UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

SOUSTAVA DVOU VENTILÁTORŮ S REGULACÍ OTÁČEK

Michal Pulkráb

Bakalářská práce

2022

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Michal Pulkráb**
Osobní číslo:	**I19191**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Řízení procesů**
Téma práce:	**Soustava dvou ventilátorů s regulací otáček**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Cílem práce je vytvořit praktickou úlohu pro zpětnovazební řízení skládající se ze soustavy dvou ventilátorů s možností měření otáček. Jako napájecí zdroj bude využit adaptér o napětí max. 12V. Součástí řešení bude jednoduché rozhraní pro ovládání ventilátorů a zobrazování aktuálních hodnot otáček, např. s využitím software MATLAB. V druhé fázi bude realizováno zpětnovazební řízení otáček jednoho z ventilátorů regulátorem PI, druhý ventilátor bude využit jako nastavitelný zdroj poruchy. Jádrem řídicího systému bude zvolený univerzální vývojový modul s jednočipovým počítačem. Regulátor bude nastaven vhodnou metodou. Získané časové průběhy budou v práci prezentovány.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

HLAVA, J. *Prostředky automatického řízení II: Analogové a číslicové regulátory, elektrické pohony, průmyslové komunikační systémy*. Praha: Vydavatelství ČVUT, 2000.
CVEJN, J. *Řízení procesů* [online]. Pardubice: Univerzita Pardubice, FEI, 2012-. Elektronický studijní materiál k předmětu Automatizace 1.
LÁNÍČEK, R. *Elektronika, obvody, součástky, děje*, Praha: BEN –technická literatura, 1998.

Vedoucí bakalářské práce: **doc. Ing. Jan Cvejn, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **17. prosince 2021**
Termín odevzdání bakalářské práce: **13. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Daniel Honc, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 7. ledna 2022

Prohlášení

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

Michal Pulkráb

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce doc. Ing. Janu Cvejnovi, Ph.D. za veškeré rady, které mi velice pomohly. Také za jeho věnovaný čas a trpělivost při konzultacích.

V Pardubicích dne dd. mm. rrrr

Michal Pulkráb

ANOTACE

Bakalářská práce je věnována praktické úloze pro zpětnovazební řízení, která se skládá ze dvou ventilátorů, kde jeden ventilátor slouží jako volně nastavitelná porucha a druhý je regulován pomocí PI regulátoru. Funkčnost je realizována pomocí vývojového modulu Arduino. Zobrazení otáček je realizováno v software MATLAB.

KLÍČOVÁ SLOVA

regulace, ventilátor, MATLAB, Arduino, PI regulátor

TITLE

A pair of ventilators with rotation speed regulation

ANNOTATION

The Bachelor thetis is devoted to a practical task for feedback control, which consists of two fans, where one fan serves as a freely adjustable fault and the other is controlled by a PI controller. The functionality is implemented using the Arduino development module. Speed display is realized using MATLAB software.

KEYWORDS

Regulation, fan, MATLAB, Arduino, PI controller

OBSAH

SEZNAM ZKRATEK A ZNAČEK

A/D	analogově digitální převodník
PI	proporcionálně integrační (regulátor)
PID	proporcionálně integračně derivační (regulátor)
PWM	pulzně šířková modulace
USB	universal serial bus

SEZNAM SYMBOLŮ VELIČIN A FUNKCÍ

d	porucha
e	regulační odchylka
$F(s)$	přenos funkce
$h(t)$	přechodová funkce
k	statické zesílení
k_1	zesílení
R	regulátor
r_0	váha proporcionální složky
r_1	váha integrační složky
S	soustava
T	časová konstanta, s
T_n	doba náběhu, s
T_u	doba průtahu, s
u	akční veličina
w	žádaná veličina
y	výstupní veličina
$\delta(t)$	Diracův impulz
$\eta(t)$	jednotkový skok

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

ÚVOD

Cílem zadané bakalářské práce je vytvořit úlohu pro regulaci otáček soustavy dvojice ventilátorů pomocí PI regulátoru. Pro realizaci je využit mikropočítačový modul Arduino. Dalším cílem je vytvořit zobrazení otáček v reálném čase obou ventilátorů a požadované hodnoty otáček. Součástí práce je také příslušné elektronické zapojení a vhodný napájecí zdroj pro ventilátory.

Práce poukazuje na reálné problémy při realizaci regulace a vzájemném ovlivňování obou ventilátorů mezi sebou.

1 TEORETICKÁ ČÁST

V té to části práce jsou rozebrány základy automatizace a základní informace o využití mikropočítačových modulů.

1.1 ARDUINO

Vznik prvního Arduina začal roku 2005 ve městě Ivrea, kde se členové italského Interaction Design Institute rozhodli, že vytvoří set, který by byl dostupný pro běžné lidi a nebyl drahý, jako například BASIC Stamp. Arduino se velice rychle uchytilo a bylo dále dostupné po celém světě.

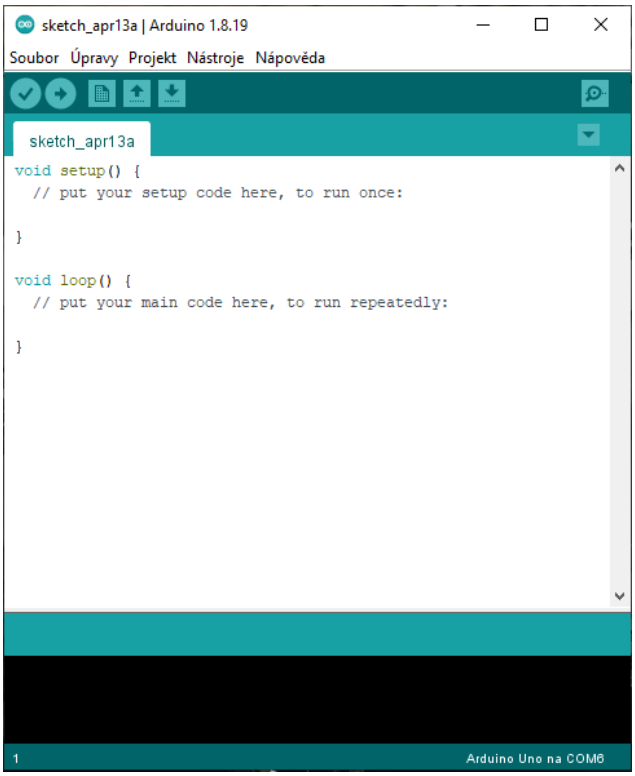
Arduino, co se týče programové části, bylo založeno na knihovně jazyku C++. U tohoto programovacího setu jde hlavně o jednoduchost programování, kde je možnost použít rozsáhlou škálu knihoven. Na trhu je pestré množství jiných programovacích platforem, ale tato platforma je i v současné době velice oblíbená a prodalo se již několik stovek tisíc desek od Arduina. Tato platforma spolupracuje například se společností Intel (Voda, 2017).

Arduino IDE (integrated development enviroment, integrované vývojové prostředí) je software, který vznikl z výukového prostředí Processing. Arduino je napsáno v programovém jazyce Java (Voda, 2017).

Software Arduino IDE je možné získat na oficiálních stránkách Arduina, na webu https://www.arduino.cc/. Software je zcela bezplatný, takže ho lze využívat dle libosti. Programovací prostředí je dostupné nejen na systému Windows, ale také na Mac OS nebo Linux. U systému Windows se stáhne ZIP soubor a po rozbalení ZIP souboru do zvolené složky na disku je program zcela funkční.

Složka obsahuje důležité podsložky, jako jsou ovladače, které jsou určené pro komunikaci desky s počítačem. Dále se ve složce nachází složka knihovny, kde se ukládají stažené knihovny. Také se zde nachází složka s příklady, což může pomoci při prvním programování.

Před zapnutím samotné aplikace Arduino IDE je potřeba si vytvořit složku Arduino v dokumentech počítače. Do této složky se budou ukládat nadále vytvořené programy (Voda, 2017).



Obrázek 1.1 – Arduino IDE

Na obrázku 1.1 v levém rohu nahoře pod názvem a pod základním pásem karet lze vidět pět tlačítek. První tlačítko slouží pro ověření, zda se v programu nenacházejí chyby. Pokud se nějaká chyba najde, tak se dole v černém okně zobrazí a zvýrazní daný řádek, na kterém se chyba nachází. Následující tlačítko nahraje na desku program, je-li připojena. Poslední tři tlačítka jsou pro založení nového programu, otevření již vytvořeného programu a pro uložení programu.

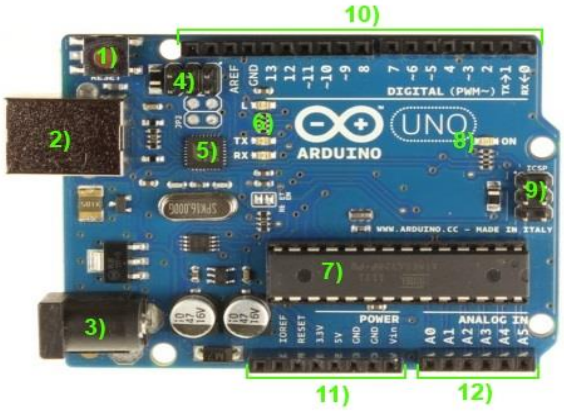
Před nahráním programu je potřeba mít k počítači připojenou desku Arduino. V nástrojích lze zjistit, zda je vybrána příslušná deska a správný port.

Při otevření nového programu je předem připravený kód „setup a loop“. Nad setupem se zavádí globální proměnné a předdefinují se příslušné piny desky. Ve funkci setup se nastavuje například tzv. „baudrate“, nastavování digitálních pinů jako výstup a další potřebné náležitosti, které jsou potřeba pouze jednou.

Ve funkci „loop“ je napsán program, který je v nekonečné smyčce.

Na pravé straně se nachází tlačítko pro zapnutí sériového monitoru, kterým lze pomocí příkazu „Serial.print();“ zobrazit obsah, který je uložen v proměnné.

1.2 POPIS DESKY ARDUINO UNO



Obrázek 1.2 – Arduino Uno popis desky (Voda, 2017)

Arduino Uno je v dnešní době nejvíce rozšířená deska. Je vybavena procesorem ATmega328. Pro napájení a programování slouží USB port typu B. Dále pro napájení můžeme využít i Jack konektor (Voda & HW Kitchen, 2017).

1) Číslo jedna na obrázku 1.2 představuje tlačítko pro reset. Toto tlačítko resetuje nahraný program a pustí ho od začátku celý znovu.
2) Jedná se o USB konektor typu B. Konektor je využíván pro programování a napájení desky Arduino.
3) Napájecí DC konektor. Využívá se k napájení desky, pokud není využito napájení z USB.
4) Jde o ICSP hlavici, kterou lze použít pro programování USB-seriál převodníku.
5) Tato součástka je USB-serial převodník. Jeho účelem je zajistit komunikaci mezi počítačem a ATmega328.
6) LED diody L, TX a RX. LED dioda L je připojená na digitální pin třináct. LED diody TX a RX jsou použity pro indikaci komunikace po sériové lince.
7) Čip ATmega328.
8) LED dioda pro indikaci připojeného napájení.
9) ICSP hlavice, která je na desce pro externí programování čipu.
10) V této sadě pinu lze nalézt třináct digitálních pinů pro připojení obvodů, kde piny s „vlnkou“ jsou piny podporující PWM. Dále je zde GND pin a AREF pin.

11) Piny sloužící pro napájení například obvodů. Je zde možnost 5 V nebo 3,3 V. Nebo lze do desky přivést vstupní napětí, ovšem ne větší, něž je napájecí napětí Arduina.
12) Jde o vstupní analogové piny sloužící pro měření. Netřeba je v programu definovat jako vstup (Voda, 2017).

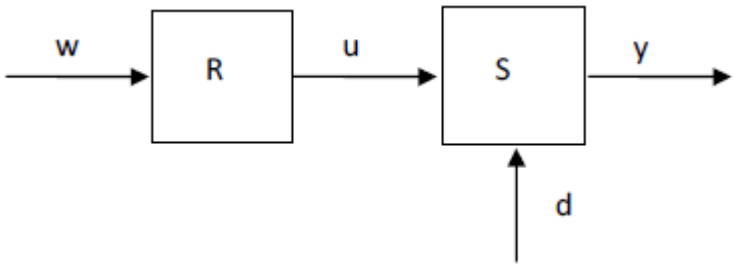
1.3 AUTOMATICKÉ ŘÍZENÍ

Úkolem automatizace je, aby ve všech možných oborech, kde je potřeba řízení nějakého procesu, byl zásah člověka co nejmenší. V ideálním případě, aby nebyl zásah člověka vůbec vyžadován.

1.4 ZÁKLADNÍ POJMY

1.4.1 Otevřené řízení

Na obrázku 1.3 lze vidět blokové schéma otevřeného řízení, které má za cíl, aby výstupní veličina byla v ideálním případě shodná s požadovanou veličinou. Otevřené řízení lze použít pouze při znalosti působení vnějších vlivů. Na obrázku 1.3 *w* představuje požadovanou veličinu, která je vstupem do regulátoru R, který má výstup akční veličinu *u*. Akční veličina spolu s poruchou „d“, působí na řízený systém S“, jehož výstupem je výstupní veličina *y*.



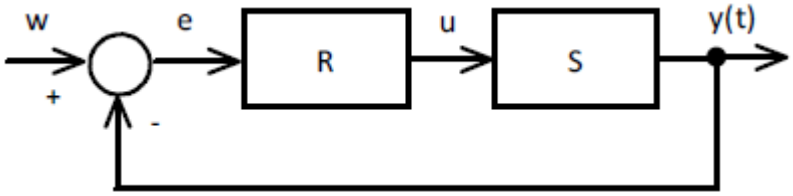
Obrázek 1.3 – Otevřené řízení (Cvejn, 2012)

1.4.2 Zpětnovazební regulace

Cílem regulace je, aby výstupní veličina systému byla shodná s požadovanou veličinou. Příkladem regulace může být například vytápění pokojové místnosti na požadovanou teplotu. Nastavená teplota je např. na 23°C. V zimním období je venkovní teplota například - 10°C. Vnitřní teplota je 15°C. Topení se sepne a topí se do té doby, než má místnost takovou teplotu,

která je nastavena jako žádaná. Poté se topení buď vypne, nebo sníží (záleží na druhu regulace) proto, aby vykompenzovalo ztráty. Pokud se topení vypne a teplota klesne pod určitou úroveň, tak se topení znovu sepne, aby rozdíl mezi žádanou teplotou a teplotou v pokoji byl v rozmezí nastavených mezí.

Na obrázku 1.4 lze vidět žádanou veličinu *w*. To je požadovaná veličina, která by měla být shodná s výstupní veličinou *y*. Ze soustavy vystupuje záporná zpětná vazba, která jde do součtového členu, kde se odečítá výstupní veličina od požadované veličiny. Jejich výsledkem je regulační odchylka *e*. Tato regulační odchylka je vstupem regulátoru, který podle velikosti regulační odchylky nastavuje akční veličinu *u*, která vystupuje z regulátoru a působí na regulovanou soustavu (Cvejn, 2012).



Obrázek 1.4 – Zpětnovazební regulační obvod (Cvejn, 2012)

1.4.3 Dynamické systémy

Stav dynamického systému je definován jako minimum potřebných veličin, jejichž aktuální hodnota vypovídá o minulém průběhu systému. Potom při znalosti průběhu vstupních veličin lze stanovit nadcházející průběh systému.

Stavové veličiny jsou veličiny, kterými je popsán stav systému. Jsou to veličiny výstupní a vnitřní.

1.4.4 Dynamické systémy a jejich typy modelů

Zjištění chování reálného lze provést těmito způsoby:

1) simulací,
2) měřením,
3) matematicko-fyzikální analýzou.

Modely jsou:

1) fyzikální,
2) matematické.

U fyzikálního modelu uvažujeme fyzikální zákony (například bilance hmoty, atd).

Pro získání matematického modelu je využito matematického popisu fyzikálních jevů, které jsou dostatečně elementární, aby byly respektovány vazby mezi prvky struktury systému. Hloubka a komplexnost uvažovaných jevů záleží na účelu využití modelu. Čím přesnější model, tím obtížnější je vytvoření a často i využití. Nejčastěji se dynamické systémy popisují pomocí soustav diferenciálních rovnic (Cvejn, 2012).

1.5 CHARAKTERISTIKY DYNAMICKÝCH SYSTÉMŮ

1.5.1 Přechodová charakteristika

Přechodová charakteristika je reakcí systému na jednotkový skok v čase $t = 0$. To platí za předpokladu, že počáteční podmínky jsou nulové. Přechodová charakteristika má výhodu, že je snadno změřitelná.

Matematický popis jednotkového skoku je

$$\eta(t) = 0 \; pro \; t \in (-\infty, 0)$$
$$\eta(t) = 1 \; pro \; t \in \langle 0, \infty), \tag{1.1}$$

kde $\eta(t)$ je jednotkový skok,

t je čas, s.

V reálných podmínkách lze změřit přechodovou charakteristiku na reálném systému v ustáleném bodě výstupní veličiny, označeném y_0. Systém zareaguje v tomto bodě na skokovou změnu vstupní veličiny velikosti Δu.

Pak pro hodnoty přechodové charakteristiky platí

$$h(t) = \frac{y(t) - y_0}{\Delta u}, \tag{1.2}$$

kde $h(t)$ je označení přechodové charakteristiky,

$y(t)$ je hodnota výstupní veličiny, která se mění v čase,

y_0 je hodnota výstupu před jednotkovým skokem,

Δu je akční veličina.

1.5.2 Impulzní charakteristika

Impulzní charakteristika je reakcí systému na Diracův impulz, který je pouze teoretický. Nelze ho v reálném světě realizovat. Impulzní charakteristika se většinou označuje g(t) (Cvejn, 2012).

Diracův impulz je matematicky popsán takto

$$\delta(t) = 0 \; pro \; t \neq 0, \int_{-\infty}^{\infty} \delta(t)dt = 1. \tag{1.3}$$

kde $\delta(t)$ je Diracův impulz.

Vztah mezi impulzní a přechodovou charakteristikou je

$$g(t) = \frac{dh(t)}{dt}, \tag{1.4}$$

kde d$h(t)$ je derivace přechodové charakteristiky,

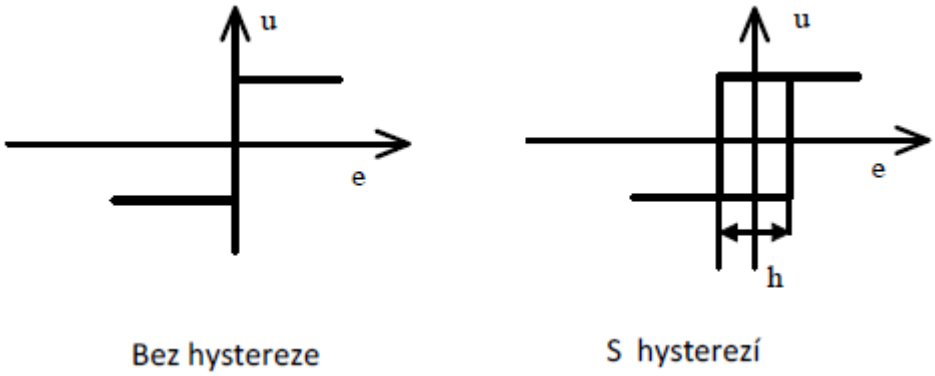
dt je derivace času.

1.6 DVOUPOHOVÝ A PID REGULÁTOR

1.6.1 Dvoupolohový regulátor

Dvoupolohový regulátor v závislosti na žádané hodnotě spíná dvě různé hodnoty akční veličiny. Klasickým případem pro využití dvoupolohového regulátoru je například regulace výšky hladiny.

Relé pracuje s malým zpožděním a tím u soustav prvního řádu dojde k rychlému spínání zdroje relé. To by mohlo vést k opotřebení relé a z tohoto důvodu se zavádí uměle vytvořená hystereze, která je značena jako h. Na obrázku 1.5 je graf hystereze, kde na ose y je akční veličina a na ose x regulační odchylka. Čím je hystereze větší, tím dojde k menší frekvenci spínání, ale zvětší se tím rozkmit regulované veličiny (Cvejn, 2012).



Obrázek 1.5 – Závislost akční veličiny na regulační odchylce (Cvejn, 2012)

Při použití dvoupolohového regulátoru výstupní veličina osciluje, což může být nežádoucí. Proto je vhodnější použít například regulátor PI nebo PID, kde je regulace kvalitnější.

1.6.2 PID regulátor

PID regulátor je jedna z nejvíce využívaných možností pro regulaci, není ale jediná. PID regulátor pracuje s regulační odchylkou, integrálem regulační odchylky, popřípadě s derivací regulační odchylky.

PID regulátor má následující výhody:

1) lze sestavit pomocí z elektronických součástek,
2) je jednoduchý,
3) je univerzální.

Obecná rovnice PID regulátoru je ve tvaru

$$u(t) = r_0 e(t) + r_1 \int_0^t e(t)\,dt + r_2 \frac{de(t)}{dt}, \tag{1.5}$$

kde $u(t)$ je akční veličina,

r_0 je zesílení proporcionálního členu,

$e(t)$ je regulační odchylka,

r_1 je zesílení integračního členu,

r_2 je zesílení derivačního členu.

Regulátor má tři složky a to:

1) proporcionální (P),

2) integrační (I),
3) derivační (D).

PID regulátor má následující výhody:

1) P-regulátor (zesílení zpětné vazby, r_1 a r_2 parametry jsou nulové),
2) PD-regulátor (zesílení zpětné vazby a přidaná derivační složka, r_1 parametr je nulový),
3) PI-regulátor (zesílení zpětné vazby a přidaná integrační složka, r_2 parametr je nulový),
4) I-regulátor (pouze integrační složka, parametry r_0 a r_2 jsou nulové).

1.6.3 Složky PID regulátoru a jejich význam

U P-složky se jedná o zesílení záporné zpětné vazby. Při vzrůstajícím zesílení dojde k rychlejšímu regulačnímu ději, avšak při příliš vysokém zesílení se soustava rozkmitá. Což může zapříčinit vzrůstající rozkmit (nestabilitu).

U statických soustav P-regulátor není dostačující. Nedocílí se požadované hodnoty, jelikož pro výstupní veličinu různou od nuly je třeba nenulový výstup z regulátoru a nenulová regulační odchylka.

Při příliš velkém zesílení dojde k saturaci výstupu soustavy, jelikož výstup akční veličiny je omezen v určitém rozsahu. Příkladem může být regulační ventil. Ten je omezen v rozsahu od uzavřeného do plně otevřeného, proto se regulátor bude chovat jako dvoupolohový regulátor (Cvejn, 2012).

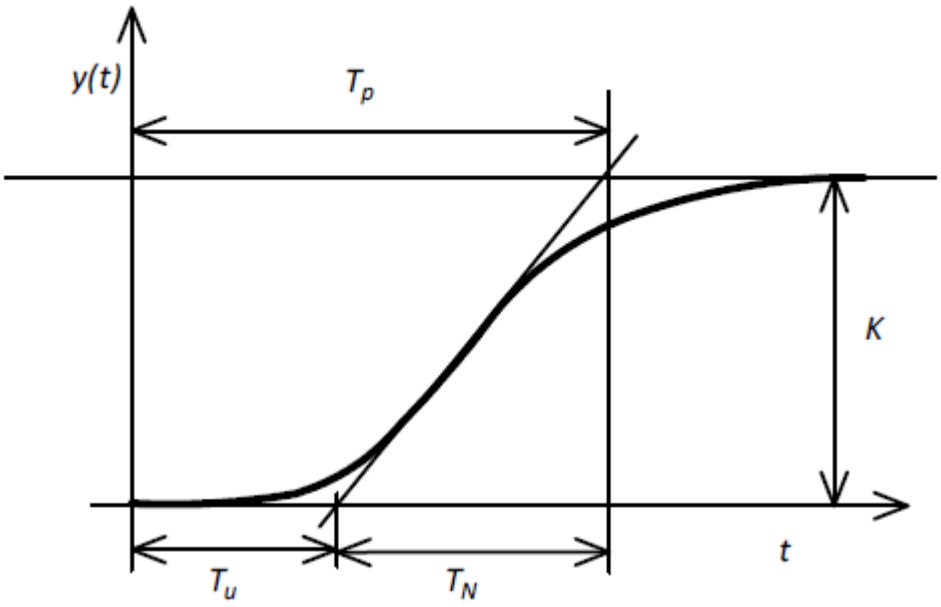
Integrační složka pomáhá docílit, aby byla regulační odchylka nulová. Avšak přidáním integrační složky se zvýší řád a tím i prodlouží regulační děj (Cvejn, 2012).

Derivační složka urychluje reakci zpětné vazby a tím urychlí průběh regulace. Hlavně u soustav vyššího řádu a soustav, kde je dopravní zpoždění (Cvejn, 2012).

1.7 METODY PRO NASTAVENÍ PARAMETRŮ PID REGULÁTORU

Metod pro nastavení PID regulátoru je tu celá řada. Nejvíce využívaná metoda je Zieglera a Nicholse. Tato metoda nevyžaduje znalost přenosu systému. Při znalosti přenosu je k dispozici řada metod, které zajištují mnohem lepší průběh regulace, ale jsou o to komplikovanější.

1.7.1 Zieglerova-Nicholsova metoda na základě přechodové charakteristiky



Obrázek 1.6 – Získání parametrů z přechodové charakteristiky (Cvejn, 2012)

Z přechodové charakteristiky soustavy se vyčtou časové průběhy doba průtahu – T_u, doba náběhu – T_n a zesílení k_1. Dále jsou dopočítány parametry regulátoru dle tabulky 1.1.

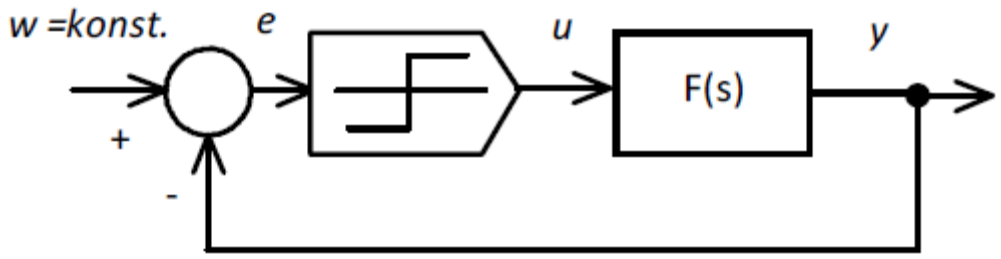
Tabulka 1.1 – ZN metoda – nastavení regulátoru z přechodové char.

Regulátor	k_P	T_I	T_D
P	$\frac{T_n}{k_1 T_u}$	-	-
PI	$0{,}9\frac{T_n}{k_1 T_u}$	$3{,}33T_u$	-
PID	$1{,}2\frac{T_n}{k_1 T_u}$	$2T_u$	$0{,}5T_u$

Dobou průtahu a náběhu lze rozumět jako určitou aproximaci modelem prvního řádu s dopravním zpožděním.

1.7.2 Zieglerova-Nicholsova metoda na základě kritických parametrů

Při aplikaci Zieglerovy-Nicholsovy metody na základě kritických parametrů je třeba přivést regulovanou soustavu na mez stability (periodické kmitání výstupní veličiny s periodou T_k). To lze realizovat přímo změnou zesílení P-složky, nebo je možné využít dvoupolohového regulátoru, který přepíná hodnotu $u = \pm M$. Výstupní veličina kmitá s určitou amplitudou kolem konstantně nastavené žádané veličiny – *w*.



Obrázek 1.7 – Soustava s relé pro určení kritického zesílení (Cvejn, 2012)

Lze využít harmonický rozklad signálu

$$u(t) = \frac{4M}{\pi} \sum_{k=0}^{\infty} \frac{1}{2k+1} \sin(\omega(2k+1)t), \tag{1.6}$$

kde $u(t)$ je akční veličina,

M je přepínaná hodnota akční veličiny,

k je rostoucí konstanta.

Při zvyšujícím se zesílení klesá amplituda jednotlivých složek a jelikož soustava tlumí vysoké frekvence, tak lze výstup regulátoru nahradit první harmonickou složkou

$$u(t) \approx \frac{4M}{\pi} \sin \omega t. \tag{1.7}$$

Regulační obvod kmitá (je na mezi stability), jelikož výstupní veličina harmonicky kmitá kolem žádané hodnoty a fázový posun mezi vstupním a výstupním signálem soustavy je roven -π.

Na vstupu relé je signál s amplitudou o hodnotě, takže platí

$$r_k = \frac{4M}{A\pi}, \tag{1.8}$$

kde r_k je kritické zesílení,

A je amplituda.

Kritickou frekvenci lze určit z kmitavého průběhu výstupní veličiny o periodě T_k

Tabulka 1.2 – ZN metoda – nastavení regulátoru z kritických parametrů

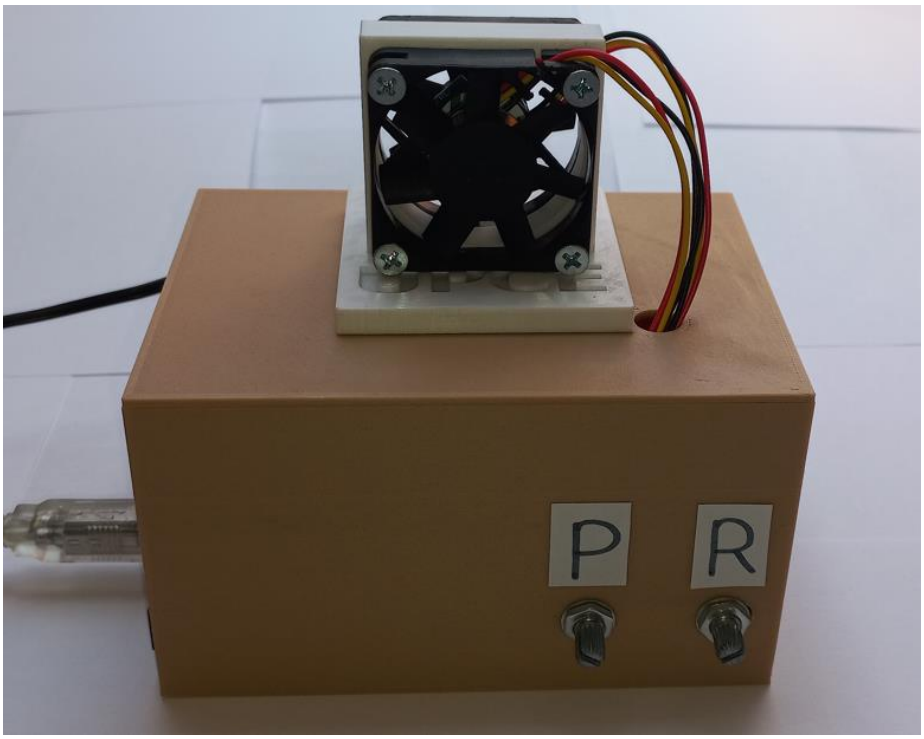
Regulátor	k_P	T_I	T_D
P	$0{,}5r_k$	-	-
PI	$0{,}45r_k$	$0{,}83T_k$	-
PID	$0{,}6r_k$	$0{,}5T_k$	$0{,}125T_k$

2 PRAKTICKÁ ČÁST

Cílem praktické části je realizace ovládání ventilátorů pomocí potenciometrů. Jeden ventilátor je využit jako porucha a příslušným otáčením potenciometru ovládá jeho rychlost otáčení. Druhý ventilátor je ovládán taktéž potenciometrem, ale s tím rozdílem, že potenciometr je zde pouze pro nastavení požadovaných otáček, neovládá ventilátor napřímo. Potenciometr je vstupní veličinou. Dalším úkolem je naprogramování regulátoru PI a měření otáček obou ventilátorů.

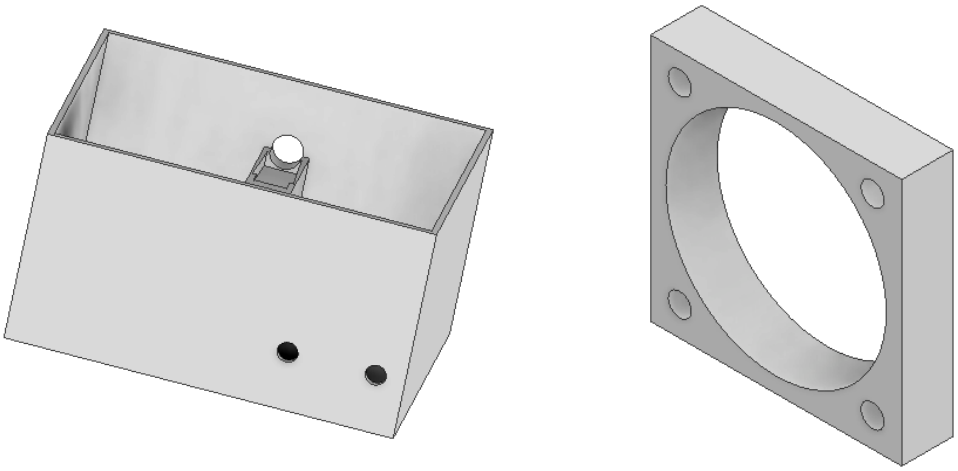
Zobrazení otáček obou ventilátorů a hodnoty požadovaných otáček je naprogramováno v softwaru MATLAB.

2.1 VYTVOŘENÉ ZAŘÍZENÍ



Obrázek 2.1 – Výsledná sestava praktické části

Na obrázku 2.1 lze vidět dva potenciometry. Jeden je označen písmenem „P“ (ovládání poruchového ventilátoru, který je na obrázku umístěn směrem k potenciometrům) a druhý písmenem „R“ (nastavování požadované hodnoty otáček). Z levé strany krabičky je připojen USB kabel pro napájení desky Arduina a posílání dat na sériovou linku. Ze zadní strany krabičky je připojen zdroj pro napájení ventilátorů.



Obrázek 2.2 – Hlavní modely zařízení

Na obrázku 2.2 jsou hlavní modely sestavy zařízení, které je na obrázku 2.1. Vytvořeny jsou v programu Inventor a jsou součástí přílohy. Vedlejší části konstrukce, jako jsou víko krabice, která se nachází na obrázku 2.2 na levé straně, nebo podstava pro držák ventilátorů, který je na obrázku 2.2 na pravé straně, jsou také součástí přílohy.

2.2 VÝBĚR VENTILÁTORŮ

2.2.1 Volba typu ventilátoru

Pro realizaci praktické části je vybrán ventilátor od značky SUNON. Jedná se o 12 V ventilátor s maximálním příkonem 1,32 W. Model ventilátoru je ME45101V1-000U-G99. Z modelu ventilátoru lze vyčíst základní informace o daném ventilátoru.

1) ME – sériové číslo,
2) 45 – rozměr ventilátoru,
3) 10 – tloušťka ventilátoru,
4) 1 – napájecí napětí je 12 V,
5) V – VAPO ložisko,
6) 1 – vysoká rychlost,
7) 000U – kód zákazníka,
8) G99 – kód funkce.

Vybraný ventilátor má tři vodiče. Červený a černý vodič jsou pro napájení. Žlutý vodič je signálový, který je zde přítomný pro snímání pulzů při běhu ventilátoru.

Ventilátor nemá vodič pro řízení pomocí PWM modulace. Je tedy potřeba k ventilátoru přidat obvod, aby bylo možné ventilátor ovládat.

2.2.2 Snímání otáček ventilátoru

K signálnímu vodiči je potřeba připojit tzv. pull-up rezistor, který je připojen na stejné napájení, jako je připojen ventilátor. Zabudovaný Hallův senzor každou půl otáčku vyšle pulz a nastaví signální vodič na logickou nulu. Poté pull-up rezistor nastaví vodič na logickou jedničku. Podle datasheetu je potřeba vypočítat minimální hodnotu rezistoru.

Vzorec pro pull-up rezistor je

$$R_L = \frac{V_{CC}}{I_C}, \tag{2.1}$$

kde R_L – vypočítaná hodnota pull-up rezistoru, Ω,

V_{CC} – napájecí napětí ventilátoru, V,

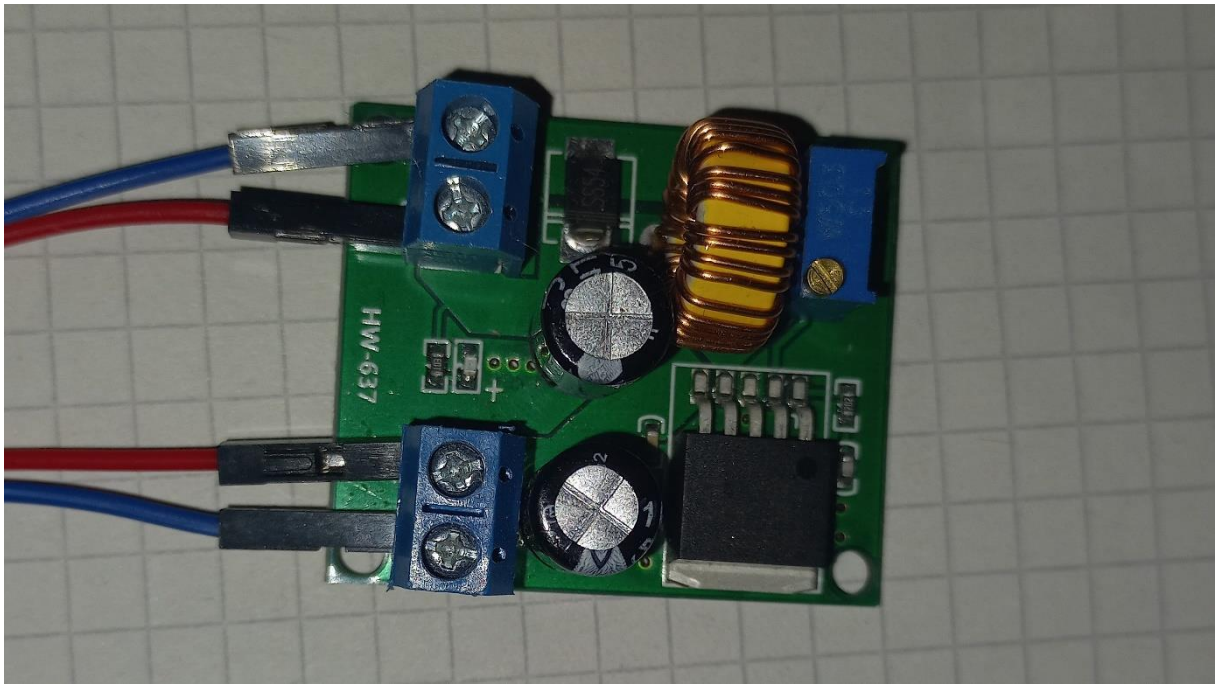
I_C – maximální proud zjištěný z datasheetu, A.

2.2.3 Princip Hallova senzoru

Hallův senzor využívá působení tzv. Hallova jevu, který vznikne, pokud je v přítomnosti magnetického pole vodič a vznikne na něm napětí. Magnetické pole působí kolmo na procházející proud vodičem a kolmo na magnetický tok siločar.

Hallův senzor reaguje na změny magnetického pole. Jde o tenkou destičku tvořenou z polovodiče typu – P. Pokud žádné magnetické pole nepůsobí, tak se nosiče nábojů pohybují v přímce. Při působení magnetického pole se nosiče nábojů nahromadí na opačné straně destičky a vznikne malé napětí, které lze měřit (Herres, 2015).

2.3 NÁPÁJENÍ OBVODU PRO VENTILÁTORY



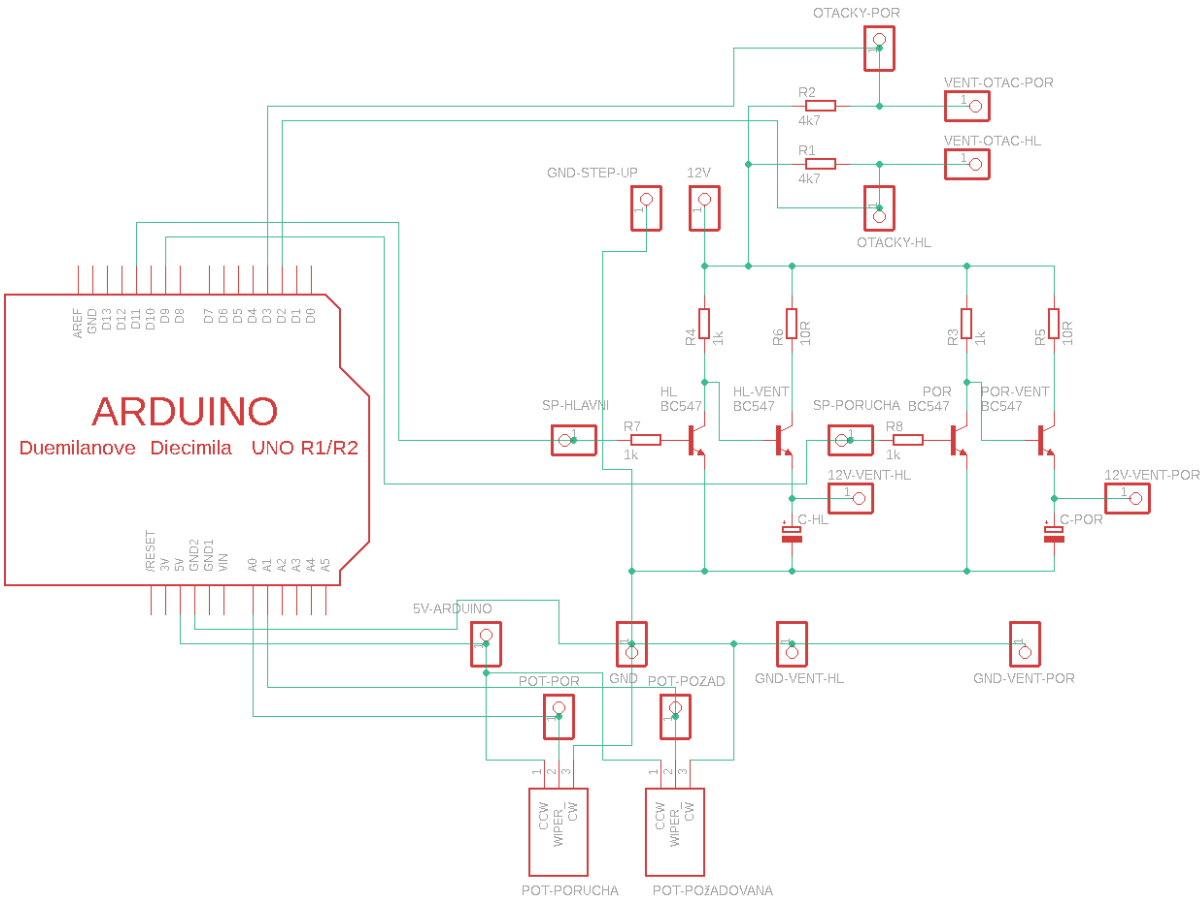
Obrázek 2.3 – Step-up modul

Při prvotním zapojení byl vybrán step-up modul – HW-637, který je dostupný a levný měnič napětí pro napájení ventilátorů. Napájení step-up modulu mělo být realizováno napájením z desky Arduina. Avšak step-up modul měl příliš velký odběr a to způsobovalo, že vznikl pokles napájecího napětí a na potenciometrech pro ovládání ventilátorů nebylo 5 V. To mělo za následek nižší hodnotu napájení z Arduina.

Nastavení modulu na 12 V se provedlo pomocí multimetru. Na obrázku 2.2 lze vidět dvě modré svorkovnice. Dolní svorkovnice je pro vstupní napájení a horní svorkovnice pro výstup. Multimetr se připojí do horní svorkovnice a pomocí odporového trimru se nastaví požadované napětí.

Problém s napájením vyřešil síťový zdroj, který napájí step-up modul. Síťový zdroj značky Vigan má výstupní parametry 5 V a 1 A s Jack konektorem.

2.4 SCHÉMA ZAPOJENÍ



Obrázek 2.4 – Schéma zapojení

2.4.1 Popis zapojení

Ventilátor, který slouží jako simulace poruchy a je připojen napájecím kabelem mezi kolektor tranzistoru – POR-VENT a zemí – GND-VENT-POR. Napájecí vodič je připojen na konektor – 12 V-VENT-POR a černý vodič je připojen na konektor – GND-VENT-POR. Kabel pro snímání otáček je připojen na konektor – VENT-OTAC-POR. K vodiči pro snímání otáček je připojen druhý digitální pin Arduina na konektor – OTACKY-POR. Dále je do uzlu zapojen pull-up rezistor, který je připojen na stejné napájení, jako je napájení ventilátoru, tedy 12 V. Tímto pull-up rezistorem disponují oba ventilátory.

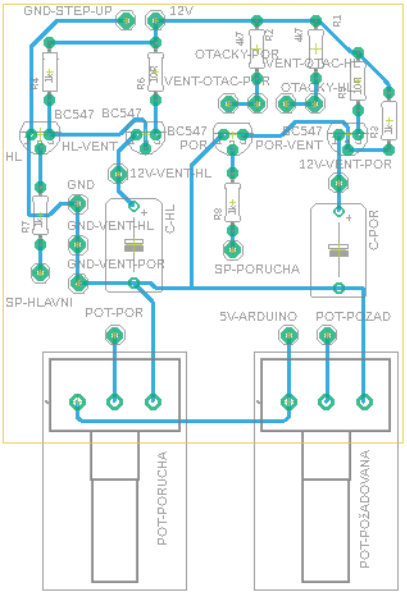
Ventilátor, který je regulován, je připojen napájecím kabelem mezi kolektor tranzistoru – 12 V-VENT-HL a zemí – GND-VENT-HL. Vodič pro snímání otáček je připojen na konektor – VENT-OTAC-HL. Třetí digitální pin Arduina je připojen na konektor – OTACKY-HL.

Oba ventilátory mají k jejich řízení dva spínací tranzistory – BC547. Pokud je první tranzistor zavřen, například tranzistor – HL, tak druhý tranzistor – HL-VENT je otevřen a ventilátor se otáčí. Pokud se první tranzistor otevře, tak druhý tranzistor je zavřený a ventilátor se netočí. To znamená, že je obrácená logika. Pokud je požadováno, aby se ventilátor točil, musí být PWM 0 %. Za požadavku, aby se ventilátor netočil, musí být PWM 100 %.

Pro ovládání ventilátoru pomocí PWM zde jsou dva potenciometry. Pro ovládání ventilátoru simulující poruchu je to potenciometr – POT-PORUCHA a pro regulovaný ventilátor je zde potenciometr – POT-POŽADOVANÁ, kterým se nastavuje požadovaná hodnota.

Pro snímání úbytku napětí na potenciometrech jsou využity vstupní analogové pin A0- pro POT-PORUCHA, pin A1 pro POT-POŽADOVANÁ.

Kondenzátory zapojené paralelně jsou zde pro udržení náboje, když je PWM ve stavu vypnuto.



Obrázek 2.5 – Schéma plošného spoje

Na obrázcích 2.5 a 2.4 lze vidět schéma plošného spoje a schéma zapojení. Schémata jsou vytvořena v programu Eagle a jsou součástí přílohy.

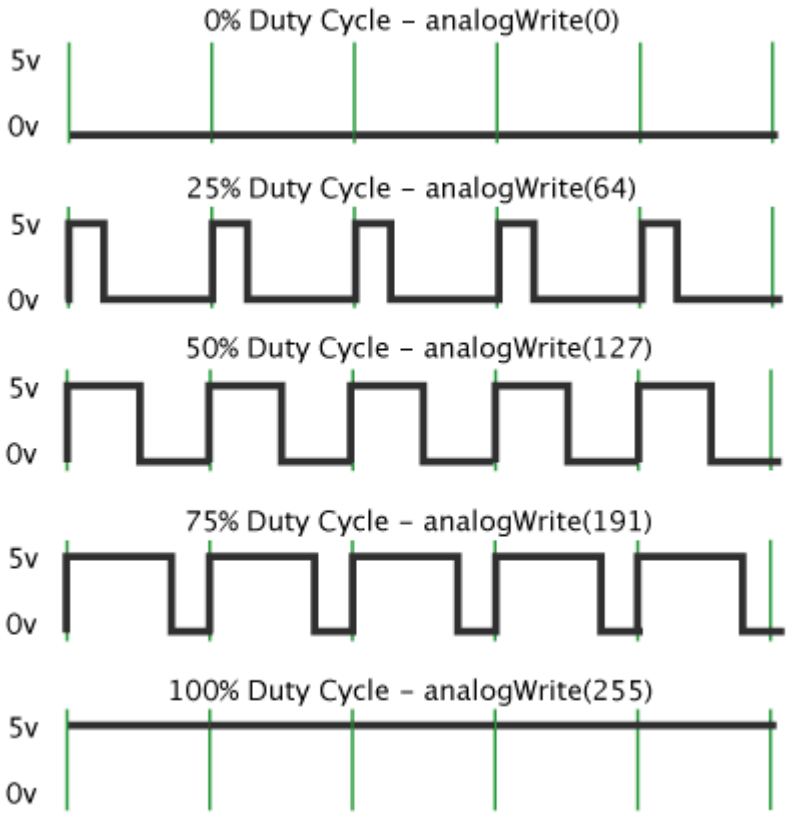
2.4.2 Seznam součástek

Tabulka 2.1 – Seznam všech součástek k realizaci zapojení

Součástka	Značka ve schématu	Hodnota/Typ	Pouzdro
Rezistor	R1	4k7	207
Rezistor	R2	4k7	207
Rezistor	R3	1k	207
Rezistor	R4	1k	207
Rezistor	R5	10R	207
Rezistor	R6	10R	207
Rezistor	R7	1k	207
Rezistor	R8	1k	207
Tranzistor	HL	BC547	TO92
Tranzistor	HL-VENT	BC547	TO92
Tranzistor	POR	BC547	TO92
Tranzistor	POR-VENT	BC547	TO92
Kondenzátor	C-HL	470uF	
Kondenzátor	C-POR	470uF	
Potenciometr	POT-PORUCHA	10k	Kov
Potenciometr	POT-POŽADOVANA	10k	Kov
Step – up modul	12 V	HW-637	-
Zdroj	-	Vigan VSZ-05-01	-
Ventilátor	VENT-HL	SUNAN	-
Ventilátor	VENT-POR	SUNAN	-

2.5 ŘÍZENÍ RYCHLOSTI OTÁČENÍ POMOCÍ PWM

PWM (pulzně šířková modulace) využívá vysílání obdélníkových pulzů o určité šířce a frekvenci. Jedná se o zapínání a vypínání napájení, poměr mezi zapnuto a vypnuto (šířka pulzu), neboli střída a udává se v procentech. U Arduina se využívají digitální piny, které podporují PWM (označeny vlnovkou). K sepnutí ventilátorů slouží funkce analogWrite(). v rozsahu PWM u Arduina pracuje od 0 do 255 (Hirzel, 2018).



Obrázek 2.6 – Arduino PWM (Hirzel, 2018)

2.6 PROGRAM PRO ARDUINO

2.6.1 Funkce setup

Na obrázku 2.7 lze vidět funkci „setup," která se provede pouze jednou. Prvně se nastaví rychlost komunikace tzv. „baudrate." Dále se nastaví piny „spousteni" a „hlavni" jako výstupy. Jedná se o piny, na kterých jsou připojeny ventilátory.

Ve funkci se nachází definování prvního přerušení „attachInterrupt," které slouží pro počítání pulzů otáček. Je třeba definovat, na jaký pin, podporující toto přerušení, je daný vývod ventilátoru pro snímání otáček připojen a také na jaký tvar signálu bude reagovat. V tomto případě na náběžnou hranu.

U druhého přerušení je využita knihovna, instalovaná v prostředí Arduino IDE. Nejdříve se definuje časovač a vloží hlavičkový soubor knihovny. Dále se inicializuje ITimer1, který byl povolen. Poslední krok spočívá v tom, že se definuje čas, po kterém dojde k přerušení. V tomto případě je to proměnná *T*.

2.6.2 Ovládání ventilátoru pomocí Arduino

```
#define USE_TIMER_1 true
#include "TimerInterrupt.h"
void setup()
{
  Serial.begin(2000000);
  pinMode(spousteni,OUTPUT);
  pinMode(hlavni,OUTPUT);

  attachInterrupt(digitalPinToInterrupt(2),otacky_por,RISING);
  attachInterrupt(digitalPinToInterrupt(3),otacky,RISING);
  ITimer1.init();
  ITimer1.attachInterruptInterval(T,Preruseni);

  analogWrite(spousteni,255);
  analogWrite(hlavni,255);
  delay(500);
}
```

Obrázek 2.7 – Funkce setup

Oba ventilátory jsou ovládány potenciometry, které jsou napájeny 5 V. Na potenciometru je změřena hodnota napětí pomocí instrukce „analogRead." Pokud přečteme hodnotu napětí na potenciometru, tak zabudovaný A/D převodník toto napětí převede na rozsah od 0 do 1023. Hodnota 1023 odpovídá napětí 5 V desky Arduina.

```
void porucha()
{
  por = analogRead(cteni_por);
  por = por/4;
  analogWrite(spousteni,por);
  }
```

Obrázek 2.8 – Kód pro ovládání ventilátoru simulující poruchu

Jak lze vidět na obrázku 2.8, změřená hodnota pro ventilátor, který simuluje poruchu, je vydělena čtyřmi, jelikož PWM pracuje v rozsahu od 0 do 255. Nakonec je tato hodnota zapsána na pin „spousteni.“

Nastavení požadovaných otáček pro regulovaný ventilátor je realizováno pomocí potenciometru. Měří se napětí na potenciometru a dále pomocí příkazu map je změněn výstupní rozsah A/D převodníku na rozsah otáček ventilátoru.

```
void pozadovane_otacky()
{
  poz = analogRead(chtena);
  poz_otac = map(poz,1023,0,0,6500);
  }
```

Obrázek 2.9 – Kód pro žádanou hodnotu

Nastavení požadovaných otáček pro regulovaný ventilátor je řešeno změřením napětí na potenciometru. Výstupní rozsah z A/D převodníku je změněn na rozsah otáček ventilátoru, jak lze vidět na obrázku 2.9. K změně rozsahu je použita funkce map.

2.6.3 Měření otáček ventilátorů a funkce přerušení

```
void Preruseni()
  {rpm = ((otac/2)*(60));rpm_hl = ((otac_hl/2)*(60));
  otac = 0;otac_hl = 0;regulace();}
void otacky_por(){otac = otac + 1;}
void otacky(){otac_hl = otac_hl + 1;}
```

Obrázek 2.10 – Měření otáček

Měření otáček je realizováno pomocí přerušení, které reaguje na vzestupnou hranu na žlutém vodiči ventilátoru. Při jedné otáčce dojde ke dvěma vzestupným hranám napětí. Při každé vzestupné hraně je program přerušen a přičítají se pulzy.

Funkce „přerušení“ se vykoná po každé sekundě a je zde pro vykonávání regulace a snímání otáček. Časový interval jedna sekunda je takto nastaven, protože k regulaci je potřeba dostatečná doba, aby byl napočítán dostatečný počet pulzů od ventilátoru, který je regulován. Pokud by doba měření otáček pro regulaci byla nižší, došlo by k nekvalitní a kmitající hodnotě otáček regulovaného ventilátoru.

2.6.4 Realizace číslicového regulátoru

```
void regulace()
 {
  reverse_poz = map(poz_otac,0,6500,6500,0); odchylka = reverse_poz - rpm_hl;
  akcni = Kp*odchylka + Ki*S;
 if (akcni <0){akcni = 0;}
 else if (akcni > 255){akcni = 255;}
 else
  {S = S + odchylka*T/1000;} akcni_hl = 255-akcni; analogWrite(hlavni,akcni_hl)
  }
```

Obrázek 2.11 – Přerušovací rutina regulátoru

Regulace pro ventilátor je řešena pomocí regulátoru PI. Nejdříve se v programu vypočítá regulační odchylka. To je rozdíl požadovaných otáček a změřených otáček. Násobením regulační odchylky pomocí vhodně zvolené konstanty – *K*p získáme proporcionální člen. Integrační člen se získá vhodně zvolenou konstantou – *K*i, která je násobena proměnnou – *S*. Proměnná – *S* představuje součet dvou regulačních odchylek a časovou konstantou – *T*, která představuje periodu měření otáček.

Podmínky na obrázku 2.11 jsou zde pro omezení akční veličiny, aby nevznikal tzv. wind-up efekt. Nakonec se akční veličina upraví na hodnotu v rozsahu Od 255 do 0 a zapíše na

pin, kde je připojen regulovaný ventilátor. Je to potřeba z toho důvodu, jelikož zapojením je obrácena logika pro ovládání ventilátoru. Tedy hodnota 255 by za normální situace ventilátor roztočila na maximální otáčky, ale v tomhle případě ho touto hodnotou zastaví.

2.7 PROGRAM V MATLABU

2.7.1 Komunikace mezi Arduinem a MATLABem

Komunikace mezi Matlabem a Arduinem je řešena tak, že při poslání dat z Arduina pomocí příkazu Serial.println() na sériovou linku se v Matlabu vyvolá tzv. zpětná funkce (při každém zápisu na port), která přečte pomocí příkazu fscanf() sériový port.

2.7.2 Skript pro nastavení

Ve skriptu nastavení je potřeba si nastavit proměnou – arduino. Tato proměnná představuje sériovou komunikaci, je potřeba provést určité nastavení. Musí se určit příslušný port, ke kterému je deska připojena. Dále je potřeba nastavit tzv. baudrate. Musí se nastavit stejný, jako je v programu Arduino IDE, aby komunikace fungovala.

```
arduino = serial('COM7','BaudRate',2000000);
arduino.BytesAvailableFcn={@zobrazeni};

cas = zeros(1,50);
cas(51) = 1;

cas2 = zeros(1,50);
cas2(51) = 1;

cas3 = zeros(1,50);
cas3(51) = 1;
fopen(arduino)
```

Obrázek 2.12 – Skript pro nastavení

Proměnné – cas, cas2, cas3 na obrázku 2.12 jsou zde jako matice o jednom řádku a padesáti hodnotách, které jsou připravené na posuv dat v grafu. Důvodem přidání dalšího prvku do matice na obrázku 2.11 a vymazání prvního prvku na obrázku 2.12 je, aby se data nahrávala a posouvala z pravé strany k levé.

2.7.3 Skript pro zobrazení

```
function zobrazeni(arduino,~)
global cas; global cas2;global cas3;
%%Čtení ze seriového portu a převedení dat na číslo
poz1 = fscanf(arduino);poz=str2num(poz1);
rpm1 = fscanf(arduino);rpm = str2num(rpm1);
rpm_hl1= fscanf(arduino);rpm_hl = str2num(rpm_hl1);
%%Zápis přečtených dat
cas(51) = poz;cas(1)  = [];
cas2(51) = rpm;cas2(1)  = [];
cas3(51) = rpm_hl;cas3(1)  = [];
%%Vytvoření zobrazovacího prostředí
figure(1)
plot(cas,'b','LineWidth',1.5);
hold on
plot(cas2,'g','LineWidth',1.5);
hold on
plot(cas3,'r','LineWidth',1.5);
legend('Požadované ot','Porucha','Regulace','Location','northwestoutside');
ylabel('Otáčky');
grid on
hold off
end
```

Obrázek 2.13 – Kód pro zobrazení

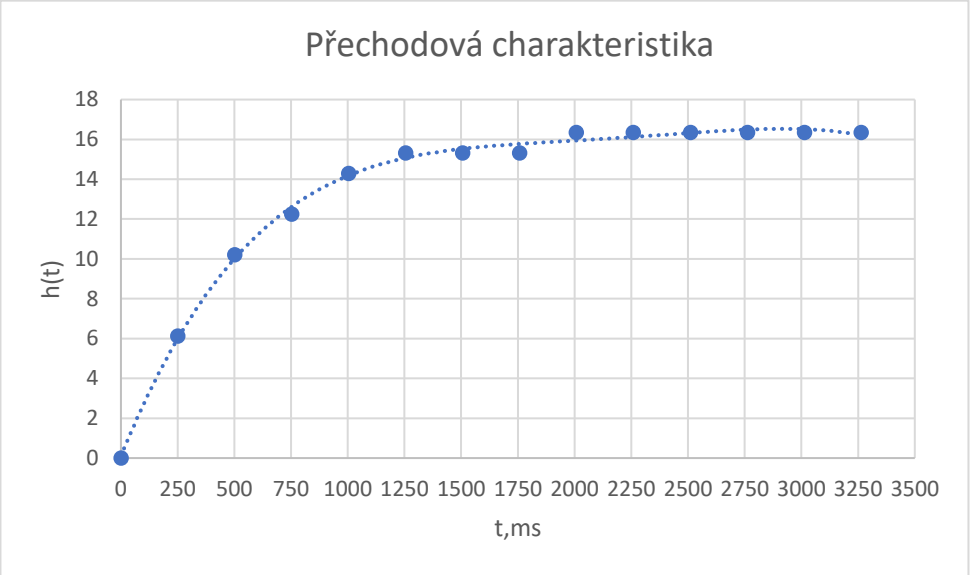
Ve skriptu zobrazení na obrázku 2.13 se načtou data poslaná z Arduina, která představují požadované otáčky pro regulovaný ventilátor, otáčky ventilátoru simulující poruchu a otáčky regulovaného ventilátoru. Tato data se převedou na čísla. Nakonec se hodnoty zobrazí do grafu, kde data jsou zobrazována v závislosti na zobrazovací periodě v programu Arduino. Zobrazení je prováděno každou půl sekundu.

Pro zobrazení se v první fázi nahraje program na desku Arduina. Druhým krokem je potřeba spustit skript – nastavení, to se provede tím způsobem, že do příkazového řádku se napíše jeho název. Druhý skript – zobrazení není nutnost spouštět, tak jako první skript. V prvním skriptu, který je na obrázku 2.12, se na konci kódu nachází instrukce fopen(Arduino), která spustí druhý skript a ten se nachází na obrázku 2.13. A poté se otevře graf s průběhem otáček.

3 EXPERIMENTÁLNÍ ČÁST

3.1 PŘECHODOVÁ CHARAKTERISTIKA VENTILÁTORU

Prvně se změřila přechodová charakteristika. Ta byla měřena pomocí softwaru Arduino. Z ustálené hodnoty otáček 2400, což odpovídalo hodnotě 20 (8 % střídy signálu). Z této hodnoty přišel skok na 255, což odpovídá střídě 100 %. Jednotlivá data jsou vzorkována každých 250 milisekund. Bylo nutné zvolit kratší periodu vzorkování z důvodu délky přechodového děje. Ustálená hodnota otáček je 6240 a přechodová charakteristika je vypočítána ze získaného průběhu otáček pomocí vztahu (1.2).



Obrázek 3.1 – Přechodová charakteristika ventilátoru

Výpočet pro statické zesílení je

$$k = \frac{6240 - 2400}{255 - 20} = 16{,}34, \tag{3.1}$$

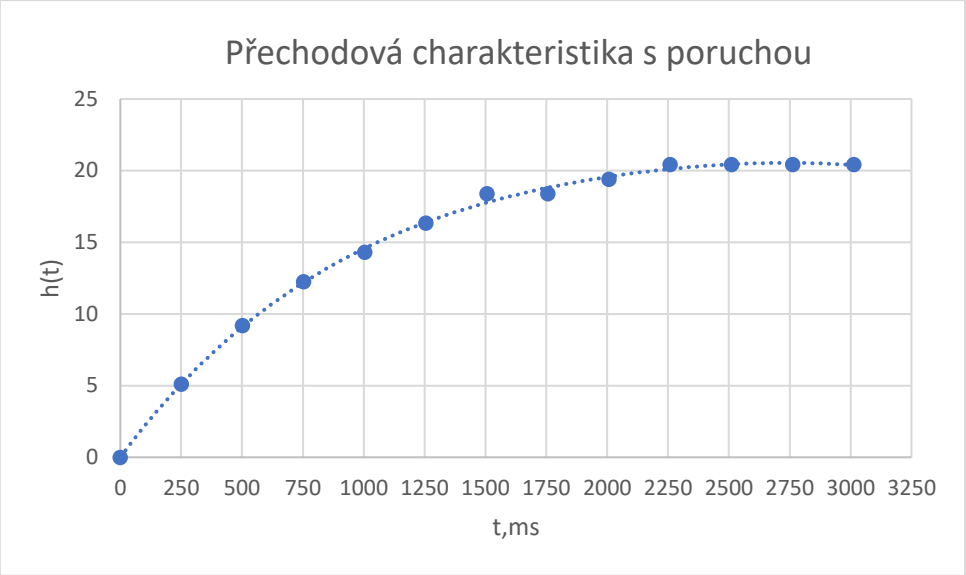
kde k je statické zesílení.

Přenos soustavy je

$$F(s) = \frac{k}{Ts + 1} = \frac{16{,}34}{0{,}5s + 1}, \tag{3.2}$$

kde $F(s)$ je obrazový přenos,

T je půl sekundy a je to doba, kdy má přechodová funkce hodnotu 0,63k (63 % své ustálené hodnoty).



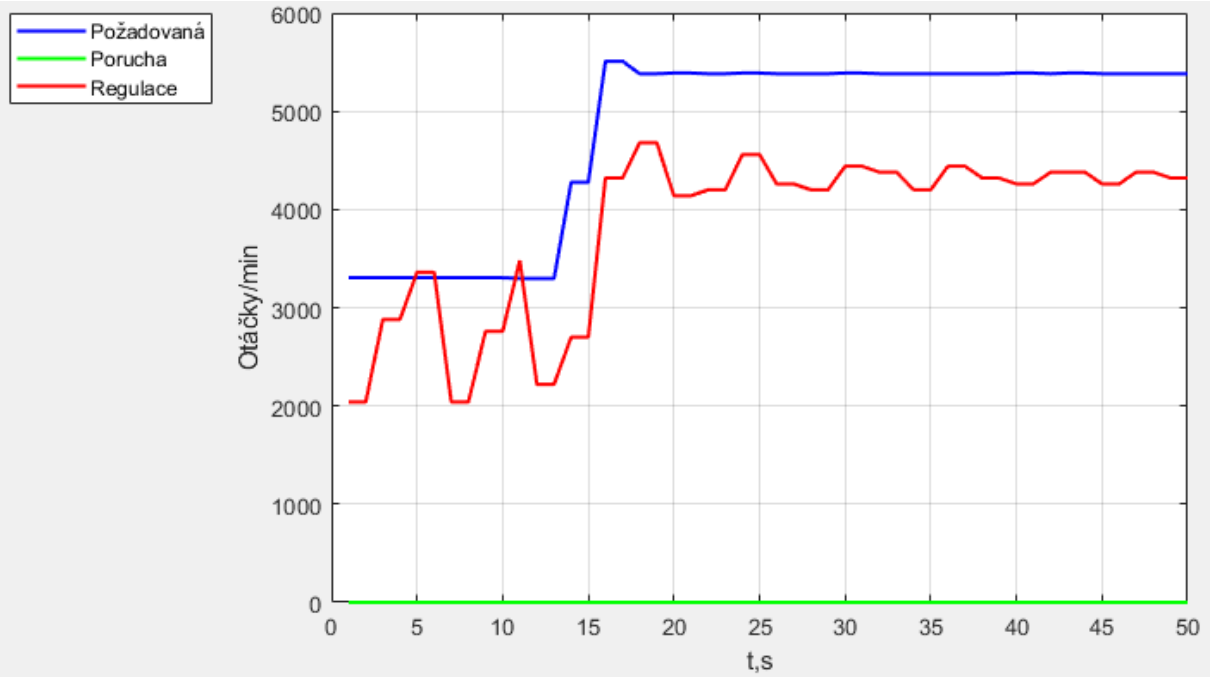
Obrázek 3.2 – Přechodová charakteristika s poruchou na 50 % max. otáček

Přechodová charakteristika na obrázku 3.2 je provedena stejně jako přechodová charakteristika na obrázku 3.1, ale s tím rozdílem, že na ventilátor působí i druhý ventilátor. Ten je roztočen na konstantní otáčky a ty odpovídají 50 % rychlosti rozsahu otáček.

Při porovnání přechodových charakteristik s poruchou a bez poruchy lze vypozorovat, že poruchový ventilátor výrazně brání druhému ventilátoru se rozběhnout do plných otáček. Ventilátor realizující poruchu také snižuje maximální otáčky ventilátoru, na kterém byla měřena přechodová charakteristika.

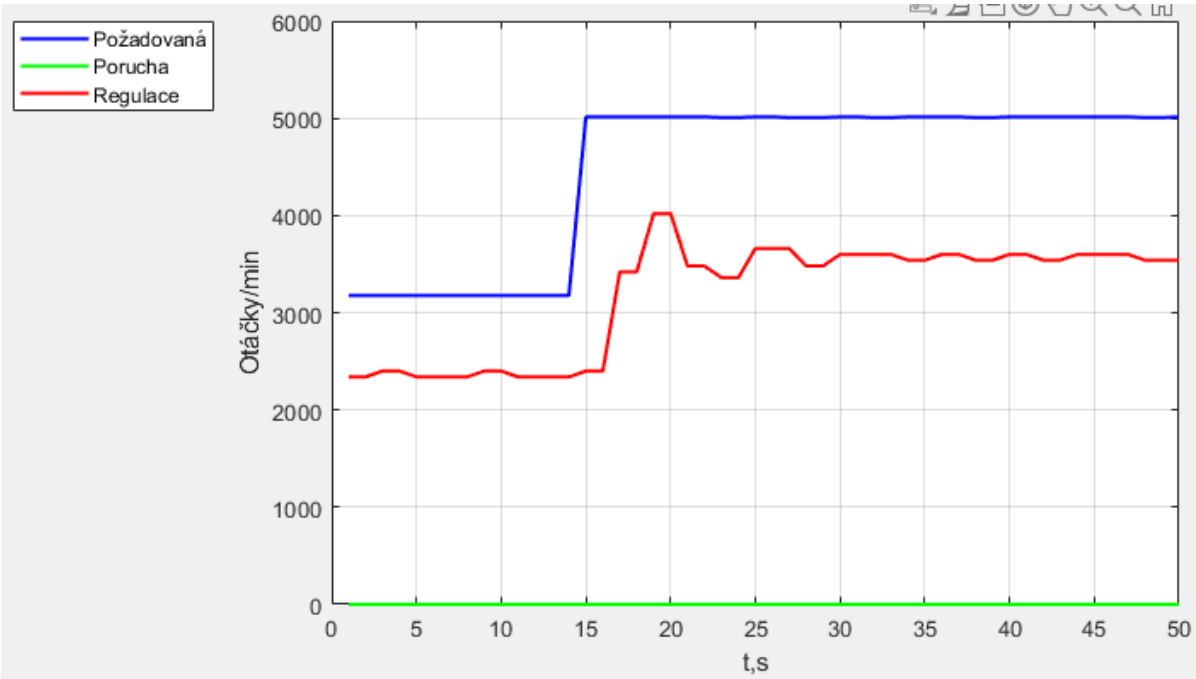
3.2 REGULACE OTÁČEK – REGULÁTOR P

3.2.1 Reakce na změnu požadovaných otáček při různém nastavení



Obrázek 3.3 – Reakce na změnu žádané hodnoty při prvním nastavení

Na obrázku 3.3 lze vidět regulační pochod, kde parametr r_0 je nastaven jako převrácená hodnota statického zesílení. Z průběhu lze vypozorovat, že při tomto nastavení je průběh otáček (červený průběh) velice kmitavý, zvlášť v nižších otáčkách. Průběh požadované hodnoty je modrou barvou a průběh poruchového ventilátoru barvou zelenou. Vzorkovací perioda regulátoru je jedna sekunda.

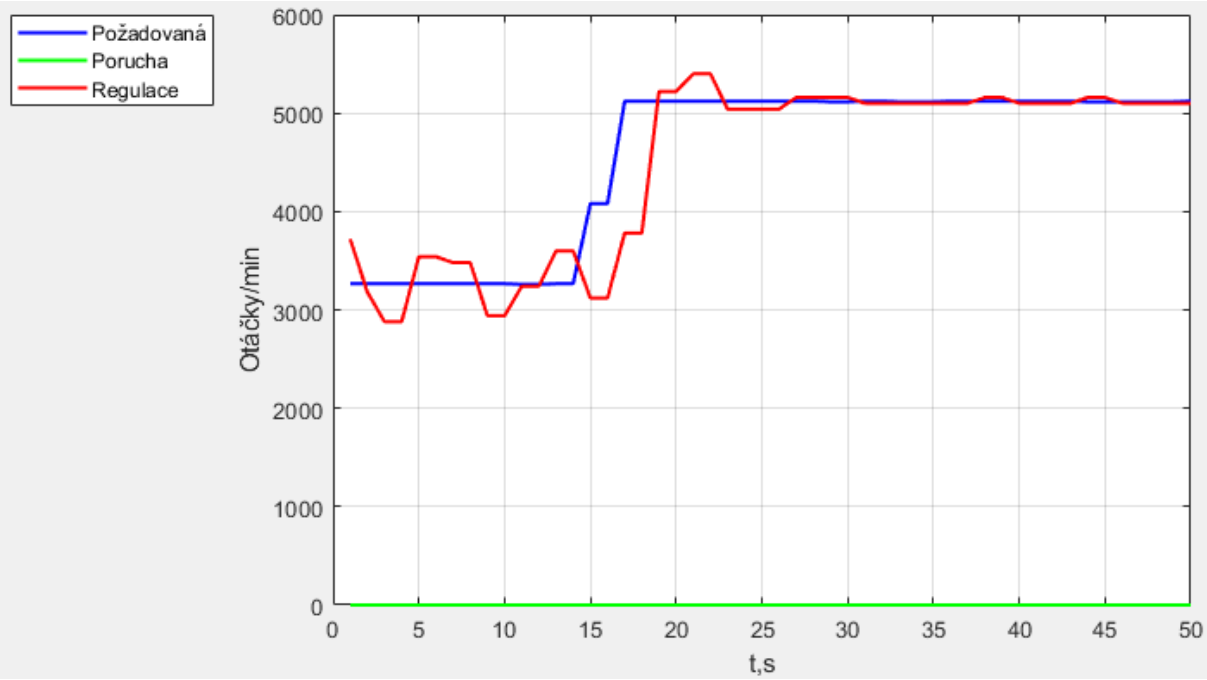


Obrázek 3.4 – Reakce na změnu žádané hodnoty při druhém nastavení

V případě druhého nastavení, kde je parametr r_0 pouze vydělen číslem dva, je sice kmitání otáček zmenšeno, ale regulační odchylka je oproti regulačnímu pochodu na obrázku 8.4 větší ve vyšších otáčkách.

3.3 REGULACE OTÁČEK – REGULÁTOR PI

3.3.1 Reakce na změnu požadovaných otáček při různém nastavení



Obrázek 3.5 – První nastavení regulátoru

V první fázi byly nastaveny parametry regulátoru takto

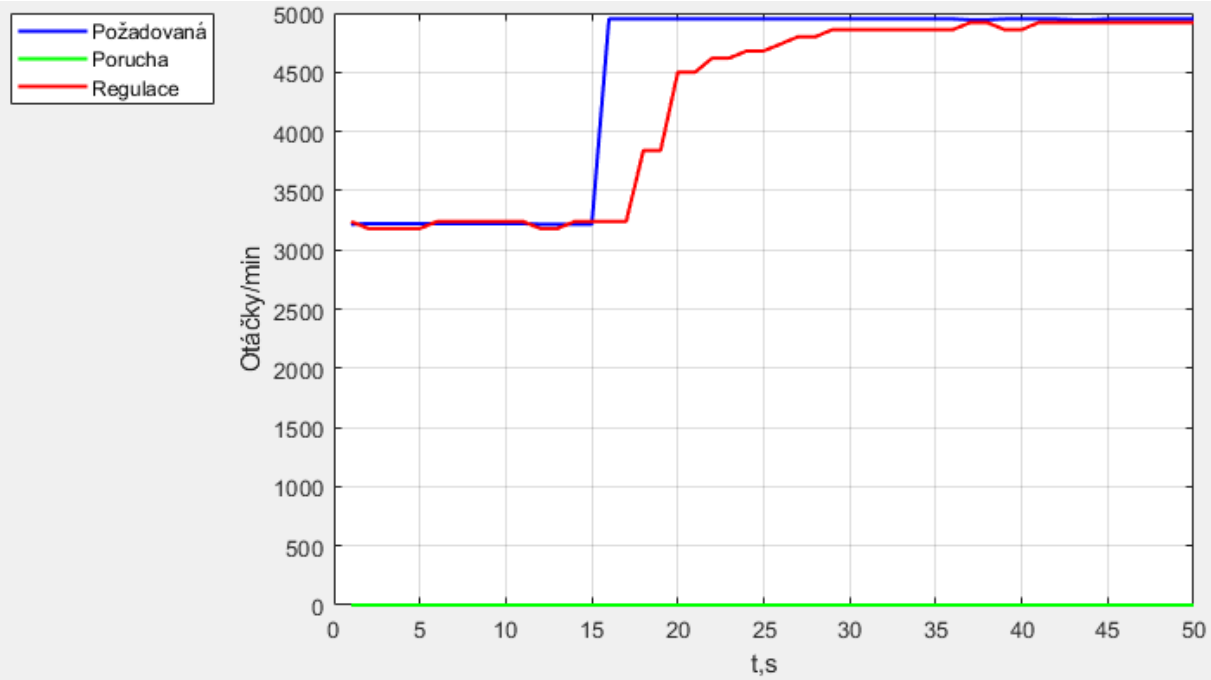
$$r_0 = \frac{1}{k}, r_1 = \frac{r_0}{2}, \tag{3.3}$$

kde r_0 je zesílení proporcionálního členu,

k je statické zesílení,

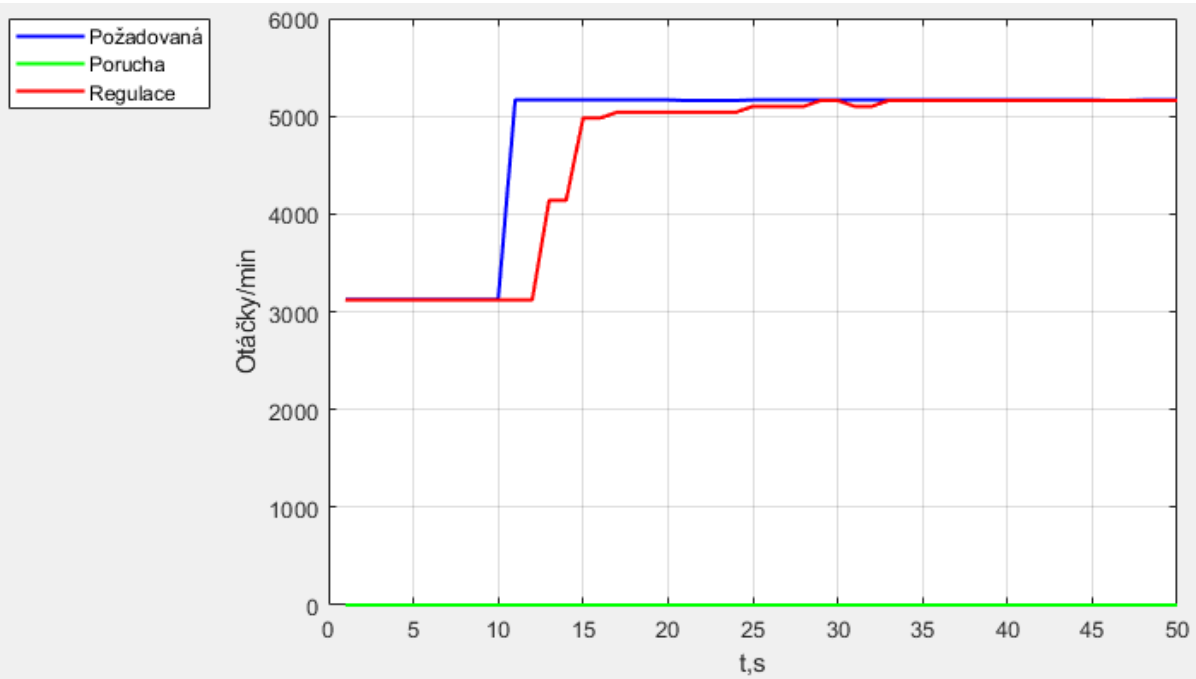
r_1 je zesílení integračního členu.

Z obrázku 3.5 je vidět, že regulace otáček s tímto nastavením je v nižších otáčkách relativně kmitavá, ale po přidání integrační složky se odstranila regulační odchylka. Kmitání si lze povšimnout před změnou požadovaných otáček. Tím pádem je potřeba snížit zesílení.



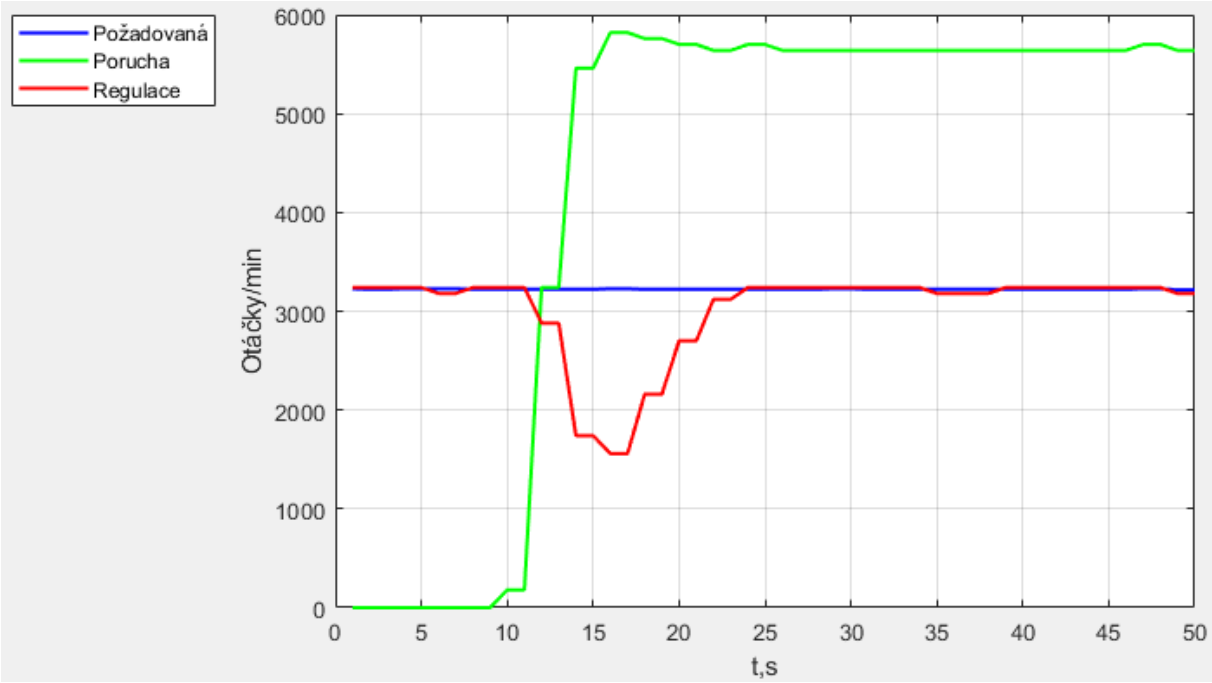
Obrázek 3.6 – Regulace při zmenšení zesílení

Na obrázku 3.6 je oproti průběhu na obrázku 3.5 zesílení vyděleno třemi a zesílení integračního členu zůstává stejné. Regulace je bez mírného kmitání kolem žádané hodnoty, ale doba regulace je příliš dlouhá.



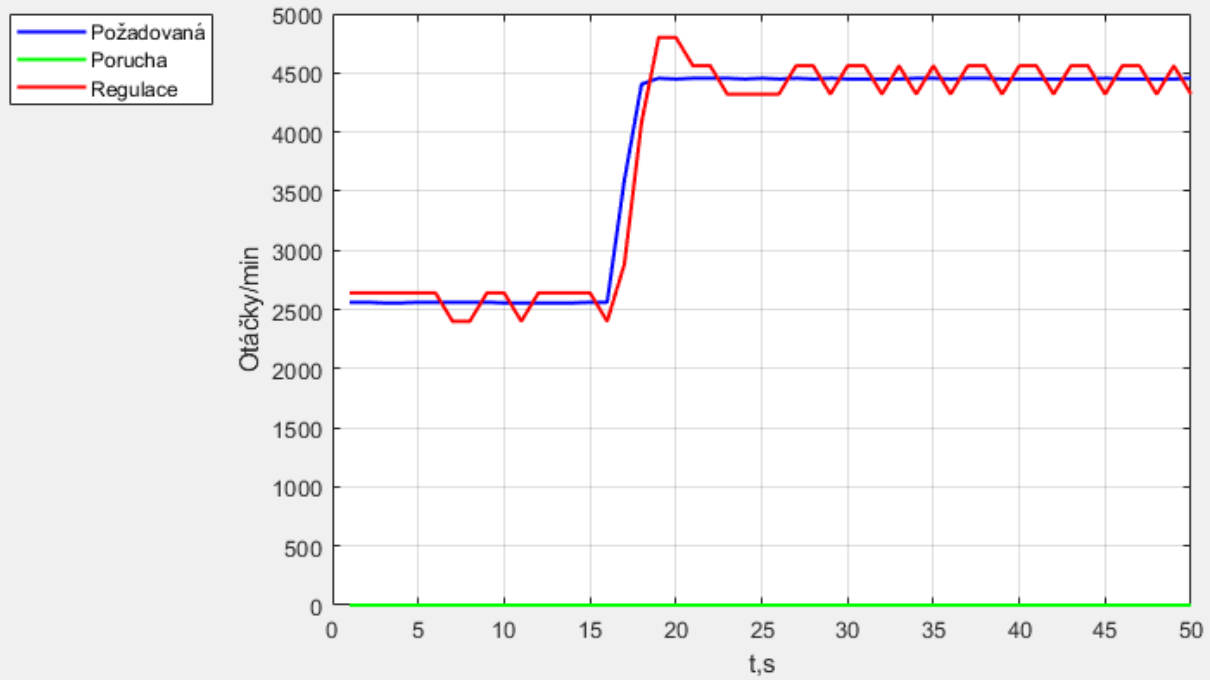
Obrázek 3.7 – Finální nastavení PI regulátoru

V posledním nastavení je zvoleno zesílení $k_0 = \frac{0,5}{k}$. Což oproti průběhu na obrázku 3.6 zmenšilo regulační dobu a bylo zvoleno jako finální nastavení regulátoru PI. Odpovídající průběhy jako na obrázku 3.7.



Obrázek 3.8 – Reakce na poruchovou veličinu

Na obrázku 3.8, při ustálené hodnotě regulovaného ventilátoru, je ventilátor realizující poruchu spuštěn na maximální možné otáčky. Lze vypozorovat, jakým způsobem se ventilátory ovlivňují. Regulovaný ventilátor musí vykompenzovat působení protiproudu vzduchu a také možno vypozorovat, že regulovaný ventilátor působí jako porucha na druhý ventilátor. Za normálních podmínek by se ventilátor, který je spuštěn na maximální chod, měl otáčet v rozsahu od 6000 do 6500 otáček za minutu. Také je možnost si povšimnout, že při reakci regulátoru se poruchový ventilátor mírně zpomalí. Regulátorem s integrační složkou se podařilo dosáhnout plné kompenzace poruchy.



Obrázek 3.9 – Reakce na změnu žádané hodnoty při periodě vzorkování 250ms

Na obrázku 3.9 lze vidět průběh regulace, který má stejnou vzorkovací periodu jako přechodová charakteristika na obrázku 3.1. Volba periody 250 milisekund zapříčinila méně přesné měření otáček, tím regulované otáčky kmitají kolem žádané hodnoty. Parametry regulátoru jsou stejné jako na obrázku 3.7. Při porovnání průběhů na obrázku 3.7 a na obrázku 3.9, je vidět, že při volbě vzorkovací periody jedna sekunda je průběh regulace kvalitnější.

4 ZÁVĚR

Bylo vytvořeno zařízení, skládající se ze dvou ventilátorů, kde otáčky prvního ventilátoru jsou regulovány na požadovanou hodnotu, a druhý ventilátor slouží jako zdroj poruchy.

Podařilo se naprogramovat ovládání ventilátoru realizující poruchu a regulaci ventilátoru pomocí regulátoru PI. Dále se podařilo naprogramovat zobrazení průběhů otáček obou ventilátorů a nastavitelnou hodnotu otáček pro regulovaný ventilátor. Ovládání obou ventilátorů je realizováno pomocí potenciometrů. Pro realizaci programu byla využita deska Arduino Uno a pro zobrazení otáček a průběhu regulace byl využit software MATLAB.

V rámci řešení práce byl sestaven jednoduchý obvod pro ovládání ventilátorů s příslušným napájením. Pro konstrukci zařízení bylo využito 3D tisku.

V práci je demonstrován rozdíl mezi regulátorem P a PI, kde P regulátor má trvalou regulační odchylku a regulátorem PI se tato regulační odchylka natrvalo eliminuje. Dále se ukázalo, že ventilátory připojené k sobě se ovlivňují navzájem, a to i v případě, když je jeden z ventilátorů v nečinnosti a druhý je nastaven na maximální výkon. Dokonce v tomto případě může nastat situace, že aktivní ventilátor může nečinný ventilátor roztočit.

POUŽITÁ LITERATURA

CVEJN, Jan. 2012. *Řízení procesů*. [online]. [Cit. 14.4.2022]. Dostupné z: http://matlab.fei.tuke.sk/ons/doc/Cvejn_rizeni_procesu.pdf

HERRES, David. 2015. *Basic of Hall effect sending.* [online]. [Cit. 3.5.2022]. Dostupné z: https://www.testandmeasurementtips.com/basics-of-hall-effect-sensing/

HIRZEL, Timothy. 2018. *PWM.* [online]. [Cit. 12.5.2022]. Dostupné z: https://www.arduino.cc/en/Tutorial/Foundations/PWM

VODA, Zbyšek. 2017. *Průvodce světem Arduina*. Vydání druhé. Bučovice: Martin Stříž, ISBN 978-80-87106-93-8.

PŘÍLOHY

A – CD

Příloha A

Příloha k bakalářské práci

SOUSTAVA DVOU VENTILÁTORŮ S REGULACÍ OTÁČEK

Michal Pulkráb

CD

OBSAH