jako záložní zdroj napětí pro obvod RTC.



Obrázek 2.11: Elektrické schéma záložního zdroje pro RTC

Dioda ve schématu na Obrázek 2.11 zabraňuje vybíjení superkondenzátoru jako primárního zdroje při vypnutí přívodního napájení a 120Ω rezistor slouží jako omezovač nabíjecího

proudu. Pro správnou funkci vstupu VBAT na vývojovém kitu Nucleo je potřeba ještě odstranit nulový rezistor SB45, který spojuje napájecí napětí a tento vstup.

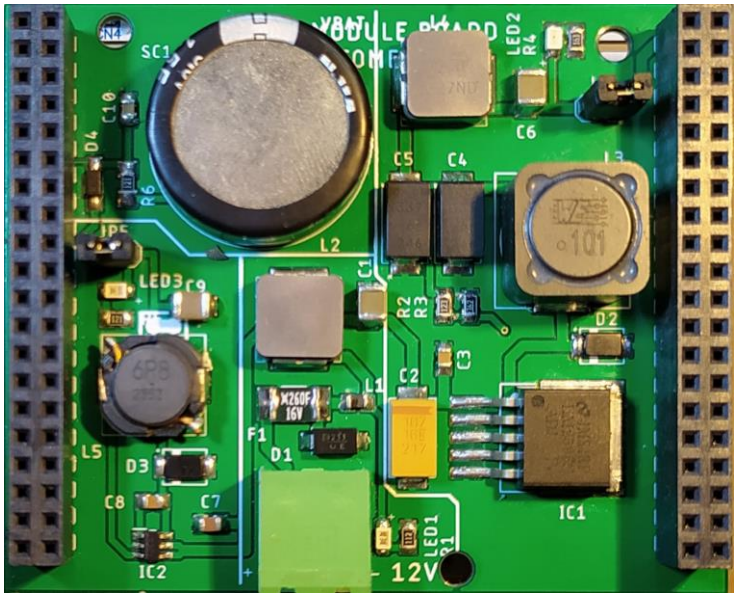
Trochu nestandardní napájecí napětí 3,8 V pro SIM7600 je realizováno pomocí nastavitelného DC/DC měniče LM2596. Při obdržení vývojového modulu od DFRobot bylo zjištěno, že už obsahuje vlastní snižující DC/DC měnič. Hodnota výstupního napětí pro tento modul tak musela být zvýšena alespoň na 5 V.

Hodnota výstupního napětí LM2596 se vypočte ze vzorce (2.6.1)

$$V_{OUT} = V_{REF}\left(1 + \frac{R_2}{R_1}\right) \quad (2.6.1)$$

kde V_{REF} je definováno jako 1,23 V. Z různých zakoupených odporů byla nejvhodnější kombinace odporů 3,6 kΩ a 12 kΩ, díky kterým bylo nastavené výstupní napětí 5,33 V.

Na desce plošných spojů viz Obrázek 2.12, jsou jednotlivé části zdroje vizuálně odděleny bílou nepřerušovanou čárou. V horní části desky si lze povšimnout špatně připravených výrobních dat, kdy se potisk (angl. silk) s označením desky objevil na přední straně desky a je tak částečně zakrytý okolními součástkami.



Obrázek 2.12: Osazená deska napájecího modulu

2.7 Wi-Fi/Bluetooth

Jako datová brána pro komunikaci na blízkou vzdálenost je použit modul ESP32-C3 s integrovanou anténou od společnosti Espressif Systems. Elektrické schéma je opět převzato z doporučení výrobce [19]. Pro komunikaci s mikrokontrolérem má využívat UART sběrnici.

Obrázek 2.13: Poškození ESP32 modulu

Modul ESP32 má vespod pájecí plochu pro lepší odvod tepla do dalšího DPS a tak byl osazován na pájecím stole. Bohužel se modul natolik ohřál, že se roztekla i pájka držící stínící kryt. Při následných korekcích pro estetické uložení, byl kryt nedopatřením posunut a odstranil několik součástek nacházejících se pod ním. Na Obrázek 2.13 vpravo nahoře jsou vidět drobné součástky uvnitř krytu. Jejich opětovné připájení by bylo velmi náročné a kvůli stejným rozměrům pouzder by se některé součástky i mohly zaměnit. Proto byl objednán modul nový. V době objednání nového modulu z internetového obchodu Mouser.com, který sídlí v USA, však probíhalo omezení obchodu s Čínou a hlavně bylo uvaleno embargo pro obchodování s Ruskem, zapříčiněné vpádem jeho vojsk na území Ukrajiny. Prodej každé součástky, která byla označena jako potenciálně využitelná pro zbrojní průmysl, byl tak přísně regulován. Objednaný modul ESP32-C3-WROOM-02-H4 se na tomto seznamu nacházel také kvůli své zvýšené provozní teplotě. Bylo nutné přesně určit aplikaci, lokalitu provozu, finální specifikace a spoustu dalších informací ohledně záměru použití. V dokumentu „Potvrzení odběratele o koncovém uživateli“ byla exportním úřadem USA vyžadována i přesná slovní deklarace, že modul nebude nasazen pro armádní využití. I přes vyplnění těchto formulářů stále nemohl být modul, který je mimochodem od čínského výrobce a v Číně vyrobený, vydán do zahraničí. Řešením bylo použití identického modulu s koncovkou 02-N4, který už je certifikován pro použití pouze do 85 °C a nevztahují se na něj žádné regulace o exportu.

2.8 Hlavní program

Spojení všech modulů probíhá právě v hlavní části programu. Po startu mikrokontroléru probíhá inicializace jednotlivých sběrnic a také všech modulů, na sériovou linku jsou vypisovány zprávy o jednotlivých výsledcích. V nekonečné smyčce programu probíhá kontrola přijmu zprávy po CAN sběrnici, viz kapitola 2.3.3 a také záznam dat ze snímače pohybu. V aktuální verzi jsou vždy vybírány ty největší hodnoty. Za pomoci funkce `HAL_GetTick()` jsou vytvořeny dvě obslužné procedury. První je vykonávaná každých 100 ms a pokud nalezne v externí paměti uložená a tudíž neodeslaná data, pokusí se je jednotlivě poslat přes internetovou bránu. Pokud se operace podaří, daná zpráva se z paměti smaže. Proceduru popisuje Zdrojový kód 2.8.

```
currentTime = HAL_GetTick();
if (MEM.topAddress - MEM.bottomAddress > 2 && currentTime - memloopDuration > 100)
{
   if (MEM.loadPacket(MC, MEM.bottomAddress, sizeMC) > 0) {
      HAL_Delay(5);
      if (SIM.MCtoSend(MC, topic, sizetopic) == 1) {
         MEM.eraseBottomPacket();
      }
   } else {
      MEM.eraseBottomPacket();
   }
   memloopDuration = HAL_GetTick();
}
```

Zdrojový kód 2.8: Kontrola neodeslaných dat

V druhé proceduře, vykonávané jednou za cca 900 ms je vyčtena současná poloha GPS, aktuální čas a také jsou přeloženy hodnoty ze snímače pohybu do datového pole `MC[]`. Celé toto pole se poté zkusí poslat na vzdálený server a pokud se to nezdaří, je pole uloženo do jedné stránky v externí paměti.

Pro obsluhu RTC jsou napsány dva podprogramy. Jeden slouží pouze k jeho čtení, ten druhý zabezpečuje jeho kalibraci z přijaté časové značky od mobilního operátora. Aby bylo možné přijaté informace z datové proměnné `char` použít jako běžné číslo, je vytvořen jednoduchý podprogram na jeho převod, využívající jednoduchosti ASCII tabulky. V ní jsou číslice 1 až 9 umístěné za sebou ve vzestupném pořadí od kódové hodnoty 48_{10}. Pro určení správného čísla tak stačí hodnotu 48_{10} odečíst od zakódovaného údaje.

```
int RTCCalibrate(){
   char RecData[21];
   RTC_DateTypeDef Date = {0};
   RTC_TimeTypeDef Time = {0};
   if (SIM.AT_CCLK_Q(RecData) < 0) {
      return -1;
   }
   Date.Year = charTou8t(RecData[0])*10 + charTou8t(RecData[1]);
   Date.Month =charTou8t(RecData[3])*10 + charTou8t(RecData[4]);
   Date.Date = charTou8t(RecData[6])*10 + charTou8t(RecData[7]);
   if (HAL_RTC_SetDate(&hrtc, &Date, RTC_FORMAT_BIN) != HAL_OK){
      return -2;
   }
…
```

Zdrojový kód 2.9: Kalibrace vnitřního kalendáře obvodu RTC

V hlavní smyčce probíhá ještě reset počítadla nezávislého WatchDogu. Pokud vy se z nějakého důvodu, ať už kvůli zacyklení programu, nebo chybě ve firmwaru nepodařilo tuto operaci provést do 31 s, mikrokontrolér se automaticky restartuje.

Jednotlivá elektrická schémata společně s náhledem DPS se v Příloze A nacházejí ve formě obrázků. Pro plnohodnotné zobrazení EDA informací poslouží soubor dat z programu EAGLE přiložený na CD, viz Příloha B.IV.

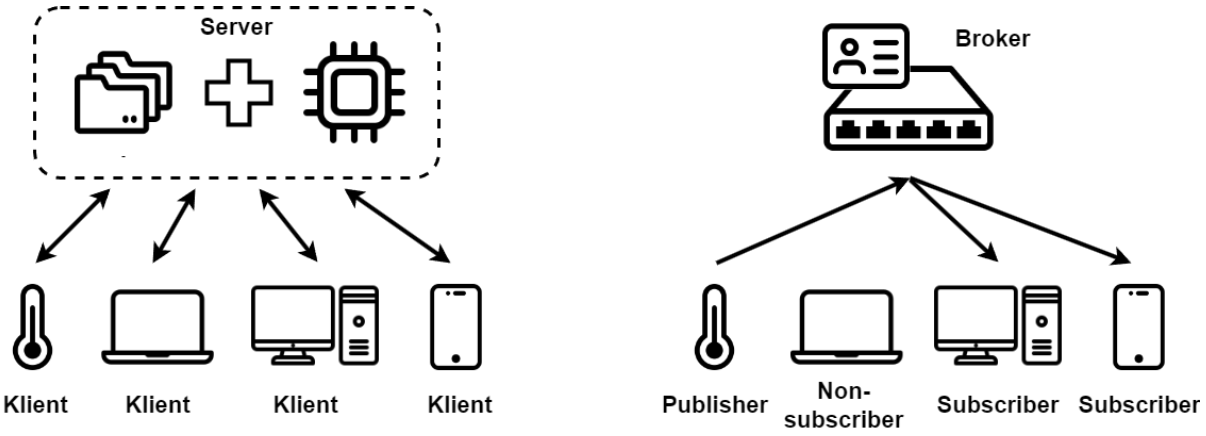
Všechny vytvořené funkce programu jsou k dispozici na CD jako samostatné soubory, viz Příloha B.I nebo jako součást projektu z prostředí STM32CubeIDE, viz Příloha B.III.

3 Vzdálený server a uložiště dat

3.1 Podoba serveru

U použitého protokolu MQTT se zde nevyskytuje server v tradičním slova smyslu. Běžný server tvoří hlavní bod sítě, ke kterému se jednotliví klienti dotazují a očekávají odpověď, jedná se o tzv. client-server model. Takový server má obvykle vlastní uložiště s databází a značným výpočetním výkonem. Pro získání aktuální informace na koncovém zařízení je potřeba na server přijmout informaci prvního klienta, uložit ji a na dotaz druhého klienta ji vyčíst a odeslat. Všichni klienti jsou si tak rovni a každý si vytváří vlastní komunikační kanál se serverem.

Naproti tomu MQTT používá pro centrální uzel alternativního termínu „broker“. Broker si udržuje pouze tabulku témat, angl. „topics“, která obsahuje seznam jejich příjemců. Subscriber, neboli příjemce se může kdykoliv přihlásit k odběru daného tématu. Když poté na brokera přijde zpráva od publishera (tedy odesílatele) s tímto tématem, je okamžitě rozeslána všem vypsaným příjemcům. Při analogii z topologie počítačové sítě se broker chová podobně jako hub, jen s tím rozdílem, že dokáže maskovat své výstupy. Nutno ještě zmínit, že jeden klient může být zároveň odesílatelem a také příjemcem na různých nebo i stejných tématech.



Obrázek 3.1: Rozdíl mezi typickým serverem a brokerem

Na jednom brokeru mohou být přihlášeni i tisíce příjemců a protože zde zcela odpadají jejich požadavky o informace, protokol MQTT dokáže být velmi efektivní a datově nenáročný.

MQTT broker může být provozován různými způsoby. Z pohledu prvotní konfigurace je nejjednodušším řešením cloudová služba. Poskytovatel v tomto případě zajišťuje veškerý hardware, sítovou infrastrukturu a také nabízí dodatečné možnosti konfigurace, obvykle přes webové rozhraní. V dnešní době rozmachu IoT a nástupu Industry 4.0 zle najít mnoho poskytovatelů. Nabídky často začínají jako bezplatné, jsou ale vždy omezené co do počtu připojených zařízení, objemu přenesených dat nebo doby připojení zařízení. Pro úplnější představu o současném trhu jsou níže uvedeny ceníky a podmínky vybraných portálů při měsíčním provozu.

EMQX [20]

Zdarma

- Do 1 GB nebo 1 milionu připojených minut (odpovídá 23 stále připojeným IoT zařízením)
- Zaručena 99.9% dostupnost (uptime SLA)
- sdílený cloud, bez autorizace klientského certifikátu

Pay as you go

- Při překročení limitů bezplatné služby umožněno připlacení za prodloužení limitů
- $2.00 za1 milion připojených minut a $0.15 za 1 GB
- Ostatní podmínky stejné jako u služby zdarma

Dedikovaný cloudový prostor – Tier 1

- $259 + $0.15 za 1 GB při překročení 100 GB
- Maximálně 1 000 současně připojených zařízení
- volba cloudového providera, vlastní doména, předpřipravené integrační řešení
- Zaručena 99.9% dostupnost (uptime SLA)

…

Dedikovaný cloudový prostor – Tier 4

- $1073 + $0.15 za 1 GB při překročení 100 GB
- Maximálně 10 000 současně připojených zařízení

HIVEMQ [21]

Zdarma

- 10 GB, 100 připojených klientů
- Bez podpory IPv6, bez autorizace klientského certifikátu,
- Nezaručená dostupnost brokeru (uptime SLA)

Dedikovaný cloudový prostor – Starter

- $245 + $0.80 za 1 milion zpráv
- Vlastní doména, umožňuje integraci Amazon Kinesis
- Zaručena 99,95% dostupnost (uptime SLA)

Amazon – implementace do AWS IoT Core [22]

Zdarma

- Do 500 000 zpráv nebo 2,25 milionu připojených minut (odpovídá 52 stále připojeným IoT zařízením)

Postupné navyšování

- $1.00 za1 milion připojených minut + $0.08 za 1 milion připojených minut
- Zaručena 99,95% dostupnost (uptime SLA)

Trochu složitějším řešením je vlastní hosting. Díky němu je zaručená úplná kontrola nad brokerem. To umožňuje větší konfiguraci a přizpůsobení. Takové řešení sebou nese ale i řadu nevýhod. Placené cloudové služby vždy zaručují vysokou míru spolehlivosti a dostupnosti brokera. U vlastního hostingu je náročnější takových standardů docílit a povětšinou to obnáší dodatečné finanční náklady. Pro zajištění větší spolehlivosti je potřeba mít brokera zálohovaného a také připojeného na UPS, která ho ochrání před krátkodobými výpadky elektrické sítě. Aby byl z internetu vždy přístupný, musí být u poskytovatele ještě zjednaná pevná veřejná IP adresa.

Tabulka 3.1: Ceny známých poskytovatelů za pevnou IPv4 adresu

O2	Vodafone	Starnet	Nej.cz
254,1 Kč	257,19 Kč	99 Kč	300 Kč

* uvedené ceny jsou za jeden měsíc provozu, informace jsou platné k 14.5.2023

Open-source softwarů pro služby MQTT se vyvíjí hned několik. Níže budou představeny 3 populární [23].

EMQX

Od roku 2012 je ve vývoji open-source broker psaný v jazyce Erlang. Tento programovací jazyk byl vytvořen a stále je udržován společností Ericsson, známou svou výrobou telekomunikačních technologií. Díky tomu je broker od EMQ určen pro vysoce stabilní a jednoduše rozšiřitelné systémy s podporou současného vykonávání vícero větví programu (tzv. multithreading). Jeho pokročilé funkce na druhou stranu vyžadují větší výpočetní výkon oproti dalším zde jmenovaným.

NanoMQ

Stejná společnost začala v roce 2020 vyvíjet i velmi odlehčeného MQTT brokera. Přestože si stále zachovává podporu multithreadingu, vývojáři slibují velmi malé hardwarové nároky a podporu napříč mnoha platformami.

Mosquitto

Nadace Eclipse Foundation zaštiťuje od roku 2009 vývoj nejpopulárnějšího open-source brokera. Pyšní se hlavně jednoduchým použitím a širokou uživatelskou základnou. Dá se nainstalovat na všechny potenciálně použitelné operační systémy jako je Windows, Mac nebo Linux. Bohužel nedokáže využívat multithreading a je spíše zaměřen pro jednodušší aplikace.

Právě velká uživatelská základna byla rozhodujícím faktorem při zvolení projektu Mosquitto do této práce.

3.2 Vlastní hosting a tvorba databáze

Na provoz brokera lze využít i běžný počítač. Takový provoz je ale neekonomický, protože osobní počítač má poměrně značnou spotřebu. Konkrétně testovaný počítač s jedním pevným diskem, low-end procesorem Ryzen páté generace a grafickou kartou RTX 4070 má s běžícím MQTT brokerem na operačním systému Windows 10 průměrnou spotřebu 82,6 W. Další zkoušenou možností je využití dedikovaného zařízení. V tomto případě je zvolen jednodeskový počítač Raspberry Pi třetí generace. (Když trochu přeskočíme, konečná spotřeba tohoto počítače je pouhých 2,42W.)

Během zprovozňování operačního systému Raspbian, aktuálně pojmenovaného Raspberry PI OS, se vyskytlo spoustu komplikací, které občasně vyžadovaly i kompletní reinstalaci operačního systému. Mezi první problémy patřilo vypadávající bezdrátové připojení. Částečným řešením je změna ovladače síťového rozhraní z dhcpcd na NetworkManager. Lze tak učinit v nástroji `raspi-config` spouštěném v terminálu. S novým ovladačem je systém stabilněji připojen přes Wi-Fi s pevně nastavenou IP adresou. Po vyřešení nestabilního připojení už je možné problémový video výstup HDMI nahradit spolehlivější službou vzdálené plochy – VNC. Dokonce i samotný systém se občas nenačetl. V tomto případě nepomohlo už ani použití nové a kvalitní SD karty nebo celého jiného počítače. Instalací starší verze operačního systému se ale tento problém zcela odstranil.

Na stabilní systém se pomocí standartního nástroje `apt` nainstaluje balíček mosquitto brokera a mosquitto klientů. Dále se vytvoří soubor s přihlašovacími údaji *pwdfile*. Odkaz na něj se zapíše do konfiguračního souboru pro MQTT brokera společně s příkazem na poslech určeného portu. Celá konfigurace je ve Zdrojový kód 3.1.

```
pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log
include_dir /etc/mosquitto/conf.d

listener 1883
allow_anonymous false
password_file /etc/mosquitto/pwdfile
```

Zdrojový kód 3.1: Obsah konfiguračního souboru MQTT brokera

Pomocí nástroje MQTTX lze ověřit funkci brokera v lokální síti. Po připojení na brokera a přihlášení k odběru vybraného tématu stačí už jen na dané téma odeslat nějakou zprávu. Pokud se nástroj připojí a odeslaná zpráva se zase vrátí, broker je plně funkční. Příkaz `sudo systemctl enable mosquitto` pak zajišťuje automatické spuštění brokera po startu systému.

Protože broker už ze své podstaty neobsahuje uložiště informací, je potřeba databázi umístit v dané topologii za jednoho z klientů. Broker i databáze mají běžet vždy ve stejný čas, oba nástroje se proto budou nacházet na jednom zařízení. Kvůli dřívější pozitivní zkušeností je použita databázová platforma InfluxDB. Po nainstalování je v terminálovém prostředí `influx` založena nová databáze „MotorkaTUL“ spolu s přístupovým jménem a heslem. K ukládání dat přijímaných na brokera je použit open-source agent Telegraf. Ten se přes svůj vstupní plugin připojí k MQTT brokeru jako jeden z klientů a přihlásí se k odběru žádaného tématu. Data z přijaté

CSV tabulky přeformátuje a uloží do databáze ve formě line protokolu na základě konfiguračního souboru, jehož část je vypsána ve Zdrojový kód 3.2.

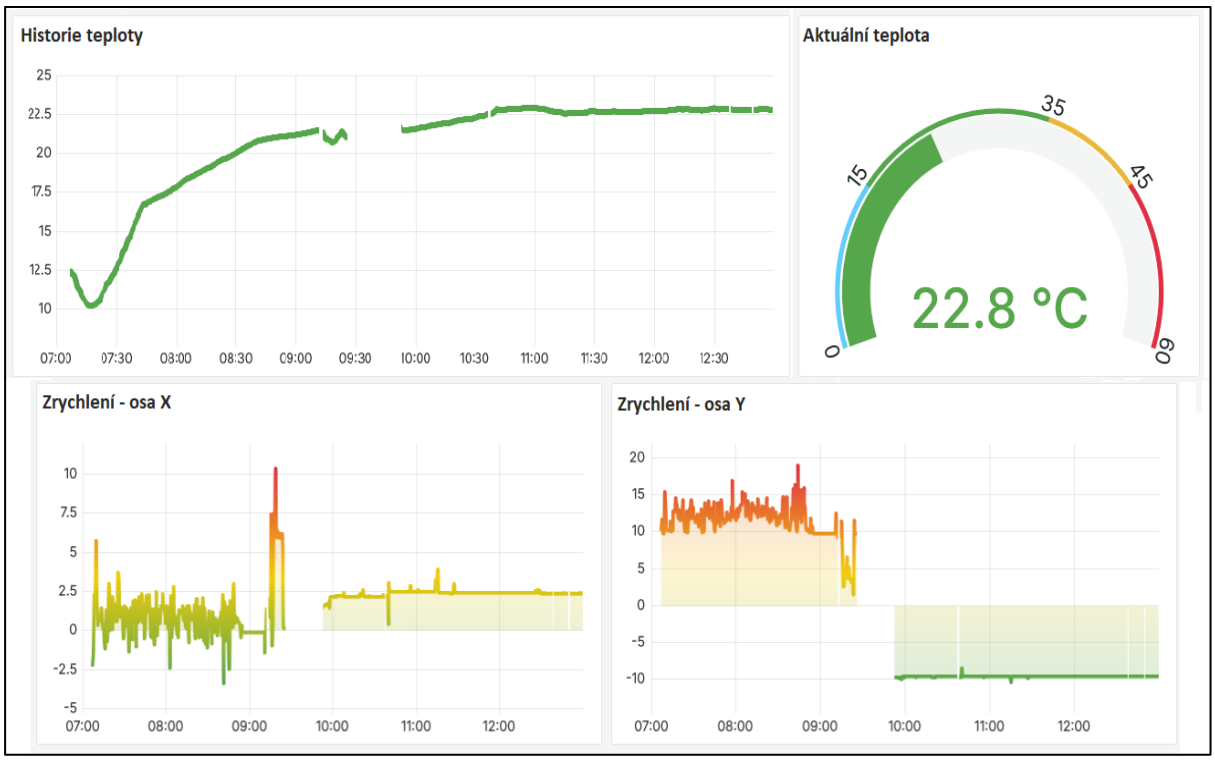
```
[[inputs.mqtt_consumer]]
   servers = ["tcp://localhost:1883"]
   topics = [
       "DATA"
   ]
   username = "Martin"
   password = "******"
   data_format = "csv"
   csv_header_row_count = 1
   csv_reset_mode = "always"
   csv_timestamp_column = "34"
   csv_timestamp_format = "2006-01-02T15:04:05"
   csv_timezone = "Europe/Prague"
…
[[outputs.influxdb]]
   urls = ["http://localhost:8086"]
   database = "Motorka"
   skip_database_creation = true
   username = "Martin"
   password = "******"
```

Zdrojový kód 3.2: Konfigurace agenta Telegraf

3.3 Vizualizace dat

Pro jednoduché zobrazení dat poslouží nástroj Grafana. Ten je také nainstalován na lokálním zařízení jako plně oddělený program od databáze. Ke spojení s databází dochází až vložením nového datového zdroje ve webovém rozhraní, které se dá otevřít v libovolném moderním prohlížeči. Tento nástroj umožňuje nativní přidání databází postavených na časových řadách jako je Graphite, Prometheus nebo použítá InfluxDB, stačí k tomu zadat jen adresu databáze společně s přihlašovacími údaji. Aby bylo možné se do webové aplikace dostat odkudkoliv, bylo na domácím serveru založeno pravidlo přesměrování portu 3000 z venkovní sítě na počítač Raspberry.

Ve webové aplikaci lze volně prohlížet veškeré údaje obsažené v databázi. Po určení zdroje dat, způsobu vyhodnocení a zadání parametrů časového rozmezí, které může být fixní nebo plovoucí, Grafana vytvoří žádanou vizualizaci. Portfolio vizualizací je poměrně velké, od časových plotů, přes budíky až po svíčkové grafy nebo histogramy a heatmapy. Na základě vybraných dat dokáže nástroj i doporučit vizualizace doplněné o různé stylizace. Jednotlivé vizualizace se mohou ještě skládat to tzv. dashboards, neboli palubních desek, které už mohou sloužit například jako plnohodnotné kontrolní středisko. Jedna vytvořená palubní deska, znázorněná na Obrázek 3.2 je dále využita při testu v silničním provozu.



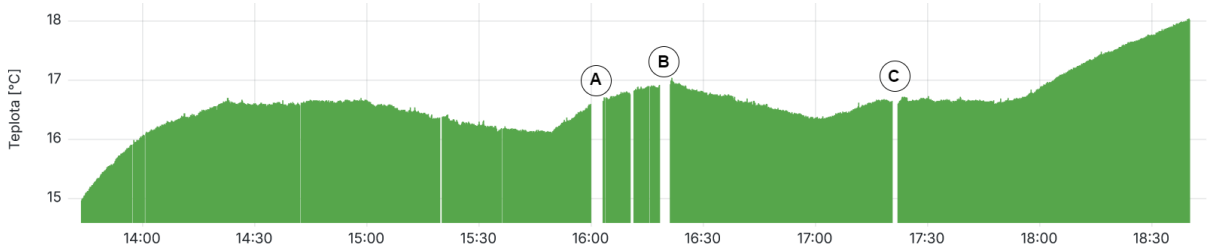
Obrázek 3.2: Ukázka palubní desky v nástroji Grafana

4 Finální testování

Po zprovoznění jednotlivých modulů, kompletaci hlavního programu a vytvoření stabilního serveru je na čase testování jednotky jako celku.

4.1 Test stability

Při prvním testu byla jednotka ponechána na pracovním stole spolu s připojenými debugovacími nástroji po dobu několika hodin. Perioda pro odesílání dat byla nastavena na 2 s. Cílem testu bylo ověřit základní stabilitu systému a spolehlivost odesílání dat. Pro názornost byl vytvořen Graf 4.1.



Graf 4.1: Uložená teplota při testu č. 1

Od spuštění jednotky ve 13:45 až do 16:00 bylo přijato přes 98,8 % dat. Ze záznamu zpráv v MQTT loggeru:

```
…,16.28,…,2024-04-24T15:36:17
…,16.22,…,2024-04-24T15:36:17
```

je patrné, že na broker proběhlo odeslání dvou rozdílných zpráv, avšak se stejnou časovou značkou. Proto byl ve vyobrazeném bodě „A“ nahrán do jednotky nový program se sledováním procesních hodnot vlastní funkce `RTCtoTimestamp()`. Dle očekávání funkce občasně ukládala do datového pole předešlou hodnotu času. Protože se do datového pole ukládají přímo hodnoty vyčtené z bloku RTC pomocí standartních funkcí z knihovny HAL, problém způsobují nejspíše právě ony. Okamžitým voláním dvou po sobě jdoucích funkcí pro přístup k RTC registru totiž dochází k jeho zahlcení a nestíhá se tak včas aktualizovat. Řešením je vložení krátké čekací smyčky procesoru ve formě `HAL_Delay()`. Opravený Zdrojový kód 4.1 je k nahlédnutí níže.

```
RTC_DateTypeDef Date = {0};
RTC_TimeTypeDef Time = {0};

HAL_RTC_GetDate(&hrtc, &Date, RTC_FORMAT_BIN);
HAL_Delay(5);
HAL_RTC_GetTime(&hrtc, &Time, RTC_FORMAT_BIN);

MC[163] = (uint8_t) Date.Year;
MC[164] = (uint8_t) Date.Month;
MC[165] = (uint8_t) Date.Date;
MC[166] = (uint8_t) Time.Hours;
MC[167] = (uint8_t) Time.Minutes;
MC[168] = (uint8_t) Time.Seconds;RTC_DateTypeDef Date = {0};
```

Zdrojový kód 4.1: Opravená funkce pro přečtení RTC

Do jednotky byl nahrán v čase zvýrazněném bodem „B“. Od této doby se již na server úspěšně přijímá rovných 100 % dat. V bodě „C“ jsou z programu vyjmuty procesní výstupy na terminál,

které by také mohly dobou svého provádění ovlivnit zpracování dat v jednotce. Také je snížena inicializační doba jednotlivých modulů pro rychlejší start jednotky.

4.2 Jízdní test

Další, již náročnější zkouškou je test spolehlivosti v silničním provozu. Prvotní záměr cílil na připojení jednotky k palubnímu napětí automobilu Škoda Octavia I. generace, následovaný zkušební jízdou. Při přípravě před jízdou byl napájecí modul ještě testován na laboratorním CC/CV zdroji, kde došlo ke zjištění vážné slabiny v jeho návrhu.

Protože starší automobily neobsahují regulátory palubního napětí, běžně tak jeho hodnota kopíruje nabíjecí napětí autobaterie a pohybuje se tedy v rozmezí 13,8-14,5 V [24]. Pokud by došlo k selhání regulátoru alternátoru, může se toto napětí vyšplhat i výše.

Jednotka má na svém vstupu však dříve zmíněnou oboustrannou zenerovu diodu, chránící jednotku před napěťovými špičkami. Bohužel dioda je natypována na 12 V a její průrazné napětí je dle katalogového listu 13,3-14,7 V [25]. Při zvýšení napájecího napětí nad 14,4 V na laboratorním zdroji se dioda prorazila a způsobila zkrat na vstupu napájecího modulu. Zdroj se díky nastavenému maximálnímu proudu 0,5 A přepnul na CC režim a ochránil tak diodu před úplným zničením.

Jelikož obě napájecí větve zdrojového modulu dokáží pracovat i s vyšším napětím, nejlepším řešením by byla výměna ochranné diody. Dodání jedné komponenty je však ekonomicky neúsporné a také by se musel daný test odložit. Dalším, značně provizorním, řešením by bylo odpájení problematické komponenty. Pro funkčnost v laboratorních podmínkách sice není tato komponenta nutností, při reálném provozu by ale mohly potenciální napěťové špičky v palubní síti automobilu způsobit nenávratné poškození jednotky.

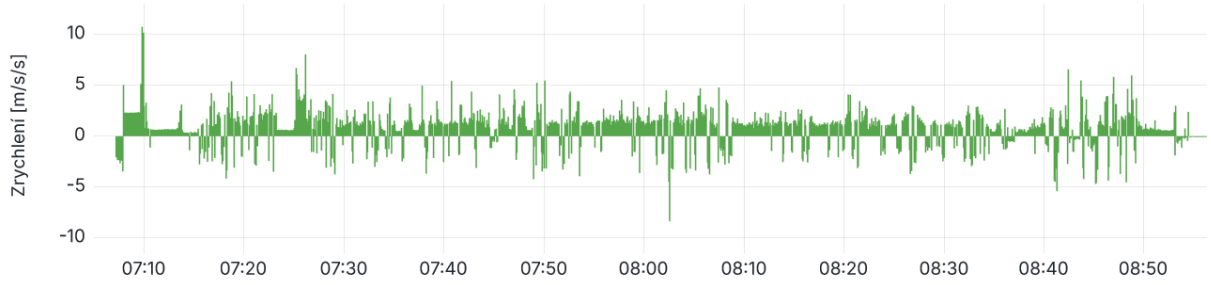
Jako výsledné řešení je zvolen plně separátní zdroj elektrické energie. Akumulátor z ručního nářadí je pro své jmenovité napětí 18,5V vhodným kandidátem. Lze na něj připojit laboratorní zdroj čínské provenience – Riden RD6006. Svou podstatou se jedná o step-down měnič s regulovaným napěťovým výstupem. Celou jednotku je tak možno uložit do přepravní krabice a za jejího provozu libovolně přenášet či vozit.

Obrázek 4.1: Pohled na jednotku před testem v silničním provozu

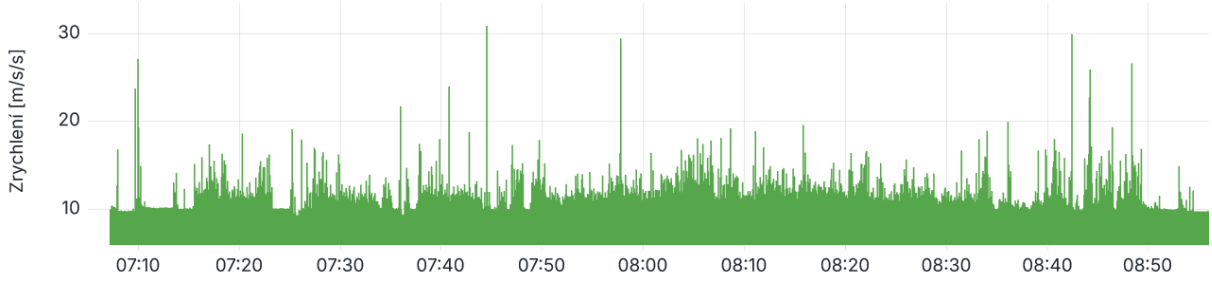
Naplánovaná trasa začíná z Dolní Poustevny (obec ve Šluknovském výběžku sousedící s Německem) v 7:05 a končí příjezdem k Technické univerzitě v Liberci v 8:55. Během cesty dlouhé cca 80 km se záměrně projíždí obcemi a úseky se známou nízkou dostupností mobilních sítí. Na Obrázek 4.2 je možné vidět oblasti na trase, které dle operátora T-Mobile neposkytují pokrytí vysokorychlostní mobilní sítí LTE.

Obrázek 4.2: Mapa pokrytí LTE v úsecích testovací jízdy

Nejobtížnějším úsekem je bezesporu oblast okolo kopce Velký Buk (736 m n. m.) a přilehlé obce Svor (435 m n. m.), kde z běžné praxe není dostatečné pokrytí ani sítí 2G. Ve Svoru se jednotka nacházela v čase 7:53. Z Graf 4.2 a Graf 4.3 lze říci, že byly do databáze uloženy provozní data i z této doby, kdy jednotka nemohla být připojena k internetu. Z toho lze usoudit, že spolehlivě funguje mimo jiné i paměťový modul a obsluha RTC. Všeobecně kladný trend u zrychlení ve směru jízdy je způsoben zanesením gravitačního zrychlení kvůli mírnému náklonu jednotky od rovnovážné polohy při uložení ve vozidle. Naopak u Graf 4.3 tvoří gravitační zrychlení dominantní složku a občasné špičky představují jen zhoupnutí vozu a rázy od podvozku.



Graf 4.2: Zrychlení jednotky ve směru jízdy při testu v silničním provozu



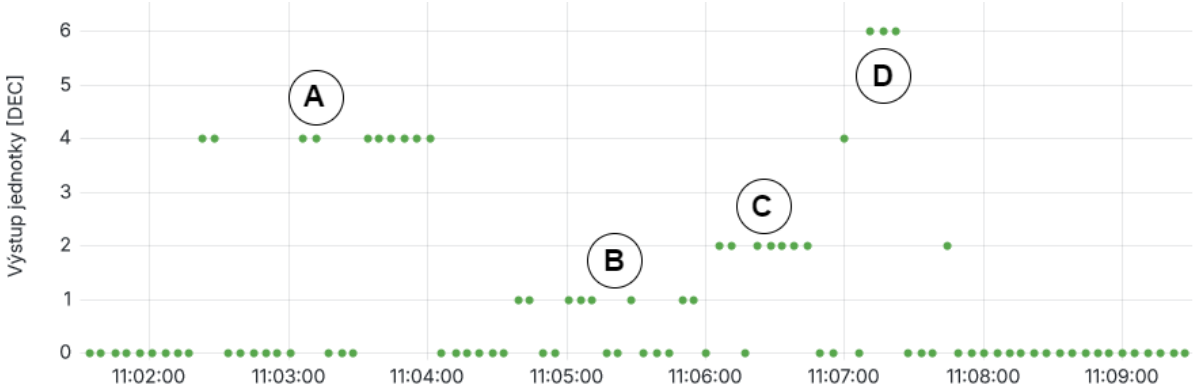
Graf 4.3: Zrychlení jednotky směrem k zemi při testu v silničním provozu

Velkým zklamáním je při tomto testu modul GPS, který během cesty i s občasnými zastávkami nedokázal ani jednou určit přesnou polohu jednotky. Při následném zjišťování problému byla objevena chyba v poslední aktualizaci kódu, kdy nesprávné vyhodnocení návratové proměnné

při inicializaci modulu umožnilo pouze tzv. „studený start“ GPS modulu. Pokud už byl modul předem zapnutý, ale ještě nebyl inicializován, program tuto skutečnost nereflektoval a GPS považoval za plně připravené. Ve finálním kódu v přiloženém DVD je již tato chyba opravena.

4.3 Připojení k motocyklu

Posledním testem funkčnosti je připojení přímo k elektrickému motocyklu. Z praktických důvodů si jednotka ponechává separátní napájení z akumulátoru a zapojeny jsou jen dva vodiče datové sběrnice CAN. Ještě před připojením je potřeba rozhodnout o terminačním rezistoru na sběrnici. Ve stávajícím zapojení elektrického motocyklu se již dva rezistory o jmenovité hodnotě 120Ω vyskytují. Připojením dalšího by se o ⅓ zvedlo zatížení budičů při vysílání, což není žádoucí. Z modulu jsou proto úplně odstraněny propojky pro konfiguraci terminačního rezistoru a je více dbáno na co nejkratší připojovací vodiče k uzlu páteřní sběrnice pro zamezení odrazů na vedení. Kvůli modulárnímu provedení jednotky by nebylo bezpečné s motocyklem jezdit, všechna měření jsou tak prováděna na zastaveném stroji. Po správném určení linek CAN H, CAN L a rekonfiguraci prescaleru pro určenou datovou rychlost 500 kbit/s už lze na MQTT brokeru a v databázi číst první data.



Graf 4.4: Data z jednotky Front ECU

Z matice odesílaných dat na CAN sběrnici, jejíž část je zobrazena v Tabulka 4.1, lze zpětně dekódovat události v jednotlivých úsecích Graf 4.4.

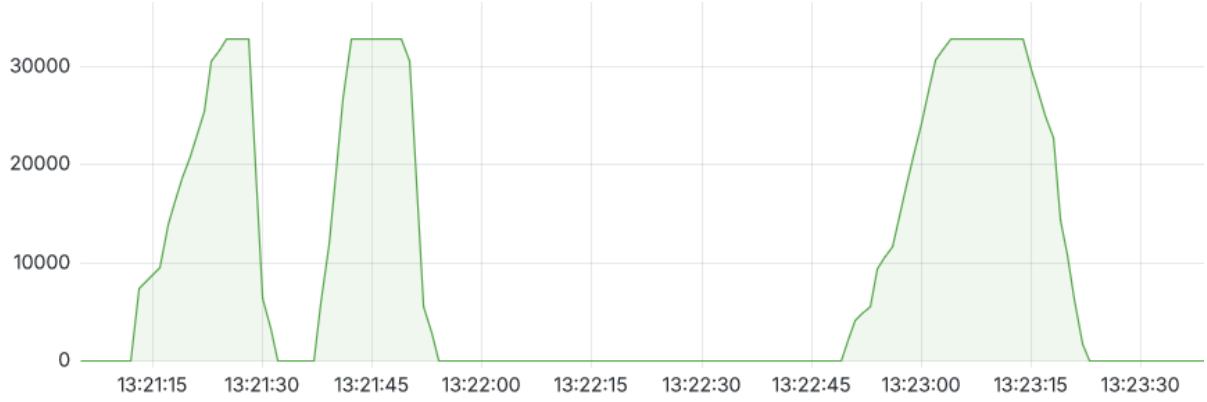
A: $4_{10} = 0100_2$ - aktivní brzda
B: $1_{10} = 0001_2$ - aktivní levý blinkr
C: $2_{10} = 0010_2$ - aktivní pravý blinkr
D: $6_{10} = 0110_2$ - aktivní brzda + pravý blinkr

Tabulka 4.1: Kódování bitů u zprávy Front ECU

4	3	2	1	0
Sabvoton ready	Dálková světla	Brzda	Pravý blinkr	Levý blinkr

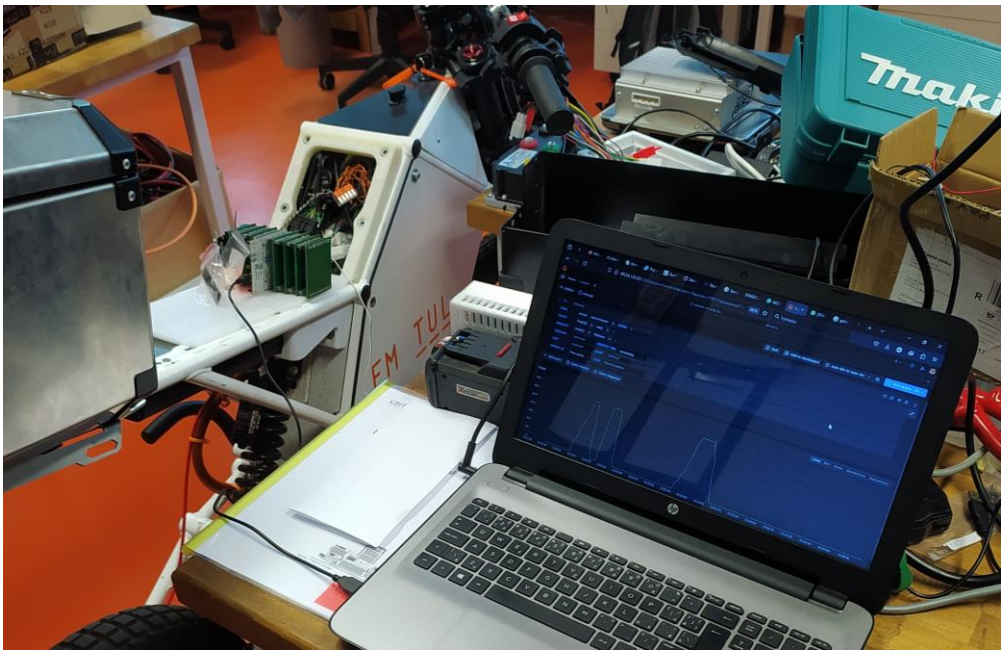
Během sepnutých blinkrů v bodě “B“ a “C“ je i přes relativně stabilní periodu blikání velmi nepravidelný záznam v databázi. To je způsobeno delší periodou odesílání zpráv (cca 5 s) společně s absencí vyhodnocování dat v komunikační jednotce. Na brokera jsou tak vždy odeslána jen nejnovější data a protože je střída rozsvíceného blinkru větší než 50%, častěji se v datech vyskytuje jeho aktivní stav. Pro další měření je proto alespoň snížena perioda odesílání dat na 1 s. Z jednotky „motor controller“ se po sběrnici CAN vysílá zpětná vazba hodnoty rukojeti

plynu. Jde o poslední dva byty (4 + 5), ze zprávy ID 401, která měla být do databáze ukládána pod číslem 22. Záznam o přidání plynu na Graf 4.5 se v databázi ale nachází pod číslem 20.



Graf 4.5: Zaznamenaná pozice rukojeti plynu

Tím se odhalila další chyba v kódu. Problém vzniká během skládání textové zprávy pro odesílání na brokera. Aby správně fungovala funkce `memcpy()` pro převod mezi datovými typy, je nutné ukládat byty z CAN sběrnice v sestupném pořadí. Do paměti se ale celé zprávy ukládají v opačném pořadí. Když se poté vyčítají jednotlivé údaje pro sestavení odesílací tabulky, měly by se také v opačném pořadí zapisovat. Během psaní knihovny pro SIM modul se na toto však zapomnělo. Jednoduchá oprava ve formě opačného směru `for` smyčky je ve zdrojovém kódu poznamenána, v nahrané verzi firmware se ale nevyskytuje kvůli dřívějšímu zápisu informací do daných polí databáze. Změnou zasílaných dat by došlo ke znehodnocení části dat ve smyslu dlouhodobých statistik.



Obrázek 4.3: Jednotka připojená k univerzitnímu motocyklu

Závěr

Během návrhu jednotlivých modulů došlo k několika chybám a problémům. Většinu z nich se však podařilo opravit buď za pomoci jednoduché rekonfigurace propojek zabudovaných hřebínků, nebo i dodatečným zásahem do desek plošných spojů. Nepříjemným zjištěním bylo, že i některé komerčně prodávané moduly obsahují vady ve svém konceptu a je také nutné je opravovat. Jednotlivé zákroky sice plní svou funkci, esteticky jsou ale značně nedokonalé. Jednotka je tak hardwarově plně funkční, pro konečné nasazení ale ještě postrádá určité prvky zvyšující její odolnost vůči vnějším vlivům.

Zvolená metodika programování do separátních knihoven zajistila konečnou funkčnost a do jisté míry i zlepšila přehlednost kódu. Nově vytvořené knihovny jsou solidním základem pro samostatné použití i v jiných projektech, nicméně jim v některých ohledech chybí dodatečná bezpečnost a kontrola. Funkce dokáží detekovat základní vzniklé problémy, hlavní program už ale není tak pružný, aby reagoval a vyřešil veškeré potenciálně vzniklé chyby. Teoreticky se tak může stát, že jediným řešením problému bude jen restart celé jednotky. Při tvrdém zaseknutí programu zde funguje alespoň nezávislý WatchDog, jeho užití však limituje časovou délku pro odesílání zpráv a v lokalitách se špatným pokrytím mobilního připojení vynucuje občasný zbytečný zápis do paměti pro pozdní odeslání. Protože byla současná jednotka vždy programována jako modulární řešení s krátkodobým provozem, kompletně postrádá funkce pro úsporu energie.

Komunikační jednotka ve většině ohledů splnila očekávaní v testech simulujících skutečný provoz. Mimo amatérské chyby s obsluhou GPS všechny funkce a podprogramy pracovaly stabilně a nezpůsobily žádné chyby nebo poruchy. Částečným zklamáním během testů je absence reálné jízdy na motocyklu, obzvlášť v době, kdy už má stroj homologaci pro provoz na pozemních komunikacích. Testy také odhalily nevyužitý potenciál jednotky během odesílání dat. Aktuálně nejnižší možná perioda odesílání jedné sekundy, je pro některé zaznamenávané veličiny příliš dlouhá a odesílaná data tak ztrácí na využitelnosti. Pokud by se jednotka ocitla mimo dosah mobilní sítě, s takto nastavenou periodou ukládání dat by ale byla schopna zaznamenávat informace až po dobu 18 hodin.

Vlastní hosting MQTT brokera společně s databází se ukázal jako velmi funkční řešení, které dodává celému projektu flexibilitu, v současné době je ale stále posazeno na dynamické IP adrese a občasně nespolehlivém jednodeskovém počítači. Pro budoucí použití by bylo proto vhodné tento systém migrovat na stabilnější platformu ať už ve formě lokálního serveru s pevným přístupem k internetu, nebo i do plně cloudového prostředí.

Použitá literatura

[1] History of the Tachograph | Total Compliance. *https://totalcompliance.co.uk/* [online]. [vid. 2024-05-12]. Dostupné z: https://totalcompliance.co.uk/history-of-the-tachograph/

[2] *Drivers' hours and tachographs: goods vehicles - 4. Tachograph rules - Guidance - GOV.UK* [online]. [vid. 2024-05-12]. Dostupné z: https://www.gov.uk/guidance/drivers-hours-goods-vehicles/4-tachograph-rules

[3] Products - CAN Bus Data Loggers & Telematics. *CSS Electronics* [online]. [vid. 2024-05-13]. Dostupné z: https://www.csselectronics.com/pages/can-bus-hardware-products

[4] A.S, Alza. Mobilly OBD-II BT - Diagnostika | Alza.cz. *Alza* [online]. [vid. 2024-05-13]. Dostupné z: https://www.alza.cz/mobilly-obd-ii-bt-d4624328.htm

[5] *eCall in-vehicle system — type-approval | EUR-Lex* [online]. [vid. 2024-05-12]. Dostupné z: https://eur-lex.europa.eu/EN/legal-content/summary/ecall-in-vehicle-system-type-approval.html

[6] Telit eCall Solution. *Telit* [online]. [vid. 2024-05-06]. Dostupné z: https://www.iot.com.tr/uploads/pdf/Telit_eCall_Solution_Application_Note_r6.pdf

[7] *Volvo On Call – the first SOS system with pan-European coverage* [online]. [vid. 2024-05-11]. Dostupné z: https://www.media.volvocars.com/global/en-gb/media/pressreleases/4929/

[8] Avanci 4G Vehicle. *Avanci* [online]. [vid. 2024-05-12]. Dostupné z: https://www.avanci.com/vehicle/4gvehicle/

[9] u-blox eCall Whitepaper. *u-blox* [online]. [vid. 2024-05-11]. Dostupné z: https://content.u-blox.com/sites/default/files/products/documents/u-blox-eCall_WhitePaper_(MNS-X-11005).pdf

[10] CLARK, Adam. Switching off 2G and 3G in the UK [online]. 2024 [vid. 2024-05-12]. Dostupné z: https://commonslibrary.parliament.uk/research-briefings/cbp-9959/

[11] PUBLISHED, Leon Poultney. Nissan drops app support for the original Leaf – are EVs now as disposable as iPhones? *TechRadar* [online]. 5. březen 2024 [vid. 2024-05-12]. Dostupné z: https://www.techradar.com/vehicle-tech/hybrid-electric-vehicles/nissan-drops-app-support-for-the-original-leaf-are-evs-now-as-disposable-as-iphones

[12] ŠINDELÁŘ, Jan. ELORYKS aneb konec riskantních honiček. Vědci ukázali, jak policie zastaví auto jedním klikem. *Zdopravy.cz* [online]. 15. únor 2024 [vid. 2024-05-12]. Dostupné z: https://zdopravy.cz/eloryks-aneb-konec-riskantnich-honicek-vedci-ukazali-jak-policie-zastavi-auto-jednim-klikem-194360/

[13] Datasheet of STM32L476xx microcontrollers. *STMicroelectronics* [online]. [vid. 2024-04-25]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32l476rg.pdf

[14] Datasheet of IAM-20680 6-axis inertia measurement unit. *TDK InvenSense* [online]. [vid. 2024-04-25]. Dostupné z: https://product.tdk.com/system/files/dam/doc/product/sensor/mortion-inertial/imu/data_sheet/ds-000196-iam-20680-v1.1-typ.pdf

[15] WEE, Inhwan. *finani/ICM20689* [online]. C++. 14. leden 2024 [vid. 2024-05-06]. Dostupné z: https://github.com/finani/ICM20689

[16] RICHIEQIANLE. Forum - CAN bus signal integrity. *StackExchange Electronics* [online]. [vid. 2024-05-07]. Dostupné z: https://electronics.stackexchange.com/questions/312179/can-bus-signal-integrity

[17] Datasheet of W25N01GVxxxG/T/R series memory. *Winbond* [online]. [vid. 2024-05-01]. Dostupné z: https://www.winbond.com/resource-files/W25N01GV%20Rev%20R%20070323.pdf

[18] *WinbondW25N/src/WinbondW25N.cpp at master · squaresausage/WinbondW25N · GitHub* [online]. [vid. 2024-05-11]. Dostupné z: https://github.com/squaresausage/WinbondW25N/blob/master/src/WinbondW25N.cpp

[19] ESP32-C3-WROOM-02 Datasheet. *Espressif* [online]. [vid. 2024-05-06]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32-c3-wroom-02_datasheet_en.pdf

[20] *EMQX Platform Console* [online]. [vid. 2024-04-24]. Dostupné z: https://cloud-intl.emqx.com/console/deployments/new

[21] *HiveMQ Pricing – Self-Managed MQTT Platform & Full-Managed MQTT Platform* [online]. [vid. 2024-04-25]. Dostupné z: https://www.hivemq.com/pricing/

[22] Securely Connect IoT Devices – AWS IoT Core Pricing – Amazon Web Services. *Amazon Web Services, Inc.* [online]. [vid. 2024-04-25]. Dostupné z: https://aws.amazon.com/iot-core/pricing/

[23] JOEY. Top 3 Open Source MQTT Brokers for Industrial IoT in 2023. *www.emqx.com* [online]. [vid. 2024-05-04]. Dostupné z: https://www.emqx.com/en/blog/top-3-open-source-mqtt-brokers-for-industrial-iot-in-2023

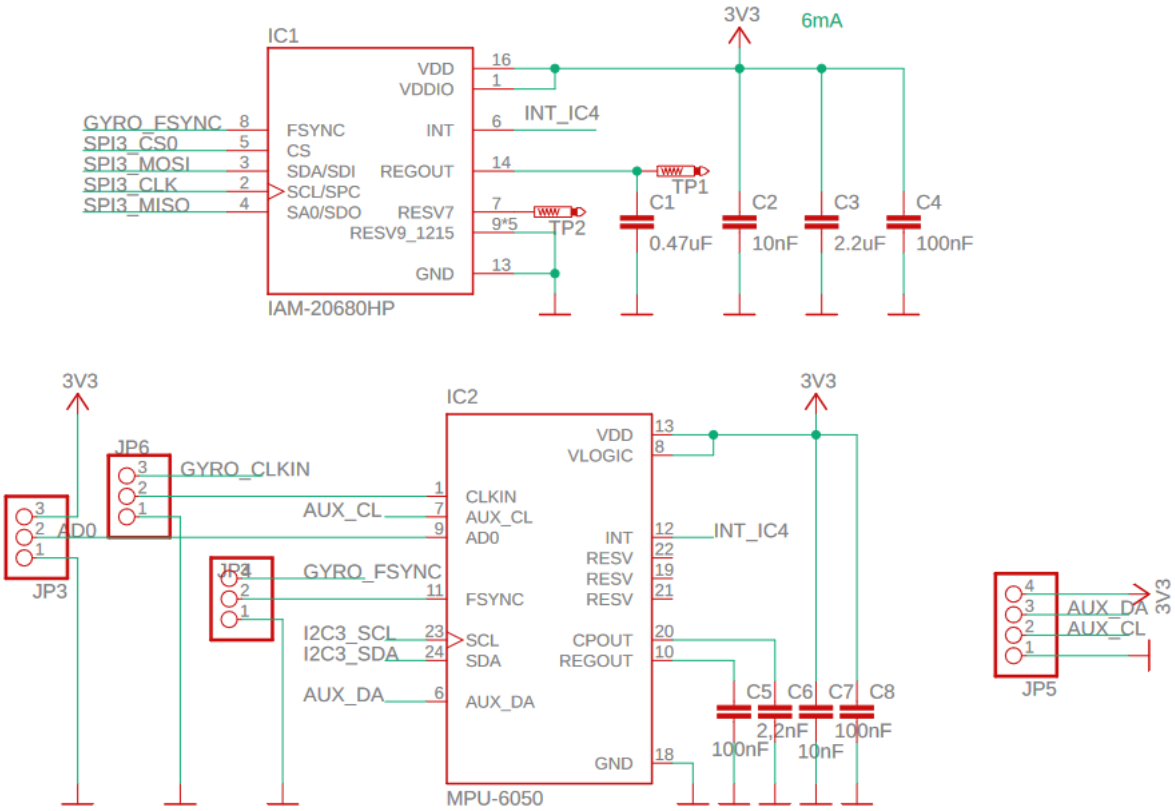
[24] *How to charge all lead acid batteries; how to charge SLA lead acid batteries, a tutorial for engineers about lead acid chargers and charging.* [online]. [vid. 2024-05-13]. Dostupné z: https://www.powerstream.com/SLA.htm

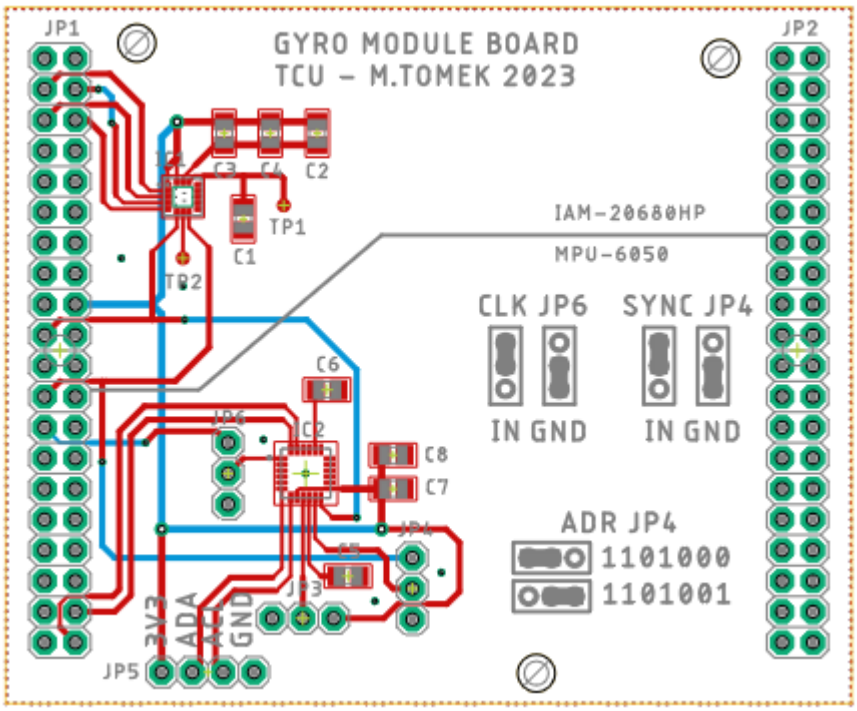
[25] SMAJ Transient Voltage Suppressor Diode Series. *BOURNS* [online]. [vid. 2024-04-25]. Dostupné z: https://www.bourns.com/docs/product-datasheets/smaj.pdf

Přílohy

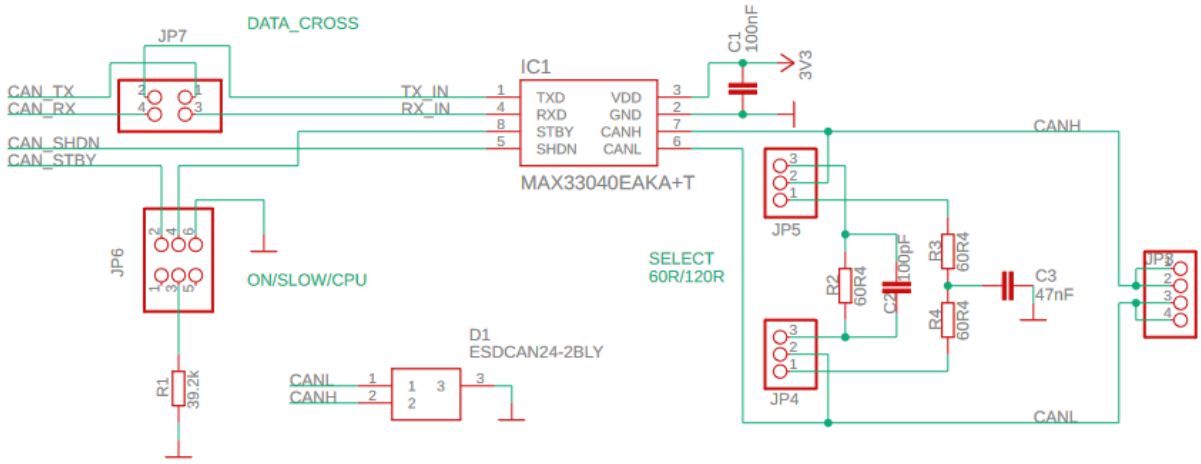
A Elektrické schéma a DPS jednotlivých modulů

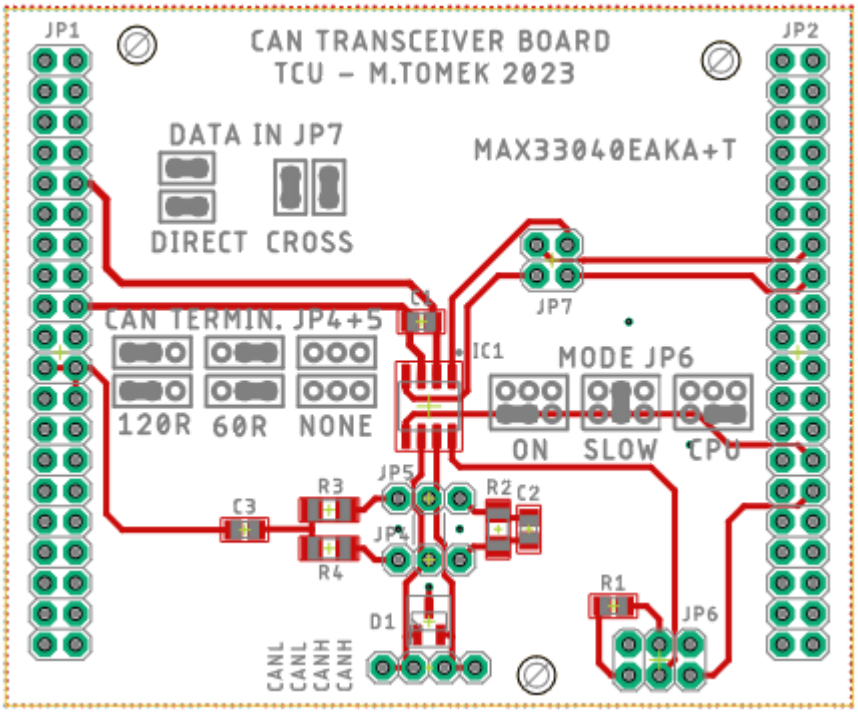
A.1 Snímač pohybu



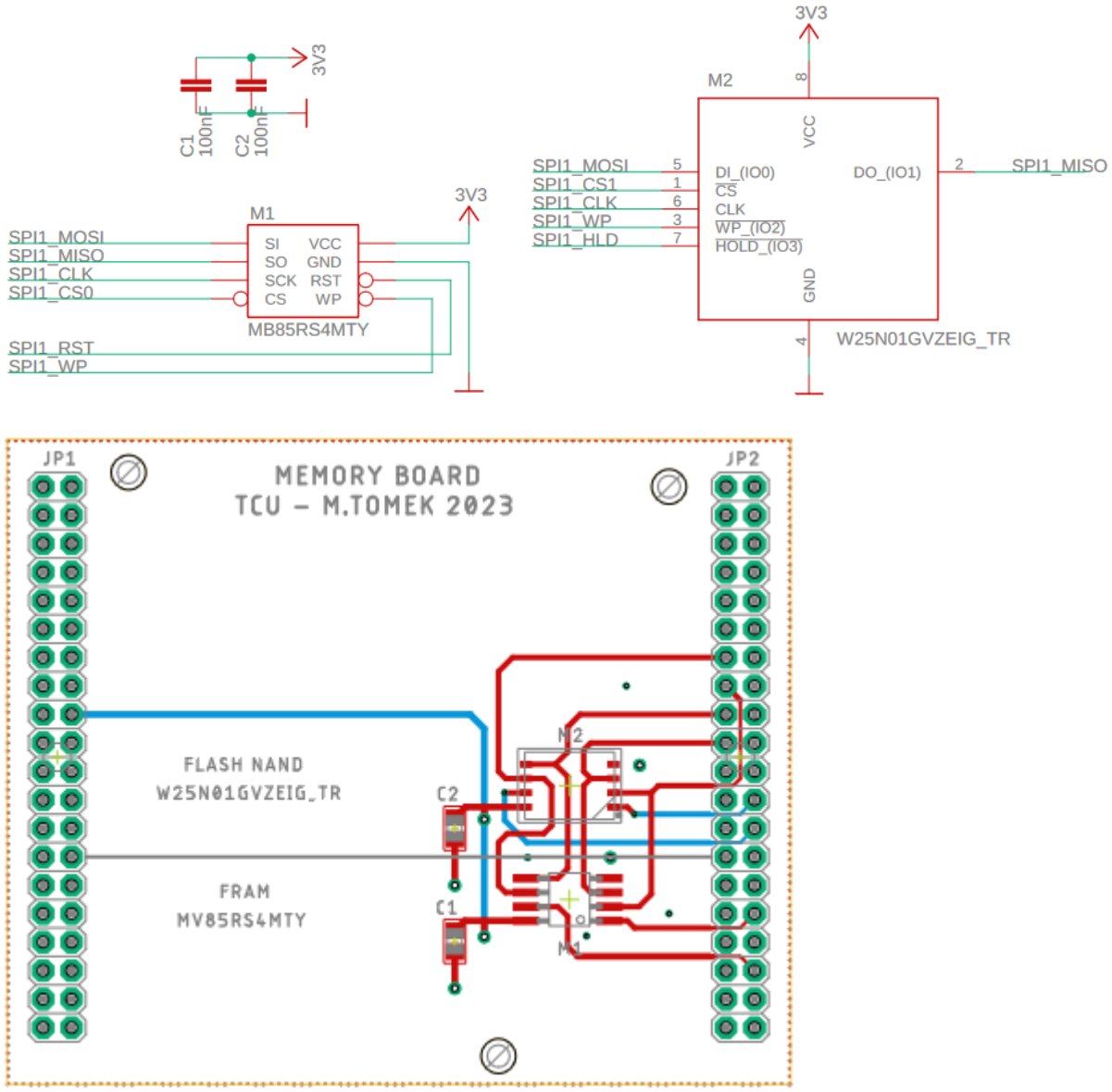


A.2 CAN

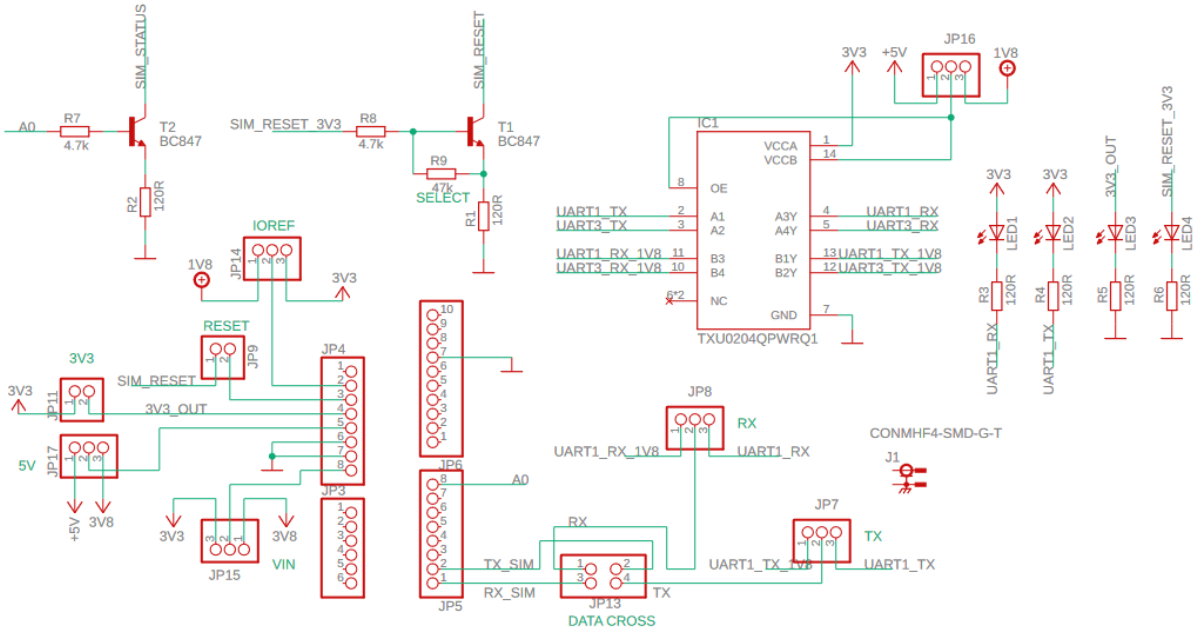

JP7
DATA_CROSS
CAN_TX
CAN_RX
CAN_SHDN
CAN_STBY
TX_IN
RX_IN
IC1
TXD
RXD
STBY
SHDN
VDD
GND
CANH
CANL
MAX33040EAKA+T
C1
100nF
3V3
CANH
JP5
SELECT
60R/120R
R2
60R4
C2
100pF
R3
60R4
R4
60R4
C3
47nF
JP3
JP4
CANL
JP6
ON/SLOW/CPU
R1
39.2k
D1
ESDCAN24-2BLY
CANL
CANH

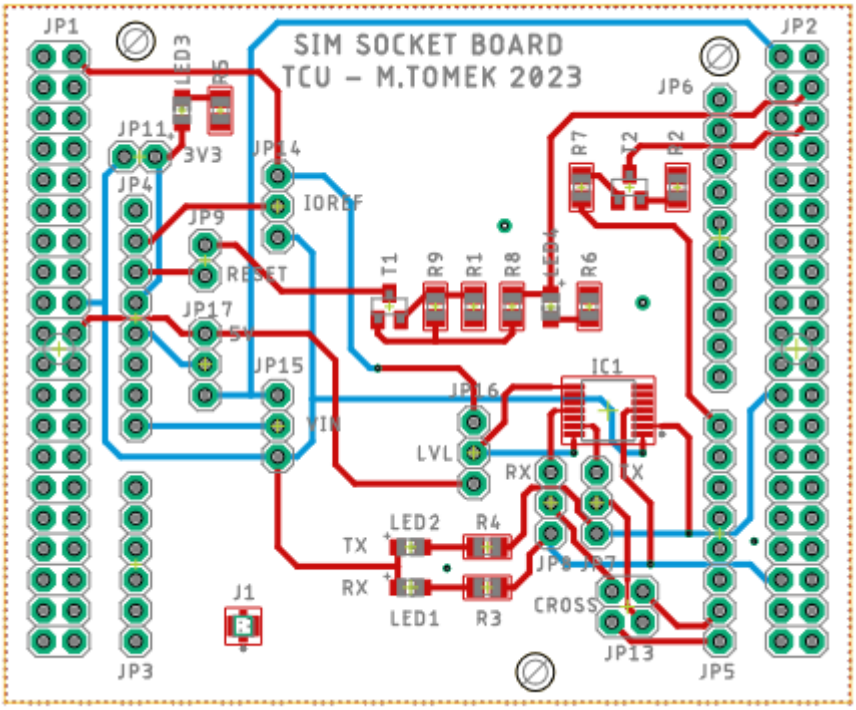

JP1
JP2
CAN TRANSCEIVER BOARD
TCU – M.TOMEK 2023
DATA IN JP7
DIRECT CROSS
MAX33040EAKA+T
JP7
CAN TERMIN. JP4+5
120R 60R NONE
IC1
MODE JP6
ON SLOW CPU
JP5
R3
R2
C2
C3
JP4
R4
R1
JP6
D1
CANL CANL CANH CANH

A.3 Paměť

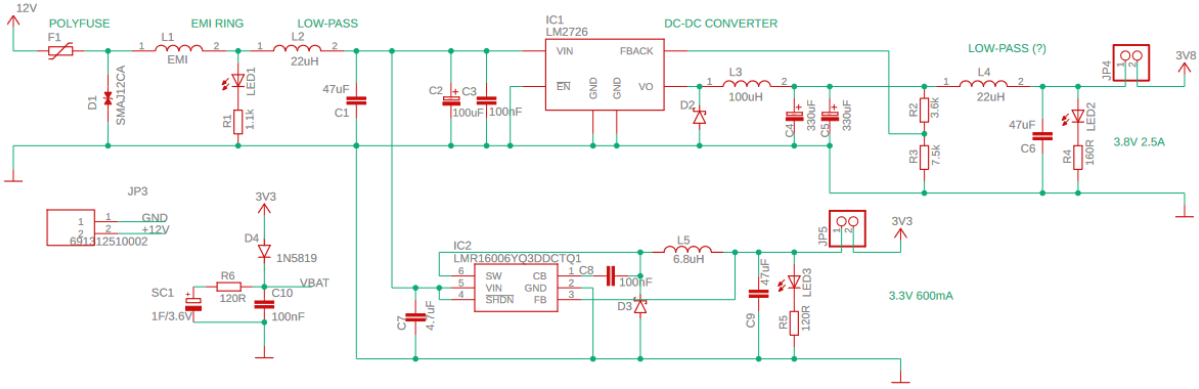

3V3
C1
100nF
C2
100nF
M1
SPI1_MOSI
SPI1_MISO
SPI1_CLK
SPI1_CS0
SI
SO
SCK
CS
VCC
GND
RST
WP
MB85RS4MTY
SPI1_RST
SPI1_WP
M2
VCC
SPI1_MOSI
SPI1_CS1
SPI1_CLK
SPI1_WP
SPI1_HLD
DI_(IO0)
CS
CLK
WP_(IO2)
HOLD_(IO3)
DO_(IO1)
SPI1_MISO
GND
W25N01GVZEIG_TR
JP1
JP2
MEMORY BOARD
TCU – M.TOMEK 2023
FLASH NAND
W25N01GVZEIG_TR
FRAM
MV85RS4MTY
C2
C1
M2
M1

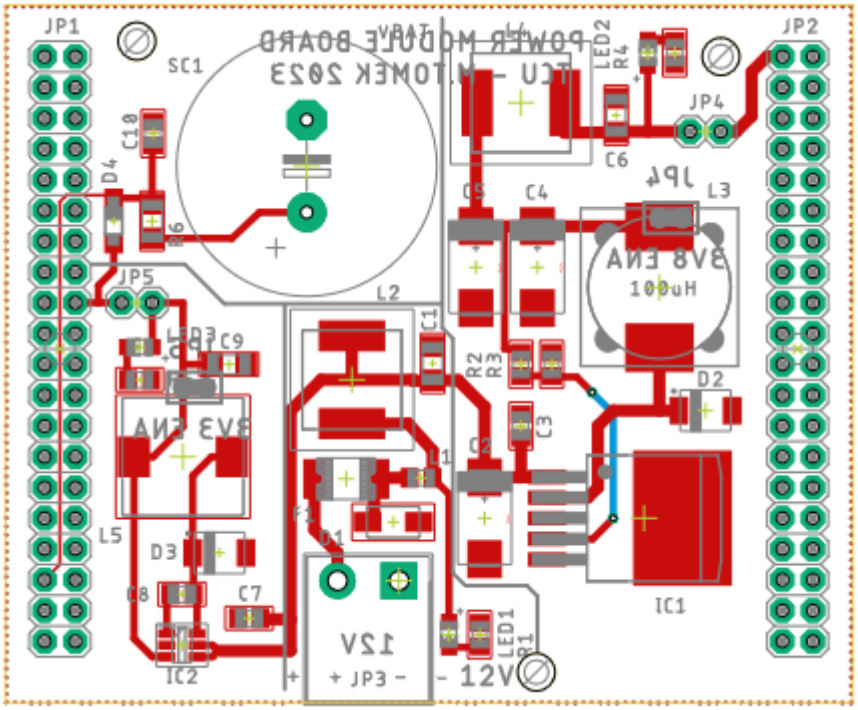
A.4 Internetová brána a GPS



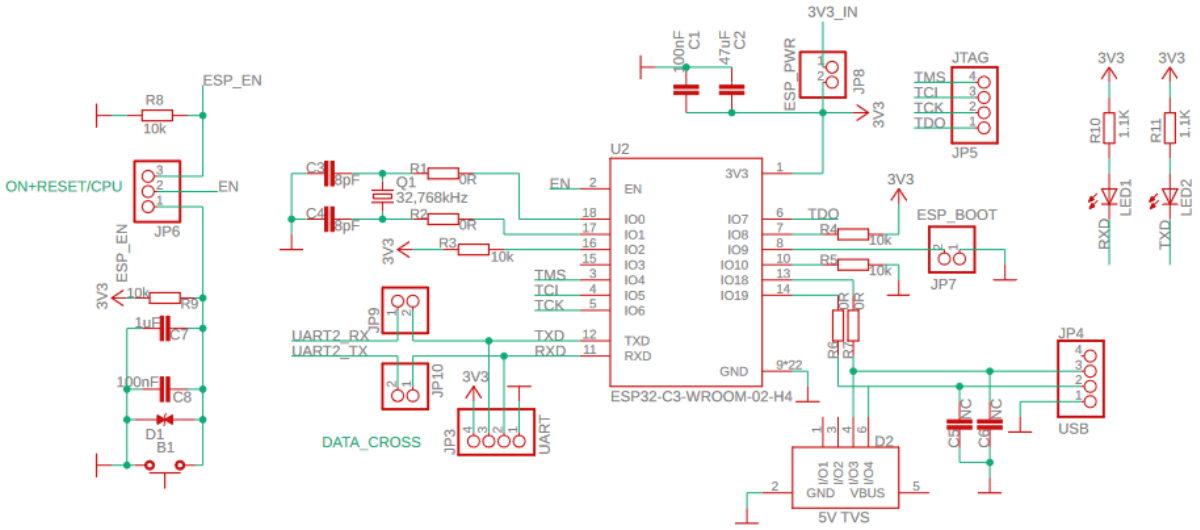


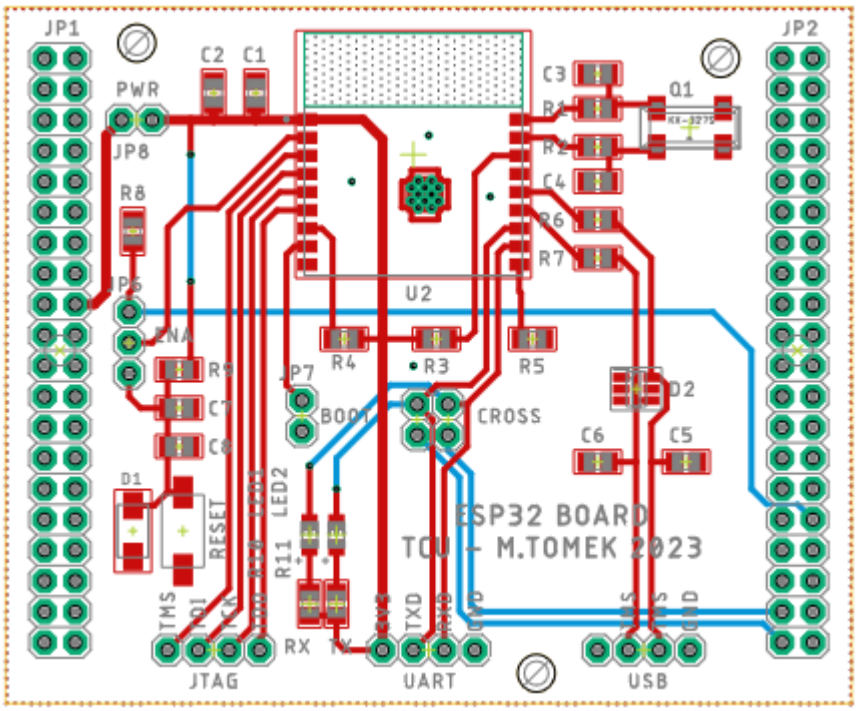
A.5 Napájení


12V
POLYFUSE
F1
EMI RING
L1
EMI
LOW-PASS
L2
22uH
D1
SMAJ12CA
LED1
R1
1.1k
47uF
C1
IC1
LM2726
DC-DC CONVERTER
VIN
FBACK
EN
GND
GND
VO
C2
C3
100uF
100nF
D2
L3
100uH
C4
330uF
C5
330uF
LOW-PASS (?)
L4
22uH
R2
3.6k
R3
7.5k
47uF
C6
R4
160R
LED2
JP4
3V8
3.8V 2.5A
JP3
GND
+12V
3V3
D4
1N5819
SC1
1F/3.6V
R6
120R
VBAT
C10
100nF
IC2
LMR16006YQ3DDCTQ1
SW
CB
VIN
GND
SHDN
FB
C8
100nF
D3
L5
6.8uH
C7
4.7uF
C9
47uF
R5
120R
LED3
JP5
3V3
3.3V 600mA

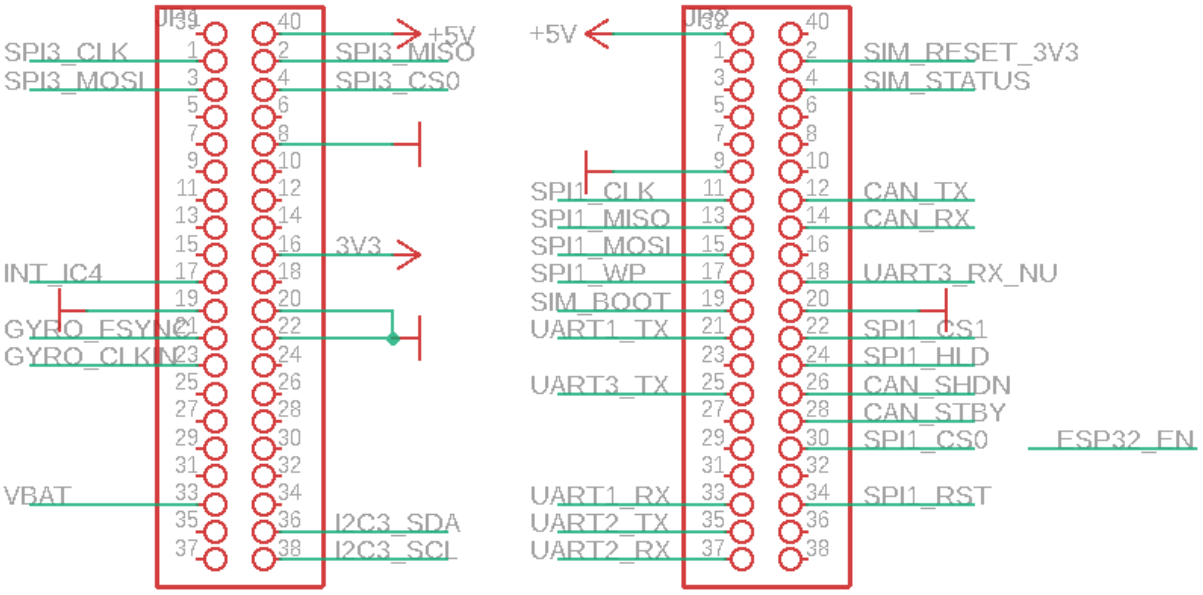

JP1
JP2
SC1
C10
D4
R6
JP5
C9
L2
C1
C5
C4
L3
JP4
C6
LED2
R4
D2
C2
C3
R2
R3
L5
D3
C8
C7
IC2
IC1
LED1
R1
12V
JP3
12V

A.6 Wi-Fi/Bluetooth





A.7 Obsazení upraveného Morpho konektoru



B Připojené CD

Obsah:

I. Vytvořené funkce a knihovní soubory
   a. Hlavní program
      i. main.cpp
      ii. main.h
   b. Snímač pohybu
      i. IAM20680.cpp
      ii. IAM20680.h
   c. Paměť
      i. W25N.cpp
      ii. W25N.h
   d. Internetová brána a GPS
      i. SIMCOM.cpp
      ii. SIMCOM.h

II. Tabulkové soubory
   a. Pametove_pole_MC.xlsx
   b. Kodovaci_tabulka_odesilanych_dat.xlsx
   c. Matice_dat_CAN.xlsx

III. Projekt_STM32.zip – Zakomprimovaný projekt z prostředí STM32CubeIDE

IV. Projekt_EAGLE.zip – Zakomprimovaný projekt z prostření EAGLE pro návrh elektrických schémat a DPS