Česká zemědělská univerzita v Praze

Technická fakulta

Katedra využití strojů

Modernizace telemetrie palubní jednotky

Formule Student

Bakalářská práce

Autor: **David Koudelka**

Vedoucí práce: Ing. Zdeněk Votruba, Ph.D.

2024

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Technická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

David Koudelka

Informační a řídicí technika v agropotravinářském komplexu

Název práce

Modernizace telemetrie palubní jednotky Formule Student

Název anglicky

Modernization of Formula Student on-board unit telemetry

Cíle práce

Cílem práce je posoudit stávající řešení přenosu telemetrických dat palubní jednotky monopostu studentské formule řešení přenosem ZigBee a navrhnou vhodnou modernizaci včetně návrhu na on-line zpracování získaných dat. Výstupem bude ověření praktického nasazení navrhované technologie.

Metodika

Na základě provedeného testování a měření analyzovat stávající přenosy telemetrie palubního počítače monopostu a statisticky vyhodnotit parametry přenosu. Navrhnout nové vhodné řešení založené na alternativní technologii a toto prakticky ověřit. Porovnat výsledky měření obou technologií a koncipovat návrh finálního řešení

Doporučený návrh osnovy:

1. Úvod
2. Cíl práce a metodika
3. Koncepční a technologický popis stávajícího řešení včetně provozních zkušeností
4. Definice problematických oblastí
5. Návrh na modernizaci
6. Popis nového technického řešení včetně provozních testů
7. Analýza výsledků měření
8. Finanční kalkulace
9. Závěr a doporučení

Doporučený rozsah práce

30 až 40 stran textu včetně obrázků, grafů a tabulek

Klíčová slova

ZigBee, GPS, snímače, 4G

Doporučené zdroje informací

BRADÁČ, Z. FIEDLER, P. KAČMÁŘ, M.: Bezdrátové komunikace v automatizační praxi VII Další standardy. Automa, 2004, roč. 10, č. 7, s. 44 46, ISSN 1210-9592.

firemní a zdrojová literatura

Očenášek, P., Trchalík, R.: Průmyslová bezdrátová síť ZigBee, Brno, 2008

Švéda, M., Trchalík, R.: ZigBee-to-Internet Interconnection Architectures, In: Proceedings of the Second International Workshop on Mobile Communications and Learning MCL 2007, Saint Luce, Martinique, MQ, IEEE CS, 2007, s. 6, ISBN 0-7695-2807-4

Předběžný termín obhajoby

2023/2024 LS – TF

Vedoucí práce

Ing. Zdeněk Votruba, Ph.D.

Garantující pracoviště

Katedra využití strojů

Elektronicky schváleno dne 10. 3. 2023

doc. Ing. Petr Šařec, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 3. 2023

doc. Ing. Jiří Mašek, Ph.D.

Děkan

V Praze dne 24. 10. 2023

Čestné prohlášení

„Prohlašuji, že svou bakalářskou práci Modernizace telemetrie palubní jednotky Formule Student jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.“

V Praze dne …… …….

(podpis autora práce)

Poděkování

V první řadě bych chtěl poděkovat svému vedoucímu práce Ing. Zdeňkovi Votrubovi, Ph.D. za možnost pracovat pod jeho vedením a za jeho trpělivost, kterou se mnou měl. Dále bych chtěl poděkovat studentskému týmu CULS Prague formula racing za možnost toto téma zpracovat a za jejich podporu a zázemí.

Abstrakt

V práci bylo analyzováno stávající řešení telemetrie týmu CULS Prague formula racing a bylo hledáno vhodné řešení pro modernizaci. Hlavními problémy bylo nespolehlivé ukládání dat a málo přesný GPS modul. Dále tým řešil přechod na hybrid a novou řídící jednotku, což přinášelo velké množství nových dat, a proto bylo potřeba zvýšit rychlost komunikace. Při hledání vhodných řešení byla vytvořena analýza možných technologií v oblasti živého zobrazení dat, ukládání dat, GPS modulů, zobrazování stavů a komunikačních protokolů. Na základě této analýzy bylo navržené kompletní řešení s použitím Arduina Portenta H7 Lite, mikro SD karty, dvoufrekvenčním GPS modulem, LTE routerem a ethernetovým kabelem pro připojení na Arduino Cloud a využitím dvou CAN bus sběrnic pro rychlou a spolehlivou komunikaci mezi jednotlivými jednotkami na monopostu. Tento návrh se v této práci realizoval kvůli nedostupnosti některých jednotek jen částečně. Na této částečné realizaci byla otestována funkčnost CAN bus sběrnic a ukládání dat na SD kartu.

Klíčová slova

Portenta H7 Lite, telemetrie, IoT, Formula Student, CAN bus, Arduino Cloud, mikro SD karta, GPS

Abstract

In this thesis, was analyzed the existing telemetry solution of the CULS Prague formula racing team and a suitable solution for modernization was sought. The main problems were unreliable data storage and low accuracy GPS module. Furthermore, the team dealt with the transition to a hybrid and a new control unit, which brought a large amount of new data for which it was necessary to increase the speed of communication. An analysis of possible technologies in the areas of live data display, data storage, GPS modules, status display, and communication protocols was created in the search for suitable solutions. Based on this analysis, a complete solution was proposed using an Arduino Portenta H7 Lite, a micro SD card, a dual frequency GPS module, an LTE router and an Ethernet cable to connect to the Arduino Cloud and using two CAN bus buses for fast and reliable communication between the different units on the monopost. This proposal was only partially implemented in this thesis due to the unavailability of all units. The functionality of the CAN bus buses and data storage on the SD card was tested on this partial implementation.

Keywords

Portenta H7 Lite, telemetry, IoT, Formula Student, CAN, Arduino Cloud, micro SD card, GPS

OBSAH

1 Úvod

Studentský tým CULS Prague formula racing se pro závodní sezónu 2024 mezinárodní soutěže Formula Student rozhodl pro realizaci výrazných změn v oblasti telemetrie ve snaze zlepšit a vyvinout konkurenceschopný monopost v dalších letech. Soustředění bylo směřováno na sběr dat pro budoucí vývoj podvozku, aerodynamiky, motoru a elektroniky vozu. Dalším důvodem pro modernizaci telemetrie byla statická disciplína EDR, kde je hodnocen návrh, znalosti, konstrukce vozidla a především podklady, ze kterých se pro daný návrh vycházelo. Tyto podklady jsou zajišťovány právě daty z telemetrie. Pro sezónu 2024 bylo učiněno rozhodnutí změnit řídící jednotku motoru. Toto rozhodnutí znamenalo pro tým velký krok do neznáma. Tento krok by ale mohl být s kvalitními daty pro budoucnost vývoje velmi důležitý. Přínosem této modernizace bude lepší návrh a výkonnost monopostu, což přinese lepší výsledky týmu na závodech. A lepší výsledky přinesou větší prestiž týmu i univerzitě.

V minulosti byl velkým problémem nefunkční přenos a sběr telemetrických dat, což mělo přímý negativní dopad na provoz vozu a hledání příčin vzniklých problémů. Se vzrůstající složitostí systému a množstvím elektronických prvků, bylo pro následující generaci monopostu třeba tento nedostatek vyřešit.

Tato práce byla zaměřena na návrh modernizovaného konceptu telemetrie pro závodní monopost týmu CULS Prague formula racing a rozšíření možností zobrazování a sbírání dat. Vzhledem k průběhu předchozích závodů vznikl požadavek na živý přenos dat, aby hlavně mechanik na boxové zídce věděl, jaký je stav monopostu, a mohl dát instrukce druhému pilotovi před výměnou pilotů při vytrvalostním závodě a zajistil tak vyšší šanci dokončení nejvíce bodované disciplíny Endurance. Dalším důvodem je přehled ostatních členů o dění v závodě a ušetření jejich nervů při sledování závodu.

V průběhu závodu jsou kladeny vysoké nároky na systém, zahrnující například vibrace a teplotní změny. Hlavním zaměřením je spolehlivost a funkčnost systému získávání telemetrických dat, přičemž všechny použité součástky musí být v souladu s pravidly soutěže Formula Student, zejména součástky integrované do závodního monopostu.

TEORETICKÁ VÝCHODISKA.

2 Cíl a metodika

Cílem práce bylo zmodernizovat telemetrii v projektu Formula Student. Hlavním důvodem modernizace byla nespolehlivost stávajícího řešení. Dalším důvodem byl přechod na hybridní pohon, a tedy potřeba většího množství senzorů a tedy i dat. V rámci projektu Formula Student je potřeba vyvíjet stále lepší řešení. Pro analýzu jsou telemetrická data jedním z nejdůležitějších ukazatelů, jak monopost funguje, a jsou zpětnou vazbou navržených řešení. Zároveň bude díky telemetrii možné dát lepší zpětnou vazbu pilotům a tím dosáhnout v dynamických disciplínách lepších výsledků. Výsledkem by měl být kompletní návrh nového řešení a připravený kód s dostupnými jednotkami pro možnost budoucího rozšíření při kompletizaci monopostu na závody.

V rámci této práce bude nejdříve zjištěn aktuální stav telemetrie palubní jednotky, její kladné stránky a nevýhody daného řešení. Následně bude zjištěno, jaké jsou potřeby týmu pro nové řešení palubní jednotky. Z toho vyplyne zadání, pro které bude udělána analýza možných řešení a zvolí se to nejvhodnější pro potřeby týmu CULS Prague formula racing, z čehož pak vznikne návrh konceptu. Dalším krokem bude realizace řešení s dostupnými jednotkami, což obnáší programování a schéma zapojení. Nakonec bude vše otestováno, aby se ověřila funkčnost a spolehlivost tohoto řešení. Tyto testy budou provedeny na stole s dostupnými jednotkami v době zpracování této práce. Reálné testy na monopostu proběhnou po jeho dostavbě se všemi jednotkami, což tato práce bohužel vzhledem k termínu odevzdání nebude obsahovat.

3 Analýza stávajícího stavu telemetrie palubní jednotky na FS07

Telemetrie na monopostu FS07 byla tvořena Arduinem Mega, jakožto hlavním prvkem pro zpracování a distribuci dat, dále řídící jednotkou motoru Ignijet a jednotkou řazení. Nedílnou součástí telemetrie byly snímače, díky kterým byla získávána následující data: otáčky motoru, teplota chladící kapaliny, napětí baterie, signalizace řazení, teplota nasávaného vzduchu, tlak v sání, poloha škrtící klapky, rychlost předních kol, rychlost zadních kol, poměr lambda, aktuální zařazený stupeň, úhel řadícího mechanismu, GPS souřadnice, počet GPS satelitů, aktuální čas, zrychlení ve třech osách, náklon, XBee komunikace, tlak paliva, tlak mazání motoru, teplota oleje motoru a úhel natočení volantu. Komunikace v monopostu mezi snímači a Arduinem Mega zajišťovala primárně CAN bus sběrnice, ale také se využívala sériová linka, I2C sběrnice a analogové signály. (11)

K zobrazení dat také sloužil displej ve volantu monopostu, kde byly zobrazeny kritické hodnoty pro kontrolu stavu monopostu. Konkrétně se v displeji zobrazovalo napětí baterie, teplota vody a oleje, tlak v sání, otáčky a zařazený převodový stupeň.

3.1 Přenos dat pomocí rádiové komunikace s použitím XBee modulů

Jediné ukládání dat bylo řešené bezdrátovou komunikací pomocí dvou XBee modulů, kde jeden byl na monopostu v režimu vysílače a druhý modul byl v blízkosti trati zapojený do notebooku v režimu přijímače. Pro spolehlivější přenos dat byl tento přijímač na třímetrovém prutu. Tímto řešením bylo možné posílat maximálně 250 kilobitů za sekundu. Komunikace pracovala po sériové lince nastavené na rychlosti 38400 baudů. Bohužel toto řešení nebylo dostatečně spolehlivé a na závodech docházelo ke ztrátě dat, z důvodu ztráty signálu XBee modulů. Navíc byl pohyb s dlouhým prutem nepohodlný a nebezpečný pro ostatní účastníky závodů.

3.2 Zpracování získaných dat

Pro ukládání a zároveň zobrazování dat byla použita aplikace Microsoft Office Excel s doplňkem Data Streamer. Data se po ukončení komunikace uložila ručně ve formátu csv. Během streamování byl v aplikaci list, který zobrazoval živá data, případně vykresloval grafy například pro akcelerometr nebo plyn.

4 Potřeby týmu CULS Prague formula racing

Tým se na závodech FS Alpe Adria setkal s problémem, že vypadla komunikace pomocí XBee modulů, a z toho důvodu neměl žádná data. Proto byl u nového návrhu kladen velký důraz na spolehlivost ukládání dat. Dále tým dospěl k rozhodnutí soutěžit s hybridním monopostem, což vyžaduje větší množství dat.

4.1 Výhody a nevýhody stávajícího řešení

Jednou z hlavních výhod byla nízká spotřeba energie XBee modulů, jejich nízká hmotnost a malé rozměry. Dalším velmi dobrým řešením byl displej ve volantu pro zobrazení kritických hodnot, ať už pro mechanika, nebo pro pilota při závodě, díky kterým může snadno přizpůsobit jízdu stavu monopostu. Výhodou byla i možnost sledovat data v reálném čase i mimo monopost.

Hlavní nevýhodou byla nespolehlivost ukládání dat v případě, že selhala komunikace mezi XBee moduly. Pro zjišťování polohy byl používán běžný GPS modul, který nedosahuje přesnosti pro analýzu stopy pilota na trati. Další nepříjemností byl třímetrový prut pro anténu přijímajícího XBee modulu a relativně pomalé ukládání dat.

4.2 Specifikace zadání a požadavků od modernizace palubní jednotky

Pro tým je velmi důležitá spolehlivost ukládání dat. Dále je velkou prioritu okamžité zobrazení stavů jednotek na monopostu, pro snadnou a přehlednou kontrolu funkčnosti jednotlivých částí monopostu, a proto, aby v případě problému bylo zřejmé, čeho se týká. Tým požaduje přesnější GPS data, což znamená výběr nového GPS modulu. Dále přípravu telemetrie pro přidání dalších senzorů, které by v budoucnu mohly být potřeba. V poslední řadě by se týmu líbilo živé zobrazení dat, aby při závodech věděl, zda jsou určité hodnoty na monopostu v normálních hodnotách či nikoliv.

S týmem bylo rozhodnuto zachovat řešení, která již fungovala spolehlivě. Sběr dat z řídící jednotky pomocí CAN bus – je možné využít vyšších rychlostí než původních 250 kbps. Zobrazení dat pomocí displeje Nextion ve volantu po sériové lince rychlostí

512 000 bps. Použití akcelerometru MPU9250 s gyroskopem přes I2C – změnit rozsah z 2 g na 4 g.

Dále byly specifikovány nové úkoly a požadavky. Hlavním úkolem bylo vymyslet spolehlivé řešení ukládání dat v různých rychlostech dle potřeby a ve větším množství. Dále bylo potřeba vyřešit nové řešení zobrazení dat mimo monopost. Zjistit, zda jsou cenově dostupná přesnější řešení GPS moduly. Možnost přidání akcelerometrů do kol a teplotních senzorů kol pro testování podvozku. Pro indikaci stavů jednotek zakomponovat větší množství sledovaných jednotek na monopostu, pro lepší přehled o funkčnosti jednotlivých částí a snazší hledání problémů.

5 Analýza možných technologií

V rámci této analýzy byly zjištěny možnosti řešení pro jednotlivé části telemetrie. Zaměření bylo na průzkum nových i využívaných technologií pro jejich porovnání a volby nejvhodnějších řešení pro aktuální potřeby týmu specifikované v předchozí kapitole. Konkrétně bylo potřeba analyzovat možnosti živého zobrazení dat, ukládání dat, GPS modulů, zobrazování stavů jednotek monopostu a komunikačních protokolů.

5.1 Technologie pro živé zobrazení dat

Předchozí verze telemetrie využívala pro živé zobrazení dat zigbee síť, jejíž nedostatky byly zmíněny v kapitole 4.1, a dle požadavků týmu bylo třeba najít vhodnější a spolehlivější technologii. Pro živé zobrazení dat je v první řadě potřeba zajistit jejich bezdrátový přenos z monopostu, což zahrnuje dosah a rychlost přenosu. V druhé řadě je potřeba najít způsob, jakým data zobrazit pro členy týmu.

5.1.1 Bluetooth (IEEE 802.15.1)

Jde o bezdrátový standard, který využívá volné frekvenční pásmo 2 400 až 2 483,5 MHz bez nutnosti licencí. Tento standard byl navržen tak, aby umožňoval přenosy dat buď mezi dvěma body (bod-bod), nebo od jednoho bodu k více bodům (bod-více bodů). Dosah tohoto bezdrátového komunikačního řešení se pohybuje od 10 do 100 m, což je pro naše použití příliš krátký dosah. Maximální rychlost komunikace dosahuje až 721 kbps, což se dá považovat za výhodu této technologie. Avšak pro naše potřeby je tato technologie nevyhovující kvůli nedostatečnému dosahu. (1, 2)

5.1.2 Wi-Fi (IEEE 802.11)

Wi-Fi technologie je tvořena bezdrátovou síť WLAN, což nahrazuje klasickou LAN. Na volném prostranství má dosah až 500 m s propustností přibližně 500 kbps. Tyto parametry se jeví velmi dobře, ale je obavaz nespolehlivosti této metody. Dosah signálu je na hraně a pravděpodobně by signál nepokryl celou trať. Výhodou této metody je, že přímo podporuje komunikaci mezi Arduino Cloudem a Arduinem. (3)

5.1.3 Zigbee (IEEE 802.15.4)

Profil protokolu Zigbee/IEEE 802.15.4 je navržen pro nízkoenergetické sítě, určený pro bezdrátové monitorování. Pracuje na nelicencovaném pásmu 2,4 GHz na rychlosti 250 kbps. Zigbee síť podporuje stromovou, hvězdicovou a mesh topologii. Dosah mezi jednotlivými vysílači je ve venkovním prostranství do 100 m. Tento dosah není dostačující. Dal by se sice prodloužit rozmístěním více vysílačů pro prodloužení signálu, ale to většinou na závodech není možné. Výhodou této technologie je nízká energetická náročnost. (8)

5.1.4 GSM

GSM síť představuje digitální buňkovou radiokomunikační infrastrukturu, která umožňuje přenos hlasu a dat. V Evropě používá systém GSM dvě rádiová pásma 900 MHz a 1 800 MHz. Aby bylo možné přenášet hlasové signály, je nezbytné transformovat původní analogový signál do digitální podoby, která je vhodná pro přenos v rámci GSM sítě. Přenosová rychlost této technologie je do 100 kbps. Dosah závisí na mobilním signálu v dané oblasti. Tato technologie připadá v úvahu pro dané použití. Výhodami jsou pro většinu oblastí dosah a datová rychlost, která by měla být pro dané účely dostačující. Nevýhodou je, že tuto komunikaci přímo nepodporuje Arduino cloud. (4)

5.1.5 LTE router a ethernet (IEEE 802.3)

LTE router po vložení SIM karty umožňuje konektivitu. Dle parametrů na stránkách výrobce je přenosová rychlost nahrávání 10 Mbps. Pro připojení k routeru se využije ethernet. Výhodou tohoto řešení je, že bezdrátovou konektivitu řeší přímo k tomu určený router. Spojení mezi Arduinem a routerem zajišťuje relativně spolehlivý ethernet. Další výhodou je také, že připojení přes ethernet podporuje Arduino Cloud pro komunikaci s Arduino Portenta H7 Lite. Nevýhodou tohoto řešení je relativně větší hmotnost routeru, ale měla by zde být výrazně vyšší spolehlivost. (7, 9)

5.2 Technologie spolehlivého ukládání dat

Spolehlivé ukládání dat bylo pro tento návrh telemetrie jednou z nejdůležitějších částí. Z toho důvodu bylo potřeba najít ideální řešení. Vzhledem k tomu, že byla telemetrie navrhovaná na závodní monopost, tak bylo potřeba kromě spolehlivosti najít co nejmenší a tím i nejlehčí technologii.

5.2.1 Mikro SD karta

Jednou z možností, jak ukládat data je na mikro SD kartu, která bude připojená na snadno přístupném místě na desce telemetrie. Toto řešení se jeví jako spolehlivé. Další výhodou je jednoduchá práce, kde mikro SD kartu lze rychle a snadno vyměnit a po vyjmutí ručně data uložit na týmové uložiště. Nevýhodou je limitovaná velikost uložiště mikro SD karty, ale kapacita uložiště by pro testování i závody měla být dostatečná. V dnešní době se dají sehnat mikro SD karty s kapacitou až 1 TB.

5.2.2 Bezdrátové ukládání na cloudové uložiště

Tento způsob ukládání dat funguje na konektivitě k síti, díky které se potřebná data pošlou na cloudové uložiště a zde jsou po určitou dobu uloženy. Při bezdrátovém přenosu může vzniknout problém s množstvím přenášených dat a přehlcením serveru, z toho důvodu považuji toto řešení pro naše účely za nedostatečné. Výhoda tohoto řešení je v dostupnosti dat okamžitě na cloudu.

5.3 Porovnání GPS modulů v poměru cena přesnost na trhu

Tým doposud používal běžné GPS moduly. Pro letošní sezónu se rozhodl, že by bylo vhodné získávat přesnější data o poloze pro možnost vykreslování trati a v ideálním případě i analýzy stopy jednotlivých pilotů. Tato data v kombinaci s dalšími senzory monopostu budou využity pro nastavení řízení hybridu.

5.3.1 Standartní GPS

GPS neboli Globální polohový systém, je satelitní navigační systém, který umožňuje určit polohu a čas kdekoli na Zemi nebo poblíž ní. Systém GPS využívá souboru satelitů umístěných ve vesmíru, které pravidelně vysílají signály. Tyto signály mohou

být přijímány a zpracovávány přijímači GPS na Zemi. Standardní GPS systémy dosahují v ideálních podmínkách přesnosti 3 až 5 m. Za zhoršených podmínek se přesnost zhoršuje.

5.3.2 RTK GPS

Princip RTK GPS spočívá v korigování dat v reálném čase. Základní stanice (base station) je na známém místě a nepohybuje se. Zároveň přijímá data o svojí poloze ze satelitu. Pokud se data neshodují s polohou základní stanice, tak se vypočítají korekční data. Ta se pošlou pohyblivé RTK GPS, o jejíž poloze potřebujeme znát přesné údaje. Tato metoda dosahuje přesností až 2,5 cm.

Jsou dvě možnosti base station. První možností je mít vlastní základní stanici a vše si nastavit. Tato možnost má výhodu, že nikdo jiný neovlivňuje množství korekčních dat. Ale je potřeba mít vlastní zařízení a vše správně nastavit. Druhou možností je využít komerční služby, která poskytuje korekční data ze základních stanic v okolí. U této metody může být omezené množství poslaných korekčních dat v čase a také může nastat problém v oblasti, kde tyto komerční stanice nejsou. (12)

5.3.3 Multifrekvenční GPS

Multifrekvenční GPS dokáže díky více frekvencím překonat více překážek mezi satelitem a GPS modulem. Tím se zvyšuje přesnost získané polohy. Touto technologií lze dosáhnout přesnosti 1,5 m. (12)

5.4 Možnosti ukazatelů stavů jednotek na monopostu

Ukazatele stavů jednotek jsou zásadní pro kontrolu monopostu před jízdou, při jízdě i po dojetí. V minulosti se týmu velmi dobře osvědčil displej. Další možností je využívat indikační LED diody.

5.4.1 Displej

Pro zobrazení dat lze využít displeje, na kterém lze zobrazit téměř cokoliv, co bude naprogramováno. Při programování displeje je potřeba přemýšlet nad rozložením zobrazovaných údajů a jejich množstvím. Případně je možnost vytvořit více obrazovek

pro různé potřeby. Do displeje je potřeba přímo posílat data, která se mají zobrazovat. (11)

5.4.2 Indikační LED diody (kontrolky)

Jednoduchý způsob, jak rychle zjistit stav funkčnosti jednotlivých jednotek ve formuli, je zobrazení pomocí indikační LED diody. Jsou dvě možnosti, jak LED diody využít. První možností je, že se LED dioda rozsvítí v případě, že daná jednotka není v pořádku. Další možností je měnit barvu pro více stavů dané jednotky. Například modře pro studený olej, zeleně pro optimální teplotu oleje, žlutě pro stav lehce nad optimálním rozsahem a červeně pro přehřátý olej. Tato metoda si vyžaduje stejný počet LED diod, kolik údajů je potřeba indikovat. Následně lze jednoduše měnit barvy, případně rozsvěcet nebo zhasínat LED diody podle stavů. (11)

5.5 Komunikační protokoly

Pro přenos dat mezi jednotlivými jednotkami na monopostu bylo třeba zvolit vhodný komunikační protokol. V automobilovém průmyslu je pro komunikaci velmi rozšířená CAN bus sběrnice.

5.5.1 CAN bus

CAN (Controller Area Network) je sběrnice tvořená dvěma vodiči CAN_L (low) a CAN_H (high) zakončená na každé straně 120 Ω rezistorem. Dosahuje přenosové rychlosti až 1 Mbit/s pro délku sběrnice do 40 m. Na sběrnici je možné vysílat za určitých podmínek, které určuje priorita zprávy. Každá zpráva má svoje identifikační číslo, díky kterému lze zprávu přijmout. Zprávy jsou také rozděleny do čtyř rámců na datový rámec, žádost o data, chybový rámec a rámec přeplnění. Norma protokolu udává dvě specifikace rámců CAN 2.0 A s délkou 11 bitů a CAN 2.0 B s délkou 29 bitů. Mezi nevýhody CAN bus lze zařadit omezený počet dat posílaných jednou zprávou a náročnost nastavení registrů. (15)

5.5.2 SPI

SPI (Serial Peripheral Interface) je synchronní sběrnice, která je využívána ke komunikaci mezi mikrokontrolery a připojenými periferiemi, jako jsou například snímače nebo SD karta. Využívá časové signálu (SCK) pro vzorkování podle náběžné nebo sestupné hrany. Dále využívá pinů periferie vstup / kontrolér výstup (PICO) a periferie výstup / kontrolér vstup (POCI) pro komunikaci mezi kontrolérem a periferiemi a Chip Select (CS) pro odpojování od sběrnice, respektive připojení. pouze pokud to je potřeba. (16)

5.5.3 I2C

I2C (Inter-Integrated Circuit) je komunikační sériová sběrnice, která umožňuje připojení více zařízení typu master (hlavní) a slave (podřízená). Slave zařízení mohou posílat data, pouze pokud jsou k tomu vyzvána od mastera. Každé zařízení má svou adresu, pomocí které se rozlišují. I2C je tvořena dvěma vodiči SDA (Seriál Data Line) pro data a SCL (Seriál Clock Line) jako signál s informací o čase. (6)

PRAKTICKÁ ČÁST

6 Návrh konkrétního řešení

V rámci modernizace telemetrie palubní jednotky byla, na základě předchozích zkušeností týmu a jejich aktuálních potřeb, zvolena vývojová deska Arduino Portenta H7 Lite, která je řídícím operačním centrem telemetrie. Hlavním komunikačním kanálem mezi Portentou a prvky v monopostu je CAN bus, kde bylo navrženo využívat dvě separátní CAN sběrnice. Jedna CAN sběrnice (CAN 1) je určena pro přenos kritických dat, na kterých je závislá funkčnost monopostu. Po druhé CAN sběrnici (CAN 2) jsou přenášena všechna ostatní data. Pro ukládání dat bylo navrženo použití mikro SD karty. Bezdrátový přenos dat byl navržen prostřednictvím Arduino Cloud, kde je konektivita k síti řešená pomocí routeru MIRO-L200 a ethernetového kabelu. Dále byl vybrán GPS modul s dvou frekvenční anténou s přesností 1,5 m, která stejně jako displej komunikuje po sériové lince. Ve finálním řešení bude k telemetrii připojena přístrojová deska s ovládacími prvky jako jsou přepínače, tlačítka a LED diody.

6.1 Metodika realizace návrhu

Na realizaci v rámci této práce bude využita rozšířená platforma pro Arduino Portenta H7 Lite, kterou je Portenta Breakout. Jako první věc bude zapojena Portenta H7 Lite do Portenta Breakout pomocí HD konektorů. Bude také napsán kód pro ukládání času do proměnné pomocí RTC, který je součástí Portenta Breakout.

Dále bude potřeba zprovoznit hlavní komunikační kanál, kterým je CAN bus. Pro testování bude využit CAN převodník a teplotní senzor teploty pneumatik. Pro správné zapojení CAN sběrnice je potřeba zapojit terminační rezistory (120 Ω) na dva konce sběrnice. Následně bude zvolena vhodná knihovna a s využitím jejich příkladů napsán kód, který bude potřeba ladit do fáze, kdy hodnoty na sériovém okně budou ukazovat předpokládané hodnoty senzoru.

Dalším krokem bude návrh ukládání dat na mikro SD kartu, tedy připojení mikro SD karty do Portenta Breakout, napsání kódu pro ukládání času na mikro SD kartu a ukládání dat z teplotního senzoru. Funkčnost bude ověřena vložením mikro SD karty

do čtečky karet v notebooku a porovnáním dat na mikro SD kartě s výpisem na sériovém okně v programu Arduino IDE, který by měl být ve stejný čas shodný.

Následně bude otestována funkčnost posílání živých dat pomocí Arduino Cloud, a to připojením pomocí routeru. Router bude potřeba nastavit podle návodu od výrobce. Poté bude připojen router s Portenta Breakout pomocí ethernetového kabelu a vše bude nastaveno ve webovém prostředí Arduino Cloudu. Následně bude potřeba ověřit, zda se zobrazovaná data v aplikaci Arduino Cloud shodují s daty na sériovém monitoru. Pro zprovoznění funkcionality bude využito bezplatné verze Arduino, ve které je možné zobrazit pouze 5 proměnných.

Poté bude potřeba vyzkoušet, zda vše funguje, i když není připojené Arduino pomocí USB-C kabelu k notebooku. K napájení Arduina bude využita baterka, která je doporučena na stránkách Arduino Portenta H7 Lite konkrétně „Li-Po Single Cell, 3.7 V, 700 mAh“ (13). Po připojení bude ověřena shoda hodnot, které se zobrazí na Arduino Cloudu s hodnotami z mikro SD karty.

V dalším kroku budou všechna zařízení komunikující po CAN bus sběrnici připojena a rozdělena na zařízení, která budou komunikovat po CAN 1 (kritické zprávy pro chod monopostu) a po CAN 2 (ostatní zprávy). Vzhledem k tomu, že Arduino Portenta H7 Lite nabízí pouze jeden CAN, bude potřeba využít pro CAN 2 SPI sběrnici (SPI 1) a MCP2515 modul. Poté bude ověřeno, zda příchozí zprávy odpovídají daným stavům a zda nedochází k rušení nebo jiným nežádoucím jevům zobrazením komunikace na osciloskopu.

Nakonec bude otestován celý kód se všemi připojenými zařízeními a ověřeno, zda vše běží spolehlivě a zda jsou data odpovídající stavům senzorů. Celé zapojení je schematicky zobrazeno na Obrázek 1.

6.2 Arduino Portenta H7 Lite

Arduino Portenta H7 Lite je oproti Arduino Portenta H7 cenově dostupnější řešení bez možnosti video výstupu a pokročilých bezpečnostních vychytávek. Díky dvěma procesorům dokáže v reálném čase spustit více částí kódu současně, což je pro nás velkou výhodou pro CAN komunikaci. Portenta je vybavena dvěma 80 pinovými HD konektory s velkou hustotou pinů a je kompatibilní s Arduino Cloudem. Pro připojení

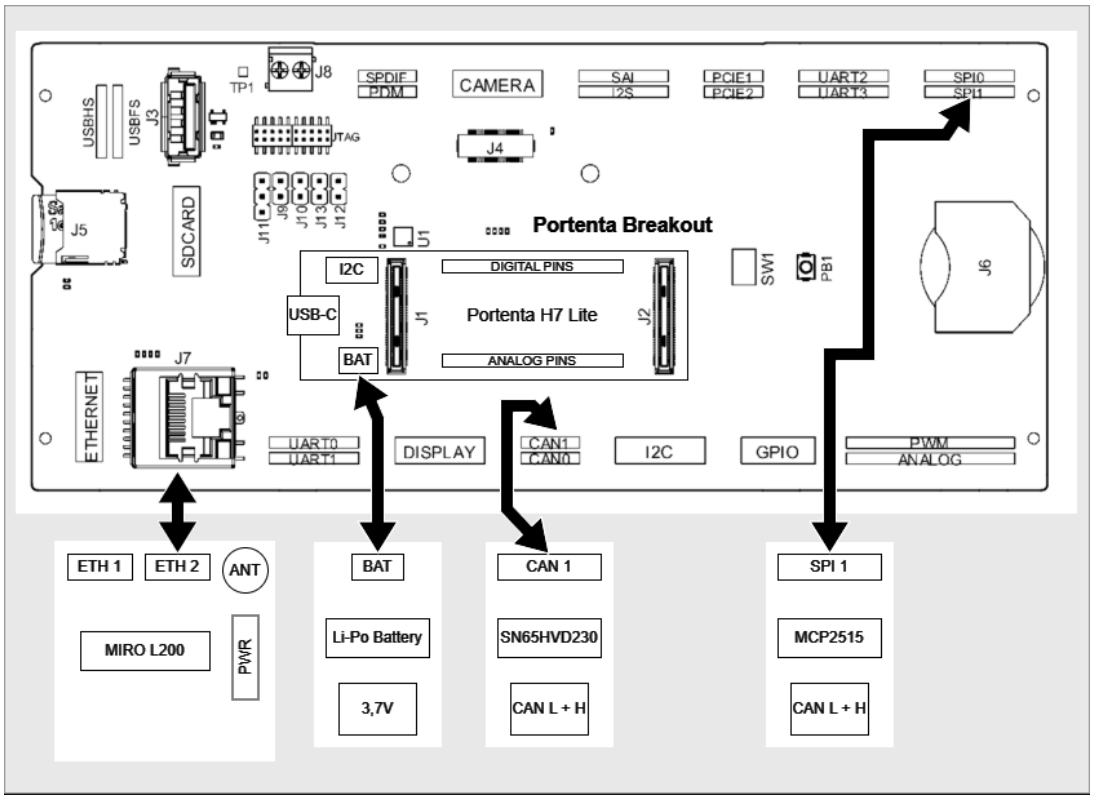
Arduina za účelem programování je možné využít USB-C konektor (13). Technické specifikace jsou v následující Tabulka 1.

Tabulka 1: Technické specifikace Arduino Portenta H7 Lite

Microcontroller	STM32H747XI dual Cortex®-M7+M4 32bit low power Arm® MCU
Secure Element (default)	Microchip ATECC608
Board Power Supply (USB/VIN)	5V
Supported Battery	Li-Po Single Cell, 3.7V, 700mAh Minimum (integrated charger)
Circuit Operating Voltage	3.3V
Current Consumption	2.95 µA in Standby mode (Backup SRAM OFF, RTC/LSE ON)
Timers	22x timers and watchdogs
UART	4x ports (2 with flow control)
Ethernet PHY	10 / 100 Mbps (through expansion port only)
SD Card	Interface for SD Card connector (through expansion port only)
Operational Temperature	-40 °C to +85 °C
MKR Headers	Use any of the existing industrial MKR shields on it
High-density Connectors	Two 80 pin connectors will expose all of the board's peripherals to other devices
Camera Interface	8-bit, up to 80 MHz
ADC	3× ADCs with 16-bit max. resolution (up to 36 channels, up to 3.6 MSPS)
DAC	2× 12-bit DAC (1 MHz) available, only one is accessible by the user through the external A6 pin
USB-C	Host / Device, High / Full Speed, Power delivery

Zdroj: (13)

Obrázek 1: Schéma zapojení realizovaného řešení



Zdroj: Upraveno z (22)

6.3 Podsystémy telemetrie

Jedná se o kompletní koncept, který bude využit na monopostu FS09 (pro sezónu 2024). Vzhledem k období, kdy se tato práce odevzdává, nejsou všechny podsystémy dostupné, ale v rámci vývoje byly tyto podsystémy vybrány. Na Obrázek 1 je vidět schéma zapojení realizované části v období odevzdání této práce.

6.3.1 CAN 1

Jedná se o CAN bus sběrnici, která je označována jako CAN 1 a která je určena pro komunikaci kritických hodnot pro běh monopostu. Rychlost je 1 Mbps fixně určená řídící jednotkou ECU Master. Tato komunikační sběrnice je využívána na kritická data potřebná pro fungování monopostu. Komunikace probíhá mezi hybridní jednotkou, jednotkou spalovacího motoru a jednotkou telemetrie. Tuto komunikaci jednotka telemetrie čte a ukládá na mikro SD kartu. Konkrétně se jedná o data polohy plynového

pedálu, polohu škrtící klapky, vstupní výkony elektromotorů, zařazený stupeň, otáčky motoru a startovací signál.

6.3.2 CAN 2

Jedná se o CAN bus sběrnici, která je označována jako CAN 2 a která je určena pro ostatní data, aby se nepřehlcoval CAN 1 a zvýšila se tak spolehlivost. Rychlost je nastavena na 500 kbps. Tato komunikační sběrnice je určena pro všechny ostatní data, které nejsou kritické pro chod monopostu, ale jsou důležité pro možnost analýzy a ověření správné funkčnosti, případně validaci návrhu vůči realitě. Tato sběrnice je využívána pro posílaní dat ze senzoru tlaku v sání, teplot před a za chladičem, lambda sondy, teploty oleje, brzdového tlaku, teploty vzduchu, napětí LV baterky, teplotních senzorů pneumatik, otáček kol, natočení volantu a přepínačů.

6.3.3 Mikro SD karta

Slouží pro ukládání veškerých dat, které telemetrie přečte. Pro příjemnou práci s uloženými daty byl zvolen formát .csv, ve kterém je první sloupec určen pro čas, aby bylo možné data zpětně spustit chronologicky a s informací o čase. Následně je každý sloupec určen pro jeden konkrétní údaj, ukládaný se změnou času na jednotlivé řádky. Do mikro SD karty se ukládají data polohy plynového pedálu, vstupní výkony elektromotorů, zařazený stupeň, otáčky motoru, teploty vzduchu v sání, tlaku v sání, teploty vody před a za chladičem, lambda sondy, teploty oleje, brzdového tlaku, napětí LV baterky, otáček kol, akcelerometru, teploty pneumatik, natočení volantu, přepnutý režim, přepnutí větráku, GPS data, napětí HV baterky a status HV baterky.

6.3.4 Router MIRO-L200

Pro připojení k síti byl vybrán router MIRO-L200 od firmy INSYS icom, protože splňoval požadavky týmu a zároveň byla s firmou navázána spolupráce. Tento router podporuje 4G/LTE připojení pomocí mobilních dat ze SIM karty, která je od společnosti T-Mobile. Router má dva RJ45 konektory s přenosovou rychlostí 10/100 Mbit/s, kde jeden bude využit pro připojení telemetrie a druhý pro připojení hybridu k síti. Dále je router malý (šířka 26 mm, hloubka 77 mm a výška 99 mm)

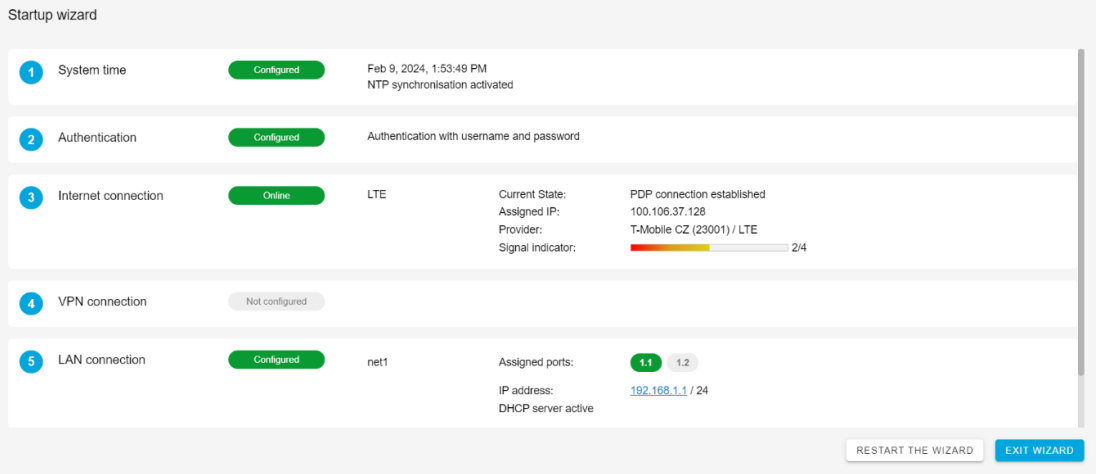
a lehký (100 g s krabičkou, 48 g bez krabičky). Router s krabičkou splňuje IP40, operační teplota je -40 až +75 °C a k napájení routeru je potřeba 12 až 24 V (20).

Pro nastavení routeru se vycházelo z návodu od výrobce (17), podle kterého byla vložena SIM karta, namontována anténa, připojeno napájení 12 V, připojen router ethernetovým kabelem s notebookem a zadána IP adresa routeru do prohlížeče, čímž bylo otevřelo rozhraní pro nastavení routeru. Následně byl spuštěn „Startup wizard“.

Po stisknutí tlačítka start bylo nastaveno časové pásmo, poté NTP server a zaškrtnuto pole pro synchronizaci času při připojení k internetu. V dalším kroku byla požadovaná autentizace, kam bylo vyplněno jméno a heslo. Poté bylo potřeba vyplnit typ připojení, PIN k SIM kartě a vložit APN SIM karty viz. Ve čtvrtém kroku byla vybírána VPN, kterou bylo zvoleno nevyužívat žádnou, aby nebyl problém s IP adresou při připojení na Arduino Cloud.

V dalším kroku, byla možnost nahrání před vytvořeného nastavení z webu výrobce, který však nebyl k dispozici, proto byl tento krok pouze přeskočen a zobrazil se souhrn nastavení “Startup wizard”, ve kterém byla informace o připojení k síti a sílu signálu, vzhledem k testování uvnitř budovy byl signál slabší viz Obrázek 2.

Obrázek 2: Startup wizard routeru MIRO-L200

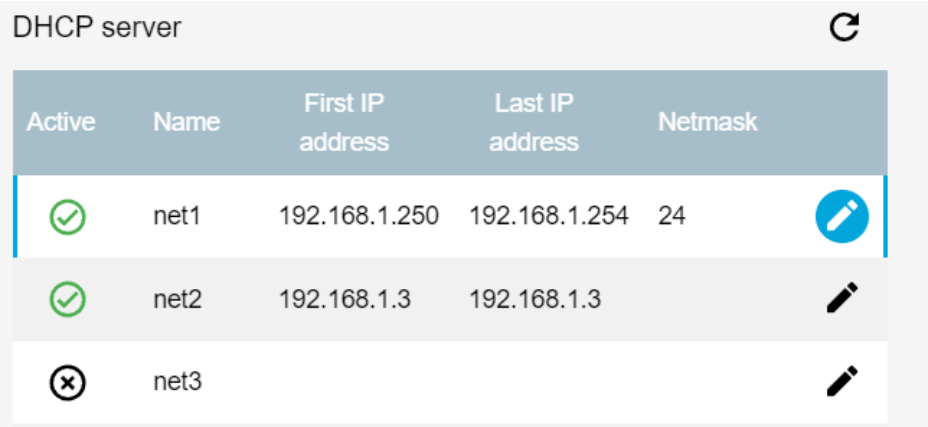


Zdroj: (17)

Dále bylo potřeba nastavit jedinečnou IP adresu pro připojené Arduino Portenta H7 Lite, která se běžně nastavuje prostřednictvím DHCP serveru a MAC adresy zařízení.

Bohužel se nepodařilo zjistit MAC adresu zařízení. Tento problém byl vyřešen pomocí konfigurace druhé sítě na DHCP server s jedinou IP adresou viz Obrázek 3.

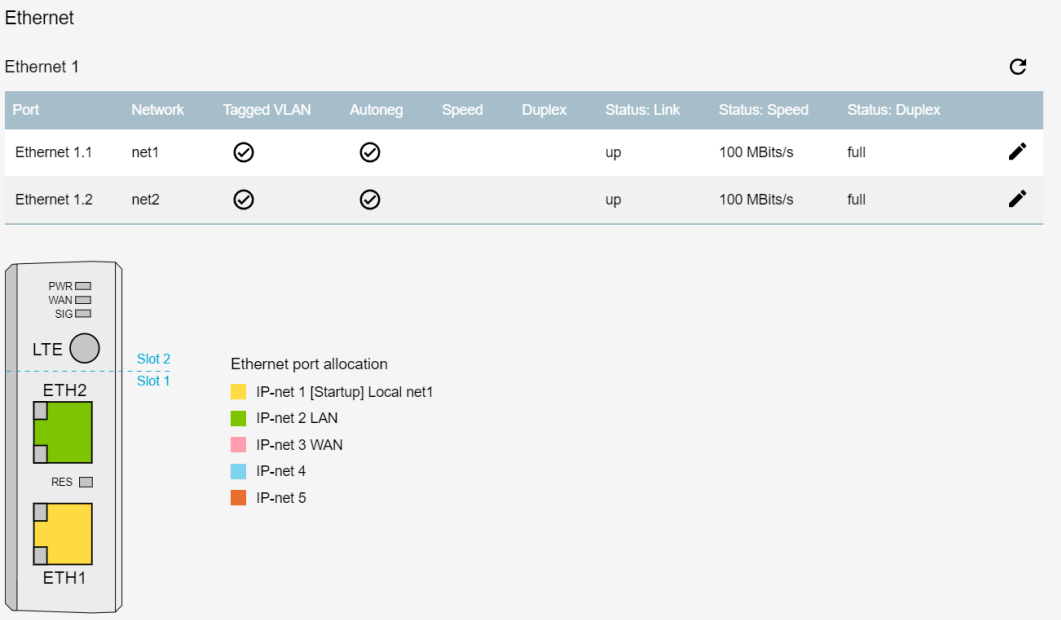
Obrázek 3: Nastavení DHCP serveru na routeru MIRO-L200



Zdroj: (17)

Tato síť 2 byla následně přiřazena na port ETH2, který byl využíván pro připojení Arduina Portenty H7 Lite s routerem viz Obrázek 4. Na tomto obrázku lze také pozorovat rychlost přenosu dat, konkrétně se jedná o 100 Mbit/s. Port ETH1 bude využíván pro připojení hybridní jednotky k síti, která je mimo rozsah této práce.

Obrázek 4: Nastavení ethernetových portů routeru MIRO-L200



Zdroj: (17)

6.3.5 Arduino cloud

Arduino Cloud je web od firmy Arduino určený pro správu a připojení zařízení přes internet. Na tomto webu jsou k dispozici návody a dokumentace Arduino Cloudu, které přibližují funkcionality této webové stránky. Je zde možnost přidávat zařízení, vytvářet proměnné a skici, což jsou v podstatě kódy pro vývojové desky Arduino. Pro zobrazení dat je určen Dashboard, který nabízí spoustu widgetů s různými funkcionalitami. Web nabízí i šablony pro různé projekty z oblasti internetu věcí. Primárně je určený pro připojení Arduino vývojové desky, ale je možné připojit i jiné vývojové desky.

Prostřednictvím Arduino Cloudu budou z formule živě zobrazována data, důležité pro vzdálené zjištění, zda je s formulí vše v pořádku a v případě problému jeho včasném odhalení. Konkrétně bude zobrazována teplota oleje, vzduchu a vody, napětí LV baterky a počet GPS satelitů.

6.3.6 RTK-DUAL GPS od LOCOSYS

Dvou frekvenční dvou anténový modul operující na frekvencích L1 a L5. V našem režimu použití Autonomous dosahuje přesnosti pozice 1,5 m CEP a přesnost určení směru je 0,16° až 0,32°. Baud rate modulu je 115 200 bps. Výstup modulu je ve formátu NMEA. Modul je dále vybaven MEMS senzorem (12).

Tento GPS modul vyžaduje dvě antény. Byly vybrány AA.178 Magma X od firmy Taoglas, protože podporují frekvence L1 a L5 a zároveň mají kompaktní rozměry vhodné pro použití na závodním monopostu. Napájecí napětí antény je v rozsahu 1,8 – 5 V, její operační teplota má rozsah od -40 °C do +85 °C a splňuje IP67. Dále je vybavena 3 m kabelem s SMA konektorem (21).

6.3.7 Akcelerometry

Hlavní akcelerometr bude umístěn za přístrojovou deskou a bude přišroubován k rámu monopostu, aby byla data co nejméně zkreslená. Jedná se o akcelerometr s gyroskopem MPU9250. Rozsah bude nastaven na ±4 g a po montáži na monopost, budou zkalibrovány počáteční hodnoty pro jednotlivé osy ve vodorovné klidné pozici monopostu. Dále budou připraveny konektory pro připojení dalších 4 akcelerometrů,

které budou umístěny v případě potřeby do kol monopostu. Tato data budou využitelná při testování podvozku monopostu, kde je možnost pozorovat chování jednotlivých částí zavěšení vůči hlavnímu akcelerometru neboli monopostu jako celku.

6.3.8 Displej Nextion NX8048P050

V rámci závodů Formula student se tým účastní čtyř dynamických disciplín, pro které budou vytvořeny tři módy displeje a jeden pro testování aerodynamiky. Přepínání mezi těmito módy bude zajišťovat čtyřpolohový rotační přepínač umístěný na přístrojové desce monopostu. Pátý mód displeje je určen pro zobrazení dat důležitých pro ladění monopostu a zajímavých primárně pro mechanika monopostu. Tento mód bude na displeji zobrazen, vždy když bude monopost v klidu.

Akcelerace je disciplína, při které monopost z klidu jede rovně 75 m a jde o co nejkratší čas od startu do projetí cílové pásky (10). Proto je v tomto módu informace o stavu launch kontrolu, který si pilot zapne zmáčknutím tlačítka na volantu a omezí otáčky motoru na optimální hodnotu pro rozjezd monopostu. Také je zde číselná hodnota otáček motoru, aby pilot věděl, kdy přeřadit. Zároveň je tato informace zobrazována pomocí LED pásku na přístrojové desce. Dále je zobrazen zařazený převodový stupeň a indikace o stavu hybridního pohonu, který si pilot přepíná na přístrojové desce.

Druhou dynamickou disciplínou je Skidpad. V této disciplíně se jedou dvě kola doprava o průměru vnitřní kružnice 15,25 m a šířce tratě 3 m, kde je měřené druhé kolo, a následně se jedou dvě kola doleva, kde je opět měřeno druhé kolo (10). Výsledný čas disciplíny je průměr obou měřených kol. Pro tuto disciplínu je pro pilota důležité hlavně vnímání limitu monopostu a při měřeném kole nemá čas sledovat displej. Proto je zde zobrazena zařazená rychlost a kritické teploty, které si může zkontrolovat mezi jízdami.

Další dynamickou disciplínou je Autocross. Autocross je závod na jedno kolo, kde trať je pravidly omezená na délku do 1,5 km, maximální délka rovinek na trati je 80 m, s konstantními zatáčkami do průměru 50 m, vlásenkovými zatáčkami s minimálním vnějším průměrem 9 m a slalomem, kde jsou kuželky rozloženy v linii ve vzdálenosti 7,5 – 12 m. Minimální šířka tratě je pravidly specifikována na 3 m (10). Při tomto závodě se na displeji zobrazuje zařazená rychlost, informace o stavu hybridní baterie,

teplota oleje a voda. Dále se pilotovi při tomto režimu zobrazí, pokud je na monopostu něco v nepořádku, výraznou změnou s informací o daném problému, například vysoká teploty vody nebo oleje, případně pokud je vybitá baterka nízkého napětí nebo je problém s jakýmkoliv jiným systémem na monopostu.

Poslední dynamickou disciplínou je Endurance, což je vytrvalostní závod na 22 km s výměnou pilotů v polovině. Trať na Endurance je specifikovaná podobně jako na Autocross, s rozdílem, že trať pro Endurance je uzavřený okruh, aby bylo možné jet více kol po sobě. Dalším rozdílem je délka, která je pravidly určena přibližně na délku 1 km, a vyskytují se zde předjíždějící zóny, přes které pouští pomalejší vůz rychlejší po indikaci modré vlajky rozhodčími (10). Pro tento závod je mód stejný jako pro Autocross, protože jsou si tyto dvě disciplíny velmi podobné.

Další dynamický režim je určený pro testování aerodynamiky. Konkrétně se jedná o validaci dat ze simulací s realitou aerodynamického packetu, při tomto testování je důležité vidět aktuální rychlost. Testování aerodynamiky probíhá na dlouhé rovince, na které je potřeba jet ideálně konstantní rychlostí.

Poslední režim se zapne vždy, když má formule zařazený neutrál, je určen pro kontrolu mechanika, zda je vše v pořádku a je možné vypustit formuli na trať. Zobrazuje postupně správně spuštěné jednotky a zobrazí jejich ikony v zelené barvě a výpis špatně fungujících jednotek a jejich ikony červeně. Dále zobrazuje hodnoty teploty vody, teploty oleje, zařazený stupeň, tlak v brzdách, polohy plynového pedálu, polohu škrtící klapky, otáčky kliky motoru, tlak v sání, teplota v sání, lambda, zvolený režim, kill switche a inertia switch.

7 Testování

V rámci testování bylo nejdůležitější otestovat spolehlivost komunikace CAN bus a ukládání dat na SD kartu. Pro ověření funkčnosti budou využity dva teplotní senzory komunikující po CAN bus sběrnici a 16 GB mikro SD karta.

7.1 CAN bus

Pro ověření funkčnosti CAN 2 byly využity dva teplotní senzory pneumatik ALS Tire. Na

Obrázek **5** je vidět jejich výpis po sériové lince. V poli „cM_TEMPS“ na pozicích 0–3 je první polovina dat z prvního senzoru, která jsou posílány na ID 0x310, druhá půlka dat je na pozicích 4–7 na ID 0x311. Hodnoty je třeba vydělit 10, aby byly ve °C. Je tomu tak z toho důvodu, aby se zkrátila zpráva o desetinou čárku, takže hodnota 261 odpovídá teplotě 26,1 °C. Druhý senzor podobně posílá data na ID 0x350 a 0x351 v poli na pozicích 8–15.

Obrázek 5: Výpis přijatých dat po CAN2

```
18:32:06.482 -> cM_TEMPS[0]: 261 cM_TEMPS[1]: 246 cM_TEMPS[2]: 253 cM_TEMPS[3]: 253
18:32:06.482 -> Received data for ID 0x311:
18:32:06.482 -> cM_TEMPS[4]: 255 cM_TEMPS[5]: 257 cM_TEMPS[6]: 258 cM_TEMPS[7]: 257
18:32:06.666 -> Received data for ID 0x350:
18:32:06.666 -> cM_TEMPS[8]: 261 cM_TEMPS[9]: 256 cM_TEMPS[10]: 253 cM_TEMPS[11]: 252
18:32:06.666 -> Received data for ID 0x351:
18:32:06.666 -> cM_TEMPS[12]: 255 cM_TEMPS[13]: 259 cM_TEMPS[14]: 254 cM_TEMPS[15]: 260
18:32:06.666 -> Received data for ID 0x310:
18:32:06.666 -> cM_TEMPS[0]: 257 cM_TEMPS[1]: 249 cM_TEMPS[2]: 254 cM_TEMPS[3]: 251
18:32:06.666 -> Received data for ID 0x311:
18:32:06.666 -> cM_TEMPS[4]: 252 cM_TEMPS[5]: 253 cM_TEMPS[6]: 254 cM_TEMPS[7]: 255
18:32:06.836 -> Received data for ID 0x350:
18:32:06.836 -> cM_TEMPS[8]: 263 cM_TEMPS[9]: 253 cM_TEMPS[10]: 252 cM_TEMPS[11]: 254
18:32:06.836 -> Received data for ID 0x351:
18:32:06.836 -> cM_TEMPS[12]: 255 cM_TEMPS[13]: 257 cM_TEMPS[14]: 252 cM_TEMPS[15]: 255
18:32:06.836 -> Received data for ID 0x310:
18:32:06.836 -> cM_TEMPS[0]: 259 cM_TEMPS[1]: 251 cM_TEMPS[2]: 250 cM_TEMPS[3]: 255
18:32:06.836 -> Received data for ID 0x311:
18:32:06.836 -> cM_TEMPS[4]: 251 cM_TEMPS[5]: 258 cM_TEMPS[6]: 255 cM_TEMPS[7]: 256
18:32:07.025 -> Received data for ID 0x350:
18:32:07.025 -> cM_TEMPS[8]: 264 cM_TEMPS[9]: 249 cM_TEMPS[10]: 249 cM_TEMPS[11]: 249
18:32:07.025 -> Received data for ID 0x351:
18:32:07.025 -> cM_TEMPS[12]: 251 cM_TEMPS[13]: 262 cM_TEMPS[14]: 254 cM_TEMPS[15]: 258
18:32:07.072 -> Received data for ID 0x310:
```

Zdroj: vlastní zpracování

7.2 Mikro SD karta

Pro ověření funkčnosti SD karty bylo použito taktéž teplotních senzorů, jejichž data byla na SD kartu ukládána. Z teplotních senzorů jsou data posílána 5x za sekundu, proto tak bylo nastaveno i ukládání na SD kartu. Na SD kartu se taktéž ukládal aktuální čas. Na Obrázek 6 je vidět csv soubor s uloženými daty z teplotních senzorů.

Obrázek 6: Data z teplotních senzorů uložená na SD kartě

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	timeRTC	cM_TEMPS0	cM_TEMPS1	cM_TEMPS2	cM_TEMPS3	cM_TEMPS4	cM_TEMPS5	cM_TEMPS6	cM_TEMPS7	cM_TEMPS8	cM_TEMPS9	cM_TEMPS10	cM_TEMPS11	cM_TEMPS12	cM_TEMPS13	cM_TEMPS14	cM_TEMPS15
2	18:32:6	252	248	251	252	257	250	258	257	257	249	247	250	247	249	253	249
3	18:32:6	261	244	255	251	257	255	259	257	258	263	257	256	260	255	253	261
4	18:32:6	261	246	253	253	255	257	258	257	260	260	255	255	258	256	253	260
5	18:32:6	257	249	254	251	252	253	254	255	261	256	253	252	255	259	254	260
6	18:32:6	259	251	250	255	251	258	255	256	263	253	252	254	255	257	252	255
7	18:32:7	261	246	253	253	255	257	258	257	260	260	255	255	258	256	253	260
8	18:32:7	257	249	254	251	252	253	254	255	261	256	253	252	255	259	254	260
9	18:32:7	259	251	250	255	251	258	255	256	263	253	252	254	255	257	252	255
10	18:32:7	261	248	253	252	254	256	257	256	259	259	254	254	257	255	252	259
11	18:32:7	258	250	255	252	253	254	255	256	262	257	254	253	256	260	255	261
12	18:32:8	260	252	251	256	252	259	256	257	264	254	253	255	256	258	253	256
13	18:32:8	260	247	252	251	253	255	256	255	258	258	253	253	256	254	251	258
14	18:32:8	259	251	256	253	254	255	256	257	263	258	255	254	257	261	256	262
15	18:32:8	262	254	253	258	254	261	258	259	266	256	255	257	258	260	255	258
16	18:32:8	262	249	254	253	255	257	258	257	260	260	255	255	258	256	253	260
17	18:32:9	258	250	255	252	253	254	255	256	262	257	254	253	256	260	255	261
18	18:32:9	261	253	252	257	253	260	257	258	265	255	254	256	257	259	254	257
19	18:32:9	263	250	255	254	256	257	258	259	265	260	257	256	259	263	258	264
20	18:32:9	264	256	255	260	256	263	260	261	268	258	257	259	260	262	257	260
21	18:32:9	264	251	256	255	257	259	260	259	262	262	257	257	260	258	255	262
22	18:32:10	262	249	254	253	255	257	258	257	260	260	255	255	258	256	253	260
23	18:32:10	258	250	255	252	253	254	255	256	262	257	254	253	256	260	255	261
24	18:32:10	261	253	252	257	253	260	257	258	265	255	254	256	257	259	254	257
25	18:32:10	263	250	255	254	256	257	258	259	265	260	257	256	259	263	258	264
26	18:32:10	264	256	255	260	256	263	260	261	268	258	257	259	260	262	257	260
27	18:32:11	264	251	256	255	257	259	260	259	262	262	257	257	260	258	255	262
28	18:32:11	265	253	258	255	257	259	260	261	268	258	257	259	260	262	257	260
29	18:32:11	264	251	256	255	257	259	260	259	262	262	257	257	260	258	255	262
30	18:32:11	265	252	257	254	256	258	259	266	256	255	257	258	260	255	258	264
31	18:32:11	262	249	254	253	255	257	258	257	260	260	255	255	258	256	253	260
32	18:32:12	258	250	255	252	253	254	255	256	262	257	254	253	256	260	255	261
33	18:32:12	261	253	252	257	253	260	257	258	265	255	254	256	257	259	254	257
34	18:32:12	263	250	255	254	256	257	258	259	265	260	257	256	259	263	258	264
35	18:32:12	264	256	255	260	256	263	260	261	268	258	257	259	260	262	257	260
36	18:32:12	264	251	256	255	257	259	260	259	262	262	257	257	260	258	255	262
37	18:32:13	261	246	253	253	255	257	258	257	260	260	255	255	258	256	253	260

Zdroj: vlastní zpracování

8 Výsledky návrhu

Vzhledem k dostupným prvkům v době psaní této bakalářské práce nejsou ve výsledcích návrhu uvedeny všechny aspekty popisované v kapitole 6. Ovšem v kompletním návrhu se při realizaci dbalo na budoucí rozšíření o další prvky a přizpůsobil se tomu kód aktuálně dostupných prvků.

8.1 Obecné části zdrojového kódu

Vzhledem ke komplexnosti finálního kódu, který bude nasazen na závodech v roce 2024, bylo potřeba mít kód přehledný a snadno laditelný. Z toho důvodu byl vytvořen hlavičkový soubor „defines.h“, který obsahuje definice s názvy částí kódu. Díky těmto definicím lze pomocí komentování („//#define …” nebo „/*#define …*/“) zamezit spuštění dané části kódu. Pro snadné rozlišení začínají tyto definice předložkou „MY_“. Pro debugování jsou taktéž v hlavičkovém souboru definice „DEBUG“, které zajišťují výpisy na sériovou linku, díky kterým lze ověřit, zda jednotky naběhly úspěšně, případně ověřit jaké jsou hodnoty senzorů.

Na Obrázek 7 je začátek hlavního kódu ve kterém se vkládají knihovny a hlavičkové soubory. Knihovna „<Arduino_CAN.h>” zajišťuje fungování CAN 1, který je tak označen také na Portenta Breakout. Další použitou knihovnou je „<mbed.h>“, která je určená pro mbed procesory, které jsou na Portenta H7 Lite. Dále je používána knihovna „<Arduino_PortentaBreakout.h>“, která umožňuje práci s piny a dalšími náležitostmi, které obsahuje Portenta Breakout. Propojení mezi Portentou H7 Lite a Portentou Breakout zajišťují dva 80 pinové HD konektory. Pomocí knihovny „<Scheduler.h>“ je možné vytvořit další smyčky, které běží nezávisle na ostatních smyčkách v kódu a procesy se automaticky rozdělují mezi oba procesory na Portentě H7 Lite. Knihovny „<mcp_can.h>” a „<SPI.h>“ zjišťují komunikaci po CAN 2. Dále jsou zde vygenerované hlavičkové soubory z Arduino Cloudu pro bezdrátové posílání dat a knihovna „<Arduino_UnifiedStorage.h>“ pro práci s SD kartou.

Obrázek 7: Použité knihovny v kódu

```
#include "defines.h"

#include <Arduino_CAN.h>
#include <mbed.h>
#include <Arduino_PortentaBreakout.h>


#include <mcp_can.h>
#include <SPI.h>

#include <Scheduler.h>

#ifdef MY_ARDUINO_CLOUD
  #include "arduino_secrets.h"
  #include "thingProperties.h"
#endif

#ifdef MY_SD_CARD
  #include <Arduino_UnifiedStorage.h>
#endif

```

Zdroj: vlastní zpracování

Struktury kódu v Arduino IDE obsahuje v základu funkci setup, která proběhne jednou na začátku spuštění kódu. Tato funkce se nachází po vytvoření globálních proměnných. Příkaz „Serial.begin(115200);“ spustí komunikaci po sériové lince na rychlosti 115200 bps, tato sériová linka slouží pro výpisy při ladění kódu. Na dalším řádku je příkaz, který čeká 1500 ms, aby se stihla sériová komunikace rozběhnout.

8.2 Zdrojové kódy CAN

V kódu na Obrázek 8 jsou vytvořené globální proměnné pro práci s CAN 2. Tato část kódu je umístěna mezi vložením knihoven a funkcí setup. Proměnná CAN2 typu MCP_CAN je určená pro práci s CAN bus sběrnicí prostřednictvím SPI. V závorce je CS pin, který je na Portenta Breakout. Tento pin při logické nule aktivuje příjem a vysílání dat. Pole „data“ typu byte bylo využito pouze pro poslání testovací zprávy ve funkci pro budoucí použití. Velikost pole je staticky nastavena na 8, což odpovídá délce zprávy po CAN bus. Další proměnou je „rxId“, do které se pří čtení zprávy uloží

ID zprávy a podle něj lze zprávy dále třídit. Obdobně se ukládá do proměnné len délka zprávy a do pole „rxBuf“ přijatá data. Poslední proměnnou je pole pro ukládání teplot z teplotních senzorů kol. Jsou použity dva teplotní senzory a každý má 8 hodnot.

Obrázek 8: Globální proměnné pro CAN 2

```
62  // Globalní proměnné pro CAN 2
63  #ifdef MY_CAN2
64    MCP_CAN CAN2(SPI1_CS);
65    byte data[8] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};
66    long unsigned int rxId;
67    unsigned char len = 0;
68    unsigned char rxBuf[8];
69    int16_t cM_TEMPS[16];
70  #endif
71
```

Zdroj: vlastní zpracování

V kódu na Obrázek 9 je začátek komunikace pomocí CAN 1, nachází se ve funkci „void setup()“. Tato komunikace využívá knihovnu „<Arudino_CAN.h>“ a ve srovnání s inicializací CAN 2 na Obrázek 10 je patrné, že jsou oba kódy rozdílné, ale ve výsledku fungují oba velmi podobně. Konkrétně v inicializaci CAN 1 je nastavena rychlost na 1000 kbps a funkce „begin“ vrací v případě úspěšného spuštění komunikace „true“ a v případě selhání „false“. Na tyto návratové hodnoty reaguje podmínka „if“ a výpis na sériovou linku s informací o selhání funkce „begin“.

Obrázek 9: Inicializace pro CAN 1

```
76    // Inicializace pro CAN1
77    #ifdef MY_CAN1
78      if (!CAN.begin(CanBitRate::BR_1000k)) {
79        Serial.println("CAN.begin(...) failed.");
80      }
81    #endif
82
```

Zdroj: vlastní zpracování

Na Obrázek 10 je pro inicializaci komunikace CAN 2 využita knihovna „<mcp_can.h>”, která ve funkci „begin“ má tři volitelné parametry. První nastavuje mód přijímaných ID, druhý určuje rychlost, konkrétně 500 kbps, a třetí parametr určuje clockset krystalu v modulu a v tomto případě se jedná o 8 MHz. V této knihovně je

návratová hodnota v případě úspěšného spuštění komunikace „CAN_OK“. V případě této návratové hodnoty se na sériovou linku vypíše zpráva o úspěšné inicializaci MCP2515 modulu a v opačném případě zpráva o neúspěšné inicializaci.

Obrázek 10: Inicializace pro CAN 2

```
87    // Inicializace pro CAN2
88    #ifdef MY_CAN2
89      if(CAN2.begin(MCP_ANY, CAN_500KBPS, MCP_8MHZ) == CAN_OK) {
90        Serial.println("MCP2515 Initialized Successfully!");
91      } else {
92        Serial.println("Error Initializing MCP2515...");
93      }
94      CAN2.setMode(MCP_NORMAL);
95    #endif
96
```

Zdroj: vlastní zpracování

Na následujícím Obrázek 11 je kód s vytvořením vláken v podobě smyček pomocí knihovny „<Scheduler.h>“. Tento kód je ve funkci „void setup()“. Smyčka loop1 je určena pro CAN 1 a smyčka loop2 pro CAN 2.

Obrázek 11: Vytvoření smyček pro vícevláknové operace

```
105   // Vytvoření více smyček pro CAN1 a CAN2 (vícevláknová operace)
106   #ifdef MY_CAN1
107     Scheduler.startLoop(loop1);
108   #endif
109   #ifdef MY_CAN2
110     Scheduler.startLoop(loop2);
111   #endif
112
```

Zdroj: vlastní zpracování

Na Obrázek 12 je kód s použitím smyček pro funkce pracující s CAN. Tyto smyčky fungují stejně jako klasická smyčka „void loop()“ používaná ve většině Arduino programů. Na Obrázek 12 je vidět, že se v obou smyčkách spouští pouze příjem jednotlivých CAN. Funkce „CAN1_recive()“ je dále na Obrázek 13 a funkce „CAN2_recive()“ na Obrázek 14.

Obrázek 12: Smyčka pro CAN 1 a pro CAN 2

```
#ifdef MY_CAN1
  void loop1()
  {
    CAN1_recive();
  }
#endif
```

```
#ifdef MY_CAN2
  void loop2()
  {
    CAN2_recive();
  }
#endif
```

Zdroj: vlastní zpracování

Funkce „CAN1_recive()“ v případě, že jsou k dispozici data pro čtení, vrátí nenulovou hodnotu, čímž se splní podmínka a vytvoří se konstantní proměnná „incomingMsg“, do které se uloží přečtená data, a následně se tyto data vypíšou na sériovou linku. Dále je v této funkci připravený switch pro rozdělení zpráv podle jejich ID pro další zpracování.

Obrázek 13: Funkce pro příjímání zpráv po CAN 1

```
#ifdef MY_CAN1
  void CAN1_recive() {
    if (CAN.available()) {
      CanMsg const incomingMsg = CAN.read();
      Serial.println(incomingMsg );
      // Příprava pro budoucí zpracování zpráv
      switch (incomingMsg.id) {
        case 0X310:

          break;
        default:
          // Zpracování neznámých zpráv, pokud je třeba
          break;
      }
    }
  }
#endif
```

Zdroj: vlastní zpracování

Funkce „CAN2_recive()“ dělá v podstatě to stejné jako funkce „CAN1_recive()“, ale vzhledem k rozdílné knihovně se zdrojový kód liší v názvech používaných funkcí. V této funkci jsou již zpracována data z teplotních senzorů.

Obrázek 14: Funkce pro příjem zpráv po CAN 2

```
243   void CAN2_recive() {
244     if (CAN2.checkReceive() == CAN_MSGAVAIL)
245     {
246       CAN2.readMsgBuf(&rxId, &len, rxBuf);
247       switch (rxId)
248       {
249         case 848:
250           CAN2_processTemperature(0);
251           break;
252 
253         case 849:
254           CAN2_processTemperature(4);
255           break;
256 
257         case 784:
258           CAN2_processTemperature(8);
259           break;
260 
261         case 785:
262           CAN2_processTemperature(12);
263           break;
264       }
265     }
266   }
267 
```

Zdroj: vlastní zpracování

Na Obrázek 15 je funkce, do které se vkládá offset podle ID příchozí zprávy s daty. Ve funkci je smyčka „for“, ve které se zpracují a uloží data teplotního senzoru do pole na místo podle offsetu. Dále je v této funkci výpis uložených dat z pole, pro kontrolu správnosti při ladění kódu.

Obrázek 15: Funkce pro zpracování zpráv teplotních senzorů pneumatik

```
268   void CAN2_processTemperature(int offset) {
269     for(int i = 0; i < 8; i += 2)
270     {
271       // Spojení dvou bytů do jednoho 16-bitového čísla (teplota v 0,1 °C)
272       cM_TEMPS[i / 2 + offset] = ((rxBuf[i] << 8) | rxBuf[i + 1])-2000;
273     }
274     #ifdef DEBUG_TEMPERATURE
275       Serial.print("Received data:");
276       for(int i = offset; i < 4 + offset; i += 1)
277       {
278         Serial.print(" cM_TEMPS[");
279         Serial.print(i);
280         Serial.print("]: ");
281         Serial.print(cM_TEMPS[i]);
282       }
283       Serial.println();
284     #endif
285   }
286 #endif
```

Zdroj: vlastní zpracování

8.3 Mikro SD karta

Na Obrázek 16 jsou vytvořené globální proměnné pro práci s SD kartou. První proměnná je „sd“, pomocí které je možné se na SD kartu odkazovat. Druhou proměnnou je „document“, která funguje jako ukazatel na soubor na SD kartě. Dále je zde vytvořená proměnná pro uložení cesty k používanému souboru, a nakonec logická proměnná, do které se ukládá informace o otevření SD karty pro zápis.

Obrázek 16: Globální proměnné pro SD kartu

```
30  // Globalní proměnné pro mikro SD kartu
31  #ifdef MY_SD_CARD
32    SDStorage sd = SDStorage();
33    UFile document = UFile();
34    String path;
35    bool SDopened = false;
36  #endif
```

Zdroj: vlastní zpracování

Ve funkci „setup()“ je inicializována SD karta viz Obrázek 17 – na řádku 116 je kód pro debug výpisy z knihovny pro SD kartu, které jsou aktivovány v případě true. Dále je zde začátek komunikace s SD kartou a vytvoření cesty pro zápis a následně zapsání hlavičky do csv souboru. Dále je zde Nastaven pin z Portenta Breakout „CAMERA_D0N“ jako vstup, který simuluje budoucí odpojení napájení od formule, pro uložení dat na SD kartě.

Obrázek 17: Inicializace SD karty

```
113   // Inicializace SD karty
114   #ifdef MY_SD_CARD
115     #ifdef DEBUG_SD_CARD_3
116       Arduino_UnifiedStorage::debuggingModeEnabled = true;      // Debug výpisy z knih
117     #endif
118     sd.begin();     // Spuštění komunikace s SD kartou
119
120     Folder root = sd.getRootFolder();
121     path = root.getPathAsString() + "/" + name4SDcard() + ".csv";
122     document.open(path, FileMode::STORAGE_APPEND);
123     document.write(createHeader());
124     document.write("\n");
125     document.close();
126
127     if (Breakout.pinMode(CAMERA_D0N, INPUT_PULLUP) == -1) {   // Nastavení vypínacího
128       Serial.println("Error: pin not connected");
129     }
130   #endif
131 }
```

Zdroj: vlastní zpracování

Na Obrázek 18 je kód, který je v hlavní smyčce programu a zajišťuje zápis dat na SD kartu každých 200 ms, a v případě přizemnění pinu „CAMERA_D0N“ se data uloží, soubor se zavře a SD karta se softwarově odpojí. Poté se kód dostane do nekonečné smyčky, ve které čeká na restart.

Obrázek 18: Kód SD karty v hlavní smyčce programu

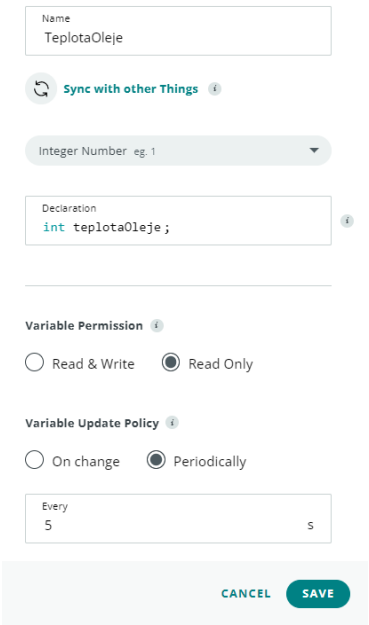
```
  #ifdef MY_SD_CARD
    if(!SDopened && Breakout.digitalRead(CAMERA_D0N) == 0) {
      SDopened = document.open(path, FileMode::STORAGE_APPEND);
      #ifdef DEBUG_SD_CARD_2
        Serial.println("SDopened");
      #endif
    } else if(SDopened) {
      #ifdef MY_RTC
        if(millis() - myTime >= TIMER) {
          myTime = millis();
          write2SDcard();
          #ifdef DEBUG_SD_CARD_2
            Serial.println("SDwrite");
          #endif
        }
      #else     //
        write2SDcard();
          #ifdef DEBUG_SD_CARD_2
            Serial.println("SDwrite");
          #endif
      #endif
    }
    if((Breakout.digitalRead(CAMERA_D0N) == 1) && SDopened) {
      document.close();
      sd.unmount();
      SDopened = false;
      #ifdef DEBUG_SD_CARD_2
        Serial.println("SDclosed");
      #endif
      while(1);   // Po odpojení vypínacího pinu čeká na restart v nekonečné smyčce
    }
  #endif
}
```

Zdroj: vlastní zpracování

8.4 Arduino Cloud

Pro vytváření jednotlivých proměnných bylo potřeba přidat tzv. „Things“. V bezplatné verzi je počet „Things“ omezen na 5. V rozhraní Arduino Cloud bylo zvoleno přidání pomocí tlačítka „ADD“ a nakonfigurování proměnné viz Obrázek 19.

Obrázek 19: Vytvoření proměnné v Arduino Cloud

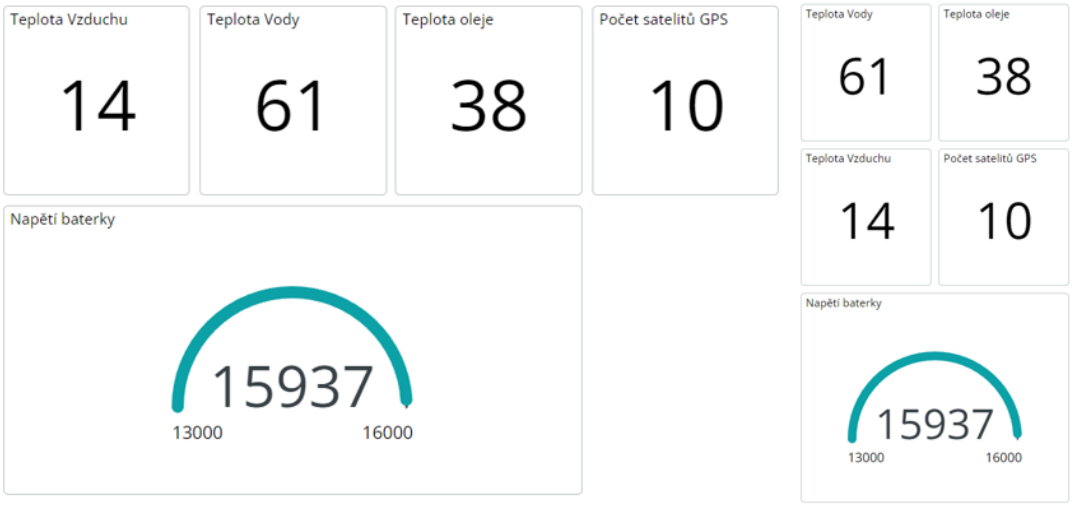


Zdroj: (19)

Dále bylo potřeba přidat zařízení Arduino Portenta H7 Lite do tzv. „Devices“, kliknutím na tlačítko „ADD DEVICE“. V okně byl zvolen „Arduino board“. Poté se objevilo okno, ve kterém bylo třeba připojit zařízení do počítače. Po krátké době bylo vyhledáno zařízení Arduino Portenta H7. Bylo zvoleno tlačítko „CONFIGURE“, které otevřelo okno, ve kterém byl zvolen název zařízení. Následně byl vybrán způsob připojení k síti, a to prostřednictvím ethernetu. Poté se nakonfigurovalo nastavení pro správnou komunikaci s Arduino cloudem. A nakonec přidávání zařízení se zobrazilo okno s informací, že bylo vše nastaveno.

Na Obrázek 20 je vlevo dashboard, pomocí kterého se zobrazují data živě prostřednictvím Arduino cloudu na PC zařízení. V aplikaci Arduino Cloud je taktéž dashboard pro mobilní zařízení, který je na Obrázek 20 vpravo. Data se v obou dashboardech zobrazují stejná, ale jejich rozmístění je přizpůsobeno právě využívanému zařízení.

Obrázek 20: Dashboardy v Arduino Cloudu



Zdroj: (19)

8.5 RTC

Pro práci s časem a jeho ukládání byly vytvořeny globální proměnné viz Obrázek 21. Globální proměnná „utcTime“ slouží pro zjištění času ve formátu utc, ze kterého je pomocí příkazů možné získat jednotlivé datové údaje, jako je rok, měsíc, den, hodina, minuta a sekunda. Pro ukládání aktuálního času je zde proměnná „timeRTC“, do které se v datovém typu „String“ ukládá aktuální čas a zapisuje se do prvního sloupce tabulky dat. Dále je zde proměnná „myTime“, která je použita pro časování, a definovaný „TIMER“, který určuje po kolika ms bude volán zápis do SD karty.

Obrázek 21: Globální proměnné pro práci s RTC

```
22  // Globalní proměnné pro zjišťování času z RTC
23  #ifdef MY_RTC
24    Timestamp utcTime;
25    String timeRTC;
26    unsigned long myTime;
27    #define TIMER 200
28  #endif
29
```

Zdroj: vlastní zpracování

Zdrojový kód na Obrázek 22 se nachází v hlavní smyčce programu a za podmínky, že není otevřena SD karta pro zápis, uloží do proměnné „myTime“ návratovou hodnotu

z funkce „millis()“. To je z toho důvodu, aby se při prvním zápisu po otevření zápisu do SD karty vědělo o počátku prvního zápisu a byl tak časovač od prvního zápisu v intervalu 200 ms. Dále je zde volána funkce „timeUTC();“.

Obrázek 22: Práce s RTC v hlavní smyčce programu

Zdroj: vlastní zpracování

Funkce „timeUTC();“ je vidět na Obrázek 23, ve které se uloží do proměnné „utcTime“ z Portenty Breakout RTC čas ve formátu UTC. Tato funkce dále v případě definovaného „DEBUG_UTC“ vypíše na sériovou linky právě hodnotu UTC a dále rok, měsíc, den, hodinu, minutu a sekundu. Poslední věcí, kterou tato funkce dělá, je uložení aktuálního času, tedy hodiny, minuty a sekundy ve vhodném formátu pro zápis na SD kartu do proměnné „timeRTC“.

Obrázek 23: Funkce pro zpracování času

Zdroj: vlastní zpracování

8.6 Akcelerometry

Akcelerometrům bylo potřeba v první řadě změnit rozsah. Tuto změnu rozsahu zajišťuje následující kód na Obrázek 24. Je zde využita knihovna „<Wire.h>“ která zajišťuje práci s I2C sběrnicí. Dále je v kódu definována adresa používaného akcelerometru. Ve funkci „void setup()“ je začátek komunikace po I2C sběrnici, začátek komunikace po sériové lince, samotné nastavení akcelerometru na ±4g a nastavení gyroskopu, kterým je modul MPU9050 taktéž vybaven, na rozsah ±500 °/s. Toto nastavení bylo provedeno na prototypové verzi telemetrie, kde byla použita vývojová deska Arduino Due. U této verze bylo dosaženo limitu u CAN sběrnic, kde se kvůli neschopnosti více vláknových operací komunikace zasekávala, a zprávy tak nebyly pro dané použití použitelné.

Obrázek 24: Kód pro nastavení rozsahu akcelerometrů na ±4 g

```
#include <Wire.h>

#define MPU9050_ADDR 0x68 // Adresa akcelerometru MPU-9050

void setup() {
  Wire.begin();
  Serial.begin(9600);

  // Nastavení rozsahu měření pro akcelerometr na ±4g
  Wire.beginTransmission(MPU9050_ADDR);
  Wire.write(0x1C); // Adresa registru ACCEL_CONFIG
  Wire.write(0x08); // Hodnota pro ±4g rozsah
  Wire.endTransmission();

  // Nastavení rozsahu měření pro gyroskop na ±500 stupňů za sekundu
  Wire.beginTransmission(MPU9050_ADDR);
  Wire.write(0x1B); // Adresa registru GYRO_CONFIG
  Wire.write(0x08); // Hodnota pro ±500 °/s rozsah
  Wire.endTransmission();
}

void loop() {
}
```

Zdroj: vlastní zpracování

9 Vlastní doporučení

Během práce bylo zjištěno, že vybraná vývojová deska disponuje pouze jedním CAN bus převodníkem, proto by bylo pro jednoduchost výsledného řešení vhodnější využít místo vývojové desky Arduino Portenta H7 Lite vývojovou desku Arduino Portenta X8, která zároveň disponuje vyšším výpočetním výkonem a bude proto vhodná i pro budoucí přesun do kategorie EV. Dále by bylo vhodnější psát práci pro hotové řešení, které odjelo sezónu na formuli. Z toho důvodu práce obsahuje kompletní návrh, ale realizace bohužel kompletní není. Dále by bylo vhodné zhotovit aplikaci pro zpětné zpracování dat pro jejich snazší analýzu a lepší zpětnou vazbu pro piloty. Pokud by bylo potřeba zobrazovat více dat živě, lze jednoduše zaplatit předplatné a přidat další proměnné, které budou potřeba. V případě, že by tým potřeboval ještě přesnější GPS data, tak bych doporučil postavit RTK base station, se kterou by tento GPS modul měl dosahovat přesnosti až 1,5 cm, ovšem se jedná o další náklady.

10 Závěr

Bakalářská práce na téma „Modernizace telemetrie palubní jednotky Formule Student“ se zabývala návrhem vhodného řešení pro tým CULS Prague formula racing v oblasti zpracování a ukládání telemetrických dat monopostu FS09 pro sezónu 2024. Cílem byl kompletní návrh spolehlivého řešení a realizace dostupné části navrženého řešení.

V teoretické části byl analyzován současný stav telemetrie a následně specifikované požadavky, které tým měl. Dále se teoretická část práce zabývala průzkumem možných řešení pro spolehlivé ukládání dat, živého zobrazení dat, GPS technologií, zobrazení stavů jednotek a komunikačními protokoly.

V praktické části byl nejprve zhotoven kompletní návrh modernizace telemetrie. A poté byla zrealizovaná dostupná část tohoto návrhu v době psaní této práce. Závody s monopostem FS09 se totiž konají 4 měsíce po odevzdání této práce, z toho důvodu nebyly některé díly a komponenty, které bude telemetrie z návrhu modernizace využívat, dostupné a ve výsledcích byly zakomponovány do kódů a testování pouze dostupné jednotky. Nicméně byl kód tvořen tak, aby se v budoucnu snadno rozšířil o navržené prvky, až budou připraveny.

Jako spolehlivé řešení pro ukládání dat bylo zvoleno ukládání na mikro SD kartu. Hlavním komunikačním kanálem jsou dvě CAN bus sběrnice. Pro živé zobrazení dat byl zvolen Arduino Cloud, na který byla jednotka telemetrie připojena pomocí LTE routeru a ethernetového kabelu. Realizované části fungovaly spolehlivě, ale velká část jednotek nebyla dostupná. Telemetrie však byla navrhnuta s dostatečným výkonem pro všechny potřebné operace.

Lze konstatovat, že tato práce bude sloužit týmu CULS Prague formula racing pro sezónu 2024 a její obdoba velmi pravděpodobně i v dalších letech. Data, která díky navržené telemetrii tým získá, budou důležitá pro návrh všech dalších monopostů.

11 Seznam použitých zdrojů

(1) BRADÁČ, Z., FIEDLER, P. a KAČMÁŘ, M.: *Bezdrátové komunikace v automatizační praxi I: historie a současnost*. Automa. 2003, roč. 9, č. 5, ISSN 1210-9592.

(2) BRADÁČ, Z., FIEDLER, P. a KAČMÁŘ, M.: *Bezdrátové komunikace v automatizační praxi II: standard Bluetooth*. Automa. 2003, roč. 9, č. 7, ISSN 1210-9592.

(3) BRADÁČ, Z., FIEDLER, P. a KAČMÁŘ, M., lektoroval: ČAPEK, J.: *Bezdrátové komunikace v automatizační praxi III: standard IEEE 802.11 (část 2)*. Automa, 2003, roč. 9, č. 12, ISSN 1210-9592.

(4) BRADÁČ, Z., FIEDLER, P. a KAČMÁŘ, M., lektoroval: ČAPEK, J.: *Bezdrátové komunikace v automatizační praxi IV: Datové přenosy v GSM – GPRS*. Automa, 2004, roč. 10, č. 1, ISSN 1210-9592.

(5) BRADÁČ, Z., FIEDLER, P. a KAČMÁŘ, M.: *Bezdrátové komunikace v automatizační praxi VII Další standardy*. Automa, 2004, roč. 10, č. 7, s. 44-46, ISSN 1210-9592.

(6) VALDEZ, J. a BECKER J.: *Understanding the I2C Bus*. [online]. Texas Instruments Incorporated, 1995 [cit.2024-1-23]. Dostupné z: https://www.ti.com/lit/an/slva704/slva704.pdf

(7) SENEVIRATNE, P.: *Internet of Things with Arduino Blueprints*. Packt Publishing, Limited. 2015. ISBN 978-1785285486.

(8) ŠVÉDA, M. a TRCHALÍK, R.: *ZigBee-to-Internet Interconnection Architectures*, In: Proceedings of the Second International Workshop on Mobile Communications and Learning MCL 2007, Saint Luce, Martinique, MQ, IEEE CS, 2007, s. 6, ISBN 0-7695-2807-4.

(9) TP-LINK. *Parametry: TL-MR100*. [online]. [cit. 2023-12-02]. Dostupné z: https://www.tp-link.com/cz/home-networking/3g-4g-router/tl-mr100/#specifications

(10) *Pravidla Formula student*. In: Formulastudent.de [online]. Německo: Formula student Germany, 2024 [cit. 2023-12-06]. Dostupné z: https://www.formulastudent.de/fileadmin/user_upload/all/2024/rules/FS-Rules_2024_v1.1.pdf

(11) *Archiv týmu CULS Prague formula racing*

(12) Datasheet pro GPS modul RTK-DUAL-B od LOCOSYS. *RTK-DUAL_datasheet_v0.2* [online]. [cit. 2024-1-20]. Dostupné z: https://www.neven.cz/user/7262/upload/stuff/files/27121072-rtk-dual_datasheet_v0.2.pdf

(13) Arduino. *Arduino Portenta H7 Lite* [online]. [cit. 2024-1-22]. Dostupné z: https://store.arduino.cc/products/portenta-h7-lite

(14) GUADALUPI A.: *Dokumantace Arduino Portenta H7 Lite* [online]. [cit. 2024-1-23]. Dostupné z: https://content.arduino.cc/assets/PortentaH7_Lite%20Schematics.pdf

(15) TARABA, R.: *Aplikování sběrnice CAN* [online]. vyvoj.hw.cz 2004-11-9 [cit. 2024-1-23]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/rozhrani/aplikovani-sbernice-can.html

(16) *Serial Peripheral Interface (SPI)*. [online]. SparkFun. 2017 [cit. 2024-01-23]. Dostupné z: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all
(17) *Quick Installation Guide – further information*. [online]. INSYS icom. [cit. 2024-02-09]. Dostupné z: https://www.insys-icom.com/en/qig/
(18) INSYS icom. *Webové rozhraní routeru*. [online]. INSYS icom. [cit. 2024-02-09]. Dostupné z: https://192.168.1.1
(19) *Arduino cloud*. [online]. Arduino. [cit. 2024-02-09]. Dostupné z: https://id.arduino.cc/
(20) *Technical Data Sheet- MIRO, MIROdul*. [online]. INSYS icom. [cit. 2024-02-11]. Dostupné z: https://public.centerdevice.de/e73f4e80-eeae-4050-91b3-1cf1c4da90cc?embedded=true
(21) CONROY, SOUSA a WEST: *Data sheet Magma X*. [online]. Taoglas, 2023. [cit. 2024-02-11]. Dostupné z: https://www.taoglas.com/datasheets/AA.178.301111.pdf
(22) *Datasheet Portenta Breakout Board.* [online]. Arduino. [cit. 2024-03-11]. Dostupné z: https://docs.arduino.cc/resources/datasheets/ASX00031-datasheet.pdf

12 Seznam obrázků a tabulek

13 Seznam použitých zkratek

APN	Access Point Name
CAN	Controller Area Network
CEP	Circular Error Probability
CS	Chip select
DHCP	Dynamic Host Configuration Protocol
EDR	Engineering Design Report
EV	Electric vehicle
FS	Formula Student
GPS	Global Positioning Systém
GSM	Groupe Spécial Mobile
HD	High Density
HV	Hight voltage
I2C	Inter-Integrated Circuit
ID	Identity
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
LED	Light Emitting Diode
Li-Po	Lithium Polymer
LTE	Long-Term Evolution
LV	Low voltage
MAC	Media Access Control
MEMS	MicroElectroMechanical Systems

NMEA	National Marine Electronics Association
NTP	Network Time Protocol
PICO	Peripheral In / Controller Out
PIN	Personal Identification Number
POCI	Peripheral Out / Controller In
RTC	Real Time Clock
RTK	Real Time Kinematic
SCK	Serial Clock
SCL	Serial Clock line
SD	Secure Digital
SDA	Serial Data line
SIM	Subscriber Identity/Identification Module
SMA	SubMiniature version A
SPI	Serial Peripheral Interface
VPN	Virtual Private Network
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network

14 Seznam příloh

Příloha 1: Zdrojový kód Telemtrie.ino

```
#include "defines.h"

#include <Arduino_CAN.h>
#include <mbed.h>
#include <Arduino_PortentaBreakout.h>


#include <mcp_can.h>
#include <SPI.h>

#include <Scheduler.h>

#ifdef MY_ARDUINO_CLOUD
  #include "arduino_secrets.h"
  #include "thingProperties.h"
#endif

#ifdef MY_SD_CARD
  #include <Arduino_UnifiedStorage.h>
#endif

// Global variable for save time from RTC
#ifdef MY_RTC
  Timestamp utcTime;
  String timeRTC;
  unsigned long myTime;
  #define TIMER 180
#endif

// Global variable for SD card
#ifdef MY_SD_CARD
  SDStorage sd = SDStorage();
  UFile document = UFile();
  String path;
  bool SDopened = false;

#endif


// Global variable for GPS data
#ifdef MY_GPS
  float GPS_Latitude;
  float GPS_Longitude;
  float GPS_Speed;
  int GPS_Satelites;
  int GPS_UTC_year;
  int GPS_UTC_month;
  int GPS_UTC_day;
  int GPS_UTC_hour;
  int GPS_UTC_minute;
  int GPS_UTC_second;
  float GPS_ACC_X;
  float GPS_ACC_Y;
  float GPS_ACC_Z;
#endif

#ifdef MY_DISPLAY
```

```
  unsigned long previousMillis_display_pomaly = 0; // Display
  int interval_display_pomaly = 2000;
#endif

// Global variables for CAN 1

// Global variables for CAN 2
#ifdef MY_CAN2
  MCP_CAN CAN2(SPI1_CS);
  byte data[8] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};
  long unsigned int rxId;
  unsigned char len = 0;
  unsigned char rxBuf[8];
  int16_t cM_TEMPS[16];
#endif


void setup() {
  Serial.begin(115200);
  delay(1500);    // while(!Serial);

  // Init CAN1
  #ifdef MY_CAN1
    if (!CAN.begin(CanBitRate::BR_500k)) {
      Serial.println("CAN.begin(...) failed.");
    }
    //CAN.watchForRange(0x310, 0x351);
  #endif

  // Init CAN2
  #ifdef MY_CAN2
    if(CAN2.begin(MCP_ANY, CAN_500KBPS, MCP_8MHZ) == CAN_OK)
Serial.println("MCP2515 Initialized Successfully!");
    else Serial.println("Error Initializing MCP2515...");
    CAN2.setMode(MCP_NORMAL);   // Change to normal mode to allow
messages to be transmitted
  #endif

  // Init RTC time
  #ifdef MY_RTC
    timeUTC();
  #endif

  // Definition variables from thingProperties.h for Arduino Cloud
  #ifdef MY_ARDUINO_CLOUD
    initProperties();

    // Conneect to Arduino IoT Cloud
    ArduinoCloud.begin(ArduinoIoTPreferredConnection);
    setDebugMessageLevel(2);
    ArduinoCloud.printDebugInfo();
  #endif

  // Create more loops for CAN1 and CAN2
  #ifdef MY_CAN1
    Scheduler.startLoop(loop1);
  #endif
  #ifdef MY_CAN2
    Scheduler.startLoop(loop2);
```

```
  #endif

  // SD card
  #ifdef MY_SD_CARD
    //String header = createHeader();
    Arduino_UnifiedStorage::debuggingModeEnabled = true;
//debug sd card
    sd.begin();     // Mounting sd card

    Folder root = sd.getRootFolder();
    path = root.getPathAsString() + "/" + name4SDcard() + ".csv";
    document.open(path, FileMode::STORAGE_APPEND);
    document.write(createHeader());
    document.write("\n");
    document.close();

    if (Breakout.pinMode(CAMERA_D0N, INPUT_PULLUP) == -1) {
      Serial.println("Error: pin not connected");
    }
  #endif
}

/*-----------------------------------------looop------------------
------------------------*/
void loop() {
  #ifdef MY_ARDUINO_CLOUD
    ArduinoCloud.update();
    arduinoCloudTest();
  #endif

  #ifdef MY_RTC
    myTime = millis();
    timeUTC();
  #endif

  #ifdef MY_SD_CARD
    if(!SDopened && Breakout.digitalRead(CAMERA_D0N) == 0) {
      SDopened = document.open(path, FileMode::STORAGE_APPEND);
      Serial.println("SDopened");
    } else if(SDopened) {
      write2SDcard();
      Serial.println("SDwrite");
    }
    if((Breakout.digitalRead(CAMERA_D0N) == 1) && SDopened) {
      document.close();
      sd.unmount();
      SDopened = false;
      Serial.println("SDclosed");
      while(1);
    }
  #endif
  #ifdef MY_RTC
    while(millis() - myTime <= TIMER){
      //Serial.println(millis() - myTime);
    }
  #else
    delay(1000);
  #endif
}
```

```
#ifdef MY_CAN1
  void loop1()
  {
    CAN1_recive();
  }
#endif

#ifdef MY_CAN2
  void loop2()
  {
    CAN2_recive();
  }
#endif

#ifdef MY_RTC
  void timeUTC() {
    utcTime = Breakout.RTClock.getTime();
    #ifdef DEBUG_UTC
      Serial.println(utcTime.getUnixTimestamp());
      Serial.print("year: ");
      Serial.println(utcTime.year() + 1900);
      Serial.print("month: ");
      Serial.println(utcTime.month() + 1);
      Serial.print("day: ");
      Serial.println(utcTime.day());
      Serial.print("hours: ");
      Serial.println(utcTime.hour() + 1);
      Serial.print("minutes: ");
      Serial.println(utcTime.minute());
      Serial.print("seconds: ");
      Serial.println(utcTime.second());
    #endif
    timeRTC = String(utcTime.hour() + 1) + ":" +
String(utcTime.minute()) + ":" + String(utcTime.second());
  }
#endif

#ifdef MY_ARDUINO_CLOUD
  void arduinoCloudTest() {
    batteryLV = 15937;
    airTemperature = utcTime.second();
    satelites = 10;
    waterTemperature = 61;
    oilTemperature = 38;
    delay(1000);
  }
#endif

#ifdef MY_CAN2
  void CAN2_transmit() {
    byte sndStat = CAN2.sendMsgBuf(0x100, 0, 8, data);
    if(sndStat == CAN_OK)
    {
      #ifdef DEBUG_CAN2_TRANSMIT
        Serial.println("Message Sent Successfully!");
      #endif
    } else {
      #ifdef DEBUG_CAN2_TRANSMIT
        Serial.println("Error Sending Message...");
```

```
      #endif
    }
  }

  void CAN2_recive() {

    //if(!digitalRead(CAN0_INT))                         // If
CAN0_INT pin is low, read receive buffer
    if (CAN2.checkReceive() == CAN_MSGAVAIL)
    {
      CAN2.readMsgBuf(&rxId, &len, rxBuf);
      switch (rxId)
      {
        case 848: //CAN2_processTemperature(0)
          CAN2_processTemperature(0);
          break;

        case 849: //CAN2_processTemperature(4)
          CAN2_processTemperature(4);
          break;

        case 784: //CAN2_processTemperature(8)
          CAN2_processTemperature(8);
          break;

        case 785: //CAN2_processTemperature(12)
          CAN2_processTemperature(12);
          break;
      }
    }

  }

  void CAN2_processTemperature(int offset) {
    for(int i = 0; i < 8; i += 2)
    {
      // Spojení dvou bytů do jednoho 16-bitového čísla
      cM_TEMPS[i / 2 + offset] = ((rxBuf[i] << 8) | rxBuf[i + 1])-
2000;
    }
    #ifdef DEBUG_TEMPERATURE
      Serial.print("Received data:");
      for(int i = offset; i < 4 + offset; i += 1)
      {
        Serial.print(" cM_TEMPS[");
        Serial.print(i);
        Serial.print("]: ");
        Serial.print(cM_TEMPS[i]);
      }
      Serial.println();
    #endif
  }
#endif

#ifdef MY_CAN1
  void CAN1_recive() {
    if (CAN.available()) {  // přidat podmínku pro vytřídění jen
zpráv které chceme
      CanMsg const incomingMsg = CAN.read();
```

```
      //Serial.println(incomingMsg );
      switch (incomingMsg.id) {
        case 0X310: //zmenit id

          break;
        default:
          // Zpracování neznámých zpráv, pokud je třeba
          break;
      }
    }
  }
#endif

#ifdef MY_SD_CARD
  void write2SDcard() {
    document.write(timeRTC);
    document.write(";");
    String temp;
    for(int i = 0; i < 16; i++) {
      temp += String(cM_TEMPS[i]) + ";";
    }
    document.write(temp);
    document.write("\n");
  }

  String name4SDcard() {
    utcTime = Breakout.RTClock.getTime();
    String name = String(utcTime.year() + 1900) +"-"+
String(utcTime.month() + 1) + "-" +
    String(utcTime.day()) + "_" + String(utcTime.hour() + 1) + "-" +
String(utcTime.minute());
    #ifdef DEBUG_SD_CARD
      Serial.println(name);
    #endif
    return name;
  }

  String createHeader() {
    String header;
    #ifdef MY_RTC
      header += "timeRTC;";
    #endif
    #ifdef MY_CAN1
    #endif
    #ifdef MY_CAN2
      for(int i = 0; i < 16; i++) {
        header += "cM_TEMPS" + String(i) + ";";
      }
    #endif
    #ifdef MY_GPS
    #endif

    Serial.print(header);
    return header;
  }
#endif
```

Zdroj: vlastní zpracování

Příloha 2: Zdrojový kód hlavičkového souboru defines.h

```
// my pices of code
#define MY_ARDUINO_CLOUD
#define MY_CAN1
#define MY_CAN2
#define MY_SD_CARD
#define MY_DISPLAY
#define MY_RTC
#define MY_GPS

// debugs printers
//#define DEBUG     // Comment in final to skip all serial prints
#ifdef DEBUG
  //#define DEBUG_UTC // Comment for skip serial prints of UTC
  //#define DEBUG_TEMPERATURE
  #define DEBUG_CAN2_TRANSMIT
  #define DEBUG_SD_CARD
#endif
```

Zdroj: vlastní zpracování

Příloha 3: Zdrojový kód hlavičkového souboru thingProperties.h

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char IP[]      = SECRET_OPTIONAL_IP;
const char DNS[]     = SECRET_OPTIONAL_DNS;
const char GATEWAY[] = SECRET_OPTIONAL_GATEWAY;
const char NETMASK[] = SECRET_OPTIONAL_NETMASK;


float batteryLV;
int airTemperature;
int oilTemperature;
int satelites;
int waterTemperature;

void initProperties(){

  ArduinoCloud.addProperty(batteryLV, READ, 5 * SECONDS, NULL);
  ArduinoCloud.addProperty(airTemperature, READ, 5 * SECONDS, NULL);
  ArduinoCloud.addProperty(oilTemperature, READ, 5 * SECONDS, NULL);
  ArduinoCloud.addProperty(satelites, READ, 5 * SECONDS, NULL);
  ArduinoCloud.addProperty(waterTemperature, READ, 5 * SECONDS,
NULL);

}

EthernetConnectionHandler ArduinoIoTPreferredConnection(IP, DNS,
GATEWAY, NETMASK);
```

Zdroj: vlastní zpracování

Příloha 4: Zdrojový kód souboru sketch.json

```
{
  "cpu": {
    "fqbn": "arduino:mbed_portenta:envie_m7",
    "name": "Arduino Portenta H7",
    "type": "serial"
  },
  "secrets": [
    {
      "name": "SECRET_OPTIONAL_DNS",
      "value": ""
    },
    {
      "name": "SECRET_OPTIONAL_GATEWAY",
      "value": ""
    },
    {
      "name": "SECRET_OPTIONAL_IP",
      "value": ""
    },
    {
      "name": "SECRET_OPTIONAL_NETMASK",
      "value": ""
    }
  ],
  "included_libs": []
}
```

Zdroj: vlastní zpracování