VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VESTAVĚNÁ ŘÍDICÍ JEDNOTKA PRO OVLÁDÁNÍ LABORATORNÍHO ZAŘÍZENÍ

EMBEDDED CONTROL UNIT FOR INSTRUMENTATION OF LABORATORY APPLIANCE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. ZBYŠEK VODA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VÁCLAV ŠIMEK**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Voda Zbyšek, Bc.**
Program: Informační technologie Obor: Počítačové a vestavěné systémy
Název: **Vestavěná řídicí jednotka pro ovládání laboratorního zařízení**
Embedded Control Unit for Instrumentation of Laboratory Appliance
Kategorie: Vestavěné systémy
Zadání:

1. Seznamte se s principem činnosti specializovaného laboratorního zařízení pro analýzu léčiv. Zaměřte se na použité typy aktuátorů, senzorů a způsob jejich ovládání.
2. Detailně prostudujte stávající koncepci řídicí jednotky pro uvedené zařízení a formát komunikačního protokolu pro interakci s uživatelským rozhraním na PC.
3. S výzkumným týmem z VFU Brno konzultujte požadavky týkající se inovace řídicí jednotky přístroje a návaznosti na uživatelský software. Proveďte detailní analýzu možností jejich realizace.
4. Na základě zjištěných poznatků navrhněte modulární koncepci řídicí jednotky, která bude zajišťovat všechny potřebné funkce a současně nabídne možnosti snadného rozšíření.
5. Zvolte vhodné komponenty pro realizaci řídicí jednotky a připravte schéma jejího zapojení na obvodové úrovni. Následně vytvořte desku plošných spojů a proveďte její oživení.
6. Vytvořte obslužný firmware řídicí jednotky, který bude zajišťovat interakci s aktuátory a senzory laboratorního přístroje. Dbejte na jeho modulární charakter a možnost doplnění o další funkce dle potřeby.
7. Vytvořené řešení detailně otestujte v reálných podmínkách. Zhodnoťte dosažené výsledky a navrhněte případná rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 4 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Šimek Václav, Ing.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 25. října 2019

Abstrakt

Tato práce se zabývá návrhem modulárního řídicího systému pro zařízení Golem používaného pro výzkum léků. Toho je docíleno rozdělením systému do modulů, komunikujících po CAN sběrnici, které mají přesně danou oblast, jež řídí. Pro řízení systému byl využit počítač Raspberry Pi 4. Součástí práce je design desek plošných spojů, obslužný firmware jednotlivých modulů, backendová aplikace pro řízení systému, poskytující aplikační rozhraní pro řízení systému, a také návrh jednoduchého komunikačního protokolu pro komunikaci mezi moduly. Navržený systém byl důkladně otestován v simulovaných podmínkách.

Abstract

Aim of this thesis is modular design of control system for Golem device, which is used for drug research. Several modules with specified controlled area were designed. They communicate using CAN bus and are controlled by Raspberry Pi computer. The thesis includes PCB design and control firmware for modules, backend application for Raspberry Pi, which provides application interface, and simple protocol used for communication between modules. Designed system was tested in simulated environment.

Klíčová slova

disoluce, léčiva, řízení, raspberry pi, modulární, deska plošných spojů, kinetis, sběrnice CAN

Keywords

disolution, drugs, control systems, raspberry pi, modular, printed circuit board, kinetis, CAN bus

Citace

VODA, Zbyšek. *Vestavěná řídicí jednotka pro ovládání laboratorního zařízení.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Václav Šimek

Vestavěná řídicí jednotka pro ovládání laboratorního zařízení

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Václava Šimka. Další informace, týkající se zařízení Golem a procesů, které zkoumá, mi poskytli vědečtí pracovníci z Veterinární a farmaceutické univerzity Brno, jmenovitě PharmDr. Martin Čulen, Ph.D., Mgr. Tomáš Bílik a PharmDr. Jakub Vysloužil, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Zbyšek Voda
10. června 2020

Poděkování

Rád bych poděkoval vedoucímu mé práce, Ing. Václavu Šimkovi, za cenné rady a vedení v průběhu realizace práce. Dále děkuji Bc. Janu Truhlářovi, který ve své diplomové práci vytvořil uživatelské rozhraní pro moji práci a se kterým jsme úzce spolupracovali po celou dobu realizace našich prací. Také děkuji vědeckým pracovníkům z Veterinární a farmaceutické univerzity Brno za poskytnutí informací o současném stavu zařízení Golem a požadavků na jeho vylepšení. V neposlední řadě děkuji Mgr. Michaele Mařánové za podporu, péči a psychickou oporu při tvorbě práce.

Obsah

1 Úvod **4**

2 Disoluční zařízení Golem a jeho využití **5**
2.1 Golem 1 . . . 7
2.2 Trávicí soustava člověka . . . 7
2.2.1 Dutina ústní . . . 7
2.2.2 Hltan . . . 9
2.2.3 Jícen . . . 9
2.2.4 Žaludek . . . 9
2.2.5 Tenké střevo . . . 9
2.2.6 Tlusté střevo . . . 10
2.3 Lékové formy . . . 10
2.3.1 Dělení dle uvolňování léčivé látky . . . 10

3 Stávající koncepce řídicího systému **12**
3.1 Varianty zařízení Golem . . . 12
3.2 Subsystémy zařízení Golem 2 . . . 13
3.2.1 Systém regulace teploty . . . 14
3.2.2 Reakční vaky . . . 15
3.2.3 Míchání reakčních vaků . . . 15
3.2.4 Měření a regulace pH . . . 16
3.2.5 Vstřikování enzymu . . . 19
3.2.6 Peristaltické pumpy . . . 19
3.2.7 Napájení . . . 19
3.2.8 Rozšiřující obvody pro Arduino UNO . . . 19
3.3 Komunikační protokol . . . 21
3.3.1 Řízení pístových pump Cavro XE1000 . . . 22
3.3.2 Řízení ventilů Cavro Smart Valve . . . 24
3.3.3 Řízení obvodů míchání . . . 24
3.3.4 Řízení modulu Arduino . . . 24
3.4 Řídicí aplikace v PC . . . 25
3.4.1 Hlavní obrazovka aplikace . . . 25
3.4.2 Specifikace experimentů . . . 26
3.4.3 Výstupní textový záznam . . . 26

4 Nové disoluční zařízení Golem 3 **27**
4.1 Aspekty modulárního přístupu . . . 27
4.2 Výběr komunikační sběrnice . . . 29

4.3 Sběrnice CAN . . . 30
4.4 Komunikační protokol . . . 32
4.5 Rozdělení systému do modulů . . . 33
4.6 Výběr cílové platformy . . . 33

5 Specifikace funkčních bloků 35
5.1 Společný základ modulů . . . 35
5.1.1 Mikrokontrolér . . . 35
5.1.2 Zdroj napětí . . . 37
5.1.3 CAN . . . 37
5.1.4 Stavová LED . . . 41
5.2 Časovač, měření času, PWM . . . 42
5.3 Ovládání vstupů a výstupů . . . 42
5.4 PID regulátor . . . 42
5.5 UART . . . 44
5.6 Měření teploty . . . 44
5.6.1 OneWire sběrnice . . . 45
5.6.2 DS18B20 . . . 45
5.7 Topení . . . 46
5.8 Senzor síly na tabletu . . . 46
5.9 Řízení krokových motorů . . . 49
5.9.1 Koncový spínač . . . 49
5.9.2 TMC2209 . . . 50
5.10 Měření pH . . . 52
5.10.1 Zdroj referenčního napětí 1 V . . . 52
5.10.2 Analogový frontend . . . 52
5.10.3 Zesílení signálu z analogového frontendu . . . 54
5.10.4 Softwarové zpracování . . . 54
5.11 Pumpy a ventily Cavro . . . 56
5.11.1 RS485 . . . 56
5.11.2 Připojení a napájení . . . 56
5.11.3 Software pro řízení . . . 57
5.12 Peristaltické pumpy . . . 58

6 Realizace systémových modulů 59
6.1 Prototyp . . . 59
6.2 Společný základ modulů . . . 60
6.2.1 Architektura firmware . . . 60
6.2.2 Stavy modulu . . . 61
6.2.3 Společné CAN příkazy . . . 61
6.3 Vstřikování enzymu . . . 62
6.4 Regulace teploty . . . 64
6.5 Řízení míchání a přečerpávání vaků . . . 66
6.6 Regulace pH ve vacích . . . 68
6.7 Master zařízení, převodník UART-CAN . . . 70
6.8 Umístění a propojení modulů . . . 71

7 Softwarové komponenty řídicí jednotky 73

7.1 Webová aplikace . . . 73
7.2 Vyhledání Raspberry Pi v síti . . . 73
7.3 Řídicí aplikace . . . 73
7.3.1 Aplikační rozhraní . . . 74
7.3.2 Persistence dat . . . 74
7.3.3 Správa uživatelů . . . 75
7.3.4 Vytváření záloh . . . 75
7.3.5 Kalibrace pH sond . . . 75
7.3.6 Nastavení simulací . . . 75
7.3.7 UART . . . 76
7.3.8 Řízení modulů . . . 76
7.3.9 Třída Golem . . . 77
7.3.10 Třída System . . . 77

8 Kalibrace a ladění částí systému 79
8.1 Kalibrace pH sond . . . 79
8.2 Kalibrace teploměrů . . . 80
8.3 Kalibrace senzorů síly na tabletu . . . 80
8.4 Kalibrace peristaltických čerpadel . . . 81
8.5 Ladění PID regulátoru teploty . . . 83
8.6 Ladění PID regulátoru pH . . . 83

9 Testování systému 84

10 Závěr 85

Literatura 86

A Přiřazení pinů pro řízení funkčních bloků 90

B Specifikace aplikačního rozhraní 92

C Postup přidání nového modulu 94

D Výstup převodníku použitý pro kalibraci senzoru síly působící na tabletu 95

E Obsah elektronické přílohy 97

Kapitola 1

Úvod

Při vývoji léčiv je v současné době využívána celá řada technického vybavení. Je tomu tak nejenom proto, že je tato oblast přísně hlídána legislativou, ale také proto, že technické vybavení usnadňuje a urychluje jejich vývoj. Kromě vlivů na lidské zdraví jsou zkoumány i další vlastnosti, jako je například rozpustnost účinných látek a časový průběh jejich uvolňování. Tyto procesy jsou odborně nazývány disoluce.

Součástí této práce je návrh řídicího systému pro laboratorní přístroj Golem, který je pro zkoumání disoluce léčiv navržen a používán vědeckými pracovníky a studenty na Veterinární a farmaceutické univerzitě v Brně (dále VFU). Přístroj Golem se od ostatních podobných zařízení liší tím, že je uspořádán tak, aby fyzicky simuloval rozdělení lidského trávicího ústrojí. Díky tomu je možné simulovat průchod léčiva žaludkem a částmi tenkého střeva včetně zajištění prostředí, které se v těchto částech vyskytuje.

V současné době je na VFU využívána druhá verze přístroje Golem. Tato práce se tedy zabývá realizací třetí verze a je zaměřena na vytvoření elektroniky a firmware pro její ovládání. Hlavní důraz je kladen na modularitu systému s možností případného rozšíření o další funkční celky. Systém je složen z individuálních modulů, které komunikují po sběrnici CAN s hlavní řídicí jednotkou, která také poskytuje aplikační rozhraní pro komunikaci s uživatelským rozhraním. To v podobě webové aplikace pro specifikaci experimentů, kontrolu jejich běhu a zobrazování výsledků ve své diplomové práci zpracovává Bc. Jan Truhlář.

Práce je členěna následovně. Kapitola 2 obsahuje úvod do výzkumu léčiv, představuje zařízení Golem, trávicí soustavu člověka a také různé lékové formy. V kapitole 3 je popsána stávající koncepce řízení zařízení Golem. Kapitola 4 popisuje návrh inovovaného řídicího systému a jeho rozdělení do modulů a také je zde představen komunikační protokol nad CAN sběrnicí, použitý pro komunikaci mezi moduly. V kapitole 5 jsou popsány softwarové a hardwarové bloky, které jsou dále použity v jednotlivých modulech, což popisuje kapitola 6. V kapitole 7 je představena aplikace pro Raspberry Pi, použitá pro řízení celého systému a poskytování aplikačního rozhraní. Kapitola 8 obsahuje postup kalibrace použitých čidel a popisuje ladění PID regulátorů. V závěrečné kapitole 9 je uveden postup testování jednotlivých částí i celého systému.

Kapitola 2

Disoluční zařízení Golem a jeho využití

Při výzkumu vlivu léčiv na organismy je nejpřesnější tento vliv zjišťovat *in-vivo*, tedy přímo na zkoumaných organismech. To ale v raných fázích vývoje není u léčiv určených pro člověka z etických ani legislativních důvodů možné. Kvůli tomu se přistupuje k výzkumu *in-vitro*, kdy dochází ke zjišťování vlastností zkoumané substance mimo živý organismus. Může to být pouze ve zkumavce, popřípadě za pomoci specializovaného přístroje. Ten se může různou mírou přibližovat k morfologii i fyziologii zkoumaného organismu, tedy jeho uspořádání a činnosti.

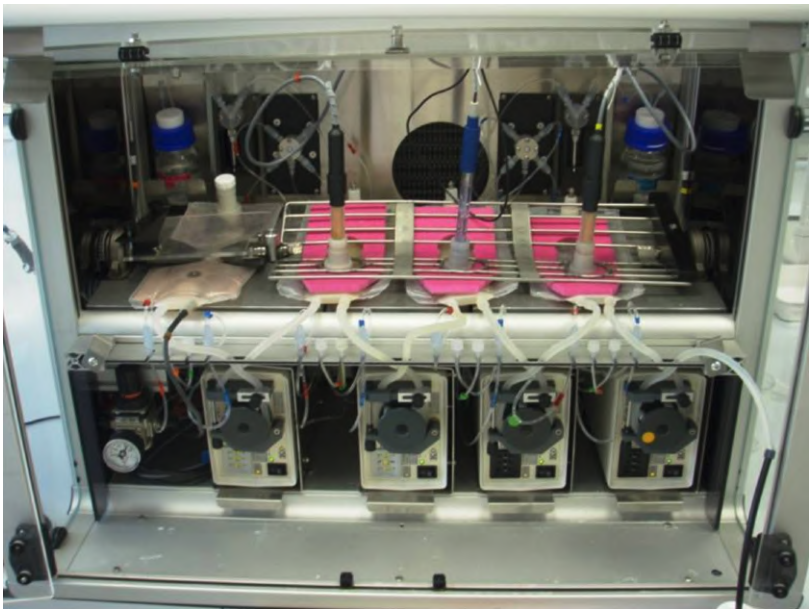
Jako příklad takového přístroje si můžeme uvést přístroj Sotax AT7, který slouží k provádění testů odpovídajících „2.9.3. Dissolution test for solid dosage forms“ z European Pharmacopoeia ([36]), což je tzv. lékopis, tedy „*základní farmaceutické dílo normativního charakteru, které přispívá k zajištění bezpečných, účinných a jakostních léčiv*“ ([43]). Pod kapitolou „2.9.3. Zkouška disoluce pevných lékových forem“ najdeme tento test i v Českém lékopisu [30]. Dle zmíněných lékopisů se jedná o test, při kterém je pevná léková forma (například tableta či kapsle) umístěna do skleněné nádoby s roztokem o stálé teplotě a je zajištěn pohyb roztoku pomocí míchací hlavy o definovaném tvaru a rozměrech. Na obrázku 2.1 je vidět přístroj Sotax AT7, ke kterému je připojena pumpa pro automatické odběry vzorků a jejich čerpání k analýze na spektrometru. Tím je možné zjišťovat průběh uvolňování látek ze zkoumané lékové formy v čase.

Přístroje tohoto typu sice můžou regulací podmínek simulovat fyziologii zkoumaného organismu, ale morfologicky se značně liší. Proto pro přesnější simulaci morfologie lidského trávicího ústrojí vyvinuli vědečtí pracovníci z Ústavu technologie léků z VFU zařízení s názvem Golem.

Obrázek 2.1: Disoluční zařízení Sotax AT7. Do skleněných kopulí s vhodným roztokem, umístěných v nádobě s vodou s konstantní teplotou, je vložena zkoumaná léková forma, která se za stálého míchání rozpouští. Připojená pumpa periodicky odebírá vzorky z nádob a předává je k analýze na spektrometru, čímž je možné zjistit časový průběh uvolňování jednotlivých látek.

2.1 Golem 1

Vývoj první verze zařízení Golem byl zahájen již v roce 2008 [50] a v roce 2011 na něj byl podán patent. Toto zařízení simuluje lidské trávicí ústrojí pomocí čtyř vaků, které odpovídají jeho čtyřem částem - přesněji žaludku a třem částem tenkého střeva (dvanáctník - *duodenum*, lačník - *jejunum*, kyčelník - *ileum*).



Obrázek 2.2: První verze disolučního přístroje Golem

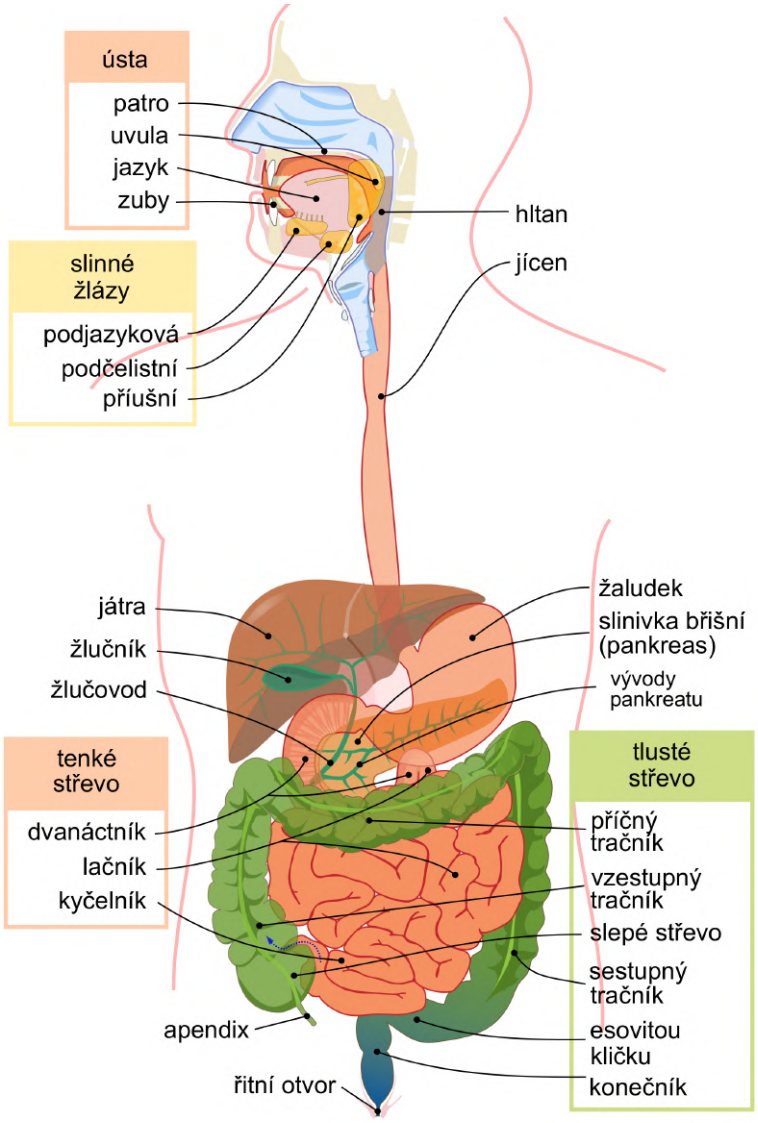
Na obrázku 2.2 je vidět první verze systému Golem. Protože bude celý systém podrobněji popsán dále, jsou představeny pouze základní části. Hlavní část tvoří čtyři vaky, zleva žaludek, dvanáctník, lačník a kyčelník, mezi kterými je možné přečerpávat objemy pomocí peristaltických pump. Zařízení je také vybaveno systémem pro udržování teploty, pH ve vacích apod. První vak, odpovídající žaludku, byl v první verzi míchán hydraulicky pomocí vzduchu, ostatní vaky mechanicky ocelovou kolébkou.

2.2 Trávicí soustava člověka

Pro lepší pochopení oblasti, jejímž výzkumem se zařízení zabývá, si uveďme základní části lidské trávicí soustavy a jejich funkce. Tato soustava slouží ke zpracování potravy, získání dostupných živin a následnému vyloučení nezpracovaných (či nezpracovatelných) zbytků. Její části jsou zakresleny na obrázku 2.3 a také dále popsány.

2.2.1 Dutina ústní

Ústní dutina je vstupem trávicí soustavy. Potrava je zde nejdříve mechanicky zpracována za pomoci zubů a jazyka a také zde začíná chemické zpracování díky enzymům ve slinách, které produkují slinné žlázy [40]. Jedná se převážně o enzym *ptyalin*, který je schopný metabolizovat škrob na kratší sacharidy [46]. Ty je možné dále zpracovávat pro účely získávání energie. Kromě toho ve slinách nalezneme ještě *mucin* (hlen), který slouží ke spojení potravy a zajištění jejího snadnějšího průchodu dále.



Obrázek 2.3: Trávicí soustava člověka (převzato z [47])

2.2.2 Hltan

Další částí trávicí soustavy je hltan. V této oblasti dochází ke křížení dýchací a trávicí soustavy. Aby nedocházelo ke vniknutí sousta do dýchací soustavy, je při polknutí vstup dýchací soustavy chráněn hrtanovou příklopkou, která mechanicky brání vstupu potravy do hrtanu. Dále odsud sousto pokračuje do jícnu.

2.2.3 Jícen

Funkce jícnu je převážně transportní. Potrava je peristaltickými stahy hladké svaloviny postupně přesouvána směrem do žaludku [46].

2.2.4 Žaludek

V žaludku je potrava dále zpracována mechanicky i chemicky. Jeho prostředí je výrazně kyselé – pH se pohybuje mezi hodnotami 2 a 4 [46]. Aby nedošlo k chemickému poškození stěn žaludku, jsou chráněny vrstvou zásaditého hlenu. Stěny žaludku tvoří hladká svalovina, jejíž stahy způsobují mechanické rozmělňování potravy. Kromě kyseliny chlorovodíkové (HCl) nalezneme v obsahu žaludku také chlorid sodný (NaCl), chlorid draselný (KCl) a další. V žaludku také probíhá enzymatické zpracování potravy (za přítomnosti lipáz, *pepsinu* a dalších enzymů) [40].

2.2.5 Tenké střevo

Tenké střevo je hlavním místem, kde dochází ke zpracování živin z natrávené potravy. Na povrchu jeho stěn se nacházejí *klky*, což jsou výběžky vrchní vrstvy střevní sliznice. Díky nim je povrch střeva zvětšen a dochází tak snadněji k získávání živin z potravy. Zde jsou do potravy přidávány výměšky dalších orgánů a žláz, například žluč z jater a šťávy slinivky břišní. I tenké střevo je v neustálém pohybu, čímž dochází k posunu trávené potravy [40]. Tenké střevo se dále dělí na dvanáctník, lačník a kyčelník.

Dvanáctník (Duodenum)

Dvanáctník je první částí tenkého střeva, do které se dostává trávenina postupující ze žaludku. Jeho stěny jsou ze tří částí tenkého střeva nejvíce pokryty klky. Ústí sem také vývody dalších žláz, jako například slinivky břišní. Její šťávy obsahují hydrogenuhličitan, který snižuje kyselost tráveniny [46]. Kromě toho do dvanáctníku ústí střevní žlázky, produkující slabě zásaditou střevní šťávu, která pomáhá k dalšímu zvyšování pH [40] a to až přibližně k hodnotě 6 [41].

Lačník (Jejunum)

Lačník tvoří prostřední část tenkého střeva. Kyselost prostředí postupně klesá a dochází zde k dalšímu získávání živin z potravy. Počet klků na stěnách se postupně snižuje a lačník bez jasné hranice přechází v kyčelník.

Kyčelník (Ileum)

Kyčelník je asi o třetinu kratší, než lačník [40]. Hodnota pH zde opět roste k neutrálním až mírně zásaditým hodnotám [41]. Oproti předchozí části je tato trubice užší, kratší, méně

prokrvená a na jejích stěnách nalezneme nejméně klků z celého tenkého střeva. Na svém konci ústí do tlustého střeva.

2.2.6 Tlusté střevo

Tlusté střevo již neprodukuje žádné trávicí enzymy a jeho hlavním účelem je vstřebávání solí a vody. Úbytek vody v trávenině způsobuje zahušťování obsahu střeva a jeho hromadění. Nalezneme zde také velkou řadu symbiotických bakterií, které jsou člověku prospěšné, například tvorbou vitaminů K a B_{12} (viz [38], [42]). Zahuštěné zbytky potravy jsou zde postupně formovány ve stolici, která je vylučována konečníkem [46].

2.3 Lékové formy

Léková forma určuje podobu podávaného léčiva. Jedná se tedy o popis jeho fyzických i chemických vlastností. Lékové formy lze dělit podle různých kritérií, například dle konzistence, způsobu podání, způsobu uvolňování apod. [44].

2.3.1 Dělení dle uvolňování léčivé látky

Český lékopis ([30]) rozděluje lékové formy dle způsobu uvolňování následovně:

- Lékové formy s neřízeným uvolňováním - Nemají záměrně upravené uvolňování léčivé látky.
- Lékové formy s řízeným uvolňováním - Uvolňování léčivé látky je cíleně upraveno. Toho může být dosaženo například speciálním výrobním postupem. Tyto formy se dále dělí na:
  - Lékové formy se zpožděným uvolňováním - Léčivá látka se uvolňuje později, než u forem s neřízeným uvolňováním.
  - Lékové formy s prodlouženým uvolňováním - Léčivá látka se uvolňuje pomaleji, než u forem s neřízeným uvolňováním.
  - Lékové formy s pulzním uvolňováním - Léčivá látka je uvolňována po částech.

Generace lékových forem

Dále lze formy rozdělit do tří generací podle způsobů uvolňování a cílení účinné látky [44].

1. generace - v této skupině je většina současných běžně dostupných léků. Účinná látka je uvolněna velmi rychle a její koncentrace je ovlivňována pouze procesy v lidském organismu.
2. generace s řízeným prodlouženým uvolňováním - je speciální forma léčiv, které jsou schopny dodávat stanovené dávky léků v daných časech. Například může jít o tablety, které uvolňují léčivou látku ve správné části trávicího ústrojí, nebo o různé náplasti na kůži.
3. generace s cílenou distribucí - úkolem léčiv z této generace je dopravit léčivou látku nejkratší cestou přesně na určené místo bez ovlivňování dalších tkání.

Dělení dle konzistence

Podle konzistence je možné lékové formy dělit na:

- tekuté - kapky, sirupy, kloktadla, spreje, injekce
- polotuhé - pasty, gely, masti
- tuhé (tvarově specifické, tvarově nespecifické) - granuláty, zásypy, tablety, tobolky, čípky, globule
- plynné - aerosoly, inhalátory

Dělení dle způsobu aplikace

Podle způsobu aplikace je lékové formy možné dělit na:

- Gastrointestinální - léky podávané prostřednictvím zažívacího traktu (kapky, tablety, prášky, ...)
- Parenterální - většinou injekční podání do žily, nebo svalu (infuze, injekce, implantáty, ...)
- Topické léky - léky s místním účinkem. Je možné je dělit na léky s podáním inhalací, aplikací na kůži, oční léky, rektální léky a další.

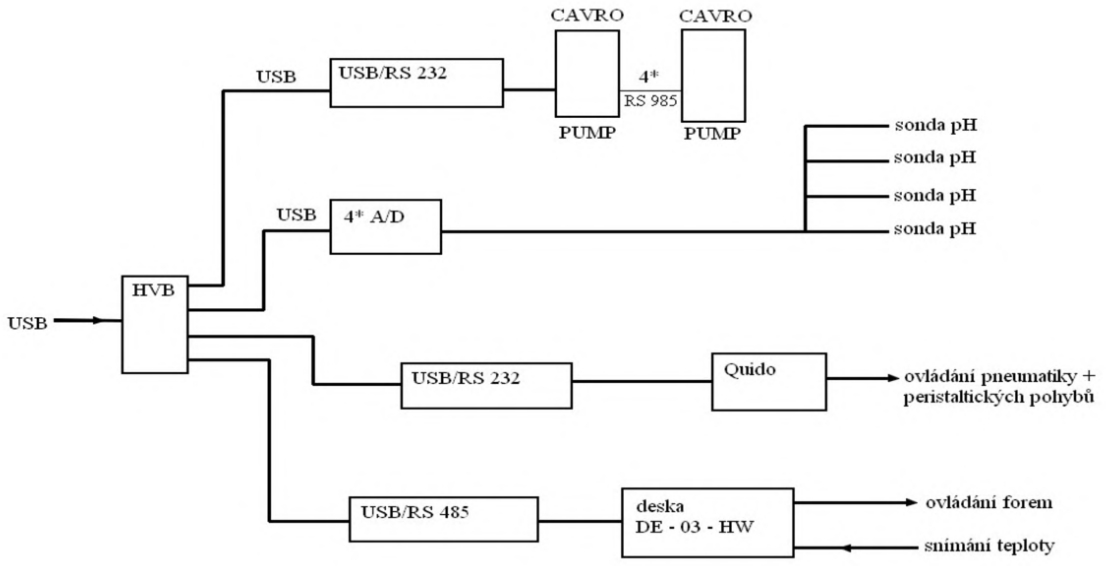
Kapitola 3

Stávající koncepce řídicího systému

V kapitole 2.1 bylo zařízení Golem představeno z pohledu výzkumu léčiv. V této kapitole bude toto zařízení popsáno z pohledu jeho původní technické realizace. Na VFU je v současnosti v provozu druhá verze zařízení. Pro úplnost bude ale na začátek v krátkosti představena i jeho první verze z roku 2008.

3.1 Varianty zařízení Golem

Na obrázku 3.1 je vidět schéma první verze zařízení Golem. Z něj (a také z fotky zařízení na obrázku 2.2) je patrné, že zařízení bylo řízeno přes rozhraní USB. Prostřednictvím tohoto rozhraní byly připojeny převodníky na sběrnice RS485 a RS232, které řídily pístové pumpy a ventily Cavro, určené k dávkování enzymu a regulaci pH prostředí ve vacích. Přes I/O modul Quido byly řízeny peristaltické pumpy určené k přečerpávání obsahu vaků. Přes modul s mikrokontrolérem Atmega (DE-03-HW) byla řízena teplota prostředí a míchání vaků. Měření pH ve vacích probíhalo prostřednictvím čtyř A/D převodníků.



Obrázek 3.1: Schéma řízení první verze systému Golem

USB HUB, spojující komponenty systému, byl připojen k PC, na kterém běžela aplikace ve Visual Basic, která sloužila k řízení běhu experimentů a zpracování výsledků.

Golem ve své první verzi postupem času přestal týmu výzkumníků na VFU stačit a vznikl tedy požadavek na úpravu systému na novou verzi Golem 2. Přestavba Golem 1 na Golem 2 byla uskutečněna v rámci veřejné zakázky "Disoluční zařízení GOLEM 2"[28], vypsané Veterinární a farmaceutickou univerzitou Brno.

Od roku 2013 [28] je v provozu druhá verze systému Golem, která je aktivně využívána při výzkumu. Z poslední doby je možné zmínit například práci [48], která mj. porovnává výsledky měření více operátorů systémů a snaží se tak popsat vliv člověka na výsledky experimentů. Současná verze zařízení je vidět na obrázku 3.2.

Obrázek 3.2: Disoluční zařízení Golem 2 (čtvrtý vak není v probíhajícím experimentu použit)

Ve druhé verzi byla hlavní funkční část zachována, ale došlo ke změně architektury řízení systému (viz obrázek 3.3). USB HUB nahradil modul Arduino Uno, který slouží jako adaptér pro spojení disolučního zařízení a počítače s řídicí aplikací a také řídí topení, peristaltické pumpy a provádí měření výstupů z analogových čidel (pH sondy a teploměry).

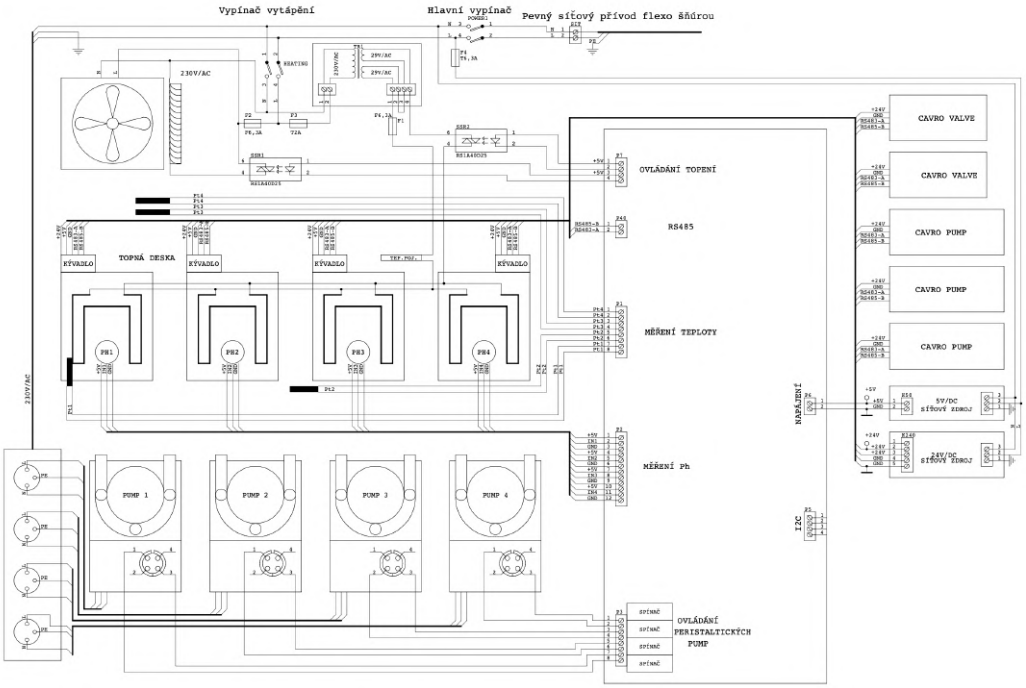
S dalšími prvky systému, kterými jsou pístové pumpy, ventily a obvody pro řízení míchání vaků, modul Arduino komunikuje prostřednictvím sběrnice RS485, která tak tvoří hlavní komunikační kanál celého systému. A protože Arduino Uno obsahuje převodník z USB na sériovou linku, je možné zařízení připojená na RS485 ovládat přímo z PC, ke kterému je Arduino připojeno.

Komunikační protokol nad RS485, kterým jsou jednotlivé části systému řízeny, vychází z protokolu pístových pump a ventilů Cavro, které jsou použity pro vstřikování enzymu, kyseliny a zásady do vaků. Tento protokol je dále popsán v kapitole 3.3.

3.2 Subsystémy zařízení Golem 2

V systému je možné identifikovat několik základních bloků, které jsou vidět ve schématu na obrázku 3.3. Jedná se o subsystém pro regulaci teploty, reakční vaky a jejich míchání,

měření a regulace pH, vstřikování enzymu, přečerpávání obsahu vaků, napájení a řídicí modul Arduino. Tyto subsystémy jsou dále detailněji představeny.



Obrázek 3.3: Blokové schéma Golem 2

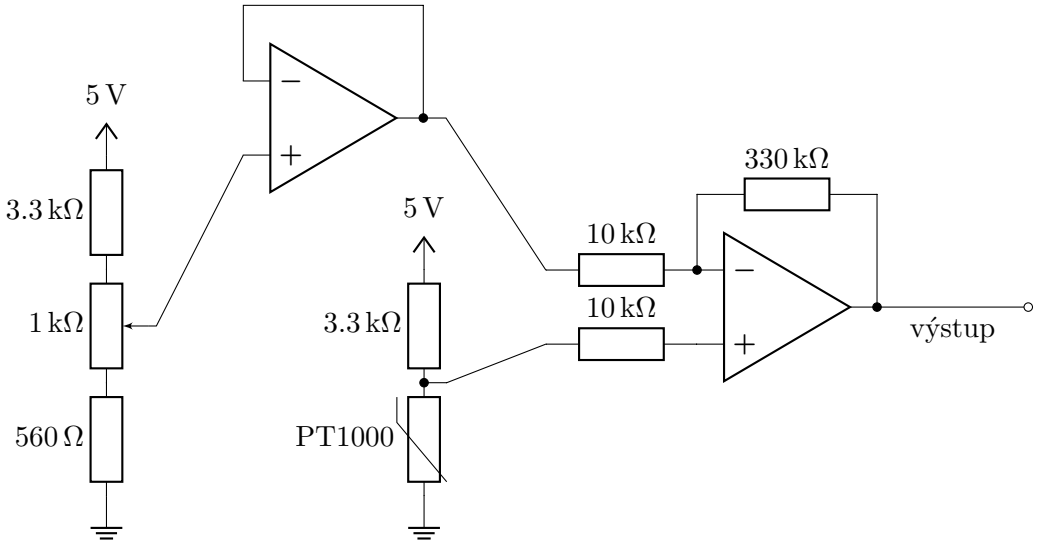
3.2.1 Systém regulace teploty

Pro měření teploty jsou využita čtyři analogová teplotní čidla PT1000 [4]. Dvě z těchto čidel jsou umístěna na stranách vnitřního prostoru zařízení a slouží ke snímání teploty vzduchu. Jedno čidlo je umístěno na platformě, na které jsou položeny vaky. Poslední z čidel je zasunuto přímo do prvního vaku (žaludek). Čidla jsou chráněna nerezovými, chemicky odolnými pouzdry.

Výhodou teplotních čidel PT1000 je lineární závislost odporu na teplotě v širokém teplotním rozsahu [4]. To byl zřejmě také důvod jejich výběru. Teplotní čidlo je společně s $3.3\,k\Omega$ resistorem zapojeno jako dělič napětí. Výstup tohoto děliče je připojen na kladnou větev operačního zesilovače, který je zapojený jako diferenciální zesilovač. Na zápornou větev je přiveden výstup obvodu sloužící jako zdroj referenčního napětí, proti kterému se porovnává výstup z teplotního čidla. Výstup diferenciálního zesilovače je přiveden na vstup osmnáctibitového analogově-digitálního převodníku MCP3424, který je přes I2C sběrnici připojen k modulu Arduino Uno [45]. Schéma analogové části je vidět na obrázku 3.4.

Pro účely regulace teploty je Golem vybaven ventilátorem s ohřevem a také čtveřicí topných elementů, které jsou umístěny pod platformou s vaky. Ventilátor je napájen síťovým napětím, napětí pro topné elementy je nejdříve transformátorem sníženo na 58 V. Jak ventilátor, tak topné elementy jsou ovládány dvojicí polovodičových relé RS1A40D25, jejichž spínání zajišťuje modul Arduino Uno.

Řízení topení probíhá v aplikaci na PC, se kterou si Arduino vyměňuje zprávy po sériové lince (viz kapitola 3.3). Ze zdrojového kódu aplikace plyne, že teplota vzduchu je pouze informační a není použita pro řízení (při vizualizaci je použit průměr z teplot). Hlavní

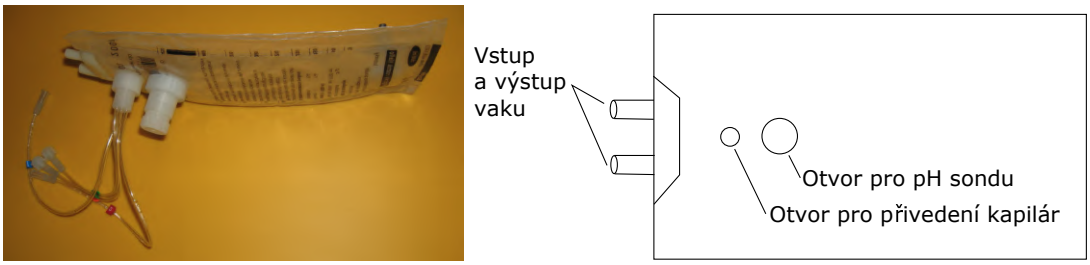


Obrázek 3.4: Obvod pro úpravu signálu z teploměru Pt1000

jsou teplota v prvním vaku a teplota platformy. Ty jsou použity jako vstup pro jednoduché proporcionální regulátory topení. Z bezpečnostních důvodů je systém vybaven i tepelnou pojistkou na platformě s vaky, aby se zabránilo poškození, či vzplanutí topných elementů pod vaky.

3.2.2 Reakční vaky

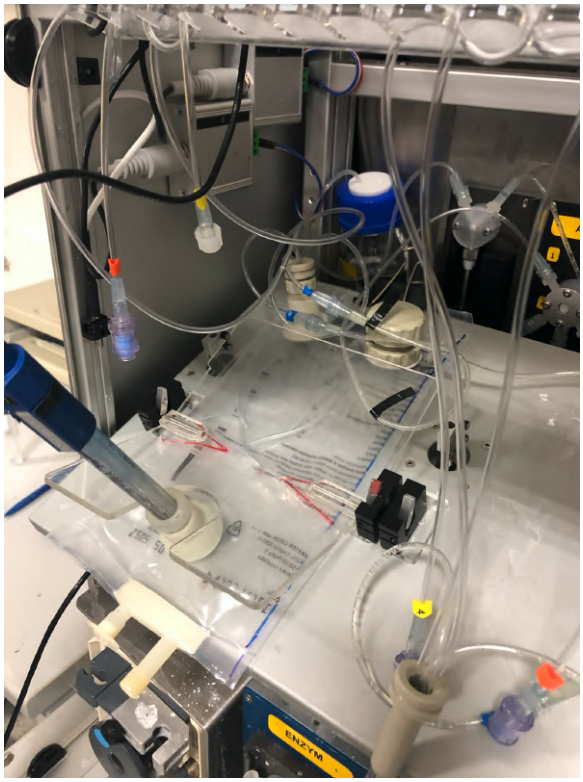
V systému mohou být umístěny až čtyři reakční vaky, které simulují jednotlivé částí lidského trávicího traktu (viz kapitola 2.2). Každý z vaků má několik otvorů - dva pro vstup a výstup peristaltické pumpy, jeden velký vstup pro vložení pH sondy a také čtveřici kapilár pro zavedení kyseliny, zásady a enzymu (jedna obvykle zůstává volná).



Obrázek 3.5: Reakční vak a jeho schéma

3.2.3 Míchání reakčních vaků

Na každém vaku je umístěna kolébka z plexiskla, která slouží k míchání obsahu uvnitř vaků (simuluje fakt, že i v lidském trávicím traktu je obsah neustále v pohybu). Platforma pod vaky je mírně sklopená, takže dochází k hromadění kapaliny ve spodní části vaků, odkud může být peristaltickými pumpami dále přečerpávána. Umístění vaku je vidět na obrázku 3.6.



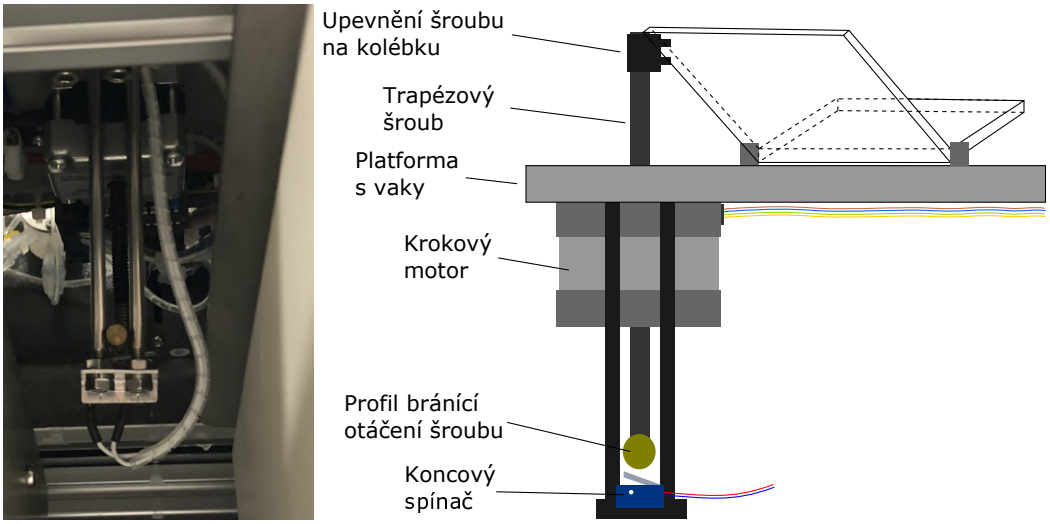
Obrázek 3.6: Umístění reakčního vaku

Kolébka je uváděna do pohybu za pomoci krokového motoru SX16-0402LA-120 s dutou hřídelí. Krokové motory jsou umístěny pod platformou s vaky a prochází jimi trapézový šroub, který dále prochází platformou a je připevněn k míchací kolébce. Motor pohybuje trapézovým šroubem periodicky v rozsahu cca 8 cm, čímž dochází k pohybům kolébky a míchání vaků. Na jednom konci rozsahu pohybu je poloha snímána koncovým spínačem. Na druhém konci je limit pohybu určen na základě čítání provedených otáček motoru. Umístění motoru, trapézový šroub a koncový spínač jsou vidět na obrázku 3.7.

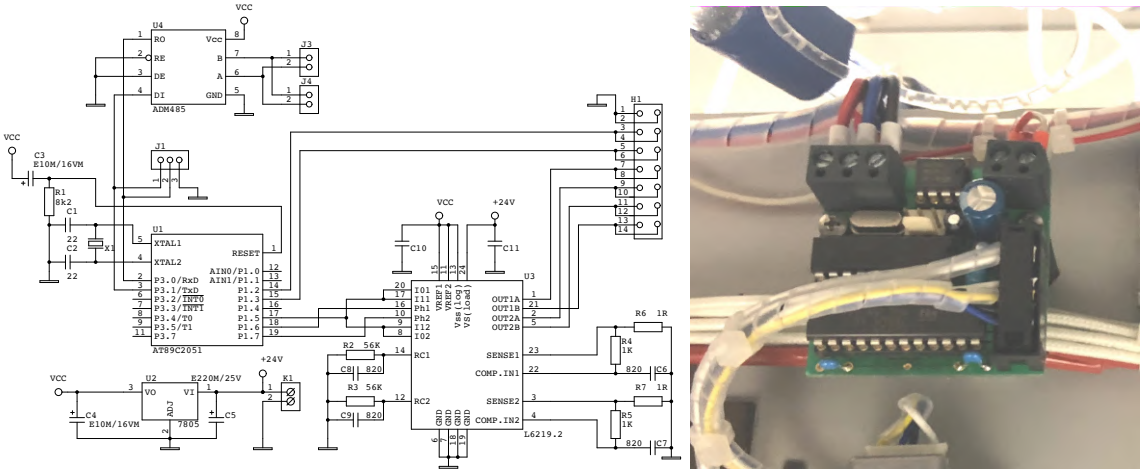
Každý z motorů má vlastní řídicí jednotku. Ta sestává z mikrokontroléru AT89C2051, budiče krokových motorů L6219.2 a převodníku z UART TTL na RS485. Také zpracovává signály z koncového spínače. Ke komunikaci s PC je opět použit společný protokol - viz 3.3. Prostřednictvím tohoto protokolu je možné nastavit rychlost kývání kolébky, o zbytek se stará již mikrokontrolér samotný.

3.2.4 Měření a regulace pH

Při zkoumání procesů v trávicí soustavě musíme reflektovat i pH jejích jednotlivých částí. Jak bylo popsáno v kapitole 2.2, nejkyselejší prostředí je v žaludku a v dalších částech již kyselost klesá (tedy pH roste). Hodnota pH je při experimentech podstatná, protože roztok se změnou pH má i jiné chemické vlastnosti a ovlivňuje tak i například rychlost rozpouštění léčiva. Různá pH v různých částech soustavy se snaží udržovat i zařízení Golem. Z toho důvodu je vybaveno senzory a aktuátory pro měření a úpravu pH.

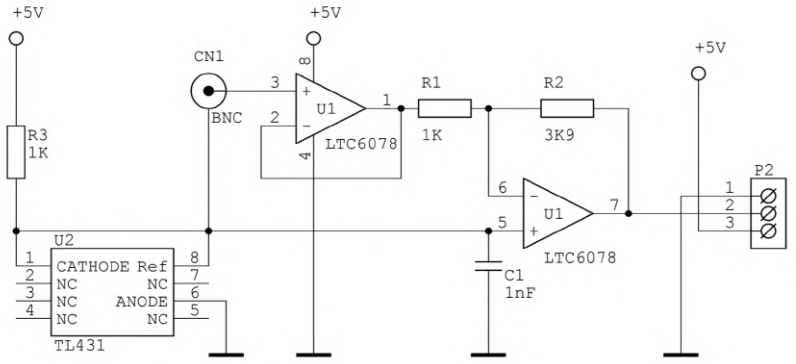


Obrázek 3.7: Umístění motoru pro pohon kolébky



Obrázek 3.8: Schéma řídicího obvodu pohonu míchací kolébky a jeho provedení.

V každém vaku je zasunuta elektroda schopná měřit pH. Konkrétně se jedná o sondy FlaTrode od společnosti Hamilton. Ty se vyznačují tím, že mají plochý konec, takže jsou schopné měřit pH i v případě, že není možné je celé ponořit do měřeného roztoku [6]. Měření pH využívá dle [23] jevu, který v roce 1906 objevil Max Cremer. Ten popsal, že mezi dvěma roztoky s rozdílným pH, oddělenými tenkou skleněnou přepážkou, existuje elektrický potenciál. Elektrická sonda tedy obsahuje roztok se známým pH, na který přivedeme referenční napětí, oproti kterému měříme potenciál ve zkoumaném roztok. Dle dokumentace sondy ([6]) je rozdíl potenciálu přímo úměrný měřenému pH a jeho velikost je při 25 °C asi 58 mV/pH. Při pH 7 je rozdíl potenciálů cca 0 V (referenční roztok uvnitř sondy má tedy pH 7).



Obrázek 3.9: Obvod pro zpracování signálu z pH sondy

Referenční elektroda pH sondy je připojena k výstupu obvodu TL431 (viz obrázek 3.9), který je dle [33] zapojen tak, aby poskytoval stabilní úroveň napětí 2.5 V. Hodnota napětí na druhé z elektrod je poté dále upravena operačními zesilovači. První operační zesilovač je zapojen jako sledovač napětí a slouží k impedančnímu oddělení měřící elektrody od zbytku obvodu. Druhý z operačních zesilovačů zesiluje rozdíl výstupu elektrody oproti hodnotě referenčního napětí (2.5 V) a zároveň tento rozdíl invertuje.

Arduino poté periodicky snímá hodnotu napětí na analogových vstupech a po dotázání ze strany PC ji odesílá do řídicí aplikace, kde je dále zpracována a přepočtena na skutečnou hodnotu pH. Při 58 mV/pH je tedy výstup sondy a napětí po úpravě operačními zesilovači zachycen v tabulce 3.1.

Tabulka 3.1: Závislost výstupního napětí analogového obvodu pro zpracování signálu z pH sond na pH měřeného roztoku

pH	**sonda [V]**	**výstup obvodu [V]**	**pH**	**sonda [V]**	**výstup obvodu [V]**
0	2.094	4.083	8	2.558	2.274
1	2.152	3.857	9	2.616	2.048
2	2.21	3.631	10	2.674	1.821
3	2.268	3.405	11	2.732	1.595
4	2.326	3.179	12	2.79	1.369
5	2.384	2.952	13	2.848	1.143
6	2.442	2.726	14	2.906	0.917
7	2.5	2.500			

Pokud se požadovaná hodnota pH ve vaku liší od naměřené hodnoty, zahájí řídicí aplikace sekvenci pro úpravu pH. K tomu jsou využity pístová pumpa (Tecan Cavro XE1000) [24] a ventil (Tecan Cavro Smart Valve), [21] sloužící ke směrování kapaliny do správného vaku. V systému nalezneme jednu dvojici (pumpa + ventil) pro dávkování kyseliny a jednu dvojici pro dávkování zásady. S těmito zařízeními komunikuje řídicí aplikace po sběrnici RS485.

Při požadavku na úpravu pH je nejdříve do patřičné pístové pumpy natažena kyselina či zásada (pokud v ní již není), je nastaven ventil pro směrování do správného vaku, a dále je vstříknut požadovaný objem kyseliny či zásady. Komunikační protokol je popsán v kapitole 3.3.

3.2.5 Vstřikování enzymu

Systém je také vybaven jednou pístovou pumpou Cavro XE1000 pro vstřikování enzymu do vybraného vaku. Hadička z výstupu pumpy musí být manuálně přivedena k jednomu z vaků před začátkem experimentu. Většinou se jedná o druhý vak, který odpovídá dvanáctníku.

3.2.6 Peristaltické pumpy

Mezi vaky je možné přečerpávat pomocí peristaltických čerpadel PCD 81E. Jejich řízení je pouze dvoustavové a je tedy možné ovlivnit pouze to, jestli čerpají nebo nečerpají. Přímo na čerpadle je ale možné manuálně nastavit rychlost čerpání (viz obrázek 3.10). Ovládání čerpadla probíhá na základě spínání dvojice vodičů, které jsou vývodem konektoru na jeho zadní straně. Ze schématu rozšiřujících obvodů modulu Arduino Uno (viz 3.12) plyne, že jsou ke spínání použity optomosfety bez dalšího určení.

3.2.7 Napájení

Jednotlivé části systému jsou napájeny různým napětím. Peristaltická čerpadla a ventilátor s ohřevem jsou napájeny přímo síťovým napětím 230 V. Toto střídavé napětí je dále transformátorem sníženo na 58 V, kterými jsou napájeny topné elementy pod platformou s vaky. Systém dále obsahuje zdroje stejnostměrného napětí 24 V a 5 V (jejich umístění v zadní části přístroje je vidět na obrázku 3.11). Napětím 24 V jsou napájeny pístové pumpy a ventily a dále také krokové motory pro pohon míchacích kolébek. Řídicí elektronika je napájena napětím 5 V. Jedná se o mikrokontrolery v ovládání míchání, elektroniku pH sond a také modul Arduino Uno.

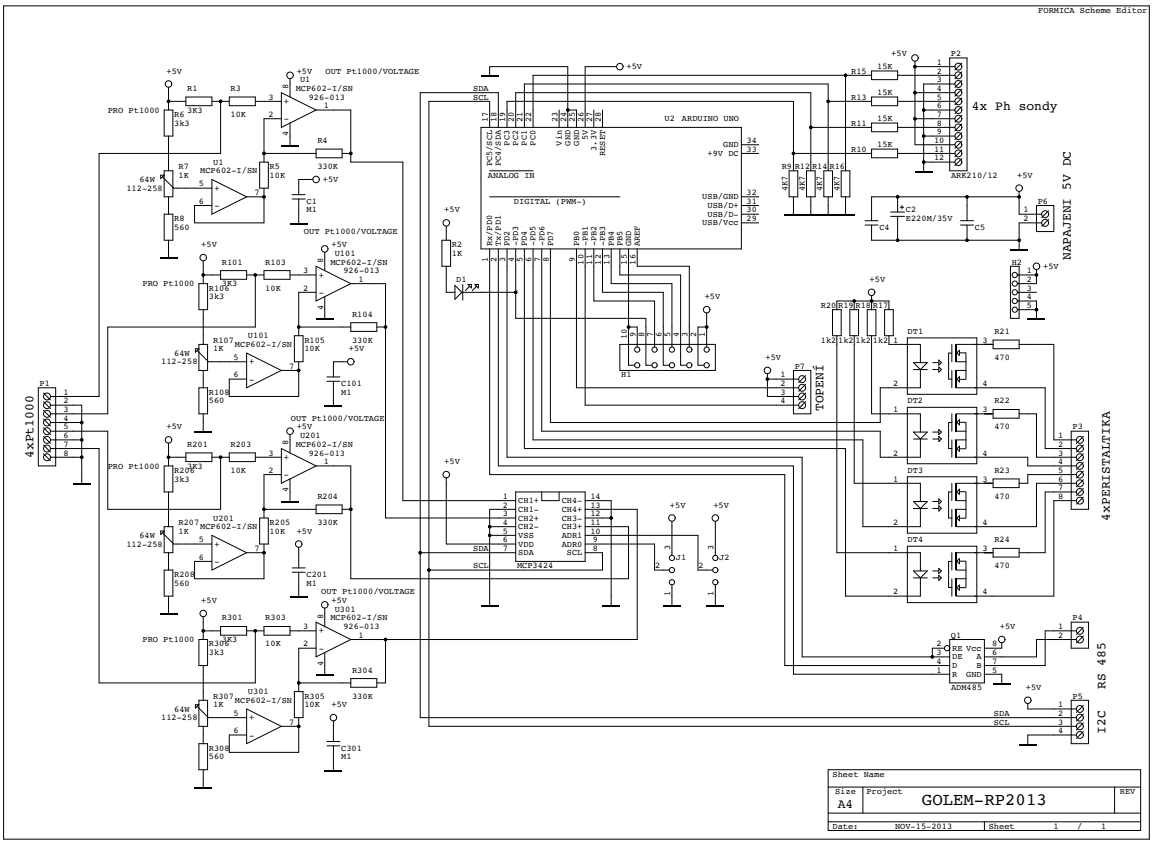
3.2.8 Rozšiřující obvody pro Arduino UNO

Pro účely připojení všech částí systému k modulu byla vyvinuta rozšiřující deska s množstvím konektorů pro připojení výstupu elektroniky pH elektrod, ovládání peristaltických pump, teploměrů, sběrnice RS485, ovládání topení apod. Dále zde také najdeme obvody pro úpravu vstupních a výstupních signálů, například analogově-digitální převodník, zpracovávající signál z teplotních čidel, optospínače pro řízení peristaltických pump, převodník z TTL UART na RS485 a další. Schéma rozšiřujícího obvodu je vidět na obrázku 3.12.

Modul Arduino s připojeným rozšiřujícím obvodem je, stejně jako zdroje napájení a další komponenty, umístěn na zadní straně zařízení Golem, kde je mimo dosah běžného uživatele – viz obrázek 3.13.

Obrázek 3.10: Peristaltická pumpa PCD 81E. Červeným přepínačem je možné nastavit výkon pumpy.

Obrázek 3.11: Zdroje stejnosměrného proudu o napětí 5 V a 24 V

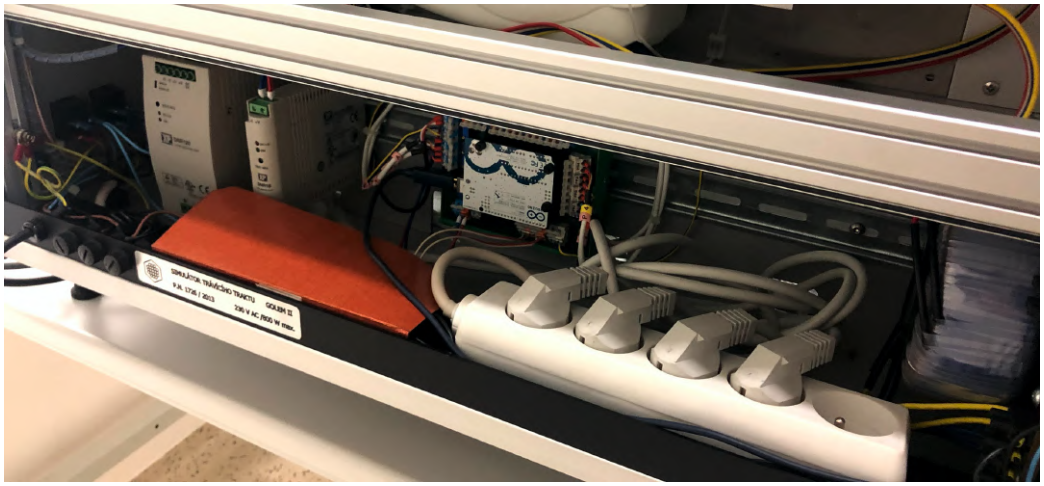


Obrázek 3.12: Schéma rozšiřujících obvodů pro modul Arduino Uno. V levé části jsou vidět analogové obvody pro úpravu signálu z teplotních čidel, jejichž výstup je zpracován osmnáctibitovým analogově-digitálním převodníkem MCP3424. Na pravé straně nahoře jsou značeny konektory pro připojení pH sond, uprostřed optomosfety pro spínání peristaltických pump a dole RS485 převodník sběrnice ADM485 a konektory I2C sběrnice. Uprostřed schématu je vidět připojení komponent k modulu Arduino Uno.

Převodník z USB na UART, který je na Arduino modulu dostupný, je také využit jako hlavní komunikační brána mezi počítačem s řídicí aplikací a celým zařízením. Jak je vidět ve schématu na obrázku 3.12, je převodník sběrnice RS485 připojen přímo na piny Tx a Rx modulu Arduino, které jsou na tomto modulu fyzicky připojeny k vývodům převodníku z USB na UART TTL sériovou linku.

3.3 Komunikační protokol

Jak již bylo několikrát zmíněno, pístové pumpy a ventily Cavro je možné ovládat mj. prostřednictvím sběrnice RS485 (další možností jsou dle [24] sběrnice RS232 a CAN). Na této sběrnici zařízení komunikují s rychlostí 9600 baud, používají 8 datových bitů a jeden stop bit bez parity. Protokol, jehož zprávami je možné zařízení řídit, výrobce označuje jako *Data Terminal protocol* (dále jako DT protokol). Zpráva tohoto protokolu začíná vždy znakem lomítka ('/'), následuje adresa zařízení, dále n datových bytů, které jsou zakončeny znakem CR. Pumpa/ventil poté odpovídají podobnou sekvencí, zakončenou znaky CR a LF. Datové



Obrázek 3.13: Umístění desky Arduino do zadní části zařízení Golem 2

byty obsahují pro ventily například kontrolní znaky jejich natočení a pro pístové pumpy objem, který má být přečerpán. V tabulce 3.2 je vidět struktura zpráv protokolu [24].

Tabulka 3.2: Struktura zpráv protokolu pro řízení pístových pump a ventilů Cavro

Offset	**Význam**
Řídicí zpráva	
1	Úvodní znak (ASCII '/' nebo $2F_{HEX}$)
2	Adresa
3	Blok dat (délka n)
3+n	Znak CR ($0D_{HEX}$)
Odpověď	
1	Úvodní znak
2	Adresa master zařízení (ASCII '0' nebo 30_{HEX})
3	Status byte
4	Blok dat (délka n)
4+n	Znak ETX (03_{HEX})
5+n	Znak CR ($0D_{HEX}$)
6+n	Znak LF ($0A_{HEX}$)

Adresu pumpy či ventilu je možné volit pomocí přepínačů na jejich zadní straně a byly zvoleny v rozsahu '1' až '5'. Použitý protokol připouští až 15 zařízení na jedné RS485 lince s adresami 31_{HEX} až $3F_{HEX}$, tedy znaky '1' až '?'. Adresa '0' je vyhrazena master zařízení a je použita při odesílání odpovědí na kontrolní povely. Zbývajícím zařízením na sběrnici RS485 byla přiřazena adresa tak, aby nebyla v konfliktu s již přiřazenými adresami a zároveň byla v požadovaném rozsahu. Adresy všech zařízení na sběrnici jsou zaneseny v tabulce 3.3.

3.3.1 Řízení pístových pump Cavro XE1000

Blok dat DT protokolu může mít až 32 řídicích bytů (tato délka je limitována vstupním bufferem Cavro zařízení). Jedná se o byty kódující ASCII znaky s definovaným významem.

Tabulka 3.3: Adresy zařízení na sběrnici RS485

Adresa	**Zařízení**
1	Pístová pumpa kyseliny
2	Ventil kyseliny
3	Pístová pumpa zásady
4	Ventil zásady
5	Pístová pumpa enzymu
6	Modul míchání vaku 1
7	Modul míchání vaku 2
8	Modul míchání vaku 3
9	Modul míchání vaku 4
:	Modul Arduino

Dále jsou představeny sekvence příkazů, které jsou použity při ovládání pump. Popis všech příkazů je dostupný v dokumentaci pump [24]. Uvedena je vždy jen datová část.

Tabulka 3.4: Příkazy pro řízení pístových pump Cavro

Sekvence	**Popis**
Zn Yn	Inicializace. Pumpa se sestává z pístu a malého ventilu. Ten je při nasávání natočen jedním směrem, při vypouštění pístu druhým směrem. První znak sekvence určuje, jaký port ventilu je vstupní, a jaký výstupní. Hodnota n udává rychlost pohybu pístu (1-20).
Q	Získání status byte
F	Zjištění stavu vstupního bufferu příkazů
I	Přetočení ventilu na vstupní port
O	Přetočení ventilu na výstupní port
Mn	Čekání n milisekund
An	Přesunutí hlavy pístu do polohy n (0-1000).
R	Spuštění dříve zaslané sekvence příkazů.

Většina příkazů se neprovádí ihned, ale je ukládána ve vstupním bufferu. K jejich provedení dojde až po přijetí znaku 'R'. Rozsah pohybu hlavy pístu je 0 až 1000 a tyto limity značí úplné nabrání, či vypuštění obsahu pístu. Písty na pumpě jsou vyměnitelné, díky čemuž je možné měnit jejich objem. V zařízení Golem jsou na pumpách použity písty o objemu 1 ml. To dává při 1000 dílcích výsledné rozlišení 1 µl. Pokud tedy chceme dávkovat 100 µl, může být sekvence příkazů následující: "IA1000OA900R". Nejdříve je tedy ventil nastaven do vstupní polohy, hlava pístu je přesunuta do pozice 1000 (tím je nasáto 1000 jednotek), dále je ventil otočen do výstupní polohy a hlava pístu je posunuta do pozice 900, čímž dojde k vytlačení 100 jednotek – v tomto případě 100 µl.

Součástí odpovědi na příkaz je i byte obsahující informace o stavu zařízení (*status byte*). Ten začíná vždy sekvencí bitů "01", po kterých následuje bit s informací, jestli je zařízení schopné přijímat další příkazy (1 = ano, 0 = ne). Zbývajících pět bitů obsahuje informace o případné chybě [24]. Chybové kódy jsou popsané v tabulce 3.5.

Tabulka 3.5: Chybové kódy pístových pump Cavro

Bity								Popis chyby
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0	1	X	0	0	0	0	0	Bez chyby
0	1	X	0	0	0	0	1	Chyba inicializace
0	1	X	0	0	0	1	0	Neplatný příkaz
0	1	X	0	0	0	1	1	Neplatný operand
0	1	X	0	0	1	0	0	Neplatná sekvence příkazů
0	1	X	0	0	1	1	1	Zařízení neinicializováno
0	1	X	0	1	0	0	1	Hlava pístu přetížena
0	1	X	0	1	0	1	0	Ventil přetížen
0	1	X	0	1	0	1	1	Pohyb hlavy není povolen
0	1	X	0	1	1	1	1	Přetečení bufferu příkazů

3.3.2 Řízení ventilů Cavro Smart Valve

Struktura zpráv i chybové kódy jsou u Cavro ventilu stejné, jako u pístových pump. Stejná je i velikost bufferu příkazů (32 bytů) [21]. Jistá odlišnost je v dostupných příkazech – ty používané jsou v tabulce 3.6.

Tabulka 3.6: Příkazy pro řízení ventilů Cavro

Sekvence	**Popis**
ZmYn	Inicializace. Na osu ventilu je možné nasadit různé profily, například profil ve tvaru písmene Y. Zvolený profil je poté nastaven parametrem m. V případě systému Golem je zvolen profil s jedním vstupem a čtyřmi výstupy ($m = 4$). Parametr n nastavuje směr pohybu ventilu (1 po směru hodinových ručiček, 2 proti směru).
An	Nastavení výstupu na port n
R	Spuštění dříve zaslané sekvence příkazů.

3.3.3 Řízení obvodů míchání

Mikrokontrolér AT89C2051, který je součástí obvodů míchání, přijímá příkazy ve stejném formátu, jako pumpy a ventily. Adresy obvodů míchání jsou zachyceny v tabulce 3.3. Jediným příkazem, který obvod podporuje je nastavení rychlosti míchání (viz tabulka 3.7).

Tabulka 3.7: Řízení obvodu míchání

Sekvence	**Popis**
Sn	Nastavení rychlosti míchání ($n \in \langle 0, 5 \rangle$).

3.3.4 Řízení modulu Arduino

Posledním prvkem systému, který přijímá příkazy protokolu představeném v této kapitole, je modul Arduino Uno. Ten udržuje informace o pH a teplotě a také ovládá topení (více viz 3.2.8). Dostupné příkazy jsou zaneseny v tabulce 3.8.

Tabulka 3.8: Řízení modulu Arduino Uno

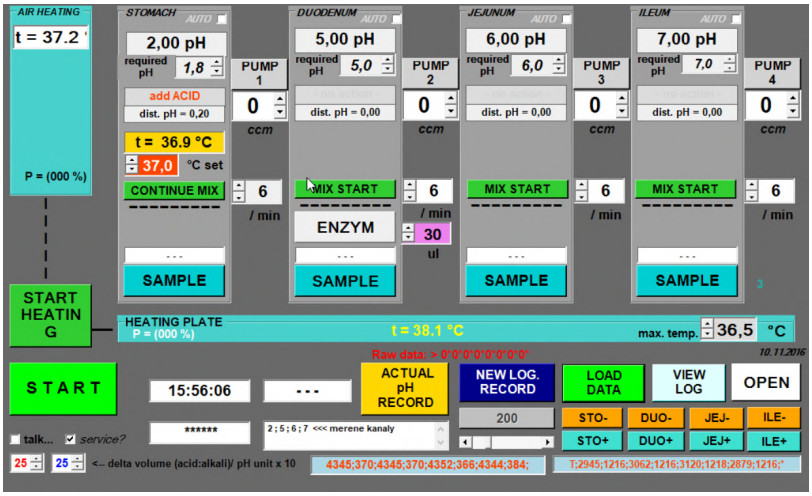
Sekvence	Popis
D	Řídicí aplikace žádá o data. Ty jsou odeslány v textovém formátu odděleném čárkou. Odeslány jsou čytři hodnoty pH a čtyři teploty.
Hn	Nastavení výkonu horkovzdušného topení ($n \in \langle 0, 100 \rangle$).
hn	Nastavení výkonu topení platformy ($n \in \langle 0, 100 \rangle$).
?	Ověření funkčnosti komunikace. Po přijetí příkazu modul odesílá zpět řetězec "GOLEM-2".

3.4 Řídicí aplikace v PC

Veškerá vyšší logika systému je realizována v aplikaci napsané ve Visual Basic, která běží na připojeném PC. Pro specifikaci běhu experimentu je použita CSV tabulka s předdefinovaným významem jednotlivých sloupců, kterou je možné do aplikace nahrávat a také z ní exportovat.

3.4.1 Hlavní obrazovka aplikace

Ve většině případů uživatel pracuje při experimentech s hlavní obrazovkou aplikace, na které je vidět aktuální stav systému a zároveň obsahuje prvky pro ruční změnu parametrů. Snímek této obrazovky je na obrázku 3.14.



Obrázek 3.14: Uživatelské rozhraní řídicí aplikace

Ústřední část obrazovky tvoří čtyři sloupce, každý s informacemi a možnostmi ovládání parametrů jednoho z vaků. Ovládací prvky ve sloupci jsou následující:

- informace o pH
- nastavení požadovaného pH
- možnost přidání kyseliny či zásady
- rozdíl aktuálního a požadovaného pH

- teplota (je dostupná jen pro první vak)
- nastavení teploty (také jen pro první vak)
- nastavení míchání
- nastavení vstřikování enzymu (dostupné jen pro jeden vak)
- tlačítko pro zaznamenání odběru vzorku (odběry se provádí manuálně, po zobrazení upozornění na odběr je nutné odběr potvrdit)

Nalevo a pod sloupci s nastavením vaků jsou informace o teplotě vzduchu a platformy s vaky a také možnost nastavení požadované teplot a zahájení vyhřívání. To se provádí vždy před zahájením experimentu, aby se teplota prostředí dostala na požadovanou hodnotu. Ve spodní části jsou (zleva) dostupná tlačítka pro zahájení/ukončení běhu experimentu, aktuální čas, záznam pH, nový log, načtení dat experimentu, zobrazení logu, otevření specifikace experimentu a také osmice tlačítek pro přečerpávání objemů mezi vaky.

3.4.2 Specifikace experimentů

Experimenty je možné specifikovat za pomoci textového zápisu ve formátu CSV. Každý řádek má svůj index a čas. Dále je možné specifikovat pH v jednotlivých vacích, přečerpávané objemy, jestli má v daném čase dojít k odběru vzorku, vstřikování enzymu a také nastavení rychlostí míchání. Záhlaví sloupců specifikačního souboru je v seznamu níže.

- Index
- Čas[min]
- Stomach
- Posun St.
- Duodenum
- posun Du.
- Jejunum
- posun Je.
- Ileum
- posun Il.
- Vzorek St.
- Vzorek Du.
- Vzorek Ju.
- Vzorek Il.
- Enzym St.
- Enzym Du.
- Kývač St.
- Kývač Du.
- Kývač Ju.
- Kývač Il.

3.4.3 Výstupní textový záznam

Součástí výstupu aplikace je textový soubor se záznamem událostí, ke kterým v průběhu experimentu došlo společně s aktuálním stavem zařízení. Jak je vidět na obrázku 3.15, je zaznamenáno datum a čas, provedená událost a také teplota a pH ve vacích.

```
      date/time          action       t [°C]   pH1    pH2    pH3    pH4
-------------------------------------------------------------------------
1/19/2009 9:45:59 AM ;  START      ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:51:36 AM ;  START      ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:51:52 AM ;  pH ?       ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:51:55 AM ;  pH ?       ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:51:58 AM ;  S-Stomach  ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:52:00 AM ; PUMP1(01ml);  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:52:19 AM ;  S-Jejunum  ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:52:20 AM ;  S-Ileum    ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:52:21 AM ;  pH ?       ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:52:22 AM ;  pH ?       ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:55:04 AM ;  S-Stomach  ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:55:10 AM ;  S-Duodenum;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:55:17 AM ;  S-Jejunum  ;  00.0;  02.0;  04.0;  06.0;  07.5
1/19/2009 9:55:27 AM ;  S-Ileum    ;  00.0;  02.0;  04.0;  06.0;  07.5
```

Obrázek 3.15: Textový výpis událostí zařízení Golem 2

Kapitola 4

Nové disoluční zařízení Golem 3

Požadavky na úpravy systému pocházejí hned z několika zdrojů. Nejdůležitější jsou samozřejmě ty od vědeckého týmu z VFU. Ty jsou založeny na zkušenostech s aktuální verzí systému a také na plánech jeho vylepšení a rozšíření. Součástí nové verze systému Golem je i webová aplikace, sloužící k jeho ovládání, kterou ve své diplomové práci zpracovává Bc. Jan Truhlář [49]. Důležitým faktorem je také požadavek na modularitu systému, který vychází ze zadání této diplomové práce.

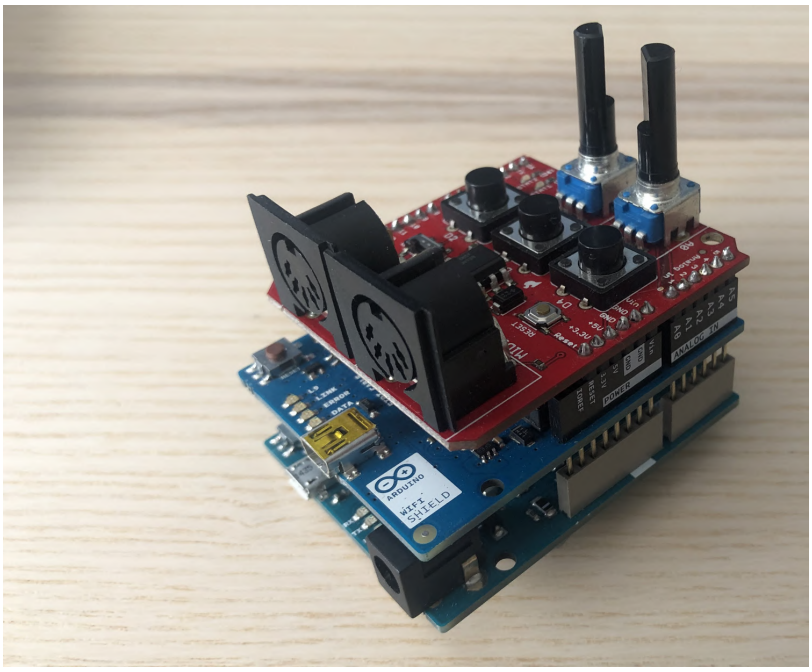
Z uživatelského hlediska jsou změny nejvíce zřetelné hlavně v části webové aplikace. Ta oproti předchozí verzi přináší například pokročilejší správu schémat experimentů, opakování běhů experimentu, grafy běhů a podobně. Výraznou změnou ale prošla i celá řídicí elektronika, která byla kompletně přepracována. Všechny hlavní funkční celky byly zachovány, ale došlo ke změně architektury řízení systému. Po mechanické stránce je Golem 2 modifikován minimálně.

4.1 Aspekty modulárního přístupu

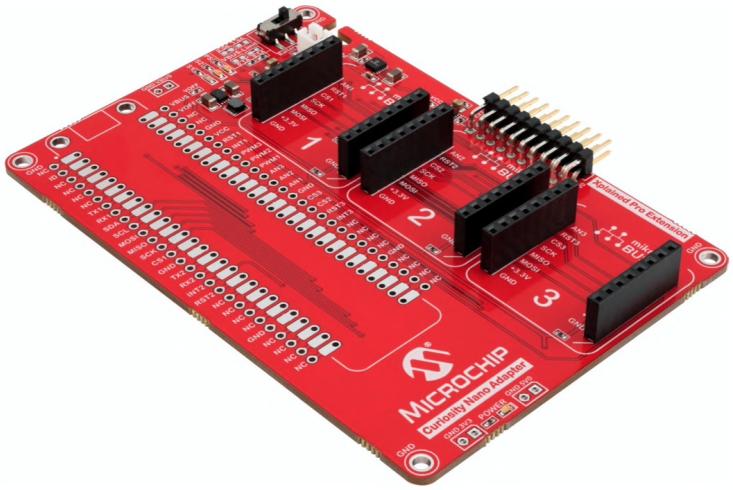
Návrh modulární architektury elektroniky a vytvoření řídicího systému tvoří hlavní část této práce. Na počátku tedy bylo nutné dobře zvážit, jakou formu by měl modulární systém mít. Jednou z možností bylo navržení systému obdobným způsobem, jakým jsou například tvořeny rozšiřující moduly pro platformu Arduino. Jedná se o desky, které mají často stejný tvar, jako samotná základní deska, které je možné zasunout do modulu Arduino a rozšířit tak jeho funkcionalitu. Díky tomu, že mají tyto moduly na spodní straně samčí a na horní straně samičí konektor, je možné je vrstvit na sebe. V tom případě je pak nutné pohlídat, jestli nejsou nějaké piny v konfliktu. Tento způsob uspořádání modulů je zachycen na obrázku 4.1.

Nevýhodou tohoto řešení je, že mohou nastávat konflikty řídicích pinů, například při použití více modulů stejného typu apod. Tomu by v případě potřeby šlo předejít použitím vhodné sběrnice pro komunikaci mezi moduly, například I2C. Další nevýhodou takového uspořádání modulů je jejich vzájemná blízkost. Kvůli tomu je náročné navrhnout rozložení konektorů tak, aby připojené protikusy nebyly v mechanickém konfliktu.

Kromě tohoto vertikálního způsobu připojování rozšiřujících desek existují i platformy, které umožňují horizontální připojování desek. Příkladem takové desky může být Curiosity Nano Base, která obsahuje patici pro připojení modulu s mikrokontrolérem a také tři menší patice pro zasunutí rozšiřujících modulů (tzv. *Click Board*) – viz obrázek 4.2.



Obrázek 4.1: Arduino a rozšiřující desky



Obrázek 4.2: Základní deska Curiosity Nano s paticemi pro připojení mikrokontroléru a rozšiřujících modulů Click Board

Nevýhodou tohoto přístupu je fakt, že musíme předem znát maximální počet modulů, které bude systém podporovat. Zároveň s přidáním dalších modulů roste i celková plocha základní desky. Tento přístup je vhodný, pokud jsou moduly dostatečně malé. V případě systému Golem by bylo třeba na některé rozšiřující moduly umístit například i BNC konektory, což by výrazně zvyšovalo celkovou plochu (obzvlášť v případě, kdy bychom chtěli zachovat stejné rozměry všech modulů).

Jako třetí se nabízí cesta, kdy jsou od sebe moduly fyzicky oddělené a komunikují po vhodné sběrnici. Tímto způsobem je navrženo nové řízení systému Golem. Díky tomu je možné přesně specifikovat rozhraní těchto modulů, zajistit jejich bezpečnost i z mechanického hlediska uložením do vhodné krabičky a také je možné je mít v systému v místě, kde je to nejvhodnější – moduly, ke kterým by měl mít uživatel přístup umístit do dostupné části zařízení, ostatní mimo dosah běžného uživatele apod.

4.2 Výběr komunikační sběrnice

Po zvolení modulární architektury bylo nutné vybrat vhodnou komunikační sběrnici, která by umožnila řízení celého systému. Zvažované komunikační sběrnice, včetně jejich kladů a záporů jsou uvedeny v tabulce 4.1.

Tabulka 4.1: Srovnání běžných komunikačních sběrnic

Sběrnice	**Popis**	
TTL UART	+	Plně duplexní komunikace
		Jednoduchá obsluha
		Malý počet vodičů pro komunikaci
	–	Určeno k propojení dvou zařízení
		Vhodné pro spojení na kratší vzdálenosti
SPI	+	Rychlost
		Jednoduchá obsluha
		Plně duplexní
	–	S počtem zařízení roste počet vodičů
		Vhodné pro komunikaci v rámci jedné desky
I2C	+	Možnost adresování
		Jednoduché přidávání dalších zařízení
	–	Poloduplexní
		Vhodné pro komunikaci v rámci jedné desky
RS232	+	Vhodné pro propojení zařízení na delší vzdálenost
		Možnost adresování (s vhodným protokolem)
	–	Navrženo pro propojení dvou zařízení
		Při použití s mikrokontroléry nutnost nutnost použití převodníku
RS485	+	Možnost komunikace více zařízení
		Komunikace na delší vzdálenost
		Diferenciální vedení odolné proti rušení
	–	V základu poloduplexní komunikace
		Při použití s mikrokontroléry nutnost nutnost použití převodníku
Ethernet	+	Rychlost
		Možnost adresování
		Komunikace na delší vzdálenost

	−	Náročné na zpracování
		Velký počet vodičů
CAN	+	Možnost adresování
		Komunikace na delší vzdálenost
		Konflikty řešeny přímo na základě protokolu
		Malý počet vodičů pro komunikaci
		Diferenciální vedení signálu
	−	Poloduplexní
		Nutnost použití převodníku

Na základě představených kritérií byla pro propojení modulů vybrána sběrnice CAN.

4.3 Sběrnice CAN

Protože je sběrnice CAN v projektu použita jako hlavní komunikační kanál, představíme si ji v této kapitole podrobněji. Dle dokumentu [51] se jedná o standardní protokol používaný pro komunikaci v počítačových systémech, který specifikuje fyzickou a datovou vrstvu modelu OSI. Jejím typickým uplatněním je automotive průmysl, kde slouží ke komunikaci mezi různými zařízeními ve vozidle.

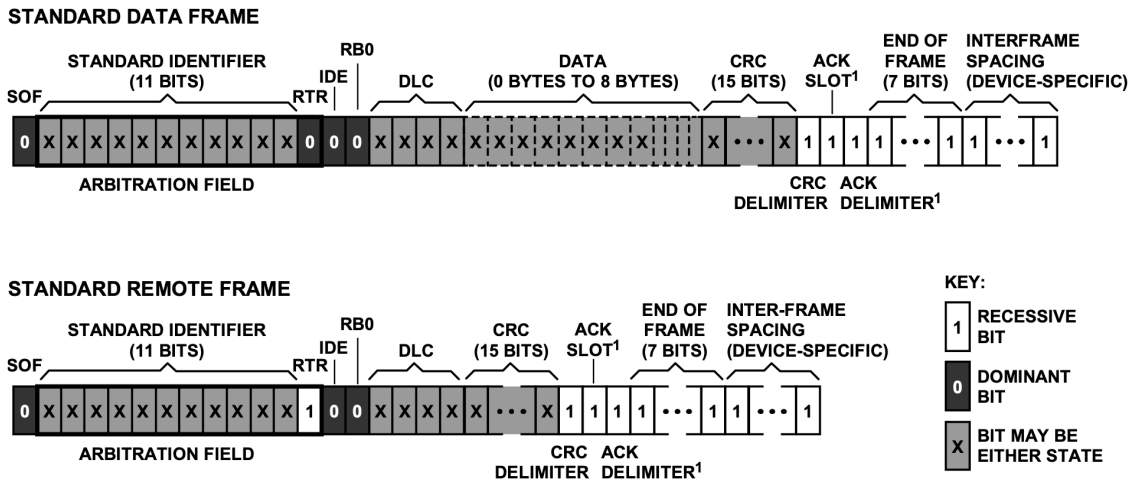
Komunikace po CAN sběrnici je založena na výměně krátkých zpráv a je decentralizovaná. To umožňuje mít na jedné sběrnici i více master zařízení. Pokud se na sběrnici snaží vysílat více zařízení současně, dostane přednost zařízení, které vysílá zprávu o nižším ID.

Sběrnice je asynchronní (komunikační frekvence musí být všem uzlům předem známá), poloduplexní (na sběrnici může vysílat vždy jen jedno zařízení) a k výměně zpráv používá diferenciální pár, často v podobě kroucené dvojlinky. To zajišťuje zvýšenou odolnost proti rušení a zvyšuje tak spolehlivost komunikace. Kromě dvou diferenciálních vodičů (označovaných *CANH* a *CANL*) již nejsou potřeba další signální vodiče, ale většinou se k tomuto páru přidávají ještě dva vodiče pro napájení [51]. Na koncích sběrnice je z důvodu minimalizace odrazů nutné použít zakončovací resistory.

Sběrnice může být ve dvou stavech – *dominantním* nebo *recesivním*. V dominantním stavu je na CANH vodiči kladné napětí, na CANL napětí 0 V. V recesivním stavu jsou všechny budiče ve stavu vysoké impedance. Logická 0 je poté reprezentována dominantním stavem, logická 1 recesivním. Nečinnost sběrnice je detekována na základě čítání počtu recesivních bitů po posledním odeslaném rámci.

CAN rozlišuje několik typů rámců - *datový*, *řídicí*, *chybový* a rámec signalizující zahlcení [37]. Nejčastěji používaným rámcem je datový rámec, který slouží k přenosu dat mezi uzly. Řídicí rámec slouží k tomu, aby mohly řídicí uzly zažádat o rámec s požadovaným ID. Struktura datového a řídicího rámce je na obrázku 4.3. Chybový rámec je generován zařízeními při detekci chyby přenosu. Rámce signalizující zahlcení jsou na sběrnici posílány ve chvíli, kdy je potřeba zanést zpoždění mezi datové či řídicí rámce. Struktura datového rámce je podrobněji popsána v tabulce 4.2.

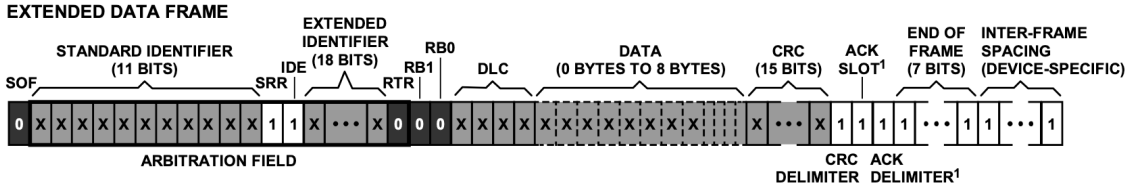
Kromě standardního formátu podporuje CAN sběrnice od verze 2B i rozšířený formát. Ten se od standardní verze liší tím, že rozšiřuje pole identifikátoru o dalších 18 bitů (identifikátor je tedy celkem 29 bitů dlouhý). Struktura rozšířeného datového rámce je vidět na obrázku 4.4.



Obrázek 4.3: Standardní datový (nahoře) a řídicí (dole) rámec CAN sběrnice [51]. Začátek rámce je signalizován dominantním hodnotou, po které následuje 11 bitů identifikátoru zprávy. Následující bit (*RTR*) určuje, jestli se jedná o řídicí rámec. Bit *IDE* určuje, zda se jedná o standardní, či rozšířený rámec (více viz dále). Pole *DLC* obsahuje délku přenášených dat. V poli *CRC* najdeme kontrolní součet a v závěru zprávy je prostor pro potvrzující bit, který odesílá přijímající zařízení.

Tabulka 4.2: Pole datového rámce CAN sběrnice

Pole	**Délka (bit)**	**Význam**
SOF	1	Bit označující začátek rámce
ID	11	Identifikátor zprávy. Slouží i k prioritizaci.
RTR	1	Výzva k přenosu dat (využívá řídicí rámec)
IDE	1	Typ identifikátoru (0 = standardní, 1 = rozšířený)
RB0	1	Rezervovaný bit
DLC	4	Délka dat v bytech (0 až 8)
Data	0 až 64	Prostor pro data
CRC	15	Cyklický redundantní součet
CRC oddělovač	1	Bit oddělující CRC a ACK
ACK	1	Potvrzující bit. Odesílá ho zařízení přijímající zprávu
ACK oddělovač	1	Bit oddělující ACK a EOF
EOF	7	Konec rámce



Obrázek 4.4: Rozšířený datový rámec CAN sběrnice [51]

4.4 Komunikační protokol

Nad CAN rámci byl vytvořen proprietární protokol sloužící k řízení a komunikaci v systému Golem. Protokol je navržen tak, že na jedné sběrnici je možné mít jedno master zařízení a až 128 slave zařízení. Slave zařízení mohou komunikovat pouze s master zařízením, které je zároveň vždy iniciátorem komunikace. Zmíněné vlastnosti sice omezují možnosti CAN sběrnice, ale to pro účely projektu nevadí.

Protokol využívá rozšířené CAN datové rámce s 29 bitovým identifikátorem. Protože je vhodné mít k dispozici maximální počet (8) bytů pro data, není datové pole využito pro přenos jiných informací, než samotných dat. Kvůli tomu jsou všechny řídicí informace vepsány do ID rámce.

Navržený protokol také podporuje hierarchické adresování s možností adresovat patnáct typů modulů a šestnáct instancí každého modulu. To mj. usnadňuje vývoj, neboť je z odchycených zpráv vždy zřejmé kdo je odesilatel a kdo příjemce. Některé CAN periferie navíc umožňují filtrovat příchozí zprávy na základě bitové masky identifikátoru, čehož je možné při zvoleném způsobu adresování jednoduše využít. Rozdělení pole identifikátoru pro účely adresování a řízení zařízení Golem je vidět v tabulce 4.3.

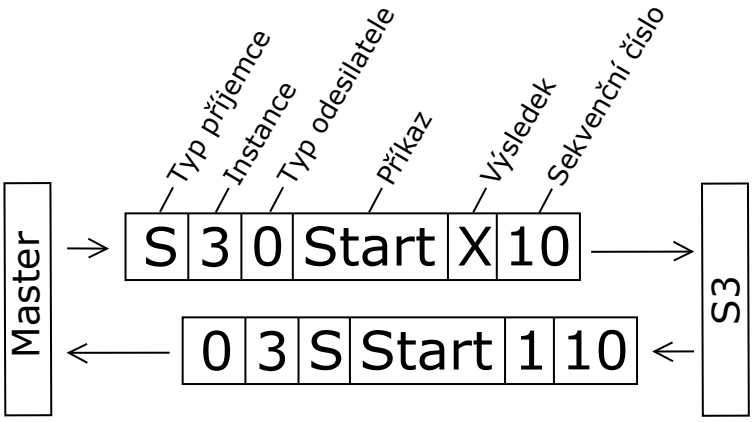
Tabulka 4.3: Hlavička proprietárního protokolu nad CAN sběrnicí, sloužícího pro řízení částí disolučního zařízení Golem (seřazeno od nejvýznamnějších bitů)

Název pole	**Bity**	**Popis**
Typ příjemce	4	Typ zařízení, které je adresátem zprávy.
Instance	4	Instance zařízení. Při komunikaci ve směru od master zařízení je v tomto poli instance cíle. Ve směru od slave zařízení je v poli instance odesílatele.
Typ odesílatele	4	Typ zařízení, které odesílá zprávu.
Příkaz	9	Kód příkazu
Výsledek	1	Bit s výsledkem příkazu (0 = neúspěch, 1 = úspěch)
Sekvenční číslo	7	Sekvenční číslo zprávy

Protokol je navržen tak, aby zprávy směřující od slave k master zařízení byly prioritizovány. To je zařízeno umístěním adresy příjemce na začátek, přiřazením typu 0 master zařízení a také tím, že CAN sběrnice prioritizuje zprávy s nižším ID.

Všechny slave moduly podporují základní sadu příkazů, sloužící ke spuštění, pozastavení, restartování a také získání informací o stavu modulu. Tyto základní příkazy jsou popsané v kapitole 6.2.3. Kromě toho má každý modul svoji vlastní sadu příkazů dle jeho účelu. Popis těchto příkazů je uveden vždy v části popisující konkrétní modul v kapitole 6.

Komunikaci iniciuje vždy master zařízení odesláním příkazu. Na ten slave zařízení odpovídá zprávou s totožným sekvenčním číslem a kódem příkazu a také informací o tom, jestli se příkaz zdařil, nebo ne (pole *výsledek*). Součástí některých odesílaných a přijímaných zpráv mohou být i data. Příklad komunikace je vidět na obrázku 4.5.



Obrázek 4.5: Příklad hlaviček při výměně zpráv úspěšné inicializace modulu S3 (S = typ, 3 = instance)

4.5 Rozdělení systému do modulů

Celý systém je možné rozdělit do částí, které reflektují současné technické a mechanické řešení zařízení Golem a také logické celky systému. Na základě analýzy předchozích verzí zařízení a požadavků na nový systém bylo navrženo rozdělení, které je (společně s číslem typu modulu a počtem jeho instancí v systému) zaneseno v tabulce 4.4. Každý modul má přesně dané rozhraní, příkazy, které přijímá, a také část systému, jejíž řízení má na starosti.

Tabulka 4.4: Typy modulů v systému, jejich označení a počet instancí

Modul	**Číslo typu**	**Počet instancí**
Master zařízení	0	1
Regulace pH	1	1
Řízení míchání a přečerpávání vaků	2	4
Regulace topení	3	1
Vstřikování enzymu	4	1

4.6 Výběr cílové platformy

Jednotlivé moduly bylo možné realizovat několika způsoby. Prvním z nich bylo využití již existující platformy, jako je například Arduino, Teensy a podobně. Nevýhodou tohoto přístupu je menší možnost přizpůsobení tvaru a velikosti a také vazba na existující rozšiřující desky, popřípadě nutnost tvorby nové desky. Proto bylo přistoupeno přímo k tvorbě vlastních desek s vhodným mikrokontrolérem. Hlavní parametry pro výběr mikrokontroléru byly následující:

- CAN periferie - přítomnost CAN periferie přímo na čipu
- analogově-digitální převodník – u analogových senzorů je vhodné rozlišení alespoň 12-bit (pro pH sondy to dává teoretické rozlišení až 0.004 pH).
- Standardní sběrnice (UART, SPI, I2C)

- Malé rozměry

Na základě těchto kritérií byl vybrán mikrokontrolér z řady Kinetis KV11 s označením MKV11Z128VFM7. Jedná se o mikrokontrolér v pouzdře HVQFN32 s CAN periferií a šestnáctibitovým analogově-digitálním převodníkem. Jeho další vlastnosti jsou vidět v tabulce 4.5.

Tabulka 4.5: Vlastnosti mikrokontroléru MKV11Z128VFM7 dle dokumentace [32]

Parametr	**Hodnota**	**Parametr**	**Hodnota**
Typ jádra	Arm Cortex-M0+	PWM	20x 16 bit
Maximální frekvence	75 MHz	ADC	2x 16 bit
Flash	128 kB	DAC	1x 12 bit
SRAM	16 kB	GPIO	28
Sériová komunikace	1x I2C, 1x SPI, 2x UART	Napájecí napětí	1.71 V až 3.6 V
CAN	1x	Pracovní teplota	−40 °C až 105 °C
Bezpečnostní periferie	CRC, FAC		

Dále bylo nutné vybrat vhodné master zařízení. Jednou z možností bylo opět využít mikrokontrolér MKV11Z128VFM7. To bylo ale zavrženo, protože by bylo náročné zařídit, aby mikrokontrolér zvládnul prostřednictvím WiFi připojení uživatelům poskytnout webovou aplikaci, ukládat množství dat z experimentů apod. Proto byl nakonec tento mikrokontrolér použit pouze jako chytřejší převodník CAN sběrnice a o samotné řízení se stará Linuxový počítač Raspberry Pi verze 4. Ten je vybavený slotem pro SD kartu, WiFi modulem, Ethernet rozhraním a USB porty a v případě potřeby i HDMI výstupem, takže je pro účel poskytování webové aplikace pro řízení experimentu vhodnější. S mikrokontrolérem, použitým jako CAN převodník, komunikuje Raspberry Pi pomocí sběrnice UART.

Kapitola 5

Specifikace funkčních bloků

Při tvorbě systému byly moduly pro snazší návrh dále rozděleny do menších bloků (dále jako *funkční bloky*, nebo *bloky*). Některé bloky tvoří pouze elektronická část v podobě schématu a rozložení součástek na desce plošných spojů (jako například blok zdroj 1 V referenčního napětí). Některé bloky jsou pouze softwarové – například ovládání časovače mikrokontroléru či řízení PWM. Zbylé bloky, jako například řízení krokových motorů, obsahují jak hardwarovou, tak softwarovou část. Každý z modulů má na starost omezenou oblast řešeného problému. Protože moduly vznikly právě skládáním těchto bloků, představíme si nejdříve jednotlivé bloky a až poté jejich samotné složení do modulů.

Firmware pro obsluhu jednotlivých bloků je psaný v jazyku C. Části, obsluhující jednotlivé periferie použitého mikrokontroléru, využívají knihovnu standardních funkcí (dále jako SDK), kterou pro MKV11Z128VFM7 poskytuje jeho výrobce – viz [11]. Všechny zdrojové soubory, stejně jako podklady pro výrobu desek plošných spojů, jsou dostupné v příloze.

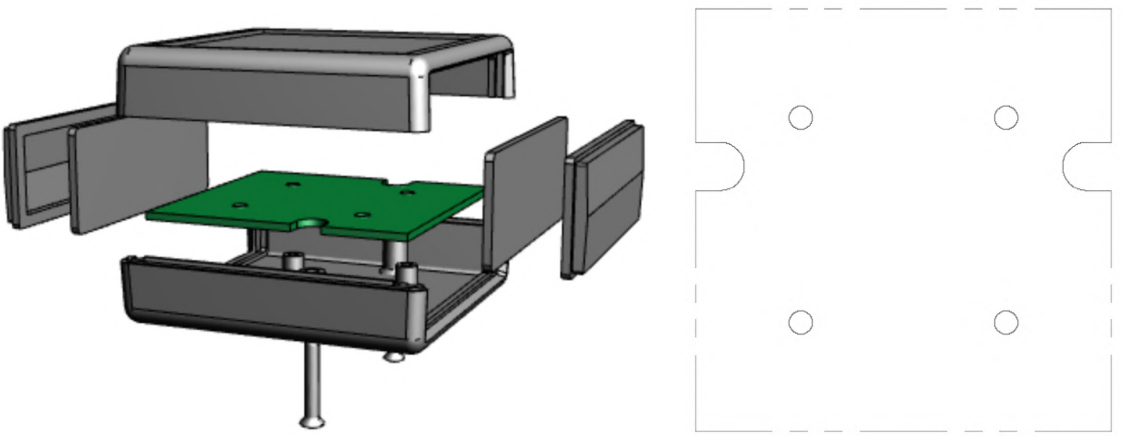
5.1 Společný základ modulů

Po zvolení modulární architektury bylo nutné vybrat vhodné fyzické provedení. Od něj se potom dále odvíjel vývoj desek jednotlivých modulů. Po průzkumu různých dostupných krabiček byly vybrány ty od společnosti Hammond ze série 1593 (viz [2]). Krabičky jsou černé a mají rozměr 66 mm × 66 mm × 28 mm. Jejich výrobce dodává i 3D model desky plošných spojů, kterou lze do krabičky uložit (viz [9]). Snímek modelu krabičky a tvar desky plošných spojů jsou vidět na obrázku 5.1.

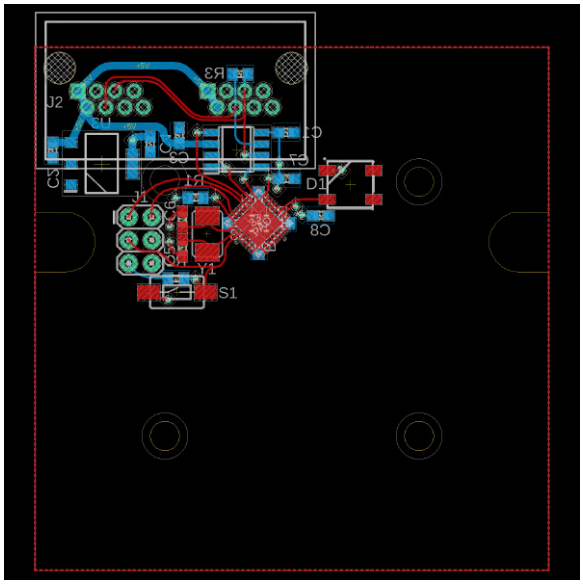
Desky plošných spojů byly zvoleny dvouvrstvé, přičemž na horní vrstvu je připojeno napětí 3.3 V a na spodní vrstvu 0 V. Deska plošných spojů, obsahující všechny komponenty společného základu, je vidět na obrázku 5.2.

5.1.1 Mikrokontrolér

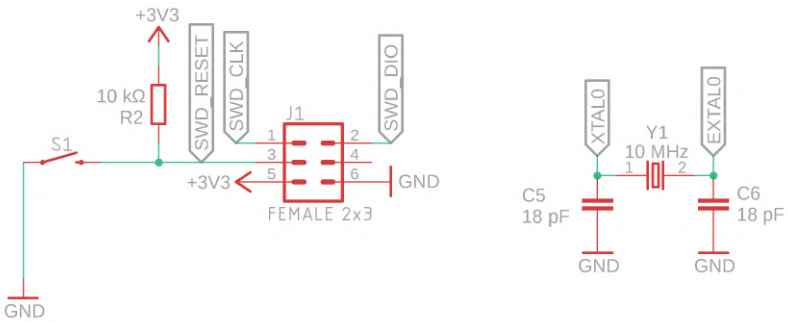
Na všech modulech nalezneme mikrokontrolér MKV11Z128VFM7 v pouzdře HVQFN32. K němu jsou v základu kromě bypass kondenzátorů připojeny ještě resetovací tlačítko, stavová LED, programovací konektor, CAN převodník a 10 MHz oscilátor. K programování je použito rozhraní SWD. Na obrázku 5.3 je vidět schéma programovacího konektoru a zapojení oscilátoru.



Obrázek 5.1: Model krabičky a tvar desky plošných spojů dostupné z [9].



Obrázek 5.2: Deska plošných spojů obsahující komponenty společného základu modulů

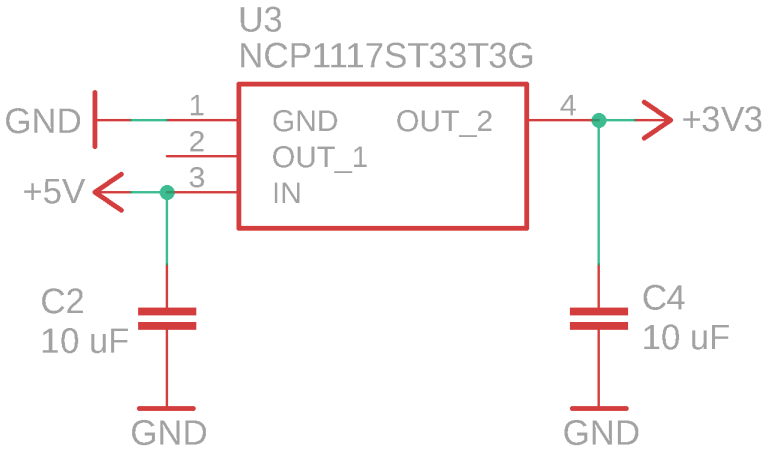


Obrázek 5.3: Zapojení SWD programovacího konektoru a 10 MHz oscilátoru

Zapojení oscilátoru je stejné, jako zapojení použité u modulu FRDM-KV11Z, což je evaluační deska pro mikrokontroléry řady KV11 (více informací na webu desky [8]). Z příkladů pro tuto desku, které jsou dostupné v SDK, je převzato také nastavení hodin, které je umístěno v souboru *clock_config.h*. Díky FLL obvodům uvnitř čipu je ze vstupních 10 MHz odvozena frekvence jádra 75 MHz.

5.1.2 Zdroj napětí

Na všechny desky je společně s vodiči sběrnice CAN přivedeno i napájení 5 V. Protože použitý mikrokontrolér pracuje s maximálním napětím 3.6 V, musí být toto napětí sníženo. K tomuto účelu je použit regulátor s nízkým úbytkem z řady NCP1117 od společnosti ON Semiconductor (datasheet viz [22]). Regulátory z této řady jsou schopné poskytnout výstupní proud až 1 A. Byl vybrán regulátor s výstupním napětím 3.3 V, který je zapojen dle doporučení výrobce (viz [22]). Schéma zapojení je na obrázku 5.4.

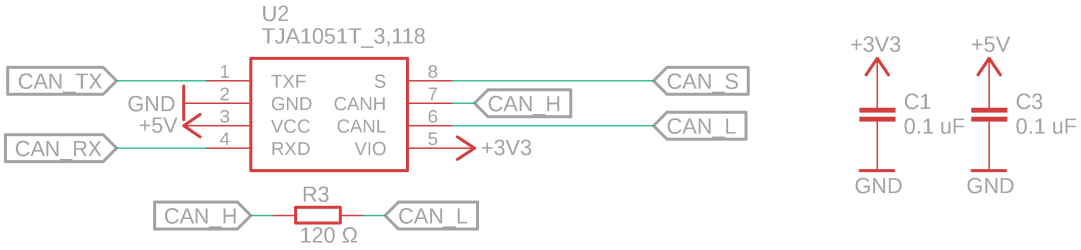


Obrázek 5.4: Schéma zapojení regulátoru napětí 3.3 V dle doporučení výrobce z dokumentace [22].

5.1.3 CAN

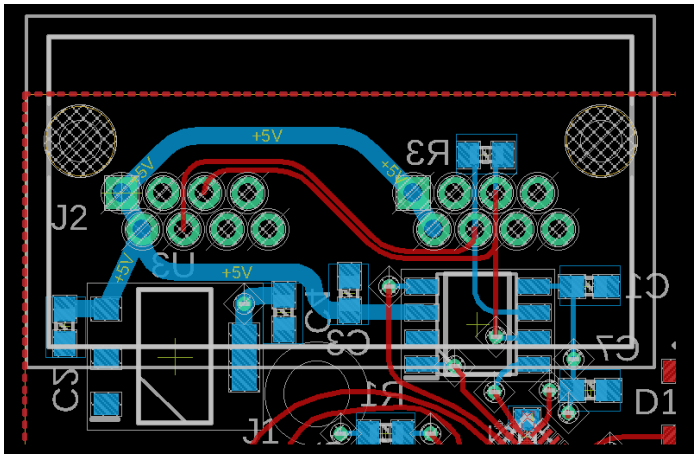
Použitý mikrokontrolér je vybaven periferií, implementující datovou vrstvu CAN sběrnice, která ale není schopná generovat diferenciální signál. Z toho důvodu je na desce použit převodník TJA1051T od firmy NXP, implementující fyzickou vrstvu CAN sběrnice [25]. Kromě toho každý modul obsahuje plochu pro připojení zakončovacího resistoru. Ten je osazen pouze na modulech na koncích sběrnice. Schéma zapojení CAN převodníku je na obrázku 5.5. Protože CAN sběrnice pracuje na napětí 5 V je k převodníku přivedeno jak napájecí napětí mikrokontroléru, tak zmíněných 5 V.

Jako konektor pro propojení modulů a přivedení napájení byl vybrán konektor RJ45, převážně z důvodu existence velké řady kvalitních kabelů různých délek, provedení a stínění. Na každém modulu jsou dva tyto konektory, aby bylo možné za sebou moduly řetězit. Dvojice prostředních kontaktů konektoru, která odpovídá jednomu diferenciálnímu páru vodičů v přímém kabelu RJ45, je využita pro vedení signálů CAN sběrnice. Dále je po jedné dvojici využito pro vedení zemního potenciálu a pro vedení napájecího napětí 5 V. Jedna dvojice tak zůstává nevyužita. Na obrázku 5.6 je vidět část desky obsahující dvojitý konektor



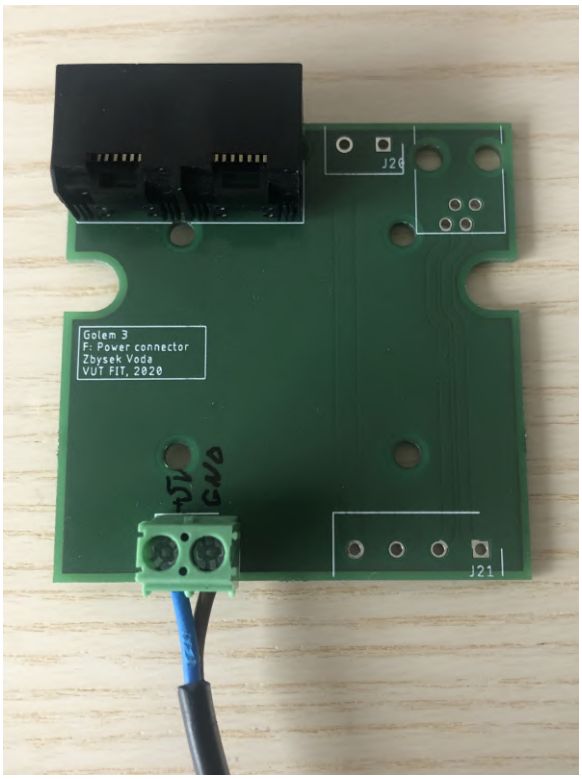
Obrázek 5.5: Schéma CAN převodníku

RJ45, mezi jehož dvěma porty je po svrchní straně desky (červeně) esovitě vedena dvojice vodičů *CANH* a *CANL*. Modře značené komponenty leží na spodní straně desky. Jedná se o regulátor napětí (vlevo dole), CAN převodník (vpravo dole) a resistor pro zakončení sběrnice (vpravo nahoře, zrcadlově označen R3).



Obrázek 5.6: Konektor pro připojení CAN sběrnice a esovité vedení diferenciálního páru po desce plošných spojů

Pro účely přivedení napájení 5 V k použité sběrnici byla navržena i jednoduchá deska se svorkovnicí pro připojení zdroje napětí na jedné straně a RJ45 dvojkonektoru na straně druhé – viz obrázek 5.7.



Obrázek 5.7: Deska plošných spojů sloužící k připojení napájecího napětí 5 V k použitému rozhraní.

Softwarová obsluha CAN sběrnice je implementována v souboru *can.c*. Přijímání rámců sběrnice probíhá asynchronně na základě přerušení z CAN periferie mikrokontroléru. Po přijetí je rámec pouze uložen do FIFO fronty čekajících rámců, která je implementována jako kruhový buffer. Ve výpisu 5.1 je vidět ukázka funkce obsluhující příchozí rámce při přerušení.

```
volatile flexcan_frame_t _CAN_incomingPackets[CAN_INCOMING_PACKETS_BUFFER_LENGTH];
volatile uint8_t _CAN_incomingPacketsStartIndex;
volatile uint8_t _CAN_incomingPacketsStopIndex;
volatile uint8_t _CAN_incomingPacketsCounter;

void CAN0_IRQHandler(void) {
    if (0U != FLEXCAN_GetMbStatusFlags(CAN0, 1U << CAN_RX_BUF_NUM)) {
        if (_CAN_incomingPacketsCounter == CAN_INCOMING_PACKETS_BUFFER_LENGTH) {
            // buffer full
        } else {
            uint8_t newIndex = (_CAN_incomingPacketsStopIndex + 1) % \
                CAN_INCOMING_PACKETS_BUFFER_LENGTH;

            (void) FLEXCAN_ReadRxMb(
                CAN0,
                CAN_RX_BUF_NUM,
                (flexcan_frame_t * )(_CAN_incomingPackets + newIndex)
            );
            _CAN_incomingPacketsStopIndex = newIndex;
            _CAN_incomingPacketsCounter++;
        }

        FLEXCAN_ClearMbStatusFlags(CAN0, 1U << CAN_RX_BUF_NUM);
    }
}
```

Výpis 5.1: Funkce ukládající příchozí rámce CAN sběrnice do FIFO fronty

Ostatní zdrojové soubory nemají přímý přístup k frontě, ale k získání čekajících bufferů využívají funkce, které jsou exportovány v hlavičkovém souboru *can.h*. Tyto funkce jsou popsané v tabulce 5.1.

Pro účely jednodušší práce jsou CAN rámce v programu reprezentovány jako datová struktura z výpisu níže, která obsahuje všechny informace, které jsou pro odesílání a příjem rámce podstatné. Tato struktura je vidět ve výpisu 5.2.

```
typedef struct {
    uint8_t data[8];
    uint8_t from;
    uint8_t to;
    uint8_t instance;
    uint16_t command; uint8_t success;
    uint8_t sequence; uint8_t length;
} T_CAN_PACKET;
```

Výpis 5.2: Struktura pro práci s CAN rámci v programu

Tabulka 5.1: Funkce pro obsluhu CAN sběrnice exportované v hlavičkovém souboru *can.h*

`void CAN_init(`
`  uint8_t rxId,`
`  uint8_t rxInstance,`
`  uint32_t mask`
`)`
	Inicializace CAN periferie a nastavení filtrování příchozích rámců dle typu a id modulu.

`void CAN_write(`
`  T_CAN_PACKET *packet`
`)`
	Odeslání CAN rámce. Je implementováno blokujícím způsobem a rámec je odeslán ihned po zavolání funkce.

`uint8_t CAN_readNext(`
`  T_CAN_PACKET *packet`
`)`
	Pokud není FIFO fronta prázdná, je do parametru *packet* uložena hodnota prvního rámce ve frontě a funkce vrací hodnotu 1. V opačném případě je vrácena hodnota 0.

Na této vrstvě staví ještě jedna vrstva, starající se o CAN komunikaci. Ta již více reflektuje navržený protokol (viz kapitola 4.4) a je využívána master modulem. Obsahuje frontu čekajících příkazů, které jsou synchronně vyčítány a odesílány. U odeslaných příkazů je evidován čas odeslání, aby bylo možné detekovat případný výpadek. Pokud dojde k návratu rámce s informací o chybě (bit *výsledek* = 0), dochází k opětovnému vyslání stejného příkazu (celkem tři pokusy). Také je možné nastavit, jaká funkce se má zavolat po obdržení rámce potvrzující úspěšné provedení příkazu (ten může i nést data). V této vrstvě je nadefinovaná i délka parametrů a dat jednotlivých příkazů, takže je možné kontrolovat validitu odesílaných a přijímaných dat. Funkce pro obsluhu této vrstvy exportuje hlavičkový souboru *can_cmd_queue.h* a jsou vidět v tabulce 5.2.

Tabulka 5.2: Funkce pro obsluhu fronty čekajících CAN příkazů

`void CAN_CMD_QUEUE_init(`
`  T_CAN_FIFO_CALLBACK callback`
`)`
	Inicializace fronty příkazů. Parametrem *cb* je možné nastavit funkci volanou při obdržení úspěšného potvrzení příkazu.

`uint8_t CAN_CMD_QUEUE_addCommand(`
`  uint8_t canId,`
`  uint8_t instance,`
`  T_CMD command,`
`  uint8_t messageId,`
`  const void *data`
`);`
	Vloží příkaz a jeho parametry do fronty.

`void CAN_CMD_QUEUE_tick(`
`  T_CAN_PACKET *canPacket`
`)`
	Funkce volaná při každé iteraci hlavní smyčky.

5.1.4 Stavová LED

Pro účely ladění systému obsahuje společný základ i jednu RGB LED. Díky ní je možné například vizualizovat stav modulu, popřípadě přijetí či provedení příkazu. Zvolena byla digitálně řízená LED WS2812B – datasheet viz [20]. Tato LED je řízena jedním vodičem a vyžaduje přesné časování. Jeden bit má vždy fixní délku 1.25 µs, logická hodnota je určena poměrem vysoké a nízké úrovně napětí. Každá z barev má rozlišení 8 bitů, takže je celý řídicí

rámec dlouhý 24 bitů. Vytvořeny byly také obslužné funkce, které jsou dostupné v souboru *WS2812B.h*. Jejich výpis je v tabulce 5.3.

Tabulka 5.3: Funkce dostupné pro obsluhu digitálně řízených RGB LED WS2812B

`void WS2812B_init(`
`  uint32_t idColor`
`)`
	Inicializace LED. Pin použitý pro obsluhu LED je nastaven jako výstupní a LED blikne nastavenou barvou.

`void WS2812B_set(`
`  uint8_t red,`
`  uint8_t green,`
`  uint8_t blue`
`)`
	Trvalé nastavení barvy LED

5.2 Časovač, měření času, PWM

Zvolený mikrokontrolér obsahuje několik časovačů, které je možné využít pro potřeby uživatele. V projektu jsou využity dva časovače, v SDK označené jako *FTM1* a *FTM2*. První z nich má nastavenou frekvenci 1 kHz a slouží k čítání milisekund uplynulých od začátku běhu programu. Při obdržení přerušení z časovače je volaná funkce pro obsluhu přerušení, ve které je inkrementována hodnota globální proměnné *TIMER1_millis*, kterou mohou číst všechny další funkce a odvozovat od ní své akce. Například je informace o času využita při odeslání CAN packetu – viz kapitola 5.1.3.

Druhý z časovačů má frekvenci 25 Hz a slouží ke generování pomalého PWM, které je použito pro řízení topení – více viz kapitola 5.7.

5.3 Ovládání vstupů a výstupů

Pro možnost rychlého nastavování vstupů a výstupů je v projektu dostupná dvojice souborů *pin_control.c* a *pin_control_macros.h*, obsahující makra pro spuštění hodin portu a také nastavování módu pinu. Druhý ze souborů byl sestaven generátorem, vytvořeným autorem tohoto projektu. Vstupem generátoru je tabulka z dokumentace mikrokontroléru [31], která specifikuje funkce jednotlivých pinů. Výstupem je soubor maker sloužících k ovládání pinu. V tabulce 5.4 jsou zdokumentována makra, používaná pro práci s pinem.

Makra pro obsluhu analogových vstupů volají ve skutečnosti funkce *ADC_initPolling* a *ADC_readPolling*, implementované v souboru *pin_control.c*. Slouží ale k předvyplnění vhodných parametrů těchto funkcí.

5.4 PID regulátor

Moduly pro řízení teploty a regulaci pH využívají ke své funkci PID regulátor. Jedná se o typ regulátoru, který k určení výstupní hodnoty používá **P**roporcionální, **I**ntegrační a **D**erivační člen [5]. Poměr vah těchto členů je možné nastavit pomocí konstant regulátoru (k_p, k_i, k_d). Jejich volba patří mezi hlavní výzvy při používání PID regulátorů. Proporcionální člen zkoumá pouze odchylku aktuálního vstupu od požadované hodnoty. Při použití pouze tohoto typu regulátoru má výstup tendenci oscilovat. Integrační člen sčítá tuto odchylku v čase,

Tabulka 5.4: Příklad maker pro práci s jedním pinem ze souboru *pin_control_macros.h*

`PTD6_GPIO_SET_OUTPUT()`	Nastaví pin jako digitální výstup
`PTD6_GPIO_WRITE0()`	Nastaví na pinu logickou 0
`PTD6_GPIO_WRITE1()`	Nastaví na pinu logickou 1
`PTD6_GPIO_WRITE(BIT)`	Nastaví na pinu hodnotu *BIT*
`PTD6_GPIO_TOGGLE()`	Zneguje logickou hodnotu na pinu
`PTD6_GPIO_SET_INPUT()`	Nastaví pin jako digitální vstup
`PTD6_GPIO_SET_INPUT_PULLUP()`	Nastaví pin jako digitální vstup s vnitřním pull-up resistorem
`PTD6_GPIO_SET_INPUT_PULLDOWN()`	Nastaví pin jako digitální vstup s vnitřním pull-down resistorem
`PTD6_GPIO_READ()`	Vrátí logickou hodnotu na digitálním pinu
`PTD6_ADC1_SET_POLLING()`	Nastaví pin jako analogový vstup se synchronním čtením
`PTD6_ADC1_READ_POLLING()`	Přečte analogovou hodnotu na pinu

takže je schopný regulovat i drobné výchylky od požadované hodnoty. Derivační člen zkoumá změnu aktuální odchylky a je tak schopný reagovat i na rychlé změny vstupní hodnoty. PID regulátor je v projektu reprezentován strukturou, ve které jsou uloženy konstanty regulátoru a také hodnoty integrálu a poslední chyby. Tato struktura je vidět ve výpisu 5.3.

```
typedef struct {
    float I;
    float D;
    float kP;
    float kI;
    float kD;
    float lE; // last error
} T_PID;
```

Výpis 5.3: Struktura pro uložení PID regulátoru

Funkce pro práci s touto strukturou a výpočet odezvy PID regulátoru jsou implementovány v souboru *pid.c* a jsou zdokumentovány v tabulce 5.5.

Tabulka 5.5: Funkce pro práci s PID regulátorem

`void PID_init(`
`  T_PID *pid,`
`  float kP,`
`  float kI,`
`  float kD`
`)`
	Inicializace regulátoru v parametru *pid*

`float PID_response(`
`  T_PID *pid,`
`  float target,`
`  float current`
`)`
	Výpočet odezvy jednoho kroku regulátoru v parametru *pid*

5.5 UART

Nad SDK funkcemi pro obsluhu UART periferie je implementováno rozhraní pro snazší práci s UART. Fronta příchozích znaků je implementována jako kruhový buffer a je plněna asynchronně při obdržení přerušení z periferie. Vyčítání již probíhá synchronně, stejně, jako odesílání. Funkce dostupné v knihovně jsou zaneseny v tabulce 5.6.

Tabulka 5.6: Funkce pro obsluhu UART periferie

`void UART_init(`
`  uint32_t baudrate`
`)`
	Inicializace komunikace po UART sběrnici se zvolenou rychlostí

`uint8_t UART_available()`	Počet dostupných bytů v bufferu

`void UART_readAndFlush(`
`  uint8_t *buffer,`
`  uint8_t *length`
`)`
	Nakopíruje maximálně *length* bytů do pole *buffer* a nastaví *length* na zkopírovanou délku. Po přečtení smaže byty z bufferu.

`void UART_readFromOffsetAndFlush(`
`  uint8_t *buffer,`
`  uint8_t offset,`
`  uint8_t *length`
`)`
	Stejné, jako UART_readAndFlush, ale začne číst byty s posunutím.

`int16_t UART_peekIndex(`
`  uint8_t i`
`)`
	Vrátí byte s indexem i od aktuálního začátku bufferu

`void UART_readFromStart(`
`  uint8_t *buffer,`
`  uint8_t *length`
`)`
	Stejné, jako UART_readAndFlush, jen po přečtení nemaže přečtené byty

`void UART_write(`
`  uint8_t *data,`
`  uint32_t length`
`)`
	Odeslání pole bytů

`void UART_flush()`	Smaže obsah bufferu

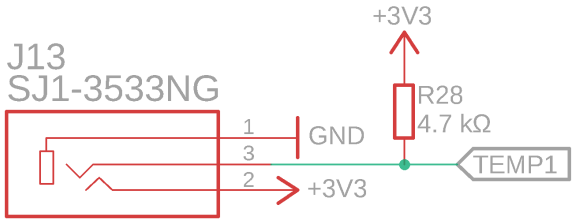
`void UART_unwind(`
`  uint8_t length`
`)`
	Posune začátek bufferu o *length* bytů dále (pokud jsou zde byty ke čtění)

`int8_t UART_findPos(`
`  char c`
`)`
	Vrátí pozici prvního výskytu zadaného znaku od aktuálního začátku příchozího bufferu

5.6 Měření teploty

K měření teploty byly vybrány čidla DS18B20. Jedná se o digitální teploměry s volitelným rozlišením 9 až 12 bitů, schopné měřit v rozsahu −55 °C až 125 °C. Teploměry komunikují prostřednictvím sběrnice OneWire, díky které je možné komunikovat s více teploměry na jednom vodiči. Toho ale není v projektu využito a každý teploměr má svůj vlastní konektor, takže je možné jednoduše rozlišit, o který teploměr se jedná (teplota vaku, platformy,

atd.). Pro připojení čidel byl zvolen klasický 3.5 mm konektor. Jeho zapojení je vidět na obrázku 5.8.



Obrázek 5.8: Schéma zapojení komnektoru teploměru DS18B20

5.6.1 OneWire sběrnice

OneWire sběrnice umožňuje komunikaci prostřednictvím jednoho vodiče. Zařízení mohou na sběrnici nastavit explicitně pouze logickou nulu. Logické jedničky je docíleno pomocí pull-up resistoru [34]. Díky tomu může na sběrnici komunikovat více zařízení současně bez rizika, že jedno zařízení zapíše explicitně log. 1 a druhé log. 0. Tím by mohlo dojít ke zkratu a poškození portů zařízení.

Pro účely komunikace po OneWire sběrnici obsahuje projekt funkce implementované v souboru *one_wire.c*. Ty vycházejí z kódu do laboratoří předmětu NAV pro obsluhu OneWire sběrnice, použitého s laskavým svolením Ing. Šimka, a dále upraveny dle doporučení z webových stránek Maxim Integrated [1]. Ve zmíněném souboru jsou implementovány funkce pro ovládání a komunikaci se zařízeními na sběrnici, jako jsou například restart zařízení, odeslání a čtení bytů apod. Zajímavé jsou funkce končící na „ToAll“, které slouží k práci se všemi čtyřmi piny, ke kterým jsou připojeny teploměry, najednou. Jednoduché obsluhy těchto pinů bylo dosaženo použitím pinů ze stejného portu.

5.6.2 DS18B20

Každé zařízení na sběrnici má vlastní 64 bitové ID, na jehož základě je možné zařízení adresovat. Pokud je na sběrnici pouze jedno zařízení, je možné speciálním příkazem *SKIP_ROM* (viz [34]) adresování přeskočit, čehož je využito i v případě tohoto projektu.

Funkce pro obsluhu teploměrů DS18B20, implementované v souboru *ds18b20.c*, jsou založeny na těch pro práci s OneWire sběrnicí. Ty nejdůležitější jsou vidět v tabulce 5.7.

Tabulka 5.7: Příklad maker pro práci s jedním pinem ze souboru *pin_control_macros.h*

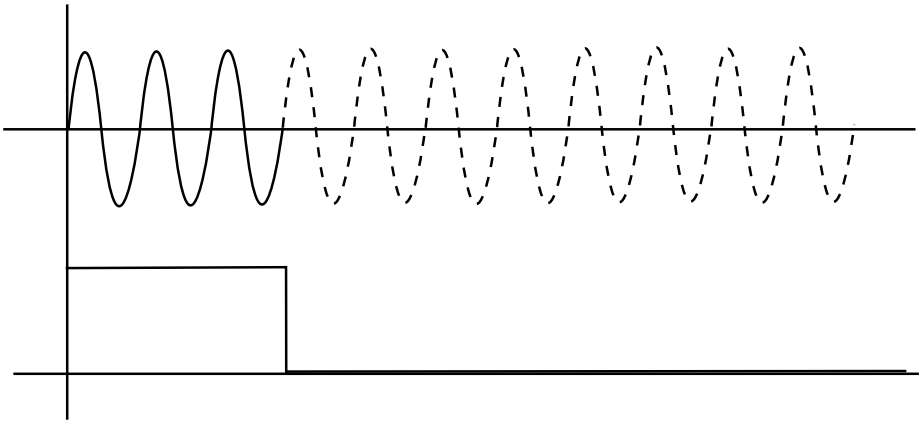
`float DS18B20_measure(`
`  THERMOMETER therm`
`)`
	Změří a vrátí teplotu zvoleného teploměru.

`void DS18B20_measureAllRaw(`
`  uint16_t raw[4]`
`)`
	Vrátí surový výstup všech teploměrů. Této funkce je využito pro přenos dat, kdy není nutné přenášet čtyřbytovou float hodnotu, protože výstup teploměru je maximálně dvanáctibitový.

`void DS18B20_init()`	Inicializace komunikace s teploměry.

5.7 Topení

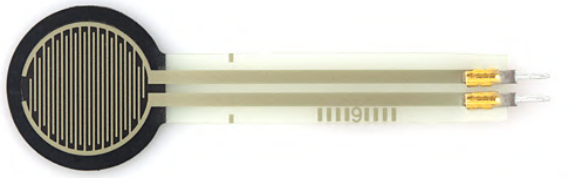
K řízení topení jsou využita dvě polovodičová relé RS1A40D25 (dokumentace viz [35]). Tato relé jsou schopná spínat proudy až do 25 A a jsou vybavena diodou indikující sepnutí. Pro řízení relé je použita PWM s frekvencí 5 Hz. Protože jsou relé vybavena obvody pro spínání v nule, není vyšší frekvence vhodná. Proud v elektrické síti má frekvenci 50 Hz. Při řízení relé pomocí PWM s frekvencí 0.25 Hz (dáno frekvencí časovače 25 Hz a možnosti nastavení PWM ve sto krocích) a spínáním v nule tak můžeme regulovat výsledný výkon v krocích od 0 do 100. Obrázek 5.9 ilustruje spínání v nule pomocí PWM. Hardwarová část obvodu topení je velice jednoduchá – jedná se pouze o vyvedení dvou pinů a zemního potenciálu na dvojici svorkovnic pro připojení vodičů připojených k relé.



Obrázek 5.9: Spínání střídavého proudu pomocí SSR relé a PWM

5.8 Senzor síly na tabletu

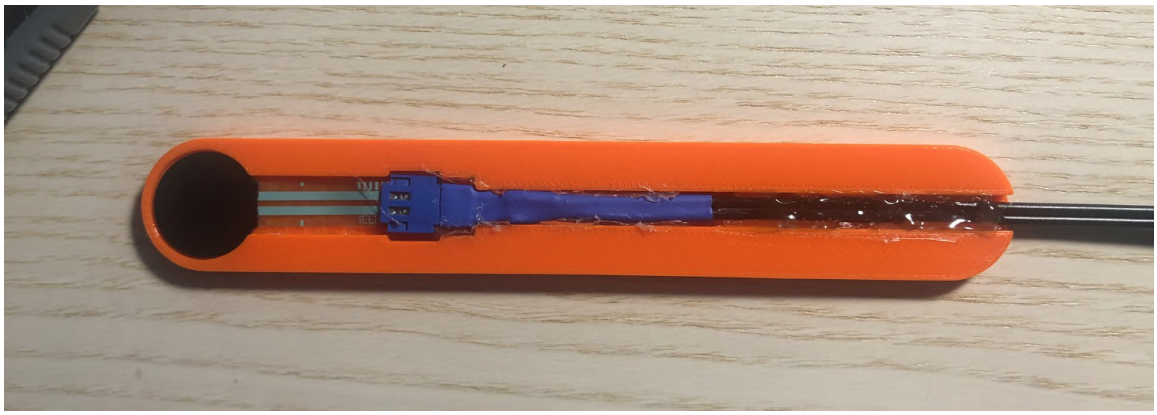
Jedním z požadavků od vědeckého týmu z VFU bylo mít možnost zjistit, jak moc míchací kolébka tlačí na tabletu vloženou do vaku. K tomu účelu byl vyvinut speciální senzor, který pro snímání síly využívá plochý tenzometr FSR402, jehož výrobcem je společnost Interlink Electronics. Ten je vidět na obrázku 5.10.



Obrázek 5.10: Plochý tenzometr SEN-09375. Obrázek převzat z webu SparkFun [7].

Z obrázku 5.10 je patrné, že je senzor tvořen tenkou membránou a dá se tak předpokládat, že při použití bez dalších součástí by neměl dlouhou životnost. Proto bylo navrženo a vymodelováno pouzdro, do kterého by mohl být tenzometr uložen. Toto pouzdro, vytištěné na 3D tiskárně, je včetně osazeného tenzometru vidět na obrázku 5.11.

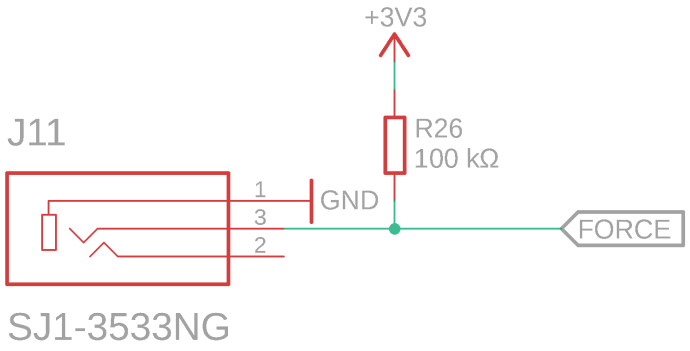
Kvůli zvýšení chemické odolnosti bylo pouzdro zasunuto do prstu ustřiženého z nitrilové rukavice a zataveno tepelně smršťovatelnou fólií. Výsledný senzor je vidět na obrázku 5.12.



Obrázek 5.11: Pouzdro pro uložení tenzometru vytištěné na 3D tiskárně.

Obrázek 5.12: Senzor pro měření síly působící na tabletu

Pro připojení senzoru k modulu byl zvolen 3.5 mm konektor. Díky tomu je možné senzor jednoduše odpojit v případě, že není potřeba. Z dokumentace tenzometru [39] plyne, že odpor tenzometru klesá se vzrůstající silou, která na něj působí a tato závislost odporu na síle je přibližně logaritmická. Pro převod odporu na úroveň napětí je tenzometr zapojen do děliče napětí společně s 100 kΩ resistorem (viz obrázek 5.13).



Obrázek 5.13: Schéma obvodu pro měření síly

Pro výpočet napětí na pinu označeném *FORCE* můžeme využít následující vztah pro dělič napětí:

$$V_{OUT} = V_{CC}\frac{R_T}{R_1 + R_T} \tag{5.1}$$

Kde V_{CC} je napájecí napětí obvodu, R_T odpor tenzometru a R_1 odpor rezistoru v děliči. Protože hodnotu V_{OUT} jsme schopni měřit pomocí analogově-digitálního převodníku, je neznámou v této rovnici hodnota R_T. Vhodnou úpravou tedy získáme rovnici:

$$R_T = \frac{R_1 V_{OUT}}{V_{CC} - V_{OUT}} \tag{5.2}$$

Dále víme, že závislost odporu R_T na síle působící na tenzometr je logaritmická. Dokumentace tenzometru [39] obsahuje i graf této závislosti, ale pro dosažení vyšší přesnosti byla navržena následující kalibrační procedura:

1. Senzor je položen na váhu
2. Na senzor je umístěna tableta
3. O tabletu opřeme plochý předmět (například tenké plexisklo), který ve dvou bodech podepřeme. Třetí opěrný bod tvoří tableta.
4. Na plochu nad tabletou postupně přidáváme závaží, například padesátikorunové mince, či podložky.
5. Po každém přidání závaží zaznamenáme hodnotu na váze a výstup analogově-digitálního převodníku

Na základě změřených bodů provedeme logaritmickou regresi, čímž získáme parametry logaritmické funkce pro výpočet vztahu mezi silou (hmotnost přepočteme vztahem pro výpočet gravitační síly $F_G = mg$) a odporem. Tento vztah bude mít tvar:

$$R_T = A + Bln(F) \tag{5.3}$$

Kde A a B jsou parametry získané logaritmickou regresí a F je síla působící na tenzometr. Z tohoto ztahu jsme již schopni vyvodit vzorec pro výpočet síly

$$F = e^{\frac{R_T - A}{B}} \tag{5.4}$$

kde e je Eulerovo číslo. Výsledky kalibrace reálného senzoru jsou popsané v kapitole 8.3.

Funkce pro měření síly jsou implementované v souboru *force.c*. Kromě funkcí pro inicializaci a čtení aktuální hodnoty je v knihovně dostupná ještě funkce *FORCE_exceeded*, která informuje o tom, jestli byla překročena maximální síla na tabletu. V této funkci je implementována hystereze, aby nedocházelo k rychlému sledu hodnot „překročeno“ a „nepřekročeno“, například z důvodu zákmitu. Vzestupná hrana výstupní hodnoty této funkce značí nové překročení povolené hodnoty, sestupná její opětovné zmenšení pod patřičnou mez. Implementace této funkce je vidět ve výpisu 5.4. Statickou proměnnou ve funkci je možné využít, protože je k modulu vždy připojen maximálně jeden sensor.

```
uint8_t FORCE_exceeded(double forceMax, double force) {
    // 0 - unitialized
    // 1 - normal
    // 2 - overshoot
    static uint8_t state = 0;

    if (state == 0) {
        state = 1;
    } else if (state == 1) {
        if (force > forceMax + FORCE_HYSTERESIS) state = 2;
    } else if (state == 2) {
        if (force < forceMax - FORCE_HYSTERESIS) state = 1;
    }

    return state == 2;
}
```

Výpis 5.4: Detekce překročení povolené síly na tabletu s hysterezí

5.9 Řízení krokových motorů

Pro míchání vaků jsou v zařízení Golem využity čtyři krokové motory (viz kapitola 3.2.3). Dle katalogového listu [29] se jedná o motory s dutou hřídelí a krokem 1.8°. Jmenovitý proud motoru je 0.4 A, indukčnost 32 mH a odpor 30 Ω. Motor je zobrazen na obrázku 5.14.

5.9.1 Koncový spínač

Poloha motoru je na jednom konci rozsahu určena koncovým spínačem, na druhém je určena čítáním počtu otáček motoru (toto řešení tedy zůstává stejné, jako u předchozí verze zařízení – viz kapitola 3.2.3). Použitý koncový spínač je klasický mikrospínač s klopným mechanismem.

Obrázek 5.14: Krokový motor SX16-0402LA-120, použitý v zařízení Golem. Foto pochází z katalogového listu distributora [29].

5.9.2 TMC2209

Pro buzení motorů je použit budič TMC2209 od společnosti Trinamic. Jedná se o budič s celou řadou pokročilých funkcí včetně ultratichého chodu, efektivního provozu, detekce vynechání kroku, mikrokrokováním apod. Budič může pracovat v několika modech. První z nich je mód kompatibility se staršími budiči, kdy je řízen prostřednictvím dvou pinů udávajících směr otáčení a provedení kroku. Ve druhém módu je do budiče možné předem nahrát konfiguraci (například počet mikrokroků na jeden krok, výkon apod.) a poté jej ovládat dvojicí zmíněných pinů. Ve třetím módu je budič řízen přes UART sběrnici. Tento přístup je zvolen i v tomto projektu.

Na jedné UART sběrnici je možné mít připojeny až čtyři budiče. Ty je možné adresovat, přičemž adresa je zvolena dvěma piny *MS1_AD0* a *MS2_AD1*. Protože je v projektu na jednom modulu vždy jeden budič, není tato možnost využita a adresa zařízení je nastavena na 0.

UART je zde zvláštní, protože k přenosu používá pouze jeden vodič. Piny TX a RX mikrokontroléru jsou k sobě připojeny 1 kΩ resistorem a pin RX je připojen na pin *PDN_UART* budiče. Další zvláštností je schopnost budiče automaticky odvodit rychlost komunikace, a to v rozsahu 9.6 až 500 kBaud. To je možné díky protokolu použitému při komunikaci. Ten vždy na začátku datagramu obsahuje čtveřici bitů "1010", ze které je možné rychlost odvodit. Na obrázku 5.15 je zachycena struktura datagramu pro zápis do registru budiče. Po úvodní čtveřici bitů následují čtyři rezervované bity a dále jeden byte určený adrese budiče (ta ale může být jen 0 až 3). Následuje sedm bitů adresy registru a bit značící, zda se jedná o zápis, či čtení (1 = zápis, 0 = čtení). Následující čtveřice bytů obsahuje data pro zápis do registru a celý datagram je zakončen bytem obsahujícím kontrolní součet. Obdobnou strukturu mají i datagramy pro čtení a odpovědi čtení – více viz dokumentace budiče [19].

V souboru *tmc2209.c* jsou implementovány funkce pro komunikaci s budičem. Součástí jsou základní funkce pro čtení a zápis registrů a také inicializaci hodnot registrů. V souborech *tmc2209_types.c* a *.h* jsou dostupné struktury definující sémantiku bitů v konkrétních registrech dle dokumentace [19] a také funkce pro jejich manipulaci. Nad funkcemi z těchto dvou souborů staví knihovna *motor.c*. Jejím prostřednictvím je možné nastavit rychlost a směr pohybu motoru, popřípadě motor zastavit.

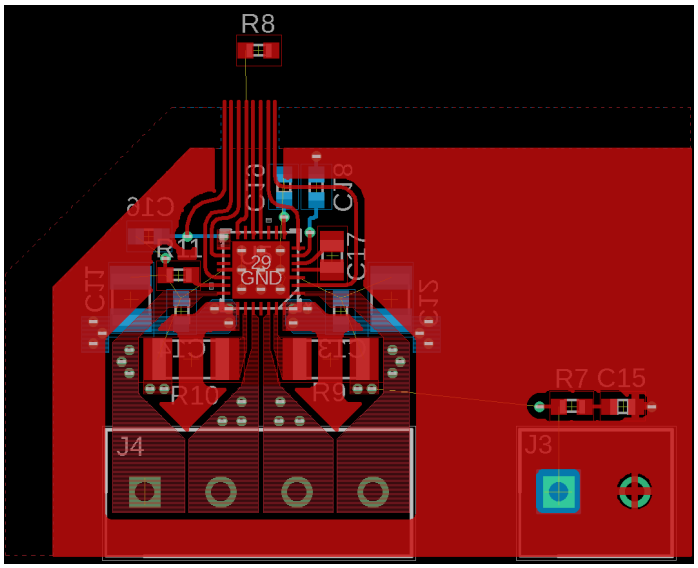
Ovládání probíhá prostřednictvím sběrnice UART. Zpětnou vazbu o provedených krocích získává mikrokontrolér na základě pulsů na pinu *INDEX* budiče, který je generován každé čtyři kroky. Po spuštění modulu se motor začne otáčet tak, aby kolébka směřovala ke koncovému spínači. Po jeho stisknutí je počitadlo kroků vynulováno a motor se začíná periodicky pohybovat v intervalu $\langle 0, MOTOR\_STEPS\_BOUNDARY \rangle$ kroků. Řízení pohybů

UART WRITE ACCESS DATAGRAM STRUCTURE																				
each byte is LSB...MSB, highest byte transmitted first																				
0 ... 63																				
sync + reserved								**8 bit slave address**			**RW + 7 bit register addr.**			**32 bit data**			**CRC**		
0...7								8...15			16...23			24...55			56...63		
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*=0...3			register address		1	data bytes 3, 2, 1, 0 (high to low byte)			CRC		
0	1	2	3	4	5	6	7	8	...	15	16	...	23	24	...	55	56	...	63	

Obrázek 5.15: Datagram pro zápis do registru budiče krokových motorů TMC2209. Obrázek pochází z dokumentace budiče [19].

motoru bere v úvahu i maximální povolenou sílu na tabletu, při jejímž překročení je obrácen směr pohybu motoru.

Deska plošných spojů vychází z ukázkové desky TMC2209-BOB od výrobce Trinamic. Podklady pro její výrobu jsou dostupné na webu výrobce [18]. Původní čtyřvrstvý design byl přepracován do dvou vrstev a některé komponenty byly přesunuty do spodní vrstvy. Součástí desky je i svorkovnice pro přivedení napájecího napětí motorů (24 V). Oproti zbytku desek plošných spojů je zde udělána výjimka a na horní desku je přivedeno 24 V namísto obvyklých 3.3 V. V horní i spodní vrstvě je izolační mezera, která část budiče motoru odděluje od zbytku desky. Návrh desky je vidět na obrázku 5.16.



Obrázek 5.16: Rozložení desky plošných spojů s budičem krokových motorů TMC2209.

5.10 Měření pH

Návrh obvodů pro měření pH vychází z použitých pH elektrod FlaTrode. Jejich vlastnosti jsou dle dokumentace [6]:

- Rozsah měření pH 0 až 14
- Citlivost 57 až 59 mV na jeden dílek pH
- Nulový bod při pH 7 je 0 V ± 20 mV

Princip určování pH byl představen již v kapitole 3.2.4 a je založen na měření rozdílu potenciálu mezi referenční elektrodou, ponořenou do roztoku s pH 7 a měřící elektrodou, ponořenou ve zkoumaném roztoku.

Napětí na měřící elektrodě pro případ, kdy je referenční elektroda připojena k zemnímu potenciálu, je zachyceno v tabulce 5.8.

Tabulka 5.8: Napětí na měřící elektrodě při připojení referenční elektrody k zemnímu potenciálu (uvažováno 59 mV/pH)

pH	Napětí [V]	pH	Napětí [V]	pH	Napětí [V]
0	-0,413			**14**	0,413
1	-0,354			**13**	0,354
2	-0,295			**12**	0,295
3	-0,236	**7**	0	**11**	0,236
4	-0,177			**10**	0,177
5	-0,118			**9**	0,118
6	-0,059			**8**	0,059

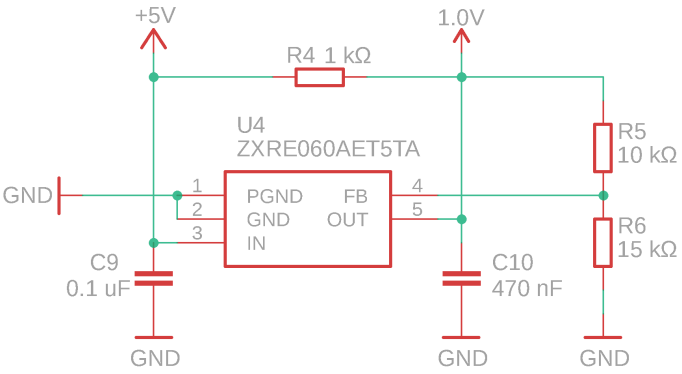
Z hodnot v tabulce 5.8 plyne, že celkový rozsah výstupního napětí pH elektrod je 823 mV. I v případě přidání dostatečné rezervy se tedy celý rozsah vejde do jednoho voltu. Protože je v projektu žádoucí vyhnout se záporným napětím (vyžadovalo by záporné napájení operačních zesilovačů, zpracovávajících signál apod.), můžeme referenční elektrodu připojit na napětí v polovině tohoto rozsahu, tedy 500 mV. Tím zajistíme, že se napětí na měřící elektrodě bude pohybovat v intervalu $\langle 0, 1 \rangle$ V.

5.10.1 Zdroj referenčního napětí 1 V

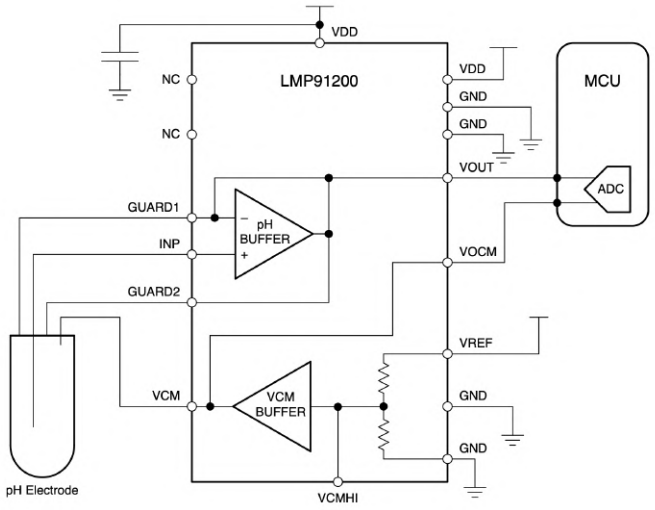
Pro minimalizaci chyb měření je třeba zajistit zdroj stabilní referenční hodnoty napětí. K tomuto účelu byl použit obvod z řady ZXRE060. Ten byl vhodnou volbou pasivních komponent (podle dokumentace [26]) nastaven tak, aby udržoval napětí 1 V. Schéma zapojení je vidět na obrázku 5.17.

5.10.2 Analogový frontend

Jednou z možností zpracování signálu z pH sond bylo vytvoření obdobného obvodu z operačních zesilovačů, jako byl obvod existující v zařízení Golem 2, který byl představen v kapitole 3.2.4. Nakonec byla ale vybrána varianta použití obvodu, který je přímo určen jako analogový frontend pro pH elektrody. Jedná se o čip LMP91200 od Texas Instruments. Jeho typické použití je zachyceno na obrázku 5.18.

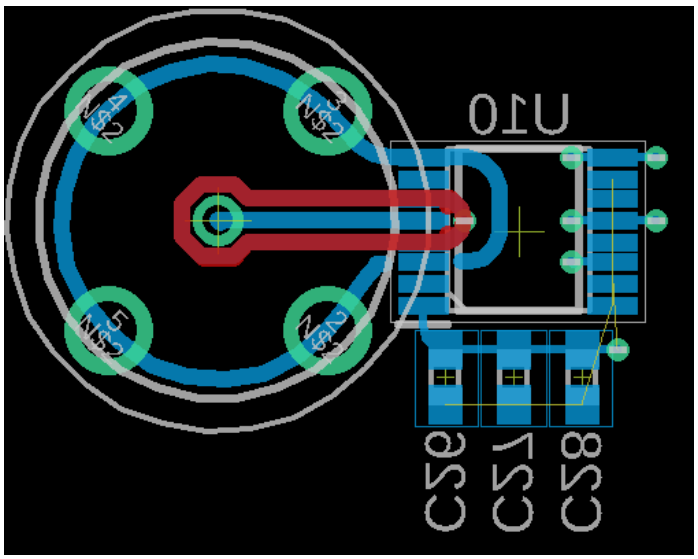


Obrázek 5.17: Zapojení obvodu ZXRE060 jako zdroje referenčního napětí 1 V



Obrázek 5.18: Typické použití analogového frontendu pro měření pH z dokumentace obvodu [27]

Z obrázku 5.18 je vidět, že uvnitř čipu nalezneme dělič napětí, který dělí *VREF* na polovinu. Na tento pin tedy můžeme přivést 1 V, ten bude snížen na polovinu a toto napětí 500 mV přivedeno na referenční elektrodu sondy. Z výše popsaných důvodů bude výstupní napětí přibližně mezi 0 a 1 V. Obvod analogového frontendu a jeho pasivní komponenty jsou umístěny na spodní straně desky. Vedení signálu je provedeno dle doporučení z dokumentace obvodu [27]. Cesta se signálem z měřící elektrody je chráněna smyčkou připojenou na piny označené *GUARD*. Tato smyčka je umístěna na obou stranách desky a slouží k impedančnímu oddělení signálu od zbytku desky. Rozložení komponent a připojení k BNC konektoru je vidět na obrázku 5.19.



Obrázek 5.19: Rozmístění komponent analogového frontendu a připojení k BNC konektoru dle doporučení z dokumentace [27].

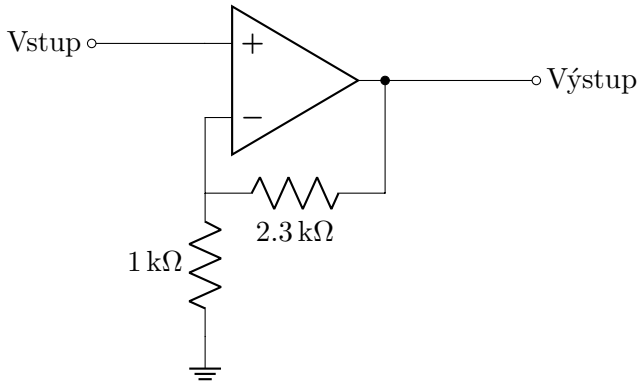
5.10.3 Zesílení signálu z analogového frontendu

Kdybychom výstupní signál analogového frontendu o maximálním napětí pod 1 V přivedli na vstup analogově-digitálního převodníku mikroprocesoru napájeného 3.3 V, využili bychom méně než třetinu rozsahu převodníku. Z toho důvodu je signál před digitálním zpracováním ještě zesílen. K tomu je využit operační zesilovač zapojený jako neinvertující zesilovač se zesílením G = 3.3. Jeho schéma je vidět na obrázku 5.20.

Protože jsou na pH modulu celkem čtyři konektory pro připojení pH sond je i tento zesilovač na desce čtyřikrát. Proto byl pro fyzickou realizaci vybrán obvod čtyřnásobného operačního zesilovače LTC6079. Tabulka 5.9 ukazuje hodnoty napětí pro jednotlivé hodnoty pH.

5.10.4 Softwarové zpracování

Výstup operačního zesilovače v rozsahu cca 0 až 3.3 V je přiveden na analogově-digitální převodník mikrokontroléru. Naměřená hodnota v podobě šestnáctibitového čísla pak musí být převedena na skutečné pH. Pro tyto účely musí být sonda nejdříve zkalibrována. Protože je vztah napětí na pH lineární, používají se ke kalibraci většinou dva až tři body (často pH



Obrázek 5.20: Zapojení operačního zesilovače pro úpravu signálu z analogového frontendu pro měření

Tabulka 5.9: Ideální hodnoty napětí v jednotlivých částech obvodu pro zpracování pH

pH	**Rozdíl referenční a měřící elektrody [V]**	**Výstup analogového frontendu [V]**	**Výstup operačního zesilovače [V]**
0	-0.413	0.087	0.287
1	-0.354	0.146	0.482
2	-0.295	0.205	0.677
3	-0.236	0.264	0.871
4	-0.177	0.323	1.066
5	-0.118	0.382	1.261
6	-0.059	0.441	1.455
7	0.000	0.500	1.650
8	0.059	0.559	1.845
9	0.118	0.618	2.040
10	0.177	0.677	2.234
11	0.236	0.736	2.429
12	0.295	0.795	2.624
13	0.354	0.854	2.818
14	0.413	0.913	3.013

4 a pH 7). Funkce pro obsluhu, měření a kalibraci pH sond jsou dostupné v souboru *ph.c*. Přehled a popis těch nejdůležitějších je v tabulce 5.10.

Tabulka 5.10: Často používané funkce ze souboru *ph.c*

`int8_t PH_measureCalibrationPoint(`
`  uint8_t sensor,`
`  float ph`
`)`
	Uloží aktuální hodnotu na analogově-digitálním převodníku jako bod se zadaným pH. Pokud byly pro senzor takto zadané alespoň dva body, provede se přepočet směrnice a posunu lineární funkce závislosti pH na měřené hodnotě.

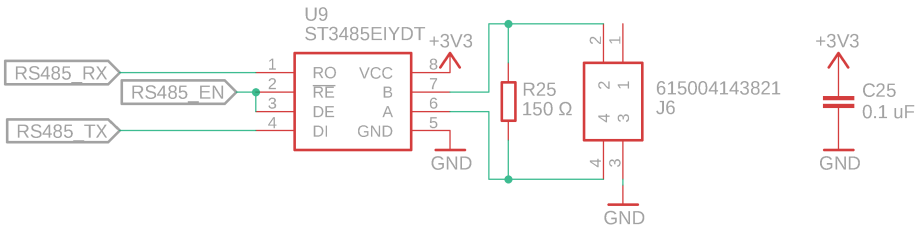
`float PH_read(`
`  uint8_t sensor`
`)`
	Vrátí hodnotu pH na zvoleném senzoru na základě kalibrace.

5.11 Pumpy a ventily Cavro

Komunikační protokol pump a ventilů Cavro byl představen již v kapitole 3.3. Tato zařízení komunikují prostřednictvím sběrnice RS485. Pro generování datagramů na této sběrnici je využita UART periferie mikrokontroléru.

5.11.1 RS485

Zařízení Cavro komunikují po sběrnici RS485 poloduplexním způsobem a ke komunikaci tedy stačí dvojice vodičů. K převodu TTL UART na RS485 byl použit obvod ST3485EIY, což je převodník RS485 sběrnice, pracující i s 3.3 V mikrokontroléry. Použité zapojení převodníku je vidět na obrázku 5.21.



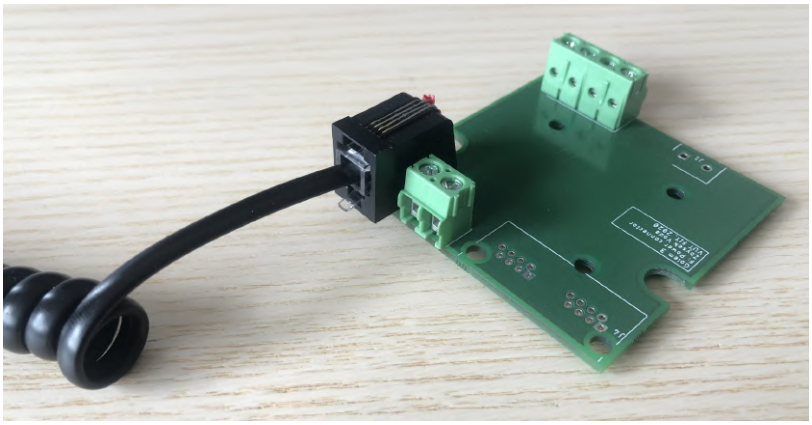
Obrázek 5.21: Zapojení převodníku sběrnice RS485

Převodník obsahuje piny *RE* a *DE* pro povolení čtení a zápisu na sběrnici, přičemž první z nich je negován. Propojením těchto vodičů lze jedním pinem mikrokontroléru řídit, jestli dochází ke čtení, nebo zápisu na sběrnici. Funkce pro obsluhu, implementované v souboru *rs485.c*, obalují funkce z UART knihovny a přidávají k nim nastavování pinu pro čtení či zápis.

5.11.2 Připojení a napájení

Pro připojení pump k modulu je použit čtyřpinový konektor RJ10. Využity jsou pouze tři vodiče – dva pro *A* a *B* piny signálu sběrnice RS485 a jeden vodič pro zemní potenciál. Napájecí napětí pump 24 V není přivedeno přímo na modul, ale mezi modulem a pumpami

je ještě vložena mezivrstva v podobě desky plošných spojů s konektorem RJ10 a svorkovnicí pro přivedení napájecího napětí na jedné straně a svorkovnicí pro připojení pump a ventilů na straně druhé. Tato deska plošných spojů je vidět na obrázku 5.22.



Obrázek 5.22: Deska pro přivedení napájení a RS485 sběrnice k pumpám a ventilům Cavro

5.11.3 Software pro řízení

Software pro řízení pump a ventilů Cavro je implementován v souboru *tecan.c*. Obsahuje funkce pro generování příkazů na inicializaci pump apod., které byly představeny v kapitole 3.3.

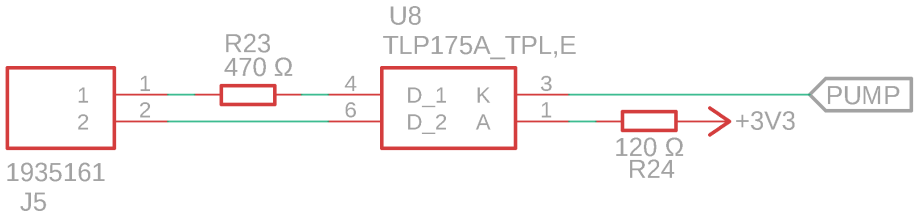
Aby bylo možné jednoduše pracovat s více zařízeními na sběrnici, byla vytvořena struktura pro uchovávání informací o zařízení, obsahující jeho adresu a další. Tato struktura je vidět ve výpisu 5.5.

```
typedef struct {
    uint8_t address;
    T_TECAN_DEVICE_TYPE type;

    // pump
    T_TECAN_PUMP_INPUT input;
    uint8_t speed;
    int16_t currentReserve;
} T_TECAN_DEVICE;
```

Výpis 5.5: Struktura pro uchování informací o zařízeních Cavro

5.12 Peristaltické pumpy

Peristaltické pumpy slouží k přečerpávání objemů mezi vaky. Obrázek peristaltické pumpy je k vidění v kapitole 3.2.6. Jak už bylo ve zmíněné kapitole popsáno, probíhá řízení spínáním dvojice kontaktů na konektoru na zadní straně pump. Ty jsou k modulu připojeny pomocí svorkovnice a k jejich spínání je použit OPTOMOSFET TLP175A, jehož zapojení je vidět na obrázku 5.23. Ten je ovládán jedním digitálním výstupem.



Obrázek 5.23: Zapojení OPTOMOSFETu pro spínání peristaltických pump

Protože je řízení dvoustavové (tedy „*čerpá*“/„*nečerpá*“), je možné přečerpaný objem v programu ovlivnit pouze časem čerpání. Určení času potřebného k přečerpání jednoho mililitru se věnuje kapitola 8.4.

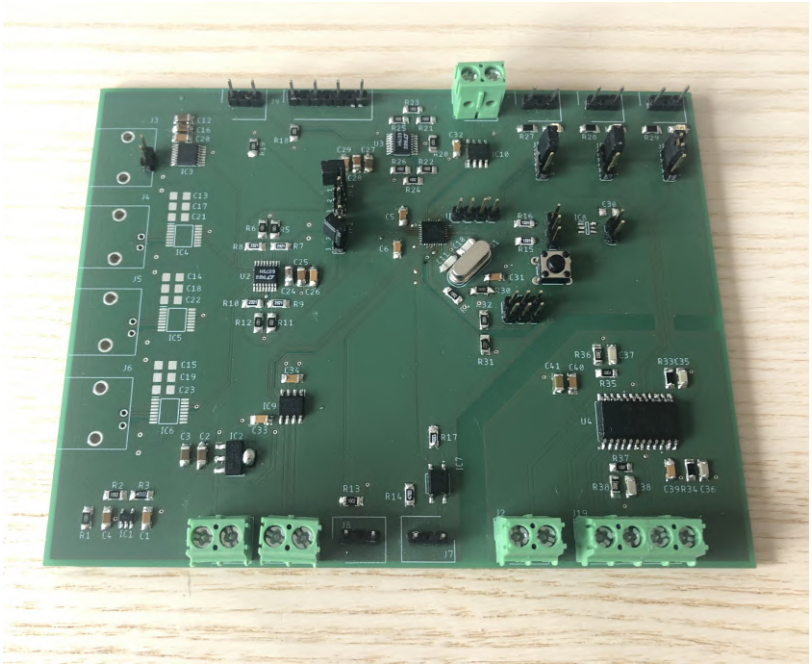
Kapitola 6

Realizace systémových modulů

V této části se již dostáváme k návrhu modulů, které reflektují rozdělení systému popsané v kapitole 4.5 a jsou složeny z funkčních bloků, představených v kapitole 5. Moduly komunikují na základě protokolu popsaného v kapitole 4.4. Příkazy, které je možné k řízení daného modulu použít, jsou uvedeny vždy v kapitole, věnující se danému modulu. V příloze C je navíc uveden postup přidání nového modulu do systému.

6.1 Prototyp

Před návrhem jednotlivých modulů byla vytvořena deska obsahující všechny bloky, aby mohly být nejdříve oživeny a otestovány. Na jejím základě byly vytvořeny knihovny pro obsluhu jednotlivých bloků a také základ firmwaru pro ovládání modulů. Pro případy, kdy jeden pin ovládá na různých modulech různé bloky je deska vybavena piny s propojkami, aby mohl být pin přiveden na patřičný blok. Prototyp je vidět na obrázku 6.1.



Obrázek 6.1: Prototyp obsahující většinu funkčních bloků modulů

Po testech s tímto prototypem byly některé komponenty vyměněny, například původní budič krokových motorů L6219 byl nahrazen již zmíněným budičem TMC2209 – viz kapitola 5.9. Také bylo určeno finální rozdělení pinů, které obsahuje příloha A.

6.2 Společný základ modulů

Po hardwarové stránce byl společný základ představen již v kapitole 5.1, ve které byly také uvedeny funkce pro obsluhu CAN periferie. V této části bude představena architektura firmware modulů a také společné CAN příkazy pro jejich řízení.

6.2.1 Architektura firmware

Všechny moduly musejí exportovat dvě funkce - **_init* a **_step*, kde *** je název modulu. Ty slouží k inicializaci modulu a provedení jednoho kroku výpočtu. V souboru *golem_config.h* je možné definovat makra určující typ a instanci modulu. Tato část je zachycena ve výpisu 6.1.

```
// MODULE TYPES
#define GOLEM_GATE 0
#define PH_CONTROL 1
#define BAG_CONTROL 2
#define TEMPERATURE_CONTROL 3
#define ENZYME_CONTROL 4

// TYPE OF MODULE
#define MODULE TEMPERATURE_CONTROL
#define MODULE_INSTANCE 0
```

Výpis 6.1: Volby instance a typu modulu na základě maker *MODULE* a *MODULE_INSTANCE*

Na základě volby *MODULE* a *MODULE_INSTANCE* dochází k překladu firmware pro konkrétní modul a je díky nim možné podmínit i které soubory budou přeloženy a které ne, což urychluje překlad. V souboru *golem.h* také dochází k vložení a volbě funkcí **_step* a **_init*. Část tohoto souboru je vidět ve výpisu 6.2.

```
...
#elif MODULE == PH_CONTROL ////////////////////////////////////////////
#include "modules/ph_control.h"
#define MODULE_init PH_CONTROL_init
#define MODULE_step PH_CONTROL_step
#elif MODULE == BAG_CONTROL ///////////////////////////////////////////
#include "modules/bag.h"
#define MODULE_init BAG_init
#define MODULE_step BAG_step
...
```

Výpis 6.2: Import funkcí **_init* a **_step* dle volby maker výše

Činnost modulů je založena na běhu nekonečné smyčky, která opakovaně volá funkci **_step*. Funkce *main*, která je společná pro všechny moduly, je vidět ve výpisu 6.3.

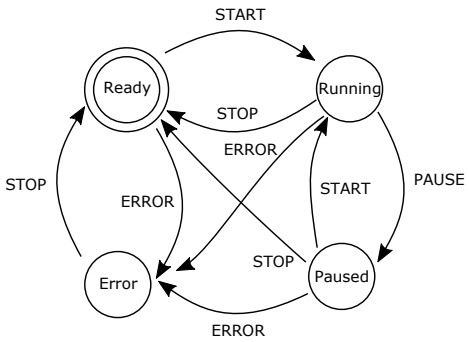
```
int main(void) {
        BOARD_InitBootClocks();

    MODULE_init();

    while (1) {
        if(CAN_readNext(&received)) {
            MODULE_step(&received);
        }
        else {
            MODULE_step(NULL);
        }
    }

    return 0;
}
```

Výpis 6.3: Funkce *main* společná pro všechny moduly

V každé iteraci je zjištěno, jestli buffer příchozích příkazů obsahuje nějaký rámec. Pokud ano, je tento rámec předán funkci **_step*, která jej dále zpracovává.

6.2.2 Stavy modulu

Moduly se mohou vyskytovat ve čtyřech různých stavech, které jsou vidět na obrázku 6.2. Do stavu *READY* se modul dostává automaticky po jeho inicializaci. Pokud v jakémkoliv stavu nastane chyba (např. inicializace) přechází modul do stavu *ERROR*, ze kterého je možné přejít do stavu *READY* provedením příkazu *STOP*, při kterém dochází k zastavení činnosti modulu a jeho reinicializaci. Příkazem *START* dochází k zahájení činnosti modulu (například pohybu motorů apod.). Příkaz *PAUSE* tuto činnost pozastavuje, ale na rozdíl od *STOP* nedochází k reinicializaci.



Obrázek 6.2: Stavy modulu a přechody mezi nimi

6.2.3 Společné CAN příkazy

Ze stavů a přechodů mezi nimi vyplývá, že všechny moduly musejí reagovat na trojicí základních příkazů. K nim je ještě přidán příkaz na zjištění stavu modulu – viz tabulka 6.1.

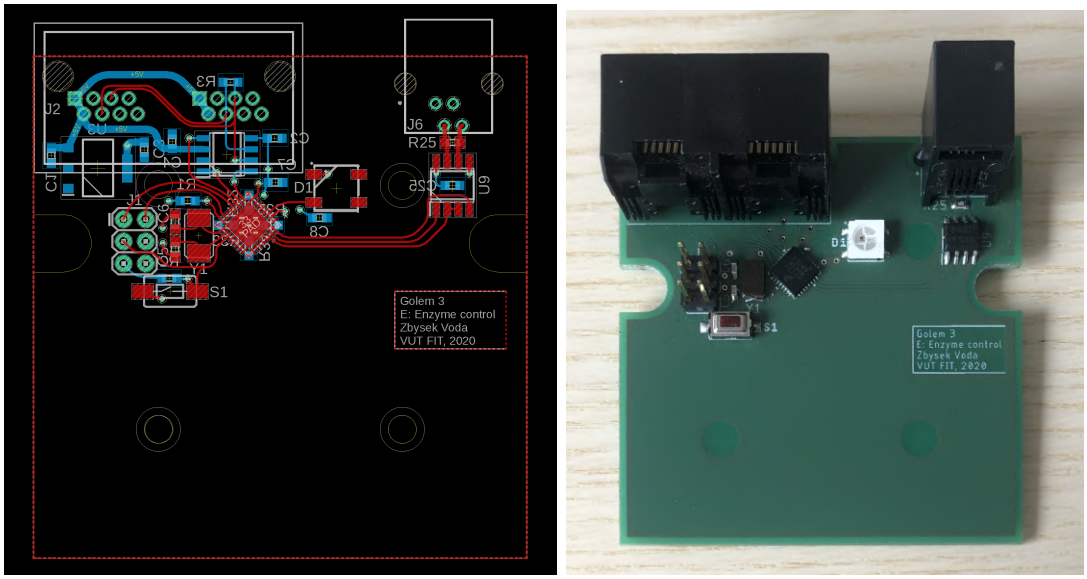
Tabulka 6.1: Společné CAN příkazy, které musejí přijímat všechny moduly

Příkaz	**Popis**	**Data master → slave**	**Data slave → master**
STOP	Zastaví činnost modulu a provede jeho reinicializaci, tedy obnovení jeho stavu a případnou inicializaci jeho periferií.	-	-
START	Zahájení činnosti modulu. Slouží k přechodu ze stavů *READY* a *PAUSED* do stavu *RUNNING*	-	-
PAUSE	Pozastaví činnost modulu bez reinicializace	-	-
GET_STATE	Vrátí aktuální stav modulu	-	uint8_t stav

Všechny příkazy dostupné v systému jsou vypsané v rámci výčtového typu *T_CMD* v souboru *common_commands.h*. Každý z modulů může na tyto příkazy reagovat mírně odlišně, přechody mezi stavy jsou ale pro všechny moduly stejné a jsou implementovány v souboru *common_commands.c*.

6.3 Vstřikování enzymu

Modul vstřikování enzymu je nejjednodušší ze všech modulů a skládá se pouze ze společného základu a bloku pro komunikaci po sběrnici RS485 (pro řízení Cavro zařízení – viz 5.11). K modulu je připojena pouze jedna pístová pumpa Cavro. Deska plošných spojů tohoto modulu a její fyzické provedení je vidět na obrázku 6.3.



Obrázek 6.3: Deska plošných spojů modulu pro řízení vstřikování enzymu a její fyzické provedení

Kromě základních CAN příkazů podporuje deska pouze dva další – viz tabulka 6.2.

Tabulka 6.2: Příkazy pro ovládání modulu na vstřikování enzymu

Příkaz	**Popis**	**Data master → slave**	**Data slave → master**
INJECT	Vstříknutí zadaného objemu enzymu	uint16_t units	-
GET_DEVICE _STATE	Vrátí informaci o stavu Cavro zařízení (1 = komunikuje, 0 = nekomunikuje)	-	uint8_t state

Řízení modulu je implementováno ve funkci *enzyme.c* a obsahuje mimo jiné i obsluhu příchozích příkazů. Protože je tento přístup obdobný u všech modulů, uvedeme si i způsob provedení reakce na příkaz. Ve funkci *ENZYME_step* dochází ke kontrole, jestli byl doručen nějaký CAN rámec. Pokud ano, je nalezena vhodná reakce na příkaz prostřednictvím pole funkcí indexovaného číslem příkazu – viz výpis 6.4.

```
T_CMD_REACTION _ENZYME_cmdReactions[] = {
    [CMD_STOP_AND_REINIT_MODULE] = _ENZYME_CONTROL_processStopAndReinitModule,
    [CMD_START_MODULE] = _ENZYME_CONTROL_processStartModule,
    [CMD_PAUSE_MODULE] = _ENZYME_CONTROL_processPauseModule,
    [CMD_GET_MODULE_STATE] = _ENZYME_CONTROL_processGetModuleState,
    [CMD_ENZYME_INJECT] = _ENZYME_CONTROL_processEnzymeInject,
    [CMD_ENZYME_GET_DEVICE_STATE] = _ENZYME_CONTROL_processGetDeviceState
};
```

Výpis 6.4: Pole s reakcemi na obdržený příkaz

Reakce na příkaz jsou funkce s parametrem pro předání obdrženého CAN rámce. Z jeho dat potom funkce může vyčíst potřebné parametry. Ve výpisu 6.5 je uveden příklad funkce pro zpracování příkazu vstříknutí enzymu.

```
void _ENZYME_CONTROL_processEnzymeInject(T_CAN_PACKET *recv) {
    uint16_t units = *(uint16_t *) (recv->data);

    _ENZYME_state.toPump += units;

    COMMON_COMMANDS_ackCommand(recv);
}
```

Výpis 6.5: Reakce na příkaz ke vstříknutí enzymu

Moduly ve funkci **_step* kromě přijímání příkazů mohou provádět ještě další činnost. V případě tohoto modulu dochází ke kontrole, jestli byl vstříknut požadovaný objem enzymu. Maximální objem vstříknutelný najednou je totiž 1 ml a větší objemy je tak nutné dávkovat postupně. Implementace funkce *ENZYME_step* je ve výpisu 6.6.

```
void ENZYME_step(T_CAN_PACKET *canPacket) {
    if (canPacket != NULL) {
        T_CMD_REACTION reaction = _ENZYME_cmdReactions[canPacket->command];
        if (reaction != NULL) reaction(canPacket);
    }
```

```
    if (_ENZYME_state.runState != RUNNING) return;

    if (_ENZYME_state.toPump > 0) {
        if (((uint64_t) TIMER1_millis - _ENZYME_state.lastInjectionRetry)
            >= ENZYME_INJECTION_RETRY_PERIOD) {
            _ENZYME_state.lastInjectionRetry = TIMER1_millis;

            int16_t result = TECAN_getStatus(&_ENZYME_pump, 500);

            delayMs(50);

            if (result != -1 && TECAN_GET_READY_BIT(result) == 1) {
                uint16_t units = boundUInt16(
                    _ENZYME_state.toPump,
                    0,
                    TECAN_PUMP_ACTIVE_VOLUME
                );
                _ENZYME_state.toPump -= units;

                TECAN_PUMP_injectUnits(&_ENZYME_pump, units);
            }
        }
    }
}
```

Výpis 6.6: Hlavní funkce modulu pro vstřikování enzymu

6.4 Regulace teploty

Modul regulace teploty obsahuje čtyři 3.5 mm konektory pro připojení teploměrů (viz kapitola 5.6) a také dvě svorkovnice pro připojení vodičů ovládajících SSR relé (viz kapitola 5.7). Návrh desky plošných spojů a její fyzická realizace je na obrázku 6.4.

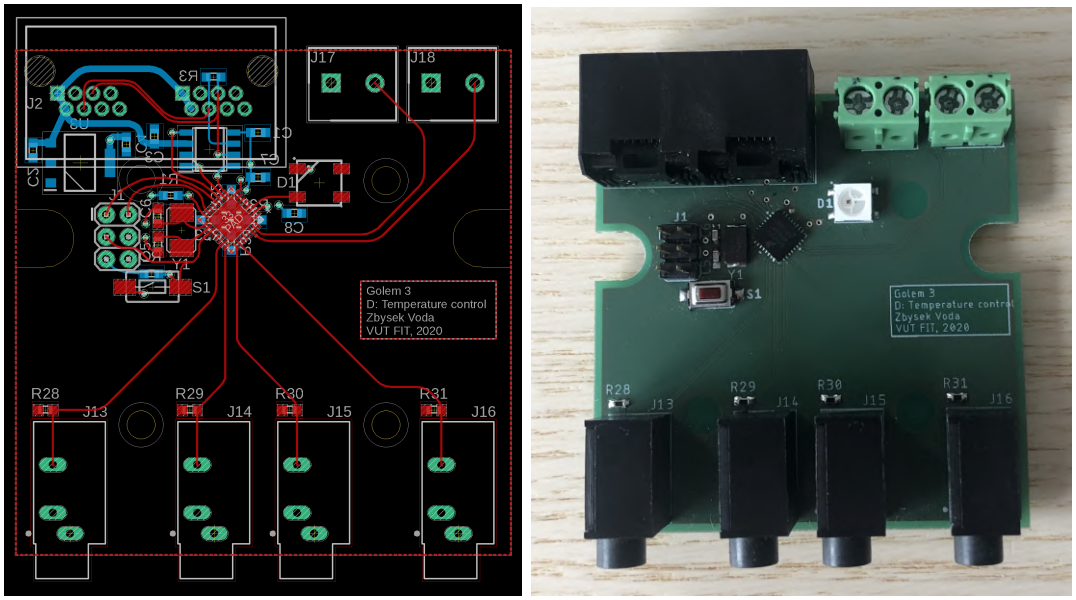
Firmware modulu je implementován v souboru *temperature_control.c*. Příkazy pro řízení modulu jsou zdokumentovány v tabulce 6.3.

Tabulka 6.3: Příkazy pro ovládání modulu na regulaci teploty

Příkaz	**Popis**	**Data master → slave**	**Data slave → master**
SET_TEMP _WEIGHTS	Nastavení vah jednotlivých teploměrů při výpočtu výsledné teploty.	uint16_t weights[4]	-
GET_TEMP _WEIGHTS	Získání vah teploměrů.	-	uint16_t weights[4]
SET_PID _CONSTANT _KP	Nastavení konstanty proporcionální složky PID regulátoru.	float kP	-

SET_PID _CONSTANT _KI	Nastavení konstanty integrační složky PID regulátoru.	float kI	-
SET_PID _CONSTANT _KD	Nastavení konstanty derivační složky PID regulátoru.	float kD	-
GET_PID _CONSTANT _KP	Získání konstanty proporcionální složky PID regulátoru	-	float kP
GET_PID _CONSTANT _KI	Získání konstanty integrační složky PID regulátoru	-	float kI
GET_PID _CONSTANT _KD	Získání konstanty derivační složky PID regulátoru	-	float kD
SET_TARGET _TEMP	Nastavení cílové teploty topení.	float temp	-
GET_TARGET _TEMP	Získání cílové teploty	-	float temp
GET_TEMPE-RATURES	Získání výstupu teplotních čidel bez převodu na teplotu	-	uint16_t temps[4]
GET_ADDRESS _ON_PORT	Přečtení adresy teploměru na zvoleném portu	uint8_t port	uint8_t addr[6]
SET_THERM _OFFSET	Nastaví posun teploty teploměru s nastavenou adresou. Slouží ke kalibraci. Vrací index záznamu o posunu.	uint8_t addr[6] int16_t offset	uint8_t index
GET_THERM _OFFSET	Přečte posun teploty teploměru s danou adresou.	uint8_t addr[6]	int16_t offset uint8_t index
GET_THERM _OFFSET _BY_INDEX	Vrátí posun a adresu teploměru ze záznamu o posunu teploty se zadaným indexem.	uint8_t index	uint8_t addr[6] int16_t offset
GET_THERM _OFFSETS _COUNT	Vrátí počet záznamů o posunu teploty.	-	uint8_t count
SET_TOTAL _OFFSET	Nastaví posuv váhového průměru celkové teploty. Určeno pro kalibraci.	int16_t offset	-
GET_TOTAL _OFFSET	Vrátí posuv váhového průměru	-	int16_t offset

Po startu modulu dochází k periodickému měření teplot ze čtyř teploměrů a výpočtu váženého průměru z těchto hodnot. Tento průměr je vstupem PID regulátoru (viz kapitola 5.4), jehož výstup je použit k nastavení výkonu topení (viz kapitola 5.7).



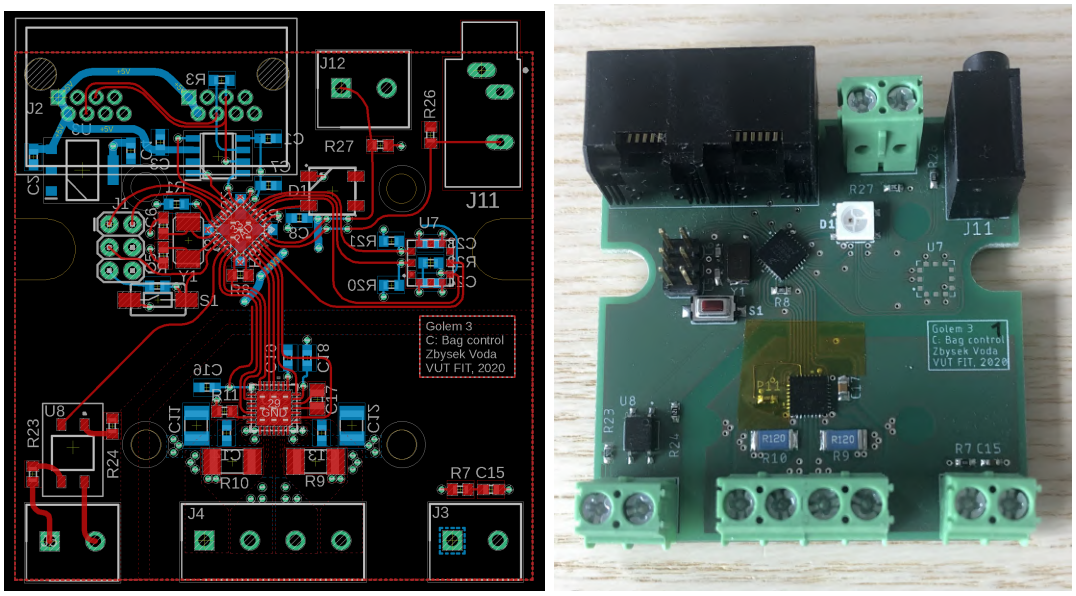
Obrázek 6.4: Deska plošných spojů modulu pro regulaci teploty a její fyzické provedení

6.5 Řízení míchání a přečerpávání vaků

Tyto moduly nalezneme v systému čtyřikrát. Jejich součástí jsou bloky buzení krokového motoru (viz kapitola 5.9), spínání peristaltických pump (viz kapitola 5.12) a také obvody pro měření síly na tabletu a připojení koncového spínače (viz kapitoly 5.8, resp. 5.9.1). Návrh desky plošných spojů a její realizaci zachycuje obrázek 6.5.

Kromě základních příkazů modul implementuje příkazy pro nastavení rychlosti míchání, měření síly na tabletu a další. Tyto příkazy jsou popsané v tabulce 6.4. Firmware pro obsluhu modulu je implementován v souboru *bag.c*.

Pokud je modul ve stavu *RUNNING* a je nastavena nenulová rychlost míchání, stará se modul o periodické pohyby míchací kolébky.



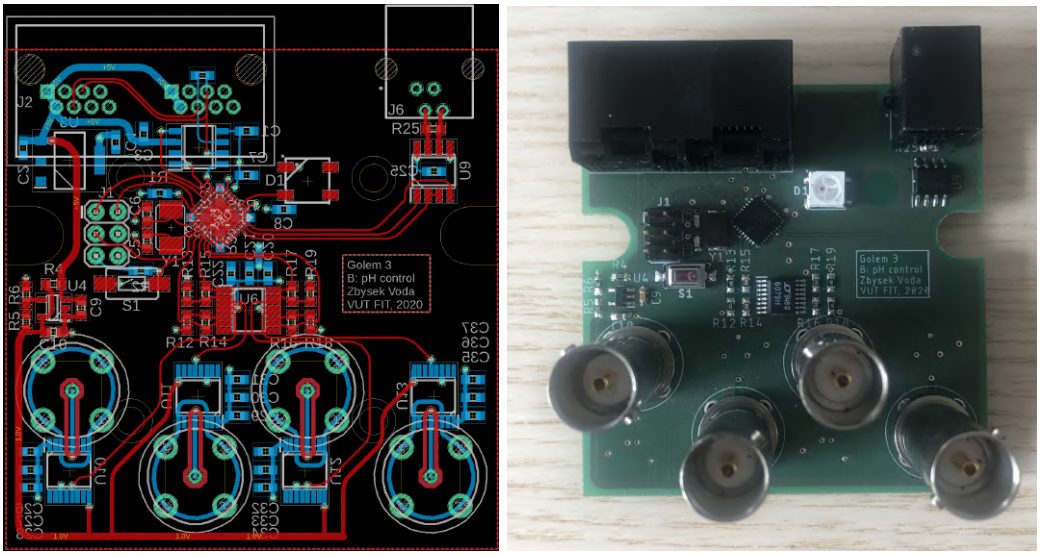
Obrázek 6.5: Deska plošných spojů modulu pro řízení míchání a přečerpávání a její fyzické provedení

Tabulka 6.4: Příkazy pro řízení modulů míchání

Příkaz	**Popis**	**Data master → slave**	**Data slave → master**
SET_MIX_SPEED	Nastavení rychlosti míchání na stupnici 0 až 10000	uint16_t speed	-
GET_MIX_SPEED	Zjištění rychlosti míchání	-	uint16_t speed
SET_MAX_FORCE	Nastavení maximální síly, která může působit na tabletu	double force	
GET_MAX_FORCE	Zjištění aktuálního nastavení maximální síly působící na tabletu	-	double force
SET_MILLILITERS_TO_PUMP	Povel k přečerpání zadaného objemu	double ml	-
GET_CURRENT_FORCE	Zjištění aktuální síly působící na tabletu	-	double f

6.6 Regulace pH ve vacích

Na tomto modulu nalezneme blok pro připojení konektoru sběrnice RS485 (viz kapitola 5.11), zdroj referenčního napětí 1 V a také čtveřici BNC konektorů s analogovými obvody pro zpracování signálu z pH sond (viz kapitola 5.10). Návrh desky modulu a její realizace je vidět na obrázku 6.6.



Obrázek 6.6: Deska plošných spojů modulu pro regulaci pH a její fyzické provedení

Řízení modulu je implementováno v souboru *ph_control.c*. Dostupné příkazy jsou v tabulce 6.5. Ve funkci *PH_CONTROL_step* dochází pravidelně k měření pH ve vacích. Naměřené hodnoty jsou použité jako vstup PID regulátoru (viz kapitola 5.4). Znaménko výstupu regulátoru slouží k určení toho, jestli se má vstřikovat kyselina, nebo zásada. Velikost výstupu určuje objem zvolené kapaliny.

Tabulka 6.5: Příkazy pro ovládání modulu regulace pH

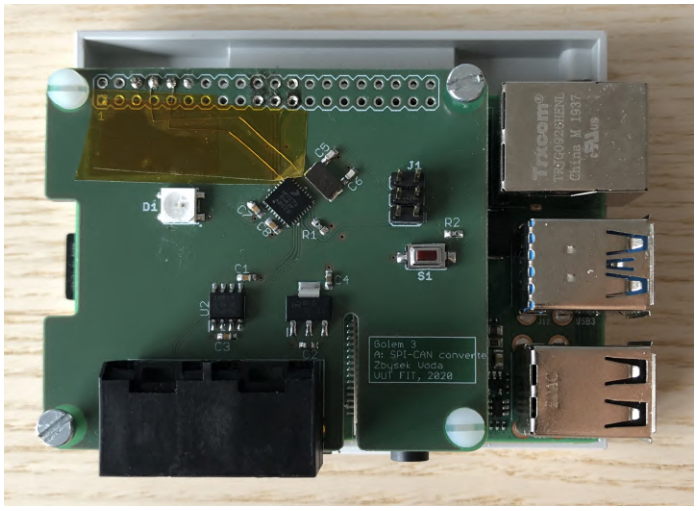
Příkaz	**Popis**	**Data master → slave**	**Data slave → master**
GET_DEVICES _STATE	Vrátí stav Cavro zařízení (1 = komunikuje, 0 = nekomunikuje)	-	uint8_t states[4]
SET_ACTIVE _BAGS	Nastavení aktivních vaků (v neaktivních regulace neprobíhá)	uint8_t active[4]	-
GET_ACTIVE _BAGS	Získání nastavení aktivních vaků.	-	uint8_t active[4]
GET_RAW _PHS	Vrátí výstup analogově-digitálního převodníku, který čte hodnotu pH sond	-	uint16_t rawPh[4]

GET_PH	Získání měřené hodnoty pH. Před použitím musí být zvolená sonda zkalibrována.	uint8_t bag	float ph
ADD_ACID	Ruční vstříknutí zadaného počtu jednotek kyseliny do vaku	uint16_t units uint8_t bag	-
ADD_ALKALI	Ruční vstříknutí zadaného počtu jednotek zásady do vaku	uint16_t units uint8_t bag	-
SET_PID_CONSTANT_KP	Nastavení konstanty proporcionální části PID regulátoru daného vaku	float kP uint8_t bag	-
SET_PID_CONSTANT_KI	Nastavení konstanty integrační části PID regulátoru daného vaku	float kI uint8_t bag	-
SET_PID_CONSTANT_KD	Nastavení konstanty derivační části PID regulátoru daného vaku	float kD uint8_t bag	-
GET_PID_CONSTANT_KP	Získání konstanty proporcionální části PID regulátoru daného vaku	uint8_t bag	float kP
GET_PID_CONSTANT_KI	Získání konstanty integrační části PID regulátoru daného vaku	uint8_t bag	float kI
GET_PID_CONSTANT_KD	Získání konstanty derivační části PID regulátoru daného vaku	uint8_t bag	float kD
MEASURE_CALIBRATION_POINT	Změření kalibračního bodu	float ph uint8_t bag	uint16_t raw uint8_t pointIndex
SET_CALIBRATION_POINT	Nastavení kalibračního bodu	float ph uint16_t raw uint8_t bag	uint8_t pointIndex
GET_CALIBRATION_POINT	Získání kalibračního bodu	uint8_t bag uint8_t pointIndex	float ph uint16_t raw
GET_TOTAL_CALIBRATION_POINTS	Vrácení počtu kalibračních bodů pro všechny senzory	-	uint8_t totalPoints[4]
RESET_CALIBRATION	Reset kalibrace vybraného senzoru	uint8_t bag	-
SET_TARGET_PH	Nastavení cílového pH ve vaku	float ph uint8_t bag	-

GET_TARGET _PH	Zjištění cílového pH vaku	uint8_t bag	float ph

6.7 Master zařízení, převodník UART-CAN

Protože je pro řízení použit počítač Raspberry Pi (viz kapitola 4.6), který periferii CAN neobsahuje, byl navržen i převodník sběrnice UART na CAN. Tento převodník je založený na stejném základu, jako všechny ostatní moduly, ale liší se tvarem desky. Ta byla totiž navržena jako tzv. Raspberry Pi HAT, tedy deska sloužící k rozšíření schopností Raspberry Pi, kterou je možné k tomuto počítači připojit prostřednictvím GPIO hlavice na něm dostupné. Podklady pro tvar této desky jsou převzaty z webu [3]. Na obrázku 6.7 je vidět převodník připojený na počítač Raspberry Pi. Původním záměrem bylo kvůli rychlosti použít sběrnici SPI. To se ale neukázalo jako vhodné řešení kvůli pomalé synchronizaci dat na převodníku. Proto byl modul dodatečně upraven pro využití UART sběrnice.



Obrázek 6.7: Převodník mezi sběrnicemi UART a CAN ve tvaru Raspberry Pi HAT pro připojení k počítači Raspberry Pi 4

Po sběrnici UART komunikuje převodník s Raspberry Pi na základě jednoduchého protokolu, který vychází z protokolu využitého pro komunikaci mezi moduly. Na začátku datagramu je vždy číslo příkazu, po kterém následuje jeden byte udávající adresu cílového zařízení (typ + instance) a dále byte se sekvenčním číslem zprávy. Po nich mohou a následovat byty obsahující data příkazu. Protože převodník zná délky příkazů (délka dat je stejná u CAN i UART příkazů), čeká vždy na doručení požadovaného počtu bytů a poté vyčte celý datagram najednou. Takto obdržený příkaz je uložen do fronty příkazů k odeslání a poté odeslán patřičnému modulu po CAN sběrnici. Po přijetí potvrzení provedení příkazu ze strany CAN sběrnice je toto potvrzení dále odesláno přes sběrnici UART i do Raspberry Pi. Zmíněné chování je implementováno v souboru *uart_commands.c*.

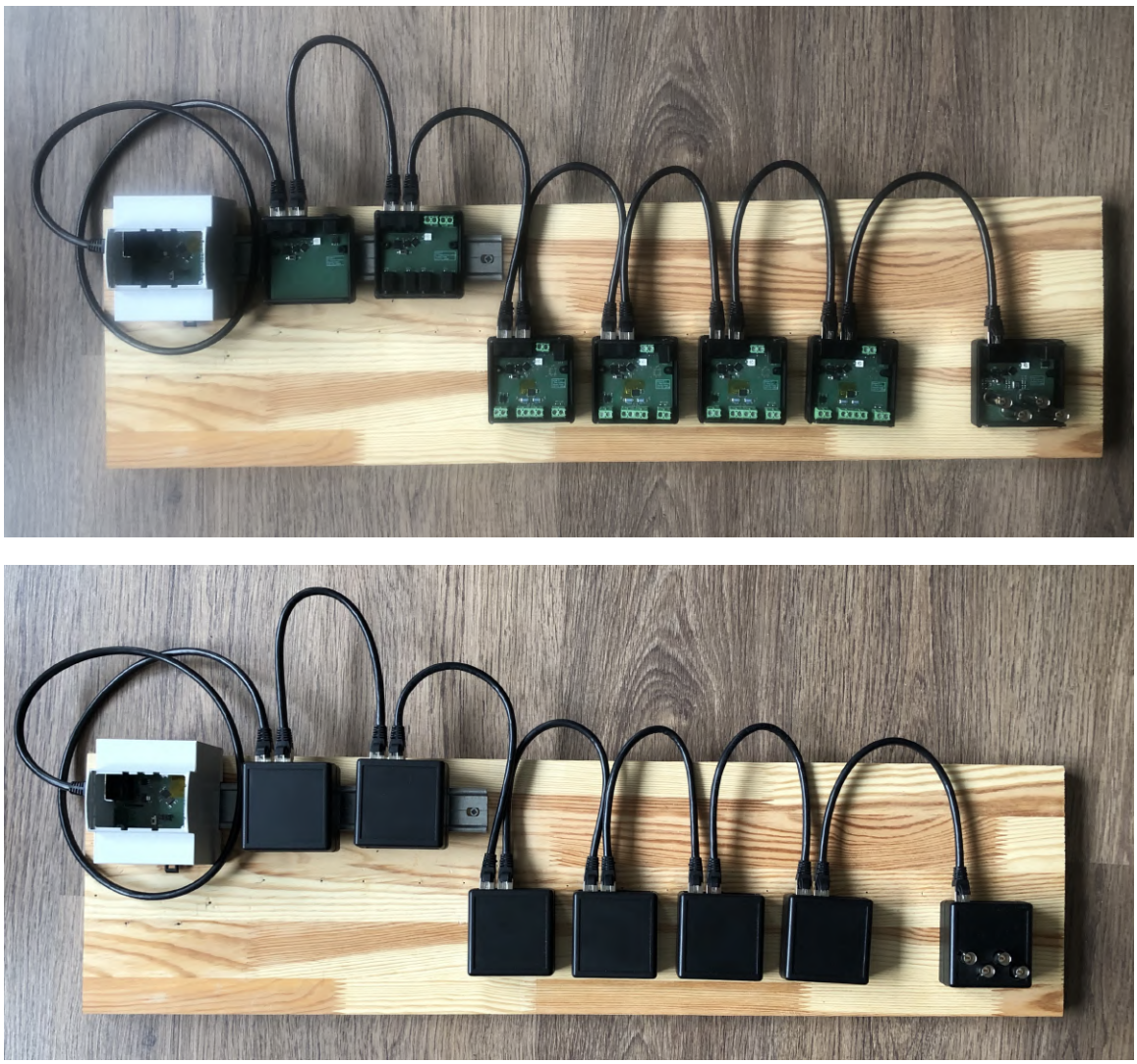
6.8 Umístění a propojení modulů

Moduly jsou propojeny kabelem RJ45 (viz kapitola 5.1.3) a jsou uloženy v plastových krabičkách (viz kapitola 5.1). Počítač Raspberry Pi a modul převodníku jsou uloženy ve větší krabičce, než ostatní moduly a jsou společně s modulem řízení enzymu a regulace teploty připevněny na DIN liště v prostoru na zadní straně zařízení Golem. Moduly pro řízení míchání jsou umístěny na spodní straně platformy s vaky, kde se také nacházejí krokové motory. Modul regulace pH je umístěn na levé straně vnitřního prostoru zařízení.

Moduly jsou vzájemně propojeny v následujícím pořadí

1. Modul převodníku sběrnic UART a CAN
2. Modul vstřikování enzymu
3. Modul regulace teploty
4. Modul řízení míchání a přečerpávání vaků 1 až 4
5. Modul regulace pH

Na prvním a posledním zapojeném modulu je osazen zakončovací resistor CAN sběrnice (viz kapitola 5.1.3). Všechny propojené moduly systému jsou zachyceny na obrázku 6.8.



Obrázek 6.8: Sestavený řídicí systém. Na snímcích jsou otevřené a zavřené krabičky s moduly. Jejich pořadí je zleva: Raspberry Pi s CAN převodníkem, vstřikování enzymu, regulace teploty, řízení míchání a modul regulace pH.

Kapitola 7

Softwarové komponenty řídicí jednotky

Hlavní řídicí jednotkou systému je počítač Raspberry Pi ve verzi 4. Ta se stará mj. o:

- řízení a koordinaci jednotlivých modulů,
- poskytování webové aplikace,
- poskytování REST aplikačního rozhraní pro webovou aplikaci,
- správu uživatelů, experimentů, kalibraci pH sond a další.

K Raspberry Pi je připojen převodník z UART na CAN sběrnici, představený v kapitole 6.7.

7.1 Webová aplikace

Návrhu webové aplikace se ve své diplomové práci věnuje Bc. Jan Trulář (viz [49]). Ta je psána pomocí webového frameworku VueJs a k sestavení projektu využívá nástroj NPM. Výstupem sestavení je několik statických souborů (*index.html*, JS a CSS soubory), které je možné poskytovat uživateli. K tomu je využit nástroj *nginx*, nastavený dle návodu [10]. Soubory, které jsou součástí webové aplikace poté stačí umístit do adresáře */var/www/html* a *nginx* se již postará o jejich poskytování při přístupu na IP adresu Raspberry Pi (port 80).

7.2 Vyhledání Raspberry Pi v síti

Raspberry Pi již v základu obsahuje mDNS server (viz [17]), díky kterému je možné zařízení v síti vyhledat pod jménem *raspberrypi*. Toto jméno je možné změnit úpravou souborů */etc/hosts* a */etc/hostname*, následovanou restartem počítače. Pro účely projektu bylo jméno změněno na *golem*, takže po přístupu na URL *golem.local* je uživateli zobrazena webová aplikace.

7.3 Řídicí aplikace

Na počítači Raspberry Pi běží aplikace, která má na starosti komunikaci s uživatelem na jedné straně a řízení systému dle jeho požadavků na straně druhé. Aplikace je psaná v jazyku

TypeScript, který je transpilován do jazyku JavaScript. Běh JS souborů na serveru je možný díky nástroji NodeJs (viz [15]). Ten přináší standardní knihovny pro práci se souborovým systémem, řízení procesů a další. Pro sestavení a poskytování aplikace je využit nástroj NPM (viz [16]). Zdrojové soubory aplikace jsou dostupné v příloze této práce. Hlavní část aplikace je obsažena ve složce *src*.

7.3.1 Aplikační rozhraní

Raspberry Pi poskytuje REST aplikační rozhraní (dále API) pro webovou aplikaci. API je psané za použití knihovny ExpressJS, sloužící k jednoduché specifikaci koncových bodů rozhraní. Kompletní specifikace API je dostupná v příloze B. Funkce pro obsluhu volání koncových bodů API jsou implementovány v souboru *index.ts*.

7.3.2 Persistence dat

Pro zajištění persistence dat byl vybrán jednoduchý databázový systém NeDB psaný v jazyku JavaScript (viz [14]). Ten pro ukládání dat vytváří pro každou databázi jeden soubor, ve kterém ukládá data ve formátu JSON (každý řádek odpovídá jednomu JSON objektu).

Na základě NeDB bylo vytvořeno několik databází, rozdělených podle typu uložených dat. Tyto databáze jsou definovány pomocí objektu, jehož zkrácená ukázka je uvedena ve výpisu 7.1. Objekt *persistentConfig* je poté využit ke generování objektu pro přístup k databázím.

```
export const persistentConfig: PersistentConfig = {
    users: {
        fileName: 'users.db',
        backup: true,
        indices: [
            {fieldName: 'id', unique: true},
            {fieldName: 'login', unique: true}
        ]
    },
    tokens: { ... },
    modules: { ... },
    ntp: { ... },
    calibration: { ... },
    simulations: { ... },
    runs: { ... },
    drugs: { ... }
};
```

Výpis 7.1: Definice použitých databází

Samotný objekt NeDB, sloužící k manipulaci s jednou databází, byl mírně upraven. Jeho původní metody, určené např. ke vkládání a mazání záznamů, totiž pro signalizaci úspěchu či neúspěchu využívají volání funkce předané v parametru. Tento přístup ale u JavaScriptu často vede k nepřehlednému kódu. Proto byly zmíněné funkce přepsány, aby vracely tzv. *Promise*, tedy objekty sloužící k reprezentaci výsledku asynchronních akcí, na které je možné pomocí jejich metody *then* registrovat akce po získání výsledku a ty také dále řetězit (viz [13]).

7.3.3 Správa uživatelů

Systém obsahuje dvě úrovně uživatelů – *standardní uživatel* a *administrátor*. Administrátor může provádět například správu ostatních uživatelů. Zpřístupněny mu jsou funkce pro jejich přidání, aktualizaci a mazání. Dále může administrátor například vidět uzamčené simulace ostatních uživatelů, provádět kalibrace, zálohy systému a další.

Při přihlášení je uživateli vygenerován kód s omezenou platností, který je uložen na serveru i v prohlížeči uživatele. Tento kód poté uživatel posílá s každým dotazem a je tak díky němu autentizován.

7.3.4 Vytváření záloh

Pro uložení aktuálního stavu (nastavení simulací, uživatelů, kalibrací, atd.) je možné vytvářet zálohy systému. Jejich konfigurace vychází z nastavení *backup* z objektu *persistentConfig*, který byl představen ve výpisu 7.1. Pokud je *backup* rovno *true*, je obsah databázového souboru při vytváření zálohy zkopírován do souboru se zálohou. První řádek souboru se zálohou obsahuje informace o záloze (id a datum zálohy). Po něm následují obsahy databázových souborů, oddělené sekvencí znaku '=' následované názvem a cestou k databázovému souboru. Ukázka části zálohy je vidět ve výpisu 7.2.

```
{"id":0,"date":"9.6.2020 13:31:44","type":1}

================================(users.db) [persistent/users.db]
{..., "login":"admin","password":"123456", ...}
...

================================(modules.db) [persistent/modules.db]
{"address":64,"offset":0, ...}
...
```

Výpis 7.2: Ukázka části souboru se zálohou systému

7.3.5 Kalibrace pH sond

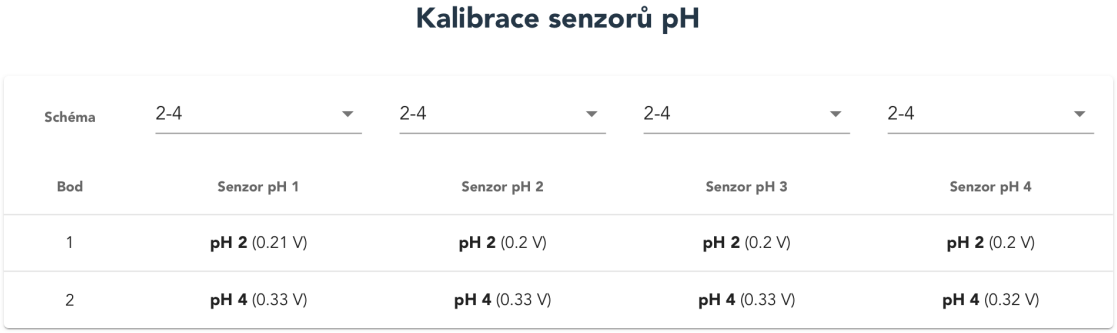
Součástí systému je také možnost kalibrace sond. Příkazy pro její provádění byly představeny v kapitole 6.6 věnované pH modulu. Uživatel má v aplikaci možnost volit z předdefinovaných kalibračních schémat (pH 2-4, nebo pH 2-4-7), popřípadě může zvolit body ručně. Snímek obrazovky kalibrace z webové aplikace je na obrázku 7.1.

7.3.6 Nastavení simulací

Struktura simulací je hierarchická a skládá se ze tří úrovní. Na nejvyšší úrovni nalezneme tzv. *schéma*. To obsahuje například nastavení požadované teploty, počáteční objem v prvním vaku, do jakého vaku je zaveden enzym a pH vaků. Jeho součástí je také rozvrh simulace, který specifikuje v jakých časech má dojít k jakým událostem (například změna rychlosti míchání, přečerpání objemu mezi vaky, vstříknutí enzymu, zobrazení notifikace uživateli a další).

Další úrovní je tzv. *experiment*, který dále specifikuje jaké léčivo je při simulaci použito, jeho forma (viz kapitola 2.3), dávkování a podobně.

Třetí úrovní jsou jednotlivé běhy experimentů, sloužící pro opakování experimentu se stejným nastavením.



Obrázek 7.1: Obrazovka kalibrace pH sond

7.3.7 UART

Pro komunikaci s převodníkem CAN sběrnice je použita sběrnice UART, po které jsou posílány zprávy jednoduchého protokolu, představeného v kapitole 6.7. V souboru *UartCommands.ts* jsou implementované funkce pro zasílání a příjem zpráv převodníku. Aby nedošlo k zahlcení sběrnice, je implementován i buffer odchozích příkazů, který se stará o to, aby byl zahájen nový přenos až po dokončení předchozího přenosu a také udržuje informace o nepotvrzených příkazech. Ty po době nastavené konstantou *COMMAND_TIMEOUT* končí neúspěchem.

Pro snazší práci s UART rozhraním jsou obslužné funkce implementovány tak, aby vracely objekty typu Promise, které po úspěšném obdržení potvrzení příkazu vracejí objekt s přijatými daty (popřípadě prázdný objekt) a také předávají zprávu o chybě v případě neúspěchu.

Na této úrovni také dochází k převodu hodnot parametrů, reprezentovaných jako čísla jazyku JavaScript, na jejich bytovou reprezentaci, reflektující jejich datový typ v C. K tomu je využit vestavěný objekt Buffer, nad kterým jsou napsané konverzní funkce (viz soubor *NumberConverters.ts*. Ty jsou využity pro převod dat odesílaných a přijímaných přes UART. Ve výpisu 7.3 je vidět použití konverzních funkcí.

```
const buffer = arrayToBuffer([U8, U16, U32, [U8, U8]])([1, 2, 3, [4, 5]]);
const pole = bufferToArray([U8, U16, U32])(buffer);

// buffer = <Buffer 01 02 00 03 00 00 00 04 05>
// pole = [1, 2, 3, [4, 5]]
```

Výpis 7.3: Konverzní funkce pro převod čísel na jejich bytovou reprezentaci

7.3.8 Řízení modulů

Řízení modulů reflektuje jejich modulární rozdělení, popsané v kapitole 4.5, a je založeno na funkcích pro obsluhu UART. Moduly jsou implementovány jako třídy, které dědí ze základní generické třídy *ModuleBase*, implementované v souboru *ModuleBase.ts*. Tento soubor i soubory s odvozenými třídami jsou ve složce *HardwareSystem/modules*.

Všechny odvozené třídy udržují informaci o typu a instanci modulu, který řídí, obsahují pole nazvané *autoUpdate*, které obsahuje definici operací sloužících k automatickému získávání dat z jimi řízených modulů (například pro účely záznamu vývoje teploty a pH v čase

apod.) a také (ve slotu *controls*) nesou funkce pro volání a reakci na příkazy, které daný modul podporuje. Funkce v tomto slotu již plně reflektují typy odesílaných dat (z pohledu jazyku TypeScript), což snižuje riziko chyby při práci s příkazy.

Moduly využívají pro perzistenci některých dat (například kalibrace sond, nastavení PID regulátorů, atd.) databázi *modules.db*. Odvozené třídy mají předdefinovaný typ, instance je předána jako parametr konstruktoru.

7.3.9 Třída Golem

Instance třídy *Golem* sdružuje všechny objekty zmíněné v kapitole 7.3.8 a obsahuje například metody pro spuštění, zastavení a získání stavu pro všechny moduly současně. Řízení modulů zpřístupňuje třída Golem ve slotu *modules*. V této části se již k modulům nepřistupuje na základě jejich typu a instance, ale jsou pro ně vytvořené objekty instancované z jejich tříd. Definice slotu *modules* je vidět ve výpisu 7.4. Postup pro přidání nového modulu do systému je popsán v příloze C.

```
this.modules = {
    enzyme: new EnzymeModule(0),
    temperature: new TemperatureModule(0, persistentDb),
    bags: [
        new BagModule(0),
        new BagModule(1),
        new BagModule(2),
        new BagModule(3)
    ],
    ph: new PhModule(0, persistentDb)
};
```

Výpis 7.4: Definice slotu objektu Golem pro obsluhu modulů

7.3.10 Třída System

Třída *System* obsahuje všechny objekty potřebné pro provádění běhů experimentů a ovládání zařízení Golem. Jedná se tak o objekt na nejvyšší úrovni, jehož prostřednictvím je možné ovládat všechny části systému – jak hardwarové části, tak i serverové aplikace.

Obsahuje informace o aktuálně prováděném experimentu (aby nemohlo dojít k běhu více experimentů najednou, což by vedlo ke konfliktu), metody pro spouštění, pozastavení a ukončení experimentu, metody pro obsluhu pokynů k ruční úpravě stavu zařízení Golem zadané uživatelem (změna teploty, pH, atd.), ukládá potvrzení akcí uživatele (například odběr vzorku), stará se o automatické snímkování hodnot při běhu experimentu a také sestavuje grafy z naměřených hodnot.

K časování událostí v průběhu experimentu je využit tzv. *Event Loop* jazyku JavaScript (viz [12]). Díky němu není nutné spouštět akce v jejich čas ručně, ale je možné je naplánovat hned na začátku experimentu pomocí funkcí *setTimeout(f, ms)* a *setInterval(f, ms)*. První ze zmíněných slouží k naplánování spuštění funkce f po uplynutí ms milisekund, druhá z nich slouží k volání funkce f každých ms milisekund. Výhodou tohoto přístupu je, že provedení funkcí je asynchronní a plánování vychází přímo z vlastností jazyku. Před zahájením běhu experimentu je nejdříve z časového rozvrhu schématu vytvořen seznam stavových událostí, ze kterých jsou odstraněny ty již provedené (pokud se jedná o pokračování pozastaveného

experimentu). Tyto události jsou naplánovány na správný čas pomocí funkce *setTimeout*. Funkce *setInterval* je využita například při snímkování stavu systému a ukládání dat z běhu experimentů.

Kapitola 8

Kalibrace a ladění částí systému

Některé části systému je před uvedením do provozu třeba zkalibrovat, popřípadě správně nastavit jejich parametry. Konkrétně se jedná o

- pH sondy (viz kapitola 5.10.4)
- teploměry (viz kapitola 6.4)
- senzory tlaku na tabletu (viz kapitola 5.8)
- peristaltické pumpy (viz kapitola 5.12)
- PID regulátor teploty (viz kapitola 6.4)
- PID regulátor pH (viz kapitola 6.6)

Každé z uvedených částí se věnuje podkapitola této sekce.

8.1 Kalibrace pH sond

Výstup pH sond byl porovnán s očekávanými hodnotami, uvedenými v tabulce 5.9. Ukázalo se, že zdroj referenčního napětí je posunut přibližně o 200 mV. Tato skutečnost bude napravena úpravou pasivních součástek zdroje referenčního napětí, popsaného v kapitole 5.10.1.

Dále byl ověřen vztah výstupu sond na vstupním pH. V této fázi byly zaznamenávány hodnoty výstupu analogového převodníku mikrokontroléru (šestnáctibitový). Statistika ze sta opakování měření roztoků s pH 7 a pH 4 je vidět v tabulce 8.1.

Tabulka 8.1: Výstup analogového převodníku při měření roztoků s pH 4 a pH 7

pH	**Průměr**	**Medián**	**Minimum**	**Maximum**
4	49938	49953	49569	50230
7	46505	46526	46140	46806

Z naměřených hodnot je vidět, že je nulová hodnota (pH 7) posunuta oproti hodnotě odpovídající polovině napájecího napětí (tj. 32768) o 13737, což je způsobeno posunem zdroje referenčního napětí (ten je teoreticky roven 13 107 dílků) – viz předchozí odstavec. Také je z naměřených dat vidět, že je výstupní funkce plošší, než udává teorie (maximální rozsah převodníku podělený čtrnácti díly stupnice pH odpovídá přibližně 4700 dílků na jednotku pH, naměřená data udávají přibližně 1140 dílků na jednotku). Nalezení příčiny této skutečnosti bude provedeno v rámci finální realizace práce.

8.2 Kalibrace teploměrů

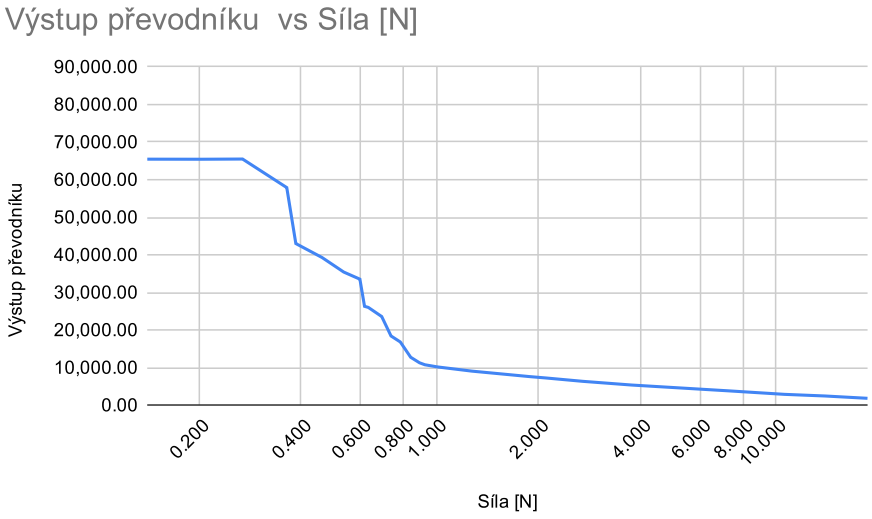
Systém Golem obsahuje čtyři teploměry DS18B20. Dle jejich dokumentace [34] jsou „zkalibrovány pro výstup ve stupních Celsia“. Pokusy provedené s teploměry ukázaly, že se jejich výstup mírně liší, a proto byla provedena kalibrace. Ta předpokládá že funkce závislosti výstupu teploměrů na teplotě je lineární, tyto funkce mají pro různé teploměry stejnou první derivaci a liší se tedy pouze posunem nulové hodnoty.

Pro zjištění vzájemného posunu výstupu čidel vůči sobě byly teploměry umístěny do termosky, která byla poté co nejlépe utěsněna a v ní byly teploměry ponechány přes noc pro ustálení teploty. Následné zjištění výstupu čidel ukázalo, že se jejich výstup liší o ±2 dílky (jeden díl odpovídá dle dokumentace $\frac{1}{16}$ °C). Příkazem *SET_THERM_OFFSET* poté byl výstup čidel upraven tak, aby všechna ukazovala stejnou hodnotu.

Ve druhém kroku bylo třeba upravit výstup čidel tak, aby odpovídala reálné teplotě. Protože autor práce při kalibraci neměl přístup k přesnému teploměru, bylo porovnání provedeno oproti výstupu z domácí stanice Netatmo Healthy Home Coach. Zjištěná odchylka byla cca 2 °C, což je po převedení 32 dílků výstupu teploměru. Tato odchylka byla odstraněna příkazem SET_TOTAL_OFFSET.

8.3 Kalibrace senzorů síly na tabletu

Senzor síly na tabletu byl zkalibrován dle postupu uvedeného v kapitole 5.8. Graf závislosti výstupu analogového převodníku na působící síle je vidět na obrázku 8.1.



Obrázek 8.1: Graf závislosti výstupu analogového převodníku na působící síle

Pro každou hmotnost bylo provedeno deset měření výstupu a z nich vypočten průměr. Dle dokumentace použitého tenzometru [39] je možné měřit působící sílu od cca 0.1 N. Graf ale ukázal, že validní hodnoty senzor vrací až od cca 0.4 N. To je zřejmě způsobeno nitrilovou membránou na sondě tenzometru a také umístěním gumového profilu přes tenzometr. Pro výpočet výsledné logaritmické funkce proto byly použity body se sílou 0.4 N a výše. Dá se předpokládat, že protože je senzor použit pro měření maximální síly, nedochází touto úpravou k újmě na obecnosti a zároveň zpřesňuje výslednou funkci.

Naměřené hodnoty byly dále převedeny na napětí na převodníku, které pak bylo použito pro výpočet odporu tenzometru. Naměřené hodnoty a jednotlivé kroky výpočtu jsou zaneseny v příloze D. Na základě odporu tenzometru již bylo možné za pomoci regrese stanovit logaritmickou funkci, která má tvar:

$$y = 48069 + -23658ln(x) \tag{8.1}$$

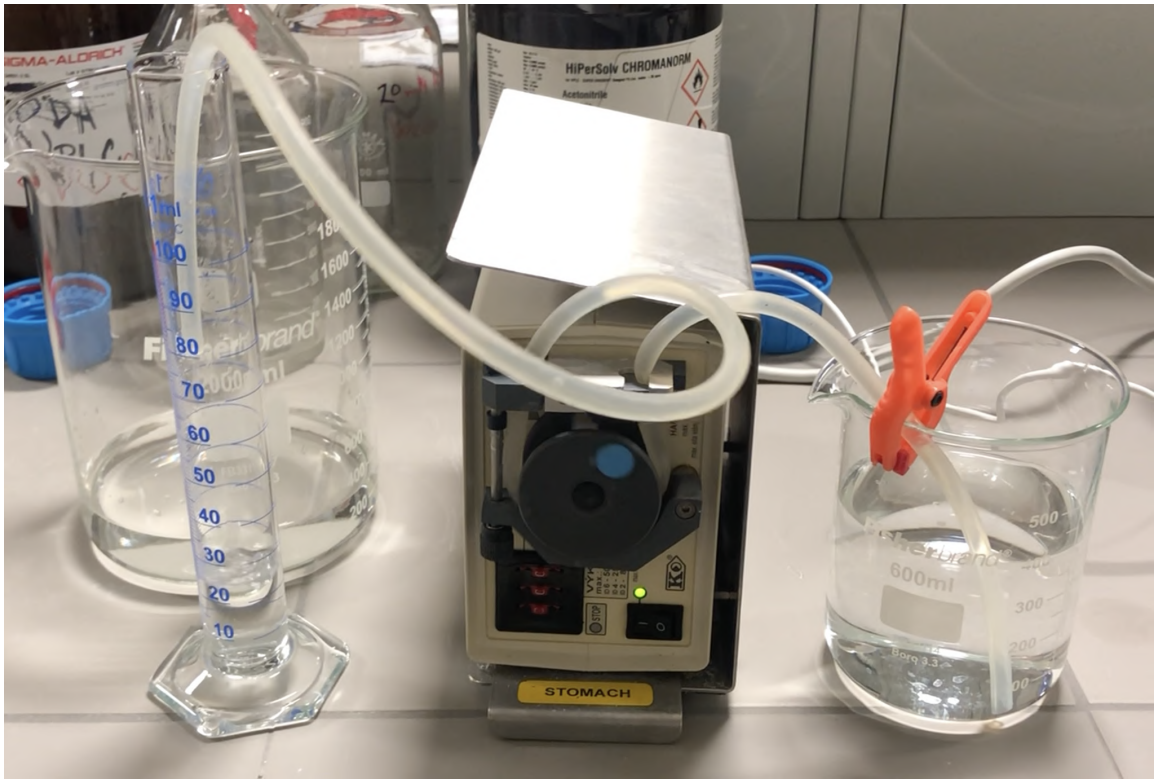
Tato funkce je využita pro výpočet síly působící na podložku, který je implementován v souboru *force.c*. Na obrázku 8.2 je vidět postup kalibrace sensoru síly.

Obrázek 8.2: Postup kalibrace senzoru síly na tabletu. Na prvním obrázku je na sensor umístěna pouze tableta, poté dřevěná lať ka a na ni postupně přidáváno závaží.

8.4 Kalibrace peristaltických čerpadel

Peristaltická čerpadla použitá v systému Golem (viz kapitola 5.12) je možné ovládat pouze dvoustavově („*čerpá*“/„*nečerpá*“) a přečerpaný objem je tedy možné regulovat pouze na základě času čerpání. Pro zjištění času potřebného k přečerpání jednoho mililitru byl proveden pokus na obrázku 8.3.

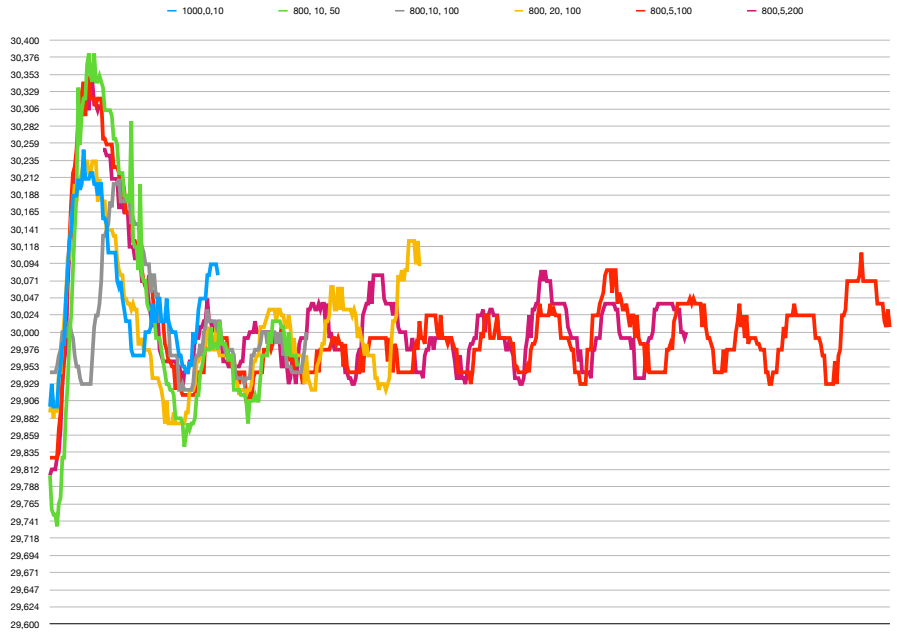
Na peristaltickém čerpadle byla přepínačem nastavena rychlost čerpání 300. Voda z kádinky vpravo byla přečerpávána do odměrného válce vlevo a byl měřen čas, za který dojde k přečerpání 100 ml. Na základě pěti měření bylo zjištěno, že na přečerpání 1 ml je potřeba cca 600 ms.



Obrázek 8.3: Pokus pro zjištění rychlosti přečerpávání peristaltických čerpadel

8.5 Ladění PID regulátoru teploty

Při použití PID regulátoru je nutné správně zvolit jeho konstanty, které se liší dle řešeného problému (viz kapitola 5.4). Část ladění těchto konstant (pro regulaci teploty i pH) obsahuje dle autora práce jisté nedostatky způsobené mj. dlouhou nepřístupností laboratoří na VFU z důvodů způsobených nemocí COVID-19. Při jedné návštěvě laboratoře v závěru realizace práce ale byly uskutečněny pokusy s topením a úpravami konstant PID regulátoru. Jejich hledání probíhalo pouze naivně bez použití systematického přístupu. Graf průběhu teploty při ladění PID regulátoru je vidět na obrázku 8.4. V rámci finálních úprav bude provedeno ladění na základě některého ze systematických ladících postupů, například Ziegler-Nicholsovy metody (viz článek [52]).



Obrázek 8.4: Graf průběhu teploty pro různá nastavení PID. Legenda os značí popořadě konstanty k_p, k_i a k_d.

8.6 Ladění PID regulátoru pH

Ladění PID regulátoru nebylo uskutečněno, protože při návštěvě laboratoře bylo zjištěno, že software obsahuje chybu komunikace s pumpami a ventily Cavro. Tato chyba byla odstraněna na základě zapůjčené pumpy, ale již nebylo možné provést pokusy s reálnou soustavou.

Kapitola 9

Testování systému

Testování bylo provedeno pouze v omezené míře. Jednak z důvodů způsobených nemocí COVID-19 (viz kapitola 8.5), druhak kvůli faktu, že Jan Truhlář, autor webové aplikace, se rozhodl odevzdat jeho práci v pozdějším termínu a nebylo tak možné provést kompletní testy. Ty totiž vyžadují provedení některých nevratných úprav současného zařízení Golem, po kterých by bylo do dokončení webové aplikace zařízení nepoužitelné. Přímo na zařízení Golem bylo otestováno:

- měření teploty a ovládání topení (viz kapitola 8.5),
- ovládání peristaltických čerpadel (viz kapitola 8.4),
- měření a kalibrace pH (viz kapitola 8.1)

V domácích podmínkách poté bylo otestováno:

- vstřikování enzymu, kyseliny a zásady – otestováno na zapůjčené pístové pumpě,
- měření síly působící na tabletu (viz kapitola 8.3),
- buzení krokových motorů (k testům byla využita čtveřice motorů SX17-0905), uvedení do krajní polohy pomocí koncového spínače a reakce na překročení maximální síly na tabletu.

Na základě testovací sestavy (viz obrázek 6.8) s připojenými dostupnými komponentami, byly provedeny i testy běhu experimentů. Při nich bylo ověřeno, že jsou události z časového rozvrhu schématu provedeny ve správný čas a také byla ověřena možnost změny nastavení v průběhu experimentu (například změna rychlosti míchání apod.),

Při těchto experimentech byly pH sondy nahrazeny potenciometrem, zapojeným jako dělič napětí tak, že na vstup BNC konektoru přivedl napětí 0 až 500 mV, spínání topení bylo ověřeno na osciloskopu a spouštění peristaltických čerpadel bylo ověřeno měřením odporu OPTOMOSFETu. K modulu vstřikování byla připojena skutečná pístová pumpa, stejně tak byly připojeny teploměry, senzory síly na tabletu, motory a koncové spínače.

Kapitola 10

Závěr

Tato práce představila návrh modulárního řídicího systému pro řízení laboratorního zařízení Golem, používaného při výzkumu léčiv.

Práce poskytuje letmý vhled do problematiky výzkumu léků a také popisuje lidskou trávicí soustavu, kterou zařízení Golem simuluje.

Součástí práce je popis vývoje zařízení Golem a důkladná analýza koncepce řídicí části současné verze. Na základě této analýzy je proveden návrh rozdělení systému do modulů se stanoveným rozhraním, které spolu komunikují prostřednictvím sběrnice CAN. Pro tuto sběrnici byl také navržen komunikační protokol, použitý pro řízení modulů.

Jako hlavní jednotka byl použit počítač Rasberry Pi, poskytující backendovou a webovou aplikaci. První z uvedených se stará o řízení celého systému a také poskytuje aplikační rozhraní pro webovou aplikaci, která slouží k interakci s uživatelem (webová aplikace není součástí této práce).

Detailně jsou popsány jednotlivé softwarové a hardwarové bloky, které byly při tvorbě modulů použity.

Výsledný systém obsahuje pět typů modulů (řízení vstřikování enzymu, regulace teploty, regulace pH, řízení míchání vaků a převodník sběrnic UART a CAN) a počítač Raspberry Pi. Pro všechny zmíněné části byl vytvořen firmware a také byly navrženy kalibrační procedury pro komponenty, které využívají.

Součástí práce bylo vytvoření prototypů modulů, na kterých byl systém v simulovaných podmínkách důkladně otestován.

Literatura

[1] *1-wire communication through software*. [cit. 2020-06-10]. Dostupné z: https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/126.html.

[2] *ABS Plastic Hand Held/Instrument Enclosures (1593 Series)*. [cit. 2020-06-07]. Dostupné z: http://www.hammondmfg.com/dwg8.htm.

[3] *B+ HAT Eagle CAD template*. [cit. 2020-06-07]. Dostupné z: https://www.raspberrypi.org/forums/viewtopic.php?t=82618.

[4] *Datasheet: PT1000*. [cit. 2020-06-01]. Dostupné z: https://cz.rs-online.com/web/p/platinove-odporove-snimace-teploty/3342616/.

[5] *Elektronická učebnice – Jednoduché spojité regulátory*. [cit. 2020-06-07]. Dostupné z: https://eluc.kr-olomoucky.cz/verejne/lekce/948.

[6] *FlaTrode: Specification Sheet*. Dostupné z: https://www.hamiltoncompany.com/laboratory-products/product-specs/49676.

[7] *Force Sensitive Resistor*. [cit. 2020-06-07]. Dostupné z: https://www.sparkfun.com/products/9375.

[8] *FRDM-KV11Z: Freedom Development Platform for Kinetis® KV1x Family 128 KB, 64 KB, 32 KB and 16 KB Flash MCUs*. [cit. 2020-06-07]. Dostupné z: https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-kv1x-family-128-kb-64-kb-32-kb-and-16-kb-flash-mcus:FRDM-KV11Z.

[9] *Hammond 1593KBK*. [cit. 2020-06-07]. Dostupné z: https://www.hammfg.com/part/1593KBK.

[10] *How to host a static website on your Raspberry Pi 3 with Raspbian Stretch Lite and Nginx*. [cit. 2020-06-07]. Dostupné z: https://www.techcoil.com/blog/how-to-host-a-static-website-on-your-raspberry-pi-3-with-raspbian-stretch-lite-and-nginx/.

[11] *MCUXpresso Software Development Kit (SDK)*. [cit. 2020-06-07]. Dostupné z: https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools/mcuxpresso-software-development-kit-sdk:MCUXpresso-SDK.

[12] *MDN: Concurrency model and the event loop*. [cit. 2020-06-07]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop.

[13] *MDN: Promise*. [cit. 2020-06-07]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise.

[14] *NeDB: The JavaScript Database, for Node.js, nw.js, electron and the browser.* [cit. 2020-06-07]. Dostupné z: https://github.com/louischatriot/nedb.

[15] *NodeJS.* [cit. 2020-06-07]. Dostupné z: https://nodejs.org/en/.

[16] *NPM Web.* [cit. 2020-06-07]. Dostupné z: https://www.npmjs.com.

[17] *Raspberry Pi Documentation: IP Address.* [cit. 2020-06-07]. Dostupné z: https://www.raspberrypi.org/documentation/remote-access/ip-address.md.

[18] *TMC2209-BOB.* [cit. 2020-06-07]. Dostupné z: https://www.trinamic.com/support/eval-kits/details/tmc2209-bob/.

[19] TMC2209 Datasheet. Dostupné z: https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC2209_Datasheet_V105.pdf.

[20] Ws2812B, Intelligent control LED integrated light source. [cit. 2020-06-04]. Dostupné z: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf.

[21] Operating Manual Cavro ® Smart Valve Plus. 2006, October.

[22] 1.0 A Low-Dropout Positive Fixed and Adjustable Voltage Regulators. 2008, s. 1–17, [cit. 2020-06-04]. Dostupné z: https://www.onsemi.com/pub/Collateral/NCP1117-D.PDF.

[23] AN-1852 Designing With pH Electrodes. *Texas Instruments.* 2008, September, s. 1–7. Dostupné z: http://www.ti.com/lit/an/snoa529a/snoa529a.pdf.

[24] Operator's Manual Model XE 1000 Pump. 2009, October.

[25] High-Speed CAN Transceiver. *Technology.* 2010, November, s. 1–24, [cit. 2020-06-07]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/TJA1051.pdf.

[26] 0.6V ADJUSTABLE PRECISION SHUNT REGULATOR. 2011, March, s. 1–13, [cit. 2020-06-03]. Dostupné z: https://datasheetspdf.com/pdf-file/693138/DIODES/ZXRE060/1.

[27] LMP91200 Configurable AFE for Low-Power Chemical Sensing Applications. 2012, June, [cit. 2020-06-05]. Dostupné z: https://www.ti.com/lit/ds/symlink/lmp91200.pdf?ts=1591824465132.

[28] *Veřejná zakázka: Disoluční zařízení GOLEM 2.* 2013. Dostupné z: https://zakazky.vfu.cz/contract_display_37.html.

[29] Hybridní dvoufázové motory řady SX. 2015, sv. 3, č. 2, s. 54–67. Dostupné z: http://repositorio.unan.edu.ni/2986/1/5624.pdf.

[30] *Český lékopis 2017: Tištěná verze (1.-4. díl).* Grada Publishing a.s., 2017. ISBN 9788027105007.

[31] Kinetis V Series KV10 and KV11, 128/64 KB Flash. 2017, [cit. 2020-06-02]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/KV11P64M75.pdf.

[32] KV11 Sub-Family Reference Manual. 2017, [cit. 2020-06-02]. Dostupné z: https://www.nxp.com/files-static/32bit/doc/ref_manual/KV11P64M75RM.pdf.

[33] TL431 / TL432 Precision Programmable Reference. *Texas Instruments*. 2018, č. 1. Dostupné z: www.ti.com:http://www.ti.com/lit/ds/symlink/tl431.pdf.

[34] DS18B20 Programmable Resolution 1-Wire Digital Thermometer. 2019, sv. 92, s. 1–20, [cit. 2020-06-06]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

[35] Industrial Solid State Relays. 2019, s. 1–6, [cit. 2020-06-04]. Dostupné z: https://www.tme.eu/Document/97fa6be68b031b7c9a22a428a9d829a1/rs1a_EN.pdf.

[36] *European Pharmacopoeia, 2.9.3. Dissolution test for solid dosage forms*. 6. vyd. Council Of Europe, 2007. Strasbourg, 2007.

[37] Cook, J. A. a Freudenberg, J. S. EECS461: Controller Area Network (CAN). 2008, s. 1–8. Dostupné z: https://www.eecs.umich.edu/courses/eecs461/doc/CAN{_}notes.pdf.

[38] Cooke, G., Behan, J. a Costello, M. Newly identified vitamin K-producing bacteria isolated from the neonatal faecal flora. *Microbial Ecology in Health and Disease*. Taylor & Francis. 2006, sv. 18, 3-4, s. 133–138. DOI: 10.1080/08910600601048894. Dostupné z: https://doi.org/10.1080/08910600601048894.

[39] Interlink Technologies. FSR 402 Data Sheet. 2013, s. 1–4. Dostupné z: http://www.interlinkelectronics.com/FSR402short.php.

[40] Jelínek, J. a Zicháček, V. *Biologie pro gymnázia: (teoretická a praktická část)*. Nakladatelství Olomouc, 2004. ISBN 9788071821779. Dostupné z: https://books.google.cz/books?id=SXWItwAACAAJ.

[41] Koziolek, M., Grimm, M., Becker, D., Iordanov, V., Zou, H. et al. Investigation of pH and temperature profiles in the GI tract of fasted human subjects using the Intellicap® system. *Journal of pharmaceutical sciences*. Elsevier. 2015, sv. 104, č. 9, s. 2855–2863. Dostupné z: https://www.jpharmsci.org/article/S0022-3549(16)30065-X/pdf.

[42] LeBlanc, J. G., Milani, C., Savoy, G., Sesma, F., Van Sinderen, D. et al. Bacteria as vitamin suppliers to their host: A gut microbiota perspective. *Current opinion in biotechnology*. Srpen 2012, sv. 24. DOI: 10.1016/j.copbio.2012.08.005.

[43] *Lékopis*. Státní ústav pro kontrolu léčiv, 2020 [cit. 2020-06-07]. Dostupné z: http://www.sukl.cz/farmaceuticky-prumysl/lekopis.

[44] Lincová, D. a Farghali, H. Základní a aplikovaná farmakologie. Leden 2005.

[45] Mudra, P. *Systém řízení GOLEM 2 – základní popis hardware a firmware Firmware*. 2018.

[46] Novotný, I. a Hruška, M. *Biologie člověka: pro gymnázia*. Fortuna, 2002. ISBN 9788071688198.

[47] Ruiz, M. *Trávicí trakt člověka*. 2020 [cit. 2020-06-02]. File: `Digestive_system_diagram_cs.svg`. Dostupné z: https://commons.wikimedia.org/wiki/File:Digestive_system_diagram_cs.svg.

[48] Stupák, I., Bílik, T., Vysloužil, J., Dohnal, J. a Čulen, M. Vytvorenie metódy pre dynamickú biorelevantnú disolúciu na prístroji Golem v2. In: *Konference interní grantové agentury*. 2018, s. 208–211. Dostupné z: https://www.vfu.cz/files/Sbornik_IGA_2018.pdf.

[49] Truhlář, J. *Softwarový nástroj pro vizuální specifikaci řízení průběhu laboratorních experimentů*. Diplomová práce. 2020.

[50] Večerek, V., Brauner, P., Král, K., Pospíšil, Z., Steinhauserová, I. et al. Časopis Veterinární a farmaceutické univerzity Brno, 2011/5. *Vita Universitas, Časopis Veterinární a farmaceutické university Brno*. Brno: [b.n.]. may 2011, s. 34. Dostupné z: https://www.vfu.cz/files/vu_2011_5.pdf.

[51] Watterson, C. AN-1123 APPLICATION NOTE Controller Area Network (CAN) Implementation Guide. *Analog Devices*. 2017, s. 1–14. Dostupné z: http://www.analog.com/media/en/technical-documentation/application-notes/AN-1123.pdf.

[52] Ziegler, J. G. a Nichols, N. B. Optimum settings for automatic controllers. *InTech*. 1995, sv. 42, č. 6, s. 94–100. ISSN 0192303X.

Příloha A

Přiřazení pinů pro řízení funkčních bloků

Tabulka A.1: Přiřazení pinů v jednotlivých modulech

	A: UART-CAN	B: Regulace pH	C: Řízení vaků	D: Regulace teploty	E: Vstřikování enzymu
1	3.3V (VDD)				
2	GND (VSS)				
3	STATUS LED (PTE16)				
4			PRESSURE_RST (PTE17)		
5			I2C_SDA (I2C0_SDA)		
6			I2C_SCL (I2C0_SCL)		
7	3.3V (VDDA/VREFH)				
8	GND (VREFL/VSSA)				
9	CAN_S (PTE30)				
10	CAN_TX (CAN0_TX)				
11	CAN_RX (CAN0_RX)				
12	SWD_CLK				
13					
14					
15	SWD_DIO				
16	NMI (NMI_b)				
17	EXTAL0				
18	XTAL0				
19	RESET (RESET_b)				
20		PH1 (ADC1_SE8)			
21		PH2 (ADC0_SE9)	PUMP (PTB1)		
22		PH3 (ADC1_SE3)	MOTOR_DIR (PTC1)		
23		PH4 (ADC0_SE11)	MOTOR_STEP (PTC2)	TEMP1 (PTC2)	
24			UART_RX (UART1_RX)	TEMP2 (PTC3)	
25	SPI_CS (SPI0_PCS0)		UART_TX (UART1_TX)	TEMP3 (PTC4)	
26	SPI_SCK (SPI0_SCK)		MOTOR_INDEX (PTC5)	TEMP4 (PTC5)	
27	SPI_MISO (SPI0_SOUT)		MOTOR_DIAG (PTC6)		
28	SPI_MOSI (SPI0_SIN)		MOTOR_SPREAD (PTC7)		
29			MOTOR_ENABLE (PTD4)		
30		RS485_EN (PTD5)	PRESSURE_EOC (PTD5)	HEAT_PLAT (PTD5)	RS485_EN (PTD5)
31		RS485_RX (UART0_RX)	FORCE (ADC1_SE6)	HEAT_FAN (PTD6)	RS465_RX (UART0_RX)
32		RS485_TX (UART0_TX)	ENDSTOP (PTD7)		RS485_TX (UART0_TX)

Příloha B

Specifikace aplikačního rozhraní

Backendová aplikace poskytuje pro komunikaci s webovou aplikací REST aplikační rozhraní. Implementaci jednotlivých koncových bodů rozhraní obsahuje soubor *index.ts*. Každý z koncových bodů má specifikováno rozhraní parametrů, těla příchozího požadavku a odchozích dat. Toho je docíleno pomocí generických funkcí *apiGet* a *apiPost*, implementovaných v souboru *expressWrapper.ts*. Význam jednotlivých koncových bodů je popsán dále.

- /login (POST) – Přihlášení uživatele na základě jeho údajů. Součástí odchozích dat je přihlašovací klíč.
- /userData (GET) – Získání informací o přihlášeném uživateli.
- /logout (POST) – Odhlášení uživatele (smazání přihlašovacího klíče z databáze)
- /config/debug (GET) – Aktuální informace o systému (aktivita čidel, dostupné zálohy apod)
- /config/debug/backup/restore (POST) – Obnovení systému dle zvolené zálohy
- /config/debug/backup/create (POST) – Vytvoření zálohy systému
- /config/users
  - GET – Získání seznamu uživatelů
  - POST – Uložení aktualizace seznamu uživatelů
- /config/users/remove (POST) – Smazání uživatele
- /config/params (GET) – Získání nastavení systému (Wifi, NTP)
- /config/params/sync (POST) – Zahájení synchronizace s NTP serverem
- /config/calibration (GET) – Získání aktuálních kalibračních dat pH sond.
- /config/calibration/:probeId/getRaw (GET) – Získání hodnoty aktuálně měřené na pH sondě
- /config/calibration (POST) – Nastavení kalibrace
- /recent (GET) – Seznam posledních experimentů

- /simulation/open (GET) – Získání seznamu schémat
- /simulation/:simulationId/edit
  - GET – Získání nastavení schématu simulace
  - POST – Uložení nastavení schématu simulace
- /simulation/:simulationId/parameters
  - GET – Získání rozvrhu simulace
  - POST – Uložení rozvrhu simulace
- /simulation/:simulationId/behavior
  - GET – Získání nastavení grafické specifikace chování simulace
  - POST – Uložení nastavení grafické specifikace chování simulace
- /simulation/:simulationId/experiment/open (GET) – Získání seznamu experimentů zadaného schématu
- /simulation/:simulationId/experiment/:experimentId/edit
  - GET – Získání nastavení experimentu
  - POST – Uložení nastavení experimentu
- /simulation/:simulationId/experiment/:experimentId/run/open (GET) – Získání seznamu běhů experimentu
- /simulation/:simulationId/experiment/:experimentId/run/create (GET) – Vytvoření nového běhu experimentu
- /simulation/:simulationId/experiment/:experimentId/run/:runId/execute
  - GET – Získání dat vykonávaného experimentu
  - POST – Provedení akce uživatele při vykonvání experimentu
- /simulation/:simulationId/experiment/:experimentId/run/:runId/update (GET) – Získání jednoho datového bodu běhu simulace (4x pH, teplota, objemy, atd.). Slouží k aktualizaci grafů a dalších čístí webové aplikace při běhu experimentu.
- /simulation/:simulationId/experiment/:experimentId/run/:runId/result (GET) – Získání přehledu dokončené simulace

Příloha C

Postup přidání nového modulu

Pro část hardware a jeho programování je doporučený postup:

1. Vytvoření hardware modulu dle kapitoly 5.1
2. Vytvoření firmware pro nový modul dle kapitoly 6.2
3. Definice kódů příkazů v souboru *common_commands.h*. Součástí implementace modulu musí být i reakce na přidané příkazy.
4. Definice typu modulu v souboru *golem_config.h*
5. Přidání modulu do souboru *golem.h*
6. Definice délky odchozích a příchozích dat v souboru *can_cmd_queue.c*.
7. Naprogramování modulu a převodníku UART-CAN

Pro přidání modulu do backendové aplikace je postup následující:

1. Přidání typu modulu do souboru *Addressing.model.ts*
2. Definice čísel nových příkazů v souboru *UartCommands.constant.ts* – shodné s příkazy v *common_commands.h*
3. Definice délek dat a převodníků parametrů příkazů v souborech *UartCommandsDataLengths.ts* a *UartCommandsDataConverters.ts*.
4. Přidání funkcí pro obsluhu UART příkazů v souboru *UartCommands.ts*
5. Vytvoření nové třídy dědící ze třídy *ModuleBase* – viz složka *HardwareSystem/modules*
6. Přidání nového modulu do třídy *Golem*
7. Použití přidaného modulu v rámci třídy *System*, přidání návaznosti na akce při běhu simulace apod.

Příloha D

Výstup převodníku použitý pro kalibraci senzoru síly působící na tabletu

Tabulka D.1: Hodnoty měřené analogově digitálním převodníkem při působení různé síly na senzor síly na tabletu

Váha [g]	**Síla [N]**	**Průměr**	**Vout**	**RT**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**0**	65437,7	3,29510	7338832,41744	65468	65399	65431	65428	65412	65428	65460	65462	65427	65462
14,1	**0,141**	65458,5	3,29615	7516503,99290	65430	65462	65463	65408	65430	65460	65428	65488	65524	65492
20,4	**0,204**	65439,1	3,29517	7350530,48637	65448	65460	65535	65335	65415	65387	65392	65535	65460	65424
26,9	**0,269**	65493,4	3,29791	7834479,53369	65512	65520	65476	65463	65456	65462	65512	65535	65535	65463
36,3	**0,363**	57896,9	2,91539	686595,31184	58376	58125	58125	58098	57951	57767	57631	57867	57518	57511
38,6	**0,386**	43007,4	2,16563	184407,28521	43041	43204	43121	43009	43109	42995	42913	42920	42913	42849
46,1	**0,461**	39299,7	1,97893	145394,70609	39257	39212	39260	39260	39263	39251	39440	39328	39246	39480
53,4	**0,534**	35431,5	1,78415	114672,97680	35384	35378	35474	35414	35414	35433	35416	35538	35480	35384
59,6	**0,596**	33532,9	1,68854	102245,47491	33668	33670	33543	33464	33542	33504	33456	33484	33478	33520
61,5	**0,615**	26309	1,32478	65739,03286	26356	26357	26359	26353	26309	26264	26281	26257	26261	26293
63,1	**0,631**	26019,5	1,31021	64548,71749	25795	25797	25727	25731	25731	25755	25751	27501	26210	26197
69,1	**0,691**	23593,5	1,18805	55207,72951	23667	23660	23627	23664	23556	23584	23477	23596	23532	23572
73,6	**0,736**	18421	0,92759	38450,48881	18385	18380	18443	18422	18438	18526	18410	18410	18398	18398
78,4	**0,784**	16819,9	0,84696	33973,10083	16872	16821	16824	16820	16800	16836	16800	16783	16787	16856
84,1	**0,841**	12756,7	0,64236	23811,95769	12793	12737	12737	12781	12761	12761	12777	12741	12738	12741
89,3	**0,893**	11288	0,56840	20508,21283	11273	11307	11289	11277	11281	11293	11269	11317	11257	11317
92,6	**0,926**	10763,8	0,54201	19371,35034	10807	10729	10773	10733	10789	10757	10741	10715	10837	10757
100,9	**1,009**	10183,3	0,51278	18137,15751	10189	10201	10197	10176	10137	10205	10169	10145	10189	10225
127,3	**1,273**	9069,8	0,45671	15839,79937	9111	9127	9093	9099	9076	9034	9031	9059	9017	9051
188,5	**1,885**	7608,7	0,38313	12957,44893	7616	7605	7608	7616	7636	7632	7638	7568	7616	7552
211,0	**2,1**	7254,5	0,36530	12280,18070	7281	7321	7230	7248	7268	7254	7244	7253	7223	7223
268,0	**2,7**	6353,7	0,31994	10593,79691	6335	6326	6324	6376	6376	6340	6358	6408	6384	6310
376,5	**3,765**	5366,3	0,27022	8802,54318	5380	5388	5474	5409	5226	5360	5450	5327	5317	5332
729,0	**7,3**	3808,3	0,19177	6091,22715	3714	3814	3815	3822	3833	3818	3825	3818	3805	3819
1060	**10,6**	2890,4	0,14555	4556,19045	2914	2894	2894	2882	2902	2884	2886	2862	2878	2908
1394	**13,94**	2456,8	0,12371	3846,40894	2486	2482	2452	2482	2455	2467	2438	2458	2434	2414
1862	**18,62**	1837,6	0,09253	2849,35610	1830	1844	1848	1852	1832	1840	1862	1818	1829	1821

Příloha E

Obsah elektronické přílohy

- firmware – složka obsahující implementaci firmware modulů
- backend – složka obsahující implementaci webové aplikace
- PCB – soubory pro výrobu desek plošných spojů
- pdf – Technické zprávy ve vysokém rozlišení
- latex – Zdrojové soubory technické zprávy
- generator.js – generátor maker pro obsluhu GPIO mikrokontroléru MKV11
- bom.xls – seznam komponent potřebných pro osázení jednotlivých modulů