Česká zemědělská univerzita v Praze

Technická fakulta

Katedra elektrotechniky a automatizace

Diplomová práce

Inovace úlohy z předmětu Automatizace s využitím Arduina

Bc. Miroslav Pecha

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Technická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Miroslav Pecha

Informační a řídicí technika v agropotravinářském komplexu

Název práce

Inovace úlohy z předmětu Automatizace s využitím Arduina

Název anglicky

Innovation of the task of the subject Automation with the use of Arduino

---

Cíle práce

Cílem diplomové práce je vytvoření virtuální laboratoře s programem Scilab a propojení s platformou Arduino. V rámci diplomové práce dojde k inovaci/vytvoření úlohy pro výuku předmětu Automatizace.

Metodika

Rešerše hardwarových a softwarových řešení.

Návrh několika variant provedení úloh.

Realizace výukové úlohy.

Ovládání přes webové rozhraní.

Tvorba uživatelského rozhraní pro vybranou úlohu.

Tvorba rozhraní pro vzdálený přístup k úloze.

Vytvoření návodů a protokolů pro měření.

Doporučený rozsah práce

50-60 stran

Klíčová slova

PLC, vizualizace, realizace

Doporučené zdroje informací

Automatizace : časopis pro automatizaci, měření a inženýrskou informatiku. Praha: ISBN 0005-125. BALÁTĚ, J. *Automatické řízení.* Praha: BEN – technická literatura, 2004. ISBN 80-7300-148-9.

BENEŠ, P. *Automatizace a automatizační technika. 3, Prostředky automatizační techniky.* Brno: Computer Press, 2003. ISBN 80-7226-239-4.

LACKO, B. *Automatizace a automatizační technika. 1, Systémové pojetí automatizace.* Praha: Computer Press, 2000. ISBN 80-7226-246-7.

MORRISS, S B. *Automated manufacturing systems : actuators, controls, sensors, and robotics.* New York: Glencoe, 1995. ISBN 0028023315.

ŠVARC, I. – VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. FAKULTA STROJNÍHO INŽENÝRSTVÍ. *Automatizace : automatické řízení.* Brno: Akademické nakladatelství CERM, 2005. ISBN 80-214-2943-7.

VORÁČEK, R. *Automatizace a automatizační technika. 2, Automatické řízení.* Praha: Computer Press, 2000. ISBN 80-7226-247-5.

Předběžný termín obhajoby

2022/2023 LS – TF

Vedoucí práce

Ing. Monika Hromasová, Ph.D.

Garantující pracoviště

Katedra elektrotechniky a automatizace

Elektronicky schváleno dne 28. 1. 2022

doc. Ing. Miloslav Linda, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 2. 2022

doc. Ing. Jiří Mašek, Ph.D.

Děkan

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci na téma: Inovace úlohy předmětu Automatizace s využitím Arduina vypracoval samostatně a použil jen pramenů, které cituji a uvádím v seznamu použitých zdrojů. Jsem si vědom, že odevzdáním diplomové práce souhlasím s jejím zveřejněním dle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů, ve znění pozdějších předpisů, a to i bez ohledu na výsledek její obhajoby. Jsem si vědom, že moje diplomová práce bude uložena v elektronické podobě v univerzitní databázi a bude veřejně přístupná k nahlédnutí. Jsem si vědom že, na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, ve znění pozdějších předpisů, především ustanovení § 35 odst. 3 tohoto zákona, tj. o užití tohoto díla.

V Praze dne 31.3. 2023 _____

Poděkování

Rád bych touto cestou poděkoval paní Ing. Monice Hromasové, Ph.D. za odborné vedení a podnětné připomínky, které mi pomohly při psaní této diplomové práce.

Inovace úlohy předmětu Automatizace s využitím Arduina

Abstrakt

Diplomová práce se nejdříve věnuje teoretické rešerši použitých hardwarových a softwarových řešení pro realizaci inovace výukové úlohy pro předmět automatizace. Následně je v práci popsán výchozí stav úlohy s návrhy na její inovaci v oblasti použitých součástek a konstrukce. Poté jsou popsány tři návrhy úloh, včetně jejich principů, způsobu realizace a možného výstupu ze strany studenta. V praktické části se práce věnuje samotné realizaci jednoho z popsaných návrhů. V realizaci je nejdříve popsaná inovace v konstrukci úlohy a poté její samotné zapojení. Následně je popsán způsob regulace kyvadla, včetně teoretických základu pro pochopení vybrané regulace. Další část práce se věnuje popisu komunikace úlohy s uživatelským rozhraním na webové stránce pomocí sériové linky a způsobu zpracování výsledných naměřených dat. Poslední část práce se zabývá vytvořením samotného uživatelského rozhraní včetně platformy pro vzdálený přístup k úloze.

Klíčová slova: Arduino, PID, regulace, matematické kyvadlo, realizace, uživatelské rozhraní, vizualizace, Scilab

Innovation of the task of the subject Automation with the use of Arduino

Abstract

This thesis is firstly focused on the theoretical research of hardware and software solutions used to implement the innovation of a learning task for the subject of automation. After that, the thesis describes the initial state of the task with suggestions for its innovation in the area of used components and design. Then the three task designs are described, including their principles, implementation method and possible output from the student side. The practical part of the thesis deals with the actual implementation of one of the described designs. In the realization, first the innovation in the task design is described and then the actual wiring of the task. Afterwards, the pendulum control method is described, including the theoretical background for understanding the chosen control. The next part of the thesis is devoted to the description of the communication of the task with the user interface on a web page using a serial link and the processing of the resulting measured data. The last part of the thesis deals with the creation of the user interface itself, including the platform for remote access to the task.

Keywords: Arduino, PID, regulation, mathematical pendulum, realization, user interface, visualization, Scilab

Obsah

1 Úvod

Automatizace je nedílnou součástí dnešních trendů moderního průmyslu a jeho technologií. V podstatě se jedná o snahu nahradit lidskou práci automatizovanými systémy, které mají schopnost plnit stejné úkoly s vyšší rychlostí, přesností a nižšími náklady. Pomocí automatizace jsou firmy schopny dosáhnout zvýšení produktivity a zlepšení kvality výrobků a služeb. V dnešní době se již automatizace nenachází pouze v průmyslovém odvětví, ale i v oblastech jako je doprava, zdravotnictví a služby.[1]

Automatizace je úzce spojena s regulací, která určuje řízení akční veličiny soustavy. Regulace zajišťuje, že automatizovaný systém bude řízen s ohledem na stanovené parametry a předem definované cíle. V podstatě jakýkoliv automatizační systém v sobě obsahuje nějaké prvky regulace. Podle této skutečnosti se dá říct, že regulace je nedílnou součástí automatizace.

Tato práce se bude věnovat inovaci výukové úlohy pro předmět automatizace. Úloha bude spočívat v konfigurování parametrů pro regulaci kyvadla s elektrickým motorem. V důsledku toho budou studenti seznámení s praktickou ukázkou konfigurování parametrů regulace. K úloze bude také vytvořeno vzdálené rozhraní pro případ distanční výuky.

2 Cíl práce

Cílem této práce je inovace výukové úlohy pro předmět automatizace. Úloha se bude zabývat regulací motoru podle vypočítané dráhy matematického kyvadla. Jako řídící prvek bude použita platforma Arduino. Uživatelské rozhraní bude vytvořeno formou webové stránky, kterou bude student úlohu ovládat. Ovládání bude probíhat volbou počátečních parametrů a spouštěním úlohy. Výsledná data budou poté zpracována v programu Scilab, který bude porovnávat naměřený průběh kyvadla s průběhem teoretickým. K úloze bude také vytvořeno spojení pro vzdálený přístup pro případnou distanční výuku. Dále bude úloha obsahovat zadání k měření, včetně návodu. Cílem úlohy by mělo být seznámí studentů s praktickou ukázkou způsobů regulace a vyzkoušení si jejího ladění.

3 Metodika práce

Začátek práce se bude zabývat teoretickou rešerší použitého hardwaru a softwaru pro ovládání úlohy. Poté se bude práce věnovat samotnému návrhu úlohy, kdy jako první bude popsán výchozí stav úlohy ve stavu předání.

Následovat pak bude návrh inovací pro tuto úlohu, které se budou věnovat vybrání vhodných součástek a návrhů na vylepšení konstrukce. Po návrhu inovací budou popsány tři návrhy úloh, kde u každého bude popsán její princip, realizace a předpokládaný výstup ze stran studentů.

Další kapitola práce se bude věnovat již samotné realizaci úlohy. Nejdříve bude popsán výběr navrhnuté úlohy a poté již samotná realizace dílčích postupů. Na konci této úlohy bude napsáno zhodnocení realizace, včetně ukázky naměřených průběhů regulace.

4 Použitý Hardware

První hardwarovou část této úlohy bude tvořit mikrokontroler Arduino. Ten bude mít za úkol ovládání úlohy přes jeho I/O porty a komunikaci s uživatelským rozhraním. Dalším použitým hardwarem bude router Mikrotik, který bude umožňovat po určitém nastavení vzdálený přístup k této úloze. Ostatní hardware, který bude přímo součástí úlohy, bude popsán až v praktické části.

4.1 Arduino

Arduino je open-source vývojová deska, která umožňuje lidem bez předchozích znalostí programování a elektroniky snadno vytvářet interaktivní projekty. Vytvoření desky Arduino bylo inspirováno potřebou vytvořit levnou a snadno ovladatelnou desku pro vzdělávací účely, ale rychle se rozšířila do mnoha oblastí včetně průmyslu, umění a zábavy.

4.1.1 Základní popis

Arduino je malá vývojová deska s procesorem od firmy Atmel s univerzálním konektorem sériové sběrnice (USB) pro připojení k počítači a s řadou konektorů, které lze připojit k externím zařízením, jako jsou motory, relé, světelné senzory, laserové diody, reproduktory, mikrofony, a mnohé další. Arduino deska může být napájena prostřednictvím připojeného USB z počítače, z 9V baterie nebo z napájecího zdroje zapojeného do elektrické sítě. Arduino lze ovládat přímo z počítače, nebo může být prostřednictvím počítače pouze nahrán program, který se poté bude cyklicky opakovat.[2]

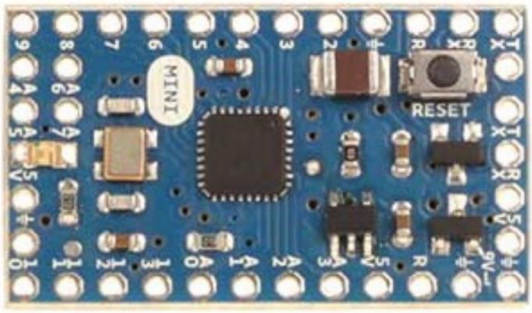
Konstrukce desky je open-source, a to znamená že kdokoli může vyrábět desky kompatibilní s Arduinem. Díky open-source řešení vznikla velká konkurence mezi výrobci desek, která nakonec vedla k nízkým pořizovacím nákladům. Základní Arduino desky mohou být také doplněny o doplňkové moduly, tzv. shieldy, které lze přímo připojit na desku Arduino. Tyto přídavné moduly dokážou desku Arduino rozšířit o nové funkce nebo vlastnosti, jako je např. ethernet port, přednastavený displej a další.[3]

4.1.2 Typy desek

Existuje mnoho různých typů desek od firmy Arduino, zahrnující základní modely, výkonnější verze a specializované desky pro různé aplikace. Na dalších odstavcích se nachází krátký popis několika základních desek Arduino:

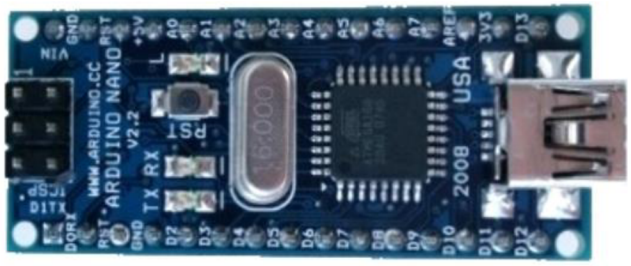
Arduino Mini: Nejmenší oficiální verze Arduino desky, která byla navržena hlavně kvůli úspoře místa. Díky opravdu malým rozměrům (33x18 mm) má tato deska absenci klasického USB portu používaného k naprogramování této desky. Tuto absenci lze vyřešit použitím externího USB 2 Seriál převodníku. I přes malé rozměry tento typ desky ale ve výkonu nijak nezaostává před většími typy. Je osazen procesorem ATmega328P s taktem 16 MHz. Jeho malými rozměry je vhodný hlavně k použití například v chytrých vypínačích, dálkových ovladačích a všude jinde, kde se klade důraz na úsporu místa.[4]

Deska obsahuje 14 digitálních vstupů a výstupů, přičemž 6 z nich podporuje PWM. Dále deska obsahuje 8 analogových vstupů.[5]



Obrázek 4.1 - Arduino Mini [4]

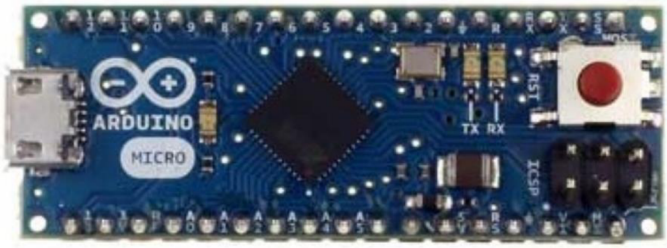
Arduino Nano: Tento typ desky se oproti minulému typu tolik neodlišuje. Největším rozdílem je USB port s převodníkem, který je na této desce osazen. Deska má jinak stejný procesor ATmega328P i stejný počet a typ vstupních a výstupních portů. Jako největší rozdíl se tak může jevit pouze jeho o 10 mm delší rozměr.[4]



Obrázek 4.2 - Arduino Nano [4]

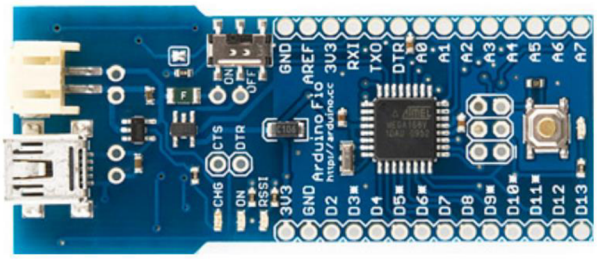
Arduino Micro: Tato deska obsahuje čip ATmega32u4, který je vybaven převodníkem. Díky tomu se po připojení do počítače skrz USB, může deska tvářit jako myš nebo klávesnice. Tato skutečnost umožňuje pomocí této desky posílat do počítače příkazy jako jsou stisk klávesy, nebo posun myši. Tyto příkazy lze sice posílat i s ostatními deskami, ale je u toho potřeba nelehké přeprogramování převodníku (nejčastěji u desek s čipem ATmega16u2, nebo ATmega8u2). Tato deska se tedy hlavně hodí pro vytvoření náhražek klávesnice nebo myši.[4]

Na desce lze najít 12 digitálních vstupů a výstupů, přičemž 5 z nich podporuje PWM. Dále deska obsahuje 4 analogové vstupy.[5]



Obrázek 4.3 - Arduino Micro [4]

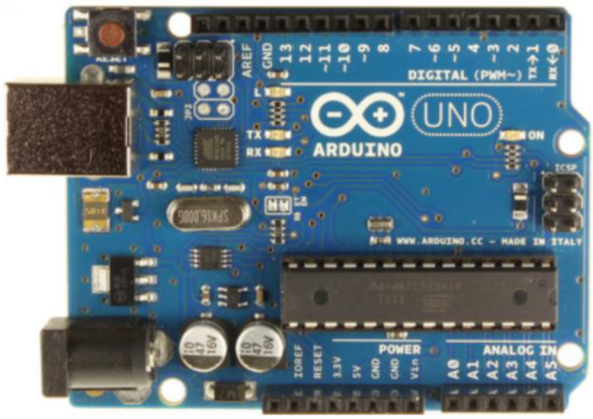
Arduino Fio: Deska Arduino Fio je speciálně uzpůsobena k připojení různých bezdrátových modulů (např. XBee modul). Takové moduly se využívají hlavně v internetu věcí. Deska je osazena čipem ATmega328P a výstupní napětí je sníženo z klasických 5V na 3,3V, právě kvůli kompatibilitě s již zmíněnými moduly.[4]



Obrázek 4.4 - Arduino Fio [4]

Arduino Uno: Jako jednou z nejpoužívanějších desek, je deska Arduino Uno. Deska je osazena čipem ATmega328P a obsahuje jak klasické USB na připojení k počítači, tak i napájecí jack konektor, kterým lze s pomocí vhodného zdroje desku napájet z elektrické sítě. Tato deska je přímým nástupcem hlavní vývojové linie, která začala prvním Arduinem se sériovým portem místo klasického USB. Z této linie si vyvinuly i další dvě speciální desky. První z nich je deska Arduino Ethernet, která obsahuje místo USB portu ethernet, přičemž má stejnou výbavu jako Arduino Uno. Druhou zmíněnou deskou je Arduino Bluetooth, kde zase naopak místo USB najdeme rozhraní bluetooth pro bezdrátovou komunikaci.[4]

Deska obsahuje 14 digitálních vstupů a výstupů, přičemž 6 z nich podporuje PWM. Dále deska obsahuje 8 analogových vstupů.[5]

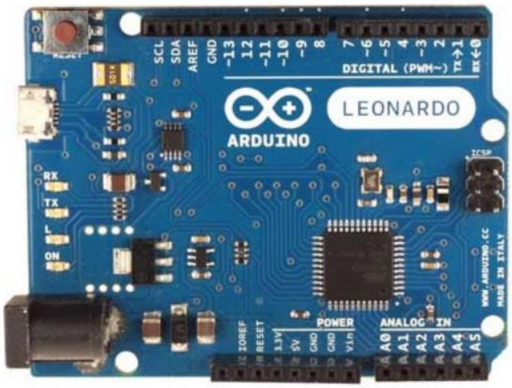


Obrázek 4.5 - Arduino Uno [4]

Arduino Leonardo: Tato deska se designově podobá desce Uno a funkčně desce Micro. Je osazena čipem ATmega32u4, takže má stejné vlastnosti jako již popsaná deska Arduino

Micro. Jediné hlavní rozdíly mezi těmito deskami jsou v jejich velikosti, počtu vstupních a výstupních pinů a přítomností napájecího jack konektoru.[4]

Konkrétně tato deska disponuje dvaceti digitálními vstupy a výstupy, přičemž 7 z nich podporuje PWM. Dále deska obsahuje 12 analogových vstupů.[5]



Obrázek 4.6 - Arduino Leonardo [4]

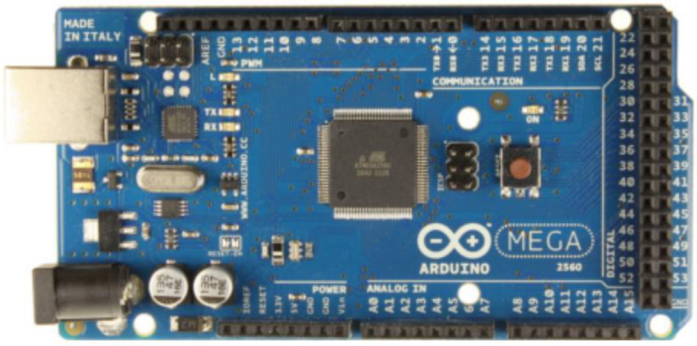
Arduino Yún: Tento model desky se oproti ostatním výrazně odlišuje tím, že obsahuje dva odlišné čipy. Prvním z nich je klasický ATmega32u4, na kterém funguje jádro Arduina. Druhý čip je Atheros AR9331, na kterém je možné spustit odlehčený linux Linimo. Deska také obsahuje softwarový bridge, který zajišťuje komunikaci mezi oběma čipy. Díky přítomnosti dvou čipů je deska osazena microUSB portem pro programování Arduina a normálním USB a Ethernet portem pro potřeby linuxu. Ve finále představuje dosti výkonnou desku s kompaktními rozměry, ale ne již tak nízkou cenou jako je tomu u předchozích typů desek.[4]

Deska obsahuje 20 digitálních vstupů a výstupů, přičemž 7 z nich podporuje PWM. Dále deska obsahuje 12 analogových vstupů.[5]

Obrázek 4.7 – Arduino Yún [4]

Arduino Mega2560: Tento typ desky vznikl prodloužením desky Arduino Uno. Díky větším rozměrům zde může být osazen výkonnější čip ATmega2560 a je zde i prostor pro více digitálních a analogových vstupů a výstupů. Tato deska se hodí hlavně tam, kde je zapotřebí většího výpočetního výkonu, nebo mnoha vstupních a výstupních pinů. Deska dále obsahuje klasický USB port s napájecím jack portem, stejně jako je tomu u Arduino Uno.[4]

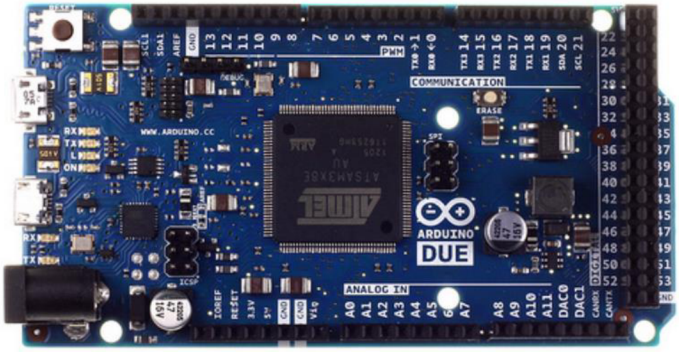
Deska obsahuje 54 digitálních vstupů a výstupů, přičemž 14 z nich podporuje PWM. Dále deska obsahuje 16 analogových vstupů.[5]



Obrázek 4.8 - Arduino Mega2560 [4]

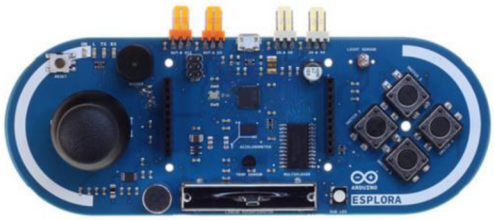
Arduino Due: Tato deska je pomyslným nástupcem předchozí desky Mega2560. Obsahuje daleko výkonnější čip Atmel SAM3X8E, který má taktovací frekvenci 84MHz a 32-bitové jádro. Oproti předchůdcům s 8-bitovými deskami a taktovací frekvencí 16MHz je tento výkonový skok opravdu znatelný. Dále deska obsahuje dva micro USB konektory, přičemž jeden se používá na programování samotné desky a druhý pro připojení periferii jako je např. myš nebo klávesnice. Nakonec deska opět obsahuje napájecí jack konektor.[4]

Deska obsahuje 54 digitálních vstupů a výstupů, přičemž 12 z nich podporuje PWM. Dále deska obsahuje 12 analogových vstupů.[5]



Obrázek 4.9 - Arduino Due [4]

Arduino Esplora: Již na první pohled je zřejmé, že tato deska je svým designem dosti specifická. Tento typ se díky přítomnosti joysticku, tlačítek a posuvného potenciometru používá převážně na vytvoření si vlastního ovladače. Deska dále obsahuje i piezoelektrický bzučák, teploměr, tříosý akcelerometr a piny pro připojení externího LCD displeje. Čipem v této desce je ATmega32u4.[4]



Obrázek 4.10 - Arduino Esplora [4]

Typů desek pro Arduino existuje mnoho a díky krátkému popisu několika z nich, si lze všimnout že existují typy které jsou již z výroby předurčeny na nějaký druh úlohy. Existují ale i více univerzální jako je např. deska Arduino Uno, která bude v této práci použita.

4.2 Mikrotik

Mikrotik je značka síťových zařízení, která se specializuje na výrobu routerů, switchů a bezdrátových síťových zařízení. Tyto zařízení jsou určena pro použití v malých a středních sítích, ale lze je použít i ve větších sítích s desítkami nebo stovkami uživatelů.

4.2.1 Základní popis

Většina zařízení Mikrotik funguje na operačním systému RouterOS, který je založen na jádře Linuxu. RouterOS nabízí řadu pokročilých funkcí, jako jsou firewall, VPN, hotspot, QoS, VLAN a další. Tyto funkce umožňují uživatelům vytvářet a spravovat velké a pokročilé sítě. Mikrotik také nabízí nástroje pro správu a monitorování sítě, jako jsou grafy a statistiky přenosového pásma, systémové logy, upozornění na chyby a další. Zařízení Mikrotik lze označit za spolehlivé a výkonné síťové zařízení, které může být využito v různých typech sítí a poskytuje mnoho pokročilých funkcí pro správu a monitorování sítě.[6]

Zařízení od firmy Mikrotik jsou vzhledem k počtu jejich funkcí cenově dostupná zařízení v porovnání s ostatní konkurencí, která nabízí podobné možnosti nastavení. Vzhledem k možnosti pokročilého nastavení mnoha funkcí nebývají tyto routery doporučovány pro domácí použití pro uživatele bez základních znalostí nastavování síťových zařízení.

4.2.2 Mikrotik hEX S

Pro nasazení na této úloze byl vybrán router hEX S od firmy Mikrotik. Pro potřeby úlohy by ale postačoval jakýkoliv router od této společnosti, který má dostatečný počet výstupních ethernet portů.

Mirkotik hEX S (obr. 4.11) je pětiportový gigabitový ethernetový router pro místa, kde není vyžadováno bezdrátové připojení (neobsahuje bezdrátové rozhraní Wi-Fi). Tento router je cenově dostupný, malý a snadno použitelný, ale zároveň je vybaven velmi výkonným dvoujádrovým procesorem s frekvencí 880 MHz a 256 MB paměti RAM, který je schopen všech pokročilých konfigurací, jež systém RouterOS podporuje. Zařízení je vybaveno rozhraním USB 2.0, výstupem PoE (napájení přes ethernet) pro ethernetový port č. 5 a 1,25 Gbit/s šachtou SFP. Port č. 5 může napájet další pasivní zařízení podporující PoE stejným napětím, jaké je přivedeno na jednotku. Zároveň je i možnost napájení samotného routeru pomocí PoE místo klasického napájecí jacku a adaptéru do elektrické sítě.[7]

Obrázek 4.11 - Mikrotik hEX S [7]

5 Použitý software

V této práci se bude pracovat s několika softwary. Jako prvním z nich je program Scilab, který bude zpracovávat naměřená data z úlohy. Další software bude webový server XAMPP, který bude sloužit jako uživatelské rozhraní pro vytvořenou úlohu. Dalším softwarem je prostředí Arduino IDE pro naprogramování Arduina a aplikace WinBox pro spravování přístupu na vzdálený přístup k úloze pomocí VPN.

V následujících podkapitolách jsou tyto softwary stručně popsány.

5.1 Scilab

Scilab je software s otevřeným zdrojovým kódem a je podporován na operačních systémech Linux, BSD, MS Windows a macOS.[8]

5.1.1 Seznámení

Scilab je interaktivní softwarový systém vyvinutý pro numerické výpočty a vykreslování 2D a 3D grafů. Je určen zejména pro maticové výpočty: řešení úloh typu soustav lineárních rovnic, provádění maticových transformací, faktorizace matic, a podobně. Vývojáři Scilabu vytvořili knihovny velkého množství vestavěných matematických funkcí. V průběhu toho vývojáři doplnili Scilab o širokou škálu balíků vestavěných programů, kterým se říká toolboxy.[9]

Scilab je vyvinut tak, aby fungoval také jako programovací jazyk ve smyslu, že uživatelé mohou kódovat své programy stejně jako v jakémkoli jiném programovacím jazyce C, C++ atd. Kromě toho má řadu grafických možností a může být rozšířen prostřednictvím programů napsaných v jeho vlastním programovacím jazyce. Díky této vlastnosti je Scilab uživatelsky přívětivý interaktivní software.[8]

Stejně jako v jeho komerčním protějšku MATLAB, jsou také ve Scilabu všechny typy proměnných, a to reálné, komplexní, logické, celočíselné, řetězcové a polynomické proměnné považované za matice. Další významnou vlastností Scilabu je, že rozumí tzv. rozdílu mezi reálnými čísly a čistě reálnými komplexními čísly.[8]

Výhody programu Scilab:[8]

- Zjednodušuje analýzu matematických modelů.
- Programování ve vyšším programovacím jazyku (je potřeba méně času při psaní kódu, ale za cenu nižší rychlosti výpočtu)
- Poskytuje rozšiřitelné programovací/vizualizační prostředí.
- Poskytuje profesionálně vypadající grafy

Nevýhody programu Scilab:

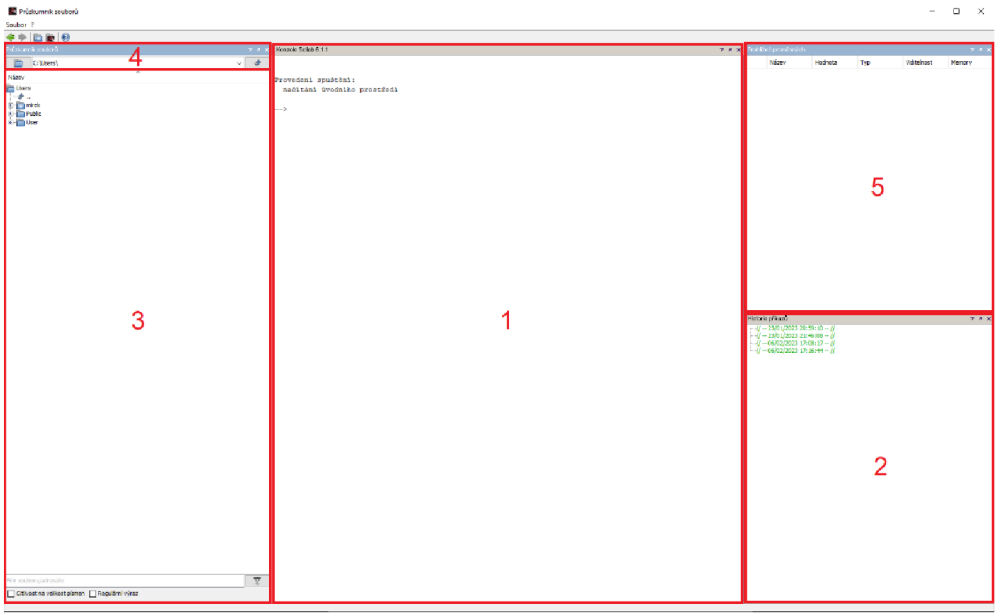
Jediná nevýhoda programu Scilab oproti výpočetnímu programování na nižších programovacích jazycích je, že jako interpretovaný (tj. nikoliv předkompilovaný) jazyk se může ukázat jako pomalý při náročných výpočtech.[8]

5.1.2 Základní elementy programu

Při spuštění programu Scilab se objeví základní okno s výchozím rozložením pracovní plochy tohoto programu viz obr. č. 5.1. Je to sada nástrojů pro správu souborů, proměnných a aplikací spojených se Scilabem. Níže budou představeny jejich základní elementy.

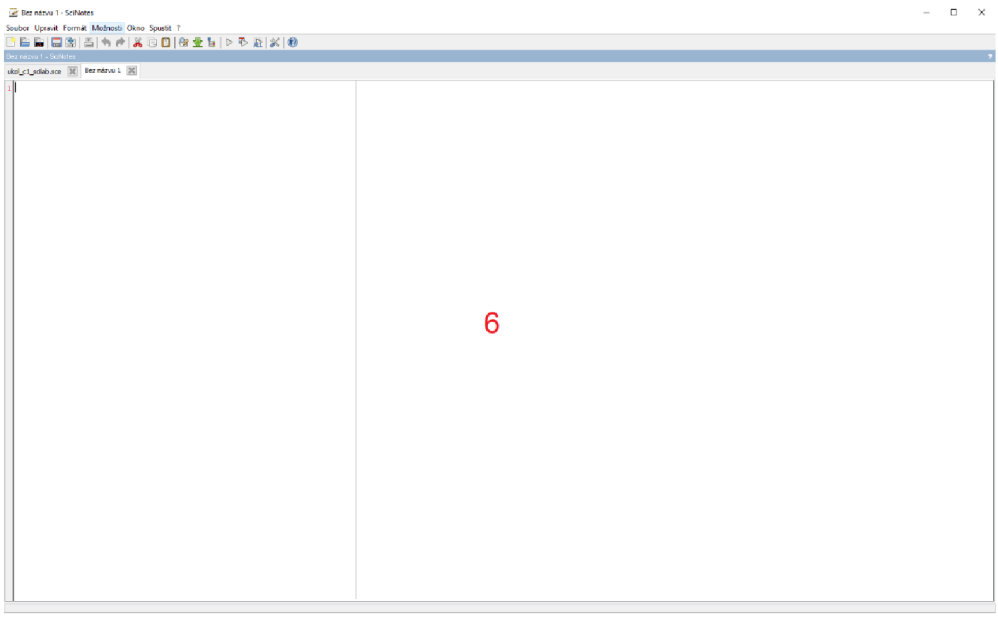
1. **Konzole** – Je příkazové okno, které slouží k textovému zadávání funkcí Scilabu, nebo dalších příkazů jako např. spuštění skriptu nebo různých výpočtů.
2. **Historie příkazů** – Ukládá se do něj historie spuštěných příkazů nebo funkcí. Dá se využít k zpětnému kontrolování provedených funkcí, nebo znovuspuštění již spuštěných funkcí.
3. **Okno pracovního prostoru** – Ukazuje složky a soubory v aktuálním adresáři, který je takto nastaven na pracovní. Znamená to, že jakékoliv spuštěné funkce budou mít tento adresář jako výchozí.
4. **Aktuální adresář** – Ukazuje celou cestu k aktuálnímu adresáři, ve kterém se pracuje. Dá se takto otevřít konkrétní složka s celou cestou, nebo se může použít okno pracovního prostoru a proklikat se k ní. Když se Scilab znova zapne, ukazuje vždy na stejnou složku, na kterou ukazoval před jeho vypnutím. Pokud se otevře již uložený skript, Scilab automaticky nastaví aktuální adresář na složku, ve které se tento skript nachází.

5. **Prohlížeč proměnných** – Ukazuje seznam proměnných, které jsou aktuálně používané.



Obrázek 5.1 - Úvodní okno programu Scilab

6. **Editor SciNotes** – Tento editor (obr. 5.2) slouží k psaní skriptů, které se dají poté uložit a znovu celé spustit. Editor SciNotes se dá otevřít v horní liště programu pod tlačítkem aplikace. Z tohoto editoru se dá skript i rovnou spustit. Uložený skript se poté ukládá do souboru s koncovkou .sce.



Obrázek 5.2 - Editor SciNotes

5.2 Webový server

Webový server bude mít v této úloze za úkol možnost vytvořit ovládání přes webové rozhraní formou uživatelského rozhraní. Pro webový server byla vybraná platforma XAMPP. K samotné komunikaci mezi vytvořeným webem a Arduinem bude sloužit protokol pro sériovou komunikaci po USB rozhraní.

XAMPP je multiplatformní webový server, který je zdarma a má otevřený zdrojový kód. XAMPP je zkratka pro Cross-Platform, Apache, MySQL, PHP a Perl. XAMPP je populární multiplatformní webový server, který umožňuje programátorům psát a testovat svůj kód na místním webovém serveru. Byl vytvořen společností Apache Friends a veřejnost může revidovat nebo upravovat jeho nativní zdrojový kód. Jeho součástí je mimo jiné databáze MariaDB, server Apache HTTP Server a interprety programovacích jazyků PHP a Perl. Díky jednoduchosti nasazení XAMPP, ho může vývojář snadno a rychle nainstalovat do operačního systému, přičemž další výhodou je, že lze načíst i běžné doplňkové aplikace, jako jsou WordPress a Joomla.[10]

V realizované úloze se bude využívat pouze samotný Apache HTTP server, na kterém bude napsaná webová stránka prostřednictvím HTML, CSS a PHP.

Mezi jeho hlavní výhody se řadí jednoduché nastavení a jeho multiplatformní možnost nasazení (funguje jak na Windows, tak i na Linuxu).[11]

5.3 Vzdálený přístup

Pro možnost vzdáleného přístupu k úloze bude využit VPN tunel, který bude vytočen ze vzdáleného počítače na router Mikrotik, který bude tvořit lokální síť s výukovou úlohou. K vytočení takového tunelu je předpoklad veřejné IP adresy na routeru, na kterém bude vytvořen VPN server.

Konfigurace routeru bude probíhat pomocí speciálního program WinBox, který firma Mikrotik vyvinula přímo ke správě jejich zařízeních. V tomto programu lze i přímo přidělovat přihlašovací údaje k samotnému VPN serveru a tím i tedy spravovat, kdo může být k dané úloze v jeden čas připojený, aby se eliminovala možnost kolize více studentů.

I když lze mikrotik router spravovat i přes klasické webové rozhraní, využití programu WinBox je jak uživatelsky přívětivější, tak i časově méně náročné.

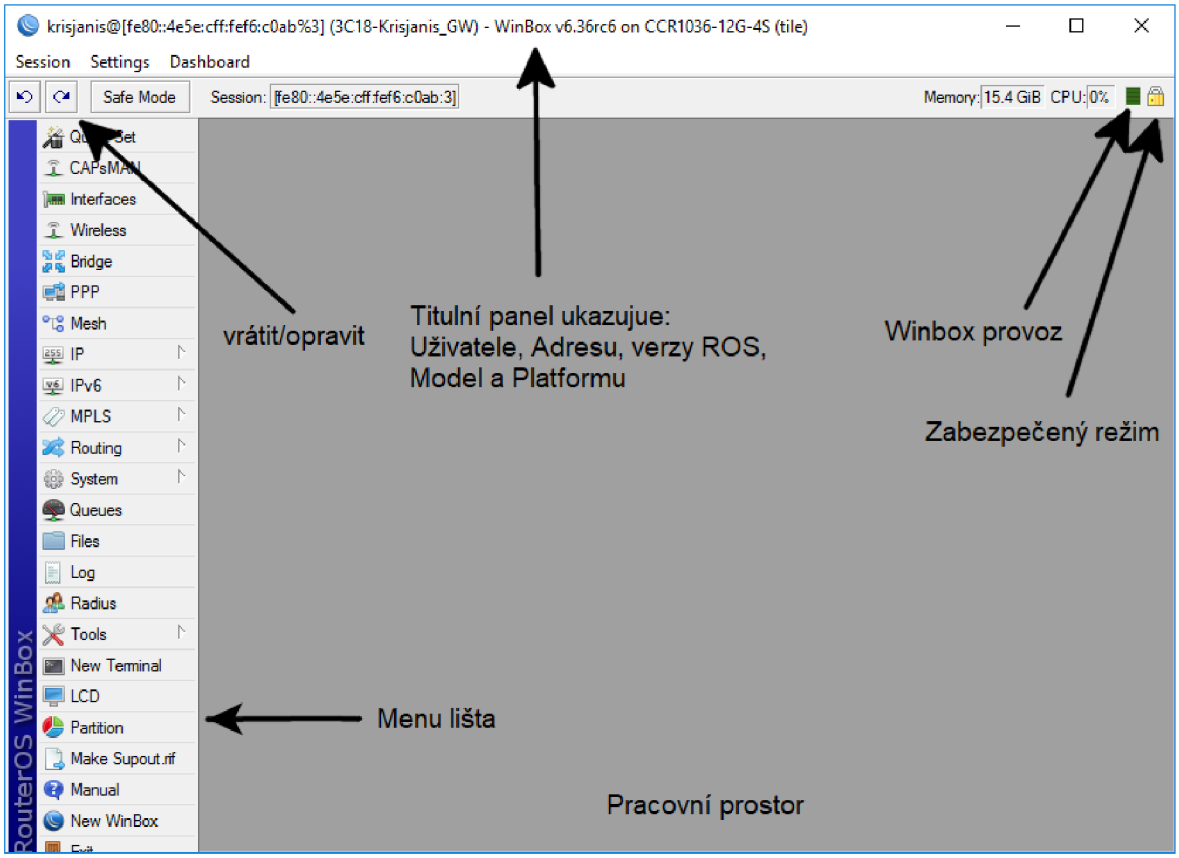
5.3.1 VPN

VPN je zkratka pro virtuální privátní síť. Je to technologie, která vytváří bezpečné a šifrované spojení skrze potenciálně nebezpečný internet. Virtuální privátní síť je způsob, jak rozšířit soukromou lokální síť pomocí veřejné sítě, většinou internetu. Název pouze naznačuje, že se jedná o virtuální "privátní síť", tj. uživatel může být součástí místní lokální sítě vytvořené na vzdáleném místě. K vytvoření bezpečného připojení využívá tunelovací protokoly.[12]

K navázání VPN tunelu je potřeba VPN server nacházející se na vzdálené sítí a VPN klienta na straně uživatele. Takový klient je například přímo obsažen v systému Windows a není tedy potřeba na straně uživatele instalace programu třetí strany.

5.3.2 WinBox

WinBox je program, který umožňuje správu systému Mikrotik RouterOS pomocí rychlého a jednoduchého uživatelského grafického rozhraní (obr. 5.3). Jedná se o nativní binární soubor Win32, ale lze jej spustit i v systémech Linux a MacOS pomocí Wine. Všechny funkce rozhraní WinBox fungují stejně jako funkce klasické konzole. Některé pokročilé a pro systém kritické konfigurace nejsou z WinBox možné, jako například změna MAC adresy.[13]

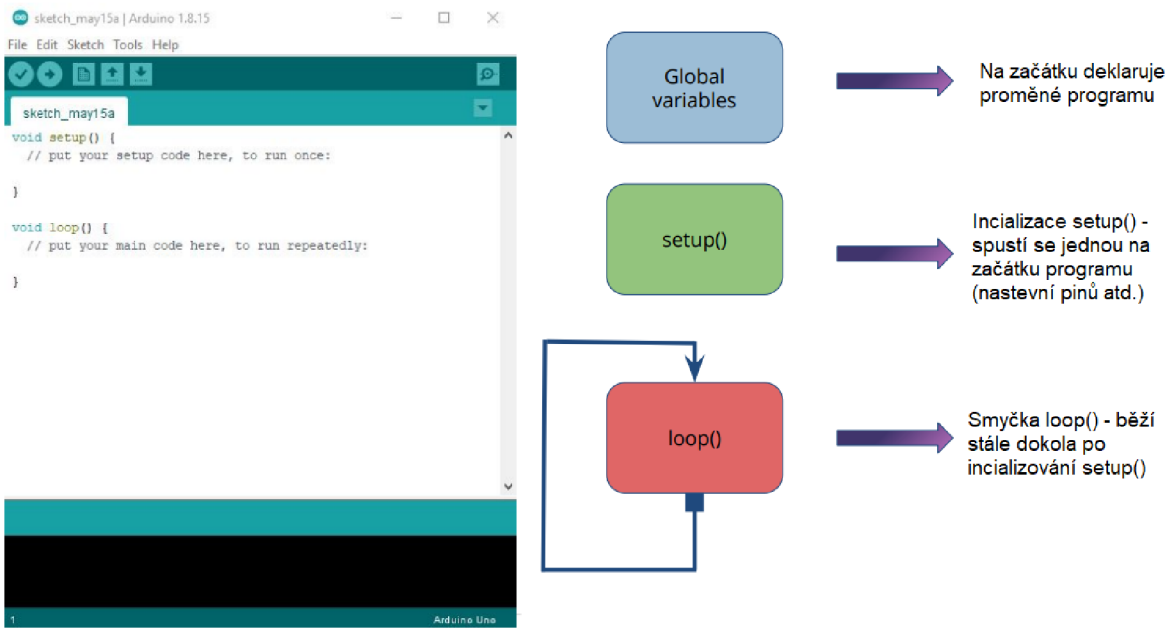


Obrázek 5.3 - Ukázka rozhraní WinBox [13]

5.4 Arduino IDE

IDE je zkratka pro Integrated Development Environment a jedná se o oficiální software (obr. 5.4), který zavedla společnost Arduino.cc. Tento software slouží především k úpravám, kompilaci a nahrávání kódu do zařízení Arduino. Téměř všechny typy Arduino modulů jsou kompatibilní s tímto softwarem, který má otevřený zdrojový kód a je snadno dostupný a zdarma.[14]

Obsahuje textový editor pro psaní kódu, oblast pro zprávy, textovou konzoli, panel nástrojů s tlačítky pro běžné funkce a řadu nabídek. Propojuje počítač s hardwarem Arduino a umožňuje nahrávat programy a komunikovat s ním. Zároveň podporuje programovací jazyky C a C++.[15]



Obrázek 5.4 - Ukázka programu Arduino IDE s náčrtem funkčnosti [15]

6 Návrh úlohy

V této kapitole bude nejdříve představena úloha v počátečním stavu. Budou zde popsány jak elektronické prvky, tak i její princip s fyzickým sestavením. Dále budou představeny návrhy na inovaci této úlohy a v poslední části budou představeny tři návrhy na vytvoření této úlohy, ze kterých se poté vybere jeden k realizaci.

6.1 Výchozí stav úloh

Úloha byla převzata v rozpracovaném stavu (obr. 6.1), přičemž základní princip této úlohy mělo být kyvadlo s elektrickým motorem a enkodérem na snímaní otáček. V dalších podkapitolách bude více rozebrána montáž této úlohy a její použitý hardware.

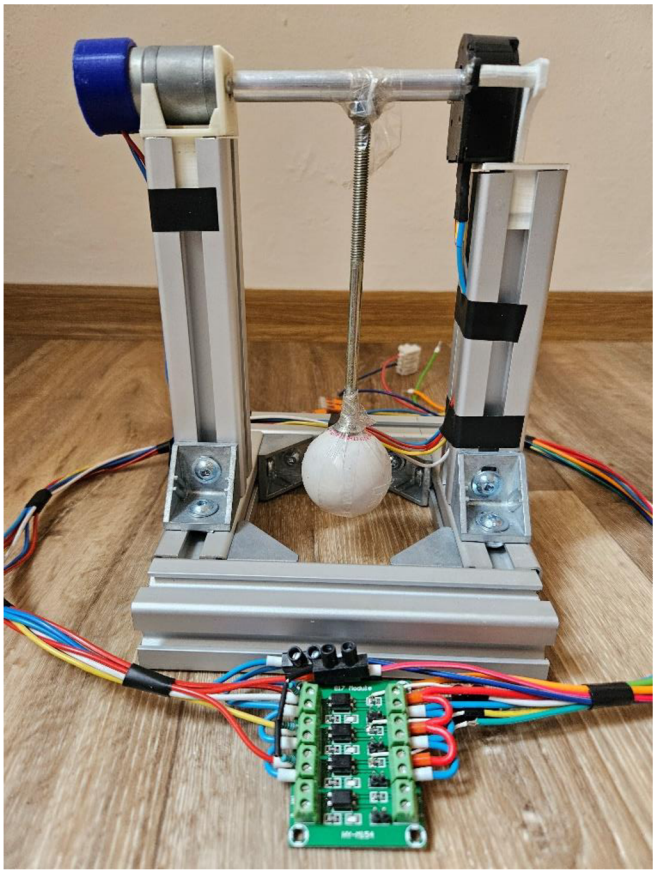
6.1.1 Popis konstrukce

Samotná úloha bez řídící elektroniky má rozměry 180x150x210 mm (hloubka, šířka, výška). Základní konstrukce celé úlohy je tvořena pomocí hliníkových profilů o rozměrech 30x30 mm. Tyto profily jsou k sobě spojeny pomocí speciálních úhelníků vytvořených přímo pro tyto profily. Úhelníky jsou poté spojeny s profilem přes speciální matice a klasické šrouby. Do profilů tedy není potřeba vrtat, ale pouze stačí vložit matku, která se zajistí o drážku v profilu a do ní je poté přes úhelník zašroubován šroub.

Na konstrukci je použito dohromady 6 hliníkových profilů. Čtyři z nich vytvářejí čtvercovou základnu a zbylé dva jsou poté umístěny kolmo na tuto základnu jako vztyčné pilíře, na kterých je umístěn motor s enkodérem. Motor s enkodérem jsou umístěny na speciálních držácích, které byly vytisknuty na 3D tiskárně. Tyto držáky mají ve spodní části výběžky, které přímo zapadají do drážek v profilu a zajištují tak stabilní uchycení. Držáky jsou tedy pouze zasunuty do vztyčných profilů. Motor a enkodérem jsou k držáku přichyceni každý pomocí dvou šroubů.

Mezi motorem a enkodérem je umístěna hliníková tyč, která je zasunuta do hřídele motoru, a poté připevněna pomocí utahovacího šroubu na samotné tyči. Do enkodéru je tyč také zasunuta a utáhnuta pomocí zajišťovacího šroubu uvnitř enkodéru. V půlce tyče je poté připevněn šroub dlouhý 120 mm o průměru 6 mm, který má simulovat vlákno kyvadla. Připevnění tohoto šroubu k tyči je řešeno pomocí matky našroubované na konci šroubu,

která je poté spojena s tyčí pomocí průhledné lepící pásky. Na druhém konci šroubu se poté nachází míček na stolní tenis, který simuluje hmotný bod kyvadla. Ten je opět připevněn ke šroubu pomocí průhledné lepící pásky. Posledním detailem na konstrukci je uchycení kabelových vývodů z motoru a enkodéru pomocí černé elektrikářské pásky. Tam kde to je možné jsou tyto kabely vedené v drážce hliníkového profilu tak, že nevyčnívají ze samotné konstrukce.



Obrázek 6.1 - Úloha ve výchozím stavu

6.1.2 Použitý hardware

Úloha ve výchozím stavu byla osazena motorem, enkodérem a izolačním modulem. V této podkapitole bude krátký popis samotných senzorů i konkrétního typu, který je použit.

Motor – V úloze je použit elektrický stejnosměrný motor s převodovkou typu GM25-300CHV-130-R (obr. 6.2). Tento motor má průměr 24,4 mm a délku 37,8 mm. Výstupní hřídel motoru má zářez tvaru D a průměr 4 mm. Díky převodovce zabudované uvnitř obalu

má tento motor nižší otáčky, ale zase větší kroutící moment. Ostatní parametry motoru jsou popsány v tabulce 6.1[16]

Tabulka 6.1 - Parametry motoru GM25-300CHV-130-R [16]

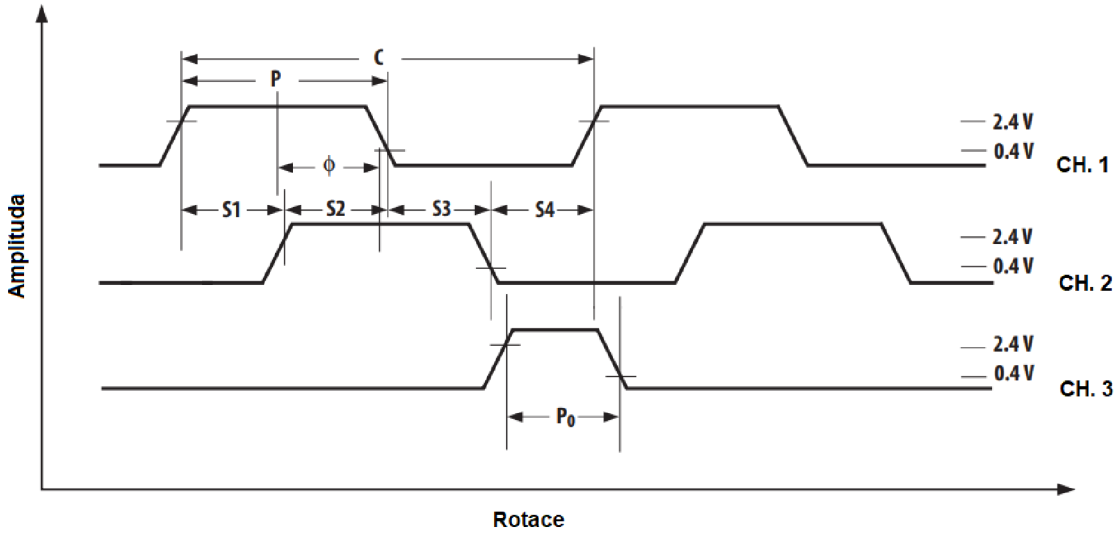
Převodový poměr	Napětí		Bez zatížení		Při maximální efektivitě					Při zastavení		
	Pracovní rozsah	Nominální	Rychlost	Proud	Rychlost	Proud	Točivý moment		Výkon	Točivý moment		Proud
			ot/min	A	ot/min	A	g/cm	mN/m	W	g/cm	mN/m	A
02:43,1	4-6 V	6 V	65	0,05	47	0,18	600	59	0,29	2000	196	0,5

Obrázek 6.2 - motor GM25-300CHV-130-R [16]

Enkodér – Pro snímání otáček motoru je použit optický tří kanálový enkodér HEDS-5640-A13 (obr. 6.4). Ten funguje na principu světelných diod a fotodetektorů. Uvnitř enkodéru je kódové kolečko, které se otáčí s hřídelí a postupně zakrývá nebo odkrývá vyzařovací světlo ze světelných diod. Díky tomu můžeme na vstupu vidět měnící se signál jednotlivých fotodetektorů. První a druhý kanál jsou vůči sobě fázově posunuty o 90°. Sledováním prvních dvou kanálu lze zjistit rychlost a směr pohybu. Třetí kanál poté indikuje otočení o celou otáčku. Při správném nastavení lze tedy tímto kanálem vytvořit výchozí bod, anebo pouze zjišťovat kdy byla provedena celá otáčka. Průběhy enkodéru lze vidět na obrázku č. 6.3 a parametry senzoru v tabulce č. 6.2.[17]

Tabulka 6.2 - Parametry enkodéru HEDS-5640-A13 [17]

Operační teplota	Napětí		Výstupní proud	Vibrace	Axiální vůle hřídele	Excentricita hřídele + radiální vůle	Rychlost	Zrychlení
	Napájecí	Výstupní						
-40°C až 85°C	-0,5V až 7V	-0,5V až Napájecí	-1 až 5mA	5 až 1000Hz	± 0,175mm	0,04mm	30 000 ot/min	250 000 rad/s^2



Obrázek 6.3 - Průběh enkodéru [17]

- **C** – Jeden cyklus kódovacích párů (360°).
- **P** – Cyklus, kdy je na výstupu kanálu 1 nebo 2 logická 1 (180°).
- **S** – Počet stupňů (90°) mezi změnou stavu jednoho z dvou prvních kanálů.
- **ϕ** – Počet stupňů (nominálně 90°) mezi středem signálu logické 1 u kanálu 1 a středem signálu logické 1 u kanálu 2.
- **P_0** – Počet stupňů (90°) kdy je u kanálu 3 na výstupu logická 1.

Obrázek 6.4 - Enkodér HEDS-5640-A13 [17]

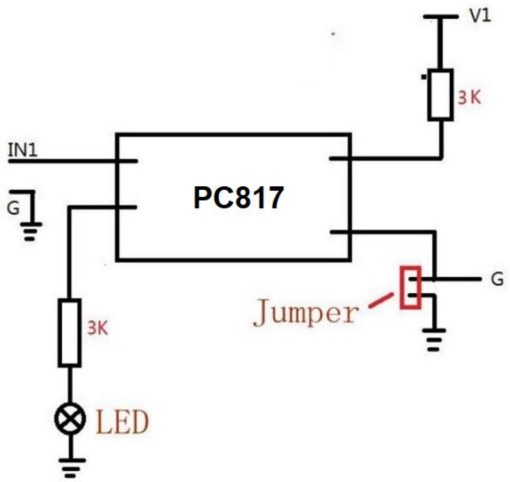
Izolační modul – U různých senzorů se často používají izolační moduly ke galvanickému oddělení senzoru a řídící elektroniky. Toto galvanické oddělení předchází poškození senzoru při neočekávaných napěťových špičkách, nebo eliminuje problém při rozdílném napětí řídícího obvodu a samotného senzoru. V úloze je použit izolační modul HY-M154/817 který

obsahuje 4 fotočlánky PC817. Tento fotočlánek se skládá z dvou hlavních částí, a to optického vysílače a optického přijímače. Optický vysílač je tvořen infračervenou diodou, která vysílá světelné impulsy v dávkách podle vstupního signálu. Optický přijímač obsahuje fototranzistor, který detekuje světelné impulsy a generuje odpovídající elektrický signál na výstupu.[18]

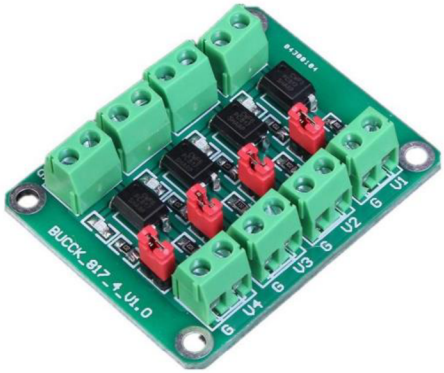
Jelikož samotný enkodér má napájecí napětí dle tabulky č. 6.2 v rozsahu -0,5 – 7 V a napájecí napětí řídícího obvodu (Arduina) který bude použit je 5V, lze předpokládat že v minulé verzi byl řídící obvod, který pracoval na vyšší napěťové hladině. V tabulce č. 6.3 jsou poznamenány samotné parametry fotočlánku PC817 a na obrázku č. 6.5 je vyznačeno schéma celého modulu HY-M154/817 a na obrázku 6.6 již samotný modul.

Tabulka 6.3 - Parametry izolačního modulu PC817 [18]

Napěťová izolace	Počet kanálů	Napětí		Proud		Frekvenční limit
		Vstupní	Výstupní	Řídící	Výstupní	
5 kV	4	3,6 až 24 V	3,6 až 30 V	20 mA	50 mA	4 kHz



Obrázek 6.5 - Schéma izolačního modulu HY-M154/817 [19]



Obrázek 6.6 - Izolační modul HY-M154/817 [18]

6.2 Návrhy na inovaci

Jelikož byla úloha v nedokončeném stavu, je na ní několik věcí, které by se dali vylepšit. Jako hlavní věci k vylepšení by mohli být upevnění vlákna k tyči na hřídeli, upevnění hmotného bodu k druhému konci vlákna, vytvoření podstavy k nosné konstrukci úlohy, navrhnutí řízení stejnosměrného motoru a volba IP kamery pro vzdálený přístup k úloze. Všechny tyto podněty k inovaci budou rozepsány v dalších podkapitolách.

6.2.1 Upevnění vlákna

V tuhle chvíli je upevnění vlákna k tyči řešeno pouze pomocí průhledné izolační pásky. Řešením na vylepšení tohoto spojení by mohlo být vytisknutí spojovacího dílu na 3D tiskárně. Požadavky na tento spojovací díl by měly být:

- Možnost oddělení spojovacího dílu od tyče (pro případnou změnu vlákna s hmotným bodem)
- Zachování nynějšího řešení vlákna formou šroubu o průměru 6 mm
- Možnost fixace spojovacího dílu s hřídelí pro zamezení pohybu

Při řešení spojení spojovacího dílu s tyčí by se mohl spojovací díl skládat ze dvou částí, které by se dali přiklopit na hřídel a spojit k sobě, čímž by se připevnili k hřídeli a zamezili pohybu. Dalším řešením by mohl být spojovací díl v jednom kuse, který by se na hřídel nasunul a připevnil šroubem. V obou případech by byl splněn požadavek na rozebrání a fixaci s tyčí.

V případě spojovacího dílu a samotného vlákna může být použit závit ve spojovacím dílu, do kterého by stačilo pouze zašroubovat vlákno. U tohoto řešení by byla možnost rozebrání, která avšak není v tomto případě až tak nutná. Dalším řešením by mohlo být slepení, které by bylo permanentní, ale stejně účinné.

6.2.2 Upevnění hmotného bodu

Hmotný bod s vláknem je ve výchozím stavu spojen opět pouze průhlednou lepící páskou. Na konci vlákna u této strany je u šroubu plochá křížová hlava. Jako hmotný bod je použit míček ze stolního tenisu. Vylepšením by mohlo být nalepením míčku přímo na hlavu šroubu po jeho zbroušení pro lepší adhezi materiálů. Druhým řešením by mohlo být vytisknutím koule na 3D tiskárně, která by byla na vrchu zploštělá tak, aby přímo dosedala na hlavu šroubu. Poté by se opět oba materiály zbrousili a přilepili k sobě.

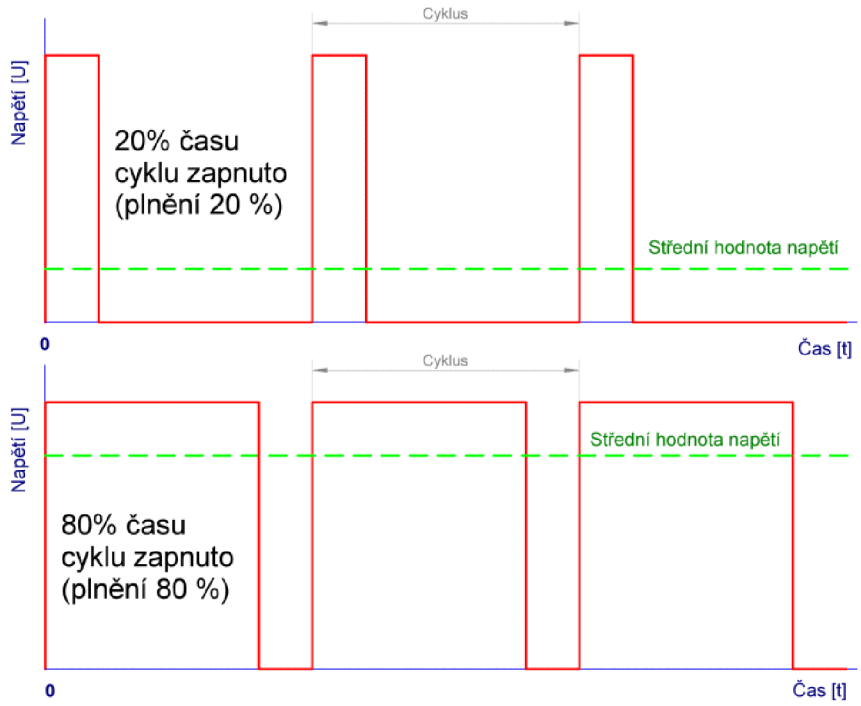
6.2.3 Podstava pro konstrukci

Nosná konstrukce úlohy je řešena čtvercovou podstavou z hliníkových profilů o rozměrech 30x30 mm. Pro lepší stabilitu konstrukce by bylo vhodné vytvořit čtyři nastavitelné nohy s protiskluzovou podložkou. Nastavitelné nohy by se skládali ze tří části. První částí by byla platforma se závitem, která by se zasunula do drážky v hliníkovém profilu. Tato podstava by se vytiskla na 3D tiskárně buďto rovnou s požadovaným závitem, nebo pouze s přípravou na závit, který by se do ní poté vyřezal. Druhou části by byl šroub s šestihrannou hlavou, který by upravoval výšku nohy podle potřeby. Poslední části by byla podstava, která by byla v kontaktu s materiálem, na kterém by úloha stála. Z jedné strany by měla výřez pro šestihranný šroub, který by se poté s podstavou slepil a na druhé straně by měla protiskluzovou podložku pro zamezení klouzavého pohybu. Tato poslední část by se opět dala vytisknout na 3D tiskárně

6.2.4 Řízení motoru

Stejnosměrný motor lze jednoduše ovládat regulací vstupního napětí motoru. Přičemž čím je menší vstupní napětí, tím jsou nižší otáčky motoru. Regulace napětí lze provádět například pomocí napěťového děliče. Dalším řešením by mohlo být řízení pomocí PWM (Pulsně šířková modulace). Tato varianta nám umožňuje řízení otáček motoru, pomocí rychlého vypínaní a zapínání vstupního napětí do motoru. Podle toho, jak dlouho je motor zapnut nebo

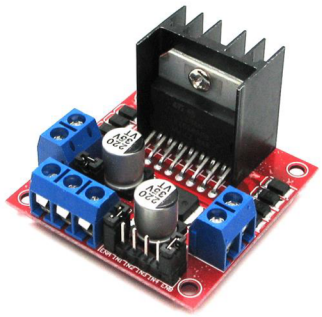
vypnut, se nám mění výsledné průměrné napětí viz obrázek č. 6.7. Další regulací otáček motoru by mohlo být ještě také regulátor proud.[20]



Obrázek 6.7 - Průběh regulace pomocí PWM [21]

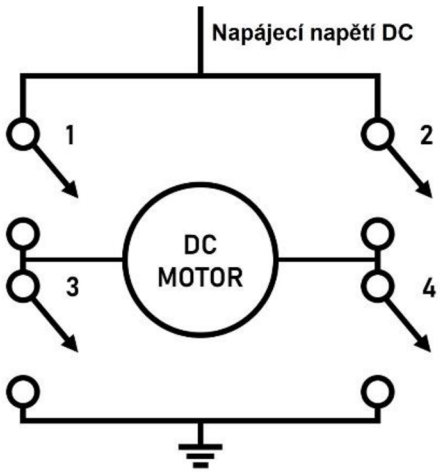
Jelikož je řešení pomocí PWM na rozdíl od napěťového děliče, nebo jiných regulátorů napětí bezztrátové a nabízí plynulou regulaci, bude pro tuto úlohu vybráno toto řešení. Platforma Arduino, přes kterou se bude motor ovládat podporuje PWM na několika výstupních pinech.[20]

Pro vypínání vstupu a výstupu motoru bude vybrán řídící H-můstek, který dokáže pomocí PWM signálu z Arduina otevírat a zavírat tranzistory které při otevřeném stavu přivedou napájecí napětí k motoru a při zavřeném stavu naopak přeruší jeho přívod. Další žádaná vlastnost H-můstku je možnost pomocí elektrického signálu z Arduina měnit samotný směr otáčení motoru. Pro úlohu byl tedy vybrán modul s H-můstek L298N viz obrázek č. 6.8.



Obrázek 6.8 - H-můstek L298N [22]

Na obrázku č. 6.9 je zjednodušené schéma principu H-můstku. Jsou zde čtyři spínače (tranzistory) ovládané v párech (1 a 4, 2 a 3), a když je některý z těchto párů sepnutý, uzavře se obvod a motor je napájen. Tento obvod v podstatě přepíná vodiče stejnosměrného motoru, který na povel změní směr svého otáčení (polaritu na vstupu motoru). Snadno se prodávají jako čipy a lze je nalézt ve většině mikroprocesorových řídicích jednotek, protože H můstek lze pomocí tranzistorů zmenšit na velmi malé rozměry.[23]



Obrázek 6.9 - Schéma H-můstku [23]

Zvolený modul obsahující H-můstek L298N, který má parametry viz tabulka č. 6.4.

Tabulka 6.4 - Parametry modulu s H-můstkem L298N [24]

Napětí motoru	Napětí logické úrovně	Proud logické úrovně	Max. proud motoru	Max. výkon
5 – 35 V	5 V	až 36 mA	2 A	25 W

6.2.5 IP kamera

Pro vizuální pohled v případě distančního ovládání úlohy je potřeba do výukové úlohy začlenit IP kameru, aby studenti byli schopni v reálném čase sledovat její chování. Pro tuto úlohu byla vybrána IP kamera RLC-522 od značky Reolink. Kamera obsahuje svoje webové rozhraní, na které se student dostane zadáním IP adresy této kamery do webového prohlížeče a poté se přihlásí pomocí předem definovaného přihlašovacího účtu.

6.3 Návrhy úlohy

V této podkapitole budou představeny tři návrhy úloh. U každé úlohy bude popsán základní princip, způsob realizace a možné úkoly pro studenta. Z těchto tří úloh se v další kapitole vybere jedna, která bude následně realizovaná.

6.3.1 Návrh č.1

První návrh úlohy se bude zabývat regulací kyvadla do inverzní polohy. Jedná se o takzvané inverzní kyvadlo.

Základní princip

Princip úlohy bude seznámit studenta s regulací polohy u praktického příkladu inverzního kyvadla. Při změně konstrukce by tato úloha mohla obsahovat podstavu, která by se pohybovala po vodorovné osa tak, že by se kyvadlo snažila vyvést z rovnovážného inverzního stavu. Při zachování konstrukce by bylo kyvadlo ve výchozí poloze směrem dolů a motor by měl za úkol kyvadlo otočit do inverzní polohy a v tomto stavu ho udržet. Samotný úkol bude již tedy dostat kyvadlo do inverzní polohy, přičemž i po této regulaci lze kyvadlo vychylovat z této polohy ručně a přinutit ho tím znovu regulovat svoji polohu do ustáleného inverzního stavu.

Způsob realizace

Pro realizaci takového typu úlohy by bylo potřeba vyměnit samotný řídící motor. Motor obsažený v úloze ve výchozím stavu je značně převodovaný a samotnou váhou hmotného bodu se nepohne ani při výchylce 90°, kdy je síla snažící se s vláknem pohnout největší. Volba jiného motoru by teda musela pracovat s poměrem samotného odporu motoru a jeho

výkonem. Druhou variantou při zachování stejného motoru by mohlo pomoci zvýšení hmotnosti hmotného bodu. Při této variantě je ale potřeba dbát na správnou volbu hmotnosti, protože při příliš velké hodnotě by vyvinutá síla motoru nemusela stačit pro korektní regulaci. Ostatní obsažené a navrhnuté elektronické prvky jako je enkodér pro snímaní otáček, modul pro řízení motoru a samotný řídící člen Arduino by v této úloze byli dostatečné.

U realizace řídícího programu by bylo nejdříve potřeba vytvořit sekvenci, která by kyvadlo pomocí pohybů tam a zpět dostala do inverzní polohy. Poté by již přišla na řadu část programu, která by měla za úkol držet kyvadlo v inverzní ustálené poloze. K této regulaci by mohl být použit PID regulátor, pro který je v samotném Arduinu několik knihoven, přičemž zpětnou vazbu regulátoru by zprostředkovával enkodér. Poté by přišlo na řadu propojení platformy Arduino s uživatelským rozhraním, přes které by se měnili parametry regulátoru. K tomuto účelu by se dalo využít komunikace po sériové lince přes rozhraní USB. Arduino by v tomto případě nejdříve z počítače přijalo počáteční parametry a po spuštění regulace by odesílalo data z enkodéru o vychýlení kyvadla z ustálené polohy. Jako samotné uživatelské rozhraní by mohla sloužit jednoduchá webová stránka, kam by se zadávali počáteční parametry a spouštěla úloha. Výstupní data z úlohy by poté mohli být zpracovány v programu Scilab, ve kterém by byl vidět průběh regulace v čase.

Úkoly pro studenta

Jako první úkol pro studenty by bylo seznámení se s inverzním kyvadlem formou samostudia. Poté by následovalo nastudování schéma zapojení se všemi senzory a krátkým popisem jejich fungování ze strany studenta. Dále již studenti přes uživatelské rozhraní nastaví parametry regulátoru a spustí úlohu. Pro nastavení jednotlivých složek regulátoru by byl k dispozici krátký návod s tím, jak postupovat při jejich volení. Konečným cílem studenta je pochopení fungování jednotlivých složek regulátoru a docílení co nejlepší regulace.

6.3.2 Návrh č.2

U druhého návrhu by se úloha věnovala simulaci nelineárního matematického kyvadla pomocí elektrického motoru.

Základní princip

Matematické kyvadlo má při počáteční výchylce menší než 5° harmonický průběh. Při větších počátečních výchylkách se kyvadlo však chová již nelineárně a perioda kmitu je oproti harmonickému průběhu větší. Nelineární kyvadlo bývá také označené jako reálné kyvadlo. Tato úloha by se věnovala simulaci matematického nelineárního kyvadla, takže při uvažování nulového tření a odporu větru. Kyvadlo by se tedy po spuštění z jeho počáteční výchylky nemělo nikdy zastavit a jeho perioda by měla být po celou dobu konstantní. [25]

Způsob realizace

Pro tento typ úlohy by měl být veškerý obsažený a navrhnutý hardware dostačující. Jediné omezení by so mohlo najít opět u elektrického motoru při velmi malých periodách kmitů. V tomto případě by motor díky jeho převodům nemusel být dostatečně rychlý.

Jako první by se u této úlohy musela vyřešit výchozí poloha, podle uživatelem nastavené počáteční odchylky úhlu. K tomu by mohl dobře posloužit optický enkodér, který na svém třetím kanálu detekuje celou otáčku. Toho se dá využít na nastavení výchozí klidové polohy kolmo k zemi. Poté by už stačilo kyvadlo vychýlit o požadovaný počet stupňů, pomocí zbývajících dvou kanálu tohoto enkodéru. Po vyřešení počátečního vychýlení kyvadla je potřeba nastavit regulaci kyvadla podle pohybové rovnice nelineárního matematického kyvadla. K tomu by se dala opět použít PID regulace jako v případě 1. návrhu úlohy. Nakonec už stačí vyřešit pouze uživatelské rozhraní s nastavením vstupních parametrů a výstup se zpracováním naměřených dat, které by mohlo být opět řešeno stejně jako je již popsáno v 1. návrhu úlohy.

Úkoly pro studenta

Studentovi by bylo jako první zadáno seznámení se s princip matematického kyvadla a rozdílem mezi jeho zjednodušenou a nelineární verzí. Poté by byl student seznámen se schématem zapojení a použitými senzory, za účelem pochopení jejich funkčnosti. Tato první teoretická část by byla zprostředkována z větší části od samotného studenta formou dohledání si těchto obecných věcí na internetu. Po teoretickém základu by přišlo na řadu spuštění úlohy se vstupními parametry. Úkolem studenta by opět bylo hledání vhodných parametrů pro regulaci jako v 1. úloze. Dále by bylo zadáno spuštění simulace při různých

periodách kmitů a jejich porovnání s jednoduchým matematickým kyvadlem. Výstup úlohy by nakonec byli grafy realizované regulace při různých periodách kmitu.

6.3.3 Návrh č.3

Třetí návrh varianty úlohy by se věnoval opět simulaci kyvadla. V tomto případě by se ale jednalo o simulaci reálného kyvadla s působením vnějších jevů. Tedy kyvadlem, které se nakonec dostane do ustáleného stavu.

Základní princip

U reálného kyvadla je oproti matematickému potřeba zavést do výpočtů pohybové rovnice účinky tření a vnější periodické síly. Tření v tomto případě bývá v místě umístění pevného bodu vlákna a vnější periodické síly působí díky odporu vzduchu. V důsledku těchto aspektů, které vytvářejí na kyvadlo sílu opačného směru, se reálné kyvadlo za určitý čas zastaví.[26]

Způsob realizace

U tohoto typu úlohy, stejně u návrhu č. 2 by měl být veškerý obsažený a navrhnutý hardware dostačující a jediné omezení by mohlo být opět v rychlosti periody kmitu.

Při realizaci programu je potřeba nejdříve opět vyřešit počáteční výchylku kyvadla, před startem simulace. V tomto případě by šlo opět využít řešení jako u úlohy č. 2, pomocí třetího kanálu enkodéru. Další možností by v případě prezenčního použití úlohy mohlo být vychýlení samotného kyvadla rukou studenta. To by fungovalo tak, že by se kyvadlo nejdříve posunulo do výchozí klidové polohy kolmo k podstavě a student by poté rukou kyvadlo vychýlil. Enkodér by zaznamenal počet otáček a program vypočítal vstupní odchylku. Poté by již stačilo spustit regulační PID program v závislosti na pohybové rovnici kyvadla. U tohoto případu by počáteční parametry nebyly pouze regulační složky PID regulátoru, ale také parametry odporu vzduchu a tření. Druhým řešením regulace by mohla být snaha regulovat kyvadlo na výchozí klidovou polohu, kdy by úkolem studenta bylo vybrání takových složek PID regulátoru, aby výsledná regulace co nejpřesněji kopírovala pohybovou rovnici kyvadla. To by znamenalo vytvořit nedokonalou regulaci, která by od požadované hodnoty překmitávala, než by se ustálila. Následné uživatelské rozhraní s nastavením vstupních parametrů a výstup zpracování naměřených dat by mohlo být stejné jako v případě úlohy č. 1.

Úkoly pro studenta

Ze začátku úlohy by studentovi bylo zadáno základní seznámení se s principem reálného kyvadla. Poté stejně jako u úlohy č. 2 by byl student seznámen se schématem zapojení a použitými senzory, za účelem pochopení jejich funkčnosti. Princip senzorů by byl zadán formou samostudia. Po základním seznámením se s principem a funkčností úlohy by bylo studentovi zadáno vybráni vhodných složek PID regulátoru pro docílení správné simulace. Nejdříve by se jednalo o regulaci podle pohybové rovnice reálného kyvadla a poté o regulaci na kostnatí bod výchozí polohy. Výstupem z tohoto měření by byly grafy pohybu kyvadla a vypočteného teoretického průběhu. Nakonec by student porovnal oba typy regulace a popsal hodnoty vnějších nastavovaných vlivů na periodu kmitů kyvadla.

7 Realizace úlohy

Pro realizaci úlohy byl vybrán návrh č. 2 z minulé kapitoly. Realizována bude tedy úloha pro simulaci matematického nelineárního kyvadla. Tento návrh byl vybrán díky větší rozmanitosti úlohy, možností jednoduššího rozšíření nebo spojení s návrhem č. 3 a také díky absenci potřeby měnit již obsahovaný hardware.

Kapitola realizace bude nejdříve rozebírat navrhnuté díly pro inovaci konstrukce, zapojení úlohy včetně finálně použitého hardwaru, samotného způsobu regulace, vytváření uživatelského rozhraní a vzdáleného přístupu úlohy. Dále bude obsahovat instrukce k měření pro studenty, a nakonec finální zhodnocení vytvořené úlohy.

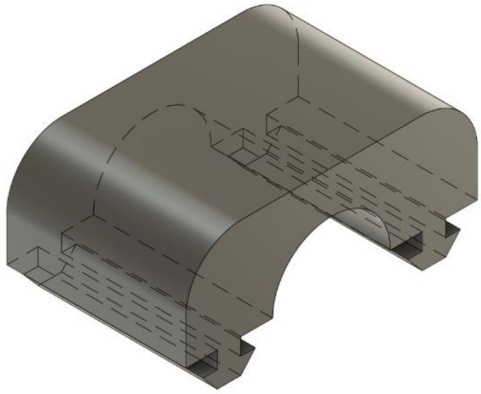
7.1 Konstrukce úlohy

Navrhnuté inovace s jejich požadavky pro konstrukci úlohy byli představeny v páté kapitole. Jednotlivé návrhy budou modelovány v programu Fusion 360, ze kterého se poté přímo exportují na formát pro tisknutí na 3D tiskárně.

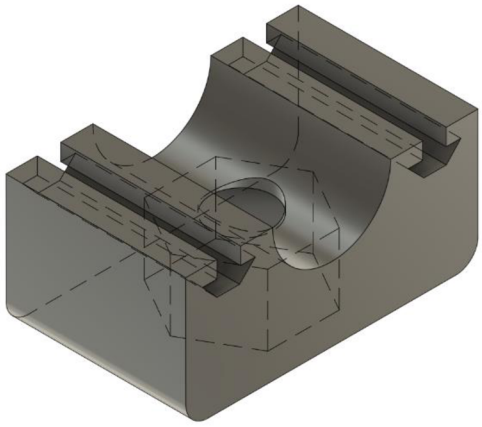
7.1.1 Upevnění vlákna

Upevnění vlákna ve formě šroubu je řešeno dvěma díly, které jsou navrženy tak, aby se do sebe daly zasunout. Díky tomu je docíleno možnosti rozebrání tohoto uchycení. Horní část dílu (obr. 7.1) se skládá z půl kruhu o průměru tyče 10 mm. Na dolní části tohoto dílu jsou poté vyčnívající části, které zapadnou do drážky dolního dílu. Celková šířka dílu je poté 26 mm, se zaoblením na okrajích o poloměru 5 mm. Hloubka dílu je 16 mm.

Dolní část dílu (obr. 7.2) je podobná horní části s tím rozdílem, že místo vyčnívající části obsahuje drážku. Na své spodní části má dále výřez pro matku o vnitřním průměru 6 mm, do které se zašroubuje vlákno s hmotným bodem ve formě šroubu. Aretace spojovacího dílu k tyči je docíleno samotným šroubem, který se zašroubováním zastaví až o samotnou tyč. Upevnění matky do dolního dílu je docíleno pomocí lepidla.



Obrázek 7.1 - Model upevnění vlákna - horní část

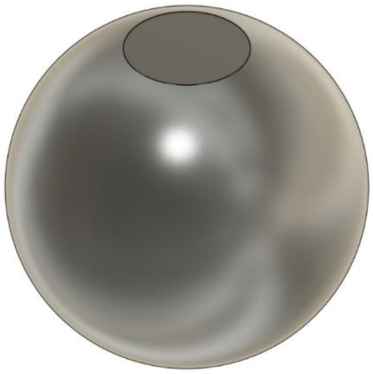


Obrázek 7.2 - Model upevnění vlákna - dolní část

7.1.2 Hmotný bod

Pro inovaci hmotného bodu byl na 3D tiskárně vytisknut nový díl ve tvaru koule (obr 7.3). Jeho velikost je stejná jako v případě míčku na stolní tenis a to průměr 40 mm. Na vrcholku této koule je zploštělá plocha ve formě kruhu o průměru 14 mm. Tento rozměr je stejný jako v případě hlavy šroubu, na který bude tento hmotný bod ve formě koule připevněn. Před spojením hlavy šroubu a hmotného bodu byli plochy obou částí zbroušeny pro docílení lepší adheze materiálu. Následně byli oba materiály k sobě přilepeny pomocí vteřinového lepidla.

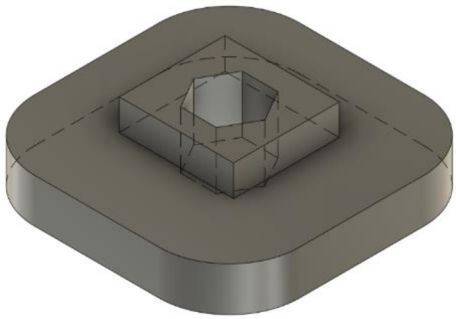
Kvůli větší váze, než v případě míčku na stolní tenis muselo být použito vlákno o délce 80 mm místo 100 mm, aby síla páky působící na motor byla snížena. Nakonec je tedy možnost použití obou hmotných bodů s rozdílnou délkou vlákna. Spojení delšího vlákna s míčkem na stolní tenis bylo realizováno stejně jako v případě nově vytisknutého hmotného bodu a kratšího vlákna.



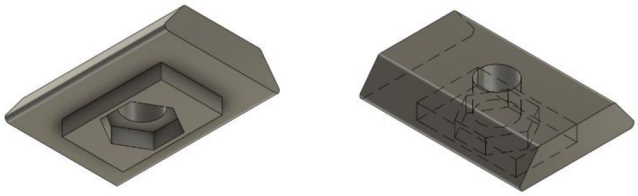
Obrázek 7.3 - Model hmotného bodu

7.1.3 Podstava pro konstrukci

Podstava konstrukce byla namodelována na dva díly. První díl na obr. 7.4 obsahuje otvor ve tvaru šestihranné hlavy šroubu o průměru 4 mm. Tento šroub se do něj vsune a zalepí. Poté se na spodní část tohoto dílu nalepí protiskluzová podložka pro lepší stabilitu. Druhá část navrhnuté podstavy na obr. 7.5 obsahuje výřez pro matici již zmíněného šroubu. Tento díl má přesný tvar, aby zapadl do drážky hliníkové profilu. Jakmile se do obou dílu vsune matice a šroub, stačí již druhý díl vložit do drážky hliníkové profilu a první díl zašroubovat do druhého. Výšku podstavy lze pak snadno regulovat pomocí utahování nebo povolování šroubu v rozmezí 9 mm. Úloha bude obsahovat na každém rohu konstrukce dvojici těchto dílů.



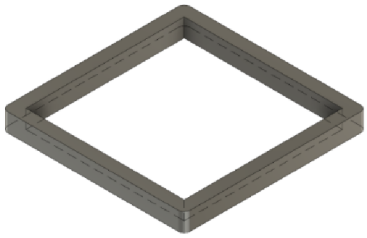
Obrázek 7.4 - Model podstavy pro konstrukci - držící mechanismus šroubu



Obrázek 7.5 - Model podstavy pro konstrukci - držící mechanismus v profilu konstrukce

7.1.4 Podložka pod motor

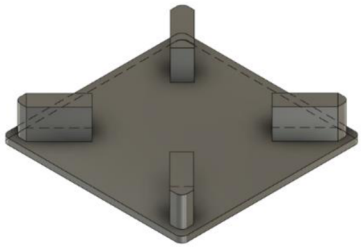
Při kompletaci úlohy nastal problém při snímání otáček enkodérem formou velké chybovosti. Příčinou této chyby byl sklon tyče, kvůli vyšší výšce enkodéru oproti motoru. K odstranění této chyby byla namodelovaná jednoduchá podložka (obr 7.6) pod motor, aby se zvedla jeho výška a tyč byla ve vodorovné rovině. Podložka má výšku 3 mm a rozměry shodné s hliníkovým profilem, tedy 30x30 mm s vnitřním čtvercovým průřezem o rozměrech 27x27 mm.



Obrázek 7.6 - Model podložky pod motor

7.1.5 Krytky průřezů profilu

Pro zakrytí uříznutých částí hliníkového profilu konstrukce na dolní části úlohy, byly namodelovány jednoduché krytky (obr. 7.7). Ty mají stejné rozměry jako samotný profil, a to 30x30 mm. Na vrchní části jsou dále namodelovány výklenky, které zapadnou přímo do tohoto profilu.



Obrázek 7.7 - Model krytky průřezu profilu konstrukce

7.2 Zapojení

V této kapitole bude představen veškerý hardware obsažený v úloze a popis schéma zapojení.

7.2.1 Použitý hardware

V úloze je použit veškerý hardware představený v páté kapitole až na optický izolační modul HY-M154/817. Jelikož Arduino pracuje na napěťové úrovni 5 V, je tento modul v tomto případě zbytečný.

Pro výběr řídící desky Arduino v této úloze se vybíralo mezi třemi deskami. Hlavní kritéria výběru byla přítomnost sériového portu pro komutaci s počítačem, přítomnost digitálních PWM portů pro ovládání motoru a aspoň dva interrupt porty pro snímání otáček z enkodéru. Na základě těchto požadavků bylo vybíráno mezi deskami Nano, Uno a Mega. Kritéria splňovali všechny tyto desky, ale největší z nich deska Mega by byla pro tuto úlohu zbytečně předimenzovaná, a proto byla z výběru vyřazena. Jelikož nebyly žádné kritéria pro šetření prostoru, byla upřednostněna deska Uno, se kterou je pohodlnější pracovat nežli s jeho menším protějškem Nano.

Pro řízení motoru byl vybrán modul L298N, který bude řídit 6V motor GM25-300CHV-130-R. Pro ovládání tohoto jednoho motoru je modul spíše předimenzovaný, ale při nízké pořizovací ceně kolem 50 Kč nebylo potřeba řešit modul pro slabší motory. Pro správnou funkčnost bylo ještě potřeba obstarat 6 V externí napájení pro dostatečný výkon motoru. Při pokusu napájet motor přímo z desky Arduina byl motor příliš pomalý a dolní hranice PWM hodnoty, kdy se motor uvedl do pohybu byla příliš vysoká na korektní regulaci. Pro externí 6V napájení byl vybrán síťový zdroj YCZX-6V-2A, který dokáže zásobit motor hodnotou proudu až 2A při napětí 6V.

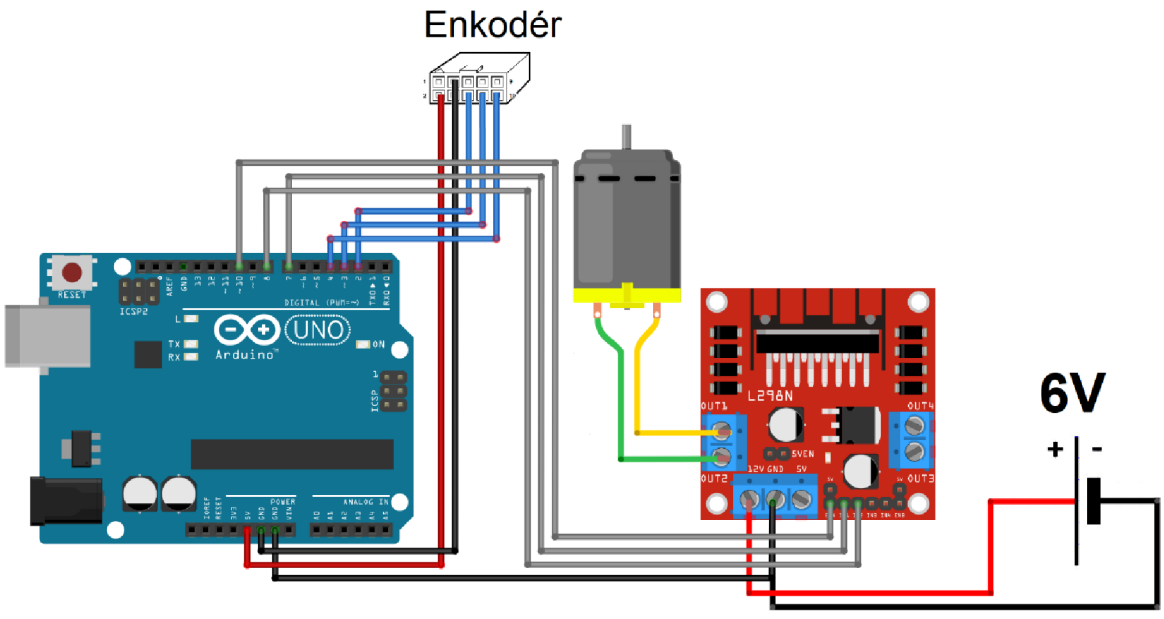
Dále se v úloze nachází enkodér HEDS-5640-A13, který byl již součásti převzaté úlohy. Bohužel dodaný enkodér byl poškozený a nefungoval mu jeden kanál. Enkodér byl tedy vyměněn za stejný typ novější výroby. Jediným rozdílem oproti starší výrobní sérii je přítomnost diferenciálního linkového ovladač AM26C31, který je integrován přímo v pouzdře enkodéru. V důsledku tohoto ovladače má enkodér nově na výstupu negované signály všech tří kanálu. Pro tuto úlohu však zůstanou nevyužity.

7.2.2 Schéma zapojení

Celé schéma zapojení je znázorněno na obrázku číslo 7.8. Napájení samotné desky je zprostředkováno pomocí USB portu, který je zapojen do počítače. Dalším prvkem napájení je síťoví zdroj, který je zapojen kladným pólem zdroje na pin 12 V řídícího modulu pro motor, záporný pól je zapojen na pin GND téže desky a zároveň i na GND pin desky Arduina.

Enkodér je propojen pouze s deskou Arduino Uno. Nejdříve jsou zapojeny napájecí porty 5V a GND na porty 2 a 3 na konektoru enkodéru. Dále jsou zapojeny samotné kanály enkodéru. Kanál A v portu 6 je připojen na digitální vstup 2 desky Arduinu, kanál B v portu 8 na vstup 3 a kanál C v portu 10 na vstup 4.

Řídící modul pro motory je propojen s kabely motoru na pinech OUT1 a OUT2. Samotný modul je s deskou Arduino propojen na portech IN1 a IN2, které určují, jakým směrem se bude motor pohybovat. Tyto porty jsou propojeny s digitálními výstupy desky Arduino 7 a 8. Poslední propoj je na port ENA řídícího modulu pro motory, který slouží k nastavení střídy PWM. Tento port je spojen s výstupním PWM pinem 10 desky Arduino.



Obrázek 7.8 - Schéma zapojení úlohy

7.3 Regulace kyvadla

V této podkapitole bude představen systém matematického kyvadla včetně jeho pohybových rovnicí. Dále bude popsán způsob regulace kyvadla podle těchto pohybových rovnicí. Nakonec bude popsán samotný program v Arduino desce, který bude tuto regulaci zprostředkovávat.

7.3.1 Matematické kyvadlo

Matematické kyvadlo je zjednodušeným modelem kyvadla reálného. Představuje hmotný bod na tenkém nepružném vlákně o zanedbatelné hmotnosti. Zanedbává se také veškerý odpor vzduchu při pohybu kyvadla a tření v závěsu vlákna.[27]

Z obrázku č. 7.9 si lze všimnout že výsledná síla působící pohyb kyvadla je rovna:

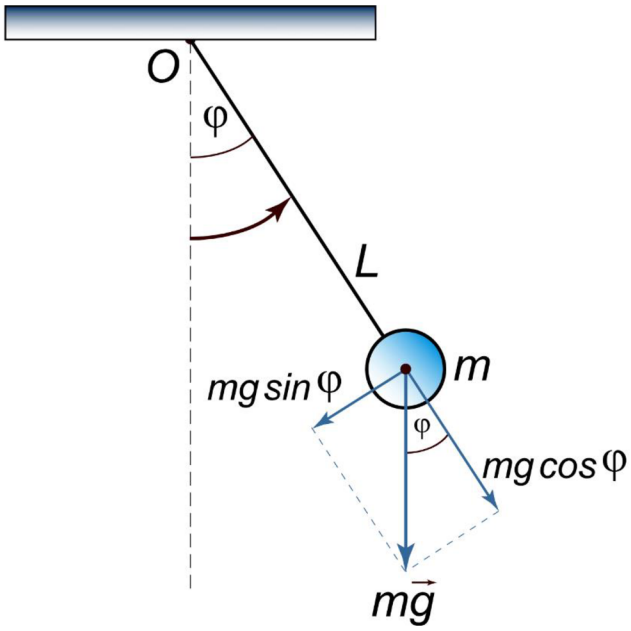
$$F = mg \sin \varphi$$

Sílu F lze podle druhého Newtonova pohybového zákona zapsat jako $F = ma$. Přičemž zrychlení a je rovno druhé derivaci dráhy objektu za určitý čas $\ddot{s}m$. Jelikož kyvadlo opisuje dráhu kružnice, lze druhou derivaci dráhy nahradit $\ddot{s} = \ddot{\varphi}L$. Po tomto nahrazení dostaneme vztah:

$$mL\ddot{\varphi} = -mg \sin \varphi$$

Vykrácením hmotnosti dostáváme rovnici nelineárního matematického kyvadla:

$$L\ddot{\varphi} + g \sin \varphi = 0$$



Obrázek 7.9 - Kyvadlo a působící síly [28]

Pomocí Euler-Cromerovi metody, jsme z této rovnice schopni vyjádřit rychlost a úhel vychýlení kyvadla v určitém čase. Euler-Cromerova metoda vyjadřuje rychlost a dráhu za určitý čas podle vzorců:[29]

$$v_{n+1} = v_n + a_n \Delta t$$

$$x_{n+1} = x_n + v_{n+1} \Delta t$$

První rovnice popisuje rychlost za určitý čas za předpokladu znalosti rychlosti za předešlí časový úsek a zrychlení pro nynější časový úsek. Druhá rovnice naopak popisuje přírůstek uražené dráhy za určitý časový úsek ze znalosti předešlého přírůstku a rychlosti za tentýž časový úsek. Pro případ matematického kyvadla lze tyto rovnice zapsat:[30]

$$\omega_{n+1} = \omega_n - \frac{g}{L} \sin \varphi \, \Delta t$$

$$\varphi_{n+1} = \varphi_n + \omega_{n+1} \Delta t$$

Kde platí:

- ω úhlová rychlost [rad/s]
- g tíhové zrychlení [m/s^2]

- L délka vlákna [m]
- φ úhel vychýlení [rad]
- Δt časový úsek [s]

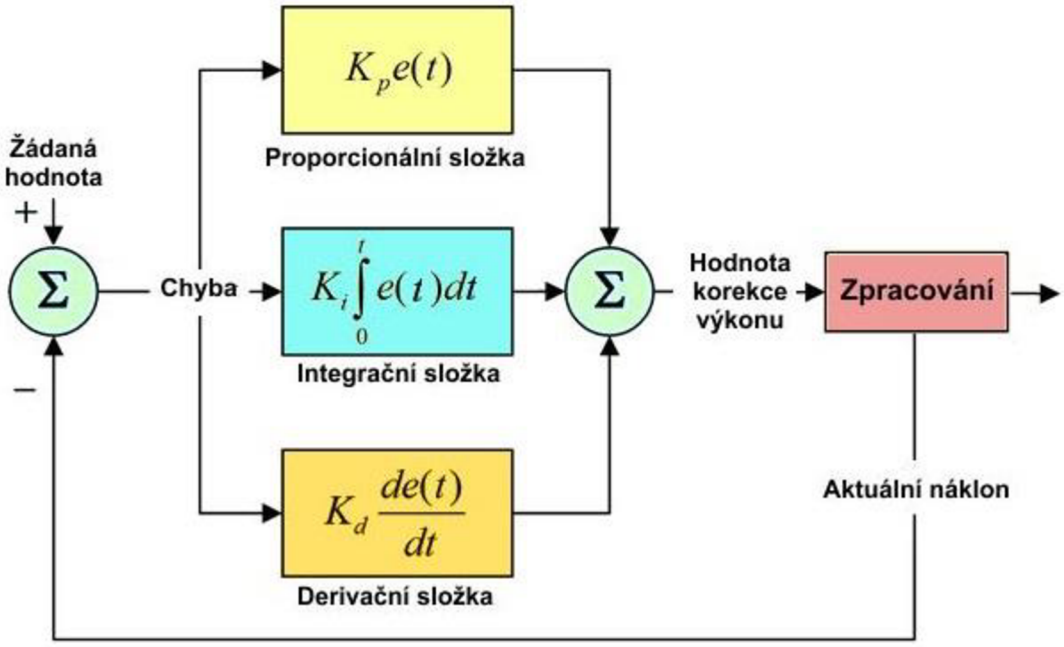
Z těchto rovnic jsme již schopni při počátečních parametrech ω_0 a φ_0 vypočítat rychlost a polohu kyvadla pro jakýkoliv časový úsek.

7.3.2 Způsob regulace

Pro regulaci pohybu kyvadla budou požity dva typy regulací. Prvním z nich bude PID regulátor, který se nejčastěji používá v uzavřených provozech průmyslové automatizace. Druhý typ regulace bude v podstatě jednodušší a bude se spíše než o regulaci jednat o ovládání. Toho bude docíleno nastavováním vypočtené rychlosti na motoru po dobu, než motor dosáhne vypočtené dráhy.[31]

PID regulátor

Zkratka PID znamená proporcionální integrální derivace. K regulaci se používají všechny tyto tři složky se zpětnou vazbou v regulační smyčce (obr. 7.10). U jednoduchých ON-OFF regulátoru jsou možné pouze dva řídící stavy, a to vypnutí nebo zapnutí akční veličiny. Naopak PID regulátor udržuje výstup tak, aby byla nulová chyba mezi procesní veličinou a žádanou hodnotou. Tento výstup udržuje pomocí tří základních složek v uzavřené smyčce.[31]



Obrázek 7.10 - Blokové schéma PID regulátoru [32]

- **Složka P** – Proporcionální neboli P regulátor dává výstup, který je přímo úměrný aktuální chybě. Porovnává žádanou hodnotu se skutečnou hodnotou ze zpětné vazby. Výsledná chyba se poté pouze vynásobí proporcionální konstantou a získá se výstup. Pokud je hodnota chyby nulová, pak je výstup také nulový. Použití samotného regulátoru P nám nikdy nezajistí ustálení systému, ale pouze kmitání kolem požadované hodnoty.[31]
- **Složka I** – Integrální regulace je přímo úměrná velikosti chyby a době trvání chyby. Integrální člen vypočítává součet okamžité chyby vzhledem k času. Tento údaj se pak vynásobí integrálním zesílením a přičte k výstupu systému. Tato regulace snižuje svůj výstup, když dojde k záporné chybě a omezuje rychlost odezvy systému. Tento člen je tedy zodpovědný za urychlení procesu směrem k získání požadované hodnoty a eliminuje chybu způsobenou proporcionálním členem.[31]
- **Složka D** – Složka I nemá schopnost předvídat budoucí chování chyby. Proto po změně žádané hodnoty reaguje normálně. D regulátor tento problém překonává tím, že předvídá budoucí chování chyby. Jeho výstup závisí na rychlosti změny chyby vzhledem k času vynásobené derivační konstantou. Dává výstupu impuls, čímž zvyšuje odezvu systému.

Nastavení parametrů PID regulátoru

Pro nastavování jednotlivých konstant složek PID regulátoru slouží Ziegler-Nicholsova metoda, která se skládá z několika kroků. Prvním krokem je nastavením složek I a D na hodnotu konstanty 0. Poté se postupně zvyšuje konstanta složky P, dokud se nedosáhne oscilujícího výstupu s konstantní amplitudou a frekvencí. Jakmile se dosáhne těchto požadavků, hodnota konstanty P se zaznamená jako hodnota Ku. Poté se zvolí metoda a vypočítají finální složky PID regulátoru.[33]

Tabulka 7.1 – Ziegler-Nicholsova metoda [32]

Metoda	Parametry		
	Kp	Ki	Kd
Ziegler-Nichols	0,6 Ku	2 Kp	Kp/8
Pessenovo integrální pravidlo	0,7 Ku	2,5 Kp	3Kp/20
Možnost překmitnutí	0,33 Ku	2 Kp	Kp/3
Žádné překmitnutí	0,2 Ku	2 Kp	Kp/3

Každá metoda je vhodná na jiný systém a je potřeba zjistit formou pokus omyl, která se hodí nejvíce. Někdy je také potřeba konstanty zvětšit při zachování jejich stejného poměru.[33]

Další metoda je více experimentální. První krok je stejný jako v předešlé metodě, tedy nastavení konstant složek I a D na 0. Poté se začíná pozvolna zvedat konstanta P složky, dokud nedosáhneme optimálního nastavení, kdy regulátor reaguje na vstupní signál rychle a přesně, aniž by příliš kmital. Po nastavení P složky se začne od nízké hodnoty přidávat složka I, dokud se opět nedosáhne optimálního nastavení. Toto nástavní by mělo pomoct eliminovat trvalou odchylku od požadovaného výstupu. Nakonec se stejně jako u P a I složky začne pomalu přidávat D složka, dokud nebudou eliminovány přechodové jevy v reakci na změnu vstupu.[34]

Regulace pomocí ovládání

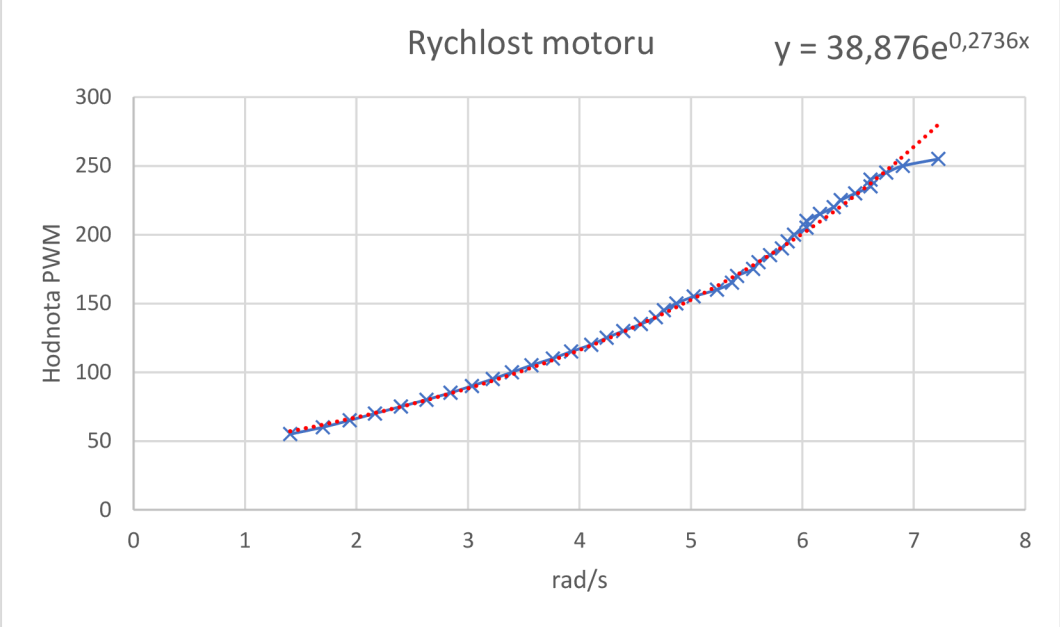
Druhým typem regulace bude ovládání se zpětnou vazbou. V podstatě se bude pouze nastavovat rychlost motoru na vypočtenou teoretickou rychlost, do té doby, než se motor otočí o vypočtený počet radiánů. Pro tento typ ovládání je potřeba zjistit reálnou rychlost motoru v závislosti na nastavené PWM hodnotě. Pro zjištění rychlosti motoru byl vytvořen krátký program v prostředí Arduino IDE (obr. 7.11).

```
void loop() {


    for (int i =55; i <251; i++){
      start = millis();
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      analogWrite(enA, i);
      while(otacky < 2000){

      }
      konec = millis();
      analogWrite(enA, 0);
      cas = konec - start;
      Serial.print("PWM = ");
      Serial.print(i);
      Serial.print("     cas = ");
      Serial.println(cas);
      i+=4;
      otacky = 0;
      delay(1000);
    }
```

Obrázek 7.11 - Program pro zjištění rychlosti motoru

Tento program počítá čas, dokud motor nedosáhne celé jedné otáčky, kterou reprezentuje 2 000 impulsů z enkodéru. Program je ve for smyčce, kdy se při každém novém cyklu přidá PWM hodnota o 5. Měřit se začíná až od PWM hodnoty 55, protože při menší hodnotě se motor neuvede do pohybu. Výsledné hodnoty měření jsou poté vypsány přes sériové rozhraní přímo v programu Arduino IDE.

Naměřené hodnoty nám udávají hodnotu čas/otáčka. Tuto hodnotu je potřeba převést na rad/s pomocí vzorce $v = \frac{1}{2\pi T}$ kde v je rychlost v radiánech za sekundu a T perioda za jednu otáčku. Přepočtené naměřené hodnoty byli vloženy do grafu (obr. 7.12). Z grafu lze vidět že průběh není lineární ale exponenciální. Průběh naměřených hodnot je proložen exponenciálou o tvaru $y = 38{,}876e^{0{,}2736x}$. Kde y znázorňuje hodnotu PWM a x rychlost v radiánech za sekundu. Pomocí této rovnice je již možné nastavovat požadovanou rychlost motoru v závislosti na vypočtené rychlosti v radiánech.



Obrázek 7.12 - Graf rychlosti motoru

7.3.3 Program pro regulaci

Jako první je potřeba nastavit inicializaci vstupů (obr. 7.13). Nejdříve je přidána knihovna ArduPID.h používaná pro výpočty PID regulace. Dále jsou pojmenovány digitální porty, ve kterých jsou zapojeny vstupy z řídícího modulu motoru a kanál C z enkodéru. Poté jsou porty 2 až 4 nastaveny na parametr INPUT, přičemž porty 2 a 3, které podle schéma reprezentují kanál A a B enkodéru plní funkci attachInterrupt s parametrem CHANGE. Tato funkce zavolá funkci pojmenovanou v druhém parametru, při jakékoliv logické změně signálu těchto dvou kanálů.

Po inicializaci vstupů enkodéru se spustí funkce začátek() (obr. 7.14), která natočí kyvadlo do výchozí polohy směrem kolmo k podstavě. Pokud není kanál C z enkodéru na kladné logické úrovni, nastaví se směr motoru a jeho rychlost na hodnotu PWM 120. Motor se točí do té doby, dokud kanál C z enkodéru nezmění svůj stav na logicky kladnou úroveň. Jakmile program zaznamená kladný impuls z kanálu C, vynulují se otáčky a nastaví rychlost motoru na 0. Po nastavení rychlosti motoru na 0, se motor ještě silou setrvačnosti vychýlí. Pro eliminování této výchylky je vytvořen cyklus, který motor otočí na opačnou stranu do té doby, než je počet otáček na hodnotě 0. To znamená v té pozici, kdy enkodér zaznamenal na kanálu C kladný impulz. Poté se motor vypne a otáčky enkodéru se nastaví zpátky na 0.

Mezi těmito dvěma akcemi je nastavená 1 vteřinová pauza, aby se po vynulování otáček nepřičetl malý inkrement v důsledku setrvačnosti motoru.

```
#include "ArduPID.h"

ArduPID myController;

#define enA 10
#define IN1 7
#define IN2 8
#define Index 4

void setup() {
  /* initialize serial
  Serial.begin(9600);

  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, INPUT);

  attachInterrupt(0, encoder_isr, CHANGE);
  attachInterrupt(1, encoder_isr, CHANGE);

}
```

Obrázek 7.13 - Program - inicializace vstupů

```
void zacatek(){
  if(digitalRead(Index) != HIGH){
    while(digitalRead(Index)  != HIGH){
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      analogWrite(enA, 120);
    }
    otacky = 0;
    analogWrite(enA, 0);
  }
  delay(500);
  while(otacky >= 0){
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    analogWrite(enA, 40);
  }
  analogWrite(enA, 0);
  delay(1000);
  otacky = 0;
  }
```

Obrázek 7.14 - Program - funkce zacatek()

Před samotnou regulací je ještě potřeba nastavit funkci pro počítaní impulsů enkodéru. Tato funkce (obr. 7.15) je díky předchozí inicializaci spuštěná pokaždé, když se změní stav vstupu na A a B kanálu enkodéru. Pro pochopení kódu je potřeba tabulka číslo 7.2. V této tabulce jsou znázorněny směry otáčení pro všechny kombinace aktuálního vstupu a minulého vstupu obou kanálů A a B. Tabulka je seřazena podle dekadických čísel od 0 do 15, přičemž každému řádku odpovídá binární hodnota všech čtyř stavů stejnému číslu jako v případě dekadického čísla. V posledním sloupci je poté pro každý z těchto stavů určen směr otáčení. Pro pohyb dopředu je kladná 1, pro pohyb dozadu záporná 1 a pro nečinný stav je 0.

```
void encoder_isr() {
    static int8_t lookup_table[] = {0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0};
    static uint8_t enc_val = 0;

    enc_val = enc_val << 2;
    enc_val = enc_val | ((PIND & 0b1100) >> 2);

    otacky = otacky + lookup_table[enc_val & 0b1111];
}
```

Obrázek 7.15 - Program - impulsy enkodéru

Podle tabulky č. 7.2 je vytvořena statická proměnná, která je podle indexu naplněna hodnotami pro směr otáčení. Dále je vytvořena 8 bitová proměnná enc_val, která uchovává aktuální a předešlou hodnotu. Na dalším řádku se tato proměnná binárně posune o 2 místa do leva, čímž se aktuální hodnota posune na místo předešlé. Na dalším řádku figuruje proměnná PIND, která se používá pro rychlé binární čtení digitálních vstupů 0 – 7. Jelikož pro potřeby enkodéru používáme piny 2 a 3, tak se výstup z PIND vynásobí binární hodnotou 00001100 a poté se posune o 2 bitové pozice doprava. Díky tomu dostaneme binární hodnotu s 6 nulami a na prvních dvou pozicích budou aktuální hodnoty kanálu A a B. poté následuje logické OR které se spojí proměnou enc_val, která uchovává hodnoty minulého stavu kanálů A a B. Z toho nám zůstane 8 bitová hodnota která má na prvních dvou místech hodnoty aktuální a na dalších dvou místech hodnoty minulé, stejně jako v tabulce 7.2. Poté už se pouze přičte k proměnné otáčky hodnota z proměnné, kde máme uloženou tabulku. Ta se zavolá podle indexu, který nám vyjde z proměnné enc_val, která se vynásobí hodnotou 00001111, abychom v ní měli pouze první 4 pozice.

Tabulka 7.2 - Směr otáčení podle stavu kanálů enkodéru [35]

Číslo	Stav				Binárně	Směr otáčení
	Minulý		Aktuální			
	A	B	A	B		
0	0	0	0	0	0000	0
1	0	0	0	1	0001	-1
2	0	0	1	0	0010	+1
3	0	0	1	1	0011	0
4	0	1	0	0	0100	+1
5	0	1	0	1	0101	0
6	0	1	1	0	0110	0
7	0	1	1	1	0111	-1
8	1	0	0	0	1000	-1
9	1	0	0	1	1001	0
10	1	0	1	0	1010	0
11	1	0	1	1	1011	+1
12	1	1	0	0	1100	0
13	1	1	0	1	1101	+1
14	1	1	1	0	1110	-1
15	1	1	1	1	1111	0

PID regulace

Pro samotnou PID regulaci (obr. 7.16) je nejdříve potřeba spustit funkci z knihovny ArduPID.h s inicializací parametrů. Nastaví se proměnné, které budou určovat zpětnou vazbu, akční člen, požadovanou hodnotu a PID složky. Poté se nastaví limity pro výstup z PID funkce. Jelikož motor lze ovládat PWM hodnotami pouze 0 – 255, limity jsou nastaveny na hodnotu -255 až 255. Dále se nastaví minimální čas mezi novým výpočtem PID výstupu na 10ms. Následuje zapsání počáteční výchylky kyvadla do proměnné otáčky. Poté se do proměnné startTime uloží výstupní hodnota z funkce millis(), která vrací aktuální čas v ms.

Následně začne samotný for cyklus, který si na začátku spočítá nové hodnoty rychlosti a úhlu posunutí podle rovnice v kapitole 7.3.1. Po vypočítání těchto hodnot se do vstupu PID kontroléru nahraje aktuální poloha kyvadla a na požadovanou hodnotu vypočítaný úhel posunutí. Na dalším řádku se pomocí funkce compute() vypočítá výstup z PID kontroléru, který je následně pomocí if podmínky zkontrolován, jestli je záporný nebo kladný. Pokud je výstup záporný, nastaví se opačný směr otáčení motoru a výstupní hodnota se zapíše pomocí

absolutní hodnoty. Jestli je vstup kladný, nastaví se pouze směr točení motoru dopředu. Po tomto porovnání následuje oříznutí spodní hranice hodnoty PWM na minimální hodnotu 75. Toto oříznutí je implementováno kvůli tomu, že při menších hodnotách se motor točí příliš pomalu, nebo vůbec. Po úpravách výstupní hodnoty se tato hodnota zapíše do výstupu pro ovládání motoru a následně program vyčká o časový úsek za který proběhl výpočet zkrácený o 2ms. Časový úsek je zkrácen kvůli zpoždění mezi výpočtem hodnoty a zapsáním výstupní hodnoty na vstup motoru. Nakonec se již zapíši hodnoty rychlosti a úhlu posunutí do hodnot minulých a poté následuje poslání informací o poloze kyvadla a hodnotě času po sériové lince.

Po skončení for cyklu se po sériové lince pošle hodnota 5 000, která indikuje konec přenosu a kyvadlo se zastaví.

```
myController.begin(&input, &output, &setpoint, kp, ki, kd);

myController.setOutputLimits(-255, 255);
myController.setSampleTime(10);
myController.setWindUpLimits(-10, 10);

b=10;
h=b/N;
otacky = uhel1/0.00314;         //nastavi otacky na nastavenou pocatecni vychylku kyvadla
perioda = 2*3.14*sqrt(L/g);     //spocita teoretickou periodu
startTime = millis();

for (int i=1; i<(perioda*N/2); i++){                              //samotny cyklus
  rychlost2 = rychlost1-(g/L*h*sin(uhel1));            //vypocet rychlosti
  uhel2 = uhel1+h*rychlost2;                           //vypocet uhlu za dobu h

  input= otacky*0.00314;
  setpoint= uhel2;

  myController.compute();
  if (output < 0) {                          //pri zaporne rychlosti otaci smerem doleva
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    output = abs(output);
  }else{                                        //pri kladne rychlosti otaci smerem doprava
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
  }

  if (output < 75){
    output = 75;
  }
  analogWrite(enA, output);
  delay(h*1000-2);
  uhel1 = uhel2;                                //zapise vypoctene hodnoty do hodnot minulych
  rychlost1 = rychlost2;
  Serial.println(otacky*0.00314);
  Serial.println((millis() - startTime)/1000);
}
Serial.println(5000);

start_kyvadlo = 0;
digitalWrite(IN1, LOW);                         //vypne motor
digitalWrite(IN2, LOW);
analogWrite(enA, 0);
```

Obrázek 7.16 - Program - PID regulace

Regulace pomocí ovládání

Začátek této regulace (obr. 7.17) začíná nastavením otáček na počáteční výchylku kyvadla. Po nastavení se do proměnné startTime uloží aktuální čas a následuje samotný for cyklus. V něm se nejdříve vypočítají hodnoty rychlosti a úhlu posunutí. Poté se z rychlosti vypočítá PWM hodnota pomocí rovnice z předchozí podkapitoly. Následně se ořízne minimální a maximální PWM hodnota. Minimální hranice je zadávána jako vstupní parametr od uživatele a maximální hodnota je oříznuta na maximální možnou PWM hodnotu 255. Po oříznutí této hodnoty se pomocí if podmínky zjišťuje, jestli je vypočítaná rychlost záporná nebo kladná. Díky této informaci nastavíme směr otáčení motoru na směr dopředu nebo dozadu. Následně se do vstupu motoru zapíše vypočtená PWM hodnota a vstoupí se do while cyklu, který zastaví program do té doby, než kyvadlo dosáhne vypočítaného úhlu. Po dosažení vypočteného úhlu se uloží hodnoty rychlosti a úhlu do hodnot minulých. Konec programu už je poté identický s tím, který je použit pro PID regulaci.

```
otacky = uhel1/0.00314;          //nastavi otacky na nastavenou pocatecni vychylku kyvadla
perioda = 2*3.14*sqrt(L/g);      //spocita teoretickou periodu
startTime = millis();

for (int i=1; i<(perioda*N*2); i++){                              //samotny cyklus
  rychlost2 = rychlost1-(g/L*h*sin(uhel1));              //vypocet rychlosti
  uhel2 = uhel1+h*rychlost2;                             //vypocet uhlu za dobu h
  rychlostPWM = (38.876*exp(0.2736*abs(rychlost2)));     //prevod hodnoty rychlosti na pwm signal

  if (rychlostPWM < Offset){
    rychlostPWM = Offset;
  }else if(rychlostPWM > 255){
    rychlostPWM = 255;                          //zastropovani PWM hodnot
  }

  if (rychlost2 < 0) {                           //pri zaporne rychlosti otaci smerem doleva
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    analogWrite(enA, rychlostPWM);
    while (otacky >= uhel2/0.00314){             //ceka dokud kyvadlo nedosahne pozadovaneho uhlu
    }
  }else{                                         //pri kladne rychlosti otaci smerem doprava
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(enA, rychlostPWM);
    while (otacky <= uhel2/0.00314){             //ceka dokud kyvadlo nedosahne pozadovaneho uhlu
    }
  }
  uhel1 = uhel2;                                 //zapise vypoctene hodnoty do hodnot minulych
  rychlost1 = rychlost2;
  Serial.println(otacky*0.00314);
  Serial.println((millis() - startTime)/1000);
}
Serial.println(5000);

start_kyvadlo = 0;
digitalWrite(IN1, LOW);                          //vypne motor
digitalWrite(IN2, LOW);
analogWrite(enA, 0);
```

Obrázek 7.17 - Program - Regulace pomocí ovládání

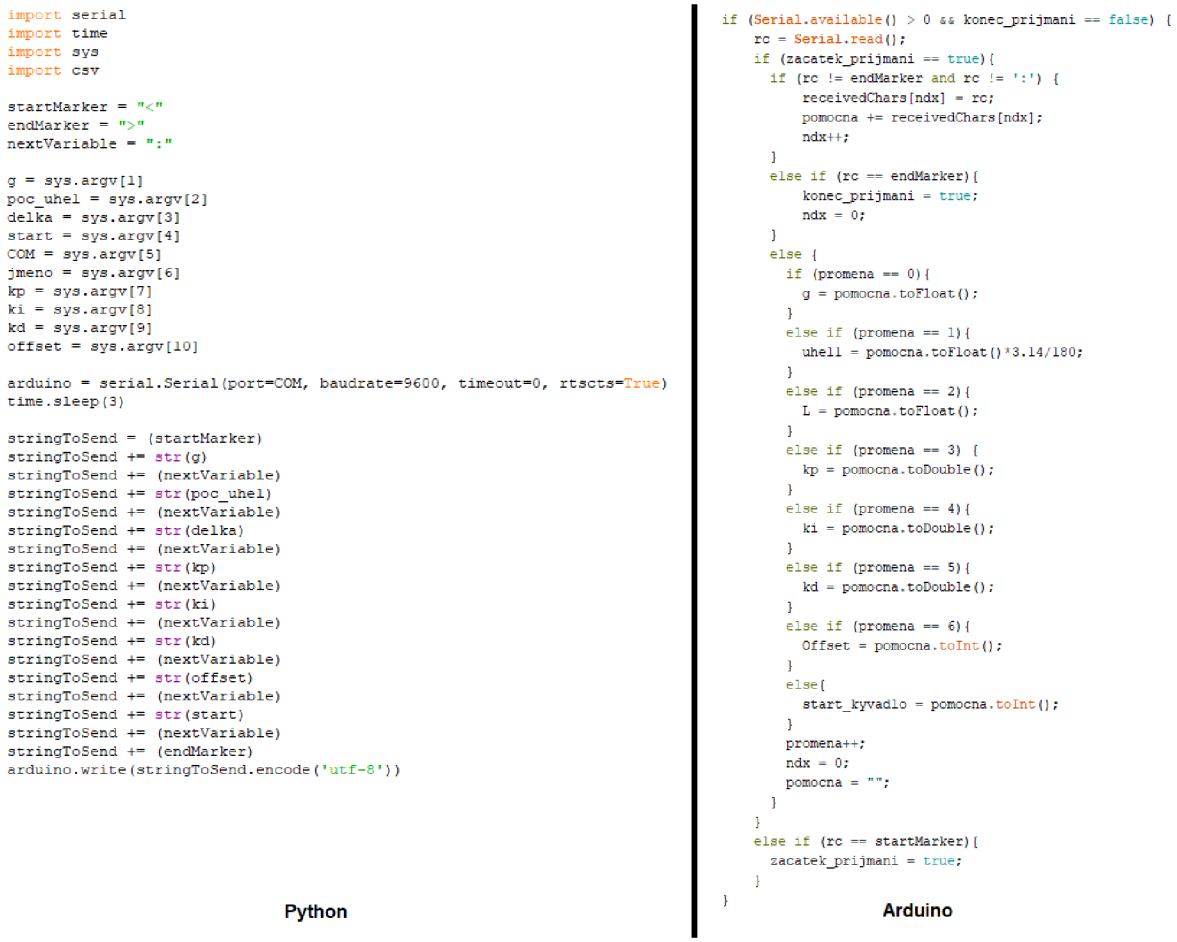
Kompletní program nahraný v desce Arduino se nachází v příloze č. 1

7.4 Komunikace po sériové lince

Aby mohl uživatel pomocí uživatelského rozhraní posílat a získávat data z desky Arduino je potřeba vytvořit sériovou komunikaci mezi počítačem a deskou. Pro tento účel byl vytvořen Python skript s knihovnou pyserial určenou pro sériovou komunikaci. Nejdříve bude popsána první část této komunikace, a to prvotní poslání parametrů do desky Arduino. K této komunikace slouží část programu na obrázku č. 7.18, kde se na levé části nachází program z Python skriptu a na pravé části program z desky Arduino. Python nejdříve převezme jednotlivé proměnné z PHP formuláře na webovém rozhraní, do kterého je zadá sám uživatel. Následuje nastavení a navázání sériové komunikace s desku Arduino. Poté se vyčká 3 vteřiny než se nabootuje deska Arduino a sloučí se všechny vstupní proměnné do jednoho textového řetězce s oddělovacím znakem mezi proměnnými a začátečním a konečným znakem. Nakonec již jen celý textový řetězec pošle po sériové komunikaci do desky Arduino.

Program v Arduino desce začíná, jakmile jsou dostupná nějaká data na sériové lince a proměnná indikující konec přijímání je FALSE. Pokud se jedná o první data, tak program zkontroluje, jestli se na první pozici nachází začáteční znak a pokud ano, tak změní proměnnou začátek přijímaní na TRUE. Poté se již postupně načítají znaky do pomocné proměnné do té doby, dokud se nenarazí na oddělovací znak. Jakmile se na tento znak narazí, hodnota z pomocné proměnné se uloží do příslušné proměnné v programu Arduina, vynulují se pomocné proměnné a cyklus se opakuje. Přenos končí načtením končícího znaku, kdy se do proměnné konec přijímání zadá hodnota TRUE a program již do této větve nevstoupí a začne samotný program regulace.

Program Arduina rozeznává typ proměnné podle sekvence, takže je potřeba dodržovat pořadí proměnných v programu Python a Arduino desce stejný. Jako poslední proměnná musí být vždy proměnná start, která nabývá hodnot 1 a 2 a určuje jaký typ regulace se má spustit. Nakonec je potřeba před uložením přijatých proměnných ve formě textového řetězce převést tyto proměnné na potřebný datový typ používaný v programu Arduina.



```python
import serial
import time
import sys
import csv

startMarker = "<"
endMarker = ">"
nextVariable = ":"

g = sys.argv[1]
poc_uhel = sys.argv[2]
delka = sys.argv[3]
start = sys.argv[4]
COM = sys.argv[5]
jmeno = sys.argv[6]
kp = sys.argv[7]
ki = sys.argv[8]
kd = sys.argv[9]
offset = sys.argv[10]

arduino = serial.Serial(port=COM, baudrate=9600, timeout=0, rtscts=True)
time.sleep(3)

stringToSend = (startMarker)
stringToSend += str(g)
stringToSend += (nextVariable)
stringToSend += str(poc_uhel)
stringToSend += (nextVariable)
stringToSend += str(delka)
stringToSend += (nextVariable)
stringToSend += str(kp)
stringToSend += (nextVariable)
stringToSend += str(ki)
stringToSend += (nextVariable)
stringToSend += str(kd)
stringToSend += (nextVariable)
stringToSend += str(offset)
stringToSend += (nextVariable)
stringToSend += str(start)
stringToSend += (nextVariable)
stringToSend += (endMarker)
arduino.write(stringToSend.encode('utf-8'))
```

Python

```arduino
if (Serial.available() > 0 && konec_prijmani == false) {
    rc = Serial.read();
    if (zacatek_prijmani == true){
      if (rc != endMarker and rc != ':') {
          receivedChars[ndx] = rc;
          pomocna += receivedChars[ndx];
          ndx++;
      }
      else if (rc == endMarker){
          konec_prijmani = true;
          ndx = 0;
      }
      else {
        if (promena == 0){
          g = pomocna.toFloat();
        }
        else if (promena == 1){
          uhel1 = pomocna.toFloat()*3.14/180;
        }
        else if (promena == 2){
          L = pomocna.toFloat();
        }
        else if (promena == 3) {
          kp = pomocna.toDouble();
        }
        else if (promena == 4){
          ki = pomocna.toDouble();
        }
        else if (promena == 5){
          kd = pomocna.toDouble();
        }
        else if (promena == 6){
          Offset = pomocna.toInt();
        }
        else{
          start_kyvadlo = pomocna.toInt();
        }
        promena++;
        ndx = 0;
        pomocna = "";
      }
    }
    else if (rc == startMarker){
      zacatek_prijmani = true;
    }
}
```

Arduino

Obrázek 7.18 - Program - Posílání vstupních parametrů po sériové lince

Následný příjem dat z desky Arduino zpátky do počítače probíhá opět pomocí Python skriptu na obrázku č. 7.19. Program v Arduinu pouze posílá potřebné proměnné po sériové lince ve formě bitového textového řetězce s ukončovacím znakem. V Python skriptu začne cyklus, který nejdříve přečte jeden celý bitový string, který následně převede na textový řetězec typu Unicode, odstraní ukončovací znaky, převede textový řetězec na datový typ float a uloží ho do příslušného pole. Rozlišení dvou proměnných opět probíhá sekvenčně, kdy se posílané proměnné v přijímání střídají. Cyklus skončí až tehdy, když Arduino pošle textový řetězec o hodnotě 5000, který indikuje konec přenosu. Mezi příjmem další várky dat je potřeba nastavit malou časovou pauzu (60 ms), jinak dochází k chybovosti při čtení těchto dat.

```
data = []
cas = []
time.sleep(10)
num = 0
pomocna = 0
while (num != 5000):
    line = arduino.readline()   # precte bitovy string
    if line:
        if pomocna == 0:
            string_n = line.decode()  # prevede byte string na unicode string
            string = string_n.rstrip() # odstrani ze stringu \n and \r
            pomocna += 1
            num = float(string)      # prevede string na float
            if num != 5000:
                data.append(num)    # vlozi hodnotu do pole data
        else:
            string_n = line.decode()
            string = string_n.rstrip()
            pomocna -= 1
            num = float(string) # prevede string na float
            num = num +1
            cas.append(num) # vlozi hodnotu do pole cas
            time.sleep(0.06)

arduino.close()
```

Obrázek 7.19 - Program - Příjem dat po sériové lince

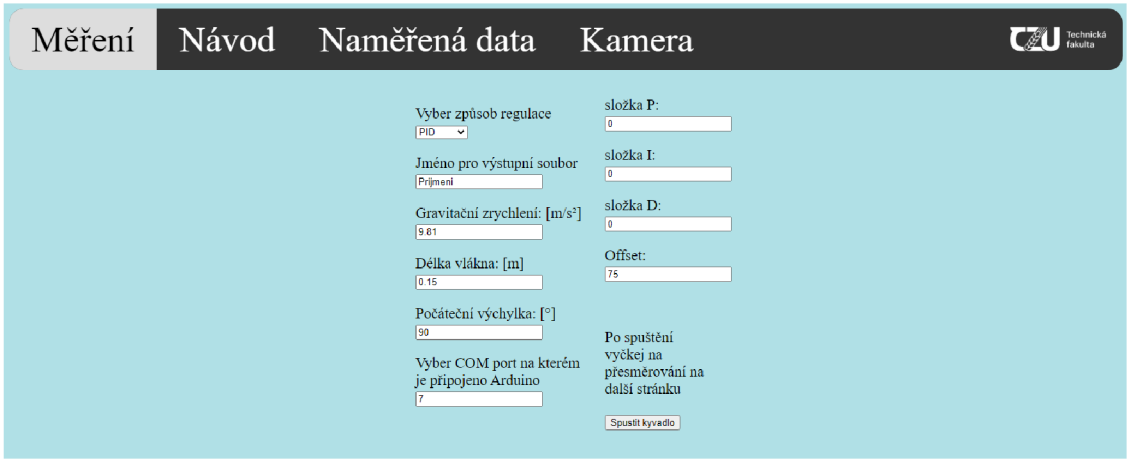
Pro správnou funkčnost Python skriptu je potřeba na počítač u úlohy nainstalovat Python software z oficiálních stránek. Dále je potřeba doinstalovat knihovnu pro sériovou komunikaci pyserial. Ta se dá nainstalovat pomocí příkazu pip install pyserial zadaného do příkazového řádku sytému Windows. Kompletní skript v programu Python se nachází v příloze č. 2.

7.5 Uživatelské rozhraní

Uživatelské rozhraní bylo vytvořeno formou webové stránky. Přes stránku napsanou pomocí HTML a PHP uživatel zadá parametry potřebné pro spuštění úlohy a ty se poté pomocí skriptu napsaném v Pythonu pošlou pomocí sériové linky do desky Arduino. Na webové stránce jsou také další podstránky návod, kamera a odkaz na umístění výstupního souboru z měření. Po vygenerování výstupního souboru ve formátu csv, je poté připraven Scilab skript, který soubor nahraje a promítne ho do grafu s teoretickým průběhem.

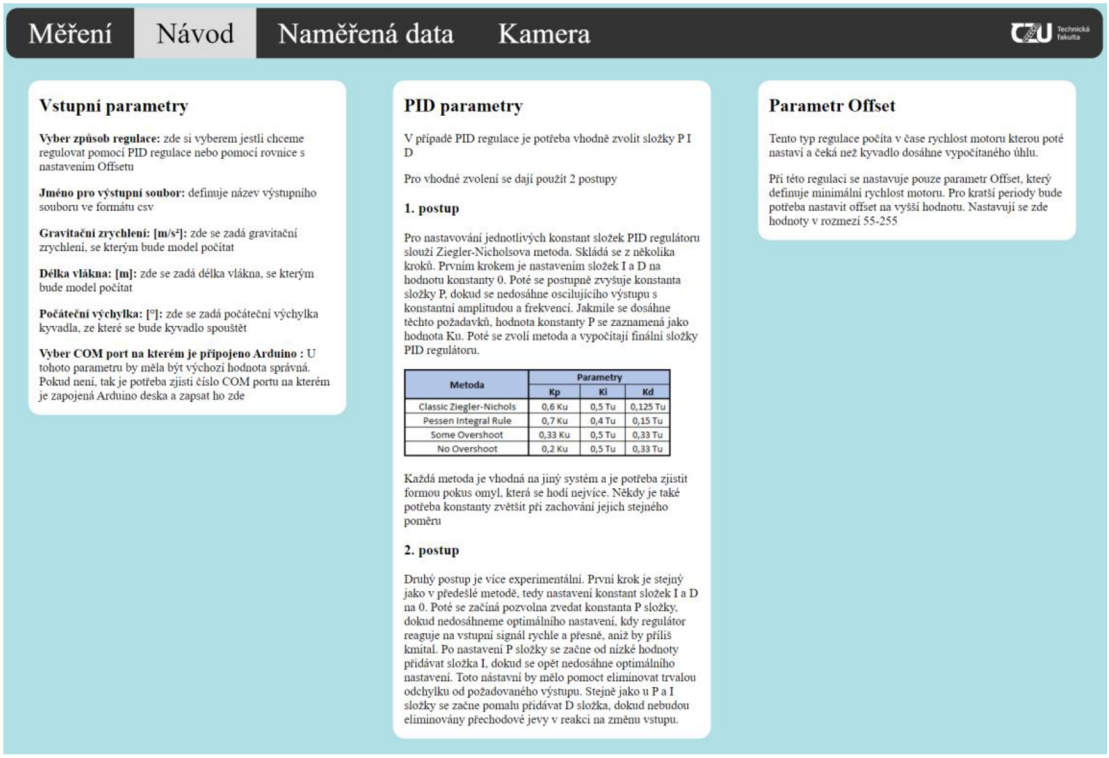
Na uživatelské rozhraní se student dostane zadáním IP adresy počítače, na kterém je spuštěn program XAMPP a je do něj zapojena Arduino deska. Úvodní stránka rozraní vypadá viz obrázek č. 7.20. Na této stránce se nacházejí veškeré vstupní parametry potřebné pro spuštění úlohy. Po vyplnění těchto parametrů pak stačí kliknout na tlačítko spustit kyvadlo a úloha se zapne. Po spuštění úlohy se musí počkat, než dojde automaticky na přesměrování na další stránku, kde je student informován o konci měření a odkázán na stránku kam se uloží

výstupní soubor z měření. Výstupní soubor je ve formátu csv a je naplněn počátečními parametry a výsledky měření. Vytvoření tohoto souboru probíhá opět přes Python skript.



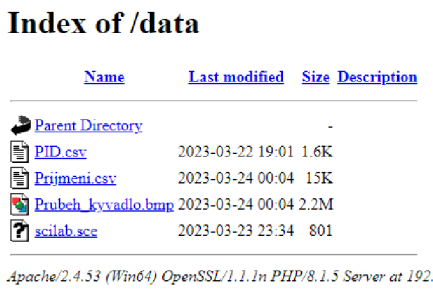
Obrázek 7.20 - Uživatelské rozhraní - Úvodní stránka

Další stránkou uživatelského rozhraní je stránka návod (obr. 7.21), kde je student seznámen s významem základních vstupních parametrů. Dále se zde nacházejí informace o způsobech regulace a krátký návod, jak postupovat při ladění jejich parametrů. Způsob ladění parametrů jednotlivých typů regulace je čerpán z této práce v kapitole 7.3.2



Obrázek 7.21 - Uživatelské rozhraní – Návod

Následující stránka naměřená data studenta odkáže na umístění výstupního souboru z měření, který lze z této stránky přímo stáhnout (obr. 7.22).



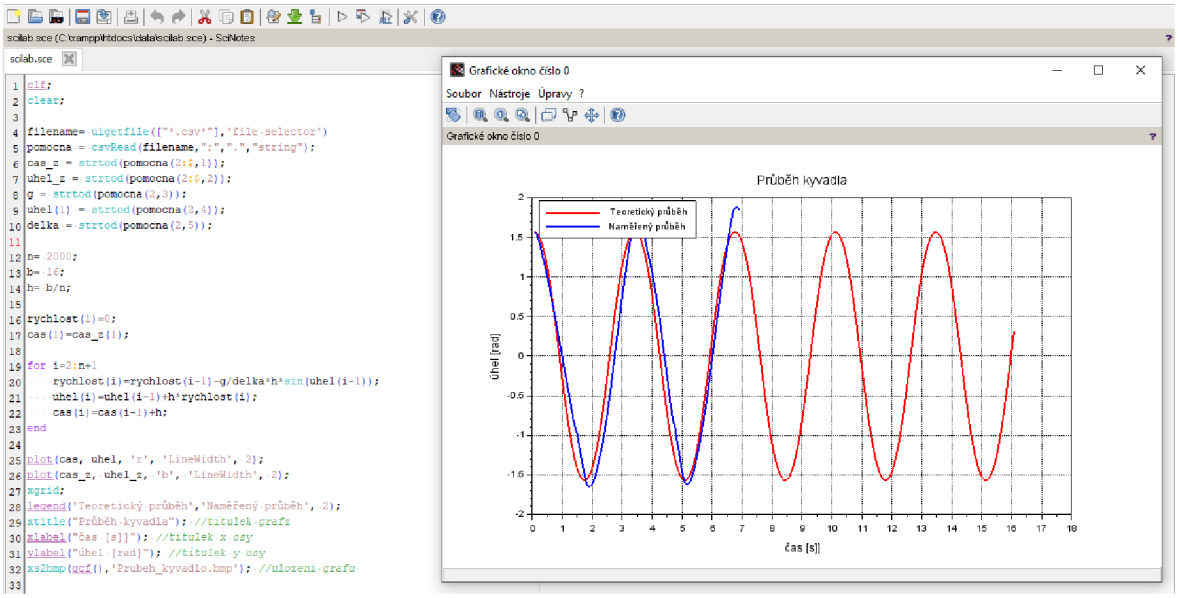
Index of /data

Name	Last modified	Size	Description
Parent Directory		-	
PID.csv	2023-03-22 19:01	1.6K	
Prijmeni.csv	2023-03-24 00:04	15K	
Prubeh_kyvadlo.bmp	2023-03-24 00:04	2.2M	
scilab.sce	2023-03-23 23:34	801	

Apache/2.4.53 (Win64) OpenSSL/1.1.1n PHP/8.1.5 Server at 192.168.0.57 Port 8080

Obrázek 7.22 - Uživatelské rozhraní - Naměřená data

Po získání souboru s naměřenými daty student spustí skript v programu Scilab, který ho vyzve k vybrání vstupního souboru. Student zde vybere jméno souboru, které vyplnil při zadávání vstupních parametrů. Po vybrání souboru program Scilab již vykreslí graf (obr. 7.23) s teoretickým a naměřeným průběhem a uloží ho jako obrázek ve formátu bmp. V Scilab skriptu je použita stejná rovnice pro výpočet teoretického průběhu jako je v desce Arduino.



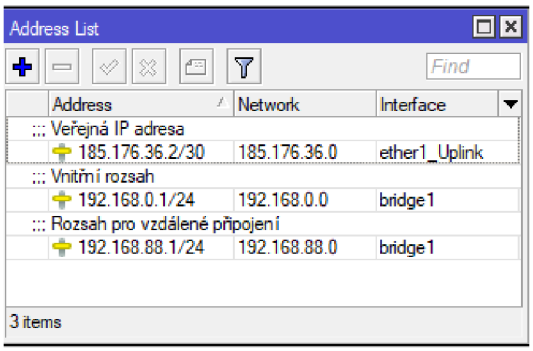
Obrázek 7.23 - Uživatelské rozhraní - Zpracování dat pomocí Scilabu

7.6 Vzdálený přístup

Jako uživatelské rozhraní pro vzdálený přístup k úloze slouží stejné uživatelské rozhraní jako v minulé podkapitole. Aby bylo umožněno studentovi se na toto rozhraní dostat z jiného počítače, než je samotný počítač u úlohy, je potřeba vytvořit VPN server na routeru u úlohy. Pro navázání VPN tunelu na router u úlohy bude dále potřeba vytvořit VPN klienta na straně studenta. Úloha bude ještě obsahovat IP kameru, aby student mohl v reálném čase sledovat chování úlohy.

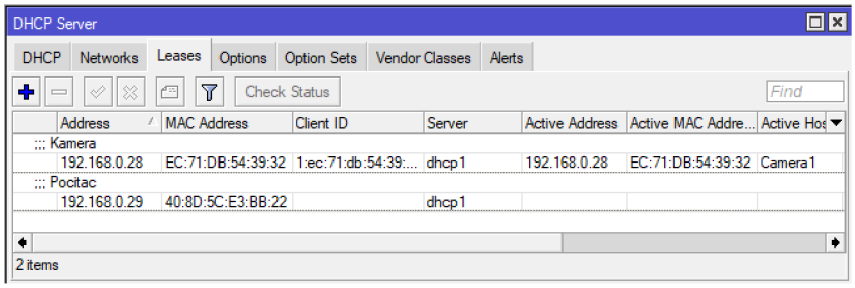
VPN server

Vnitřní síť pro počítač, do kterého je úloha zapojena bude vytvářet router Mikrotik hEX S. Pro správné nastavení routeru je nejdříve potřeba vytvořit rozhraní bridge a vložit do něj porty do kterých je zapojen počítač a IP kamera. Po vytvoření tohoto rozhraní se musí přidělit IP rozsah 192.168.0.1/24 (obr. 7.24). Na stejné rozhraní se navíc přidá i další rozsah pro uživatele připojené pomocí VPN, a to 192.16.88.1/24. Tyto IP adresy se přiřadí v záložce IP-> Addresses. Jakmile jsou IP adresy přiřazeny, je potřeba ještě nastavit výchozí bránu pro veřejnou IP adresu v záložce IP->routes.



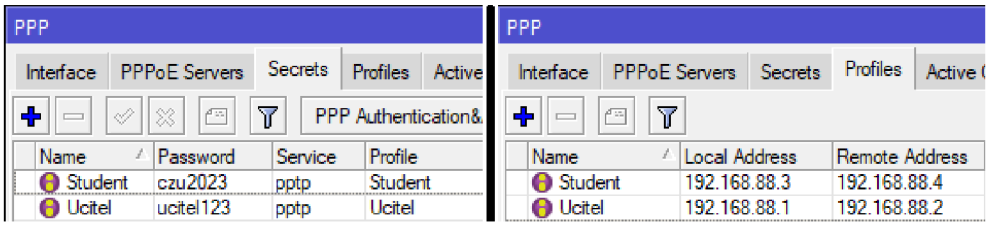
Obrázek 7.24 - Vzdálený přístup - Nastavení IP adres

Následuje vytvoření DHCP (obr. 7.25) serveru v záložce IP->DHCP Server, který přidělí IP adresu počítači a IP kameře. Přiřazené IP adresy je nutné změnit na stav statické, aby se jejich IP adresa neměnila.



Obrázek 7.25 - Vzdálený přístup - DHCP Server

Dále se přejde do záložky PPP->Profiles, kde se udělá profil pro studenta a učitele a nastaví se jim IP adresa z jejich rozsahu. V záložce PPP->Secrets se poté vytvoří jméno a heslo pro oba profily. U položky Service je potřeba zvolit pptp viz obrázek č. 7.26.



Obrázek 7.26 - Vzdálený přístup - VPN údaje

Po vytvoření těchto položek již stačí pouze zapnout samotný PPTP Server v záložce PPP->Interface. V této záložce je následně vidět v reálném čase jaký z nastavených profilů je připojen k routeru. Profily a hesla se dají měnit anebo přidávat podle potřeb vyučujícího.

Jelikož má routeru veřejnou IP adresa, znamená to, že se na něj může pokusit připojit kdokoliv ze světa. Proto se pro lepší bezpečnost vypnuly všechny přístupy na routeru v záložce IP->Services kromě připojení pomocí programu WinBox a webové stránky. Dále je také samozřejmostí nastavení silného hesla pro router. Kompletní nastavení routeru se nachází v příloze č. 3.

VPN klient

Na straně studenta, který by se chtěl k VPN serveru připojit je potřeba udělat určité nastavení v systému Windows. V nabídce „Centrum síťových připojení a sdílení“ se zvolí „Nastavit nové připojení nebo síť“. Vybere se možnost „Připojit k firemní síti“ a „Použít moje připojení k internetu (VPN)“. Následně se zadá veřejná IP adresa routeru a pojmenuje se VPN klient. Po vytvoření VPN klienta je již možné se přihlásit k VPN serveru. Přihlášení

probíhá rozkliknutím ikonky s internetovým adaptérem sítě na liště Windows, zvolením si VPN klienta a zadním přihlašovacích údajů, které se nastavili na VPN serveru. Po úspěšném přihlášení se vzdálený počítač nachází ve stejné síti jako počítač u výukové úlohy.

Konfigurace XAMPP

Aby bylo možné se dostat na webový server počítače u kterého je výuková úloha z vnitřní sítě, je potřeba ještě udělat na straně XAMPP serveru několik drobných úprav. Nejdříve je potřeba v programu XAMPP otevřít soubor pod záložkami Apache->config->Apache (httpd-xampp.conf). V tomto souboru je potřeba najít řádek který obsahuje „Listen 80“ a změnit ho na „Listen 8080“. Dále je potřeba nalézt řádek obsahující „ServerName localhost:80“ a změnit ho na „ServerName IP počítače:8080“. Nakonec je potřeba ve windows firewall na počítači odblokovat program XAMPP a povolit naslouchání přes port 8080. Po těchto úpravách už je možné se na webové rozhraní připojit z jakéhokoliv zařízení ve vnitřní síti po zadání IP adresy počítače a přidání za ní „:8080/index.php“.

IP kamera

Pro možnost vidět chování úlohy i ze vzdáleného přístupu byla k úloze nainstalována IP kamera připojená do routeru. Odkaz na přesměrování na IP kameru se nachází na webové stránce pod záložkou „Kamera“. Na stránce se poté nacházejí údaje k přihlášení do kamery a pokyny pro zaškrtnutí způsobu přenosu obrazu, pro lepší plynulost. Údaje obsažené na této stránce jsou pouze pro studenty a klient přihlášený přes tyto údaje nemá nastavené žádné oprávnění k jakémukoliv nastavení kamery.

Obrázek 7.27 - Vzdálený přístup - Kamera

7.7 Instrukce k měření

V této podkapitole bude popsáno možné zadání pro studenty k zpracování této úlohy. Pro toto zadání je předpoklad že úloha je zapojena dle schématu a připojena do počítače, kde jsou nainstalované potřebné programy a skripty popsané v této práci.

Zadání

- Seznámení se se schématem zapojení úlohy a použitým hardwarem.
- Nastudování pojmu matematické kyvadlo a zjištění rozdílu mezi lineárním a nelineárním matematickým kyvadlem.
- Připojení se na uživatelské webové rozhraní a seznámení se s návodem pro úlohu.
- Provést měření pro oba typy regulace pro 3 různé periody kmitu vzdálené od sebe aspoň 0,5 s, tak aby regulace byla co nejpřesnější.
- Protokol bude obsahovat grafy pro všechny 3 periody obou regulací a vyhodnocení jejich regulací s nastavenými parametry při regulaci.
- V protokolu bude dále popsán rozdíl mezi těmito regulacemi.

Zadání při vzdáleném přístupu k úloze bude totožné, s tím rozdílem že bude umožněno studentovi si stáhnout skript ve Scilabu a budou mu dodány údaje pro připojení k VPN serveru.

Šablona pro protokoly k měření je obsažena v příloze č. 4.

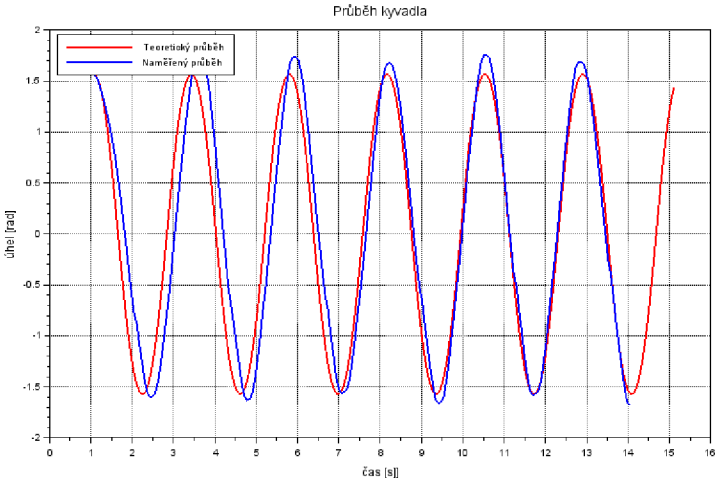
7.8 Zhodnocení

Úloha byla postavena tak, aby studenti museli najít co nejlepší parametry regulace. Pro volbu těchto parametrů je nutné porozumět základnímu principu obou regulátorů, které jsou popsány v návodu uživatelského rozhraní. Studenti si tak vyzkouší v praxi chování těchto regulátoru.

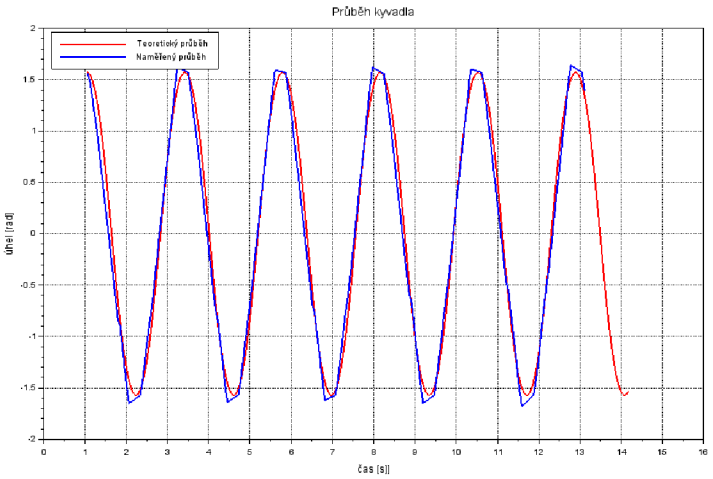
Oba realizované způsoby regulace mají každý jiný průběh a každý je lepší v jiné části. Regulace pomocí PID (obr. 7.28) lépe udržuje periodu kmitů, ale zase hůře amplitudu kmitů. Regulace pomocí ovládání (obr. 7.29) zase kopíruje lépe amplitudu, ale je velice citlivá na periodu kmitů, podle zvoleného offset parametru. Také hůře simuluje přechod kmitu z jedné

strany na druhou ve formě zubů. Ideální parametry regulace se mění s velikostí periody kmitů kyvadla, a je tedy nutné při každé změně periody hledat nové vhodné parametry regulace.

Díky skriptu v programu Scilab si studenti můžou ihned po dokončení regulace kyvadla získaná data promítnout do grafu s teoretickým průběhem. Z grafu lze pak snadno poznat, jestli byl výsledek regulace úspěšný či nikoliv. Pomocí vizuálního průběhu regulace lze také lépe porozumět jaké parametry PID regulátoru mají konkrétní efekt na výsledný průběh. Díky tomu je student pomocí praktické ukázky seznámen s použitými způsoby regulace.



Obrázek 7.28 - Graf regulace pomocí PID



Obrázek 7.29 - Graf regulace pomoci ovládání

Simulace matematického kyvadla v této úloze má určité omezení u minimální periody kmitů a počáteční výchylky kyvadla. Hodnota minimální periody kmitů pro počáteční výchylku 90° je 1,4 s. Kvůli omezené rychlosti motoru není úloha již schopna simulovat matematické kyvadlo pro menší periody při této počáteční výchylce. Počáteční výchylka kyvadla je omezena kvůli hmotnosti samotného vlákna a hmotného bodu. S rostoucí počáteční výchylkou roste i síla, kterou působí vlákno s hmotným bodem na motor. U menších period má poté motor problém překonat tuto síly při pokusu dostat se na tuto výchylku v průběhu regulace.

Fotky vyhotovené úlohy se nacházejí v příloze č. 5.

8 Závěr

Na začátku práce je popsáno základní seznámí s hardwarovým a softwarovým řešením pro vyhotovenou výukovou úlohu. Další část práce se věnuje popisu úlohy ve výchozím stavu včetně její konstrukce a použitého hardwaru. Poté jsou popsány navrhnuté inovace včetně jejich návrhu na realizaci. Na konci této části jsou pak představeny tři návrhy na realizaci úlohy včetně jejich principů, způsobu realizace a možného výstupu ze stran studentů.

Poslední část práce se věnuje samotné realizaci, která je členěna do několik podkapitol podle dílčích činností realizace. Nejdříve je popsána realizace konstrukčních inovací, které byli namodelovány v programu Fusion 360 a poté vytisknuty 3D tiskárně. Dále je popsáno celkové zapojení úlohy včetně schématu zapojení. Následující podkapitola se věnuje samotné regulace. Je v ní nejdříve odvozená pohybová rovnice matematického kyvadla, která je dále použita při určování cílové polohy při regulaci. Po odvození následuje teoretický popis způsobů regulace použitých v této úloze, včetně navrhování jejich parametrů. Dále se práce věnuje popisu programu regulace, kde je popsán základní princip programu. Konec práce se již zbývá tvorbou uživatelského rozhraní se vzdáleným přístupem k úloze a zhodnocením realizace úlohy.

Realizovaná úloha demonstruje chování regulátorů na praktické úrovni. Studenti mohou vidět chování regulátorů v závislosti na jimi zvolenými parametry. Pomocí uživatelského rozhraní mohou snadno zadávat různé parametry regulátoru a počáteční podmínky kyvadla. Vzdálené rozhraní umožňuje využití úlohy i formou distanční úlohy, kde díky kameře studenti v reálném čase vidí chování kyvadla při regulaci. Po simulaci kyvadla lze pomocí skriptu v programu Scilab porovnat teoretický průběh s průběhem naměřeným. Studenti tak dostanou vizuální informace z praktické ukázky regulace.

Seznam použitých zdrojů

[1] MAŘÍK, Vladimír. Trendy v komplexní automatizaci. Automa.cz [online]. [cit. 2023-03-29]. Dostupné z: https://automa.cz/cz/casopis-clanky/trendy-vkomplexni-automatizaci-2000_07_27781_735/

[2] Arduino For Beginners. Makerspaces.com [online]. [cit. 2023-03-29]. Dostupné z: https://www.makerspaces.com/wp-content/uploads/2017/02/Arduino-For-Beginners-REV2.pdf

[3] MONK, Simon. Programming Arduino. Sel.eesc.usp.br [online]. 2011 [cit. 2023-03-29]. Dostupné z: http://www.sel.eesc.usp.br/jcarmo/pdfs/Arduino_SimonMonk_2011.pdf

[4] VODA, Zbyšek. Průvodce světem Arduina. Robotikabrno.cz [online]. [cit. 2023-03-29]. Dostupné z: https://www.robotikabrno.cz/docs/arduino/Pr%C5%AFvodce-sv%C4%9Btem-Arduina-CZ.pdf

[5] Arduino Documentation. Docs.arduino.cc [online]. [cit. 2023-03-29]. Dostupné z: https://docs.arduino.cc/

[6] HASHEMI, mohammad. Introduction of Mikrotik Router. Ded9.com [online]. 2023 [cit. 2023-03-29]. Dostupné z: https://ded9.com/introduction-of-mikrotik-router-os-2023/

[7] HEX S. Mikrotik.com [online]. [cit. 2023-03-29]. Dostupné z: https://mikrotik.com/product/hex_s

[8] AKHILESH, KUMAR. PROGRAMMING USING SCILAB. Aagasc.edu.in [online]. 2022 [cit. 2023-03-29]. Dostupné z: http://www.aagasc.edu.in/Scilab-Book-Akhilesh.pdf

[9] BAUDIN, Michael. IINTRODUCTION TO SCILAB. Fd.cvut.cz [online]. 210n. l. [cit. 2023-03-29]. Dostupné z: https://www.fd.cvut.cz/personal/nagyivan/MatMetAnalDat/Literature/ScilabIntroduction.pdf

[10] XAMPP TUTORIAL. Javatpoint.com [online]. [cit. 2023-03-29]. Dostupné z: https://www.javatpoint.com/xampp

[11] Introduction to XAMPP. Educba.com [online]. [cit. 2023-03-29]. Dostupné z: https://www.educba.com/what-is-xampp/

[12] Virtual Private Network (VPN). Geeksforgeeks.org [online]. 2022 [cit. 2023-03-29]. Dostupné z: https://www.geeksforgeeks.org/virtual-private-network-vpn-introduction/

[13] Manual:Winbox. Wiki.mikrotik.com [online]. 2020 [cit. 2023-03-29]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:Winbox

[14] FEZARI, Mohamed a Ali AL DAHOUD. Integrated Development Environment. Researchgate.net [online]. [cit. 2023-03-29]. Dostupné z: https://www.researchgate.net/publication/328615543_Integrated_Development_Environment_IDE_For_Arduino

[15] ACEVEDO, Yarelis, Arianna COLÓN a Tiahra AVILÉS. Arduino. Indico.cern.ch [online]. [cit. 2023-03-29]. Dostupné z: https://indico.cern.ch/event/1068475/contributions/4493027/attachments/2296022/3904870/Arduino%20.pdf
[16] 25GA-300CH. Img.gme.cz [online]. [cit. 2023-03-29]. Dostupné z: https://img.gme.cz/files/eshop_data/eshop_data/5/671-033/dsh.671-033.1.pdf
[17] HEDS-5640 Data Sheet. Docs.broadcom.com [online]. 2014 [cit. 2023-03-29]. Dostupné z: https://docs.broadcom.com/doc/AV02-1046EN
[18] Čtyřkanálový izolační modul s optočleny. Dratek.cz [online]. 2019 [cit. 2023-03-29]. Dostupné z: https://dratek.cz/docs/produkty/0/991/1502443267.pdf
[19] PC817. Pajenicko.cz [online]. [cit. 2023-03-29]. Dostupné z: https://pajenicko.cz/ctyrkanalovy-modul-optoizolatoru-s-pc817
[20] Řízení DC motorů modulem ESP32. Kabinet.fyzika.net [online]. [cit. 2023-03-29]. Dostupné z: http://kabinet.fyzika.net/ESP32/ESP32-dc_motor/ESP32-rizeni-DC-motoru.php
[21] Robodoupe.cz [online]. [cit. 2023-03-29]. Dostupné z: https://robodoupe.cz/wp-content/uploads/2016/08/pwm-768x641.gif
[22] Rogegas.cz [online]. [cit. 2023-03-29]. Dostupné z: https://www.rogegas.cz/wp-content/uploads/2023/03/H-mustek-L298N-modul-pro-dva-motory.gif
[23] CAVALLO, Christian. All About DC Motor Controllers. Thomasnet.com [online]. [cit. 2023-03-29]. Dostupné z: https://www.thomasnet.com/articles/instruments-controls/dc-motor-controllers/
[24] L298N Motor Driver Module. Components101.com [online]. 2021 [cit. 2023-03-29]. Dostupné z: https://components101.com/modules/l293n-motor-driver-module
[25] Exact period of simple pendulum. Math.stackexchange.com [online]. 2020 [cit. 2023-03-29]. Dostupné z: https://math.stackexchange.com/questions/2257095/exact-period-of-simple-pendulum
[26] CHASNOV, Jeffrey. The Damped, Driven Pendulum. Math.libretexts.org [online]. 2022 [cit. 2023-03-29]. Dostupné z: https://math.libretexts.org/Bookshelves/Scientific_Computing_Simulations_and_Modeling/Scientific_Computing_(Chasnov)/II%3A_Dynamical_Systems_and_Chaos/11%3A_The_Damped%2C_Driven_Pendulum
[27] HERMAN, Russell. Nonlinear Pendulum. Math.libretexts.org [online]. [cit. 2023-03-29]. Dostupné z: https://math.libretexts.org/Bookshelves/Differential_Equations/A_Second_Course_in_Ordinary_Differential_Equations%3A_Dynamical_Systems_and_Boundary_Value_Problems_(Herman)/03%3A_Nonlinear_Systems/3.05%3A_Nonlinear_Pendulum
[28] Math24.net [online]. [cit. 2023-03-29]. Dostupné z: https://math24.net/images/second-order-equations-nonlinear-pendulum1.svg
[29] Euler-Cromer Method. Physics.udel.edu [online]. [cit. 2023-03-29]. Dostupné z: https://www.physics.udel.edu/~bnikolic/teaching/phys660/numerical_ode/node2.html

[30] HERMAN, Russell. The Nonlinear Pendulum. Math.libretexts.org [online]. 2022 [cit. 2023-03-29]. Dostupné z: https://math.libretexts.org/Bookshelves/Differential_Equations/A_First_Course_in_Differential_Equations_for_Scientists_and_Engineers_(Herman)/03%3A_Numerical_Solutions/3.05%3A_Numerical_Applications/3.5.01%3A_The_Nonlinear_Pendulum

[31] What is a PID Controller. Elprocus.com [online]. [cit. 2023-03-29]. Dostupné z: https://www.elprocus.com/the-working-of-a-pid-controller/

[32] KAČÍREK, JIŘÍ. Stavíme kvadrokoptéru: PID regulátor. Root.cz [online]. 2015 [cit. 2023-03-29]. Dostupné z: https://www.root.cz/clanky/stavime-kvadrokopteru-pid-regulator/

[33] KLÁN, Petr. Ziegler-Nicholsovo nastavení PID regulátoru. Automa.cz [online]. [cit. 2023-03-29]. Dostupné z: https://automa.cz/cz/casopis-clanky/ziegler-nicholsovo-nastaveni-pid-regulatoru-retrospektiva-2000_04_27697_3105/

[34] SEŘÍZENÍ PID REGULÁTORU. Vlab.fs.cvut.cz [online]. [cit. 2023-03-29]. Dostupné z: http://vlab.fs.cvut.cz/navody/files/hps.pdf

[35] Efficiently Reading Quadrature With Interrupts. Makeatronics.blogspot.com [online]. 2013 [cit. 2023-03-29]. Dostupné z: https://makeatronics.blogspot.com/2013/02/efficiently-reading-quadrature-with.html?showComment=1602748899528#c2428083900519675844

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha 1 – Kompletní kód desky Arduino

Příloha 2 – Kompletní kód skriptu v Pythonu

Příloha 3 – Kompletní nastavení routeru Mikrotik

Příloha 4 – Šablona pro protokoly z měření

Příloha 5 – Fotky hotové úlohy

```
#include "ArduPID.h"

ArduPID myController;

#define enA 10
#define IN1 7
#define IN2 8
#define Index 4

volatile int otacky = 0;

char receivedChars[32];
char startMarker = '<';
char endMarker = '>';
char rc;
static byte ndx = 0;
boolean zacatek_prijmani = false;
boolean konec_prijmani = false;
int promena=0;
float startTime;

String pomocna;
int start_kyvadlo;// = 2;

float g;//=9.81;
float L;//=2;
float b=3;
float N= 150;
float h= b/N;
float perioda;

float uhel1;// = 3.14/2;
float rychlost1 = 0;

float uhel2;
float rychlost2;
int rychlostPWM;
int Offset = 75;

float start;
float cas;
float konec;

double input;
double output;
double setpoint;
double kp;// = 0;
double ki;// = 0;
double kd;// = 0;

void setup() {
  /* initialize serial
  Serial.begin(9600);

  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, INPUT);

  attachInterrupt(0, encoder_isr, CHANGE);
  attachInterrupt(1, encoder_isr, CHANGE);

}
```

```
void loop() {
                                                    //TEST RYCHLOSTI MOTORU V ZAVISLOSTI NA NASTAVENEM PWM

 /* for (int i= 55; i<256; i++){
    start= millis();
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(enA, i);
    while(otacky < 2000){

    }
    konec = millis();
    analogWrite(enA, 0);
    cas = (konec - start)/1000;
    Serial.print("PWM = ");
    Serial.print(i);
    Serial.print("      cas = ");
    Serial.println(cas);
    i+=4;
    delay(2500);
    otacky = 0;
  }*/

  if (Serial.available() > 0 && konec_prijmani == false) {
      rc = Serial.read();
      if (zacatek_prijmani == true){
        if (rc != endMarker and rc != ':') {
            receivedChars[ndx] = rc;
            pomocna += receivedChars[ndx];
            ndx++;
        }
        else if (rc == endMarker){
            konec_prijmani = true;
            ndx = 0;
        }
        else {
          if (promena == 0){
            g = pomocna.toFloat();
          }
          else if (promena == 1){
            uhell = pomocna.toFloat()*3.14/180;
          }
          else if (promena == 2){
            L = pomocna.toFloat();
          }
          else if (promena == 3) {
            kp = pomocna.toDouble();
          }
          else if (promena == 4){
            ki = pomocna.toDouble();
          }
          else if (promena == 5){
            kd = pomocna.toDouble();
          }
          else if (promena == 6){
            Offset = pomocna.toInt();
          }
          else{
            start_kyvadlo = pomocna.toInt();
          }
          promena++;
          ndx = 0;
          pomocna = "";
        }
      }
```

```
    else if (rc == startMarker){
      zacatek_prijmani = true;
    }
}

  if ( start_kyvadlo == 1){     //zscatek kyvaciho cyklu
    zacatek();                  //otočí kyvadlo do výchozí polohy
    delay(500);
    while(otacky < uhel1/0.00314){    //vychílí kyvadlo o požadovaný počet stupnu
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      analogWrite(enA, 120);
    }

    analogWrite(enA, 0);
    delay(500);
    otacky = uhel1/0.00314;         //nastavi otacky na nastavenou pocatecni vychylku kyvadla
    perioda = 2*3.14*sqrt(L/g);     //spocita teoretickou periodu
    startTime = millis();

    for (int i=1; i<(perioda*N*2); i++){                              //samotny cyklus
      rychlost2 = rychlost1-(g/L*h*sin(uhel1));             //vypocet rychlosti
      uhel2 = uhel1+h*rychlost2;                            //vypocet uhlu za dobu h
      rychlostPWM = (38.876*exp(0.2736*abs(rychlost2)));    //prevod hodnoty rychlosti na pwm signal

      if (rychlostPWM < Offset){
        rychlostPWM = Offset;
      }else if(rychlostPWM > 255){
        rychlostPWM = 255;                          //zastropovani PWM hodnot
      }

      if (rychlost2 < 0) {                          //pri zaporne rychlosti otaci smerem doleva
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, HIGH);
        analogWrite(enA, rychlostPWM);
        while (otacky >= uhel2/0.00314){            //ceka dokud kyvadlo nedosahne pozadovaneho uhlu
        }
      }else{                                        //pri kladne rychlosti otaci smerem doprava
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        analogWrite(enA, rychlostPWM);
        while (otacky <= uhel2/0.00314){            //ceka dokud kyvadlo nedosahne pozadovaneho uhlu
        }
      }
      uhel1 = uhel2;                                //zapise vypoctene hodnoty do hodnot minulych
      rychlost1 = rychlost2;
      Serial.println(otacky*0.00314);
      Serial.println((millis() - startTime)/1000);
    }
    Serial.println(5000);

    start_kyvadlo = 0;
    digitalWrite(IN1, LOW);                         //vypne motor
    digitalWrite(IN2, LOW);
    analogWrite(enA, 0);
}
else if (start_kyvadlo == 2){
  myController.begin(&input, &output, &setpoint, kp, ki, kd);

  myController.setOutputLimits(-255, 255);
  myController.setSampleTime(10);
  myController.setWindUpLimits(-10, 10);

  myController.start();
  delay(250);
```

```
    zacatek();                  //otočí kyvadlo do výchozí polohy
      delay(500);
      while(otacky < uhel1/0.00314){    //vychílí kyvadlo o požadovaný počet stupnu
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
        analogWrite(enA, 120);
      }
      analogWrite(enA, 0);
      delay(500);
      b=10;
      h=b/N;
      otacky = uhel1/0.00314;         //nastavi otacky na nastavenou pocatecni vychylku kyvadla
      perioda = 2*3.14*sqrt(L/g);     //spocita teoretickou periodu
      startTime = millis();

      for (int i=1; i<(perioda*N/1.5); i++){                              //samotny cyklus
        rychlost2 = rychlost1-(g/L*h*sin(uhel1));             //vypocet rychlosti
        uhel2 = uhel1+h*rychlost2;                            //vypocet uhlu za dobu h

        input= otacky*0.00314;
        setpoint= uhel2;

        myController.compute();
        if (output < 0) {                           //pri zaporne rychlosti otaci smerem doleva
          digitalWrite(IN1, LOW);
          digitalWrite(IN2, HIGH);
          output = abs(output);
        }else{                                        //pri kladne rychlosti otaci smerem doprava
          digitalWrite(IN1, HIGH);
          digitalWrite(IN2, LOW);
        }

        if (output < 75){
          output = 75;
        }
        analogWrite(enA, output);
        delay(h*1000-1);
        uhel1 = uhel2;                                //zapise vypoctene hodnoty do hodnot minulych
        rychlost1 = rychlost2;
        Serial.println(otacky*0.00314);
        Serial.println((millis() - startTime)/1000);
      }
      Serial.println(5000);

      start_kyvadlo = 0;
      digitalWrite(IN1, LOW);                         //vypne motor
      digitalWrite(IN2, LOW);
      analogWrite(enA, 0);
  }
}

void encoder_isr() {
    static int8_t lookup_table[] = {0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0};
    static uint8_t enc_val = 0;

    enc_val = enc_val << 2;
    enc_val = enc_val | ((PIND & 0b1100) >> 2);

    otacky = otacky + lookup_table[enc_val & 0b1111];
}

void zacatek(){
  if(digitalRead(Index) != HIGH){
    while(digitalRead(Index)  != HIGH){
      digitalWrite(IN1, HIGH);
```

```
    digitalWrite(IN2, LOW);
    analogWrite(enA, 120);
  }
  otacky = 0;
  analogWrite(enA, 0);
}
delay(500);
while(otacky >= 0){
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(enA, 40);
}
analogWrite(enA, 0);
delay(1000);
otacky = 0;
}
```

```
import serial
import time
import sys
import csv

startMarker = "<"
endMarker = ">"
nextVariable = ":"

g = sys.argv[1]
poc_uhel = sys.argv[2]
delka = sys.argv[3]
start = sys.argv[4]
COM = sys.argv[5]
jmeno = sys.argv[6]
kp = sys.argv[7]
ki = sys.argv[8]
kd = sys.argv[9]
offset = sys.argv[10]

arduino = serial.Serial(port=COM, baudrate=9600, timeout=0, rtscts=True)
time.sleep(3)

stringToSend = (startMarker)
stringToSend += str(g)
stringToSend += (nextVariable)
stringToSend += str(poc_uhel)
stringToSend += (nextVariable)
stringToSend += str(delka)
stringToSend += (nextVariable)
stringToSend += str(kp)
stringToSend += (nextVariable)
stringToSend += str(ki)
stringToSend += (nextVariable)
stringToSend += str(kd)
stringToSend += (nextVariable)
stringToSend += str(offset)
stringToSend += (nextVariable)
stringToSend += str(start)
stringToSend += (nextVariable)
stringToSend += (endMarker)
arduino.write(stringToSend.encode('utf-8'))

data = []
cas = []
time.sleep(10)
num = 0
pomocna = 0
while (num != 5000):
    line = arduino.readline()   # precte bitovy string
    if line:
        if pomocna == 0:
            string_n = line.decode()  # prevede byte string na unicode string
            string = string_n.rstrip() # odstraní ze stringu \n and \r
            pomocna += 1
            num = float(string)     # prevede string na float
            if num != 5000:
                data.append(num)    # vlozi hodnotu do pole data
        else:
            string_n = line.decode()
            string = string_n.rstrip()
            pomocna -= 1
            num = float(string) # prevede string na float
            num = num +1
            cas.append(num) # vlozi hodnotu do pole cas
            time.sleep(0.06)

arduino.close()

time.sleep(0.5)
cesta = 'data/'
cesta += jmeno
cesta += '.csv'
radek1 = False
poc_uhel = float(poc_uhel)*3.14/180

csvfile = open(cesta, mode="w", newline='')
time.sleep(1)
fieldnames = ["cas [s]", "uhel [rad]", "gravitacni zrychleni [m/s2]",
              "pocatecni vychylka [rad]", "delka vlakna [m]"]
writer = csv.DictWriter(csvfile, fieldnames=fieldnames, delimiter=';')
writer.writeheader()
```

```
for time, uhel in zip(cas, data):
    if radek1 == False:
        writer.writerow({"cas [s]": time, "uhel [rad]": uhel,
                         "gravitacni zrychleni [m/s2]":g,
                         "pocatecni vychylka [rad]":poc_uhel,
                         "delka vlakna [m]":delka})
        radek1 = True
    else:
        writer.writerow({"cas [s]": time, "uhel [rad]": uhel})
csvfile.close()
```

Příloha 3 – Kompletní nastavení routeru Mikrotik

Terminal <1>

```
[admin@CZU_Uloha_kyvadlo] > export
# may/15/2022 04:40:20 by RouterOS 7.1.1
# software id = P0RN-P547
#
# model = RB760iGS
# serial number = A81509A84422
/interface bridge
add name=bridge1 protocol-mode=none
/interface ethernet
set [ find default-name=ether1 ] name=ether1_Uplink
set [ find default-name=ether2 ] name=ether2_Pocitac
set [ find default-name=ether4 ] name=ether4_Kamera
set [ find default-name=sfp1 ] disabled=yes
/interface lte apn
set [ find default=yes ] ip-type=ipv4
/interface wireless security-profiles
set [ find default=yes ] supplicant-identity=MikroTik
/ip ipsec profile
set [ find default=yes ] dh-group=modp1024 enc-algorithm=3des nat-traversal=no
/ip ipsec proposal
set [ find default=yes ] enc-algorithms=3des
/ip pool
add name=dhcp_pool0 ranges=192.168.0.2-192.168.0.254
add name=vpn ranges=192.168.88.1-192.168.88.254
add name=dhcp_pool2 ranges=192.168.0.2-192.168.0.254
/ip dhcp-server
add address-pool=dhcp_pool2 interface=bridge1 name=dhcp1
/port
set 0 name=serial0
/ppp profile
add local-address=192.168.88.1 name=Ucitel remote-address=192.168.88.2
add local-address=192.168.88.3 name=Student remote-address=192.168.88.4
/routing bgp template
set default as=65530 disabled=no name=default output.network=bgp-networks
/routing table
add fib name=""
/snmp community
set [ find default=yes ] read-access=no
/user group
set full policy="local,telnet,ssh,ftp,reboot,read,write,policy,test,winbox,password,web,sniff,sensiti\
    ve,api,romon,dude,tikapp,rest-api"
/interface bridge port
add bridge=bridge1 ingress-filtering=no interface=ether2_Pocitac
add bridge=bridge1 ingress-filtering=no interface=ether4_Kamera
add bridge=bridge1 interface=ether3
add bridge=bridge1 interface=ether5
/ip neighbor discovery-settings
set discover-interface-list=!dynamic
/ipv6 settings
set max-neighbor-entries=8192
/interface l2tp-server server
set default-profile=Ucitel
/interface pptp-server server
set authentication=mschap2 enabled=yes
/ip address
add address=192.168.0.1/24 comment="Vnit\F8n\ED rozsah" interface=bridge1 network=192.168.0.0
add address=185.176.36.2/30 comment="Ve\F8ejn\E1 IP adresa" interface=ether1_Uplink network=\
    185.176.36.0
add address=192.168.88.1/24 comment="Rozsah pro vzd\E1len\E9 p\F8ipojen\ED" interface=bridge1 \
    network=192.168.88.0
/ip dhcp-server lease
add address=192.168.0.28 client-id=1:ec:71:db:54:39:32 comment=Kamera mac-address=EC:71:DB:54:39:32 \
    server=dhcp1
add address=192.168.0.29 comment=Pocitac mac-address=40:8D:5C:E3:BB:22 server=dhcp1
/ip dhcp-server network
add address=192.168.0.0/24 dns-server=8.8.8.8,1.1.1.1 gateway=192.168.0.1
/ip dns
set servers=8.8.8.8,1.1.1.1
/ip firewall nat
add action=masquerade chain=srcnat src-address=192.168.0.0/24
add action=masquerade chain=srcnat comment="massquerade pro VPN" src-address=192.168.88.0/24
/ip route
add comment="V\FDchoz\ED br\E1na" disabled=no distance=1 dst-address=0.0.0.0/0 gateway=185.176.36.1 \
    pref-src="" routing-table=main scope=30 suppress-hw-offload=no target-scope=10
/ip service
set telnet disabled=yes
set ftp disabled=yes
set ssh disabled=yes port=122
set api disabled=yes
set api-ssl disabled=yes
/ip ssh
set allow-none-crypto=yes forwarding-enabled=remote
/ppp secret
add name=Student profile=Student service=pptp
add name=Ucitel profile=Ucitel service=pptp
/snmp
```

Příloha 4 – Šablona pro protokoly z měření

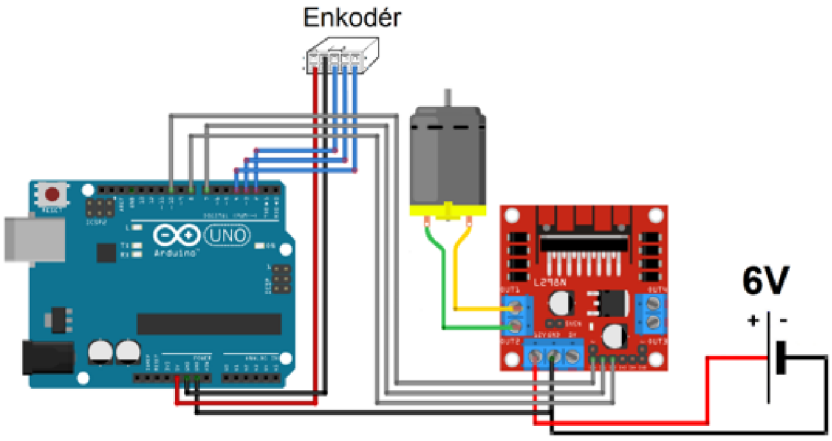
Jméno: Datum:

Protokol k úloze simulace nelineárního matematického kyvadla

1. Zadání:

1) Seznámení se se schématem zapojení úlohy a použitým hardwarem.
2) Nastudování pojmu matematické kyvadlo a zjištění rozdílu mezi lineárním a nelineárním matematickým kyvadlem.
3) Připojení se na uživatelské webové rozhraní a seznámení se s návodem pro úlohu.
4) Provést měření pro oba typy regulace pro 3 různé periody kmitu vzdálené od sebe aspoň 0,5 s, tak aby regulace byla co nejpřesnější.
5) Protokol bude obsahovat grafy pro všechny 3 periody obou regulací a vyhodnocení jejich regulací s nastavenými parametry při regulaci.
6) V protokolu bude dále popsán rozdíl mezi těmito regulacemi.

2. Schéma zapojení:



3. Princip použitých senzorů:

4. Postup měření:

5. Grafy regulací:

6. Vyhodnocení:

Příloha 5 – Fotky hotové úlohy