UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2024 **Bc. Jonáš Šafránek**

UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

LABORATORNÍ SOUSTAVA PRO VÝUKU AUTOMATIZACE

Bc. Jonáš Šafránek

Diplomová práce
2024

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

Studijní program: Automatické řízení
Forma studia: Prezenční

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení:	**Bc. Jonáš Šafránek**
Osobní číslo:	**I22196**
Téma práce:	**Laboratorní soustava pro výuku automatizace**
Téma práce anglicky:	**Laboratory system for automation education**
Jazyk práce:	**Čeština**
Vedoucí práce:	**Ing. Daniel Honc, Ph.D.** Katedra řízení procesů

Zásady pro vypracování:

Cíl práce: Cílem práce je navrhnout a vytvořit jednoduchou laboratorní soustavu "řízení otáček ventilátoru". Ovládání ventilátoru a měření otáček bude zajištěno pomocí mikrokontroléru Arduino. Model bude možné řídit z prostředí MATLAB.
Obsah teoretické části: Rešerše problematiky laboratorních modelů pro výuku automatizace, rodiny mikrokontroléru Arduino a možnosti programování a komunikace z prostředí MATLAB.

Obsah praktické části práce: Návrh a realizace modelu, naprogramování Arduina a vytvoření programu v MATLABu. Změření vybraných charakteristik a ukázky regulačních pochodů.

Seznam doporučené literatury:

VODA, Zbyšek. *Průvodce světem Arduina*. Vydání druhé. Bučovice: Martin Stříž, 2017. ISBN 978-80-87106-93-8.
SELECKÝ, Matúš. *Arduino: uživatelská příručka*. Přeložil Martin HERODEK. Brno: Computer Press, 2016. ISBN 978-80-251-4840-2.
ZAPLATÍLEK, Karel. *MATLAB® pro začínající uživatele*. Knihovnicka.cz. Brno: Tribun EU, 2020. ISBN 978-80-263-1589-6.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum:

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 3. 4. 2024

Bc. Jonáš Šafránek

PODĚKOVÁNÍ

Tímto děkuji vedoucímu mé práce Ing. Danielu Honcovi Ph.D. za pomoc, vstřícnost a konzultaci při tvorbě diplomové práce. Dále Bc. Michalu Šimonovi za 3D tisk konstrukce.
V neposlední řadě děkuji své široké rodině, pracovním kolegům, spolužákům a přítelkyni za podporu v průběhu tvorby této práce.

V Pardubicích dne 3. 4. 2024

Bc. Jonáš Šafránek

ANOTACE

Tato diplomová práce se zabývá tvorbou laboratorního modelu řízení otáček ventilátoru, využitelného při výuce automatizace. Teoretická část se zabývá komerčními laboratorními modely, rodinou mikrokontrolérů Arduino, možnostech programování v prostředí Arduino IDE a Matlab, a jejich vzájemné komunikaci. V praktické části je proveden návrh modelu, konstrukce a programového řešení v obou prostředích.

KLÍČOVÁ SLOVA

automatizace, laboratorní model, řízení otáček, Arduino

TITLE

LABORATORY SYSTEM FOR AUTOMATION EDUCATION

ANNOTATION

This thesis deals with the development of a laboratory model of fan speed control, useful in teaching automation. The theoretical part deals with commercial laboratory models, the Arduino microcontroller family, the possibilities of programming in the Arduino IDE and Matlab environment, and how they can communicate with each other. In the practical part, the model design, construction and programming solution in both environments is performed.

KEYWORDS

automation, laboratory model, fan speed control, Arduino

OBSAH

SEZNAM ZKRATEK A ZNAČEK

2D	dvojrozměrný
3D	trojrozměrný
AD	analogově-digitální
CAD	počítačem podporované projektování
EEPROM	elektricky mazatelná a programovatelná paměť pouze pro čtení
GND	elektrická zem
I/O	vstupně-výstupní
I2C	sběrnice pro vzájemné propojení integrovaných obvodů
KB	kilobajt
KHz	kilohertz
LQ	lineárně kvadratický
mA	miliampér
MHz	megahertz
NTC	termistor s negativním teplotním koeficientem
OTG	On-The-Go
PC	osobní počítač
PCB	tištěný spoj
PID	proporcionálně integračně derivační
PLC	programovatelný logický automat
PSD	proporcionálně sumačně diferenční
PWM	šířková modulace pulzu
RAM	operační paměť
RPM	otáčky za minutu

SMD	povrchová montážní součástka
SRAM	statická paměť s náhodným přístupem
USB	univerzální sériová sběrnice
V	volt
WiFi	bezdrátová síť

SEZNAM ILUSTRACÍ

ÚVOD

Automatizace je dynamicky rozvíjejícím se oborem. Tento obor je velmi komplexní a pro jeho jednodušší pochopení může být přínosné jednotlivé části vysvětlovat pomocí experimentů na reálném zařízení, se kterým je možná interakce. Praktické ukázky při výuce motivují studenty k hlubšímu pochopení dané problematiky.

Cílem práce je sestrojení jednoduchého laboratorního modelu řízení otáček ventilátoru, použitelného pro výuku automatizace. Řízený bude mikrokontrolérem Arduino, komunikace s uživatelem bude zprostředkovávána pomocí prostředí Matlab. Po realizaci návrhu modelu bude možné naměřit vybrané charakteristiky a regulační pochody.

1 LABORATRONÍ MODELY PRO VÝUKU AUTOMATIZACE

Laboratorních modelů pro výuku automatizace existuje celá řada. Zpravidla jde o složité modely a konstrukce různých soustav, které lze ovládat a regulovat z různých uživatelských prostředí. Jejich ceny na trhu se pohybují ve vysokých desítkách až stovkách tisíc korun, v závislosti na složitosti daného modelu, k sehnání jsou však i začínající projekty menších firem či skupin, které plní podobné funkce, ale jsou dostupnější.

Pro výuku automatizace na středních i vysokých školách mohou být vhodné modely, u kterých je možné nastavovat manuálně akční veličinu a zobrazovat měřené hodnoty regulované veličiny, díky čemuž uživatel může provést identifikaci soustavy. Následně se může uživatel zabývat například nastavením parametrů PID, respektive PSD regulátoru, různými metodami a sledovat vliv jednotlivých parametrů na průběh a kvalitu regulace. Další pozorovatelný vliv na regulaci PSD regulátorem má perioda vzorkování, se kterou se regulace provádí. Už díky těmto základním možnostem ovládání modelů, které uživateli umožní zobrazit průběhy veličin, může být vysvětlování i pochopení problematiky na praktické ukázce efektivnější a pro výuku velmi přínosné.

1.1 AUTOMATIONSHIELD

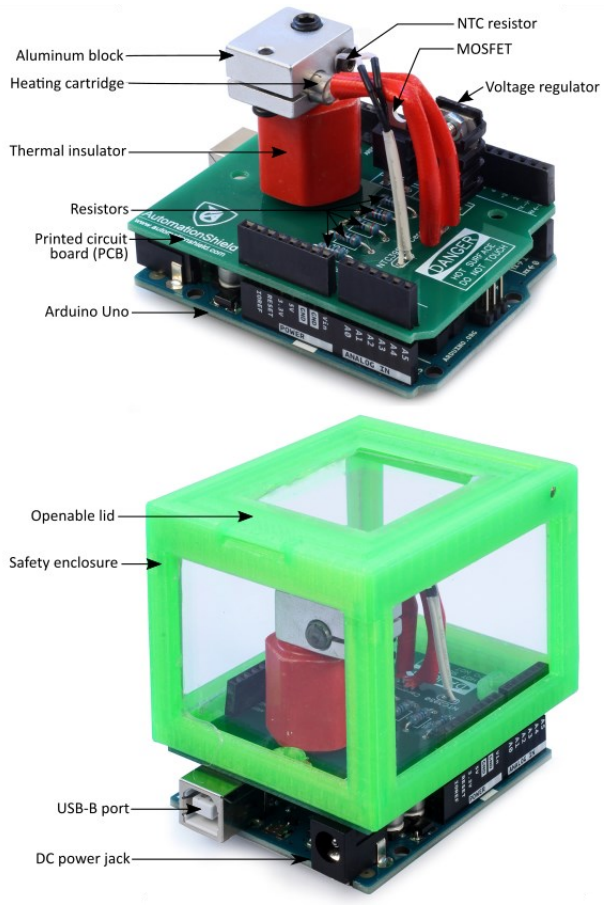
AutomationShield je open-source skupina vytvářející nástroje pro výuku automatizace, řídicí mechaniky a mechatroniky. Její snahou je tvorba laboratorních modelů v co nejdostupnější formě. Vytváří různé modely, všechny mají společné menší rozměry, a oproti ostatním velkým firmám také lepší finanční dostupnost.

Modely tvořené touto skupinou jsou založené na mikrokontrolérech Arduino, velmi často se dokonce jedná o PCB desky, které jsou kompatibilní přímo s danými rozměry Arduina a jedná se tedy o tzv. shieldy, které lze připojit přímo do desky.

Tvůrci uvádí, že nemají kapacity pro komerční výrobu svých shieldů. I to je jeden z důvodů, proč je projekt open-source, tedy, že tvůrci poskytují veškerá data volně na internetu. Jedná se o CAD soubory pro desky plošných spojů, seznamy potřebných součástek, 3D modely pro tisk, kódy a další (Takács, nedatováno).

1.1.1 HeatShield

Jedná se o soustavu vyráběnou skupinou AutomationShield přizpůsobenou přímo pro mikrokontrolér Arduino Uno REV1. Tento štít slouží pro regulaci teploty zahřívacího bloku pro 3D tiskárny. Teplo je do bloku předáváno z rezistivního ohřívače a zpětná vazba, tedy teplota, je měřena pomocí NTC termistoru.



Obr. 1.1 – Model HeatShield (Takács, nedatováno)

Jde o regulaci teploty, tedy soustava disponuje většími časovými konstantami, není tak pro výuku tak účelná, jako soustavy s menšími časovými konstantami. Zajímavé však může být porovnání právě s jinými shieldy.

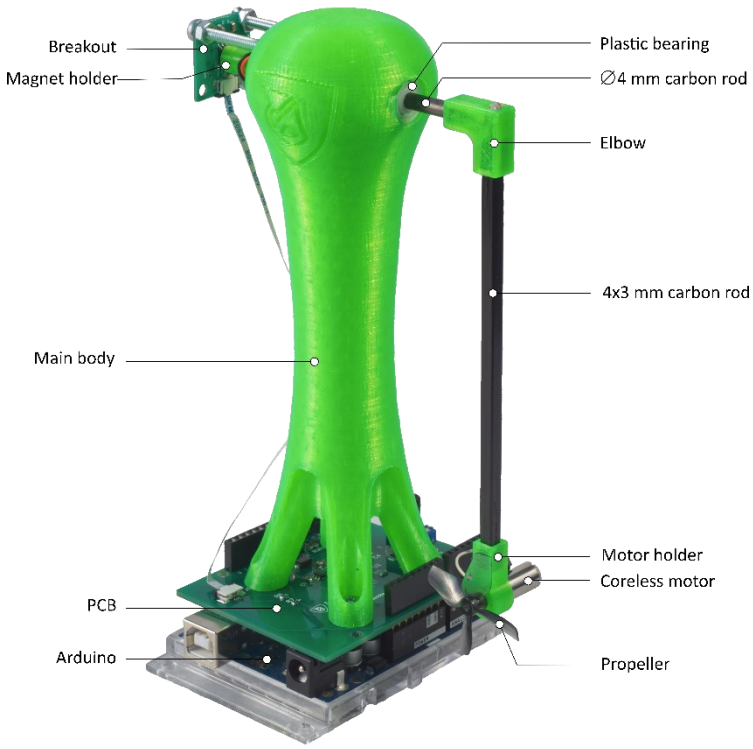
K shieldu je dostupná kompletní dokumentace, včetně odkazů na jednotlivé využité díly. HeatShield je, stejně jako většina ostatních, ovladatelný z prostředí Matlab a Simulink díky vytvořeným funkcím a zdrojovým kódům pro mikrokontrolér. Funkce jsou navrhnuty pro nastavování PID regulátoru a identifikaci soustavy. Komunikace mezi mikrokontrolérem

a aplikací Matlab je zařízena díky podpoře pro mikrokontroléry Arduino. Odhadovaná cena modelu bez mikrokontroléru se pohybuje ve stovkách korun.

1.1.2 AeroShield

Jde o kyvadlo, jehož úhel natočení je možné ovládat pomocí miniaturního stejnosměrného motoru s vrtulí, využívaného například pro drony. Úhel natočení je snímán Hallovým senzorem. Tah motoru uživatel může ovládat manuálně pomocí potenciometru nebo z prostředí Matlab či Simulink regulací podle předdefinované trajektorii natočení úhlů. Tento shield je navrhnut pro Arduino Uno REV3 a umožňuje totožné akce jako shield HeatShield s přidanou funkcí LQ regulace.

U tohoto modelu se odhadovaná cena bez mikrokontroléru pohybuje ve vyšších stovkách korun (Takács, nedatováno).



Obr. 1.2 – Model AeroShield (Takács G, nedatováno)

1.2 GUNT

Laboratorní modely od firmy GUNT jsou navrženy tak, aby poskytovaly praktické a interaktivní zkušenosti pro studenty v oblasti automatizace a procesního inženýrství.

Tyto modely umožňují studentům simulovat a analyzovat různé průmyslové procesy a systémy. Modely umožňují provádět experimenty v bezpečném a kontrolovaném prostředí, což umožňuje studentům lépe pochopit dynamiku a reakce systémů na různé podmínky bez rizika poškození drahého průmyslového zařízení. Ceny modelů nejsou volně dostupné, avšak dle informací z fakulty Elektroniky a informatiky Univerzity Pardubice, která těmito modely disponuje, se ceny základních modelů pohybují ve vysokých desítkách tisíc korun. K modelům je dodáván software, pomocí kterého je lze ovládat. Byly také vytvořeny speciální S-funkce jako uživatelský blok v Simulinku, díky kterým je možné komunikovat, ovládat a řídit modely přímo z prostředí Matlab a Simulink (GUNT, nedatováno; Honc a Dušek, 2013).

1.2.1 RT 050 Model řízení rychlosti systému

Model RT 050 umožňuje provádět základní experimenty na systému řízení rychlosti. Jako řízený systém se používá hřídel se setrvačníkem. Otáčky představují řízenou veličinu, která je určena měřicím prvkem, v tomto případě indukčním snímačem otáček. Výstupní signál ze snímače je přiváděn do softwarového regulátoru. Výstupní signál z řídicí jednotky ovlivňuje akční člen, v tomto případě motor, který způsobuje otáčení hřídele. Ke generátoru na hřídeli jsou připojeny přepínatelné rezistory jako zátěž. Pomocí softwaru je tedy možné měnit zátěže, aby bylo možné sledovat vliv rušivých veličin. Odezva řízení se zobrazuje ve formě časové závislosti. Na setrvačníku je umístěn ukazatel otáček, který umožňuje kdykoli přímo odečíst otáčky. Průhledný ochranný kryt umožňuje bezpečné sledování experimentů. K ovládání a řízení je nutné disponovat osobním počítačem pro USB propojení (GUNT model RT 050, nedatováno).

Obdobné modely GUNT produkuje pro regulaci hladiny, teploty, pozice, průtoku a tlaku. Každý z těchto modelů se liší svými vlastnostmi a může být vhodný pro praktické ukázky v různých situacích.

Obr. 1.3 – Model RT 050 (GUNT model RT 050, nedatováno)

1.2.2 RT 624 Demonstrativní model regulace průtoku

Řada modelů RT 45X se zabývá řízením modelů pomocí PLC, řada RT6X4 se pak zabývá řízením demonstrativních modelů soustav.

Model RT 624 je experimentální jednotkou poskytující ucelený experimentální úvod do základů řídicí techniky na příkladu řízení průtoku. Všechny součásti soustavy jsou přehledně uspořádány na svislém čelním panelu.

Řízeným systémem je úsek potrubí, kterým se čerpá voda. Potrubní úsek obsahuje snímač s lopatkovým kolem jako měřicí prvek, který zaznamenává průtok jako regulovanou veličinu. Průhledný rotametr umožňuje velmi přehledně sledovat proces řízení. Použitý je moderní digitální průmyslový regulátor. Akčním členem v regulační smyčce je elektrický regulační ventil. Kulový ventil v potrubní části umožňuje generovat definované rušivou veličinu. Řízenou veličinu a akční veličinu lze získat jako analogové signály na výstupních konektorech modelu. To umožňuje připojení externího záznamového zařízení, jako je

osciloskop. K modelu je dodáván přístrojový a řídicí software s modulem rozhraní USB. Ten umožňuje zobrazovat klíčové procesní proměnné a provádět řídicí funkce.

Další modely z této řady umožňují řízení soustav tlakových, teplotních či řízení hladiny, vše na demonstrativních modelech. Na základě větší komplexnosti a složitosti těchto modelů, oproti řadě RT0X0 se dá očekávat, že se cena za jeden model bude pohybovat ve vysokých desítkách až stovkách tisíc korun (GUNT model RT 624, nedatováno).

Obr. 1.4 – Model RT 624 (GUNT model RT 624, nedatováno)

1.3 QUANSER

Firma Quanser nabízí škálu edukativních modelů pokrývající širokou škálu oblastí a složitostí. Modely simulují chování reálných systémů, což umožňuje výuku řízení systémů v reálném čase. K modelům je dodáván software umožňující soustavy řídit a ovládat. K vybraným modelům také existují softwarová prostředí, které pouze simulují daný model.

1.3.1 Aero 2

Jedná se o model dvou motorů s vrtulemi na tyči. Jedná se o kyvadlo s možností pohybu kolem horizontální osy o 360° a vertikální osy o ± 45°. Oba úhly natočení jsou snímány senzory s vysokým rozlišením. Měření polohy a rychlosti pomáhá zpřesnit i integrovaná inerciální měřicí jednotka. Vrtule ventilátoru může uživatel libovolně natočit, tím tak změní vlastnosti systému. Jednotlivé osy jsou také možné uzamknout, čímž uživatel může snížit komplexnost modelu nebo ji naopak zvýšit.

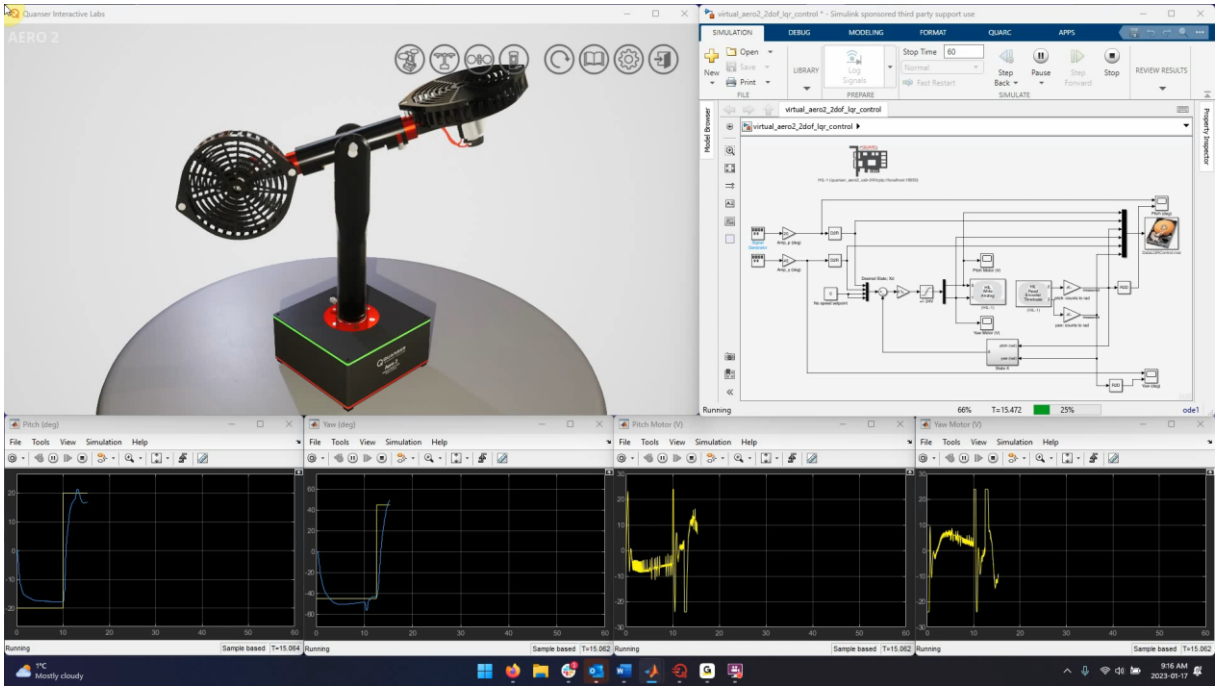
Dodávaným softwarem dokážeme tedy model řídit a ovládat. Nabízí řízení například PID a LQ regulátorem. Existuje také toolbox pro Matlab a Simulink, ze kterého lze také provádět operace s modelem. Podobnou podporou disponuje také Arduino, kterým lze model přímo ovládat a řídit. Nejde však o všechny platformy, které takovou podporou disponují.

Cena modelu včetně softwaru se pohybuje v nižších stovkách tisíc korun (Aero 2, nedatováno).

Obr. 1.5 – Model Aero 2 (Model Aero 2, nedatováno)

1.3.2 QLabs Virtual Aero2

U některých modelů, firma Quanser nabízí také jejich kompletní virtuální repliky. QLabs Virtual Aero 2 je virtuální platforma pro simulaci reálného modelu Aero 2 včetně jeho vizualizace. Dynamické vlastnosti reálného modelu byly popsány rovnicemi, a ty jsou využity pro jeho simulaci. Jeho řízení je možné pomocí programu Matlab, Simulink a dalších vývojových softwarů. Simulace je velmi složitá, je tedy nutné disponovat výkonným osobním počítačem. Konkrétní hardwarové nároky jsou pro model specifikovány na stránkách výrobce. Virtuální model umožňuje veškeré operace jako s reálným modelem, například modelování, identifikace systému, řízení polohy a rychlosti, PID řízení, řízení stavovým regulátorem a další. Dostupný je za roční předplatné okolo 35 tisíc korun, nebo jednorázovou platbu cca 120 tisíc korun (Virtual Aero 2, nedatováno).



Obr. 1.6 – Virtuální model Aero 2 (Virtual Aero 2, nedatováno)

1.4 TECQUIPMENT

Společnost TecQuipment se zabývá mimo jiné tvorbou modelů pro výuku automatizace. Nabízí škálu pokročilých modelů umožňujících experimenty v oblasti různých regulací. Mezi modely patří systémy pro regulaci teplot, hladin, tlaků, průtoku a další. Modely jsou dodávány s vlastním softwarem, ve kterém lze nastavovat jejich parametry, zobrazovat data, ovládat a řídit model a další. Některé modely je možno ovládat manuálně skrze ovládací panely nebo prostředí Matlab. Ceny se v závislosti na modelu pohybují od nízkých po vysoké stovky tisíc korun (TecQuipment, nedatováno).

Obr. 1.7 – Model řízení teploty firmy TecQuipment (TecQuipment model řízení teploty, nedatováno)

2 RODINA MIKROKONTROLÉRŮ ARDUINO

V rodině Arduino se nachází jednodeskové počítače v množství různých provedení, založených převážně na mikroprocesorech ATmega. Jednotlivé typy se od sebe odlišují například velikostí, počtem I/O pinů, konektivitou, napájecím napětím nebo dalšími specifikacemi či úpravami dané desky. Velkou část funkcí má většina desek společnou, díky stejným či velmi podobným osazeným mikroprocesorům. Většina desek disponuje funkcemi generování PWM signálu, AD převodníku, využitím čítačů a časovačů, softwarovým a hardwarovým přerušením, nebo analogovým komparátorem hodnot. V základu tedy pomocí periferií desky a programové aplikace umožňují tyto mikrokontroléry zpracovávat a vytvářet různé signály. Vše je ve velmi jednoduché a uživatelsky přívětivé formě.

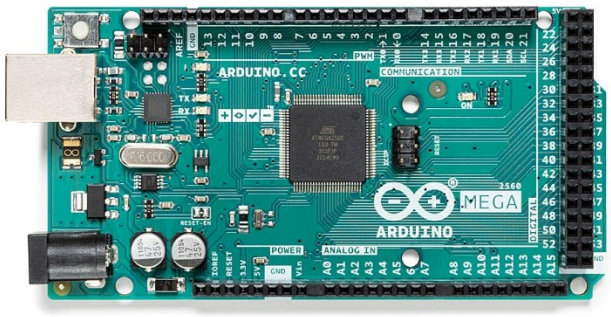
V oficiálním internetovém obchodě se v nabídce nachází k těmto deskám různé příslušenství, jako jsou například kabely, senzory, moduly, redukce, displeje a další. Rozšiřující moduly se mohou nazývat shieldy. Cena mikrokontrolérů se pohybuje v závislosti na komplexnosti a vlastnostech daného typu běžně mezi vyššími stovkami až nízkými tisíci korun.

Desky i příslušenství lze sehnat i na jiných než oficiálních stránkách, v tom případě jde však o klony. Klony bývají zpravidla levnější, jejich zpracování je však povětšinou slabší a zvláště u posledních generací desek již nemusí nabízet naprosto totožnou možnost využití jako originál daného typu. Neoficiální verze desek a příslušenství značí velkou popularitu této platformy. Velká komunita vytvořila obrovské množství projektů a podpory v podobě knihoven zjednodušení vypracovávání kódů aplikací (Arduino, nedatováno).

2.1 ARDUINO MEGA2560 REV3

Jak napovídá název, tato deska je osazena procesorem ATmega2560 s frekvencí 16 MHz. Obsahuje paměť flash o velikosti 256 kB. Velikost paměti SRAM je 8 kB a EEPROM 4 kB. Na desce se nachází 54 digitálních pinů, 15 z nich umožňuje výstup PWM. Maximální odebíraný proud z jednoho vstupně-výstupního pinu je 20 mA. Analogových pinů je 16. Z pinu určeného k napájení 5 V je možné odebírat až 800 mA, pokud je deska napájena adaptérem, v případě napájení USB není technicky možné takového výstupního proudu dosáhnout, USB poskytuje maximální napájecí proud 500 mA. Při odběrech větších, než je uvedeno, je velmi pravděpodobné, že bude poškozena. Na desce se nachází diody informující uživatele o tom, zda je deska napájena a zda přijímá či odesílá informace. Zároveň také dioda, kterou lze programově

ovládat. V neposlední řadě se na desce nachází tlačítko reset, hned vedle USB konektoru, díky kterému lze s deskou komunikovat i ji napájet. Doporučená napájení mikropočítače je mezi 7 a 12 V. Tohoto napětí logicky nemůžeme dosáhnout pomocí USB, to disponuje pouze 5 V. Větší napájení lze desce dodat konektorem pro externí napájení (Arduino MEGA2560 REV3, nedatováno).



Obr. 2.1 – Deska Arduino MEGA2560 REV3 (Arduino MEGA2560 REV3, nedatováno)

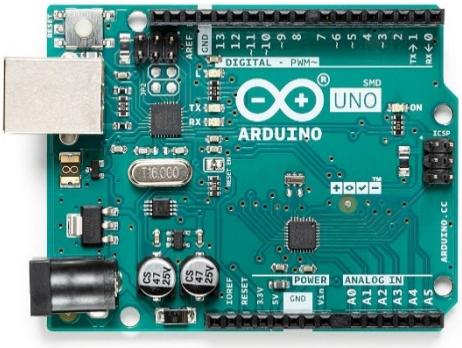
2.2 ARDUINO UNO REV3

Uno je nejrozšířenější verzí z rodiny. Je menší než Arduino MEGA. Procesor desky je ATmega328P s frekvencí 16 MHz. Má 14 vstupně-výstupních pinů, z nichž lze použít 6 pro PWM, 6 pinů analogových. Arduino Uno má menší paměti. 32 kB paměti flash, 1 kB paměti EEPROM a 2 kB paměti SRAM. Hlavním rozdílem od desky mega jsou tedy rozměry, váha, počet pinů a velikost paměti (Arduino Uno REV3, nedatováno).

Obr. 2.2 – Deska Arduino Uno REV3 (Arduino Uno REV3,

2.2.1 Arduino Uno Rev3 SMD

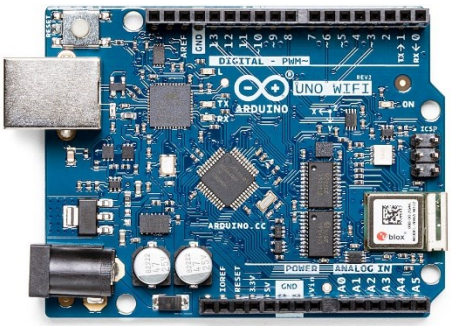
SMD v názvu tohoto typu desky poukazuje na provedení procesoru. Desky jsou téměř totožné. Rozdíl je pouze v procesoru. Tato verze disponuje procesorem ATmega328 v provedení SMD (Arduino Uno REV3 SMD, nedatováno).



Obr. 2.3 – Deska Arduino Uno REV3 SMD (Arduino Uno REV3 SMD, nedatováno)

2.2.2 Arduino Uno WiFi REV2

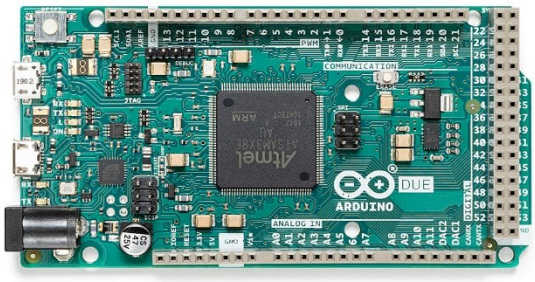
Deska je osazena novějším procesorem ATmega4809 SMD, provedení s frekvencí 16 MHz. Oproti předešlým deskám lze ze 14 vstupně-výstupních pinů pro PWM využít pouze pinů 5. Mikrokontrolér obsahuje modul pro komunikaci skrze WiFi a Bluetooth a modul s digitálním 3D akcelerometrem a gyroskopem (Arduino Uno WiFi REV2, nedatováno).



Obr. 2.4 – Deska Arduino Uno WiFi REV2 (Arduino Uno WiFi REV2, nedatováno)

2.3 ARDUINO DUE

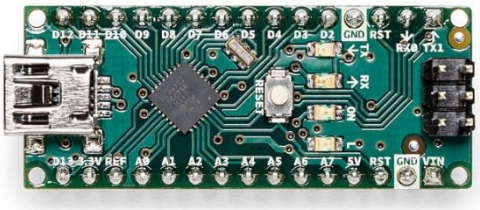
Deska disponuje 32-bitovým Atmel SAM3X8E ARM Cortex-M3 procesorem s frekvencí 84 MHz. Deska je větší, stejně jako deska MEGA2560 má 54 vstupně-výstupních pinů, z nichž lze využít 12 pro PWM výstup, 12 vstupů analogových, širší možnosti konektivity, jako například USB s podporou OTG. Na rozdíl od ostatních desek Arduino je operační napětí 3,3 V, což znamená, že připojení vyššího napětí na vstupně-výstupní piny může desku nenávratně poškodit (Arduino Due, nedatováno).



Obr. 2.5 – Deska Arduino Due (Arduino Due, nedatováno)

2.4 ARDUINO NANO

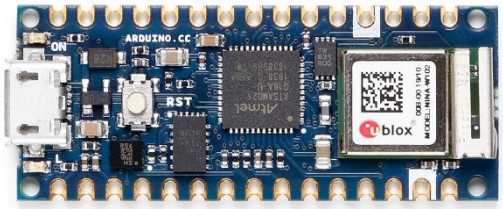
Je základem zmenšených desek rodiny Arduino. S procesorem ATmega328 s frekvencí 16 MHz, 22 vstupně-výstupními piny, 8 analogovými piny, operačním napětím 5 V je deska například velmi podobná a chybí jí jen minimum funkcí, kterými disponuje například Arduino Uno. Deska je uzpůsobena na práci na nepájivém poli, k dostání je i verze bez nožiček. Postrádá konektor pro napájení, napájení je možné pouze pomocí Mini-B USB. Velikost plošného spoje desky je 18 x 45 mm a váží pouze 7 g, což umožňuje snadné zakomponování do menších projektů (Arduino Nano, nedatováno).



Obr. 2.6 – Deska Arduino Nano (Arduino Nano, nedatováno)

2.4.1 Arduino Nano 33 IoT

S operačním napětím 3,3 V a integrovaným modulem pro komunikaci skrze WiFi a Bluetooth se jedná velmi zjednodušeně o zmenšeninu Arduino Uno WiFi. Deska má zároveň integrovaný šestiosý akcelerometr (Arduino Nano 33 IoT, nedatováno).



Obr. 2.7 – Deska Arduino Nano 33 IoT (Arduino Nano 33 IoT, nedatováno)

2.4.2 Arduino Nano 33 BLE

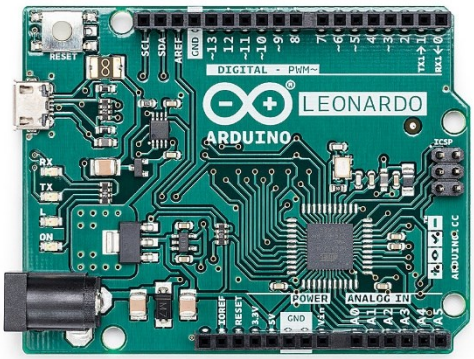
Deska je osazena procesorem s operačním napětím 3,3 V a její hlavní výhodou je integrovaný devítiosý akcelerometr. Tento senzor dělá z desky ideální volbu pro využití na nositelné zařízení jako jsou například chytré hodinky apod. K dispozici zde je k tomuto účelu i pomocný čip pro připojení Bluetooth. Deska je piny ekvivalentní k základnímu Arduino Nano, které však pracuje na 5 V operačním napětí (Arduino Nano 33 BLE, nedatováno).

2.4.3 Arduino Nano Every

Jde o vylepšenou verzi Arduino Nano. Je také založena na 5 V operačního napětí, ale disponuje silnějším procesorem ATMega4809, umožňujícím provádět složitější programy díky o 50 % větší programové paměti, než má Arduino Uno, s mnohem více proměnnými díky o 200 % větší RAM (Arduino Nano Every, nedatováno).

2.5 ARDUINO LEONARDO

Mimo rozdílný počet pinů je rozdílem od ostatních desek opět procesor. ATmega32u4 má vestavěnou USB komunikaci, není tedy potřeba, aby deska byla osazena dalším podpůrným procesorem, jako byly desky předešlé. Díky tomu se deska může jevit počítači jako klávesnice a myš a provádět tedy jiné operace než jiné desky (Arduino Leonardo, nedatováno).



Obr. 2.8 – Deska Arduino Leonardo (Arduino Leonardo, nedatováno)

3 MOŽNOTI PROGRAMOVÁNÍ

Díky znalostem a možnostem programování v prostředí pro Arduino a Matlab byly zvoleny právě tyto platformy. Obě jsou využitelné k široké škále projektů, hardwaru a bylo na nich zpracováno velké množství projektů. Prostředí umožňují veškeré operace potřebné ke splnění cílů práce.

3.1 ARDUNO IDE

Arduino IDE je volně dostupné prostředí určené pro tvorbu programů pro desky Arduino. V této chvíli je aktuální verze 1.8.19 podporována všemi běžně využívanými operačními systémy osobních počítačů. Jazyk, ve kterém jsou v tomto prostředí aplikace psány, je založen na jazyce C a C++.

Na obr. 3.1 Se nachází aplikace Arduino IDE. V hlavním panelu v horní části okna se nachází tlačítka pro správu knihoven, volbu programované desky, kam se má nahrávat vytvořený kód, spouštění sériového monitoru, správu projektu a další.

Pod hlavním panelem je panel nástrojů. Tlačítko „Ověřit“ provede testovací kompilaci programu, po které se vám informace o průběhu kompilace vypíše do stavového řádku ve spodní části obrazovky. Tlačítko „Nahrát“ slouží k odeslání a nahrání aplikace do vybrané desky skrze USB. I v tomto případě se informace o průběhu vypíše do stavového řádku. Pomocí zbylých tlačítek se vytváří nový, ukládá a otevírá projekt.

Vytvořený program musí obsahovat vždy minimálně dvě základní funkce. Na začátek okna se zpravidla deklarují proměnné a vkládají knihovny. Funkce „void setup()“ se provede pouze jednou při spuštění daného kódu, tedy po restartu desky. Funkce „void loop()“ se po první funkci vykonává stále dokola až do dalšího vypnutí či restartu. Za těmito funkcemi mohou být další, uživatelem definované funkce volané z jiných částí programu.

Knihovny slouží k ulehčení práce a zjednodušení kódu. Zpravidla umožňují práci s dalšími funkcemi, které před uživatelem již někdo definoval, a pokud by byly definovány v kódu samotném, mohly by zabírat i desítky řádků. Knihovny lze instalovat přímo z prostředí IDE v menu projekt. Zde se nachází možnost přidat knihovnu, odkud lze vkládat instalované knihovny do aktuálního kódu, případně nenainstalované knihovny vyhledávat a rovnou instalovat. Instalace knihoven lze provést i v případě, že jsou stáhnuty z internetu. V takovém

případě stačí složku knihovny vložit do složky knihoven Arduina v dokumentech využívaného počítače (Šafránek, 2022).

Obr. 3.1 – Okno prostředí Arduino IDE

3.2 MATLAB

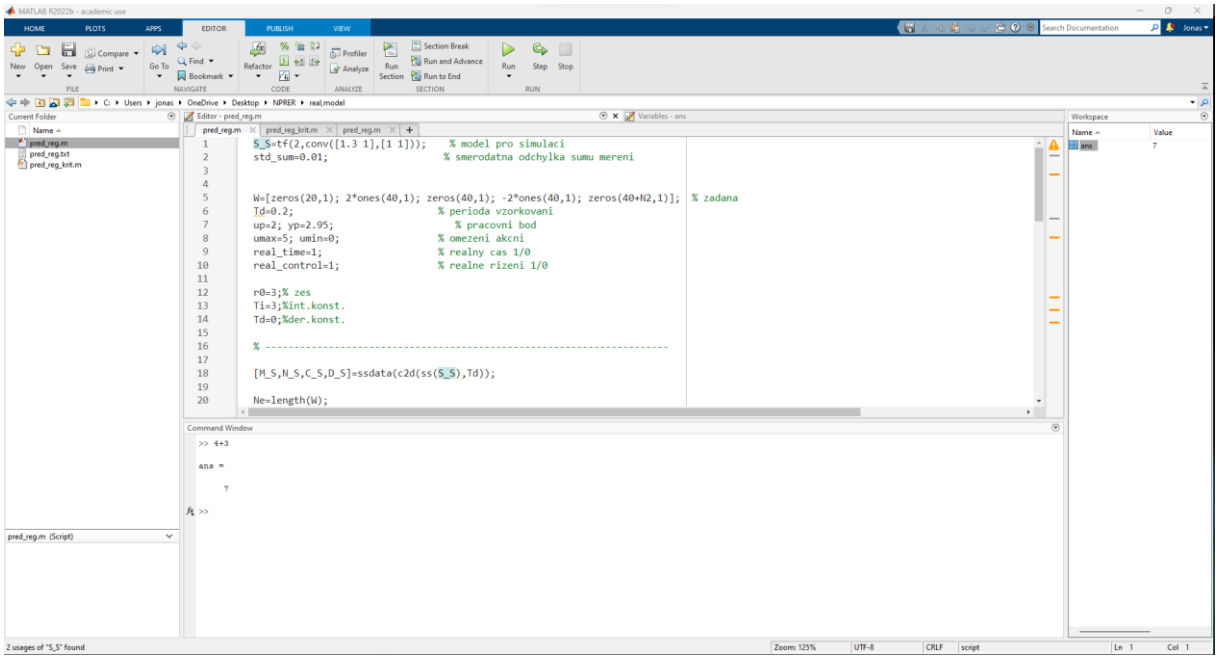
Matlab je interaktivní prostředí vyvíjené společností MathWorks. Jde o nástroj umožňující výpočty, analýzu dat, jejich vizualizaci, vývoj algoritmů a další. Aktualizace verzí probíhá zpravidla dvakrát za rok, momentální verzí je tedy R2024a. Podpora je zajištěna pro všechny běžně používané operační systémy PC. Hlavními přednostmi tohoto prostředí je velmi jednoduchá práce s maticemi, umožnění výpočtů s nimi, vykreslování dvojrozměrných i trojrozměrných grafů, dále například počítačová simulace, díky připojeném vizuálnímu prostředí Simulink a vytváření aplikací s uživatelským rozhraním.

Jednotlivé příkazy kódu lze psát přímo do příkazového okna, základně umístěného ve spodní části obrazovky, veškeré výsledky výpočtů se následně uloží do pracovního prostoru, základně v pravé části obrazovky. Pokud nenadefinujeme jinak, výsledek se uloží automaticky do proměnné *ans* do matice 1x1 datového typu double.

Pro tvorbu složitějších kódů, jejich spouštění po částech například v případě potřeby ladění se využívají tzv. m-file. Soubory *.m, ty mohou obsahovat definice funkcí, tříd, nebo čistě kód, který se má vykonat. Pokud m-file obsahuje funkci, jeho název musí korespondovat

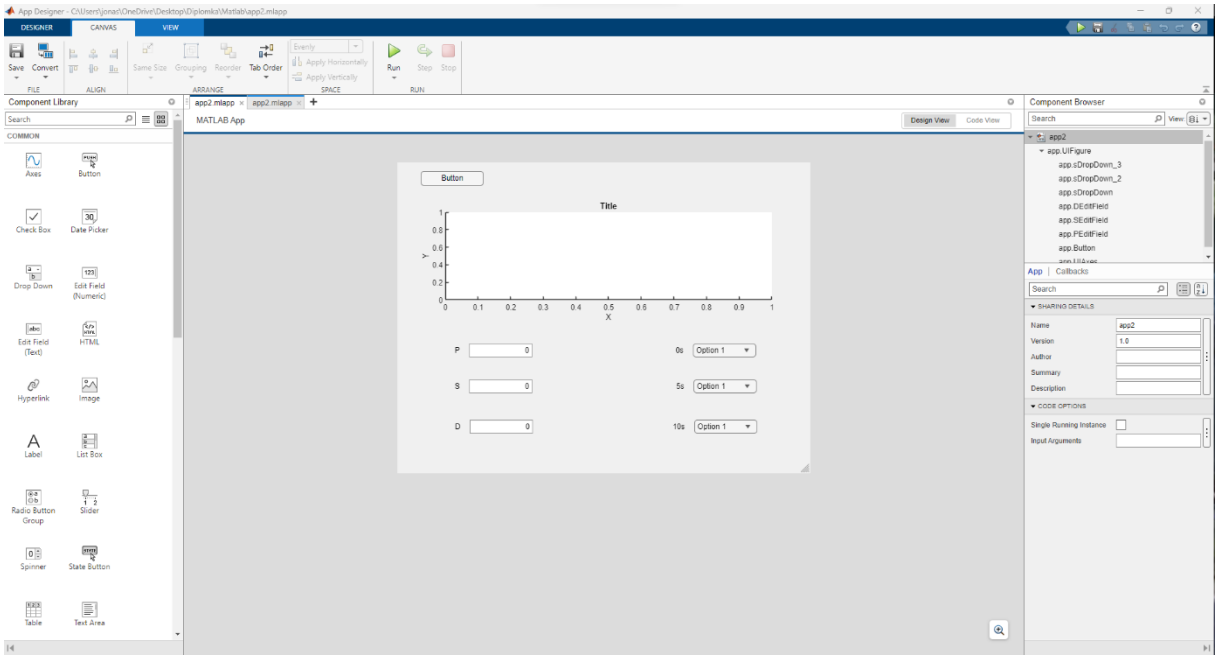
právě s názvem funkce. U funkce je nutné nastavit vstupní parametry a návratové hodnoty. Středníky v kódu slouží k zabránění výpisu na obrazovku.

Velmi důležitou součástí jsou další části samotného programu Matlab. Jedná se především o toolboxy a prostředí Simulink. Toolboxy rozšiřují možnosti výpočtů v určitých oborech matematiky, statistiky a mnoho dalších převážně technických odvětvích. Simulink slouží převážně k simulaci dynamických systémů. Jedná se z velké části o vizuální znázornění jednotlivých funkcí, ze kterých lze následně jejich propojením složit komplexní schéma. Programy jsou však intuitivně propojeny, a tak se dají využívat hodnoty z pracovního prostředí Matlab v prostředí Simulink a naopak. Zároveň může být u složitějších funkcí nutné doplnit některé bloky o části kódu z jazyku Matlab (HUMUSOFT, nedatováno; Matlab, nedatováno).



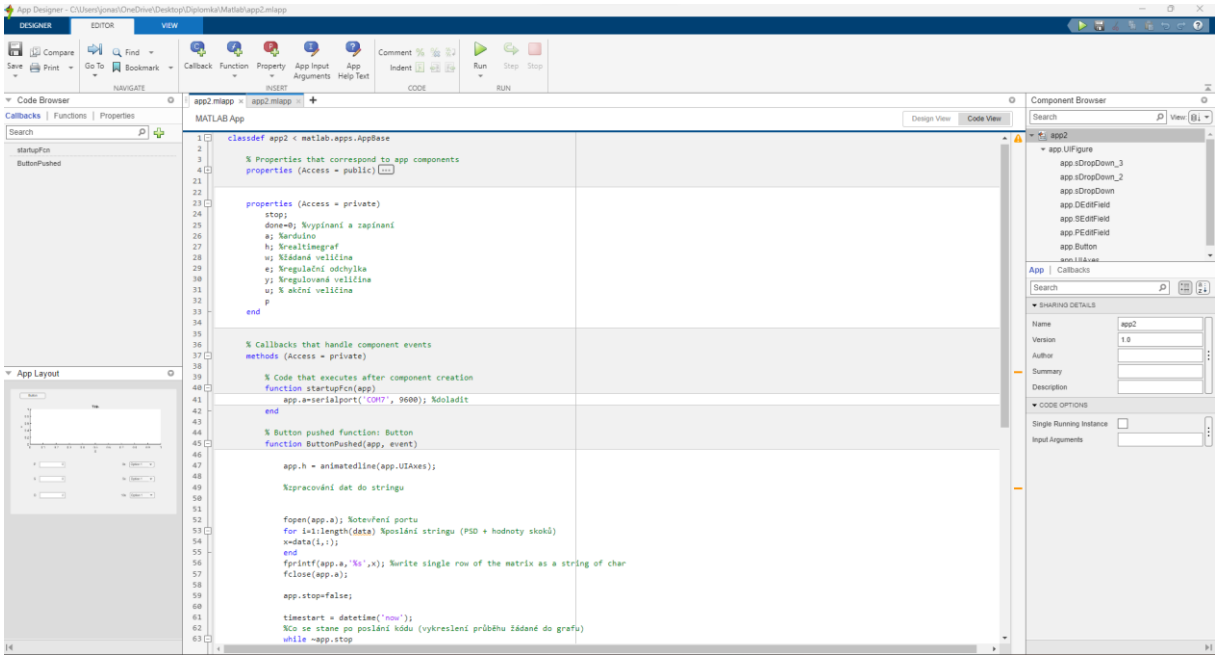
Obr. 3.2 – Okno prostředí Matlab

Pro vytváření aplikací je zde také možnost tvorby grafického rozhraní ke kódu, jehož vytváření je velmi intuitivní. Tato část Matlabu se nazývá App Designer. Po otevření se zobrazí okno, v jehož levé části je knihovna komponentů grafických, které je možné myší přetáhnout do pracovního prostoru ve střední části okna. V pravé části okna je výběr vložených komponentů. Po kliknutí na daný objekt je možno upravit jeho pozici, velikost, styly textu a další možnosti. Nechybí ani možnost vytvoření callback funkce, tedy funkce, která se vykoná po určité interakci s objektem, jako je třeba stisk tlačítka, nebo změna hodnoty v číselném poli.



Obr. 3.3 – Grafické okno prostředí AppDesigner

Tyto funkce nebo například funkce, která se vykoná po spuštění, se pak zobrazí v kódové části, kde je již práce s kódem obdobná, jako ve výše uvedených m-filech.



Obr. 3.4 – Okno textové části prostředí AppDesigner

3.3 VZÁJEMNÁ KOMUNIKACE

Komunikace mezi Arduinem a Matlabem je možná pomocí sériové linky, bluetooth modulu nebo sběrnice I2C. Pro potřeby řešené úlohy byla využita sériová linka zprostředkovaná propojením počítače a desky USB kabelem.

Sériová linka je způsob přenosu dat mezi dvěma zařízeními, která si posílají jednotlivé bity (0 nebo 1) po jednom vodiči. Každé zařízení má dva piny pro sériovou komunikaci: Rx (receive = přijmout) a Tx (transmit = odeslat). Tyto piny se propojí tak, že Rx jednoho zařízení se připojí na Tx druhého a naopak. Také je potřeba spojit zem (GND) obou zařízení.

Pro komunikaci mezi Arduinem a Matlabem je potřeba mít na obou stranách stejnou rychlost přenosu (baud rate), která se v Arduinu nastaví funkcí Serial.begin(). Například, pokud chceme komunikovat rychlostí 115200 bitů za sekundu, napíšeme Serial.begin(115200). Dále je potřeba mít stejný formát dat (data bits, parity, stop bits), který se obvykle používá 8N1, což znamená 8 bitů dat, žádná kontrola parity a 1 stop bit.

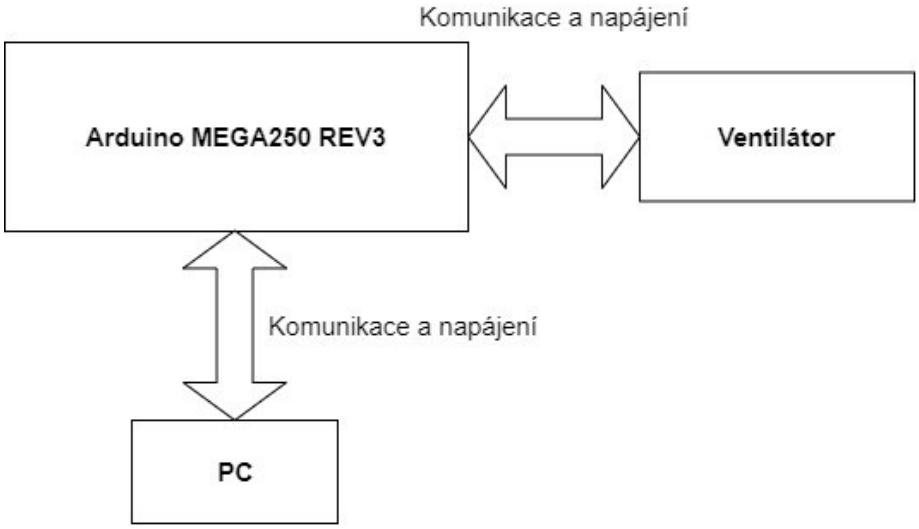
Pro odeslání dat z Arduina do Matlabu se používá funkce Serial.print(), která přijímá jako argument řetězec, číslo nebo jinou proměnnou. Například, pokud chceme odeslat hodnotu proměnné *x*, napíšeme Serial.print(x). Pro přijetí dat z Matlabu do Arduina se používá funkce Serial.read(), která vrací jeden byte dat, který byl přijat. Pokud nebyla přijata žádná data, vrací -1. Před použitím této funkce je vhodné zkontrolovat, jestli jsou nějaká data k dispozici funkcí Serial.available(), která vrací počet bytů, které čekají na přečtení.

Pro odeslání dat z Matlabu do Arduina se používá funkce writeline(), která přijímá jako argument objekt sériového portu a řetězec, číslo nebo jinou proměnnou. Například, pokud chceme odeslat hodnotu proměnné *y*, napíšeme writeline(s, y), kde *s* je objekt sériového portu. Pro přijetí dat z Arduina do Matlabu se používá funkce readline(), která přijímá jako argument objekt sériového portu a vrací řetězec dat. Například, pokud chceme přijmout celé číslo, napíšeme z = str2num(readline(s)), kde *z* je proměnná, do které se uloží přijatá data.

Pro vytvoření objektu sériového portu v Matlabu je potřeba znát název portu, na kterém je připojeno Arduino. To se dá zjistit například v Arduino IDE v menu nástroje, kde se zobrazí číslo portu, například COM7. Poté se použije funkce serialport(), která přijímá jako argument název portu a rychlost přenosu. Například, pokud je Arduino připojeno na port COM7 a chceme komunikovat rychlostí 115200 bitů za sekundu, napíšeme s = serialport('COM7', 115200) (Matlab sériová komunikace, nedatováno; Zaplatílek, 2020).

4 NÁVRH A REALIZACE MODELU

Cílem bylo vytvořit laboratorní soustavu pro výuku automatizace. Pro možnosti praktické ukázky bylo určeno, že by soustava měla umožňovat „manuální“ ovládání akční veličiny, dvoupolohovou regulaci s omezením akční veličiny a PID, respektive PSD regulaci otáček s nastavitelnou periodou vzorkování. To vše ideálně při vykreslování průběhů důležitých veličin v reálném čase. Pro tyto účely byly zvoleny následující komponenty a postupy.



Obr. 4.1 – Blokové schéma modelu

4.1 ARDUINO MEGA2560 REV3

Pro splnění cílů práce byla jako řídicí jednotka zvolena deska Arduino Mega2560 R3. Disponuje velkým počtem vstupů a výstupů, větší pamětí, čímž umožňuje případnou modifikaci například k jiným projektům, a v neposlední řadě byla deska v době tvorby projektu k dispozici.

Mikrokontroler umožňuje hardwarové přerušení, dokáže poskytnout napájení pro ventilátor, má dostatečnou paměť a všechny další nutné vlastnosti potřebné k realizace projektu. Pro fyzickou minimalizaci projektu by nejspíš šlo využít i menší mikrokontroléry, pro zvolený větší ventilátor by to však víceméně postrádalo smysl. Jediný faktor pro zvolení menší desky by mohla být v tomto případě cena.

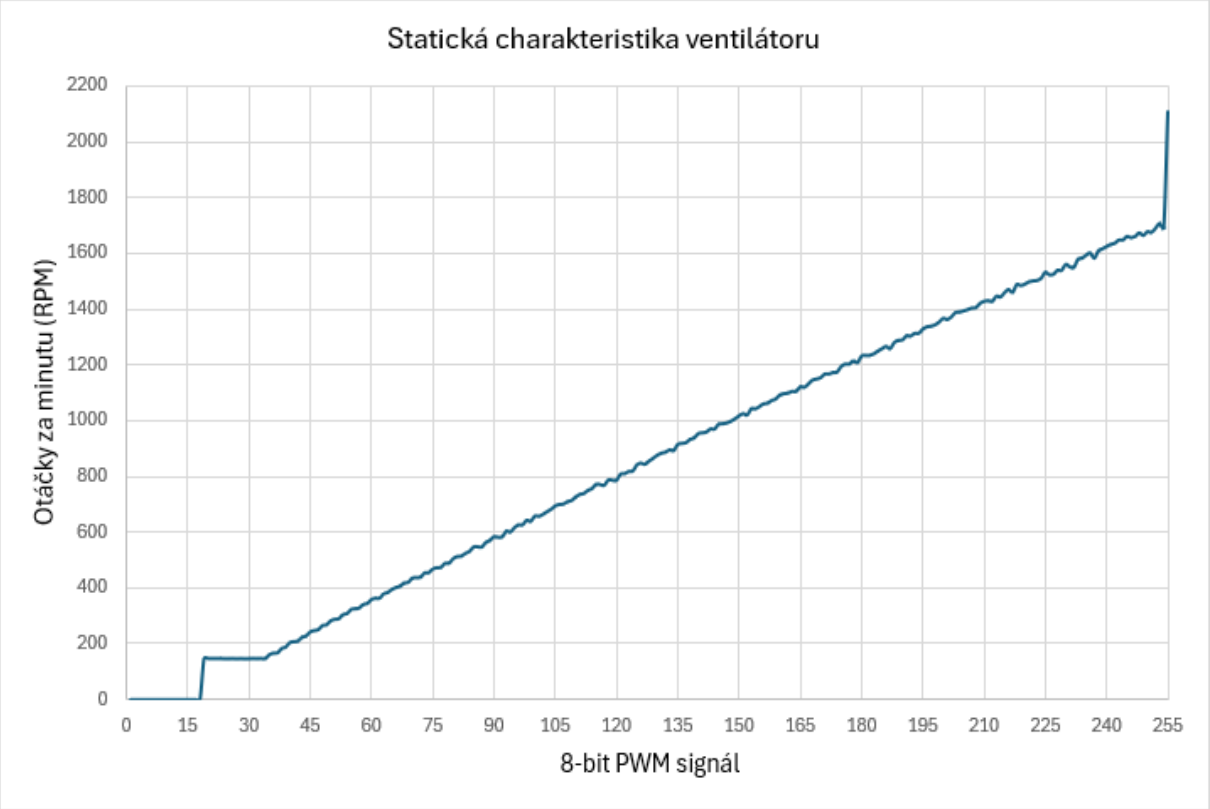
4.2 NOCTUA NF-A8 5V PWM

Jedná se o 80mm ventilátor, který ke svému chodu vyžaduje napětí 5 V, disponuje obvodem pro ovládání otáček pomocí PWM a čidlem, díky kterému lze měřit rychlost otáčení ventilátoru. Konektor ventilátoru je tedy čtyřpinový. Dva piny pro napájení, jeden pro ovládací signál, jeden pro signál měřicí. Napájení ventilátoru může být přivedeno přímo z mikrokontroléru, jelikož stačí 5 V a proud do 150 mA, které mikrokontrolér bez problému dokáže poskytnout. Specifikace uvádí minimální rychlost otáček při cca 20% PWM jako 450 RPM ± 20 %. Maximální uvedená rychlost je 2200 RPM ± 10 %.

Na trhu jsou dostupné i menší varianty ventilátoru s možností vyšších otáček, i větší ventilátory, kde jsou maximální dosažené otáčky nižší. Pro praktickou ukázku byl zvolen střední model, který lze například rukou jednodušeji brzdit a zároveň jsou maximální dosažené otáčky poměrně vysoké (Noctua specifikace, nedatováno).

Obr. 4.2 – Ventilátor Noctua NF-A8 5V PWM
(Noctua, nedatováno)

Z naměřené statické charakteristiky na obr. 4.3 je patrné, že otáčky ventilátoru jsou do nastavení 8-bitového PWM signálu na hodnotu 19 nulové. Následně se otáčky při zvyšování hodnoty až na hodnotu 34 nemění a zůstávají konstantě v blízkosti 148 RPM. Při zvyšování hodnoty PWM signálu RPM stoupají téměř lineárně až k hodnotě 254. Při nastavení hodnoty PWM na 255 poskočí rychlost otáčení o cca 400 RPM na 2106 RPM. Tato nelinearita může být způsobena ovládáním ventilátoru mikrokontrolérem, kterému bylo nutné upravit frekvenci generovaného PWM. Výstup může být zkreslený, jelikož vstup není identický, jako na který je ventilátor stavěn.

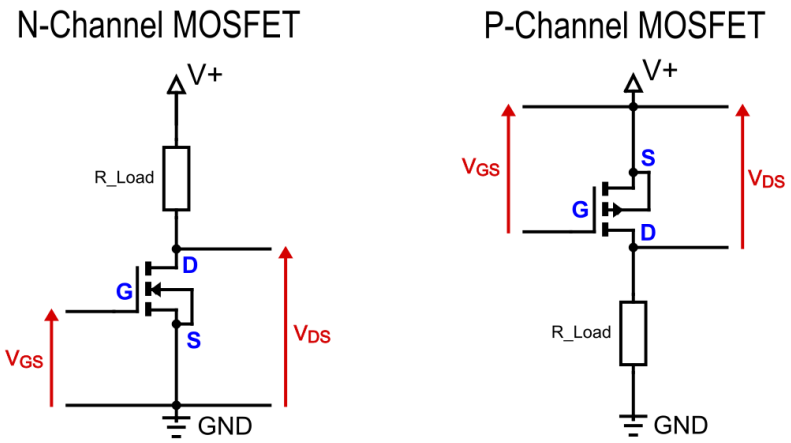


Obr. 4.3 – Naměřená statická charakteristika

4.3 OVLÁDÁNÍ OTÁČEK

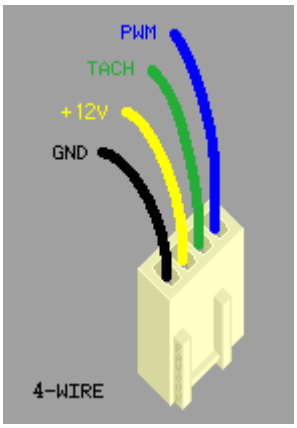
Základní princip obvodu pro ovládání otáček ventilátoru pomocí PWM spočívá ve spínání většího proudu, případně napětí, proudem menším. Piny mikrokontroléru poskytující PWM signál samy dokážou dodat maximálně cca 40 mA, což je pro chod větráku málo, potřebuje 150 mA. Využívá se tedy spínání proudu z 5V napájecího pinu z desky, který dokáže bez problému dodat požadovaný proud větráku. Nejčastěji se pro takové potřeby používají tranzistory, které mají schopnost rychle vodivě uzavřít a otevřít svůj přechod, aby správně

reagovaly na PWM signál. Na bázi takového tranzistoru se přivede signál PWM a na kolektor signál spínaný. Podle typu tranzistoru NPN, či PNP, pak tranzistor pro určité úrovně signálu funguje jako spínač. V praxi se pro tyto potřeby používají například tranzistory MOSFET, které fungují na lehce jiném principu, ale spínání umožňují sejným způsobem, jako je zobrazeno na obr. 4.4.



Obr. 4.4 – Principiální Schéma tranzistoru MOSFET jako vypínače (Ewald, 2022)

Použitý ventilátor disponuje tedy piny PWM, představujícím bázi tranzistoru, 5V, přestavujícím kolektor tranzistoru a GND, sloužící jako zem a emitor tranzistoru, barvy vodičů jsou viditelné na obr. 4.5.



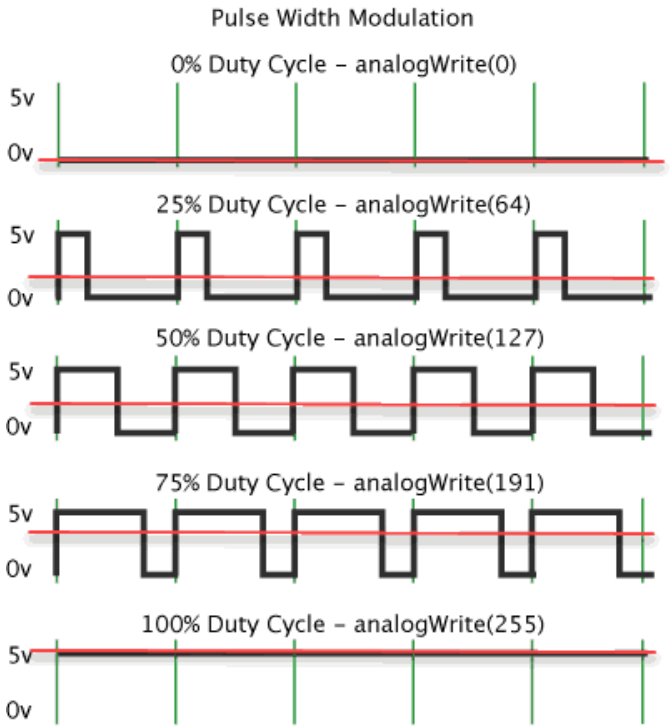
Obr. 4.5 – Konektor 4-pinového 12V ventilátoru (4-pinový konektor ventilátoru, nedatováno)

V jednoduchosti PWM signál spínáním tranzistoru určuje ventilátoru procento času, kdy je do něj v dané periodě dodáván proud. Čím větší je procento, tím vyšší budou otáčky ventilátoru. To umožňuje digitálním signálem o dvou hodnotách, vypnuto či zapnuto, analogově řídit otáčky.

Ventilátory jsou stavěné na jinou frekvenci PWM, než dokáže poskytnout použitý mikropočítač. Arduino v základu na svých pinech k tomu určených využívá PWM o frekvenci 1 KHz. PC ventilátory jsou však vyráběny zpravidla pro využití v případech, kde je frekvence PWM signálu 25 KHz, proto bylo nutné tuto frekvenci programově upravit (Dossena, nedatováno).

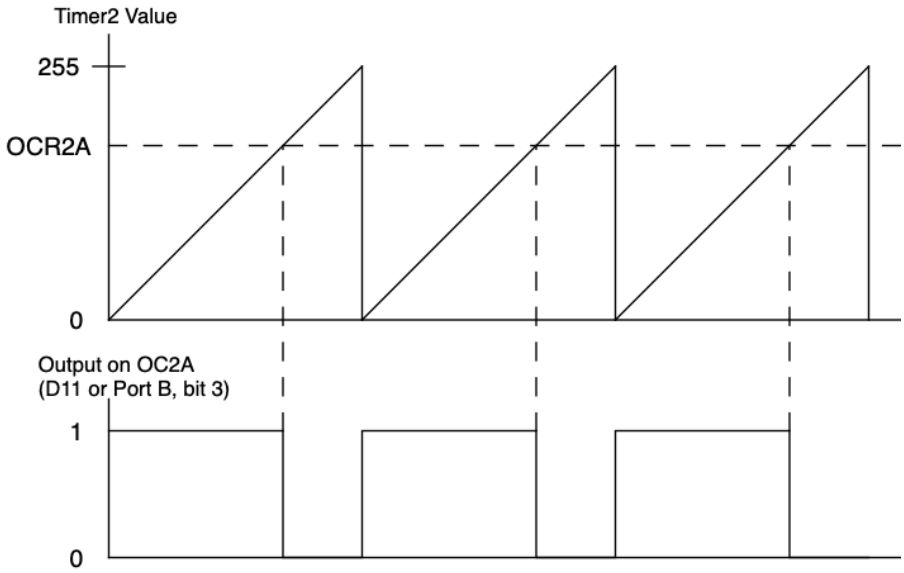
4.3.1 PWM

PWM funguje na principu přerušovaného dodávání energie do zátěže. Neposkytuje napětí a proud konstantě, nýbrž ve formě pulzů. Šířka pulzu udává, kolik energie je do zátěže dodáno. Střída pulzu, neboli duty, je poměr mezi dobou signálu v hodnotě high a celkovou dobou periody, což je znázorněno na obr. 4.6. Čím je větší střída pulzu, tím větší je efektivní napětí dodané zátěži (Christ a Wernli, 2014).



Obr. 4.6 – Znázornění střídy PWM signálu (PWM signál, nedatováno)

K vytvoření takového signálu využívají mikrokontroléry vnitřní časovače. Pomocí úpravy jejich registrů je možné měnit frekvenci signálu PWM nebo typ PWM signálu. Základem jsou tedy časovače, společně s komparátory. Časovač generuje signál určující frekvenci PWM. Komparátor následně porovnává hodnotu z časovače s nastavenou hodnotou. Na základě výsledku tohoto porovnání mění stav výstupního PWM pinu, jak je znázorněno na obr. 4.7 (Jiang et al., 2021).



Obr. 4.7 – Funkce 8-bitového Timeru2 při tvorbě signálu „Rychlé PWM“ (8-bitový PWM signál, nedatováno)

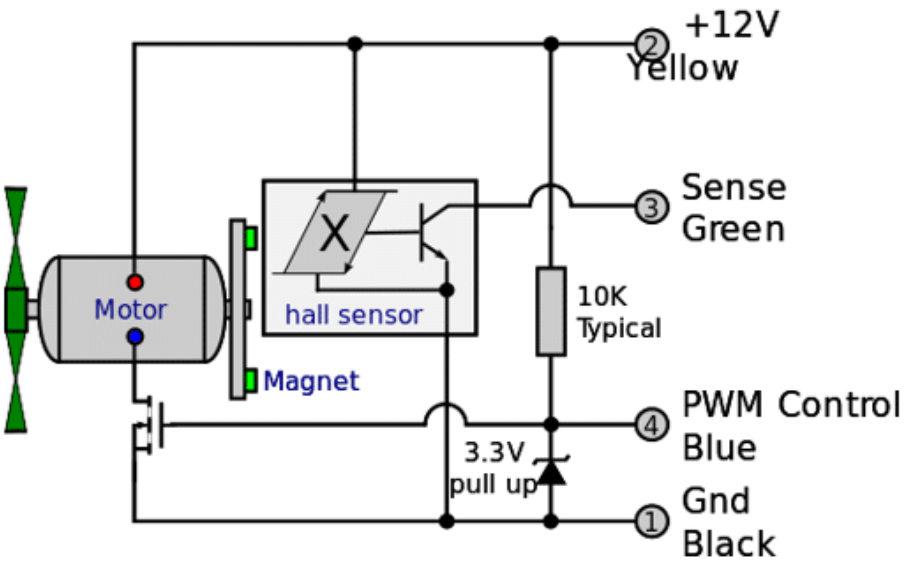
4.4 MĚŘENÍ OTÁČEK

Ventilátor disponuje Hallovým senzorem, díky kterému je následně možné programově měřit rychlost otáčení ventilátoru. Senzor vyšle dva impulzy za otáčku na pin ventilátoru, označeným jako tach. Signál z tohoto pinu je přiveden na pin mikrokontroléru, který umožňuje hardwarové přerušení při nástupné či sestupné hraně signálu. Díky tomu je pak v programu možné měřit periodu mezi pulzy, s jejíž pomocí lze vypočítat aktuální rychlost otáčení ventilátoru.

4.4.1 Hallův senzor

Tento senzor využívá Hallova jevu. Pokud elektrický proud prochází skrze vodič, využívá se polovodičová destička, umístěná v magnetickém poli, elektrony ve vodiči či polovodiči jsou vychýleny magnetickou silou na jednu stranu vodiče. Takový pohyb tvoří elektrické pole a napětí na okraji vodiče či destičky, což známo jako napětí Hallovo (Hallův jev, nedatováno).

V praxi je senzor z proužku polovodiče s elektrodami na obou koncích. Magnet je upevněn na rotující části ventilátoru a při každém jeho pohybu u polovodičového proužku vzniká na elektrodách signál. Tyto magnety jsou na rotující části ventilátoru dva, tudíž na příslušném pinu vznikají dva impulsy za otáčku, principiální schéma obdobného 12V ventilátoru je zobrazeno na obr 4.8.



Obr. 4.8 – Principiální schéma vnitřního zapojení 12V PWM ventilátoru (Maetschke, 2024)

4.4.2 Hardwarové přerušení

Hardwarové přerušení na mikrokontroléru Arduino umožňuje reagovat na určité externí události okamžitě, namísto neustálé kontroly jejich stavu v hlavní smyčce programu. Když dojde k události, která vyvolá přerušení jako je například změna napětí na pinu, mikrokontrolér

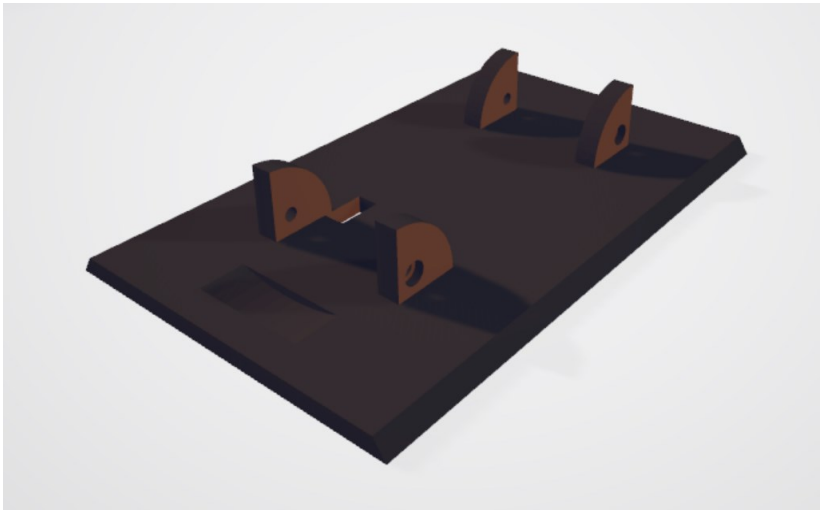
pozastaví běžící kód, vykoná funkci definovanou pro obsluhu přerušení a vrátí se k původnímu kódu.

Pro měření rychlosti otáček ventilátoru bylo využito externí přerušení, které je spouštěno změnou úrovně nebo hrany signálu na pinu 2. Při každé půlotáčce se tedy vykoná funkce, při které se uloží čas při posledním pulzu a načte aktuální čas běhu programu. Pomocí těchto dvou hodnot je dále v programu možné vypočítat aktuální rychlost otáčení regulátoru. Důležité je, aby funkce pro obsluhu přerušení byla co nejjednodušší a nezdržovala tak zbytečně chod zbytku programu, proto se aktuální rychlost nepočítá přímo v obsluze přerušení (Slinták, 2011).

4.5 KONSTRUKCE

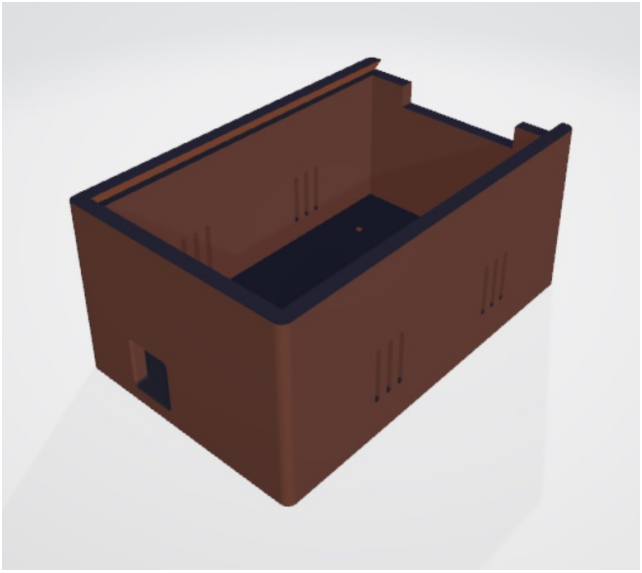
Návrh konstrukce splňuje základní požadavky, kterými jsou upevnění komponentů, jejich propojení a vývod kabelu z desky pro jednoduché připojení a komunikaci s PC.

Jedná se o krabičku navrhnutou v modelovacím 3D CAD programu Fusion 360, následně vytisknutou pomocí 3D tiskárny. Krabička má odnímatelné víko, na které lze upevnit ventilátor vertikálně, aby neměl problém s tokem vzduchu a zároveň bylo možno působit vnější chybou na jeho rotační část, což může být také součástí praktické ukázky výuky. U jednoho z upevňovacích sloupků se nachází otvor pro protažení kabelu s vodiči z ventilátoru.



Obr. 4.9 – Víko konstrukce

Spodní část disponuje průduchy a otvory pro upevnění desky Arduino a otvorem pro USB kabel, aby bylo možné propojení s PC a byly vidět indikační diody ukazující, zda je mikrokontrolér v provozu.

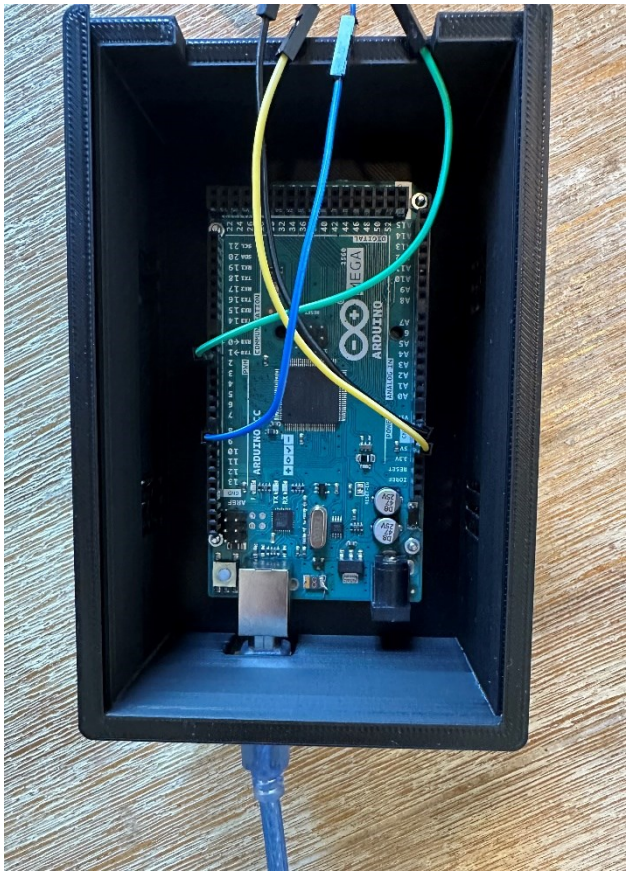


Obr. 4.10 – Spodní část konstrukce

Víko do spodní části krabičky pasuje díky drážkám s offsetem o velikosti 0,25 mm mezi nimi. Díky tomu víko po zasunutí do krabičky ani při naklopení samovolně nevypadává a drží na místě.

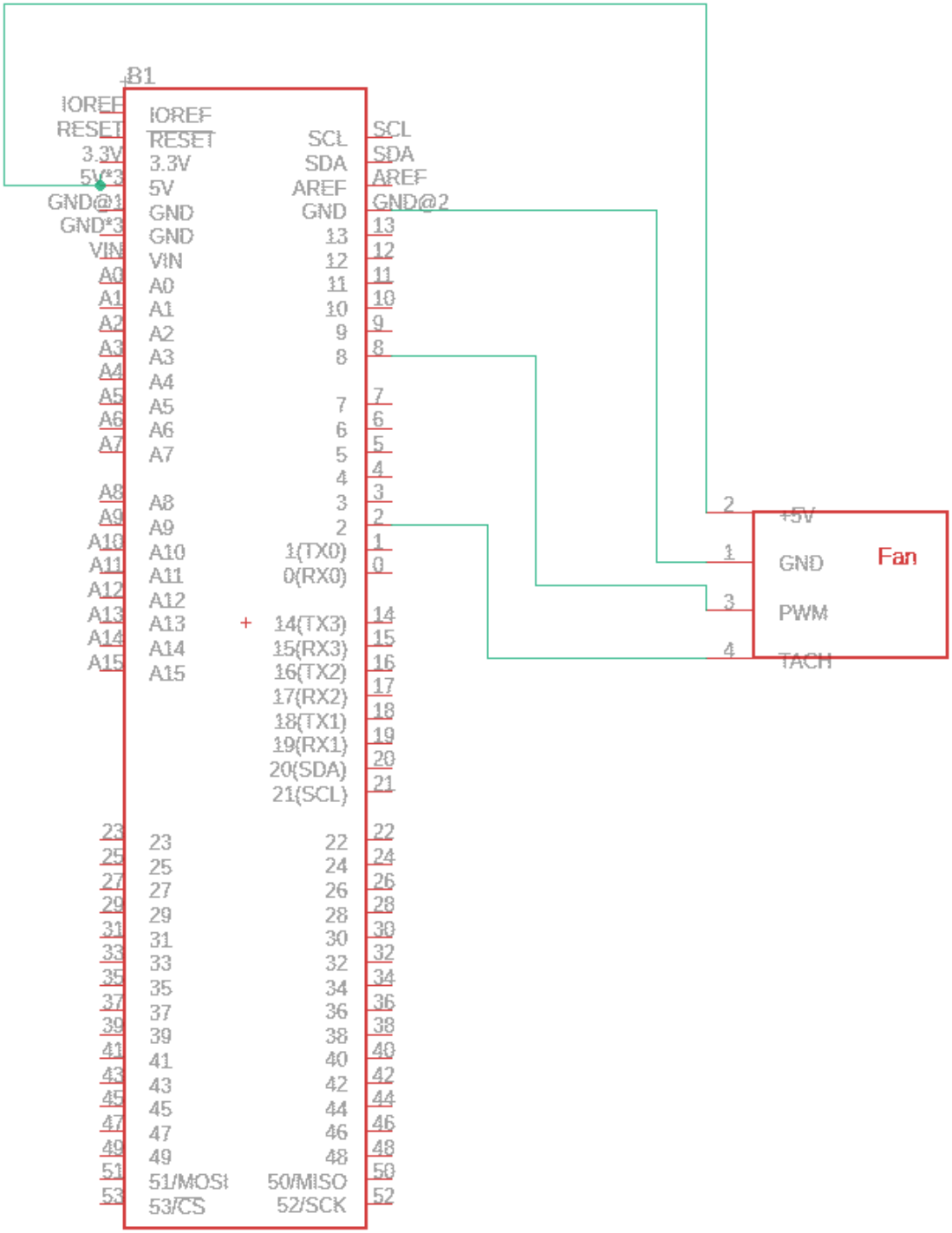
Obr. 4.11 – Osazená konstrukce

Deska je uvnitř krabičky upevněna čtyřmi šrouby, jejichž hlavy jsou zapuštěny ze spodní strany do konstrukce, aby nevyčnívaly.



Obr. 4.12 – Upevnění desky

Kompletní propojení pinů Arduina a ventilátoru je znázorněno na obr. 4.13.



Obr. 4.13 – Zapojení pinů ventilátoru do Arduina

5 APLIKACE A FUNKCE

V prostředí Matlab byly vytvořeny funkce pro „manuální“ ovládání akční veličiny, PID, respektive PSD, regulaci otáček s nastavitelnými parametry K_p, T_i a T_d a dvoupolohovou regulaci otáček s nastavitelným omezením minimální a maximální hodnoty akční veličiny. U všech těchto funkcí je vstupním parametrem i vzorkovací perioda v sekundách a název portu, ke kterému je model připojen. Žádaná hodnota, respektive průběh akční veličiny u „manuálního“ ovládání akční veličiny je také vstupním parametrem, a to ve tvaru vektoru. První hodnota tohoto vektoru představuje první žádanou hodnotu nebo akční veličinu, v závislosti na funkci. Každá další hodnota pak představuje žádanou hodnotu, případně akční veličinu opět do další periody vzorkování. Vzor takového vektoru je na obr. 5.1.

```
W=[904*ones(50,1); 534*ones(70,1); 1531*ones(60,1); 709*ones(70,1); 223*ones(70,1)];
```

Obr. 5.1 – Možná forma vstupního vektoru žádané veličiny

Pro mikrokontrolér Arduino byla vytvořena jedna aplikace, která spustí proces regulace nebo ovládání podle toho, která funkce byla v prostředí Matlab spuštěna, tuto informaci přijme stejně jako další po sériové lince. Vývojový diagram aplikace je zobrazen na obr. 5.5.

Pokud jsou zadány všechny koeficienty PID regulátoru tak, aby byly aktivní všechny tři složky regulace, je použit vzorec přírůstkového tvaru PSD regulátoru. Při jeho tvorbě se vychází ze vztahu pro diskrétní PID regulátor, který je zobrazen na obr. 5.2.

$$u(kT) = k_p\left[e(kT) + \frac{T}{T_I}\sum_{i=0}^{k} e(iT) + \frac{T_D}{T}\nabla e(kT)\right]$$

Obr. 5.2 – Vzorec diskrétního PID regulátoru (Ščevík, nedatováno)

Úpravou je získán vzorec z obr 5.3, ze kterého lze již jednoduše vyjádřit přírůstkový tvar regulátoru odečtením předešlého akčního zásahu od obou stran rovnice. Přírůstkový tvar pak nepočítá akční zásah pro každý krok, ale pouze jeho změnu. Přírůstkový tvar diskrétního regulátoru je však možné využít pouze v případech, kdy regulátor zahrnuje sumační činnost.

$$u(kT) = u[(k-1)T] + q_0 e(kT) + q_1 e[(k-1)T] + q_2 e[(k-2)T]$$

Obr. 5.3 – Vzorec pro výpočet akční veličiny (Ščevík, nedatováno)

Koeficienty *q0*, *q1* a *q2* jsou získány přepočtem dle vzorců z obr 5.4.

$$q_0 = k_P\left(1+\frac{T}{T_I}+\frac{T_D}{T}\right)$$
$$q_1 = -k_P\left(1+2\frac{T_D}{T}\right)$$
$$q_2 = k_P\frac{T_D}{T}$$

Obr. 5.4 – Vzorce pro výpočet koeficientů q0, q1 a q2
(Ščevík, nedatováno)

Pro ostatní kombinace regulátorů byl zvolen následující postup. Koeficienty PID regulátoru K_p, T_i a T_d jsou přepočteny jinými vzorci na konstanty *q0*, *q1* a *q2* se kterými je následně prováděn výpočet akční veličiny. Pokud například uživatel zadá nulovou derivační časovou konstantu, změna akční veličiny se vypočte vzorcem přírůstkového tvaru PS regulátoru. Pokud uživatel zadá integrační konstantu T_i větší než 999, akční veličina se vypočte vzorcem z obr. 5.2, jelikož program nahradí podíl *T*/*Ti* nulou, nebude zahrnuta sumační činnost a nelze tedy využít přírůstkový tvar. Konstanta *q1* tedy v tomto případě odpovídá zlomku T/T_i, konstanta *q0* odpovídá zlomku T_d/T a *q2* odpovídá K_p. Pro tyto případy jsou v Arduinu uloženy konstanty pod dvěma názvy, aby byla udržena přehlednost kódu a nedošlo k záměně konstant, jelikož dojde ke změně jejich významu.

5.1 APLIKACE V PROSTŘEDÍ ARDUINO IDE

Na začátku celého programu v prostředí Arduino je inicializována knihovna, pomocí které je jednoduchým příkazem změněna frekvence PWM signálu. Následuje inicializace veškerých proměnných, využitých při výpočtech a ukládání hodnot. Následuje definování funkce, která se vykoná při hardwarovém přerušení, které je způsobeno sestupnou hranou tach. signálu z ventilátoru. Při spouštění funkce nástupnou hranou nebyly výsledky měřených RPM tak kvalitní. Signál je přiváděn na pin 2, který hardwarové přerušení umožňuje. Pin, který generuje akční veličinu v podobě PWM signálu je pin 8.

Obr. 5.5 – Vývojový diagram aplikace v Arduinu

Ve funkci setup() je nastaven baudrate na 115200, stejná hodnota musí být nastavená pro správnou komunikaci i v prostředí Matlab. Akční veličina je generována na pinu 8, což je také definováno v setup funkci. Dále je zde nastavena správná frekvence PWM, aktivován

interní pullup rezistor pro pin, na který je přiveden tach. signál, bez něhož by mohl být signál nestabilní, a nebylo by možné správně měřit otáčky. V této sekci programu je také definováno, v jakém případě se má hardwarové přerušení spustit a jaká funkce se má provést. Následuje už pouze uložení aktuálního času do proměnných a nastavení akční veličiny na nízkou hodnotu, což je především úsporné opatření a zároveň je ventilátor po spuštění tichý.

Následuje sekce loop(). Do přijetí inicializačních dat z prostředí Matlab je program zacyklený a čeká právě pouze na inicializaci uživatelem. Ve chvíli, kdy data přijdou, je postupně načítá po řádku ve formě proměnných typu char, které zapisuje do stringu. Po každém přijatém řádku je daný řetězec vyhodnocen a jeho hodnoty jsou převedeny na číselnou hodnotu. Každá funkce přijaté proměnné využívá k různým účelům. Předposlední z nich je vzorkovací perioda, kterou používají všechny funkce. Poslední proměnná slouží pro výběr operace, která nastane. Podle přijaté hodnoty se nastaví příznaková proměnná spouštějící jednu z funkcí regulace či ovládání, provede se další nastavení proměnných a vynuluje se příznak pro inicializační funkci příjmu hodnot z prostředí Matlab.

Pokud uživatel zadá v prostředí Matlab takové konstanty regulátoru, aby byly aktivní všechny tři složky regulátoru, ve vzorci s přírůstkovým tvarem regulátoru se využívají konstanty *q0*, *q1* a *q2*. Pokud uživatel nadefinuje konstanty tak, že je jedna nebo více jeho složek neaktivních, změní se výpočet, a tedy i význam těchto přijatých proměnných. Pro přehlednost je tedy každá přijatá konstanta uložena do dvou proměnných, aby byly ve vzorci s výpočtem akční veličiny či jejího přírůstku konstanty pojmenované logicky.

```
  //funkce pro příjem a nastavení inicializačních informací a hodnot----------------------------------------
  if (Serial.available() && inicializace==true) {
    char inChar = (char)Serial.read();
    // Přidání přijatého znaku do řetězce, pokud není nový řádek
    if (inChar != '\n') {
      dataString += inChar;
    } else {
      // Pokud je nový řádek, zpracujeme řetězec
      switch (dataIndex) {
        case 0: Kp=q2 = dataString.toFloat(); break;
        case 1: Ks=q1 = dataString.toFloat();  break;
        case 2: Kd=q0 = dataString.toFloat();  break;
        case 3: setRPM = dataString.toFloat(); break;
        case 4: vzorkovaciPerioda = (dataString.toFloat()*1000); break;
        case 5: operace = dataString.toFloat(); inicializace=false;
        if (operace==1) {bezPSD=true;}
        else if (operace==2){bezONOFF=true;}
        else if (operace==3){PWM=setRPM; bezmanual=true;analogWrite(fanPin, PWM);  }
        break;
      }
      dataIndex = (dataIndex + 1) % 6; // Přechod na další index
      dataString = ""; // Reset řetězce pro další číslo
}
}
```

Obr. 5.6 – Kód obstarávající příjem informací a spouštění funkcí

Ve zbytku nekonečné smyčky se již pouze neustále aktualizuje čas běhu programu a kontroluje se, zda neuplynula vzorkovací perioda pro spuštěnou funkci. Pokud nastane taková situace, zavolá se příslušná funkce regulace, respektive ovládání, definovaná za koncem smyčky loop().

```
//načtení aktuálního času
unsigned long currentTime = millis();

  // Kontrola zda se má provést PSD řízení řízení-------------------------------------------------------------------------
  if (currentTime - lastReportTime>= vzorkovaciPerioda && bezPSD==true) { // Kontrola, zda uplynula vzorkovací perioda
    regulatePSD(PulseTime, currentTime);
  }

  // Kontrola zda se má provést dvoupolohové řízení řízení----------------------------------------------------------------
if (currentTime - lastReportTime>= vzorkovaciPerioda && bezONOFF==true) {
    regulateOnOff( PulseTime, currentTime);
  }

  // Kontrola zda se má provést manuální řízení---------------------------------------------------------------------------
if (currentTime - lastReportTime>= vzorkovaciPerioda && bezmanual==true) {
    driveFan( PulseTime, currentTime);
}
```

Obr. 5.7 – Kód podmínek pro spouštění funkcí

5.1.1 Funkce pro PSD regulaci

Po spuštění funkce se z periody mezi půlotáčkami ventilátoru vypočtou otáčky za minutu.

Obr. 5.8 – Vývojový diagram funkce regulatePSD

Následuje část kódu, eliminující chybu měření. Zvláště při nižších otáčkách RPM náhodně zvyšovaly svoji hodnotu až dvojnásobně. Tuto chybu bylo nutné eliminovat, nebo alespoň minimalizovat. Ventilátor ani při velkému skoku akční veličiny nedokáže při nižší vzorkovací periodě zvýšit otáčky o více než cca 10 %, pokud tedy nastane skok měřených otáček větší než 25 %, použije se minulá hodnota otáček. Pokud je hodnota měřených otáček vyšší i při dalším měření, program ji již bere jako hodnotu správnou.

```
if (lastRPM > 0 && newRPM > 1.25 * lastRPM) {
  debounceCounter++;
  if (debounceCounter < 2) {
    rpm = lastRPM; // Použij poslední hodnotu RPM
  } else {
    rpm = newRPM; // Aktualizuj RPM po dvou zákmitových hodnotách
    lastRPM = newRPM; // Ulož nové RPM pro další cyklus
    debounceCounter = 0; // Resetuj počítadlo zákmitů
  }
} else {
  rpm = newRPM; // Aktualizuj RPM
  lastRPM = newRPM; // Ulož nové RPM pro další cyklus
  debounceCounter = 0; // Resetuj počítadlo zákmitů
}
```

Obr. 5.9 – Kód ošetřující chybu měření otáček

I po tomto ošetření se nepodařilo veškeré chyby měření eliminovat, byla tedy aplikována funkce pro filtraci měřených otáček diferenční rovnicí prvního řádu zobrazenou na obr. 5.10, s koeficienty $a = 0{,}3$ a $b = 0{,}7$.

```
filtRPM = ((a*rpm) + (b*filtRPM));
```

Obr. 5.10 – Filtrace měřené veličiny diferenční rovnicí prvního řádu

Následuje kontrola, zda se jedná o první průchod funkcí. V případě že ano, uloží se aktuální čas. Ten se využívá k výpočtu času běhu funkce, který se odesílá do Matlabu. Poté se vypočítá regulační odchylka, její suma a zpětná diference. Následuje výběr typu regulátoru, výpočet akčního zásahu a uložení dvou historických regulačních odchylek

```
//Volba regulátoru
if (q2!=0&&q1!=0&&q0!=0){
// Výpočet výstupu z regulátoru v přírůstkovém tvaru
output += (q0*error)+ (q1*lastError)+(q2*lastError2);}// PSD

else if (q1==0){output = (Kp *error) + (Ks* Kp * sumErr) + (Kd*Kp * difference);} //regulace bez sumace P/PD
else if (q1!=0&&q2==0){output += (Ks* error)+(Kd*(error - (2*lastError)+lastError2));} // S/SD přírůstkový
else { output += (Kp * difference) + (Ks *Kp *error) );} ///PS přírůstkový
```

Obr. 5.11 – Kód pro výběr regulátoru

Výstup je následně omezen pouze do lineární části statické charakteristiky ventilátoru, tedy minimální hodnota je 34, maximální 254. Výsledná hodnota PWM je zapsána na ovládací pin.

Měřené a vypočtené hodnoty rozdělené čárkou se odešlou přes sériovou linku a načte se příští žádaná hodnota. Pokud místo žádané hodnoty přijde informace o ukončení měření, nastaví se akční veličina regulátoru na minimum a příznak pro běh funkce se nastaví na nulovou hodnotu. Pokud nešlo o poslední hodnotu, nastane návrat do smyčky loop() a čekání na další vzorkovací periodu.

5.1.2 Funkce pro dvoupolohovou regulaci

Průběh funkce je velmi podobný funkci pro regulaci PSD regulátorem.

Obr. 5.12 – Vývojový diagram funkce regulateOnOff

Rozdílem je výpočet akční veličiny, která může nabývat pouze dvou hodnot. Pokud je regulační odchylka větší než 0, tedy regulovaná veličina větší než žádaná hodnota, nastaví se akční zásah na nižší hodnotu. Pokud je regulovaná veličina 0 nebo menší, tudíž je regulovaná veličina menší než žádaná hodnota, akční veličina je nastavena na vyšší hodnotu. Z této části kódu je zřejmé využití proměnných K_d a K_s, které byly v PSD regulaci využity jako váhové konstanty pro sumační a diferenční část výpočtu.

```
// Výpočet chyby
error = setRPM - rpm;
if (error > 0) {
  PWM = Kd; // Nastavení PWM signálu na hodnotu Kd
} else {
  PWM = Ks; // Nastavení PWM signálu na hodnotu Ks
}
analogWrite(fanPin, PWM); // Nastavení PWM signálu
```

Obr. 5.13 – Kód dvoupolohové regulace

Následuje obdobný kód jako u funkce regulace PSD, tedy vzájemná komunikace a kontrola, zda má být regulace ukončena.

5.1.3 Funkce pro manuální ovládání akční veličiny

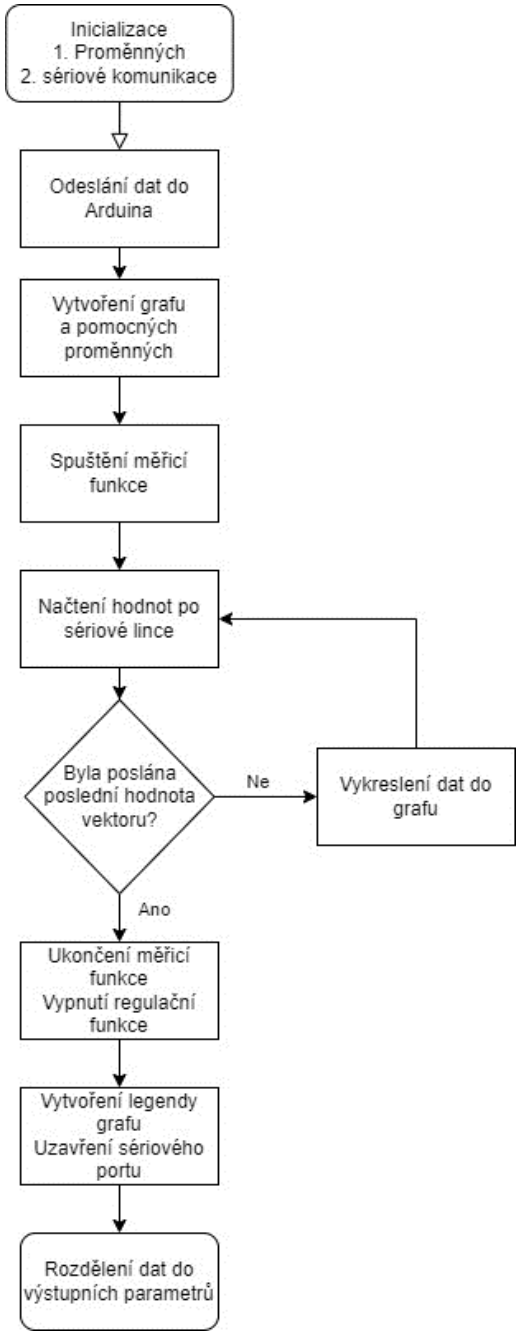
Kód funkce je i pro „manuální“ ovládání akční veličiny obdobný, jako u funkcí regulace.

Obr. 5.14 – Vývojový diagram funkce driveFan

Zásadním rozdílem je, že se v této funkci neprovádí žádný výpočet výstupu, jelikož je jeho hodnota dána přijatou hodnotou z vektoru akčních veličin z prostředí Matlab. Tudíž se po výpočtu RPM a ošetření chyby měření akční veličina nastaví na poslední přijatou hodnotu, provede se komunikace a kontrola, zda má být ukončeno ovládání.

5.2 APLIKACE V PROSTŘEDÍ MATLAB

Všechny tři vytvořené funkce odesílané z prostředí Matlab jsou velmi podobné. Každá z nich má jiné vstupní parametry, podle potřeby jejich nastavování pro regulaci či ovládání. Po spuštění každé z funkcí nastane inicializace sériové linky, odeslání šesti proměnných v podobě stringů, které obsahují pro každou z funkcí specifické informace. Následuje vytvoření proměnných dále využívaných v programu, vytvoření grafu, do kterého se budou ukládat přijaté hodnoty a nastavení příznaku pro spuštění „měřicí funkce“.



Obr. 5.15 – Vývojový diagram aplikace v Matlabu

5.2.1 Funkce pro PID regulaci

Funkce pro regulaci pomocí PID regulátoru navrací hodnoty regulované veličiny, akční veličiny, času, regulační odchylky žádané hodnoty. Vstupními parametry jsou název portu, ke kterému je mikrokontrolér připojen, vzorkovací perioda ve vteřinách, vektor žádané hodnoty a koeficienty K_p, T_i a T_d.

```
function [dataRPM, dataPWM, dataTime, dataError, dataSetpoint] = automat_PID(comPort,T, W, Kp, Ti, Td)
```

Obr. 5.16 – Definice funkce automat_PID

Tyto konstanty jsou vyhodnoceny a do desky jsou odeslány v upraveném tvaru podle toho, jaké konstanty byly zadány uživatelem jako vstupní.

```
if Kp~=0 && Ti<=999 && Td~=0 %PSD
   q0=Kp*(1+(T/Ti)+(Td/T));
   q1=-Kp*(1+2*(Td/T));
   q2=Kp*(Td/T);

elseif Ti>=1000 %P/PD regulátor
        q2=Kp; %Kp
        q1=0;
        q0= Td/T; %Kd

else %S/PS/SD regulace
q2=Kp;
q1=T/Ti;
q0=Td/T;
end

    % Odeslání PID koeficientů a první žádané hodnoty
    dataSend = [q2, q1, q0, W(1),T, 1];
    for i = 1:length(dataSend)
        writeline(s, num2str(dataSend(i)));
        pause(0.005);
    end
```

Obr. 5.17 – Vyhodnocení konstant a odeslání do Arduina

Měřicí funkce přijímá informace zaslané mikrokontrolérem, dokud jsou dostupné na sériové lince a zapisuje je do jednotlivých sloupců matice data. Po přijetí a uložení dat odešle další hodnotu žádané veličiny z vektoru *W* a vykreslí přijaté hodnoty do grafu. Tento proces se opakuje, dokud nejsou zaslány všechny hodnoty z vektoru žádaných hodnot, poté se odešle po sériové lince příkaz pro ukončení regulace. Příznak pro spuštění měřicí funkce je nastaven na nulu, do grafu se vykreslí legenda průběhů, uzavře se sériová linka a matice data se rozdělí na jednotlivé sloupce, které jsou výstupními parametry funkce.

```
while (mer)
    if s.NumBytesAvailable > 0
        % Čtení řádku z sériového portu
        line = readline(s);
        % Rozdělení přijatých informací do částí vektoru
        values = str2double(split(line, ','));
        % Uložení dat
        data(end+1, :) = values;
        % Odeslání následující žádané veličiny
        if index <= length(W)
            writeline(s, num2str(W(index)));
            index = index + 1;
             % Vykreslení dat
         if size(data,1) > 1
         h1=  plot(((data(:,3)/1000)), data(:,1), 'r-', LineWidth=1.5); % Regulovaná veličina
         h2=  plot(((data(:,3)/1000)), data(:,2), 'b-'); % Akční veličina

         h3= plot(((data(:,3)/1000)), abs(data(:,4)), 'k-'); % regulační odchylka
         h4= plot(((data(:,3)/1000)), data(:,5), 'g-.', LineWidth=2); % žádaná veličina

         end

        drawnow;
        else
            writeline(s, num2str(6666)); % Odeslání informace, která ukončí regulaci
            mer=0; % Vypnutí měřicí funkce
            clear s;
            break;
        end
```

Obr. 5.18 – Kód pro komunikaci s Arduinem a vykreslování přijatých dat

5.2.2 Funkce pro dvoupolohovou regulaci

Funkce pro regulaci dvoupolohovým regulátorem navrací hodnoty regulované veličiny, akční veličiny, času, regulační odchylky žádané hodnoty. Vstupními parametry jsou název

portu, ke kterému je mikrokontrolér připojen, vzorkovací perioda ve vteřinách, vektor žádané hodnoty a hodnoty akční veličiny pro polohy regulátoru off a on.

```
function [dataRPM, dataPWM, dataTime, dataError, dataSetpoint] = automat_ON_OFF(comPort,t, W, Umin, Umax)
```

Obr. 5.19 – Definice funkce automat_ON_OFF

Vstupními parametry je pět hodnot. Odesíláno a přijímáno deskou je vždy hodnot šest. Tudíž první zaslaná hodnota je nahrazena nulou a v regulační funkci v mikrokontroléru zůstává nevyužita. Zbytek měřicí funkce obsahuje totožný kód, jako měřicí funkce pro regulaci PID regulátorem.

5.2.3 Funkce pro manuální ovládání akční veličiny

Funkce pro „manuální“ ovládání navrací hodnoty RPM, akční veličiny a času. Vstupními parametry jsou název portu, ke kterému je mikrokontrolér připojen, vzorkovací perioda ve vteřinách a vektor akční veličiny.

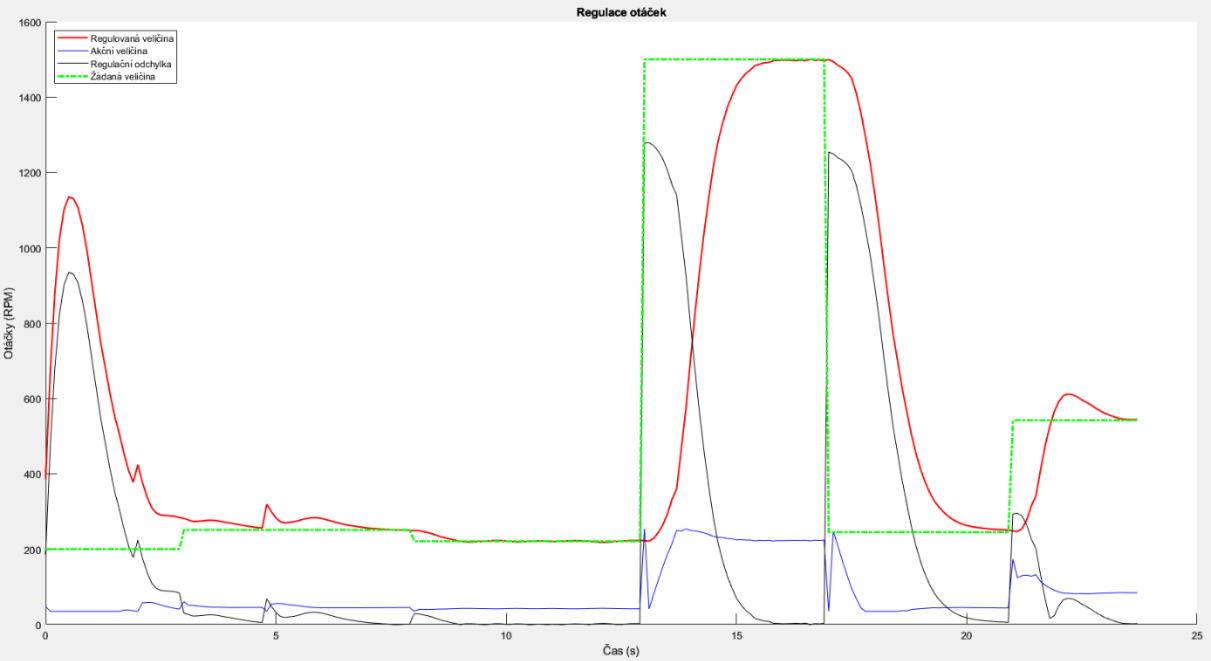
```
function [dataRPM, dataPWM, dataTime] = manual(comPort,T, U)
```

Obr. 5.20 – Definice funkce manual

Při odesílání proměnných do mikrokontroléru jsou nevyužité proměnné opět nahrazeny nulou. Měřicí funkce je opět velmi podobná. Po příjmu veličin z Arduina se z vektoru U pošle následující hodnota akční veličiny, vykreslí přijatá data a proces se opakuje, dokud nejsou zaslány všechny hodnoty z vektoru akčních veličin. Poté se odešle po sériové lince příkaz pro ukončení regulace, příznak pro spuštění měřicí funkce je nastaven na nulu, do grafu se vykreslí legenda průběhů, uzavře se sériová linka a matice data se rozdělí do sloupcových vektorů, které jsou výstupem funkce.

6 EXPERIMENTY

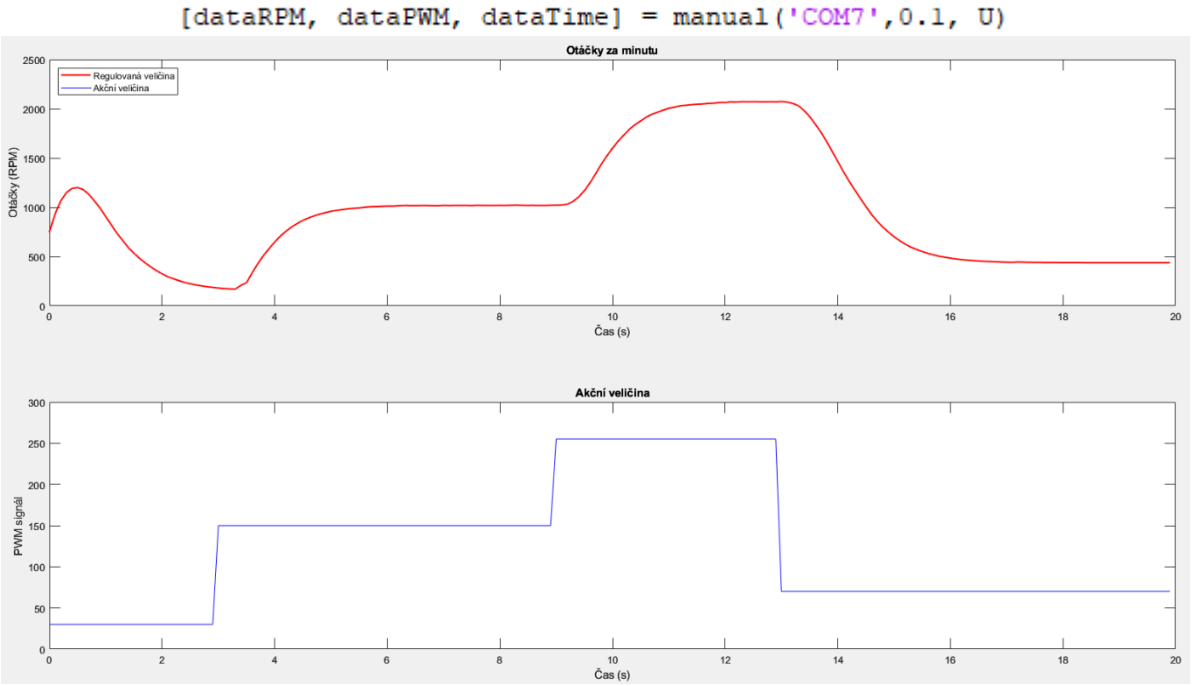
I přes programové ošetření zákmitů se v průběhu regulace může stát, že zašuměný signál z měření regulované veličiny poskytne chybnou hodnotu. Tento problém bylo možné minimalizovat softwarovou filtrací diferenční rovnicí prvního řádu. Jev se při měření vyskytuje velmi zřídka a zároveň nabízí možnost pozorování reakce regulátoru na náhodnou chybu. Chybu je možné vidět před pátou vteřinou regulace na obr. 6.1.



Obr. 6.1 – Průběh s chybou měření

6.1 MANUÁLNÍ OVLÁDÁNÍ AKČNÍ VELIČINY

Pomocí „manuálního“ ovládání akční veličiny byla naměřena statická charakteristika ventilátoru. Dá se využít také pro naměření impulzní či přechodové charakteristiky. Reakce na „manuální“ skokové změny akční veličiny je na obr. 6.2.

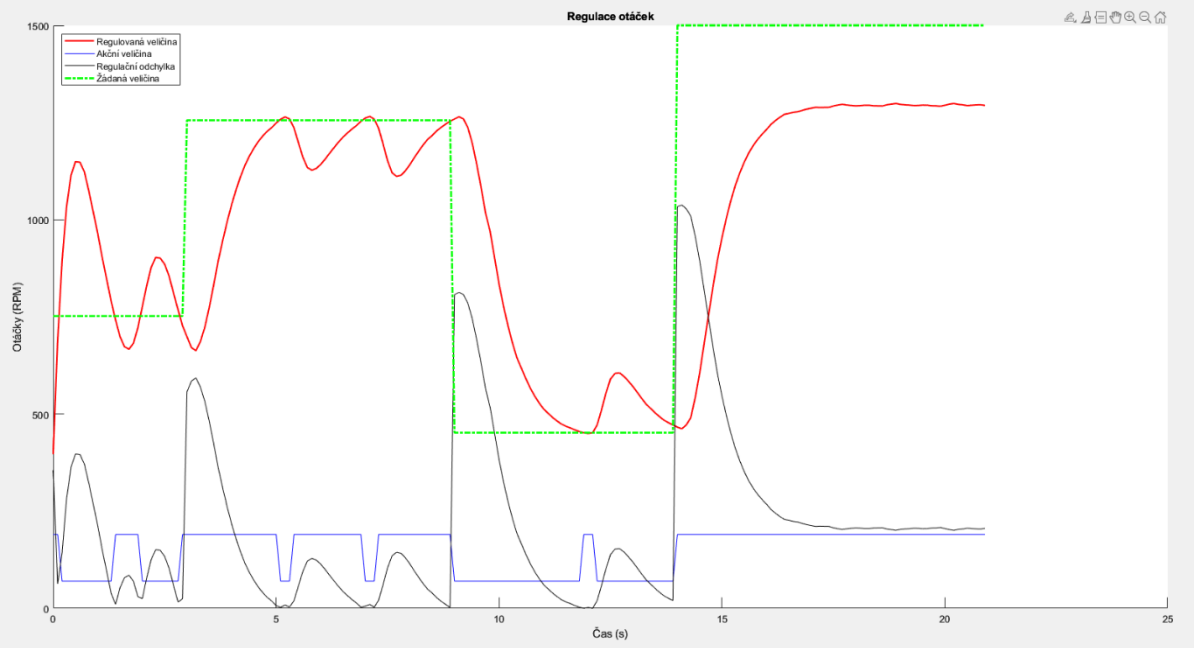


Obr. 6.2 – Průběh „manuálního“ ovládání akční veličiny

6.2 DVOUPOLOHOVÁ REGULACE

U dvoupolohové regulace byla nastavena maximální hodnota akční veličiny na 190 a minimální na 70, kvůli čemuž nebyl regulátor schopen dosáhnout nejvyšší žádané hodnoty. Kvalita regulace byla neuspokojivá i v místech, kde dvoupolohový regulátor zvládal plnit svoji funkci, pro takové případy je vhodné zvolit jiný způsob regulace.

```
[dataRPM, dataPWM, dataTime, dataError, dataSetpoint] = automat_ON_OFF('COM7',0.1, W, 70, 190)
```

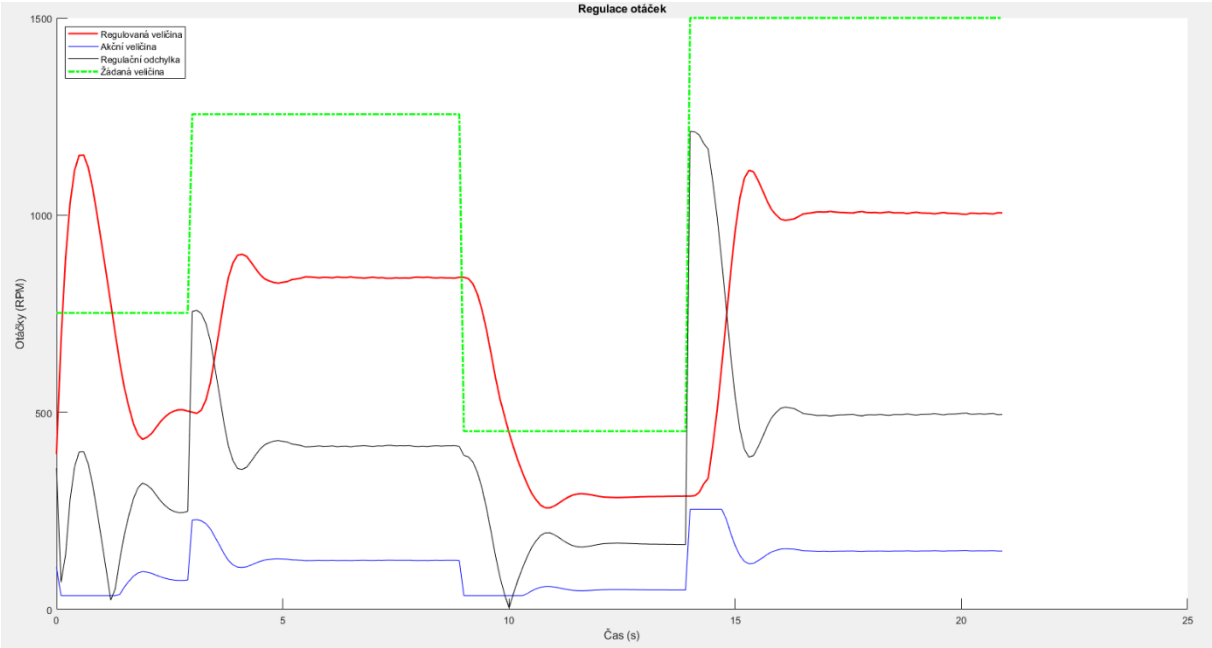


Obr. 6.3 – Průběh dvoupolohové regulace

6.3 REGULACE P REGULÁTOREM

Při regulaci P regulátorem se vyskytovala trvalá regulační odchylka, která je patrná z obr. 6.4. V prvních cca 2 vteřinách regulace je zjevný přechod na vypočítanou hodnotu akční veličiny podle vzorce, z pevně nastavené hodnoty a skoku otáček po restartu mikrokontroléru, který nastane vždy po inicializaci sériové komunikace.

```
[dataRPM, dataPWM, dataTime, dataError, dataSetpoint] = automat_PID('COM7',0.1, W, 0.2, 9999, 0)
```

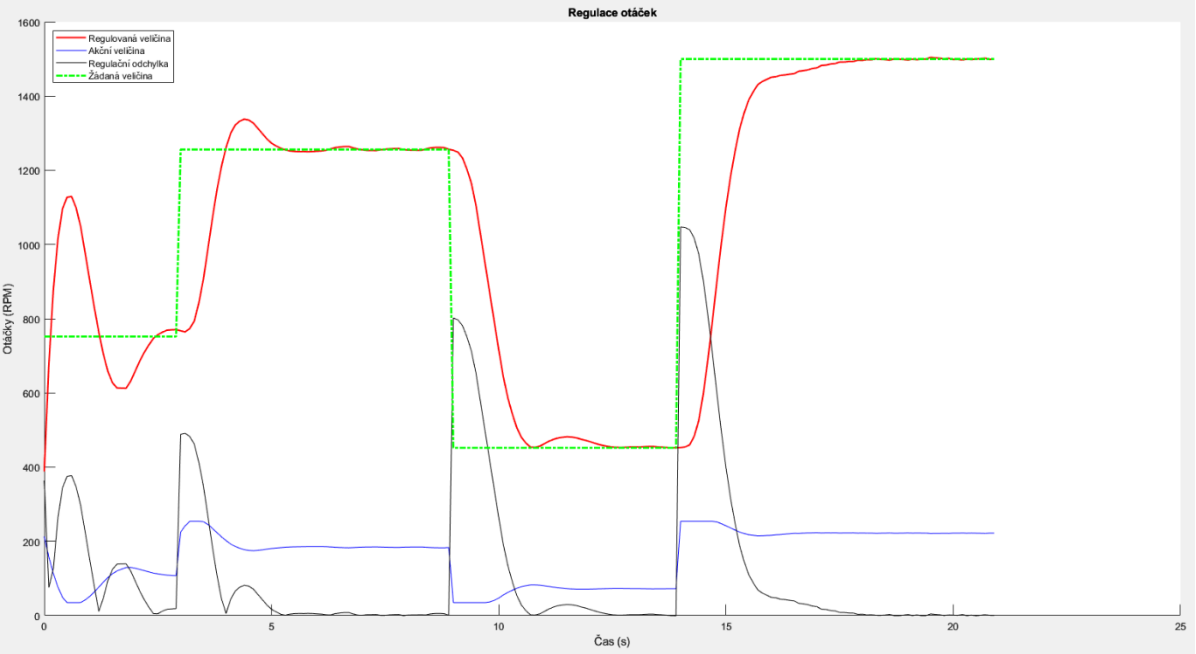


Obr. 6.4 – Průběh regulace P regulátorem

6.4 REGULACE PI REGULÁTOREM

Trvalou regulační odchylku lze odstranit zahrnutím integrační složky do regulátoru, což je patrné z obr. 6.5. Zde se již jedná o regulaci diskrétním PI regulátorem v přírůstkovém tvaru, což zabraňuje windup efektu, který je způsoben naakumulováním chyby v integračním členu.

```
[dataRPM, dataPWM, dataTime, dataError, dataSetpoint] =automat_PID('COM7',0.1, W, 0.2, 0.6, 0)
```

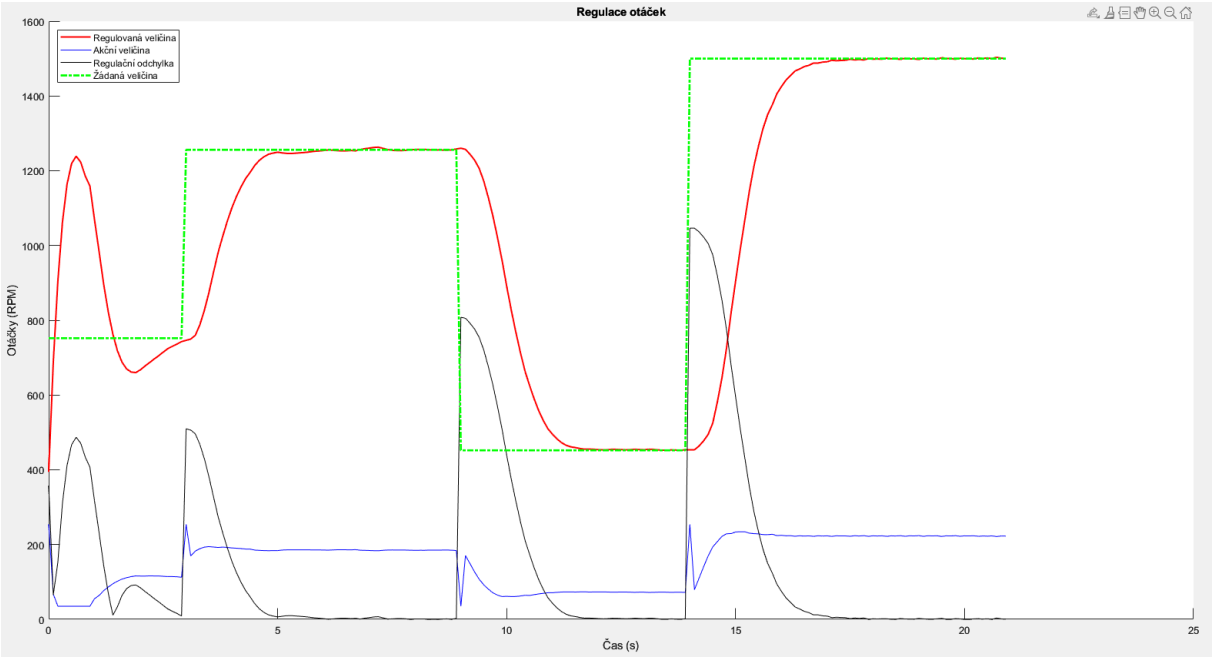


Obr. 6.5 – Průběh regulace PI regulátorem

6.5 REGULACE PID REGULÁTOREM

Zahrnutím derivační složky do regulátoru lze odbourat případné kmitavé průběhy částí regulace nebo překmity po změnách žádané veličiny, což je zjevné z obr 6.6.

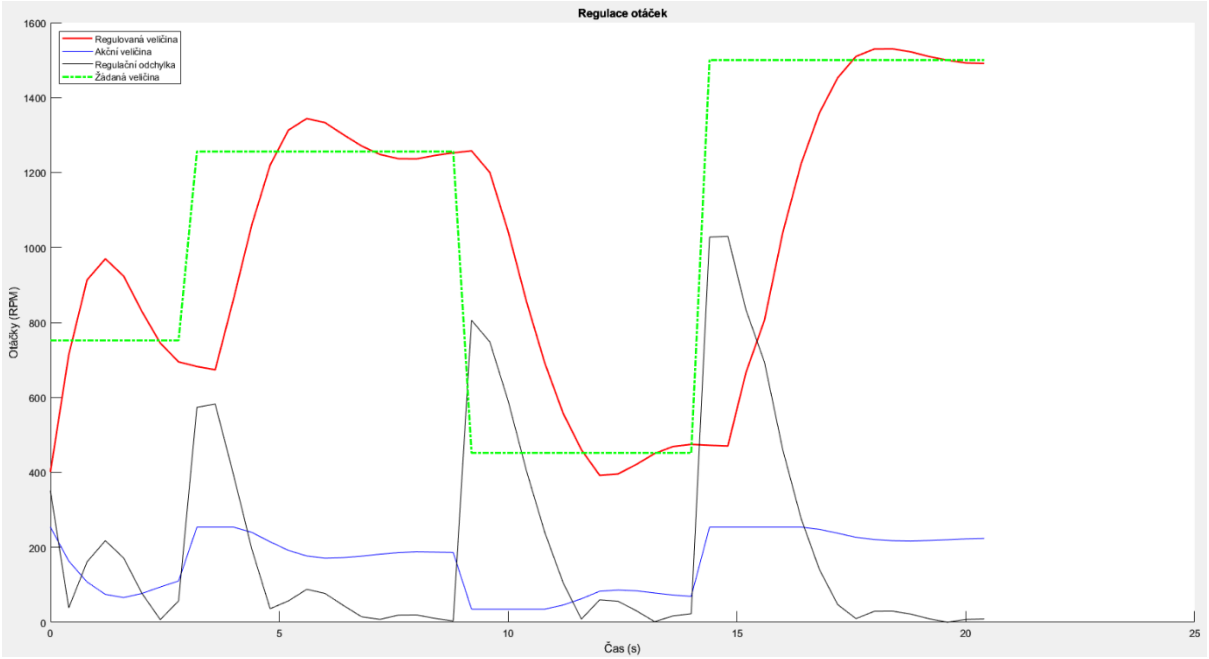
```
[dataRPM, dataPWM, dataTime, dataError, dataSetpoint] =automat_PID('COM7',0.1, W, 0.2, 0.6, 0.1)
```



Obr. 6.6 – Průběh regulace PID regulátorem

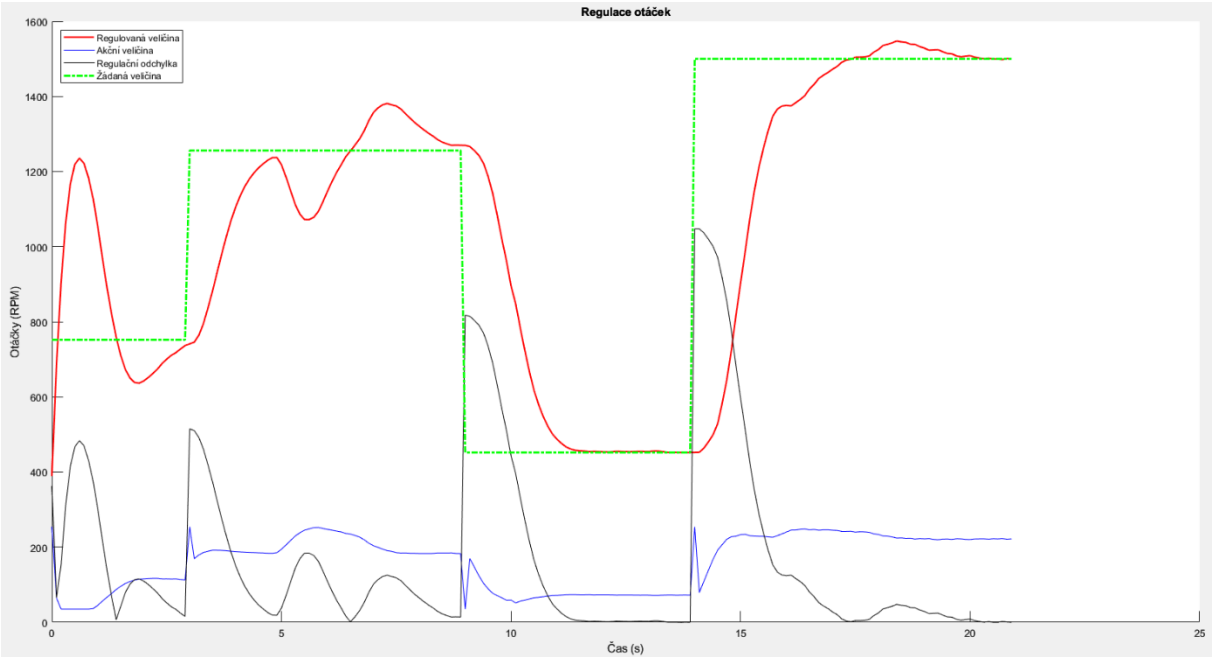
Pro demonstraci důležitosti vzorkovací periody byly zvoleny dva průběhy regulace. Na obr. 6.6 jde o průběh regulace se vzorkovací periodou 0,1 vteřiny, což je zřejmé ze spouštěcí funkce nad grafem, stejně jako parametry regulátoru. Na obr. 6.7 je regulace se stejným průběhem žádané veličiny, stejným nastavením parametrů regulátoru, ale vzorkovací periodou 0,4 vteřiny. Rozdíly jsou patrné na první pohled, regulátoru s větší vzorkovací periodou trvá déle, než stabilizuje regulovanou veličinu, což demonstruje vliv vzorkovací periody na regulační pochod.

```
[dataRPM, dataPWM, dataTime, dataError, dataSetpoint] =automat_PID('COM7',0.4, W, 0.2, 0.6, 0.1)
```



Obr. 6.7 – Průběh regulace PID regulátorem s periodou vzorkování 0,1 vteřiny

Otestována byla i reakce na poruchovou veličinu působící na ventilátor v podobě tlaku prstem na rotor. Reakce na působení poruchové veličiny je zřejmá z obr. 6.8, kdy cca v páté vteřině průběhu klesnou otáčky ventilátoru a zvyšuje se akční veličina. Cca po jedné vteřině přestala poruchová veličina působit. Tlak na rotor byl vyvíjen i v cca šestnácté vteřině průběhu, kde je zřejmé zpomalení růstu regulované veličiny.



Obr. 6.8 – Průběh s působením poruchové veličiny

ZÁVĚR

Byl sestrojen model řízení otáček ventilátoru s nastavitelnou periodou vzorkování, umožňující diskrétní verzi PID regulace s volitelnými parametry, dvoupolohovou regulaci s nastavením minimální a maximální akční veličiny a „manuální“ ovládání akční veličiny. Byla vytvořena jednoduchá konstrukce navrhnutá v 3D modelačním programu, na kterou je možné upevnit ventilátor i desku mikrokontroléru pro snazší manipulaci. Model je velmi skladný, jednoduchý na ovládání a jeho cena je nižší než dva tisíce korun, včetně materiálu pro 3D tisk. Byly vytvořeny tři ovládací funkce s uživatelskými vstupními parametry. Žádaná hodnota a „manuální“ akční veličina, ve formě vektoru hodnot, je také vstupním parametrem. Ovládací funkce je možné spouštět z prostředí Matlab. Výpočet akčních zásahů probíhá v mikrokontroléru Arduino. Na ventilátor je možné působit poruchovou veličinou. Toto zařízení je možné používat pro praktické demonstrace při výuce automatizace.

Na modelu byla ověřena správná funkčnost „manuálního“ ovládání akční veličiny a byly provedeny experimentální regulační pochody – dvoupolohové regulace a regulace diskrétními variantami různých variant PID regulátorů. Pro regulátory s integrační, respektive sumační složkou byly využity přírůstkové verze regulátorů.

POUŽITÁ LITERATURA

Aero 2, nedatováno. Aero 2. *Quanser.com* [online] [cit. 2024-04-08]. Dostupné z: https://www.quanser.com/products/aero-2/

Arduino, nedatováno. Arduino. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/

Arduino Due, nedatováno. Arduino Due. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-due?queryID=undefined

Arduino Leonardo, nedatováno. Arduino Leonardo. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-leonardo-with-headers?queryID=813f2a358fe2db701803190fca013c54

Arduino MEGA2560 REV3, nedatováno. Arduino MEGA2560 REV3. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-mega-2560-rev3?queryID=undefined

Arduino Nano 33 IoT, nedatováno. Arduino Nano 33 IoT. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-nano-33-iot?queryID=5a810332003a1004154c5e274c8e94f0

Arduino Nano, nedatováno. Arduino Nano. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-nano?queryID=undefined

Arduino Nano BLE, nedatováno. Arduino Nano BLE. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-nano-33-ble

Arduino Nano Every, nedatováno. Arduino Nano Every. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-nano-every?queryID=undefined

Arduino Uno REV3, nedatováno. Arduino Uno REV3. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-uno-rev3?queryID=cbeab054b9714a4cd91d0fc622e19ed7

Arduino Uno REV3 SMD, nedatováno. Arduino Uno REV3 SMD. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-uno-rev3-smd?queryID=undefined

Arduino Uno Wifi REV2, nedatováno. Arduino Uno Wifi REV2. *Arduino.cc* [online] [cit. 2024-04-08]. Dostupné z: https://store.arduino.cc/products/arduino-uno-wifi-rev2?queryID=undefined

4-pinový konektor ventilátoru, nedatováno. 4-pinový konektor. *Pavouk.org* [online] [cit. 2024-04-08]. Dostupné z: http://www.pavouk.org/hw/fan/en_fan4wire.html

8-bitový PWM signál, nedatováno. 8-bitový PWM signál. *bytes.usc.edu* [online] [cit. 2024-04-08]. Dostupné z: https://bytes.usc.edu/ee109/labs/lab8/

CHRIST, Robert D. a Robert L. WERNLI, 2014. Power and Telemetry. *The ROV Manual* [online]. 141–161 [cit. 2024-04-08]. Dostupné z: doi:10.1016/B978-0-08-098288-5.00007-5

DOSSENA Federico, nedatováno. Ovládání ventilátoru Arduinem. *fdossena.com* [online] [cit. 2024-04-08]. Dostupné z: https://fdossena.com/?p=ArduinoFanControl/i.md

EWALD Wolfgang, 2022. MOSFET jako přepínač. *Wolles elektronikkiste*. [online] [cit. 2024-04-08]. Dostupné z: https://wolles-elektronikkiste.de/en/the-mosfet-as-switch

GUNT, nedatováno. GUNT. *GUNT.de* [online] [cit. 2024-04-08]. Dostupné z: https://gunt.de/en/products/mechatronics/automation-and-process-control-engineering/simple-process-engineering-control-systems/glct-1:pa-148:ca-83

GUNT model RT 050, nedatováno. RT 050. *GUNT.de* [online] [cit. 2024-04-08]. Dostupné z: https://tangkasanugerah.com/products/detail/rt-050-training-system-speed-control-hsi

GUNT model RT 624, nedatováno. GUNT RT 624. *GUNT.de* [online] [cit. 2024-04-08]. Dostupné z: https://www.gunt.de/en/products/mechatronics/automation-and-process-control-engineering/simple-process-engineering-control-systems/flow-control-demonstration-unit/080.62400/rt624/glct-1:pa-148:ca-83:pr-1214

Hallův jev, nedatováno. Hallův jev. *melexis.com* [online] [cit. 2024-04-08]. Dostupné z: https://www.melexis.com/en/articles/hall-effect

HONC, Daniel a František DUSEK, 2013. MATLAB/Simulink support for GUNT control units. In: *2013 International Conference on Process Control (PC)* [online]. B.m.: IEEE, s. 534–539. ISBN 978-1-4799-0927-8. Dostupné z: doi:10.1109/PC.2013.6581466

HUMUSOFT, nedatováno. HUMUSOFT. *HUMUSOFT.cz* [online] [cit. 2024-04-08]. Dostupné z: https://www.humusoft.cz/matlab/details/

JIANG, Dong, Zewei SHEN, Qiao LI, Jianan CHEN a Zicheng LIU, 2021. Software and Hardware Implementation of Advanced PWM. In: [online]. s. 363–380. Dostupné z: doi:10.1007/978-981-33-4385-6_8

MAETSCHKE Stefan, 2024. Čtení otáček ventilátoru. *https://www.makerguides.com/* [online] [cit. 2024-04-08]. Dostupné z: https://www.makerguides.com/how-to-read-fan-speed-signal-with-arduino/

Matlab sériová komunikace, nedatováno. Matlab sériová komunikace. *MathWorks.com* [online] [cit. 2024-04-08]. Dostupné z: https://www.mathworks.com/help/matlab/ref/serialport.html

Matlab, nedatováno. Matlab. *Wikipedia.org* [online] [cit. 2024-04-08]. Dostupné z: https://cs.wikipedia.org/wiki/MATLAB

Model Aero 2, nedatováno. Model Aero 2. *Quanser.com* [online] [cit. 2024-04-08]. Dostupné z: https://www.quanser.com/products/aero-2/#details

Noctua, nedatováno. Noctua NF-A8 5V PWM. *Noctua.at* [online] [cit. 2024-04-08]. Dostupné z: https://noctua.at/en/nf-a8-5v-pwm

Noctua specifikace, nedatováno. Noctua specifikace. *Noctua.at* [online] [cit. 2024-04-08]. Dostupné z: https://noctua.at/en/nf-a8-5v-pwm/specification

PWM signál, nedatováno. Pulzně šířková modulace. *analogictips.com* [online] [cit. 2024-04-08]. Dostupné z: https://www.analogictips.com/wp-content/uploads/2017/04/Fig-1-pwm.gif

SLINTÁK Vlastimil, 2011. Hardwarové přerušení. *uArt.cz* [online]. [cit. 2024-04-08]. Dostupné z: https://uart.cz/271/arduino-a-preruseni/

ŠAFRANEK Jonáš, 2022. *Multifunkční monitorovací jednotka kvality ovzduší* [online]. Pardubice [cit. 2024-04-08]. Bakalářská práce. Univerzita Pardubice. Dostupné z: https://dk.upce.cz/bitstream/handle/10195/79572/SafranekJ_MultifunkcniMonitorovaci_LH_2022.pdf?sequence=1

ŠČEVÍK Petr, nedatováno. Číslicová regulace. *http://books.fs.vsb.cz* [online] [vid. 2024-04-08]. Dostupné z: http://books.fs.vsb.cz/cislicovaregulace/data/kapitola4.html#kap_4_1

TAKÁCS Gergely, nedatováno. AeroShield. *AutomationShield* [online] [cit. 2024a-04-08]. Dostupné z: https://github.com/gergelytakacs/AutomationShield/wiki/AeroShield

TAKÁCS Gergely, nedatováno. HeatShield. *AutomationShield* [online] [cit. 2024b-04-06]. Dostupné z: https://github.com/gergelytakacs/AutomationShield/wiki/HeatShield

TAKÁCS Gergely, nedatováno. HeatShield. *AutomationShield* [online] [cit. 2024c-04-08]. Dostupné z: https://github.com/gergelytakacs/AutomationShield/wiki/HeatShield

TecQuipment, nedatováno. TecQuipment. *TecQuipment.com* [online] [cit. 2024-04-08]. Dostupné z: https://www.tecquipment.com/process-control/pid-process-control

TecQuipment model řízení teploty, nedatováno. TecQuipment temperature process training model. *TecQuipment.com* [online] [cit. 2024-04-08]. Dostupné z: https://www.tecquipment.com/temp-process-training-system

Virtual Aero 2, nedatováno. Virtual Aero 2. *Quanser.com* [online] [cit. 2024-04-08]. Dostupné z: https://www.quanser.com/products/qlabs-virtual-aero-2/#details

ZAPLATÍLEK Karel, 2020. *MATLAB® pro začínající uživatele*. Brno: Knihovnicka.cz. ISBN ISBN 978-80-263-1589-6.