Masarykova univerzita
Fakulta informatiky

Systém automatické kalibrace modelového železničního vozidla

Bakalářská práce

Jan Horáček

Brno, podzim 2018

Masarykova univerzita
Fakulta informatiky

Systém automatické kalibrace modelového železničního vozidla

Bakalářská práce

Jan Horáček

Brno, podzim 2018

Na tomto místě se v tištěné práci nachází oficiální podepsané zadání práce a prohlášení autora školního díla.

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Jan Horáček

Vedoucí práce: RNDr. Zdeněk Matěj, Ph.D.

Poděkování

Má cesta k této práci začala před devíti lety, kdy jsem si uvědomil, že modely, s kterými si hraji, a programování, které jsem se začal učit, mají cosi společného. Za asistence mého mentora Michala Petrilaka jsem v Klubu modelářů železnic Brno I začal programovat řídicí systém modelového kolejiště, z kterého se později stal plnohodnotný zabezpečovací systém v současné době nasazený na kolejištích výše zmíněného klubu. Vždy mě fascinovalo, že několik příkazů zadaných do počítače dokáže způsobit, že se něco pohne ve skutečném světě. Děkuji vedoucímu své práce, Zdeňku Matějovi, že mi umožnil skloubit můj koníček a mé studium. Děkuji všem, kteří mi pomohli s korekturami programů a textů v rámci této práce vytvořených: Vladimíru Štillovi, Janu Mrázkovi, Janu Koniarikovi, Janu Tušilovi, Janě a Miroslavu Horáčkovým, Zuzaně Novákové, Lucii Karmové a dalším.

Abstrakt

S příchodem moderních počítačově řízených modelových kolejišť vyvstávají mnohé problémy spojené s automatizací provozu. Jedním z těchto problémů je synchronizace rychlostí a brzdných křivek digitálně řízených hnacích vozidel, která je nutná pro bezpečný a bezkolizní automatický provoz na kolejišti. Tato práce implementuje jeden z možných přístupů k řešení tohoto problému: navrhuje vlastní detektor pro měření rychlosti modelu – tzv. měřicí vůz. Měření rychlosti je založeno na optickém senzoru, komunikace s počítačem probíhá pomocí protokolu Bluetooth. Naměřená data jsou analyzována a následně je navrhnut způsob postprocessingu odstraňující z těchto dat chyby. Nad technologií vozu tato práce vytváří software, který na základě změřené rychlosti v reálném čase upravuje parametry dekodéru hnacího vozidla tak, aby bylo dosaženo nastavení žádané rychlosti. Software byl navržen jako desktopová okenní aplikace ve frameworku `Qt` s důrazem na robustnost a snadnou použitelnost uživatelem. Vyvinuté technologie výrazně zpřesnily a zkrátily proces kalibrace, čímž přispěly k bezkoliznímu a efektivnějšímu provozu na modelových kolejištích.

Klíčová slova

kalibrace, dcc, vlak, vůz, model, senzor, C++, qt, bluetooth, cv

Obsah

1 Úvod **2**
1.1 *Modelová kolejiště* 2
1.2 *Současné trendy v řízení kolejiště* 3
1.3 *Synchronizace rychlosti vozidel* 4
1.3.1 Současné řešení problému 5
1.3.2 Řešení problému v hJOP 5
1.4 *Synchronizace brzdných křivek vozidel* 6
1.5 *Cíl práce* 7

2 Požadavky na řešení **9**

3 Existující řešení **11**
3.1 *Statické bodové detektory* 11
3.2 *Statické úsekové detektory* 12
3.3 *Podvalníkové detektory* 13
3.4 *Kalibrační SW* 13
3.5 *Shrnutí* 14

4 Měřicí vůz **15**
4.1 *Umístění detektoru* 15
4.2 *Senzor* 16
4.2.1 Magnetický senzor 16
4.2.2 Magnetický senzor „cykloměřič“ 17
4.2.3 Optický senzor 17
4.2.4 Optozávora 18
4.3 *Napájení* 20
4.4 *Přenos dat do PC* 21
4.5 *Platforma* 22
4.5.1 ESP-32 22
4.5.2 ATmega + HC-05 23
4.6 *Mechanika* 24
4.7 *Elektronika* 25
4.8 *Princip měření* 27
4.9 *Firmware* 29
4.10 *Prototyp vozu* 30

4.11 *Naměřená data* 32
4.11.1 Kompenzace chyb 37

5 Software **39**
5.1 *Volba nástrojů* 39
5.2 *WSM Speed Reader* 40
5.3 *Automatic Calibration* 42
5.3.1 Měření rychlosti 45
5.3.2 Fáze kalibrace rychlosti 46
5.3.3 Kalibrace dojezdů 48
5.4 *XpressNET knihovna* 49

6 Závěr **50**
6.1 *Možná rozšíření* 50

Bibliografie **52**

A Přílohy **55**

Seznam použitých zkratek

BEMF Back Electro-motive force, technologie pro měření rychlosti ovládaného motoru. 11, 34, 35, 46

BT Bluetooth. 23, 31

CV Configuration Value, jedna konfigurační jednotka s rozsahem hodnot jeden byte (0–255). Každé CV má svoje číslo (typicky 1–1023). 4, 6, 42, 43, 45, 46, 48

DCC Digital Command Control, digitální systém řízení modelového kolejiště. 3, 15, 21, 39, 42, 43

ICP Input Capture Pin, speciální pin procesoru ATmega328p určený k tomu, aby přesně a rychle měřil periodu vstupního signálu. 26, 30

JOP Jednotné obslužné pracoviště, standard grafického rozhraní počítačového řízení skutečné železnice definovaný Správou železniční a dopravní cesty. 3

LDO Low-dropout regulator, lineární regulátor napětí s velice nízkým dropout napětím (řádově malé desetiny voltu). 26

LT Long-term measure, režim měření rychlosti měřicím vozem, kdy je rychlost měřena po delší časový úsek. 45, 47

POM Programming on Main, režim programování dekodérů DCC, kdy je dekodér programován na plného provozu na kolejišti bez nutnosti přepínání do programovacího režimu. 43, 44

SPP Serial Port Profile, profil komunikačního standardu Bluetooth. 22, 23

WSM Obchodní název vyvinutého měřicího vozu: *WSM: A Wireless Speedometer*. 40–42, 44

1 Úvod

Současná modelová kolejiště již dávno nejsou jen hračkou malých dětí a jejich rodičů. Jsou to plnohodnotné modely, na kterých se dispečeři skutečné železnice učí řídit dopravu a které slouží pro výuků oborů automatizace. Kolejiště jsou vybavena robustní a spolehlivou zabezpečovací technikou podobnou té, která se v současnosti používá na skutečné železnici.

S růstem rozměrů kolejišť a cen modelů vzniká nutnost procesy řízení automatizovat, předejít chybám způsobeným lidským faktorem a řízení kolejiště obecně zefektivnit. Cílem této práce je pomocí automatizace procesu kalibrace modelového železničního vozidla přispět právě ke zvýšení bezpečnosti a spolehlivosti provozu.

Autor rozebere možnosti řešení problému automatické kalibrace, ukáže, proč jsou současná řešení nevhodná, a nakonec navrhne (1) hardware, (2) firmware a (3) software, kterými uvedený problém řeší. Autor popíše výstupy této práce a ukáže její reálný přínos.

Tuto práci je tematicky možné zařadit jak do kategorie *proof of concept*, tak i do kategorie práce *aplikační*. Jejím primárním cílem je totiž vytvořit reálné řešení reálného problému v oblasti automatizace, který doteď ještě nikdo neřešil. Konkrétně se práce zaměřuje na automatizaci řízení dopravy na modelovém kolejišti. Cílem této práce je navrhnout kvalitní řešení od hardwaru, přes firmware až po software.

Pro přesnou formulaci problému automatické kalibrace železničního vozidla je nutné pochopit kontext této práce, o kterém pojednává tato kapitola.

1.1 Modelová kolejiště

Modelové kolejiště je model železniční tratě, typicky zmenšený v některém ze standardních měřítek, např. v měřítku „TT“ (1:120) či „H0“ (1:87). Vedle typického *hobby* využití kolejiště často slouží jako dopravní trenažéry pro výuku zabezpečovacích zařízení železnice či

dalších aspektů využívaných na skutečné železnici. Nejen v těchto případech vzniká potřeba kolejiště ovládat *zabezpečovacím zařízením*. Typickým příkladem současného zabezpečovacího zařízení na skutečné železnici je *elektronické stavědlo* spolu s grafickou nadstavbou *JOP* (*Jednotné obslužné pracoviště*), která umožňuje řízení stavědla přes počítač.

Obdobný systém je využíván také v *Klubu modelářů železnic Brno I*. V tomto klubu je nasazen systém řízení kolejišť zvaný *hJOP* [1], který je klonem JOP upraveným pro potřeby řízení modelu. Jeho primárním cílem je řídit jízdu vlaku a omezením lidského faktoru předejít možným kolizím.

1.2 Současné trendy v řízení kolejiště

K řízení modelových kolejišť je v současné době napříč zeměmi využívána celá řada dostupných SW. Typickými příklady jsou například *JMRI* [2], *TrainController* [3] nebo český SW *modelJOP* [4].

Výše zmíněné aplikace potřebují komunikovat s hardwarovými prvky kolejiště. Nejdůležitějším z těchto prvků je digitální centrála DCC. Digitální centrála je obecný pojem, konkrétními instancemi od jednotlivých výrobců jsou pak např. *Z21* [5], *NanoX* [6] nebo třeba *Intellibox* [7]. Centrála nejčastěji komunikuje s počítačem pomocí sběrnice *RS232 over USB*. Jejím hlavním úkolem je vytvářet *digitální signál DCC* [8], který moduluje do kolejí.

Lokomotiva stojící na kolejích tento signál dekóduje pomocí speciální součástky – tzv. *dekodéru DCC* – a podle přijatých dat poveluje periferie lokomotivy: motor, světla nebo třeba zvuk vydávaný vestavěným reproduktorem. Modulovaný výkonový signál v kolejích je zároveň využíván jako (1) nosič informace pro dekodéry a (2) po usměrnění jako napájecí napětí pro periferie, včetně výkonově nejnáročnější periferie – motoru. Dekodér v podstatě obsahuje jen jednoduchý mikrokontrolér a několik výkonových prvků. Jeho typická velikost je v řádu malých jednotek cm^2, takže je skutečně malý.

Každý dekodér má svoji adresu v rozsahu 1–9999. Počítač pak může vydat příkaz pro řízení lokomotivy, např. *„lokomotivo s adresou 562, jeď směrem vpřed rychlostí 15!“*

Každý dekodér si ukládá svá konfigurační data do volatilní paměti formou tzv. *CV* (*Configuration Value*). *CV* je jedna konfigurační jednotka s rozsahem hodnot jeden byte (0–255). Každé *CV* má svoje číslo (typicky 1–1023). Výrobce dekodéru (potažmo standardizační organizace) pak definuje význam jednotlivých CV. Například CV #17 a #18 jsou vyhrazena k uložení adresy dekodéru [9].

1.3 Synchronizace rychlosti vozidel

Rychlost lokomotivy typicky bývá přirozené číslo v rozsahu 0–28. Tato hodnota se označuje jako tzv. *jízdní stupeň*.

Protože každá lokomotiva obsahuje jiné mechanické prvky (převody, motor), odpovídá stejnému jízdnímu stupni u dvou různých nezkalibrovaných hnacích vozidel různá rychlost. Ukazuje se, že při řízení kolejiště je vhodné, aby (1) stejnému jízdnímu stupni u všech hnacích vozidel na kolejišti odpovídala stejná rychlost a aby (2) byl jízdní stupeň pevně spárovaný s reálnou rychlostí vozidla přepočtenou dle měřítka.

Synchronizací rychlosti vozidel myslíme právě splnění těchto dvou vlastností pro všechny lokomotivy na kolejišti. Proces synchronizace rychlostí je součásti procesu *kalibrace lokomotivy*.

Motivací pro tyto kroky je především to, abychom dosáhli modelové věrnosti a aby se vozidlo po přepočtu pohybovalo *modelovou rychlostí*. Je také nutné dbát na provozní parametry kolejiště, které umožňují průjezd některými částmi kolejiště jen omezenou rychlostí. Při překročení této rychlosti by pak mohlo dojít k vykolejení, což je samozřejmě nežádoucí.

Dalším důvodem pro synchronizaci je vytvoření předvídatelného prostředí pro obsluhu. Obsluha má často k dispozici ovladač pro řízení rychlosti jízdy vozidla v ručním režimu. Autor této práce považuje za

velice užitečné, aby platilo, že otočení trimru ovladače o daný úhel způsobí u všech vozidel pohyb stejnou rychlostí (úhel otočení trimru ovladače odpovídá jízdnímu stupni). Obsluha kolejiště je pak rychlejší a efektivnější.

Třetím důvodem k synchronizaci rychlostí vozidel je umožnění přípřeží a postrků – tj. situací, kdy v jednom vlaku jede více lokomotiv. Tyto lokomotivy se musí pohybovat stejnou rychlostí, jinak hrozí vykolejení vlaku.

1.3.1 Současné řešení problému

Současné řídicí programy problém synchronizace rychlostí typicky řeší tak, že si u každého hnacího vozidla ukládají tabulku *jízdní stupeň: rychlost*. Když tyto programy chtějí, aby dvě lokomotivy jely stejnou rychlostí, vyhledají příslušné jízdní stupně a do centrály (v obecném případě) odešlou pro každou lokomotivu jiný jízdní stupeň.

Provozu lokomotivy na kolejišti tak předchází buď

- ruční zadání této tabulky, nebo
- automatické měření rychlosti lokomotivy.

Je však nutné podotknout, že toto řešení neuspokojuje naše požadavky pro ruční řízení jízdy hnacího vozidla, protože tabulka rychlostí je uložena pouze v řídicím SW, s kterým nebývají ovladače propojeny, takže nemohou tabulku využít. Stejnému úhlu otočení trimru ovladače tak zpravidla odpovídají různé rychlosti u různých lokomotiv, přípřeže a postrky jsou prakticky nemožné.

1.3.2 Řešení problému v hJOP

Alternativním řešením problému synchronizace rychlostí je způsob, který v současné době využívá SW hJOP. Tato práce cílí právě na onen alternativní způsob řešení kalibrace. Jeho výhoda je v tom, že řeší synchronizaci rychlostí i pro ruční ovladače.

V každém lokomotivním dekodéru je možno pro každý z 28 jízdních stupňů nastavit konkrétní otáčky motoru. SW hJOP předpokládá, že uživatel provede před provozem hnacího vozidla tzv. *kalibraci*, tj. přiřazení otáček motoru každému rychlostnímu stupni každého vozidla tak, aby bylo splněno, že konkrétní jízdní stupeň odpovídá u všech vozidel stejné rychlosti.

Příklad: u všech lokomotiv platí, že jízdní stupeň 15 odpovídá rychlosti 40 km/h.

Dalšími výhodami tohoto řešení jsou:

1. odpadnutí nutnosti udržovat si u každého vozidla v SW kalibrační tabulku,
2. instantní přenos tabulky mezi kolejišti, když je přenesena lokomotiva (to proto, že tabulka je jednoduše fyzicky uložená v lokomotivě, a tudíž logicky cestuje s ní).

Nevýhodou tohoto přístupu je, že je nutné provést netriviální proces kalibrace. V současné době tento proces zahrnuje ježdění s lokomotivou na měřicím okruhu, měření její rychlosti a ruční nastavování kalibrační tabulky.

Cílem této práce je tento proces automatizovat.

1.4 Synchronizace brzdných křivek vozidel

Možná ještě fundamentálnějším problémem než synchronizace rychlostí je synchronizace délky dojezdu z určité fixní rychlosti napříč různými vozidly. Lokomotivní dekodér má pro tento účel vyhrazeno CV #4 [9], které definuje dobu brzdění. Při nenulové hodnotě tohoto CV tak dekodér po obdržení příkazu „stůj“ nezastaví ihned, ale postupně zpomaluje.

Tento parametr je naprosto klíčový, ovládací SW totiž typicky vyžaduje, aby dané vozidlo zastavilo na konkrétní pozici (před návěstidlem, u nástupiště, ...). Současné ovládací aplikace problém synchronizace dojezdů řeší typicky tak, že si dojezd změří a dále s ním kalkulují

v dalších výpočtech. Například program *TrainController* [3] dokonce při brzdění posílá dekodéru postupně povely „jeď stupněm 12“, „jeď stupněm 10“, „jeď stupněm 5“, ..., takže se efekt brzdné křivky prakticky eliminuje.

Autor této práce považuje toto řešení za nešťastné, neboť

1. se zásadně zvyšuje datový tok mezi SW a centrálou,
2. je třeba si u každé lokomotivy udržovat další data,
3. po převzetí lokomotivy na ruční ovladač se každá lokomotiva chová jinak,
4. při jízdě přes ruční ovladač jsou typicky dojezdy tak malé, že lokomotiva brzdí nemodelově rychle.

Proto byl při tvorbě SW hJOP zvolen alternativní přístup. Je striktně definováno, že *každá lokomotiva musí z rychlosti* 40 *km/h zastavit na vzdálenosti* 30–35 *cm*.[1] Do této vzdálenosti od kýženého bodu zastavení jsou do kolejiště fyzicky montovány senzory. Po projetí vozidla senzorem pošle hJOP povel „lokomotivo s adresou 1254, stůj!“, lokomotiva začne plynule brzdit a do přesně dané vzdálenosti zastaví. A to vše modelově plynule.

Kalibrace dojezdů zajišťuje to, že hnací vozidlo nikdy nepřejede návěstidlo nebo nenajede do špatně postavené výměny. Je proto ještě důležitější než kalibrace rychlosti.[2]

Součástí procesu kalibrace by tedy měla být také synchronizace brzdných křivek.

1.5 Cíl práce

Cílem této práce je automatizovat proces kalibrace lokomotivy.

1. Tato hodnota se liší pro jednotlivá měřítka, 30–35 cm je standard pro měřítko TT 1:120.
2. Kdyby uživatel zkalibroval lokomotivu tak, aby jezdila hodně rychle, ale pořád dodržel 30–35 cm brzdnou vzdálenost z rychlostního stupně 15 (ten odpovídá dle standardu KMŽ 40 km/h), lokomotiva nikdy nevjede, kam nemá.

Nejprve je nutné navrhnout metodu pro měření rychlosti hnacího vozidla a zabezpečit přenos naměřených dat do řídicího PC. Poté je nutné navrhnout SW, který na nákladě změřené rychlosti nastaví kalibrační tabulky hnacího vozidla.

Výstupem této práce je řešení, které umožní proces kalibrace především automatizovat, což povede k ušetření lidských kapacit. Dále umožní proces kalibrace zrychlit, což přispěje k minimalizaci režijního času spojeného s provozem kolejiště, a zároveň proces kalibrace zpřesnit, což povede ke zvýšení spolehlivosti provozu na kolejišti.

2 Požadavky na řešení

Pro návrh vhodného řešení procesu automatické kalibrace je nutné formulovat požadavky kladené na toto řešení. Základní požadavky seřazené od nejdůležitějšího jsou:

1. snadné použití pro koncového uživatele, snadná instalace,
2. možnost použít řešení pro libovolné modelové měřítko,
3. maximalizace automatizace a minimalizace nutné spolupráce s uživatelem,
4. urychlení procesu kalibrace,
5. možnost provést dokalibrování za skutečného provozu,
6. přehledné verzování projektu, čistý a přehledný kód, využití otevřených technologií.

První bod vyplývá z faktu, že proces kalibrace bude často spouštěn laickou obsluhou kolejiště. Proto bude kladen důraz na očekávatelné chování grafického rozhraní a jednoduchost práce se samotným měřicím hardwarem. Řešení bude také navrženo tak, aby využívalo běžně dostupné technologie, se kterými jsou uživatelé zvyklí pracovat.

Stejně jako jinde ve světě, tak i v brněnském Klubu je provozováno více modelových měřítek, autor se vynasnaží navrhnout řešení tak, aby bylo nezávislé na měřítku.

Za okomentování dále stojí bod (5). Po zakoupení nového vozidla se typicky provádí jeho kalibrace, vozidlo je následně začleněno do provozu, kde jezdí i několik let. Fyzikální realita světa je bohužel neúprosná – ve vozidle dochází k mechanickému opotřebení prvků, zaběhnutí pohyblivých částí po určité době provozu a ke změně mechanických parametrů vozidla v závislosti na teplotě. Důsledkem těchto skutečností je, že kalibrace vozidla po určité době ztrácí přesnost. Vychýlení typicky není zásadní, ale mnohdy znatelné.

Proto je vhodné navrhnout kalibrační hardware tak, aby jej bylo možné použít i v ostrém provozu na kolejišti. Vozidlo, které by mělo nedokonalou kalibraci, by bylo dispečerem označeno a ovládací SW kolejiště by se postaral o to, aby byla kalibrace obnovena. Tato práce si sice

neklade za cíl automatické dokalibrovávání implementovat – protože pro tento úkol bude nutná netriviální součinnost s řídicím SW kolejiště – bylo by ale nanejvýš vhodné, aby na tento proces bylo hardwarové řešení připraveno.

3 Existující řešení

V této kapitole budou stručně představeny existující možnosti měření rychlosti modelového hnacího vozidla. Tyto možnosti budou doplněny o komentáře vhodnosti jednotlivých řešení pro problém řešený touto prací. Dále budou představeny dostupné technologie senzorů, které jsou pro problém měření rychlosti vhodné.

Ze senzorů pro měření rychlosti je obvykle snadno možné dopočítat dráhu, proto speciální senzory pro měření dráhy nebudou uvažovány.

3.1 Statické bodové detektory

Jednou z nejjednodušších technologií pro detekci rychlosti pohybujícího se objektu jsou stopky a dva senzory detekující průchod bodem. V naší aplikaci na modelovém kolejišti si lze představit dva (např. optické) senzory umístěné v určité vzdálenosti od sebe, skutečnou rychlost pohybujícího se objektu lze pak vypočíst jednoduchým vydělením vzdálenosti senzorů a času, po který se hnací vozidlo pohybovalo mezi senzory.

Měření vychází z předpokladu, že se hnací vozidlo pohybuje konstantní rychlostí (podobně jako řada dalších rozebíraných metod měření níže), tento předpoklad je však zejména díky kontrole nad ovládáním vozidla a díky BEMF[1] možné bez problému splnit.

Konkrétní instancí této technologie je *Model Railroad Speedometer Accutrack II* [10].

Nespornými výhodami tohoto přístupu jsou jednoduchost a absence pohyblivých prvků, která vede k delší životnosti a dlouhodobé spolehlivosti měřiče. Bohužel zásadní nevýhodou tohoto řešení je nutnost měřit rychlost pouze ve vybraném úseku tratě. V požadavcích na řešení vyvíjené v této práci je sice specifikováno, že kalibrace bude

1. Back Electro-motive force, technologie pro měření rychlosti ovládaného motoru, viz `https://dccwiki.com/Back_EMF`.

probíhat na uzavřeném okruhu, avšak představme si, že pro jedno změření rychlosti by bylo třeba buď projet celý okruh, nebo se za senzorem zastavit a obrátit směr. Pomineme-li fakt, že motory často dávají při stejné střídě napájecího signálu mírně rozdílné otáčky v závislosti na směru, ve kterém se pohybují, je i tak čas nutný na celou kalibraci nesrovnatelně vyšší než u senzoru, který by byl schopen odečítat rychlost průběžně, nezávisle na poloze vozidla.

Uvědomme si ale na závěr, že jedna z klíčových nevýhod tohoto senzoru je v určitém slova smyslu i jeho výhodou. Totiž fakt, že měření probíhá jen na pevně definovaném úseku koleje, s sebou nese tu výhodu, že vozidlo měříme vždy ve stejných podmínkách. Nemusíme tedy například řešit, jestli vozidlo zrovna jede v oblouku a je zpomalováno odstředivou silou, protože senzor prostě umístíme na rovný úsek trati.

3.2 Statické úsekové detektory

Podobný přístup jako u statických bodových detektorů nabízejí statické úsekové detektory. Rozdíl oproti metodě popsané v předchozí kapitole je ve způsobu měření: senzor neměří projetí vozidla daným bodem, ale přítomnost vozidla na úseku tratě určité délky.

Typickým zástupcem této technologie je proudový detektor obsazení, konkrétně například *BD20* [11].

Proudové detektory obsazení jsou hojně využívány na modelových kolejištích k detekci volnosti úseků, a tudíž k umožnění bezkolizního pohybu vozidla po trati. Obdobné systémy se využívají na skutečné železnici, avšak tam jsou známy pod názvem *kolejové obvody*.

Konkrétní způsob měření přehledně popisují například tvůrci SW *JMRI* [12] (*A Java Model Railroad Interface*). Kalibrační kolej je rozdělena na několik částí (optimálně na tři části), kalibrační SW pak lokomotivou postupně projíždí těmito částmi a počítá čas, za který vozidlo projede prostřední částí. U této části má od uživatele zadanou její délku, z které pak vypočte rychlost vozidla.

Výhody a nevýhody tohoto způsobu měření jsou víceméně stejné jako výhody a nevýhody statických bodových detektorů. Malou výhodou úsekových detektorů je fakt, že kolejiště často bývají vybavena úsekovými detektory pro účely zapezpečovacího zařízení, takže není nutný nákup a instalace dalších senzorů.

3.3 Podvalníkové detektory

Zajímavým způsobem měření rychlosti prezentovaným v [13] je možnost měřit rychlost hnacího vozidla bez nutnosti pohybu po skutečné trati.

Tuto metodu si lze představit podobně jako měření rychlostí aut v autoservisu. Lokomotiva je položena na válce, které se pod ní mohou volně pohybovat. Lokomotiva svým motorem roztáčí válce, jejichž rychlost je následně měřena.

Zajímavými výhodami této technologie jsou použitelnost prakticky pro jakékoliv modelové měřítko, nezávislost na pozici vozidla na trati a v neposlední řadě zmizení nutnosti pořizovat kalibrační okruh. Celý proces kalibrace tak může probíhat na velice malém prostoru.

Je rozhodně nutné uznat, že toto řešení je svým důvtipem minimálně oceněníhodné. Avšak pro potřeby této práce skrývá jednu zásadní nevýhodu – totiž nemožnost dokalibrovávat vozidlo za reálného provozu bez nutnosti uvolnění vozidla z reálného provozu na trati.

Stojí za to také poznamenat, že řešení, které by umožňovalo měřit rychlost vozidla na kolejišti za provozu, by s sebou neslo tu výhodu, že jej lze použít k měření celkové ujeté vzdálenosti hnacího vozidla za delší dobu, což je údaj využitelný například k výpočtu míry opotřebení lokomotivy.

3.4 Kalibrační SW

Kromě vytvoření detektoru pro měření rychlosti hnacího vozidla je další zcela zásadní částí této práce navržení programu, který zvládne

celý proces kalibrace automaticky – bez zásahu uživatele. V tomto hledisku autor bohužel nemůže nabídnout výčet existujících řešení, protože taková řešení prakticky neexistují. Je to z toho důvodu, že většina dostupných SW pro řízení modelového kolejiště využívá odlišného principu, než na kterém je založen SW hJOP [1], pro nějž je automatická kalibrace řešena.

Většina obslužných programů totiž počítá s tím, že hnací vozidla na kolejišti jsou různá, a nesnaží se je jednotně kalibrovat. Místo toho si tyto SW před prvním použitím lokomotivy nechají zadat její parametry nebo si je změří jednou z technologií popsaných výše a u lokomotivy si tzv. *kalibrační tabulku* dlouhodobě udržují. Tyto programy tak kompenzují rozdílné vlastnosti vozidel přímo povelováním různými rychlostními stupni.

Koncepce SW hJOP je však jiná. Bylo rozhodnutím autora, že je vhodnější mít zkalibrovaná vozidla na úrovni dekodérů, než si v SW ukládat pro každé hnací vozidlo speciální *kalibrační tabulku*.

3.5 Shrnutí

Z popisu vlastností výše uvedených komerčně dostupných technologií měření rychlosti modelového hnacího vozidla vyplývá, že žádné z těchto řešení nevyhovuje autorovým požadavkům plně. Proto autor přistoupí k návrhu vlastního měřicího systému.

4 Měřicí vůz

Tato kapitola popisuje celkovou koncepci vlastního detektoru pro měření rychlosti hnacího vozidla – tzv. měřicího vozu. Dále popisuje a zdůvodňuje rozhodnutí, která autor při konstrukci detektoru zvolil.

V kontextu následujícího textu si autor dovoluje připomenout Požadavky na řešení (kapitola 2).

4.1 Umístění detektoru

Prozkoumejme nejprve, kam je a kam není možno detektor umístit. Určitě platí, že detektor není možné umístit do lokomotivy. Lokomotiva je totiž komerční produkt, který umožňuje úpravy své elektroniky nejvýše na úrovni výměny digitálního dekodéru DCC. Lokomotiva obvykle bývá vybavena deskou plošných spojů, která propojuje jednotlivé součásti lokomotivy (motor, koleje, dekodér, osvětlení, ...). Tato deska je specifická pro konkrétní model a dodává ji výrobce hnacího vozidla. Zasahovat do elektroniky lokomotivy je tedy vysoce nepraktické a mnohdy i nemožné.

Druhou možností je doplnit detektor do lokomotivy bez úpravy její elektroniky. To s sebou ale nese pořád tu nevýhodu, že je třeba mechanicky zasahovat do (poměrně drahé) lokomotivy.

Třetí možností je umístit detektor mimo hnací vozidlo. Detektor může být umístěn na kolejiště staticky (podobně jako v 3.1) nebo může být součástí vagónu připojeného za lokomotivou. Statické detektory jsou nevyhovující zejména proto, že neumožňují provádění dokalibrace za skutečného provozu.

Jako nejvhodnější řešení se tedy jeví vyrobit vůz, který se připojí ke kalibrované lokomotivě jako běžný vagón. Součástí tohoto vozu bude detektor pro měření rychlosti. Vůz pak bude možné připojit bez narušení provozu i za vlak na „produkčním“ kolejišti.

S uvážením výše uvedených argumentů se autor rozhodl vydat cestou *měřicího vozu*.

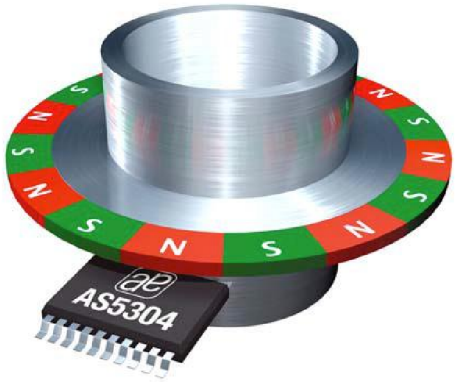
4.2 Senzor

Důležitou součástí měřicího vozu je rychlostní senzor. Zvolená technologie senzoru ovlivňuje formát a přesnost měření, dále pak mechanicko-elektrické vlastnosti měřicího vozu, jeho cenu, snadnost výroby, opakovatelnost výroby (dostupnost součástek s výhledem na několik let) a v neposlední řadě poruchovost a opravitelnost měřicího vozu.

Autor uvážil několik dostupných technologií senzorů.

4.2.1 Magnetický senzor

První uváženou technologií jsou tzv. *magnetické poziční senzory*. Tyto senzory fungují na bázi hallových sond. Umí měřit úhel natočení objektu vzhledem k senzoru na základě magnetického pole. Jejich primárním účelem je měřit natočení nejrůznějších ramen či prvků ovládaných např. servomotory, opakovaným měřením natočení lze ale dosáhnout také velice přesného měření rychlosti. Princip těchto senzorů dobře ilustruje obrázek 4.1.



Obrázek 4.1: Senzor AS5304 měřící natočení obruče na základě počítání pólů. Převzato z [14].

Výhodou tohoto způsobu měření je velice vysoká přesnost (řádově 10 bitů na jednu otáčku). Nepříznivým faktem ale je, že tyto senzory

vyžadují magnetický disk, přičemž ten nejmenší dosahuje průměru řádově 15 mm [15]. Tento průměr je pro modelovou železnici bohužel příliš velký.

4.2.2 Magnetický senzor „cykloměřič“

Technologie tzv. cykloměřiče byla historicky první metodou, jak měřit rychlost modelové lokomotivy. Tento senzor je založený na stejném principu jako počítač rychlosti na cyklistickém kole. Na osu nápravy je upevněn magnet, ve voze je pak hallova sonda. Magnet vyvolá při průchodu kolem hallovy sondy zákmit v magnetickém poli, který tato sonda detekuje. Na jednu otáčku nápravy tak připadá jeden zákmit, cyklopočítač pak zaznamenává počty zákmitů.

Toto řešení je poměrně jednoduché na instalaci, množství mechanických prvků je malé, na druhou stranu ale poskytuje poměrně malou přesnost – zejména při nižších rychlostech. Technologie „cykloměřiče“ se tedy pro potřeby této práce také nejeví jako vhodná.

4.2.3 Optický senzor

Třetí zvažovanou technologií je optický senzor. Tento senzor je proslulý svým užitím v optických myších. Pracuje na principu snímání obrazu podložky CCD čipem a porovnávání dvou obrazů pořízených v krátkém časovém intervalu za sebou. Ze dvou obrazů je pak určen rozdíl polohy myši.[1]

Toto řešení vypadá velice slibně, jeho obrovskou výhodou je absence jakýchkoliv mechanických částí. Stačí namířit senzor na pražce a jednoduše měřit, jak se vůz pohybuje. Pražce jsou navíc vůči štěrku mezi pražci velice dobře kontrastní, porovnávání obrazů by tak mohlo být přesné (srovnejte s pohybem optické myši po takřka uniformním povrchu stolu nebo podložky pod myš).

1. Zajímavostí je, že celá logika porovnávání dvou obrazů je v takovýchto senzorech implementována hardwarově.

Pro kalibrační vůz byl vybrán senzor *ADNS-3050* [16]. Tento senzor byl mezi mnohými podobnými zvolen proto, že (1) má výstup sériovou linkou a (2) zvládá měřit vysoké rychlosti (vagón se typicky pohybuje o něco rychleji než myš).

Autor osadil senzory na desku plošných spojů a začal testovat přesnost měření pomocí jednoduchého zapojení s vývojovým kitem Arduino. Nemalou výzvou bylo sehnat čočku, která autorovi umožní zaměřit pražce a zároveň ponechat senzor na ploše vagónu. Díky spolupráci s Ústavem přístrojové techniky AV ČR se však povedlo dostatečně malou čočku obstarat.

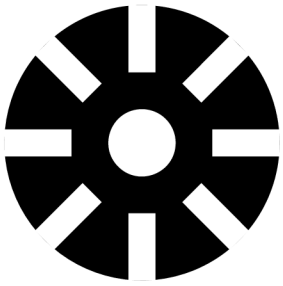
První měření ukázala, že pro dobrou funkci senzoru je třeba poměrně kontrastní snímaná plocha a poměrně intenzivní červené světlo (na to je CCD čip v senzoru nejcitlivější[2]). Navíc se ukázalo, že hloubka ostrosti senzoru je nejvýše 1 mm, což reálně znemožňovalo měřit rychlost na kolejích, které nejsou zasypány štěrkem, ale pouze připevněny k podložce. Složitý optický senzor tak degradoval na zařízení, jehož funkci by v podstatě šlo obstarat pomocí jedné LED diody a jednoho fototranzistoru [3].

Po několika měsících pokusů o zprovoznění měření optickým senzorem autor naznal, že s tímto typem senzoru nebude schopen garantovat přesné a spolehlivé měření. Padlo tedy rozhodnutí tuto technologii opustit. Autor by rád zdůraznil, že jednou z hlavních motivací pro opuštění této technologie byla přílišná komplexnost senzoru a neschopnost ovlivňovat a debugovat procesy uvnitř něj.

4.2.4 Optozávora

Čtvrtá uvažovaná technologie senzorů je inspirována návrhem kolegy modeláře Petra Trávníka. Jejím základem je paprskové kolo (viz obr. 4.2) upevněné na nápravě měřicího vozu. Paprskové kolo se otáčí spolu s nápravou, k vozu je pak připevněna optozávora, která snímá průchod jednotlivých paprsků. Klíčovým faktorem pro použitelnost

2. Proto si optická myš přisvětluje červeným světlem.
3. Stačí jednoduše měřit intenzitu odraženého světla a zkoumat, jestli je senzor zrovna na pražci, nebo mezi nimi.



Obrázek 4.2: Příklad paprskového kola.

této technologie jsou zejména technické možnosti výroby dostatečného množství paprsků při zajištění rovnoměrnosti rozestupu paprsků a malé velikosti kola. Průměr kola by totiž neměl přesáhnout 6 mm (pro měřítko TT, 1:120).

Tato technologie sice obsahuje mechanické prvky, ale pořád je jich poměrně málo. Při dostatečném počtu paprsků dosahuje dobré přesnosti, je jednoduchá, a tudíž jsou data z ní snadno zpracovatelná. Optozávora je také levná a relativně snadno dostupná.

Ze všech zvažovaných technologií autor nakonec zvolil právě technologii paprskového kola a optozávory.

Nejvhodnější technologií pro výrobu takového kola je zřejmě leptání, protože však autor nemá k této technologii přístup, rozhodl se místo jednotlivých paprsků kolo vyrobit dírkované. Jako vhodný kompromis bylo stanoveno 8 dírek na kolo, neboť tento počet je dostatečný pro měření malých rychlostí a zároveň je perforované kolo o 8 dírkách vyrobitelné konvenčním vrtáním. Při výrobě se podařilo dosáhnout přesnosti umístění jedné dírky $\pm 0,05$ mm. To jsou poměrně dobré výsledky.

Výkres kola je zobrazen v příloze A.1.

Drobným problémem pak bylo sehnat dostatečně malou optozávoru. Běžné optozávory dostupné v českých obchodech dosahují rozměrů cca $15 \times 6 \times 10$ mm [4]. To je pro účely vozu v měřítku 1:120 zcela nepřijatelné. Po průzkumu možných alternativ byl nakonec vybrán senzor

4. Viz například `https://www.gme.cz/optozavory-reflexni-optocleny`.

GP1S23 [17], který byl dříve hojně montován do disketových mechanik. Dnes je senzor běžně dostupný například na modulu k vývojové desce Arduino pod označením *KY-010* [18].

4.3 Napájení

Jedním z klíčových faktorů, který určoval výslednou podobu měřicího vozu, bylo napájení. Pro napájení elektroniky jsou tři možnosti:

1. napájení z kolejí,
2. napájení z baterie,
3. kombinace předchozích dvou (místo baterie lze případně použít kondenzátory).

Výhody a nevýhody jsou poměrně jasné: baterie má omezenou výdrž, napájení z kolejí je stálé. Na tomto místě je ale třeba zmínit, že napětí v kolejích sice je (téměř) trvale přítomné, ale zajistit jeho spolehlivé sbírání je skutečně výzva. Sbírání je typicky realizováno plíšky s kontakty na kola. Na základě odhadnuté spotřeby měřicího vozu (viz níže) byla vytvořena modelová zátěž a indikace, jak dobře je sbírání schopno tuto zátěž napájet.

Naměřená data bohužel byla naprosto tristní. Sbírání bylo realizováno skrze hrotová ložiska (obou náprav dvounápravového měřicího vozu), protože při použití sbíracích kontaktů přiložených přímo na kolo nápravy vzniká riziko, že tření mezi kontaktem a kolem bude tak vysoké, že kolo bude na kolejnici prokluzovat. A to je v kontextu faktu, že na nápravě probíhá měření rychlosti, naprosto nemyslitelné. Zátěž byla napájena cca 50 % času, a to i přes veškerou snahu o přídavné kondenzátory a případnou přídavnou hmotnost pro zlepšení kontaktu vozu s kolejemi. Zajištění kvalitního sbírání by s sebou tedy neslo nutnost – podobně jako u lokomotiv, kde sbírání funguje i přes odběry motorů kolem 400 mA – vůz zatížit, případně zvýšit počet náprav a zajistit jejich kyvnost. A to by se neobešlo bez náročných mechanických úprav měřicího vozu.

Na základě provedených měření i autorových předchozích zkušeností se sbíráním u vagónů bylo rozhodnuto zkonstruovat vůz bez sbírání.

Později byla zvažována řešení zahrnující baterie i sbírání dohromady – tedy baterie dobíjené z kolejí –, tato řešení se ale ukázala jako příliš komplikovaná.

Poslední alternativou tedy zůstala baterie. Výzvou při použití baterie je vměstnat co největší baterii do malého prostoru vozu. Minimální doba kalibrace na jedno nabití byla stanovena na 3 hodiny. Hlavní výhodou baterie je to, že vůz není nutné mechanicky upravovat (instalovat sběrací kontakty). Vůz lze také za jeho plné funkčnosti bez problému sejmout z kolejí a například tak vyzkoušet funkčnost senzoru. Tím se výrazně zjednodušuje proces vývoje a testování.

Za technologii baterií byla zvolena LiPol.

Skutečnou výzvou se tedy stalo vyrobit elektroniku tak, aby její spotřeba byla vskutku malá (viz další kapitolu).

4.4 Přenos dat do PC

Dle požadavků na měřicí technologii (viz kapitolu 2) je nutné zajistit přenos dat ze senzoru. Primární technologií na straně příjemce je pro účely této práce počítač.

Logickým komunikačním prostředkem jsou koleje. Protokol DCC [19] však umožňuje pouze jednosměrnou komunikaci (z centrály do lokomotivy) [19]. Existují sice proprietární řešení umožňující obousměrnou komunikaci (například *RailCom* [20]), avšak tato řešení vyžadují úpravy napájecí soustavy kolejiště a podporu v DCC centrále. To je zejména při požadavku na provoz vozu na reálném kolejišti nemyslitelné. Navíc je samozřejmě nutné zabezpečit sbírání vozu, které již bylo diskutováno v kapitole 4.3.

Druhou možností je bezdrátový přenos dat. Pokud se podaří vybrat takovou technologii, aby data mohla být přenášena například i na mobilní telefon či tablet, bude to pro měřicí vůz jen výhodou. Autor se proto rozhodl vyhnout se proprietárním řešením pro bezdrátový přenos dat a cílit na technologie, které jsou běžně dostupné, ideálně přímo zabudované v dnešních počítačích. V neposlední řadě je třeba

zmínit, že nutnost pořizovat či stavět přijímač by zvyšovala náročnost celého projektu, a to jak po stránce finanční, tak po stránce výrobní.

S tímto vědomím přicházejí v úvahu dvě technologie:

1. WiFi,
2. Bluetooth.

4.5 Platforma

Na základě požadavků vyplynuvších z předchozích kapitol (senzor, přenos dat) je nutné vybrat hardware, který bude umístěn do měřicího vozu. Hardware musí zajistit (1) zpracování signálu ze senzoru, (2) komunikaci s nadřazeným zařízením, (3) ochranu proti podvybití baterie a (4) možnost zařízení vypnout a zapnout.

4.5.1 ESP-32

Prvním kandidátem byl procesor *ESP-32* [21], jehož největší výhodou je vestavěný Bluetooth a WiFi. Jedná se o poměrně výkonný procesor, který lze taktovat až na 240 MHz, a to dokonce na dvou jádrech. [22]

Autor pořídil *ESP-32* vývojový kit a začal zkoušet možnosti komunikace. Jako primární komunikační kanál byl vybrán Bluetooth, jelikož vyžaduje oproti WiFi menší režii spojenou s ustanovením spojení. Tento problém by šel u WiFi řešit tím, že se měřicí vůz nepřipojuje k již existující síti, ale sám síť vytváří. Negativem tohoto přístupu je ale to, že typicky není možné na počítači, který přijímá signál z měřicího vozu přes WiFi, být zároveň připojený do sítě internet (přes WiFi). A to je nanejvýš nepraktické.

Bluetooth na *ESP-32* fungoval zdárně. Na ESP se podařilo aktivovat Bluetooth SPP profil [23], takže se zařízení poměrně snadno spárovalo s počítačem a tvářilo se jako sériová linka. Profil SPP byl vybrán, protože má poměrně dobrou implementaci na celé škále operačních systémů a vytváří dobrou abstrakci pro použití z jakéhokoliv programu.

Téměř na každém operačním systému je totiž možné „otevřít sériovou linku“. Pokud by se podařilo zvolit pro vývoj aplikace na počítač takový framework, který zvládá abstrahovat sériový port nezávisle na cílové platformě, vzniklá aplikace by byla dokonale multiplatformní.

Po nasazení profilu SPP se však ukázalo, že spotřeba *ESP-32* je cca 80 mA, což jej dělá – v kontextu napájení z baterie – nepoužitelným. Bylo zvažováno využít specifikace *BLE (Bluetooth Low Energy)*, BLE však nemá profil pro sériovou linku, takže by bylo nutné vytvořit profil vlastní. To by znamenalo dodat specifikaci profilu do klientského zařízení (do počítače) a místo abstrakce sériovou linkou přistupovat přímo k zařízení Bluetooth. Toto řešení bylo zamítnuto pro příliš velkou pracnost a ztrátu elegance.

Autor ještě zvažoval, jak problém se spotřebou řešit pomocí sbírání či zvětšení baterie, všechna navrhovaná řešení však byla příliš komplikovaná nebo nefungovala dobře. ESP-32 tak bylo zavrhnuto.

4.5.2 ATmega + HC-05

Druhou volbou byla autorovi dobře známá rodina procesorů s architekturou *AVR* názvu *ATmega* [24]. Procesory rodiny *ATmega* jsou univerzální nízkopříkonové procesory, které jsou základem například vývojové desky Arduino [25]. Rodina procesorů *ARmega* byla zvolena také z toho důvodu, že s jejím programováním má autor předchozí zkušenosti.

Procesory *ATmega* ale bohužel neobsahují vestavěný modul pro bezdrátovou komunikaci. Bylo tedy nutné využít externí Bluetooth modul propojený s procesorem pomocí sběrnice. Pro tento účel byl zvolen běžně dostupný BT modul typu *HC-05*. Výhodou tohoto modulu je jeho dostupnost a nízká cena, nevýhodou je to, že modul trpí neduhy typickými pro celou řadu zařízení vyráběných v Číně. K modulu například není k dispozici kvalitní dokumentace a v Česku je to s jeho dostupností horší. Každopádně – co se týče světové dostupnosti – je na tom modul *HC-05* velice dobře. Autor zvažoval ještě další moduly (*HC-06*, *HC-07*), tyto moduly by však pro účel této práce nepřinesly žádnou zásadní výhodu.

Autor experimentálně změřil, že spotřeba nespárovaného *HC-05* modulu je 42 mA $\pm$ 0,2 mA, spotřeba spárovaného modulu je 20 mA $\pm$ 2 mA. To jsou oproti ESP-32 výrazně příznivější čísla. Navíc se podařilo zajistit spolehlivou funkci optozávory již při proudu 10 mA, což je polovina proudu, na který je senzor dimenzován [17]. S přihlédnutím ke spotřebě procesoru a dalších drobných součástek by se měla pohybovat celková spotřeba vozu kolem cca 30 mA. Tato spotřeba byla uznána jako přijatelná.

Pro konstrukci měřicího vozu byla tedy vybrána platforma *ATmega* s Bluetooth modulem *HC-05*.

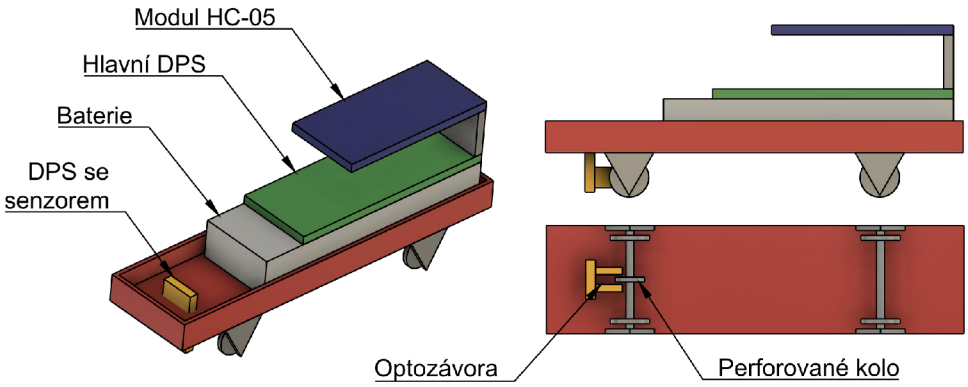
4.6 Mechanika

Vůz se bude po mechanické stránce skládat z několika základních částí:

1. perforované kolo připevněné na nápravě,
2. optozávora připevněná k vozu snímající perforované kolo,
3. baterie,
4. deska plošných spojů s hlavní elektronikou,
5. komunikační modul *HC-05*.

Jako vůz byl použit běžně dostupný vůz měřítka TT typu *Nd* [26]. Na jedné straně bylo extrahováno spřáhlo a místo něj byla k nápravě umístěna optozávora. Optozávora je na vlastní miniaturní DPS, k vozu je DPS mechanicky připevněna lepidlem. Vůz *Nd* je otevřený vůz, autor si (minimálně v prototypu) neklade za cíl měřicí elektroniku modelově maskovat. Na plošině vozu je umístěna baterie, na ní je řídicí elektronika a na ní je komunikační modul. Celou situaci přehledně zobrazuje nákres 4.3.

Rozměr plošiny vozu je 20 $\times$ 80 mm, což umožnilo instalaci LiPol baterie o kapacitě 500 mAh (napětí 3,7 V). Teoretická výdrž vozu je tedy cca 15 hodin. Je však nutno poznamenat, že toto číslo je pouze teoretické, spotřeba zařízení se výrazně liší ve spárovaném a nespárovaném stavu, navíc níže zjistíme, že baterii nevybíjíme až na její



Obrázek 4.3: Nákres mechaniky měřicího vozu.

hranice. Předestřeme však již nyní naměřenou hodnotu, tj. že spárovaný vůz vydržel reálně komunikovat 14 hodin. To je velice uspokojivý výsledek.

4.7 Elektronika

Základní požadavky na elektroniku měřicího vozu jsou:

1. umět rychle a přesně vyhodnocovat data z optického senzoru,
2. umět komunikovat s modulem *HC-05*,
3. umožnit uživateli zařízení zapnout a vypnout,
4. umět měřit napětí na baterii,
5. umět automaticky vypnout zařízení v případě vybití baterie,
6. zobrazovat stav vozu LE diodami,
7. umožnit nabíjet baterii.

Tyto požadavky na elektroniku jsou víceméně intuitivní, jen u posledního doplňme, že jeho motivací je to, aby uživatel nemusel odpojovat baterii od elektroniky a nabíjet ji externě. Tím se výrazně sníží riziko omylného přepólování, nesprávného zacházení s baterií ze strany uživatele a mechanického poškození vozu vlivem časté manipulace s kabely a konektory.

Schéma desky plošných spojů je k nahlédnutí v příloze A.2. Nyní budou krátce popsány jednotlivé prvky schématu a jak tyto prvky přispívají k naplnění požadavků.

Hlavní součástkou na DPS je procesor *ATmega328p* [27]. Tento konkrétní typ byl vybrán, protože je poměrně moderní a je dnes montován do vývojových desek Arduino, tudíž je jeho cena díky vysokému počtu produkovaných kusů malá. Procesor se stará o všechnu logiku na desce. Signál z optozávory je připojen na pin *ICP*, který je speciálně určen k tomu, aby přesně a rychle měřil periodu vstupního signálu. Procesor dále měří napětí na baterii, které je ale nejprve nutné snížit napěťovým děličem. Při vývoji hardwaru byla v jednu chvíli překážkou přesnost napěťové reference, nakonec ale bylo zvoleno jako napěťová reference přímo napájecí napětí procesoru, které by mělo kolísat nejvýše ± 2 % [28].

Součástí schématu je dále nabíjecí obvod *MCP73831*, který byl vybrán zejména pro své velice malé rozměry.

Za nabíjecím obvodem je ochrana proti přepólování (tranzistor *T1A*) a dále zapínací a vypínací obvod celé elektroniky. Tento obvod je inspirován projektem *RB3201* [29] od kolegů z Robotiky Brno [30].

Napětí pro procesor je vytvářeno obvodem LDO[5] (obvod *IC3*), procesor pracuje na napětí $3,3$ V. Tato konstrukce s sebou nese zásadní omezení pro vybíjení baterie: napětí na baterii nikdy nesmí klesnout pod $3,5$ V[6]. Tato hodnota zahrnuje i korekce vypočítané z nepřesnosti výstupu LDO, jedná se tedy o *napětí, které měří procesor*. Jakmile procesor na baterii změří méně než $3,5$ V, musí se odpojit[7].

DPS dále obsahuje konektory pro periferie, testovací konektory, testovací plošky a LE diody.

Kompletní materiály k elektronice jsou k dispozici pod licencí *CC BY-SA 4.0* na serveru GitHub [31].

5. Low-dropout regulator.
6. Protože *drop voltage* použitého regulátoru je $0,2$ V. [28]
7. Pod touto úrovní napětí přestává fungovat měření napětí baterie korektně a při delším provozování měřicího vozu by mohlo dojít k jejímu podvybití a zničení.

4.8 Princip měření

První pohled na měření rychlosti vozidla by mohl vycházet z analogie s tachometrem na kole: stačí měřit, kolik pulzů za sekundu se na senzoru vytvoří, a podle toho počítat rychlost. Ukazuje se však, že tato metoda měření nedosahuje zejména při malých rychlostech dostatečné přesnosti.

Mnohem přesnějším způsobem měření je měřit periodu mezi jednotlivými pulzy ze senzoru. Zde vyvstává otázka, jestli bude procesor schopný měřit i rychlosti cca 120 modelových km/h.

Uvažme tedy modelovou rychlost vozidla $v_m = 120\,\mathrm{km/h} \doteq 33,3\,\mathrm{m/s}$. Protože se jedná o modelovou rychlost, pro získání reálné rychlosti měřicího vozu v je třeba uvážit modelové měřítko c (např. $c = 1:120$, $c = 1:87$).

$$v = v_m \cdot c$$

Aplikováním základních vztahů platících pro rovnoměrný pohyb po kružnici snadno dostáváme následující vztah.

$$v = \omega r = 2\pi f r$$

Oba předchozí vztahy už stačí jen dát do rovnosti, vyjádřit výslednou frekvenci a tuto frekvenci vynásobit počtem děr na kole. Získáme tak nejvyšší frekvenci, kterou by měla být elektronika vozu schopna s přehledem měřit.

Po dosazení nejhorších hodnot (měřítko 1 : 87, průměr kola 8 mm, rychlost 120 km/h) dostáváme frekvenci přibližně 220 Hz. To je ve srovnání s kmitočtem krystalu procesoru $f_{CLK} = 3,6864$ MHz zanedbatelná frekvence, kterou bude možné snadno měřit velice přesně.

Naměřené hodnoty budou zaznamenávány do paměti procesoru a jednou za pevně daný interval bude jejich průměr odeslán do počítače (o průměrování více v kapitole 4.11.1). Tento interval byl zvolen na 100 ms.

Z naměřené periody T (bezrozměrné číslo, jedná se o počet kmitů časovače procesoru) lze modelovou rychlost spočítat dle vztahu 4.1,

$$v_m = 3,6 \cdot \frac{\pi d F_{CPU}}{Thcp} \tag{4.1}$$

kde d je průměr kola nápravy měřicího vozu, $F_{CPU} = 3,6864$ MHz je frekvence krystalu procesoru, $h = 8$ je počet děr v kole, $p = 64$ je *prescaler* časovače a c je modelové měřítko (viz výše).

Spolu s aktuální rychlostí je vhodné měřit také ujetou vzdálenost. A protože výpočet vzdálenosti z rychlosti až v počítači by mohl být vzhledem k vzorkování rychlosti po 100ms intervalech nepřesný, je vhodné výpočet ujeté dráhy implementovat přímo do měřicího vozu. Druhou měřenou hodnotou je tedy počet pulzů senzoru. Hodnota čítače je 32bitové číslo, které je do počítače zasíláno jednou za 500 ms. Příslušné přepočty na hodnoty v uživateli čitelných jednotkách jsou pak jak v případě rychlosti, tak v případě ujeté vzdálenosti ponechány až na počítači, neb ten má spoustu výpočetních prostředků a nepotřebuje realizovat časově citlivé operace. Navíc odpadá problém s přenášením hodnot parametrů (průměr kola, měřítko) do měřicího vozu a komunikace tak může být pouze jednosměrná. Reálnou dráhu s lze pak z počtu kmitů vypočítat pomocí vztahu 4.2,

$$s = \frac{i\pi d}{h} \tag{4.2}$$

kde i je počet pulzů senzoru, d je průměr modelového kola a $h = 8$ je počet děr v kole.

Pro komunikaci s počítačem byl navrhnut vlastní binární protokol inspirovaný protokolem XpressNET [32]. Jedná se o jednoduchý protokol, který v prvním bytu zprávy určí délku zprávy, poslední byte zprávy je pak její kontrolní součet (*XOR*). Protokol umí ve verzi 1.0 zasílat informace o:

1. aktuální rychlosti,
2. ujeté vzdálenosti,

3. napětí na baterii.

Zajímavým aspektem protokolu je fakt, že používá kódování *7B8B* a nejvýznamnější bit nechává vždy nastavený na 1. Důvodem pro tuto vskutku podivnou vlastnost je fakt, že modul *HC-05* interpretuje hodnoty bytů jako ASCII znaky a některé znaky odmítne přenést (např. `0x04` *End of transmission*). Proto byla použita pouze polovina rozsahu jednoho bytu, která neobsahuje kontrolní znaky, konkrétně interval 128–255.

Kompletní specifikace protokolu je k dispozici online [8].

4.9 Firmware

Na poli Embedded System mezi programovacími jazyky dominují zejména jazyky C a C++, a to především díky přímočarému provázání příkazů jazyka přímo s výpočetními prostředky a periferiemi mikrokontroléru. Z relativní jednoduchosti úkonů, které bude procesor řešit, plyne také poměrně malá velikost jeho programu. Spolu s faktem, že s programováním procesorů rodiny AVR v programovacím jazyce C má autor práce zkušenosti, tento argument rozhodl pro programovací jazyk C.

Firmware procesoru je – podobně jako elektronika – koncipován jako opensource. Je dostupný pod licencí *Apache License v2.0* [33].

Pro kompilaci je využit standardní kompilátor `avr-gcc`, pro nahrávání firmwaru do procesoru je využit program `avrdude`. Deska plošných spojů je vybavena programovacím konektorem, procesor je tedy možné programovat až v desce. To umožňuje snazší výrobu desek a především výrazně usnadňuje vývoj vozu (firmware je možno kdykoliv aktualizovat).

Základní činnost procesoru lze shrnout do dvou bodů.

1. Procesor pomocí přerušení a využití speciálního pinu *ICP* měří periodu mezi jednotlivými pulzy senzoru.

8. `https://github.com/kmzbrnoI/wsm-fw/blob/master/protocol.md`

ICP je speciální mód časovače 1. V tomto režimu časovač čítá a při vzestupné hraně na pinu *ICP* velice rychle (hardwarově, bez generace přerušení) zkopíruje hodnotu čítače časovače do registru procesoru. Poté je vyvoláno přerušení, ve kterém je obsloužena naměřená hodnota. Konkrétně je od této hodnoty odečten čas předchozího pulzu[9], a tím pádem je určena perioda vstupního signálu. Perioda je uložena k budoucímu odeslání do počítače.

Více o dalším zpracování dat lze nalézt v sekci 4.11.1.

2. V primární `while` smyčce programu se nacházejí časově nekritické operace (primární smyčka totiž může být kdykoliv přerušena). Ve smyčce se každých 100 ms odesílá aktuální rychlost (tj. perioda, viz bod (1)) do počítače, dále se každých 500 ms odesílá informace o ujeté vzdálenosti a každých 5 s se měří napětí na baterii.

4.10 Prototyp vozu

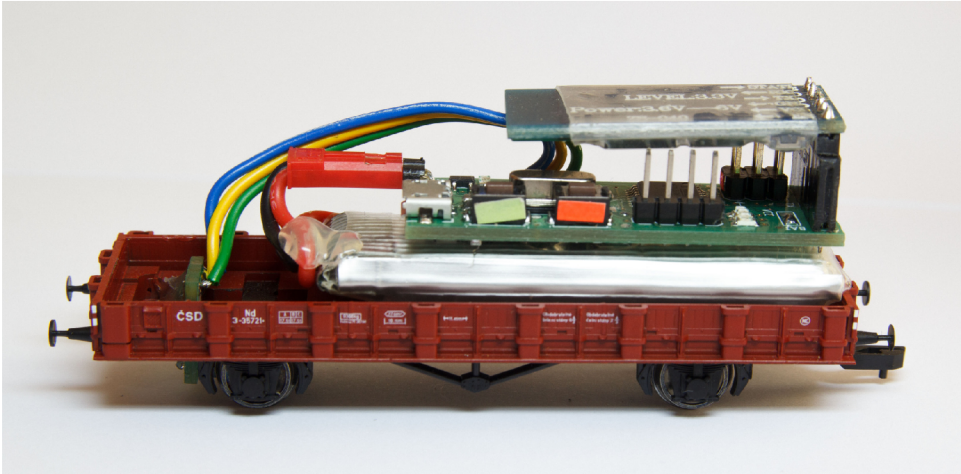
Na základě výše uvedených požadavků a specifikací se podařilo vytvořit prototyp měřicího vozu. Tento prototyp je vyobrazen na obrázku 4.4.

Tabulka 4.1 shrnuje klíčové parametry měřicího vozu.

Finální podobu vozu ovlivňovaly nejvíce ze všeho požadavky na malé rozměry všech komponent měřicího vozu, dalším důležitým faktorem pak bylo napájení – tedy sehnat takovou baterii, která se do vozu vhodně vejde a poskytne kapacitu alespoň na několik hodin kalibrace.

Po vyrobení vozu byla provedena „kalibrace“ samotného vozu – tedy určení konstant uvedených ve vztahu 4.1. Pro dosažení maximální přesnosti se totiž nelze spoléhat zejména na měření průměru kola pouhým posuvným měřidlem, rozdíly v průměru kola v řádu setin milimetru totiž způsobují rozdíly v modelové rychlosti v řádu desetin modelových km/h. Autor tedy přistoupil ke změření počtu pulzů na rovné koleji o délce 1 m.[10] Postupně byly naměřeny hodnoty 317,

9. Resp. hodnota čítače odpovídající okamžiku předchozí vzestupné hrany na pinu ICP.
10. Délka byla měřena konvenčním svinovacím metrem.



Obrázek 4.4: Fotografie prototypu měřicího vozu.

Tabulka 4.1: Přehled základních parametrů měřicího vozu.

Parametr	Hodnota	Jednotka
Rozměry vozu[a]	98 × 25	mm
Spotřeba vozu při nespárovaném BT	55	mA
Spotřeba vozu při spárovaném BT	33	mA
Kapacita baterie	500	mAh
Výdrž baterie	14	h
Pracovní napětí baterie	3,5–4,2	V
Využitá kapacita baterie	> 90	%
Minimální měřitelná rychlost	4	modelový km/h
Maximální měřitelná rychlost[b]	300	modelový km/h
Odhadovaná cena elektroniky	500	Kč

a. Rozměr přes nárazníky.
b. Účelové omezení ve firmwaru, senzor zvládá měřit alespoň do 1000 modelových km/h.

317, 316, 316 a 316. První tři hodnoty byly měřeny na jednom úseku trati v obou směrech pohybu, zbylé dvě hodnoty byly pro eliminaci chyby měřeny na jiném úseku trati, opět každá hodnota v jiném směru pohybu vozu v úseku a na rovné koleji bez jakýchkoliv nerovností.

Po zprůměrování pěti měření byl tedy průměr kola dle vztahu 4.2 stanoven na $8,05$ mm, kterážto konstanta je použita v budoucích programech.

4.11 Naměřená data

Měření rychlosti se ukázalo býti funkčním, vůz pravidelně posílal naměřená data, rychlost se pohybovala v odhadovaných intervalech. V naměřených datech byla ale vidět poměrně silná oscilace rychlosti, jejíž perioda byla řádově vyšší desetiny sekundy. Tato oscilace vykazovala amplitudu až 5 modelových km/h, což je pro účely kalibrace zcela nevyhovující.

Autor tedy přistoupil ke zkoumání příčin této oscilace a snaze o jejich odstranění. Autor upravil firmware v procesoru tak, aby zasílal do počítače každou naměřenou hodnotu (nikoliv průměr) a v počítači tak bylo možné vyhodnocovat surová data.

Při bližším pohledu na pohybující se vůz si šlo všimnout, že vůz ve spřáhlu s lokomotivou poměrně intenzivně pruží. To je dáno jak mechanickou konstrukcí spřáhla, tak tím, že vůz i lokomotiva obsahují kinematiku[11]. Autor tedy přistoupil k nahrazení spřáhla pružného za spřáhlo pevné. Tento krok pomohl ke snížení amplitudy oscilace, avšak oscilace rychlosti byla nadále přítomná.

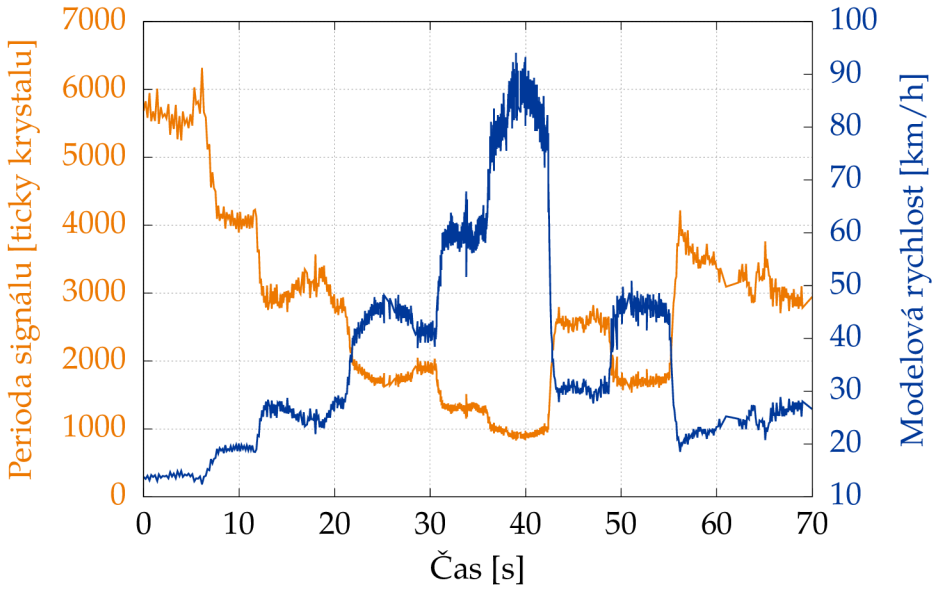
Ukázalo se tedy, že pro efektivní proces kalibrace bude nutné na kalibrované lokomotivě vždy vyměnit spřáhlo. Toto zjištění je nepříjemné, neboť jakákoliv nadbytečná manipulace s kalibrovaným modelem zvyšuje šanci jeho poškození. Autor však usuzuje, že se jedná o přijatelné riziko. Dnešní lokomotivy jsou totiž na výměnu spřáhla dobře

11. Kinematika je mechanismus uchycení spřáhla v lokomotivách a vagónech, ve němž je spřáhlo k rámu vozidla připevněno přes pružiny, které zaručují natočení spřáhla při průjezdu obloukem.

uzpůsobené, spřáhla typicky bývají v šachtách, a tak výměna spřáhla nepřináší až tak velká rizika.

Další zvažovanou možností bylo vůz tlačit přímo „nárazník na nárazník“, ukázalo se však, že pro pevné spojení s lokomotivou by bylo třeba vůz zatížit řádově o vyšší desítky gramů, což je zejména při omezeném prostoru ve vozu nereálné. Další možností je přidat soupravu se zátěží ještě před vůz a tlačit tak celý vlak, tuto možnost však autor vyhodnotil jako nepraktickou (bylo by třeba mít k dispozici další vozy, kalibrační vůz se tedy stává nekompaktním).

Naměřená data však stále vykazovala oscilaci v řádu 5 %, konkrétně perioda vykazovala chybu například 2000 ± 100 (pro modelových 20 km/h) ticků časovače 1. Chyba se navíc měnila s naměřenou rychlostí a to tak, že při vyšších intervalech (menší rychlost) byla chyba větší než při kratších intervalech (větší rychlost).



Obrázek 4.5: Průběh naměřené periody a modelové rychlosti v čase v analogovém režimu.

Graf 4.5 zachycuje průběh periody a modelové rychlosti v čase. Každý celek bodů na zhruba stejné souřadnici y zachycuje rychlost vozidla při

jednom napětí na jeho motoru. Rychlost je měřena na vozidlu s motorem připojeným přes kola náprav přímo ke koleji (tzv. varianta *analog*, bez dekodéru), aby se eliminoval případný vliv dekodéru. Čas na ose x je čas příjmu signálu do počítače. Vůči reálnému času zachycení signálu procesorem se tento čas může mírně lišit, avšak rozdíl latencí přenosů vůz–PC by měl být skutečně nepatrný.

Krom pravidelné oscilace rychlosti je na grafu místy patrný (např. v čase 37–42 s) globální „obloukový“ trend. Tento jev lze vysvětlit tím, že v průběhu měření rychlosti vozidlo zrovna najíždělo do oblouku, takže se jeho rychlost díky odstředivé síle zmenšila. Komentovaný jev je pro proces kalibrace parazitický, autor se jeho vlivu obával už při prvních teoretických návrzích vozu. Později se však ukázalo, že při kalibrování na oválu o dostatečně velkých poloměrech oblouků je změna rychlosti vlivem průjezdu obloukem minimální, resp. BEMF (viz 3.1) ji okamžitě vykompenzuje. Pokud by uživatel přeci jenom chtěl kalibrovat na obloucích o malých poloměrech, nabízí se možnost výběru používaných a nepoužívaných měřicích úseků na kalibračním okruhu. Stručně řečeno se bude měřit rychlost jen ve vybraných úsecích (typicky rovných úsecích), údaj o poloze vozidla bude založen na vyčítání ujeté vzdálenosti vozidla, viz sekci 4.8.

Všimněte si v grafu 4.5 také průběhu naměřené rychlosti. Díky způsobu, jakým je naměřená perioda přepočítávána na modelovou rychlost (4.1), dochází k dobré eliminaci velkých chyb při malých rychlostech, avšak chyby v periodě u větších rychlostí – přestože jsou v absolutních hodnotách o mnoho menší – způsobují mnohem větší oscilace modelové rychlosti.

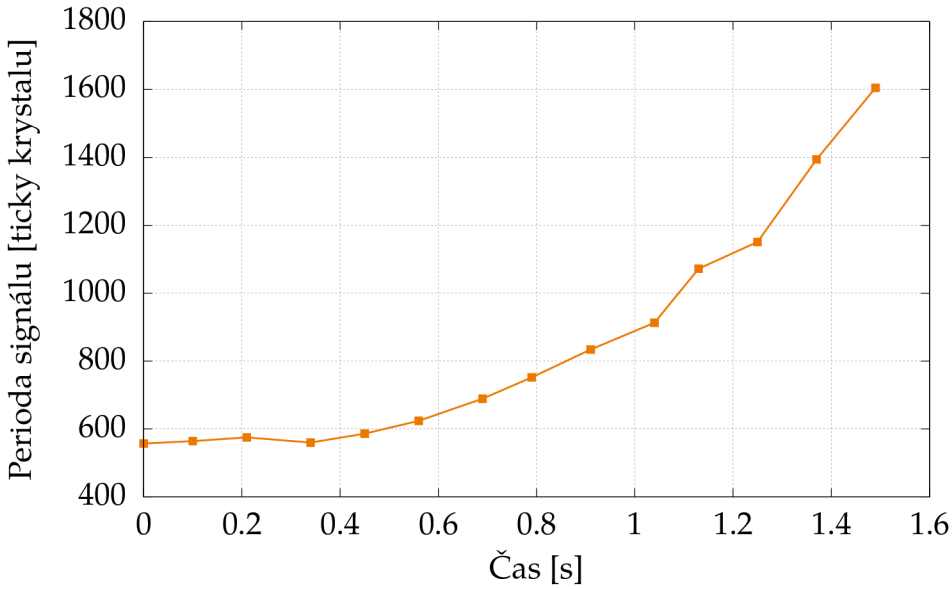
Pro oscilaci rychlosti bylo vytvořeno několik hypotéz.

1. Oscilace je způsobena nepřesností měření.

   Oscilace naměřených intervalů je v řádu stovek ticků časovače procesoru, je tedy zřejmé, že se nemůže jednat o chybu měření v procesoru. Dalším zdrojem chyb může být mechanická nepřesnost v poloze děr v perforovaném kole či chyba v umístění kola (natočené kolo).

   Pokud je tato hypotéza pravdivá, chyba v datech by se u měřicího vozu měla projevit, ať už je zdrojem pohybu vozu cokoliv. Autor

tedy přistoupil k tomu, že do vozu prostě „šťouchnul“ a sledoval jeho rychlost během zpomalování. Tuto rychlost zobrazuje graf 4.6.



Obrázek 4.6: Průběh naměřené periody zpomaleného pohybu.

Všimněte si, že rozsah na ose y je asi 4× menší než rozsah v grafu 4.5, přesto není oscilace na tomto grafu patrná.

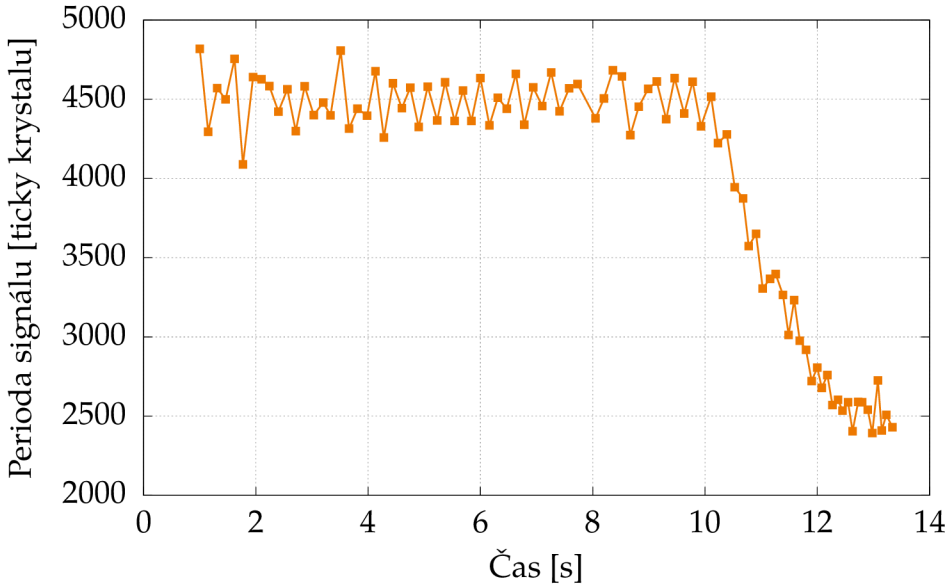
Oscilace tedy není způsobena chybou v konstrukci měřicího vozu.

2. Oscilace je způsobena BEMF regulátorem v dekodéru.

   Jak bylo zmíněno v kapitole 3.1, každý dekodér v sobě má PID regulátor, který zajišťuje udržování rychlosti hnacího vozidla při rozdílném počtu zavěšených vozů, resp. při rozdílné zátěži. Jako každý PID regulátor, i tento regulátor nemusí být dobře nastaven, výrobce dekodéru ostatně často vůbec netuší, do jakého modelu bude jeho dekodér osazen.

   Tuto hypotézu však vyvrací fakt, že oscilace se projevuje, i když měříme rychlost vozidla bez dekodéru – v tzv. režimu analog, kdy připojujeme napětí v kolejích přímo na motor.

3. Oscilace je způsobena charakteristikou motoru a mechanickými vlastnostmi modelu.



Obrázek 4.7: Detailnější pohled na oscilace rychlosti.

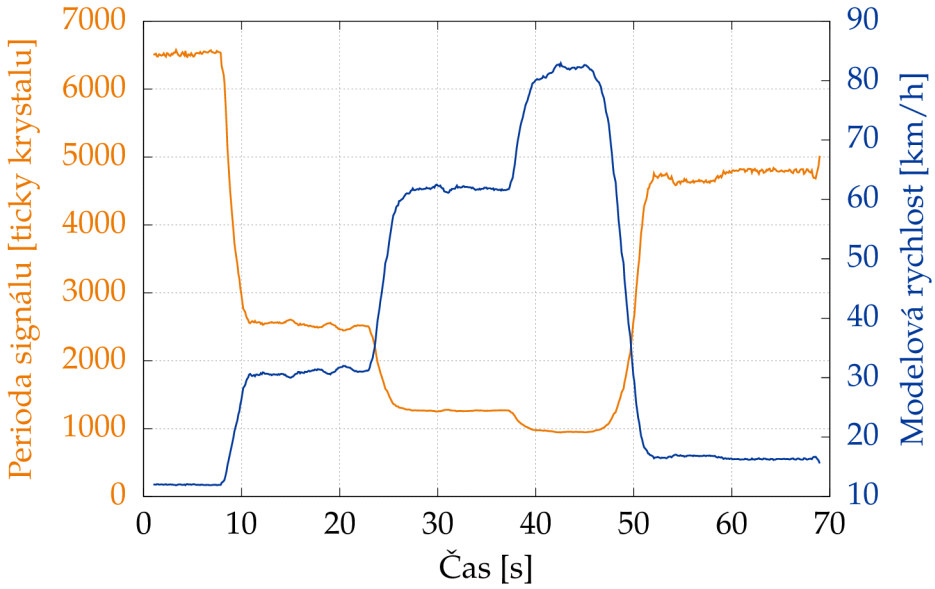
Poslední hypotézou tedy zůstává to, že oscilace, která je patrná v grafu 4.7, je přímým důsledkem postupné aktivace pólů motoru, případně důsledkem nedokonalých převodů lokomotivy a šíření vibrací přes koleje až do vozu.

Z grafu 4.7 je patrné, že měření frekvence oscilace je za hranou možností měřicího vozu. Počet děr v perforovaném kole je totiž na přesné měření frekvence příliš malý. Pravdivost poslední hypotézy tedy autor nebude detailně posuzovat.

Fakt, že amplituda oscilace je při nižších rychlostech vyšší, lze vysvětlit tím, že vozidlo má při nižších rychlostech menší setrvačnost, a tudíž se pulzní charakter regulace a nepřesnosti v převodech více projevuje.

Jako drobnou poznámku autor uvádí, že z naměřených dat nejspíš (!) plyne, že frekvence oscilace se liší u různých vozidel. Měření na různých vozidlech proběhlo, autor ale znovu podotýká, že měření frekvence oscilace je za hranou možností měřicího vozu.

4.11.1 Kompenzace chyb



Obrázek 4.8: Naměřená data za použití průměrovací metody.

Z naměřených dat vyplývá, že v signálu se vyskytuje určitá periodická chyba, kterou by bylo vhodné kompenzovat. Za tímto účelem vznikl návrh průměrovací metody *kluzné okno*. Tato metoda je implementována v produkční verzi kalibračního vozu, princip její činnosti je následující:

1. měř periody ticků ve 100ms rámcích,
2. každých 100 ms přepni na nový rámec,
3. data do počítače průměruj z posledních 10 rámců, tedy celkem z 1 s.

Tato metoda je poměrně známá, zajímavou věcí na implementaci v měřicím vozu jsou výše uvedené konstanty. Sto milisekund proto, aby data v počítači byla vidět v rozumně *reálném čase* a naopak příliš nezahlcovala komunikační kanál. Jedna sekunda proto, že se jedná o latenci, kterou je uživatel ještě schopen tolerovat, navíc za tento čas obvykle nastane alespoň jedna perioda rušivé oscilace signálu (viz naměřená data v grafech výše).

O přesné přepínání oken se v procesoru stará časovač 0.

Průběh naměřené periody a vypočítané rychlosti za použití průměrovací metody zachycuje graf 4.8. Srovnejte s grafem 4.5.

5 Software

Pro potřeby této práce byly vytvořeny 2 aplikace a 2 knihovny. První aplikace je demonstrační aplikací k měřicímu vozu a pouze zobrazuje naměřená data. Druhá aplikace je o mnoho komplexnější a řeší přímo problém automatické kalibrace.

5.1 Volba nástrojů

Zásadní otázkou, kterou je třeba zodpovědět před vytvářením programu, je, jaký programovací jazyk a případně framework zvolit. Zde měl autor poměrně jasný požadavek na programovací jazyk: musí mít poměrně silný typový systém tak, aby netriviální množství kontrol proběhlo již během překladu. Důvodem pro tento argument je fakt, že autor vyrábí aplikaci, které je tzv. *kritická*. V kontextu této práce to znamená, že vlivem chyby v aplikaci může dojít k materiálním škodám na kalibrovaném modelu (ceny vybraných malosériových modelů se pohybují až v desítkách tisíc Kč).

Dalším logickým požadavkem bylo, aby programovací jazyk měl rozumnou podporu pro komunikaci se sériovým portem, neboť toto rozhraní bude použito jak pro komunikaci s měřicím vozem, tak pro komunikaci s digitální centrálou DCC.

Dalším kritériem při výběru programovacího jazyka byla jeho předchozí znalost autorem práce. Zejména u kritických aplikací je totiž nanejvýš důležité, aby programátor přesně věděl, co dělá.

Při zvažování technologií aplikace bylo rozhodnuto, že program by měl mít grafické uživatelské rozhraní (GUI), neboť je nanejvýš vhodné zobrazovat přehledně aktuální průběh kalibrace a neboť jsou běžní uživatelé na GUI aplikace zvyklí.

Zásadní je pak otázka typu aplikace. Autor uvažoval aplikace desktopové, mobilní, webové a dokonce i embedded. Embedded aplikace s sebou nesou tu nevýhodu, že obsluha s aplikací typicky nemůže

příliš interagovat, u mobilních aplikací je v kontextu této práce omezující zejména zobrazovací plocha zařízení, na kterou se jednoduše nevejde tolik dat, kolik by aplikace chtěla zobrazovat. Navíc vývoj pro jiné zařízení, než na kterém se kód píše, vždy přináší nějakou režii navíc.

Výhodou webové aplikace by byla snadná přenositelnost na jiné počítače, aplikace by dokonce mohla být přístupná i z mobilního zařízení. Bohužel, proti tomuto formátu aplikace hovoří fakt, že autor práce nemá s webovými aplikacemi dostatečné zkušenosti, a už vůbec si netroufá v nich psát kritický kód. Oproti desktopovým aplikacím jsou interaktivní webové aplikace navíc o něco složitější (zejména co se týče předávání signálů z GUI).

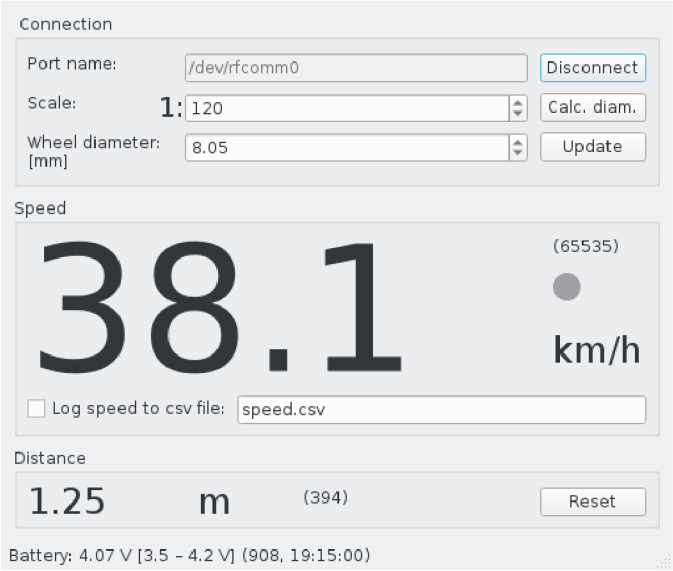
Další možností pak byly desktopové aplikace. Vzhledem k požadavkům uvedeným výše a k faktu, že se jedná už o poněkud rozsáhlejší projekt, jako nejvhodnější se autorovi jevil jazyk `C++` s použitím frameworku `Qt`. Argumentem pro tento framework byly dobré osobní reference, které na `Qt` autor práce dostal od kolegů. Ukázalo se také, že `Qt` poskytují nejen výbornou multiplatformní podporu pro GUI, ale také stejně dobrou multiplatformní podporu pro sériový port. Bylo tedy rozhodnuto, že výsledná aplikace bude psána v programovacím jazyce `C++` (konkrétně `C++14`) za použití frameworku `Qt` a bude multiplatformní (minimálně mezi OS `Windows` a `Linux`).

5.2 WSM Speed Reader

První aplikací je *WSM Speed Reader*[1] – jednoduchý zobrazovač vyčtených dat. Aplikace se skládá z jednoho okna a je především typu *proof of concept*. GUI aplikace je zachyceno na obrázku 5.1.

Aplikace se připojuje k sériovému portu, ze zadaných parametrů vypočítává rychlost a ujetou vzdálenost vozidla a tyto údaje zobrazuje. Umožňuje ukládání vyčtených dat do souboru (především pro vývoj) a ukládání nastavení aplikace do konfiguračního souboru. Aplikace je volně dostupná pod opensource licencí *Apache License v2.0* [34].

1. WSM je obchodní název vyvinutého měřicího vozu: *WSM: A Wireless Speedometer*.



Obrázek 5.1: GUI programu WSM Speed Reader.

Při programování této aplikace bylo rozhodnuto vyčlenit knihovnu umožňující komunikaci s měřicím vozem WSM do samostatného repozitáře. Knihovna se skládá z hlavičkového souboru `wsm.h`, který definuje API třídy `Wsm`, implementace této knihovny (v samostatném cpp souboru) a pomocných souborů. Knihovna je k dispozici online[2], bylo k ní vytvořeno srozumitelné README a dodány všechny náležitosti samostatně stojícího projektu.

I přes původní snahy je nutné pro užití knihovny v projektu využít frameworku `Qt`, a to zejména kvůli implementaci sériového portu.

2. `https://github.com/kmzbrnoI/wsm-lib-cpp-qt`

5.3 Automatic Calibration

Program *Automatic Calibration* umožňuje automaticky provést proces kalibrace. V souladu s požadavky kladenými na výsledné softwarové řešení umožňuje

- připojení k XpressNET centrále a měřicímu vozu WSM,
- ruční řízení libovolného hnacího vozidla,
- vyčítání základních CV (viz 1.2) lokomotivy,
- načtení závislosti *jízdní stupeň – rychlost* ze souboru[3],
- plně automatickou kalibraci rychlostní tabulky,
- poloautomatickou kalibraci hodnoty decelerace,
- konfiguraci parametrů kalibrace konfiguračním souborem,
- zobrazení průběhu kalibrace, přerušení a obnovení kalibrace,
- načtení a uložení profilu lokomotivy do souboru,
- zobrazení grafu závislosti rychlosti na otáčkách motoru.

Grafické uživatelské rozhraní aplikace je zachyceno na obrázku 5.2.

Program je k dispozici pod opensource licencí *Apache License v2.0* online[4].

Pro pochopení následujícího textu je třeba striktně rozlišovat několik pojmů.

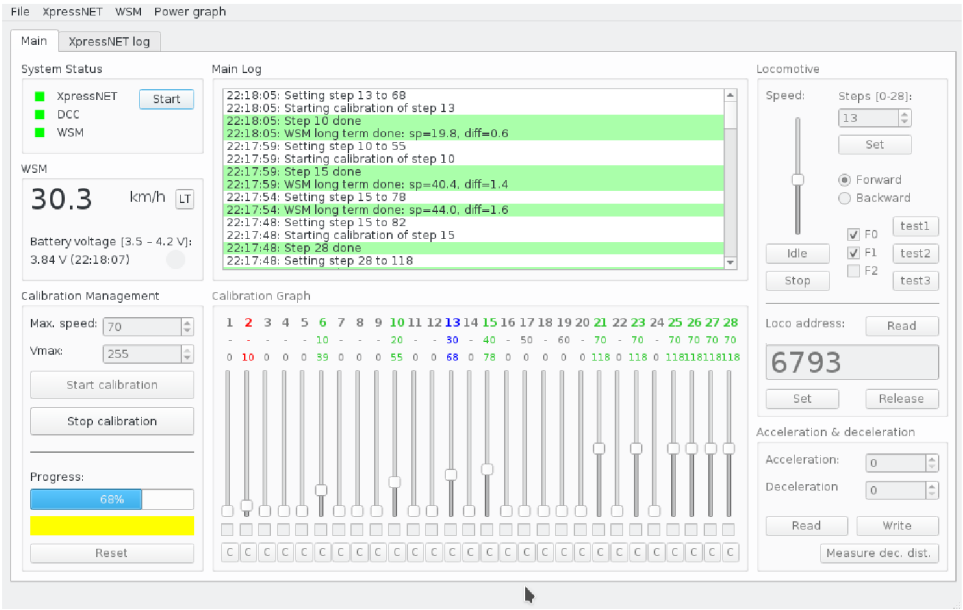
1. *Jízdní stupeň*

   Jízdní stupeň je hodnota v rozsahu 0–28. Tuto hodnotu odesílá počítač (nebo ovladač) do digitální centrály DCC během ostrého provozu na kolejišti. Centrála jízdní stupeň přeposílá do lokomotivy. Nastavením jízdního stupně se řídí rychlost lokomotivy.

2. *Otáčky přiřazené konkrétnímu jízdnímu stupni*

   Otáčky motoru přiřazené konkrétnímu jízdnímu stupni se ukládají do CV (viz 1.2) dekodéru a mají mají rozsah 0–255. Pro každý jízdní stupeň je právě jedno CV. Nastavení těchto CV lze měnit

3. Tato závislost je úzus a je typicky jediná v rámci modelářského klubu či většího celku. V KMŽ Brno I je používána závislost definovaná v tabulce A.1.
4. `https://github.com/kmzbrnoI/automatic-calibration`



Obrázek 5.2: Hlavní okno programu Automatic Calibration.

jedině programováním dekodéru a neočekává se, že se při ostrém provozu bude do tohoto nastavení zasahovat. Více jízdních stupňů může mít přiřazeno stejné otáčky.

3. *Rychlost*

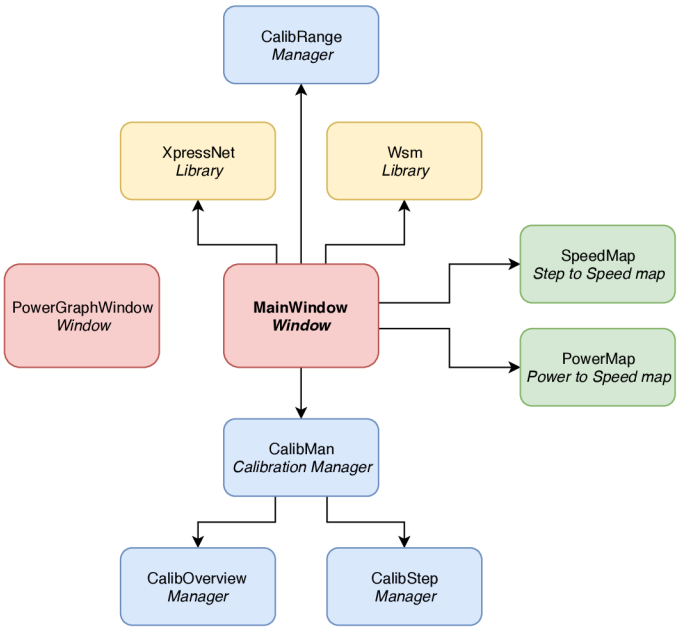
   Rychlost je skutečná rychlost modelu měřená např. měřicím vozem. Rychlost je svázaná s otáčkami motoru, více jízdních stupňů může mít nastavené stejné otáčky (a tudíž stejnou rychlost).

Veškeré programování dekodéru v lokomotivě probíhá v režimu *POM* (*Programming on Main*), který umožňuje – ačkoliv se nepředpokládá, že se tato možnost využije – programování za plného provozu kolejiště.[5] Režim *POM* byl zvolen zejména z toho důvodu, že při programování není lokomotivu nutné zastavovat (oproti programování

5. DCC centrála umí programovat dekodér ve dvou režimech: (1) programování na programovací koleji, (2) *Programming on Main – POM*. Jedině na programovací koleji lze vyčítat CV, při zvolení režimu *POM* není možné přenášet data z lokomotivy zpět do centrály, CV lze tedy pouze nastavovat. Lokomotiva ani nemůže potvrdit správné přijetí dat. Centrála ale předem počítá s případnou nespolehlivostí přenosu dat a proto posílá *POM* příkaz dekodéru několikrát. V praxi se problém chybějící

na programovací koleji), takže kalibrace rychlostí může být výrazně rychlejší.

Aplikace je vystavěna objektově. Je zde několik tříd založených na modelu *singleto*, kde každá třída má jednu konkrétní odpovědnost. Diagram tříd je zobrazen na obrázku 5.3, šipky reprezentují vztah *vlastní*.



Obrázek 5.3: GUI programu WSM Speed Reader.

Kořenovou třídou je třída obsluhující hlavní okno (`MainWindow`), v projektu jsou dále knihovny umožňující komunikaci s periferiemi (žluté třídy), nad těmito knihovnami jsou správci kalibrace (modré třídy). V projektu jsou dále pomocné třídy (zelené obdélníky), které jsou využívány v prakticky celé aplikaci. Projekt je navržen tak, aby pouze červené třídy byly propojené s GUI, zbylé třídy mezi sebou komunikují pomocí volání funkcí a Qt signálů.

Třída `SpeedMap` udržuje mapu *jízdní stupeň: žádaná rychlost*, třída `PowerMap` pak udržuje mapu *otáčky motoru: naměřená rychlost*.

---

odezvy neukazuje zásadní. Při programování na programovací koleji typicky není možný provoz na kolejišti.

O kalibraci rychlostních křivek se stará třída `CalibMan`. Kalibrace je rozdělena do čtyř fází:

1. zapsání základních CV,
2. vytvoření základního přehledu o závislosti *otáčky motoru: rychlost*,
3. kalibrace jednotlivých jízdních stupňů,
4. interpolace zbylých jízdních stupňů.

Kalibraci je kdykoliv možné přerušit, čímž dojde k zastavení právě prováděné operace a k zastavení hnacího vozidla. Protože si ale program pamatuje svůj stav, je možné kalibraci kdykoliv spustit znovu a již dokončené fáze jsou přeskočeny.

Před začátkem kalibrace je nutné mít v SW načtenou mapu *jízdní stupeň: žádaná rychlost* (viz přílohu A.1). Tato mapa je načítána z `csv` souboru a je možné definovat v ní všechny či pouze vybrané jízdní stupně. Zbylé jízdní stupně se interpolují ze sousedních nakalibrovaných hodnot nebo je nastaví uživatel manuálně. Tato mapa je dohodnutou vlastností kolejiště a typicky se nemění, proto se načítá z jediného souboru.

Před kalibrací navíc uživatel zadá *maximální kalibrovanou rychlost*. Tato rychlost je vlastností konkrétního hnacího vozidla, proto je možné ji ovlivnit před každou jednotlivou kalibrací. Maximální rychlost spolu s akcelerací a decelerací a oběma mapami jsou ukládány do `xml` souboru definice hnacího vozidla. Formát tohoto souboru byl záměrně převzat ze SW JMRI [2], aby bylo možné soubory mezi programy přenášet.

Další parametry, jako je například umístění zařízení sériového portu centrály či maximální povolená odchylka kalibrace, jsou načítány z jediného konfiguračního souboru formátu `ini`.

5.3.1 Měření rychlosti

Měření rychlosti v průběhu kalibrace probíhá ve speciálním režimu *LT – Long Term Measure*. Tento speciální režim je implementován v knihovně pro komunikaci s měřicím vozem a umožňuje průměrování více

naměřených rychlostí. Lze si všimnout, že autor právě představuje druhou vrstvu průměrování dat z měřicího vozu (první probíhá již přímo ve vozu). Ovšem cílem průměrování na úrovni SW není pouze eliminovat oscilace rychlosti, výstupem je totiž kromě průměrné rychlosti také rozdíl $v_{max} - v_{min}$ v měřeném intervalu. Tento „pseudorozptyl“ umožňuje kalibračnímu SW rozhodnout, kdy je a kdy není měření relevantní. A přesně to kalibrační SW dělá.

Průměrování typicky probíhá pro 30 naměřených rychlostí, tedy po dobu 3 s.

Kdykoliv je rozdíl $v_{max} - v_{min}$ naměřených rychlostí příliš velký (konstanta je konfigurovatelná, výchozí hodnota je 3 km/h), je měření ignorováno a inicializováno měření nové. Pokud se nepovede změřit rychlost několikrát (opět konfigurovatelná konstanta), proces kalibrace je přerušen. Velký rozptyl rychlostí může být způsoben například nekvalitním vozidlem (a takové vozidlo prostě odmítneme zkalibrovat) nebo například členitostí trati – přítomností oblouků, stoupání a klesání, s kterými se BEMF nevyrovná.

5.3.2 Fáze kalibrace rychlosti

Zapsání základních CV

V této krátké fázi program nastaví do lokomotivy základní CV. Vypne rozjezdy a dojezdy a nastaví škálovací parametry rychlostních křivek[6] tak, aby hodnoty otáček 0–255 reálně pokrývaly celé spektrum stříd signálů do motoru.

O tento krok se stará přímo třída `CalibMan`.

6. Některé dekodéry umožňují kromě přiřazení otáček každému stupni navíc zadat parametr, který určuje, jaké reálné střídě signálu odpovídá maximální hodnota otáček. Tím chtějí umožnit škálovat celou rychlostní tabulku bez nutnosti změnit otáčky odpovídající každému jízdnímu stupni.

Vytvoření přehledu

O tento krok se stará třída `CalibOverview`. Lokomotivě je nastaven jízdní stupeň 2 a na tento stupeň je zapsána hodnota otáček 10. Následně proběhne měření rychlosti. Dokud je rychlost lokomotivy neměřitelná (příliš malá), otáčky jsou postupně zdvojnásobovány a do mapy *otáčky: rychlost* jsou přidávány nové záznamy. Jakmile se lokomotiva rozjede na rychlost $> V_{min}$, program má informaci o tom, která hodnota otáček motoru odpovídá minimální rychlosti lokomotivy, a přechází do další fáze.

V další fázi zkouší program otáčky v intervalu 64–255, dokud nenaměří rychlost vyšší, než je uživatelem zadaná maximální rychlost.

Nyní má program představu o tom, které hodnotě otáček motoru odpovídá minimální rychlost a kterému maximální rychlost. Všechna měření rychlosti lokomotivy probíhají v režimu *LT* (viz kapitolu 5.3.1).

Pokud uživatel načte profil již změřené lokomotivy ze souboru a v tomto profilu jsou body, které jsou vyžadovány, tato fáze je přeskočena. Kalibrace vozidla, které už jednou bylo kalibrováno, je tedy rychlejší.

Kalibrace jízdních stupňů

Třída `CalibStep` umožňuje zkalibrovat jeden jízdní stupeň. Třída `CalibMan` pak postupně volá třídu `CalibStep`, dokud nejsou zkalibrovány všechny jízdní stupně, které si uživatel přál zkalibrovat.

Kalibrace jednoho stupně probíhá následovně.

1. V mapě *otáčky: rychlost* jsou vyhledány (interpolovány) otáčky odpovídající kýžené rychlosti.
2. Otáčky jsou zapsány do lokomotivy k právě kalibrovanému stupni.
3. Čeká se 2 s na změnu rychlosti lokomotivy.
4. Začne měření rychlosti v režimu *LT*.
5. Naměřená hodnota je přidána do mapy *otáčky: rychlost*.

6. Pokud je rychlost v ϵ toleranci (konstantní konfigurovatelná hodnota) cílové rychlosti, krok je zkalibrován, jinak se přechází znovu na krok (1)[7].

ϵ je ve výchozím nastavení 1 km/h.

Jakmile je zkalibrována rychlost, která je přiřazena více jízdním stupňům, jsou i tyto jízdní stupně nastaveny na právě zkalibrovanou hodnotu otáček.

Průběh kalibrace je průběžně zobrazován uživateli. Uživatel vidí, které jízdní stupně jsou již zkalibrovány a které ještě ne.

Interpolace

Po dokončení kalibrace všech stupňů načtených z `csv` souboru jsou zbylé stupně interpolovány.

Po této fázi zůstává typicky nezkalibrována minimální rychlost[8], pro kterou v *Klubu modelářů železnic Brno I* platí úzus, že je to taková nejnižší rychlost, při které se hnací vozidlo již začne plynule pohybovat. Po dokončení kalibrace tedy uživatel ručně vybere jízdní stupeň jedna a nastaví mu požadované otáčky za současného pozorování pohybu vozidla.

Program se pak postará o to, aby byly stupně do nejbližšího již zkalibrovaného stupně interpolovány.

5.3.3 Kalibrace dojezdů

Po zkalibrování rychlosti následuje kalibrace dojezdů. Zde se autor nakonec rozhodl pro poloautomatickou kalibraci, a to zejména z toho důvodu, že kalibrace dojezdů je již velmi rychlá.

Program uživateli umožní zadat hodnotu decelerace (CV #4) a pak pomocí kliku na jediné tlačítko změřit dojezd z rychlosti 40 km/h.

---

7. Všimněte si, že v následující iteraci je zapsána jiná (přesnější) hodnota otáček, neboť došlo ke zpřesnění mapy *otáčky: rychlost*.
8. Měřicí vůz neumí dobře měřit hodně malé rychlosti.

Program tedy vozidlo rozjede, chvíli počká na ustálení rychlosti, pak vozidlu nastaví jízdní stupeň 0 a pomocí měřicího vozu měří dráhu do zastavení. Vzdálenost pak vypíše uživateli, který na základě ní může upravit hodnotu decelerace.

Akcelerace je pak typicky nastavena stejně jako decelerace (ale volba je plně na uživateli).

5.4 XpressNET knihovna

Fundamentálním nástrojem pro fungování aplikace *Automatic Calibration* je knihovna, která umožňuje komunikaci s centrálou XpressNET. Přestože je tento protokol ve střední Evropě hojně rozšířen, pro jazyk C++ autor nenašel žádnou použitelnou knihovnu, která by umožňovala s XpressNET centrálou komunikovat.

Autor tedy implementoval vlastní opensource a robustní XpressNET knihovnu s potvrzováním příkazů z centrály a vyčlenil ji do samostatného repozitáře[9]. Autor pevně věří, že si tato knihovna najde uplatnění napříč modelářskou komunitou.

9. `https://github.com/kmzbrnoI/xn-lib-cpp-qt`

6 Závěr

Tato práce ukázala, že k automatizaci procesu kalibrace modelového železničního vozidla je nutné zvládnout širokou škálu dílčích kroků od efektivního návrhu elektroniky přes programování mikrokontrolérů, studování bezdrátových modulů a komunikačních protokolů až po schopnost kvalitně vyvíjet desktopové aplikace. Autorovi se úspěšně povedlo navrhnout vlastní senzor pro měření rychlosti založený na optozávoře a perforovaném kolu. Podařilo se mu zabezpečit přenos naměřených dat do počítače pomocí univerzálního protokolu Bluetooth a demonstrovat fungování přenosu. Jeho práci pak dovršila kompletně vlastní aplikace, která umožňuje proces kalibrace realizovat pohodlně a spolehlivě. Kromě toho autor vytvořil knihovnu, která je univerzálně použitelná pro komunikaci s libovolnou XpressNET centrálou.

Při konstrukci měřicího vozu vyvstalo několik problémů a omezení souvisejících s malou velikostí prostoru a limitovanými možnostmi napájení. Ukázalo se, že je možné minimalizovat spotřebu zařízení tak, aby jej bylo možné napájet z baterie. Napájení z baterie se nakonec ukázalo jako nanejvýš výhodné a jednoduché.

Bylo zjištěno, že rychlost vozidla je poměrně zásadně ovlivňována nejrůznějšími parazitickými jevy od pružení ve spřáhle přes odpor v oblouku až po mechaniku převodů v lokomotivě. Tyto jevy se autorovi povedlo úspěšně odstínit za použití průměrování.

Tato práce jednoznačně pomohla k zefektivnění a zpřesnění procesu kalibrace lokomotivy, který místo 25 minut za využití jednoho člověka trvá 4 minuty bez nutnosti přítomnosti obsluhy.

6.1 Možná rozšíření

Při návrhu senzoru byl brán v potaz požadavek na možnost dokalibrovávat lokomotivy za ostrého provozu na kolejišti. Nabízí se tedy možnost prozkoumat, jak moc se rychlosti lokomotiv vlivem provozu

a opotřebení mění, a za využití kalibračního vozu navrhnout mechanismus, kterým budou hnací vozidla dokalibrována za provozu, nejlépe ve spolupráci s řídicím SW kolejiště.

Při konstrukci měřicího vozu je také možno prozkoumat různé alternativní cesty – například se upnout na napájení z kolejí třeba i za cenu výměny podvozku, implementovat přenos dat počítače alternativními metodami apod.

Dalším směrem výzkumu je měření příčin oscilace rychlosti a návrh aktivního potlačení těchto oscilací. Systém efektivního potlačování výkyvů rychlostí by sice nejspíš byl výpočetně náročný (Fourierova transformace, konvoluční metody), ale vzhledem k výkonům současných počítačů uskutečnitelný. A to možná i v reálném čase. Uživateli by tak byla nabídnuta vysoce přesná data zbavená šumu prezentovaná v reálném čase. To by mohlo přispět ke zrychlení všech procesů založených na měření rychlosti měřicím vozem.

Bibliografie

1. *hJOP – Komplexní řízení modelového kolejiště dle ZTP-JOP* [online]. 2018 [cit. 2018-03-24]. Dostupné z: http://hjop.kmz-brno.cz/.
2. *JMRI: A Java Model Railroad Interface* [online]. 2018 [cit. 2018-05-25]. Dostupné z: http://jmri.sourceforge.net/.
3. *Freiwald Software: TrainController* [online]. 2018 [cit. 2018-05-25]. Dostupné z: http://www.freiwald.com/pages/traincontroller.htm.
4. *modelJOP: software pro řízení provozu digitálního kolejiště dle ZTP-JOP* [online]. 2018 [cit. 2018-05-25]. Dostupné z: http://www.modeljop.cz/.
5. *Z21: digital model railway control system* [online]. 2018 [cit. 2018-05-25]. Dostupné z: http://www.z21.eu/.
6. *Paco's Official Web Site: NanoX* [online]. 2018 [cit. 2018-05-25]. Dostupné z: http://usuaris.tinet.cat/fmco/download_en.html.
7. *Uhlenbrock Elektronik: Intellibox II* [online]. 2018 [cit. 2018-05-25]. Dostupné z: https://www.uhlenbrock.de/de_DE/produkte/digizen/I000C67D-001.htm.
8. NATIONAL MODEL RAILROAD ASSOCIATION, INC. *Electrical Standards for Digital Command Control* [online]. 2018 [cit. 2018-10-23]. Dostupné z: https://www.nmra.org/sites/default/files/standards/sandrp/pdf/s-9.1_electrical_standards_2006.pdf.
9. *CV List of Zimo Decoders* [online]. 2018 [cit. 2018-10-16]. Dostupné z: http://www.zimo.at/web2010/documents/cvliste_en.pdf.
10. *ACCUTRACK II Speedometer* [online]. 2018 [cit. 2018-03-24]. Dostupné z: http://www.sprog.us.com/speedo.html.
11. *BD20 Block Detector* [online]. 2018 [cit. 2018-03-24]. Dostupné z: https://ncedcc.zendesk.com/hc/en-us/articles/201254549-BD20-Block-Detector.
12. *JMRI Speedometer* [online]. 2018 [cit. 2018-03-24]. Dostupné z: http://jmri.sourceforge.net/help/en/html/tools/speedometer/NCE-Speedometer.shtml.

13. *Bachrus: Model Train Speedometer* [online]. 2018 [cit. 2018-10-21]. Dostupné z: http://www.bbmgroup.com/sprog/bachrus.html.

14. AMS. *AS5306 Linear Sensor* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://ams.com/as5306.

15. *Multipole Radial Ring Magnets* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://allstarmagnetics.com/multipole-radial-ring-magnets/.

16. AVAGO TECHNOLOGIES. *ADNS-3050: Entry-level Gaming Optical Navigation Sensor* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://media.digikey.com/pdf/Data%20Sheets/Avago%20PDFs/ADNS-3050.pdf.

17. SHARP. *GP1S23 Datasheet* [online]. 2018 [cit. 2018-09-26]. Dostupné z: http://pdf.datasheetcatalog.com/datasheet/Sharp/mXuuxzu.pdf.

18. *KY-010 Photo Interrupter Module* [online]. 2018 [cit. 2018-10-23]. Dostupné z: https://tkkrlab.nl/wiki/Arduino_KY-010_Optical_broken_module.

19. NATIONAL MODEL RAILROAD ASSOCIATION, INC. *Communications Standards For Digital Command Control* [online]. 2018 [cit. 2018-10-23]. Dostupné z: https://www.nmra.org/sites/default/files/s-92-2004-07.pdf.

20. DCC WIKI. *RailCom* [online]. 2018 [cit. 2018-11-11]. Dostupné z: https://dccwiki.com/Term:RailCom.

21. ESPRESSIF SYSTEMS. *ESP32 Overview* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://www.espressif.com/en/products/hardware/esp32/overview.

22. ESPRESSIF SYSTEMS. *ESP32 Series Datasheet* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

23. *Serial Port Profile* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/bluetooth_info/k5_spp.html.

24. MICROCHIP. *Microchip AVR® MCUs* [online]. 2018 [cit. 2018-09-26]. Dostupné z: http://www.microchip.com/design-centers/8-bit/avr-mcus.

25. *Arduino* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://www.arduino.cc/.

26. *TT - klanicový vůz Nd, ČSD / Tillig 14662* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://www.itvlaky.cz/2-ose-kratke/tt-klanicovy-vuz-nd--csd-tillig-14662/.

27. ATMEL. *ATmega328/P Complete Datasheet* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://tinyurl.com/atmega328-datasheet-full.

28. *LE33: Very low-dropout voltage regulator with inhibit function* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://www.st.com/resource/en/datasheet/cd00000545.pdf.

29. *Míčkoflus – sestřel je všechny!* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://mickoflus.cz/.

30. *Robotika Brno* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://www.robotikabrno.cz/.

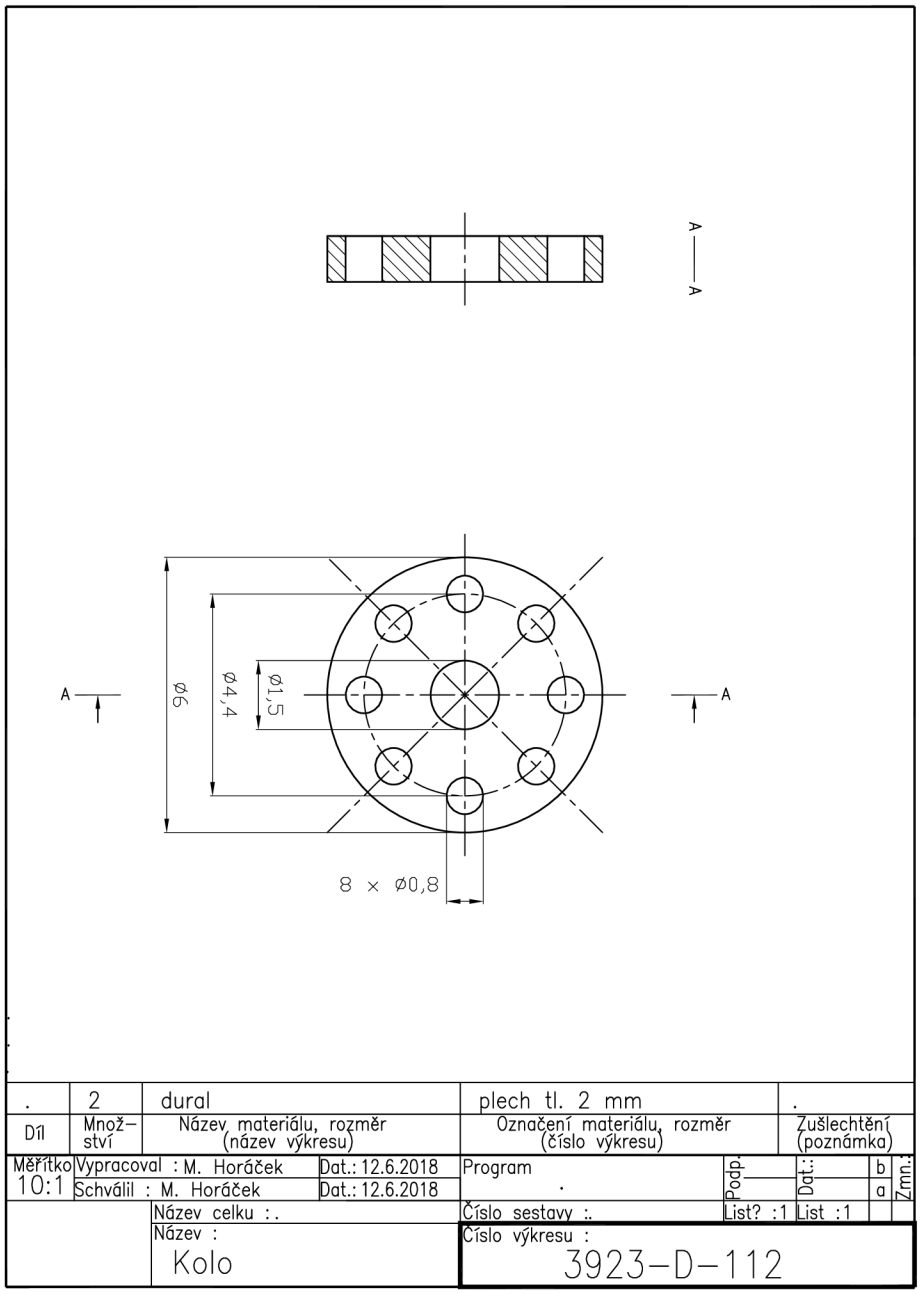
31. HORÁČEK, Jan. *A Wireless Speedometer Printed Circuit Board* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://github.com/kmzbrnoI/wsm-pcb.

32. LENZ ELEKTRONIK GMBH. *XpressNet Specification v3* [online]. 2018 [cit. 2018-09-26]. Dostupné z: http://www.lenzusa.com/1newsite1/Manuals/xpressnet.pdf.

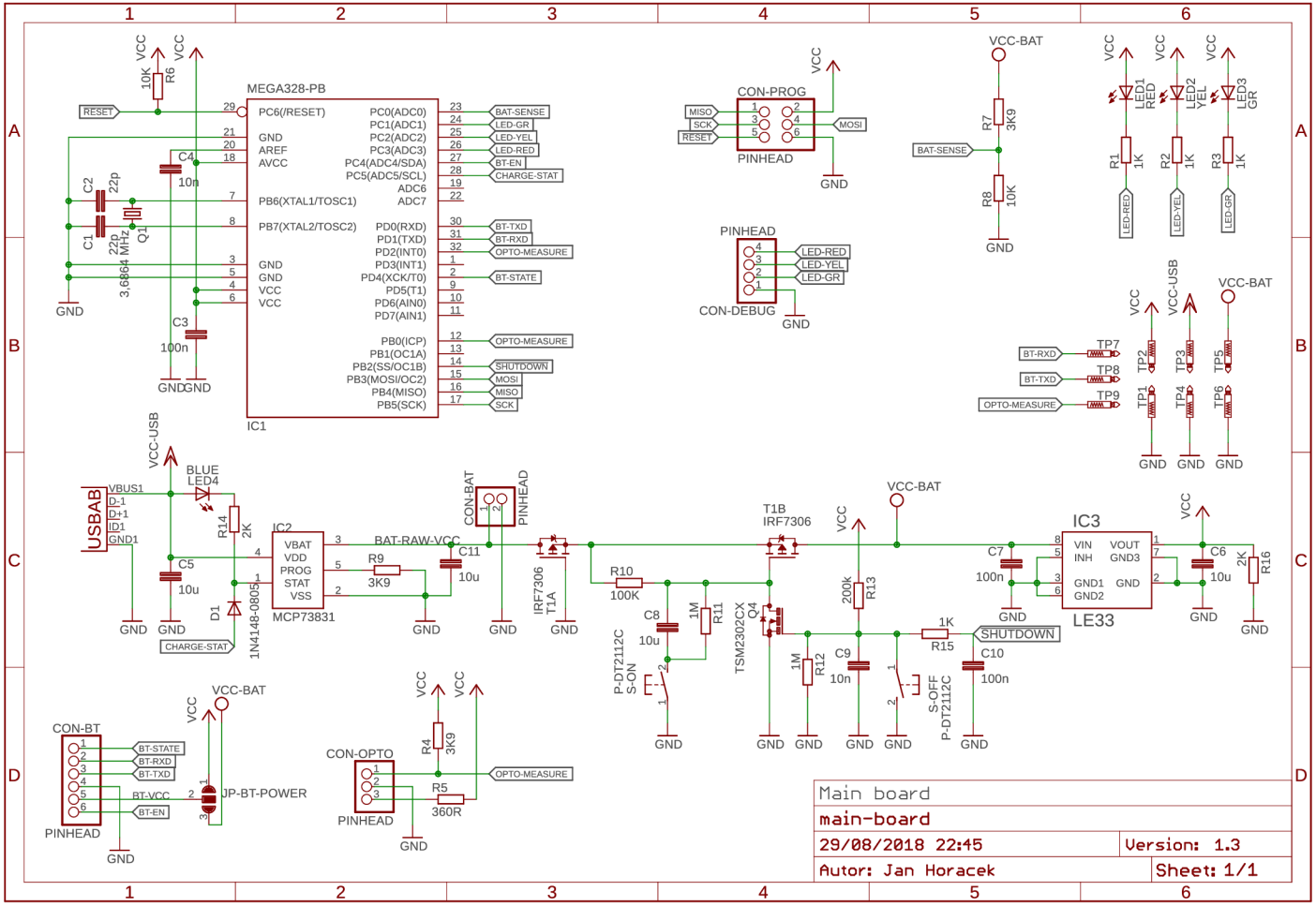
33. HORÁČEK, Jan. *A Wireless Speedometer Firmware* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://github.com/kmzbrnoI/wsm-fw.

34. HORÁČEK, Jan. *A Wireless Speedometer Speed Reader* [online]. 2018 [cit. 2018-09-26]. Dostupné z: https://github.com/kmzbrnoI/wsm-speed-reader.

A Přílohy



Obrázek A.1: Výkres perforovaného kola. Autor: Miroslav Horáček.



Obrázek A.2: Schéma hlavní DPS měřicího vozu.

Tabulka A.1: Rychlostní tabulka dle standardu KMŽ Brno I, vybrané kalibrované kroky jsou vyznačeny tučně, zbytek je interpolován.

Jízdní stupeň	Rychlost [modelový km/h]
1	1
2	2
3	4
4	6
5	8
6	**10**
7	12
8	15
9	17
10	**20**
11	22
12	25
13	**30**
14	35
15	**40**
16	45
17	**50**
18	55
19	**60**
20	65
21	**70**
22	75
23	**80**
24	85
25	**90**
26	**100**
27	**110**
28	**120**