TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Řízení pneumatických ventilů pomocí vývojového kitu Arduino

Bakalářská práce

Studijní program: B2301 – Strojní inženýrství
Studijní obor: 2301R000 – Strojní inženýrství

Autor práce: **Petr Marousek**
Vedoucí práce: Ing. Radek Votrubec, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechanical Engineering

Control of pneumatical valves using Arduino board

Bachelor thesis

Study programme:	B2301 – Mechanical Engineering
Study branch:	2301R000 – Mechanical Engineering
Author:	**Petr Marousek**
Supervisor:	Ing. Radek Votrubec, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr Marousek**
Osobní číslo:	**S13000127**
Studijní program:	**B2301 Strojní inženýrství**
Studijní obor:	**Strojní inženýrství**
Název tématu:	**Řízení pneumatických ventilů pomocí vývojového kitu Arduino**
Zadávající katedra:	**Katedra výrobních systémů a automatizace**

Zásady pro vypracování:

1. Seznamte se s parametry použitých pneumatických ventilů.
2. Seznamte se s vývojovým prostředím a programováním vývojového kitu Arduino.
3. Osvojte si práci se vstupy a výstupy.
3. Realizujte připojení ventilů a naprogramujte jejich ovládání.

Rozsah grafických prací: **podle potřeby**

Rozsah pracovní zprávy: **35 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] Snail Instruments. *Začínáme s Arduinem.* **www.hobbyrobot.cz [online]. 2014 [cit. 2015-01-09]. Dostupné z: http://www.snailshop.cz/literatura/1537-zaciname-s-arduinem-prirucka.html**

[2] Arduino. *Arduino Learning: Getting Started with Arduino.* **[online]. 2014 [cit. 2015-01-09]. Dostupné z: http://arduino.cc/en/Guide/HomePage**

[3] BALÁTĚ, J. *Automatické řízení.* **1. vyd. Praha: BEN - technická literatura, 2003. ISBN 978-80-247-4116-1.**

[4] OLEHLA, M., S. NĚMEČEK. *Automatické řízení.* **vyd. 1. Liberec: Technická univerzita v Liberci, 2013. ISBN 978-807-3729-721.**

Vedoucí bakalářské práce: **Ing. Radek Votrubec, Ph.D.**
Katedra výrobních systémů a automatizace

Datum zadání bakalářské práce: **15. listopadu 2015**

Termín odevzdání bakalářské práce: **15. února 2017**

prof. Dr. Ing. Petr Lenfeld
děkan

Technická univerzita v Liberci
L.S.
7a

Ing. Petr Zelený, Ph.D.
vedoucí katedry

V Liberci dne 15. listopadu 2015

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Radku Votrubcovi, Ph.D. za odborné vedení a profesionální přístup při vypracovávání bakalářské práce. Děkuji za ochotu, čas strávený při konzultacích a cenné připomínky.

Anotace

Bakalářská práce se zabývá realizací vývojového kitu Arduino v oblasti ovládání a řízení pneumatických ventilů. Jsou zde popsány vlastnosti Arduina, jeho programovací prostředí a možnosti rozšíření jeho funkcí. Dále jsou popsány parametry pneumatických ventilů a vybrané způsoby spojité a nespojité regulace. Spojitá regulace je zaměřena na tři typy: proporcionální, integrační a proporcionálně-integrační. Nespojitá regulace je zaměřena pouze na dvoupolohovou regulaci. V praktické části práce je popsán postup realizace vybraných regulátorů. V závěru je regulátor vytvořený pomocí Arduina aplikován do autosedačky, kde se ovládá tuhost sedáku zvyšováním a snižováním tlaku vzduchu.

Klíčová slova

Vývojový kit Arduino, programování, pneumatické ventily, regulace, řízení, autosedačka

Annotation

The aim of this bachelor thesis is realization of Arduino board and its application in controlling pneumatic valves. There is description of Arduin, his programming environment and extensibility of its functions. There is also description pneumatic valves parameters and selected methods of continuous and discontinuous regulation. The continuous regulation is aim at three types: proportional, integral and proportional-integral. The discontinuous regulation is aim at only two-position regulation. In practical part of this thesis, there is description of selected regulators implementation. In the last part, the controller, made by Arduin, is implicated to car seat, where it controls seat stiffness by increasing and decreasing air pressure.

Key words

Arduino board, programming, pneumatic valves, regulation, control, the Car Seat

Obsah

Seznam použitých zkratek

PWM	Pulse Width Modulation (pulsně šířková modulace)
IDE	Integrated Development Enviroment (integrované vývojové prostředí)
USB	Universal Serial Bus (univerzální sériová sběrnice)
LED	Light Emitting Diode (světlo emitující dioda)
TTL	transistor-transistor-logic (tranzistorově-tranzistorová logika)
CE	Conformité Européenne (Evropské shoda)
Wi-Fi	Wireless Fidelity (bezdrátové připojení)
LCD	Liquid Crystal Display
SD	Secure Digital
log	logický

1. Úvod

Pneumatické systémy jsou dnes běžnou součástí některých zařízení kolem nás. Zajišťují správnou funkčnost zařízení a mimo jiného i naše pohodlí a komfort. To se zejména týká pneumatických systémů zabudovaných v sedadlech automobilů. Bočnice sedáku a bederní opěry sedadla jsou vyplněny vzduchovými polštářky, ve kterých se dá nastavovat tlak vzduchu. Tím si uživatel do jisté míry přizpůsobí tvar sedadla pro větší pohodlí a zároveň uchopení těla může navozovat dojem sportovního sedadla. Ovšem tato technologie je značně omezená pouze na držení a oporu těla v sedadle. Nabízí se tedy myšlenka rozšířit možnosti nastavování sedadla tak, aby jízda automobilem byla daleko více komfortní. Například nastavení tuhosti celého sedáku sedačky by nabízelo o mnoho více. Pro realizaci takovéto myšlenky je důležité mít představu o řídicím systému, který by dokázal regulovat tuhost sedáku.

Vývojový kit Arduino má dobré předpoklady pro realizaci této úlohy. Díky jeho programovatelnosti může sloužit jako řídící jednotka. Má reálné vstupy a výstupy, tudíž se s ním dají řídit pneumatické ventily.

Tento systém má mnohé další výhody oproti systému, který je dnes do sedaček běžně montován. Zatím co dnes běžně používané pneumatické systémy v sedadlech jsou ovládány dvěma tlačítky pro zvyšování a snižování tlaku v bočnicích a bederních opěrkách sedadla, tak systém s Arduinem umožňuje plynulou regulaci tuhosti sedáku, měnit tuhost podle povrchu vozovky, nebo intervalově měnit tuhost, což může působit proti únavě řidiče.

Soustava sestavená z pneumatických ventilů a Arduina má nízké nároky na údržbu, vysokou spolehlivost, malé rozměry a nízkou cenu.

1.1 Cíl práce

Cílem této práce je seznámit se s parametry použitých pneumatických ventilů, s možnostmi samotného Arduina a ovládáním jeho vstupů a výstupů. Dále vytvoření řízení pneumatických ventilů pomocí vývojového kitu Arduino. Do toho spadá sestavení pneumatického a elektrického obvodu a vytvoření programu pro nespojitou (dvoupolohovou) a spojitou (P, I, PI) regulaci.

Na závěr aplikovat celé zapojení do spodní části autosedačky, kde se bude ovládat tuhost sedáku nafukováním pryžových vaků.

2. Arduino

2.1 Základní popis desek Arduino

Arduino je deska plošných spojů, která není nijak zakrytovaná a na které jsou systematicky připájeny elektronické součástky, vstupy a výstupy. To dohromady tvoří hardware. K ovládání a propojení s počítačem je nutné mít i patřičný software, který je od výrobce volně dostupný. Hardware i software dohromady tvoří kompletní platformu.

Existuje velké množství typů desek Arduino. Mohou se rozlišovat z mnoha hledisek, např. od méně výkonných, ale zato rozměrově malých, až po velmi výkonné. Dále podle počtu vstupů a výstupů, podle způsobu napájení a podobně. Jednotlivé typy desek mají mezi sebou hodně společného, ale jsou aplikace, kde se dá použít pouze konkrétní typ.

Mezi hlavní elektronické komponenty, které se nacházejí na všech typech desek Arduino, patří mikroprocesor od firmy Atmel. Mikroprocesor řídí činnost celého systému, na základě instrukcí od uživatele. Dále na desce nalezneme napájecí konektor, který může být samostatně, nebo zde nemusí být vůbec a napájení je realizováno přes USB port. Hned za napájením se řadí regulátor napětí. Ten reguluje napětí a hlídá, aby se na desku nedostalo větší napětí, které by mohlo poškodit některé součástky. Arduino disponuje mnoha piny (kolíky), které slouží k připojení dalších obvodů nebo součástí. Jsou navrženy tak, že k nim lze snadno připojit vodič s příslušným zakončením. Každý pin má své označení, svou funkci a jejich počet se na jednotlivých typech desek liší. Pin označený *GND* je zemnící kolík, těchto pinů bývá zpravidla více a jsou rozmístěny různě po obvodu desky. Piny s označením *5 V* a *3.3 V* slouží k napájení obvodu připojeného k těmto pinům, přičemž toto napájení je trvalé, a proto se zpravidla s těmito piny v kódu programu nemanipuluje. Napětí těchto pinů je, jak už označení napovídá, 5 V a 3,3 V. Kdybychom chtěli ovládat obvody, kde potřebujeme vyšší napájecí napětím než je 5 V, musíme použít externí zdroj napětí. Analogové vstupy se označují velkým písmenem A a číslem vstupu, např. *A2*. Tyto piny slouží pro čtení analogového signálu, například z tlakového snímače. Analogový signál je dále převáděn do digitální podoby. Pro digitální signál slouží piny, které jsou označeny buď pouze číslicí, která označuje konkrétní pin, nebo velkým písmenem D a číslem pinu, např. *D2* nebo *2*. Tyto piny jsou zároveň vstupy i výstupy, záleží na tom, jak jsou v programu nadefinovány.

Většina typů desek Arduino neumí generovat analogový signál. Na výstupu je hodnota napětí pouze 5 V, nebo 0 V. Absence analogových výstupů je částečně nahrazena digitálními piny, které jsou označeny zkratkou *PWM* (Pulse Width Modulation), nebo vlnovkou (~). Takto označené piny, pokud jsou správně nadefinovány v programu, v praxi rychle střídají napětí 0 a 5 V. Takže uživatel si pouze nastavuje rychlost střídání 0 a 5 V a tím lze některé elektronické součástky připojené na tyto piny ovládat podobně, jako analogovým signálem. Součástí každé desky je tlačítko RESET. Po stisknutí se kód, který je v Arduinu nahrán, spustí od začátku. To je výhodné, pokud kód nemá nastavené opakování. RESET lze ovládat i pomocí stejnojmenného pinu. Na desce také nalezneme indikační LED diody. Diodu označenou *ON*, která signalizuje, že Arduino je připojeno na napájecí zdroj. Dále diody označené *TX* a *RX*. Diody označené *TX* a *RX* signalizují sériovou komunikaci. *TX* signalizuje odesílání dat z Arduina a *RX* přijímání dat. Stejnojmenně jako diody, jsou popsány i piny, které se dají použít pro sériovou komunikaci mezi Arduinem a dalším zařízením. Popsané komponenty jsou zobrazeny na Obr. 1.

Komunikace Arduina s počítačem je realizována přes rozhraní USB. Mikroprocesory desek však pracují se sériovým TTL rozhraním, proto pro komunikaci musí být bezprostředně použit USB - serial převodník. Určité typy desek tento převodník nemají. Na jejich základně úmyslně není zabudován. Tyto desky jsou navrženy pro řízení hotových a odzkoušených projektů. Používají se především bez trvalého připojení k počítači. Důležitou roli zde hraje také jejich velikost, je kladen důraz na malé rozměry. Z těchto důvodu na nich nenajdeme USB vstup. Pro nahrávání programů se používá externí převodník, který je mimo jiné připojen na piny označené TX a RX sloužící pro sériovou komunikaci.

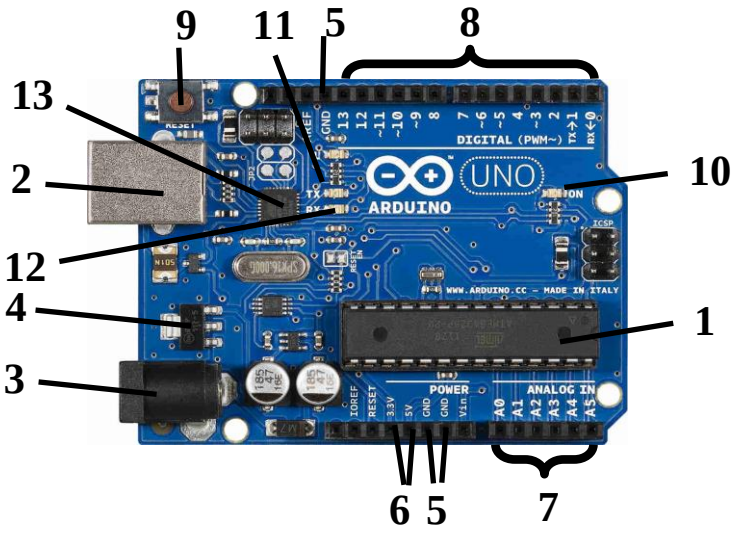
Výhody desek Arduino:

- široká možnost použití – desky Arduino je možno použít napříč všemi obory, kde se uplatňuje řízení, měření, kontrolování, snímání, ovládání a podobné úlohy
- cenová dostupnost – desky jsou relativně cenově přijatelné
- jednoduché a přehledné programovací prostředí
- univerzálnost – deska lze individuálně sestavit z komponentů, které si vyžaduje daná úloha, z toho vyplívá, že uživatel není omezen pouze oficiálními typy desek Arduino

- napájení malým napětím – většina desek obsahuje napěťově nenáročné součástky a regulátor napětí, který zajišťuje, že napájecí napětí může být v rozsahu 5 až 12 V, díky tomuto aspektu může být deska napájena i přes USB port

Nevýhody:

- absence analogového výstupu – Arduina většinou nemají analogový výstup, při potřebě analogového výstupu se používají přídavná zařízení, která dokáží generovat analogový signál
- Arduino nemá certifikaci CE (Conformité Européenne) – nemůže být použito pro projekty, které vyžadují tuto certifikaci (např. některé aplikace ve zdravotnictví)



Obr. 1 *Komponenty na desce Arduino UNO*

1- mikroprocesor Atmel, 2- USB post, 3- napájecí konektor, 4- regulátor napětí, 5- zemnící piny, 6- napájecí piny, 7- analogové vstupy, 8- digitální vstupy a výstupy, 9- tlačítko RESET, 10- LED dioda ON, 11- LED dioda TX, 12- LED dioda RX, 13 – USB-seriál převodník

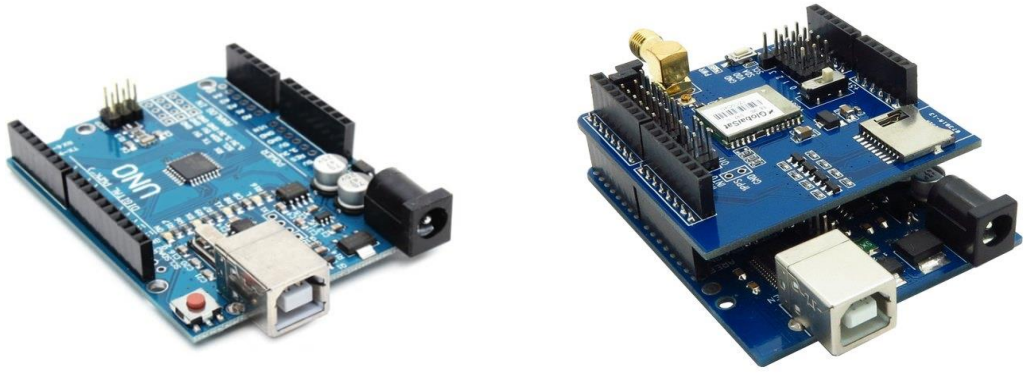
2.2 Arduino Shield

Shield (štít) je deska, která dále rozšiřuje možnosti použití Arduina. Pro některé úkoly není Arduino hardwarově vybaveno, např.: komunikace pomocí bluetooth, připojení k internetové síti, k Wi-Fi síti, lokalizace pomocí GPS, ovládání komponentů analogovým signálem a podobně. Všechny tyto a mnohé další funkce lze doplnit připojením správného Shieldu.

Každý Shield je určen pro konkrétní rozšíření a pro konkrétní použití. Například Arduino Wi-Fi Shield umožňuje připojení k bezdrátové Wi-Fi síti a dále je vybaven slotem pro SD kartu, žádná další rozšíření nenabízí. Konstrukce je navržena tak, aby bylo možno realizovat jednoduché a rychlé připojení Shieldu na deskou Arduino. To je provedeno spojením Shieldu a horní části desky Arduina do takzvaného „sendviče“, viz Obr. 2 b). Toto propojení nemusí být trvalé, dá se kdykoliv rozpojit. Tím vzniká veliká flexibilita zařízení.

Jednou z dalších výhod je, že stejně jako u samotných desek Arduino, ani zde není uživatel omezen pouze na Shieldy, které jsou oficiálně vytvořeny. Může si podle daných postupů sestavit Shield, který bude obsahovat komponenty vyhovující dané konkrétní aplikaci.

Použití Shieldu má i své nevýhody a to v tom, že zařízení zvětší svoje rozměry, dále některé výstupy z desky Arduino budou obsazeny Shieldem a nepůjde je dále využívat pro jiné operace.



a) *b)*

Obr.2 *a) Arduino UNO [5]*
b) Arduino UNO spojené s Wi-Fi Shieldem [4]

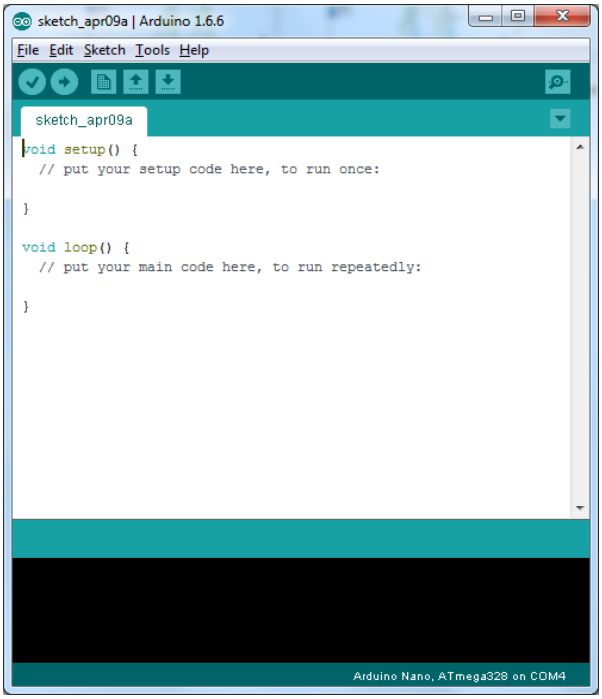
2.3 Programování vývojového kitu Arduino

2.3.1 Programovací prostředí

Arduino lze programovat v programovacím jazyce C, případně C++, to je ovšem v některých případech složité a vyžaduje to dobré znalosti programování v tomto jazyce. K ulehčení a větší srozumitelnosti programování byl vytvořen software Arduino IDE, sloužící zároveň pro rychlou a jednoduchou komunikaci s deskou. Software je vytvořený

v jazyce Java. Je tedy kompatibilní se všemi operačními systémy podporující tento jazyk. Arduino IDE pro programování používá nástroj Wiring, jedná se v podstatě o knihovnu z programovacího jazyka C++.

Po nainstalování aplikace do počítače a následném spuštění se zobrazí okno, viz Obr 3. Hlavní části okna tvoří pět tlačítek v levém horním rohu, které slouží pro ověřování, ukládání, nebo nahrávání kódu do mikroprocesoru desky. Dále tlačítka pro otevírání už vytvořených kódů nebo tlačítko pro založení nového projektu pro psaní kódu. V pravé části okna je jedno tlačítko, které má název *Serial Monitor*. Tlačítko slouží k zobrazování dat, které vysílá deska do počítače. Největší část okna tvoří textový editor pro psaní kódu. V editoru jsou předepsány dva příkazy: *void setup()* a *void loop()*. Za každý z těchto příkazů se píše daná část kódu do složených závorek. Tyto dva příkazy musí vždy kód obsahovat, i kdyby za nimi ve složených závorkách nebylo nic napsáno. Bez těchto dvou příkazů bude kód neúplný a nepůjde nahrát na mikroprocesor Arduina. V dolní části okna se objeví chybové hlášení.



Obr.3 *Úvodní zobrazení Arduino*

2.3.2 Popis struktury kódu

Psaní kódu má svá pravidla a svou strukturu, která je podobná většině programovacích prostředí. V první části kódu se většinou píše deklarace proměnných, není to však pravidlem. Proměnná se může deklarovat kdekoliv v programu, avšak musí být

dřív deklarována, než bude použita. Deklarace určuje, jakého datového typu proměnná bude, např. byte, boolean, integer, atd. Za touto první částí programu následují již zmíněné příkazy *void setup()* a *void loop()*. Za příkaz *void setup()* se píše do složených závorek část kódu, která se provede pouze jednou a to při spuštění programu, např. nastavení některých pinů jako vstupních, nebo výstupních. Za příkaz *void loop()* se do složených závorek píše ta část kódu, která se bude po spuštění programu neustále opakovat.

Příkazů, které se v kódu dají použít, je celá řada. Běžně je využíváno podmíněných příkazů, které pracují s porovnávacími operátory, což jsou např.: rovnost, nerovnost, větší hodnota, menší hodnota apod. Těchto operátorů také využívají cyklické příkazy, které cyklicky opakují část kódu spadajícího pod tento příkaz, dokud není splněna nadefinovaná podmínka. Jednotlivé příkazy se oddělují středníkem. Při psaní kódu se rozeznávají malá a velká písmena. V celém kódu se dají kdekoliv psát poznámky, ty však musí být odděleny dvěma lomítky, aby text nebyl brán, jako součást kódu.

2.3.3 Knihovny

V horní liště okna Arduina IDE se nalézá záložka *Sketch* a v této záložce je na výběr možnost *Include Library*. Zde je seznam standardních knihoven, jimiž software disponuje. Knihovny přidávají další funkce do programovacího prostředí. Prostřednictvím těchto funkcí se dá v programu jednoduše nastavit manipulace se samotným Arduinem (ukládání dat na jeho paměť, čtení a mazání paměti apod.) nebo ovládání přidaných Shieldů a modulů. Pro příklad takovým modulem může být LCD display, u kterého by bylo velmi složité a zdlouhavé programovat každý pixel. Vložená knihovna přímo umožňuje zobrazování slov, různých obrazců a spoustu dalšího, tím se práce s displayem výrazně zjednoduší.

Jednotlivé knihovny obsahují své specifické funkce, a proto se s každou knihovnou pracuje jinak. Ke knihovnám často bývají instrukce nebo názorné příklady, aby bylo zřejmé, jak s nimi zacházet a jak je aplikovat v kódu. Inicializace v programu má pro všechny knihovny stejný tvar, a tím je: *#include<jméno_knihovny.h>*. Od řádku na kterém je provedena inicializace, je v kódu nahrána knihovna a mohou se používat její příkazy. Výhodou knihoven je, že jich lze v jednom kódu využít neomezené množství a také lze doplnit Arduino IDE o knihovny vytvořené samotnými uživateli.

3. Pneumatické ventily

3.1 Obecný popis

Pneumatické ventily, někdy označovány jako pneumatické rozvaděče, slouží k ovládání pneumatických obvodů. Jejich úkolem je řízení směru, zastavení nebo uvolnění proudu tlakového média. Dále regulace tlaku nebo průtokového množství. Pro zvolení správného typu ventilu k dané operaci, jsou zásadní informace: počet vstupů a výstupů, počet a funkce stavů a způsob ovládání ventilu. Mezi další parametry, které jsou dány hlavně konstrukcí ventilu, patří maximální tlak a průtok ovládaného média a také rozmezí pracovní teploty.

Stavy neboli polohy ventilu mají své charakteristické vnitřní uspořádání. Stanovují, jak bude proudící médium ovlivněno. Nejčastěji určují směr proudění, zastavení, hrazení media a podobné úkony. Ventily mají různé počty stavů, které jim jsou uděleny už od výrobce. Stav, ve kterém se ventil nachází při nečinnosti jeho ovládacích prvků, je označován za výchozí stav. Při volném průchodu media rozvaděčem je nastaven tzv. průchozí stav. Je-li nastaven uzavřený stav, ventilem nemůže medium protékat.

Ovládání pneumatických ventilů může být realizováno pěti způsoby. Prvním způsobem je manuální ovládání, kdy ventil je vybaven buď pákou, tlačítkem, nebo aretační západkou. Ovládání zde zajišťuje lidská síla. Druhou možností je mechanické ovládání ventilu. Realizováno je pomocí narážky, kladky, kladky s volným zpětným chodem, nebo pružinou. Ventil je ovládán mechanickou silou například od pružiny, nebo od pístu, který najel na kladku, apod. Za třetí je elektromagnetické ovládání. Síla je vytvářena za pomocí elektrické cívky. V jádru cívky je umístěna kotva, která je spojena s posuvnými prvky uvnitř ventilu. Po přivedení proudu se vytvoří v jádru cívky magnetické pole, které posouvá kotvu a tím se uvádí ventil do potřebného stavu. Dalším typem je ovládání pomocí tlakového média. Rozvaděč je zkonstruován tak, že pro jeho přestavení je využívána síla tlakového signálu. Poslední možností je kombinované ovládání. Přestavení ventilu do jiného stavu může být realizováno například pomocí pružiny a elektromagnetu.

U ventilů se rozlišuje, zda se jedná o ventil monostabilní, nebo bistabilní. Rozdílnost je v tom, že monostabilní ventil má výchozí pozici a po přerušení řídícího signálu se sám automaticky přestaví do výchozí pozice. Automatické navrácení je většinou

realizováno pomocí pružiny. Bistabilní ventily mají dvě, nebo více poloh, ve kterých se mohou nacházet bez trvalého působení řídícího signálu.

Vstupy a výstupy ventilů jsou označovány dle Tab.1. Toto označení se nachází ve schématech a nezřídka bývá uvedeno i na samotných ventilech.

Tab.1 Označení vstupů a výstupů ventilů

Druh vývodu	Číselné označení	Označení písmeny
Přívod tlaku	1	P
Vývod na pracovní výstup	2,4,6	A,B,C
Vývod na odvzdušnění nebo výpusť	3,5,7	R,S,T
Vstup na řídícím přívodu	10,11,12,14	X,Y,Z

3.2 Použité pneumatické ventily

3.2.1 Elektromagnetický pneumatický ventil SMC

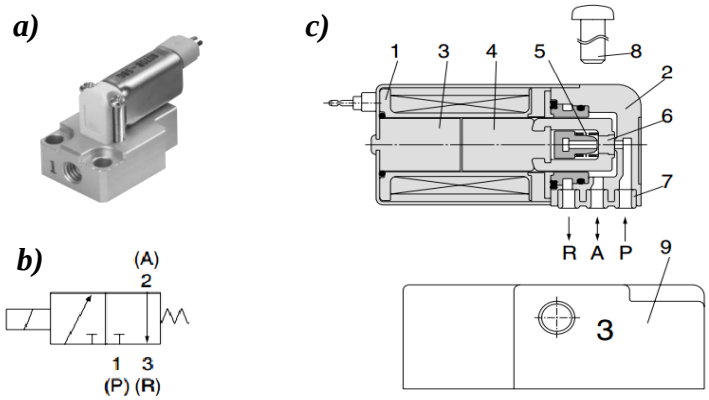
Jedná se o dvoustavový monostabilní ventil s přesným označením *SMC S070B-6AG*. Kde S070 je typové označení, B označuje uchycení s deskou podstavy pomocí šroubů, 6 značí napájecí napětí elektromagnetu 12 V (5 označuje napájecí napětí elektromagnetu 24 V). Napájecí napětí musí být stejnosměrné. Písmenem A je označena spotřeba el energie 0,35 W a maximální provozní tlak 0,1 MPa. Poslední písmeno G značí způsob elektrického připojení.

Schéma ventilu a konstrukční provedení je patrné z Obr. 4. Konstrukce je tvořena hliníkovou podstavnou deskou s kanálky pro proudící médium. K podstavné desce je dvěma šrouby připojeno pryskyřicové tělo, chránící cívku uloženou v tomto těle. V dutině cívky je jádro z nerezové oceli. Při průchodu elektrického proudu cívkou, se v dutině cívky vytvoří magnetické pole, které přitahuje kotvu s pryžovým elementem na konci, zajišťujícím těsnost. Tím se ventil přestaví z výchozího stavu do druhé polohy. Po přerušení průtoku elektrického proudu se kotva pomocí pružiny vrátí do výchozí polohy.

K dalším specifikacím tohoto ventilu patří rozsah pracovních teplot, který je -10 až 50 °C. Proudícím tlakovým médiem může být pouze vzduch, nebo inertní plyny. Mazání mechanických nebo posuvných částí ventilu není výrobcem požadováno. Tento typ ventilů vyniká malou hmotností a nízkou hlučností. Je odolný vůči vibracím v rozmezí 45 až 2000 Hz. Elektromagnet je tvořen cívkou typu solenoid. Jmenovité stejnosměrné

napájecí napětí cívky je 12 V, kde je dovoleno kolísání ±10 %. Maximální proud protékající cívkou je 30 mA.

Dle značení pneumatických prvků je tento ventil označen 3/2. Kde 3 znázorňuje počet vstupů a výstupů dohromady. Číslo 2 znázorňuje počet stavů ventilu, viz schéma na Obr. 4 b).



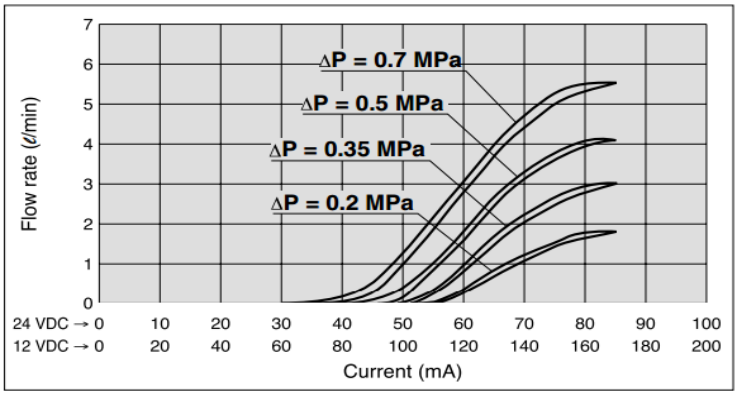
Obr. 4 *a) elektromagnetický pneumatický ventil SMC [11]*
b) schématická značka [11]
c) konstrukční provedení [11]: 1-cívka, 2- pryskyřicové tělo, 3-nerezové jádro, 4-kotva, 5-vratná pružina, 6-sedlo, 7-těsnící vložka, 8-šroub spojující podstavnou desku a tělo ventilu, 9-podstavná deska

3.2.2 Proporcionální elektromagnetický pneumatický ventil SMC

Podobně jako předešlý dvoustavový ventil, tak i tento má celé označení SMC PVQ13-6M-03-M5-A. Kde PVQ13 je typové označení, 6 opět značí stejnosměrné napájecí napětí 12 V. Typ konektoru, sloužící k připojení vodičů, je označen písmenem M. 03 vyjadřuje průměr ústí 0,3mm, které je uvnitř ventilu. K této velkosti průměru je výrobcem udáván maximální provozní tlak 0,7 MPa. M5 je značení podstavné desky, která je zde mosazná. Poslední písmeno A značí typ těsnění mezi podstavou a samotným ventilem. Těsnění je zde na bázi fluoroelastomeru a má tvar O-kroužku. Konstrukční provedení ventilu je patrné z Obr. 6. Na obrázku je zobrazena i schématická značka,

Základní vlastností proporcionálního ventilu je, že se nepřepíná mezi dvěma stavy skokově, ale plynule. Díky tomu se dá plynule ovládat průtokové množství proudícího media. Ventil je ve výchozí poloze uzavřen, což je způsobeno pružinou, která působí na kotvu s těsnícím elementem, dosedající na vstupní hrdlo uvnitř ventilu. Ovládací prvek je elektromagnet, vytvořen z cívky typu solenoid. Protékáním elektrického proudu cívkou

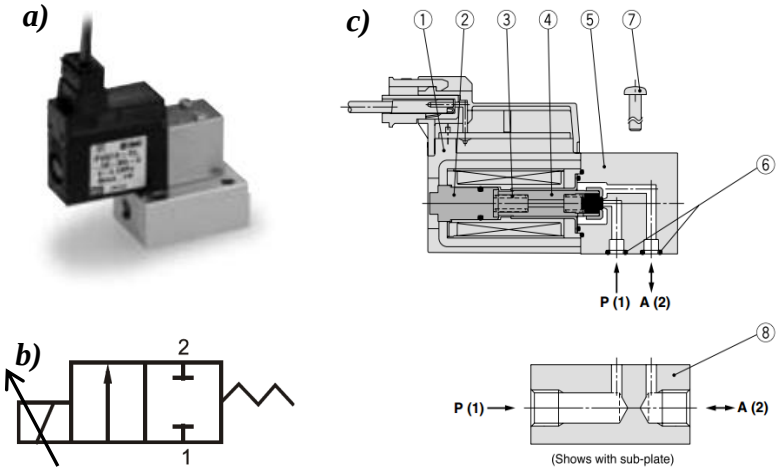
se vytvoří magnetické pole, které působí na kotvu a překonáním síly od pružiny, otevře ventil. Průtokové množství plynu ve ventilu je závislé na velikosti elektrického proudu, protékajícího cívkou, viz průtoková charakteristika na Obr. 5.



Obr. 5 *Průtoková charakteristika proporcionálního elektromagnetického pneumatického ventilu SMC [12]*

Elektrický proud, protékající cívkou ventilu, může být 0 až 170 mA. Z konstrukce je patrné, že se jedná o ventil monostabilní, protože je do výchozí polohy automaticky vracen pružinou.

Ventil je schopen pracovat s médiem o teplotách 0 až 50 °C, médium musí být inertní. Spotřeba elektrické energie jsou 2 W, při maximálním zatížení.



Obr. 6 *a) proporcionální elektromagnetický pneumatický ventil SMC [12]*
b) schématická značka
c) konstrukční provedení [12]: 1-elektromagnetická sestava, 2- jádro z nerezové oceli, 3-vratná pružina, 4-sestava kotvy, 5-tělo se vstupním a výstupním kanálkem, 6-těsnící O-kroužek, 7-šroub spojující podstavnou desku a tělo ventilu, 8-podstavná deska

4. Řízení

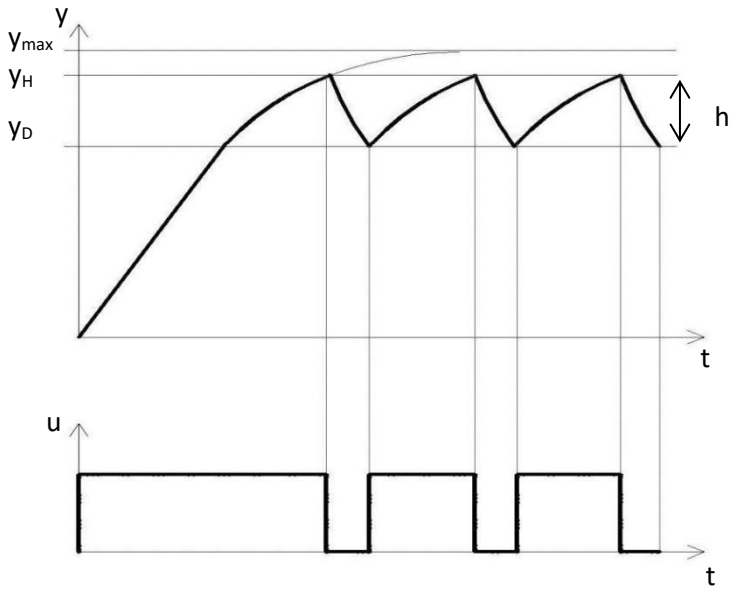
Cílevědomé působení na řízený objekt s cílem dosáhnout předem určeného stavu se dá nazvat řízením. Řízení, kdy regulátor (řídící jednotka) řídí systém pouze ze vstupních údajů, tedy hodnot řídící veličiny a nemá žádnou informaci o výstupních hodnotách, se nazývá ovládání. U ovládání regulátor nereaguje na žádnou změnu v systému, to může být výhodné, pokud do systému vstupuje veliké množství rušivých vlivů. Pokud existuje zpětná vazba a regulátor pracuje i s hodnotami regulované veličiny, tedy hodnotami na výstupu ze systému, jedná se o regulaci. Odečtením regulované veličiny od řídící veličiny vznikne regulační odchylka, to je hodnota vyjadřující, jak moc se liší stav na výstupu ze systému od požadovaného stavu. Výstupem z regulátoru je akční veličina, která je zároveň vstupem do systému.

4.1 Dvoustavová regulace

Mezi jednodušší způsoby regulace patří tzv. dvoustavová regulace, někdy nazývána jako dvoupolohová regulace. Jednoduchost je především v konstrukci regulátoru. Příkazy pro akční člen jsou u tohoto typu regulace velmi jednoduché a to buď zapnuto, nebo vypnuto. Také se používají termíny z logiky log. 1, nebo log. 0.

Její charakteristickou vlastností je, že se regulační odchylka nikdy neustálí na nule. Regulovaná veličina se periodicky pohybuje kolem požadované hodnoty v určitém rozmezí a vykazuje hysterezní chování. Rozmezí, ve kterém se regulovaná veličina pohybuje má horní a spodní hranici. Posunutím těchto hranic blíže k požadované hodnotě, zmenšíme mezní hodnoty regulační odchylky a zvýšíme přesnost. Ovšem v průběhu řízení pak dochází k rychlejšímu spínání řídících prvků soustavy a to může snižovat jejich životnost, nebo způsobit časté nárazy přenesené do systému. Naopak posunutím hranic dál od sebe, se sníží četnost spínání řídících prvků soustavy, ale zhorší se jakost celkové regulace.

Z vlastností dvoustavové regulace vyplývá, že má nespojitý charakter, jak je patrné z obrázku 7. Regulování tímto způsobem se používá u nenáročných nebo méně náročných aplikací, kde nejsou veliké požadavky na přesnost. Typické použití tohoto regulátoru je regulace teploty. Například regulování teploty vody v boileru, nebo regulace teploty pracovní části žehličky. Nicméně má využití i v jiných oblastech, jako je například udržení stanovené hodnoty tlaku vzduchu ve vzdušníku, a podobně.



Obr.7 *Průběh akční a regulované veličiny dvoustavového regulátoru 1. řádu*

y – regulovaná veličina, y_H – horní mez, y_D – dolní mez, t – čas,
h – hystereze, u – akční veličina

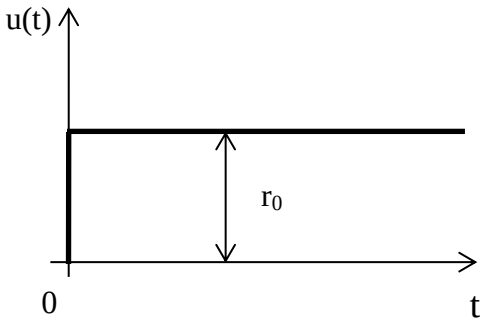
4.2 Spojitá regulace

4.2.1 P-regulátor

P-regulátor neboli proporcionální regulátor. Principem funkce tohoto regulátoru je zesílení hodnoty regulační odchylky, získané z porovnávacího členu, který je předřazen samotnému regulátoru. Porovnávací člen zjistí regulační odchylku na základě rozdílu řídící veličiny a regulované veličiny. Hodnota akční veličiny regulátoru typ P pak plyne ze vztahu (4.1).

$$u(t) = r_0 \cdot e(t) \qquad (4.1)$$

Kde u(t) je hodnota akční veličiny závislá na čase, r_0 značí zesílení P-regulátoru, e(t) vyjadřuje regulační odchylku. Zesílení r_0 je konstanta a nezávisí na čase, je tedy patrné, že akční veličina je přímo úměrná regulační odchylce. V praxi se toto zesílení dost často označuje K_p a platí $K_p = r_0$. Přechodová charakteristika P-regulátoru je na Obr. 8. Charakteristika znázorňuje odezvu systému na jednotkový skok při nulových počátečních podmínkách. Při skokové změně



Obr.8 *Přechodová charakteristika ideálního P-regulátoru*

u(t)-akční veličina, r_0-zesílení, t-čas

vstupní veličiny se výstupní veličina téměř okamžitě ustálí na nové hodnotě.

Proporcionální regulátory mají trvalou regulační odchylku, tzn. že regulační odchylka nikdy nebude nulová, za předpokladu, že máme systém v ustáleném stavu. Zvýšením hodnoty zesílení, se sníží velikost trvalé regulační odchylky, tím se zvýší přesnost regulátoru. Nicméně při vysokých hodnotách zesílené může docházet ke snižování stability (platí pro statickou soustavu, neplatí pro astatickou) a systém se může uvést do nestabilního stavu. Při nastavení regulátoru se musí volit kompromis mezi vysokou přesností a stabilitou regulačního obvodu.

U P-regulátorů se často pracuje s hodnotou pásma proporcionality, vztah (4.2). V tomto pásmu nebo na jeho hranici se nachází požadovaná hodnota regulované veličiny. Pokud se regulovaná veličina nenachází v pásmu proporcionality, akční člen není proporcionálně regulován, ale je nastaven na plný výkon. Pásmo proporcionality je nepřímo úměrné zesílení regulátoru, označuje se pp a udává se jako šířka pásma v procentech z celého regulačního rozsahu.

$$pp = \frac{1}{r_0} \cdot 100\ [\%] \qquad (4.2)$$

4.2.2 I-regulátor

Nezkráceným názvem je to Integrační regulátor. Jeho velikou výhodou je, že dokáže úplně odstranit regulační odchylku. Odstranění regulační odchylky trvá určitou dobu. Závisí to na nastavených parametrech regulátoru. U statických soustav 1. řádu je I-regulátor téměř vždy stabilní, u astatických soustav tomu tak není. V případě astatických soustav se totiž vždy chová nestabilně, proto zde nemůže být samostatně použit. Pro soustavy vyšších řádů se I-regulátor často používá v kombinaci s proporcionálním členem. Jedním z důvodů tohoto použití je zvýšení stability.

Akční veličinu na výstupu z regulátoru lze vyjádřit vztahem (4.3).

$$u(t) = r_{-1} \int_0^t e(\tau) d\tau \qquad (4.3)$$

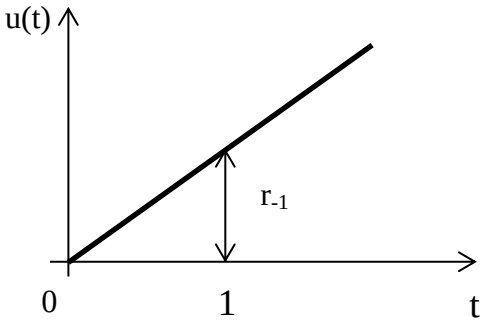
Kde u(t) je hodnota akční veličiny závislá na čase, r_{-1} značí zesílení integrační složky regulátoru, e(τ) je regulační odchylka závislá na změně času.

Jediný parametr, který u tohoto typu regulátoru lze nastavit pro seřízení regulátoru, je časová integrační konstanta. Je definována vztahem (4.4) a vyjádřená v časových jednotkách.

$$T_i = \frac{r_0}{r_{-1}} \ [s] \qquad (4.4)$$

Slovně lze časová integrační konstanta popsat: jako doba, za kterou výstupní veličina integračního regulátoru dosáhne stejné hodnoty, jaké by dosáhla, kdyby přenos regulátoru byl pouze proporcionální a pásmo proporcionality by bylo 100%. [9]

Přechodová charakteristika je zobrazena na Obr. 9, z charakteristiky je patrný astatismus.



Obr.9 *Přechodová charakteristika ideálního I-regulátoru*

u(t)-akční veličina, r_{-1}-zesílení, t-čas

4.2.3 PI-regulátor

PI je označení pro proporcionálně-integrační regulátor. Proporcionální člen bude mít stejnou úlohu jako u samotného P-regulátoru. V podstatě tedy zaručí poměrně krátkou dobu náběhu, ale nedokáže odstranit trvalou regulační odchylku v ustáleném stavu. K odstranění trvalé regulační odchylky zde poslouží integrační člen regulátoru. Ten je schopný pracovat bez trvalé regulační odchylky. Samotný I-regulátor má v mnoha případech nízkou stabilitu. V případě astatických soustav je dokonce vždy nestabilní. Spojením I regulátoru s proporcionálním členem, se stabilita výrazně zlepší.

Integrační složka regulátoru je závislá na čase, jak vyplývá ze vztahu (4.5), ve kterém je matematicky popsán výstup (akční veličina) PI-regulátoru.

$$u(t) = r_0 \cdot e(t) + r_{-1} \int_0^t e(\tau) d\tau \qquad (4.5)$$

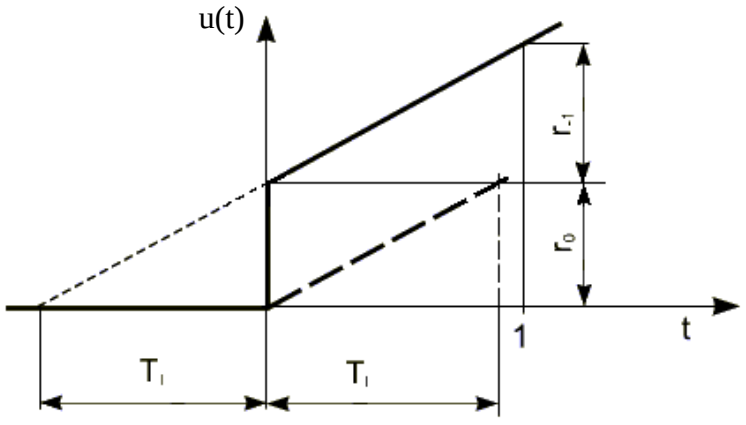
Tento vztah vzniknul součtem vztahů (4.3) a (4.1) a takovýmto způsobem regulátor funguje i v praxi. Sečtou se hodnoty proporcionální a integrační složky. V regulačním

pochodu nejprve převládá vliv proporcionální složky a s narůstajícím časem začne převládat vliv integrační složky.

Nastavení konstant PI-regulátoru už není tak jednoduché, jako tomu bylo u předchozích regulátorů. Pro seřízení se dá využít Wadeho metoda, která má následující kroky:

1. Úplně se vypne integrační složka. To se dá udělat tak, že konstantu integračního zesílení bude rovna nule ($r_{-1} = 0$). Dále se nastaví zesílení proporcionální složky (r_0) na malou hodnotu.
2. Zesílení r_0 se postupně zvyšuje tak dlouho, dokud při skokové změně řídící veličiny na vstupu, nebude mít výstup požadovaný průběh. V ustáleném stavu zde bude přítomná trvalá regulační odchylka. Po tomto postupu je seřízená proporcionální složka regulátoru.
3. K seřízení r_{-1}, tedy zesílení integrační složky regulátoru, je potřeba nejprve snížit hodnotu proporcionálního zesílení r_0 na 75% předchozí hodnoty. Poté se postupně zvyšuje hodnota r_{-1} a tím se zvyšuje vliv integračního členu. Zvyšování r_{-1} probíhá, dokud není odstraněna trvalá regulační odchylka.
4. Posledním krokem, při seřizování PI-regulátoru, je navrácení hodnoty r_0 zpět na 100% původní hodnoty.

Přechodová charakteristika PI-regulátoru je zobrazena na Obr.10.



Obr.10 *Přechodová charakteristika PI regulátoru*

u(t)-akční veličina, r_0-zesílení proporcionálního členu, r_{-1}-zesílení integračního členu, t-čas, Ti-integrační časová konstanta

5. Praktická část

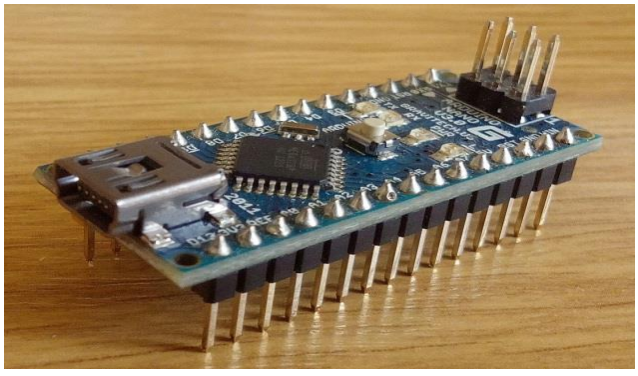
Praktická část se zabývá realizací elektrických i pneumatických obvodů, jejich řízením a využitím v praxi. Řízení je prováděno pomocí Arduina. Jsou zde tedy vysvětleny jednotlivé kroky jeho programu. Praktická část je rozdělena do jednotlivých bodů:

- Sestrojení obvodu a vytvoření algoritmu pro dvoustavovou regulaci tlaku
- Sestrojení obvodu a vytvoření algoritmu pro spojitou regulaci:
    a) pomocí proporcionálního regulátoru
    b) pomocí integračního regulátoru
    c) pomocí proporcionálně – integračního regulátoru
- Aplikování a používání vytvořených regulačních obvodů pro řízení tuhosti sedáku v autosedačce

5.1 Realizace dvoustavového regulátoru

5.1.1 Součásti obvodu a jeho sestavení

Pro dvoustavovou regulaci byla použita deska s označením Arduino NANO V3.0 (Obr. 11). Její výhody jsou: veliké množství vstupů a výstupů (pouze analogových a digitálních pinů je zde dohromady 20), dále malé rozměry a nízká hmotnost. I přes malé rozměry má deska umožněno jednoduše komunikovat s počítačem přes rozhraní USB a to díky konektoru Mini-USB, který na desce nezabere tolik místa. Zároveň je přes tento konektor přivedeno napájení. Deska je osazena mikroprocesorem ATmega328, který disponuje provozní frekvencí 20 MHz a Flash pamětí 32 kB, což je pro danou úlohu více než dostačující. Algoritmus projektu dvoustavové regulace zabírá 6 180 bytů Flash paměti.



Obr.11 *Arduino NANO V3.0*

Akční členy jsou tvořeny dvoustavovými pneumatickými ventily *SMC S070B-6AG* popsanými v kapitole 3.2.1. Tyto ventily byly v úloze použity v počtu dvou kusů. Jeden je určen pro otevírání a zavírání přívodu tlakového vzduchu do soustavy a druhý je určen pro výpust tlakového vzduchu ze sestavy. Pneumatické i elektrické schéma úlohy je vyobrazeno na Obr. 12. Pro demonstraci se řídil tlak v tlakové láhvi. Celé řešení by se dalo zrealizovat za pomoci pouze jednoho ventilu 3/2. Použití dvou ventilů vede k větší univerzálnosti zapojení. Zmíněné ventily lze jednoduše zaměnit za ventily 2/2 bez větších změn v programu.

Jelikož použité ventily mají napájecí napětí 12 V, nebylo možné je napájet přímo z Arduina, které má výstupní napětí 5 V. Napájení bylo zajištěno z extérního zdroje generujícího stejnosměrné napětí 12 V, tím vznikly dva elektrické obvody (primární 5 V a sekundární 12 V). Galvanické oddělení obvodů bylo realizováno pomocí optočlenu TLP250. Je to součástka obsahující světlo emitující LED diodu, fotodiodu a spínací tranzistory.

Primární obvod je tvořen samotným Arduinem generujícím napětí a primární částí optočlenu, která je tvořena zabudovanou LED diodou. Před LED diodou, musí být předřazen rezistor. Úbytek napětí na LED diodě je 1,2 V a protékající proud je 10 mA. Hodnoty úbytku napětí a proudu byly zjištěny v dokumentaci optočlenu. Odpor rezistoru předcházejícího diodě pak lze vypočítat z upraveného Ohmova zákona (5.1).

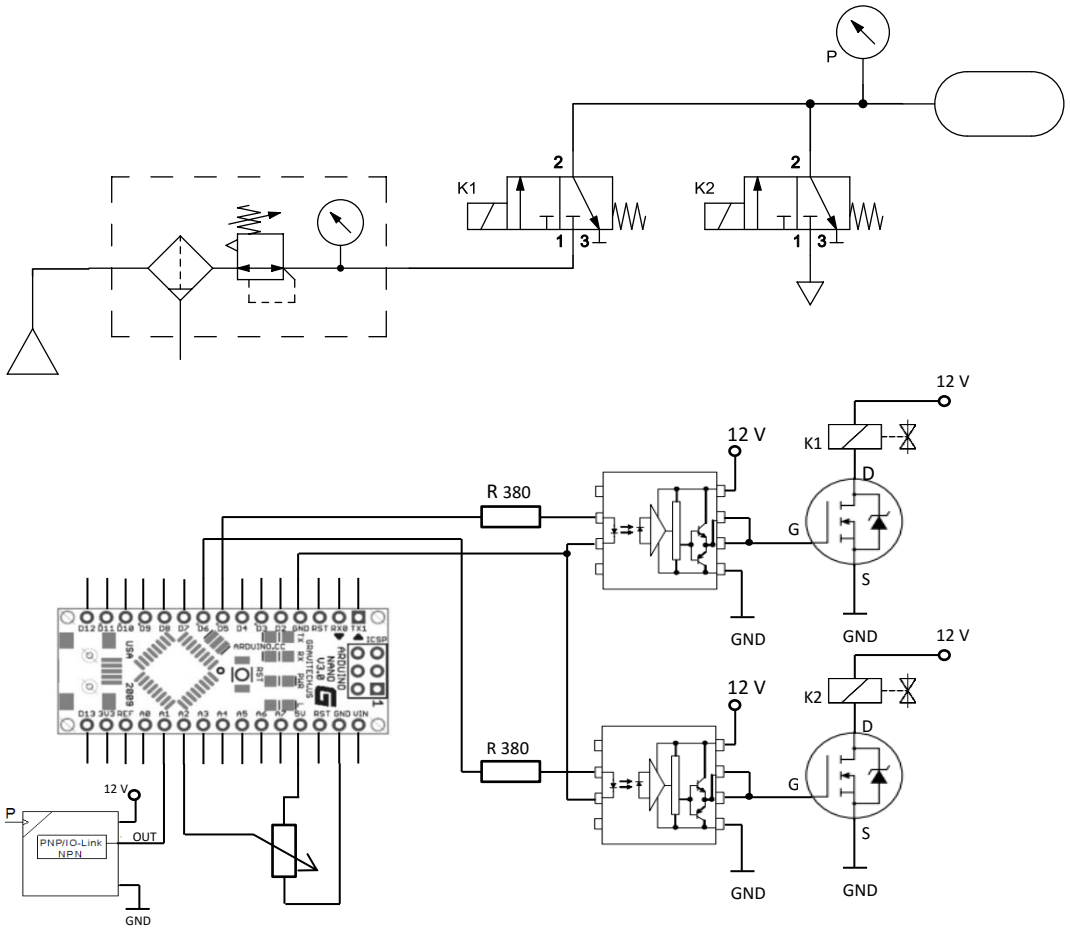
$$R = \frac{U_Z - U_D}{I} \ [\Omega] \qquad (5.1)$$

kde: R – odpor rezistoru, U_Z – napětí na zdroji, U_D – úbytek napětí na diodě optočlenu

$$R = \frac{5 - 1{,}2}{10 \cdot 10^{-3}} = 380\ \Omega$$

Sekundární část optočlenu je připojena na napájení 12 V a také z ní je vyvedeno řídící napětí. Opět z důvodu větší univerzálnosti zapojení, nebylo přivedeno řídící napětí přímo na cívku ventilu, ale na řídící elektrodu MOSFET tranzistoru. Díky tomu je možno stávající ventily zaměnit za jiné, které jsou výkonnostně náročnější a optočlen by jejich proudové zatížení nebyl schopen zvládnout. Ke každému ventilu zvlášť náleží jeden optočlen a jeden MOSFET tranzistor.

Realizace regulátoru byla simulována na vytvořeném obvodu, kde systém byl nahrazen tlakovou lahví. Velikost požadovaného tlaku je určována potenciometrem, jehož výstup je připojen přímo k Arduinu. Regulovanou veličinu neboli skutečnou hodnotu tlaku v tlakové lahvi snímá tlakový senzor, který posílá informaci o naměřeném tlaku řídícímu členu, tedy Arduinu. Díky tomu je vytvořena zpětná vazba. Nastavené a naměřené hodnoty lze zobrazovat na připojeném počítači, kde slouží k dalšímu zpracování.



Obr.12 *Pneumatické a elektrické schéma zapojení dvoustavového regulátoru*

Tlakový senzor, který je zapojen v obvodu, pracuje v rozmezí -100 až 100 kPa. Je tedy potřeba se senzorem pracovat tak, aby nedošlo k jeho přetížení vysokým tlakem. Výhodou tohoto senzoru je nízké napájecí napětí, které může být v intervalu 12 až 24 V, takže je napájen z pomocného zdroje, stejně jako ventily. Analogový signál, který se objevuje na výstupu ze senzoru je v hodnotách 0 až 5 V a je v lineární závislosti na měřeném tlaku. Linearita je v přesnosti ±0,4 %.

5.1.2 Program

V první části programu jsou nadefinovány vstupy a výstupy, které se budou používat a také proměnné. U každé proměnné je provedena její deklarace. Jde o přiřazení datových typů jednotlivým proměnným. Také je zde definovaná konstanta *a* a *b*.

Konstanta *b* slouží k posunutí rozsahu do nuly. Analogový signál se převádí na digitální, který je v rozsahu hodnot 0 až 1023. Ovšem senzor při 0 kPa má na výstupu hodnotu 3,02 V. Tomu odpovídá digitální hodnota 608 (je to způsobeno tím, že senzor měří i záporné hodnoty tlaku). Proto konstanta *b* má hodnotu 608 a dále se bude v programu odečítat od naměřené digitální hodnoty ze senzoru. Po posunutí je rozsah digitálních hodnot 0 až 415 pro kladné hodnoty tlaku (rozdíl hodnot 1023 a 608), kde 0 = 0 kPa a 415 = 100 kPa.

Konstanta *a*, které je přiřazena hodnota $\frac{50}{207}$, slouží k převodu digitálních hodnot ze senzoru nebo potenciometru na hodnoty tlaku. Vykazované hodnoty jsou pak v kPa.

```
////////////DEKLARACE PROMĚNNÝCH
byte A = 5;    // Ovládací pin prvního ventilu
byte B = 6;    // Ovládací pin druhého ventilu
byte senz = A1;  // Pin pro výstup z tlakového senzoru
byte pot = A2;  // pin pro výstup z potenciometru
double val;   // proměnná pro hodnotu tlakového senzoru
double set;  // proměnná pro hodnotu potenciometru

const float a = (50.)/(207); // konstanta pro převod odečtené hodnoty
                          //z tlakového senzoru na hodnotu tlaku [kPa]

const float b = 608 ; //konstanta posouvá rozsah do nuly
```

V další proměnné je uložena hodnota rozsahu hystereze, která je zadána v kPa. Hned poté je převedena do rozsahu signálu 0 - 415. Velikost hodnoty hystereze závisí na typu aplikace regulátoru. V našem případě bylo experimentálně zjištěno, že vyhovující hodnota je 3.

```
float hkPa = 3; // rozsah hystereze v kPa
//Pozn.: pracovní tlaky čidla: min tlak = -100 kPa, max tlak = 100 kPa
float h = hkPa / a; // rozsah hystereze [0 - 415]
```

Čas je zaznamenáván do deklarované proměnné *cas*. Udává dobu od spuštění programu.

```
unsigned long cas; //proměnná pro záznam času
```

Následující část kódu, je část nastavení. Začíná příkazem *void setup*, za tímto příkazem jsou ve složených závorkách příkazy, které se provedou pouze jednou a to hned po spuštění programu. Jsou to příkazy, které nastavují piny pro ovládání ventilů, jako výstupní piny. Dále je zde zahájena sériová komunikace s počítačem a vypsání informace o začátku regulace na sériovém monitoru.

```
void setup() {
  // Nastavení pinů, jako výstupních
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  Serial.begin(9600); // Zahájení sériové komunikace
  Serial.println("ZAČÁTEK REGULACE"); // vytvoří nápis
}
```

Dále je úsek kódu, který se bude neustále opakovat. To je zajištěno příkazem *void loop*. Zbylá část kódu se píše do složených závorek za tento příkaz. Kód ve složených závorkách začíná načtením hodnot z potenciometru a z tlakového senzoru. Hodnota z tlakového senzoru je díky odečtení konstanty *b* v rozsahu 0 – 415. Hodnotu z potenciometru převedeme do stejného rozsahu. Načtené hodnoty se díky následujícím příkazům pro sériovou komunikaci hned zobrazí v počítači, ovšem už v převedeném tvaru vykazujícím jednotky tlaku, protože jsou vynásobeny konstantou *a*. Zobrazen bude i aktuální čas v milisekundách.

```
void loop() {
  // Načtení řídící a regulované veličiny
  val = analogRead(senz)-b;
  set = map(analogRead(pot), 0, 1023, 0, 415); //načtení a změna
měřítka

  //výpis stavu tlaku [kPa] a požadované hodnoty tlaku [kPa]
  cas = millis();
  Serial.print(cas);
  Serial.print(";");
  Serial.print(set * a);
  Serial.print(";");
  Serial.println(val * a);
```

Následuje cyklus *while,* který opakuje určitý úsek kódu, pokud platí nadefinovaná podmínka. Tou je v tomto případě porovnání tlaků. Pokud hodnota naměřená senzorem je menší, než hodnota nastavená na potenciometr plus polovina hodnoty hystereze (uvažujeme, že požadovaná hodnota je uprostřed rozsahu hystereze), pak je podmínka splněna. Ventil na přívodu tlakového vzduchu je otevřen tzn. *digitalWrite(A, HIGH)*. Další příkazy cyklu opět načtou hodnotu ze senzoru a potenciometru a opět je vypsána aktuální

hodnota tlaku a aktuální čas. Když podmínka přestane být platná, postupuje se dále v programu a následuje pokyn pro zavření přívodního ventilu *digitalWrite(A, LOW)*.

```
while (val < (set + (h / 2))) {
    digitalWrite(A, HIGH);
    val = analogRead(senz)-b;
    set = map(analogRead(pot), 0, 1023, 0, 415);

  //výpis stavu tlaku [kPa]
    cas = millis();
    Serial.print(cas);
    Serial.print(";");
    Serial.print(set * a);
    Serial.print(";");
    Serial.println(val * a);
}
digitalWrite(A, LOW); // Po dosažení yh uzavře plnící ventil
```

Pro vypouštění tlaku vzduchu je použit stejný cyklus, akorát je ovládán druhý ventil. Cyklus probíhá, pokud je splněna podmínka, že tlak na senzoru je vyšší, než tlak nastavený na potenciometru mínus jedna polovina hystereze (uvažujeme požadovanou hodnotu uprostřed rozsahu hystereze). Zároveň v podmínce musí platit, že hodnota z tlakového senzoru je vyšší jak nula. Tím je opatřeno, že podmínka nebude splněna, pokud se na potenciometru nastaví nula, a tedy cyklus se nebude opakovat donekonečna, ale postoupí se dále v programu. Opět se při každém opakování příkazů v cyklu *while* vypisují naměřené hodnoty ze senzoru a potenciometru a hodnota času. Poslední příkaz v programu zavře vypouštěcí ventil a program se bude znovu opakovat od příkazu *void loop*.

```
while (val > (set - (h / 2)) && (val > 0)) {
  digitalWrite(B, HIGH);
  val = analogRead(senz)-b;
  set = map(analogRead(pot), 0, 1023, 0, 415);
  //výpis stavu tlaku [kPa] a požadované hodnoty tlaku [kPa]
    cas = millis();
Serial.print(cas);
Serial.print(";");
Serial.print(set * a);
Serial.print(";");
Serial.println(val * a);
}
digitalWrite(B, LOW);// Po dosažení yd uzavře vypouštěcí ventil
```

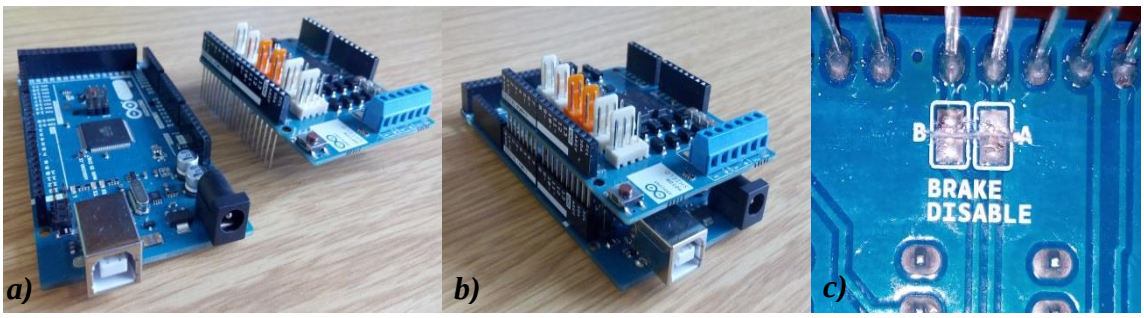
Kompletní program dvoustavového regulátoru je vložen v příloze č. 1. V příloze č. 5 jsou grafy průběhů regulace tlaku v tlakové lahvi.

5.2 Realizace spojitých regulátorů

5.2.1 Součásti obvodu a jeho sestavení

V této úloze byly použity proporcionální ventily SMC PVQ13-6M-03-M5-A, jejichž specifikace jsou vypsány v kapitole 3.2.2. Ventily zde byly použity v počtu dvou kusů. Stejně jako v aplikaci dvoustavové regulace. Jeden ventil složí pro regulaci přiváděného vzduchu do systému a druhý pro vypouštění do atmosféry. Schopnost těchto ventilů, plynule měnit průtočné množství tlakového vzduchu, je v realizaci spojitého regulátoru nepostradatelná.

K řízení spojitého regulátoru byla použita deska s označením Arduino MEGA 2560. Hlavním důvodem této volby bylo použití MOTOR Shieldu, který je s touto deskou kompatibilní ohledně „sendvičového“ spojení, viz Obr.13 b). Jinak se tento typ desky často používá k ovládání robotů, protože má širokou škálu vybavení. Disponuje 54 digitálními vstupy/výstupy (záleží na nastavení v programu) a z toho lze až 15 využít k ovládání pomocí PWM. Analogových vstupů je na desce 16. Mikroprocesor pracuje s taktovací frekvencí 16 MHz a Flash pamětí 256 KB.



Obr.13 *a) Arduino Mega a MOTOR Shield v rozloženém stavu*
b) Arduino Mega a MOTOR Shield spojené
c) deaktivace brzdy

MOTOR Shield je potřeba k ovládání proporcionálních ventilů, neboť disponuje analogovými výstupy. Samotné Arduino nemá výstup na analogový signál, má pouze výstupy generující PWM signál a právě toho umí využívat MOTOR Shield, jako vstupního řídícího signálu.

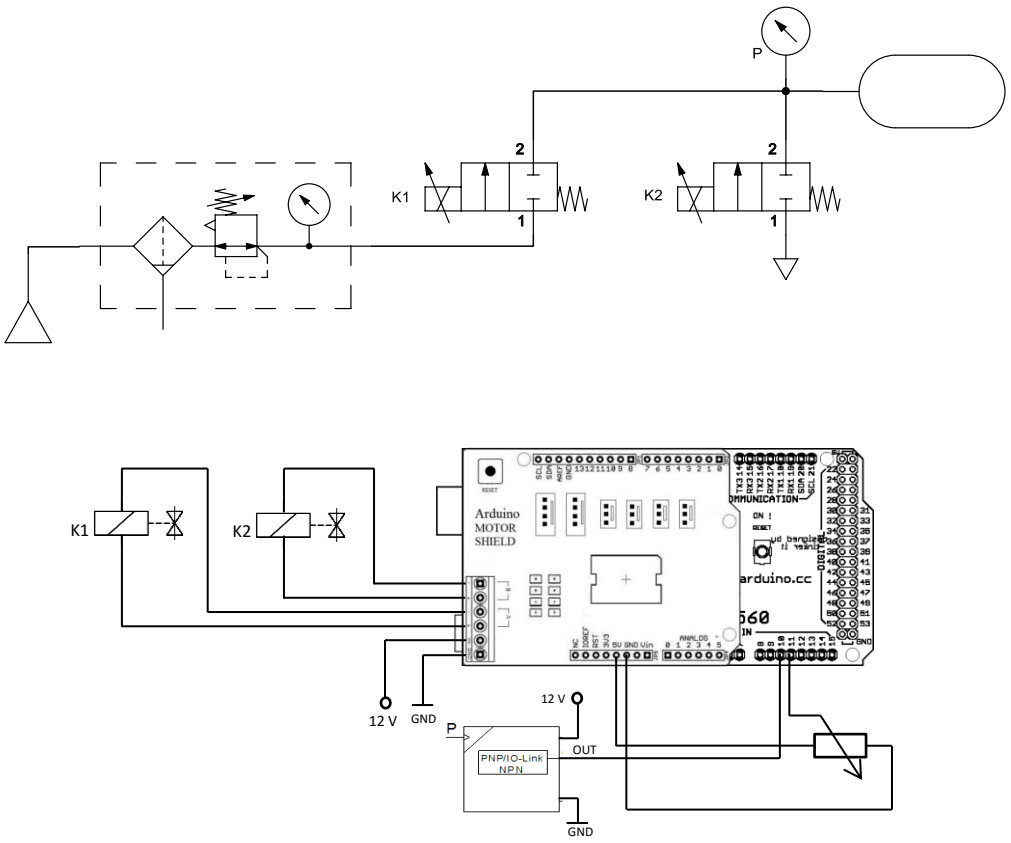
Jeho zaměření je hlavně na napájení indukčních zátěží, jako jsou elektromotory, relé, elektromagnety a také je vhodný pro napájení zmíněných proporcionálních ventilů. Napájecí napětí těchto elektroventilů je 12 V a maximální odebíraný elektrický proud je 170 mA. Z porovnání s maximální hodnotou proudu na výstupních pinech Shieldu,

která je 2 A, je zřejmé, že z tohoto hlediska nevznikne problém v překročení horní hranice dovoleného elektrického proudu. Tudíž ventily mohly být připojeny přímo na výstupy MOTOR Shieldu. Napájení Arduina a připojeného Shieldu je vyřešeno externím zdrojem stejnosměrného napětí, který dodává 12 V. Tím je také zajištěno dostatečné napájecí napětí pro elektroventily. Samotné napájení přes USB by v tomto případě nebylo vhodné, neboť napětí 5 V přivedené z USB není dostačující.

Na MOTOR Shieldu je naletováno, mimo jiného, šest šroubových svorek, které jsou pro tuto úlohu důležité. Dvě z nich jsou označeny Vin a GND. Na ně se dá připojit externí zdroj, který napájí MOTOR Shield a zároveň desku Arduino. Takže pokud zdroj nemá příslušné zakončení vodičů, které by pasovalo do napájecího konektoru Arduina, dá se využít těchto šroubových svorek. Další čtyři svorky jsou střídavě označeny (+) a (-). Jsou to kladné a záporné póly použitelné pro napájení ventilů. Jejich napětí se dá řídit a jeho maximální hodnota není 5 V, jako u ostatních výstupů, ale odpovídá hodnotě napájecího napětí, tedy 12 V. Tato hodnota napětí je zároveň nejvyšší doporučená hodnota pro napájení desky Arduino MEGA.

Velikost napětí na výstupních šroubových svorkách je ovládána pomocí digitálních pinů 3 a 11. Pro ventil ovládající vstup tlakového vzduchu je využíváno pinu 3. Ventil pro vypouštění vzduchu je ovládán pomocí pinu 11. Tyto piny jsou označeny PWM, to znamená, že umí vysílat pulzní signály. Nastavením programu se dá definovat šířka pulzu signálu, který Shield přijímá od Arduina a podle šířky pulzu nastavuje velikost výstupního napětí na svorkách.

MOTOR Shield má jednu vlastnost, která je určena speciálně pro řízení motorů a to je brždění protiproudem, tzn. že na záporném pólu se objeví napětí, pokud je požadováno snížení otáček elektromotoru. Je to nežádoucí vlastnost pro ovládání použitých elektroventilů. Ventily vrací do původního stavu pružina, není zapotřebí protiproudu. Shield je zkonstruován tak, že nabízí elegantní řešení deaktivace brždění protiproudem, a to přerušením obvodu na zadní straně plošného spoje, viz Obr. 13 c). Po lehkém proškrábnutí mezi vyobrazenými kontakty ostrým předmětem (jehlou), je obvod přerušen, a tedy brždění protiproudem je deaktivováno. Přerušení obvodu není nevratné, v případě potřeby lze za pomoci pájecí stanice obvod spojit.



Obr.14 *Pneumatické a elektrické schéma zapojení spojitého regulátoru*

Na Obr.14 je nakresleno schéma pneumatického a elektrického obvodu. Z pneumatické části je patrné, že použité ventily jsou typu 2/2 a jsou monostabilní. Šipky nakreslené přes cívky ventilů značí proporcionální ovládání ventilu. V elektrické části je schematicky zobrazen komplet Arduina a MOTOR Shieldu. Ve spodní části elektrického schématu je obvod s potenciometrem a s tlakovým senzorem. Nalevo jsou obvody s cívkami, které ovládají ventily.

Zpětná vazba spojitého regulátoru byla zrealizována stejně jako u nespojitého. Do pneumatického obvodu je zařazen tlakový senzor, který snímá hodnotu regulované veličiny a hodnota řídící veličiny je zadávána pomocí potenciometru.

Toto vytvořené zapojení bylo použito pro všechny typy spojitých regulátorů, které byly zmiňovány v předešlých kapitolách.

5.2.2 Program P-regulátoru

Program začíná typicky deklarováním a definováním proměnných. První dva příkazy definují piny pro výstup z potenciometru a z tlakového senzoru. V dalších čtyřech proměnných je uloženo označení pinů, kterými se ovládá MOTOR Shield . Jakou funkci vykonávají jednotlivé piny je dáno výrobcem. Všechny čtyři piny jsou výstupní. Pin s označením 3 vysílá PWM signál a podle jeho hodnoty MOTOR Shield přivádí napětí na vstupní ventil. Hodnota napětí přiváděná na vstupní ventil je přímo úměrná hodnotě PWM signálu. Pin 12 rozhoduje, zda napětí bude přiváděno na kladný nebo záporný pól vstupního ventilu. Pin 11 má stejnou funkci jako pin 3, ale ovládá napětí na výstupním ventilu. Stejně tak pin 13 má stejnou funkci jako pin 12, ale určuje polaritu na výstupní ventil.

```
byte cidlo = A12; // hodnota regulované veličiny
byte pot = A11; // hodnota žádané veličiny (potenciometr)

byte pwm_A = 3; // rychlost A
byte dir_A = 12; // polarizace A
byte pwm_B = 11; // rychlost B
byte dir_B = 13; // polarizace B
```

Další deklarované proměnné jsou nezbytné pro výpočet proporcionálního regulátoru. Patří mezi ně regulovaná veličina, akční veličina, řídící veličina a regulační odchylka. V následujícím příkazu je přiřazována hodnota konstantě P. Jedná se o proporcionální konstantu neboli konstantu zesílení proporcionální složky regulátoru. Experimentálním způsobem byl zjištěn vyhovující průběh P-regulátoru a to při hodnotě konstanty P = 10. Systémem byla tlaková lahev.

```
double w, y, u, e;
//y – regulovaná veličina
//w – řídící veličina
//u – Akční veličina
//e – regulační odchylka

//P – proporcionální konstanta
const double P = 10; //nastavení proporcionální konstanty
```

Posledními deklarovanými položkami jsou: konstanta *a*, konstanta *b*, a proměnná pro záznam času *cas*. Konstanta *a* s hodnotou $\frac{20}{51}$ slouží k převodu signálu do jednotek tlaku v kPa. Konstanta *b* má podobnou úlohu jako *a*, ale převádí pouze hodnotu akční veličiny na hodnotu napětí ve voltech. Těmito konstantami se vynásobí výsledná vypočtená

nebo naměřená hodnota až při vypisování hodnot do sériového monitoru, takže toto převedení nemá ve výpočtu žádný vliv.

```
const float a = (20. / 51); //převodní konstanta na kPa
const float b = (1. / 51); //převodní konstanta akční veličiny na volty

unsigned long cas; //proměnná pro záznam času
```

V části nastavení (*void setup()*) je zahájena sériová komunikace a potřebné piny jsou zde nastaveny jako piny výstupní.

```
void setup() {
  Serial.begin(9600); //Start sériové komunikace
  Serial.println("Zaciname"); // Vypíše nápis

  //Nastavení pinů jako výstupních
  pinMode(dir_A, OUTPUT);
  pinMode(dir_B, OUTPUT);
}
```

Ve smyčce (*void loop()*) je jako první stanoveno, že se bude přivádět napětí na kladné póly elektroventilů.

```
digitalWrite(dir_A, HIGH); //napětí na kladný pól vstupního ventilu
  digitalWrite(dir_B, HIGH); //napětí na kladný pól vypouštěcího ventilu
```

Načtená hodnota regulované veličiny musí být opět posunuta do rozsahu 0 až 415 odečtením hodnoty 608. Regulovaná veličina ale nemůže zůstat v tomto rozsahu, protože bude použita pro výpočet akční veličiny, která musí být v rozsahu 0 až 255. Akční veličina je totiž použita jako hodnota PWM výstupu, který akceptuje hodnoty 0 až 255. Z tohoto důvodu je regulovaná veličina převedena do rozsahu 0 – 255. Stejně tak je převedena řídící veličina hned po jejím načtení.

```
  y = map(analogRead(cidlo) - 608, 0, 415, 0, 255);
  w = map(analogRead(pot), 0, 1023, 0, 255);
```

Následují příkazy pro výpočet regulační odchylky a pro výpočet akční veličiny. Akční veličina svou hodnotou určuje, jak moc bude otevřen ventil. Při hodnotě 255 je naplno otevřen ventil pro zvyšování tlaku a při hodnotě -255 je plně otevřen ventil pro snižování tlaku. Aby nedocházelo k překročení těchto hodnot, jsou nastaveny podmínky pro akční veličinu. Pokud je akční veličina větší než 255, její hodnota se bude stále rovnat 255 a pokud je akční veličiny menší než -255, její hodnota se bude stále rovnat -255.

```
  e = w - y; //výpočet regulační odchylky
    u = P * e; // výpočet P-regulátoru (akční veličiny)

if (u > 255)
    {
      u = 255;
    }
  if (u < -255)
    {
      u = -255;
    }
```

Poslední úsek smyčky obsahuje záznam času, vypsání vypočtených hodnot a rozhodovací podmínkou, zda bude napětí přiváděno na plnící nebo vypouštěcí ventil, respektive zda se bude zvyšovat, nebo snižovat tlak. Závisí to na tom, jestli se akční veličina pohybuje v kladných, nebo záporných hodnotách. Jak moc se otevře ventil, pro zvyšování nebo snižování tlaku je dáno velikostí hodnoty akční veličiny *u*.

```
  cas = millis(); //záznam času

  /////vypsání hodnot///
  Serial.print(cas);
  Serial.print("; ");
  Serial.print(w * a);
  Serial.print("; ");
  Serial.print(u * b);
  Serial.print("; ");
  Serial.println(y * a);

///rozhodnutí o zvyšování, nebo snižování tlaku/////
  if (u >= 0)
  {
    analogWrite(pwm_B, 0);
    analogWrite(pwm_A, u);
  }
  else
  {
    analogWrite(pwm_A, 0);
    analogWrite(pwm_B, -u);
  }
```

Celková podoba kódu je vložena v příloze č. 2 a grafy průběhů jsou v příloze č. 6.

5.2.3 Program I-regulátoru

Deklarace proměnných a konstant je stejná jako v případě P regulátoru s tím rozdílem, že není definována proporcionální konstanta P, ale integrační konstanta I. Vyjadřuje zesílení integrační složky regulátoru. Testováním regulátoru byla zjištěna vyhovující hodnota integrační konstanty I = 16. Systémem byla opět tlaková lahev. Ve výpočtu také figuruje pomocná integrační proměnná *yI*, kterou musíme také deklarovat.

```
const double I = 16; // nastavení integrační konstanty
double yI = 0; //pomocná integrační proměnná
```

Na rozdíl od P-regulátoru zde hraje velikou roli čas a jeho snímání, protože se používá ve výpočtu akční veličiny. Je tedy nutno vytvořit proměnné a konstanty určující čas. Konstantě s názvem *frekvence* je přiřazena hodnota vzorkovací frekvence. Do proměnné *dt* se bude ukládat čas uplynulý od posledního výpočtu akční veličiny. V proměnné *lasttime* bude uložen čas posledního výpočtu akční veličiny a v proměnné *now* bude uložen aktuální čas, který uplynul od spuštění programu.

```
const int frekvence = 50; // vzorkovací frekvence [ms]
int dt; //proměnná pro čas mezi dvěma kroky
unsigned long lasttime; //proměnná pro čas posledního vypočítání
unsigned long now; //proměnná pro čas od spuštění programu

float x; //proměnná pro čas dt v sekundách
float g; //pomocná proměnná pro hlídání hodnoty akční veličiny
```

Pro výpočet akční veličiny je potřeba převést čas mezi jednotlivými kroky na sekundy, k tomu bude sloužit proměnná *x*. Hodnota pomocné proměnné *g* poslouží k hlídání rozsahu 0 – 255, aby nedošlo k jeho překročení akční veličinou.

Stejně jako v kódu P-regulátoru i zde budou použity konstanty *a, b* pro převod jednotek.

Úsek *void setup()* je totožný jako u P-regulátoru, proto přeskočíme k rozdílům, které jsou v části *void loop()*. První změnou, která v předchozí regulaci nebyla je záznam času. Do proměnné *now* se načte čas od spuštění programu. V proměnné *lasttime* je uložen čas, kdy proběhl poslední výpočet (při první proběhnutí programu je *lasttime* = 0). Odečtením hodnot těchto proměnných získáme hodnotu *dt*, tedy čas od posledního výpočtu. Vzorkovací frekvence nám určuje, jak často bude regulátor vyhodnocovat hodnoty. Jde o nahrazení spojitého signálu posloupností vzorků. Neustálím porovnáváním vzorkovací frekvence a času od posledního výpočtu regulátoru se zjišťuje, jestli má proběhnout další vyhodnocení. O to se stará podmínka if.

```
now = millis(); //záznam času (čas od spuštění programu)
dt = (now - lasttime); // čas od posledního výpočtu regulátoru

if (frekvence <= dt) //rozhoduje se, jestli už uplynul čas pro vzorek
```

Výpočet samotného regulátoru začíná vypočtením regulační odchylky *e*. Následuje výpočet *yI*, což je pomocná proměnná, díky které je nahrazeno analytické integrování numerickým výpočtem. Jedná se o získání vzorku vynásobením hodnoty *e* s hodnotou *x* a tento vzorek je následně přičten k součtu všech předchozích vzorků. Hodnota *x* je čas uložen v proměnné *dt*, který je převeden na sekundy.

```
e = w - y; //výpočet regulační odchylky
x = (dt * 0.001); // převedení dt na sekundy
yI = (yI + e * (x)); //yI pomocná integrační proměnná
```

Jako u předchozího regulátoru, tak i zde bude hodnota akční veličiny určovat PWM signál, který následně bude pomocí MOTOR Shieldu přeměněn na Analogový. Potřebujeme tedy, aby hodnota akční veličiny nebyla větší, než 255, nebo menší než -255. To lze efektivně vyřešit omezením pomocné integrační konstanty *yI*. K tomu nám pomůže proměnná *g*. V té je uložen podíl maximální hodnoty akční veličiny ku hodnotě integrační konstanty. Takže pokud by hrozilo, že součin $I \cdot yI$ překročil hodnotu 255, tak se hodnota *g* zapíše do proměnné *yI* a součin $I \cdot yI$, což je v podstatě akční veličina, bude mít hodnotu přesně 255. To samé platí pro hodnotu akční veličiny -255, jen s tím rozdílem, že do *yI* se zapisuje hodnota *-g*.

```
    g = 255 / I;
    if (yI > g)
    {
      yI = g;
    }
    if (yI < -g)
    {
      yI = -g;
    }
    u = I * yI; //výpočet I-regulátoru
```

Zbylá část programu je opět stejná s programem pro proporcionální regulátor, jen je navíc jeden příkaz na konci. Jde o aktualizaci času posledního výpočtu.

```
lasttime = now; // aktualizace časového rozhraní
```

V příloze č. 3 je vložen kompletní kód I-regulátoru. Průběhy této regulace jsou přiloženy v příloze č. 7.

5.2.4 Program PI-regulátoru

PI- regulátor je kombinací P a I regulátoru, to znamená, že v programu budou použity všechny proměnné a konstanty, které byly použity v kódech jednotlivých regulátorů. V úseku deklarace je nadefinována proporcionální i integrační konstanta.

```
const double P = 10; //nastavení proporcionální konstanty
const double I = 8; // nastavení integrační konstanty
```

Hodnoty konstant byly nastaveny při provozu regulátoru, kdy bylo využito Wadeho metody pro seřízení regulátoru. Při nastavení P = 10, I = 8 byl zaznamenán malý překmit a relativně rychlé ustálení regulované veličiny na požadované hodnotě.

Přesný záznam času je pro výpočet PI-regulátoru stejně důležitý, jako tomu bylo u I-regulátoru, proto je to v kódu řešeno obdobným způsobem. Je tedy nastavená i konstanta vzorkovací frekvence na hodnotu 50 ms.

```
int frekvence = 50; // vzorkovací frekvence [ms]
int dt; // čas mezi dvěma kroky
unsigned long lasttime; // čas posledního vyhodnocení regulátoru
unsigned long now; // čas od spuštění programu
```

Část *void setup()* je opět stejná jako u dvou předchozích regulátorů, ale smyčka (*void loop()*) je trochu rozdílná. Hlavní rozdíl je ve výpočtu akční veličiny *u*, jejíž výsledek je součet proporcionální a integrační složky regulátoru. Tudíž podmínky porovnávající hodnoty *yI* a *g* nezajistí, že akční veličina nebude vyšší než 255 nebo nižší než - 255. Pouze hlídají, aby se integrační složka neubírala do hodnot velmi vzdálených od nuly, ale aby se pohybovali v rozmezí -255 a 255. V programu jsou tedy přidány podmínky hlídající přímo akční veličinu, aby její hodnota byla maximálně 255 a minimální hodnota -255.

```
    e = w - y; //výpočet regulační odchylky
    yI = (yI + e * (x)); //yI pomocná integrační proměnná
    g = 255 / I;
    if (yI > g)
    {
      yI = g;
    }
    if (yI < -g)
    {
      yI = -g;
    }
    u = P * e + I * yI; //výpočet I-regulátoru
```

```
if (u > 255)
    {
      u = 255;
    }
    if (u < -255)
    {
      u = -255;
    }
```

Tím je výpočet akční veličiny ukončen a následuje výpis dat, aktualizace časových proměnných a zapsání hodnoty akční veličiny na správný PWM výstup.

```
Serial.print(now);
    Serial.print("; ");
    Serial.print(w * a);
    Serial.print("; ");
    Serial.print(u * b);
    Serial.print("; ");
    Serial.println(y * a);

    ///rozhodnutí o zvyšování, nebo snižování tlaku/////
    if (u >= 0)
    {
      analogWrite(pwm_B, 0);
      analogWrite(pwm_A, u);
    }
    else
    {
      analogWrite(pwm_A, 0);
      analogWrite(pwm_B, -u);
    }
    lasttime = now; // aktualizace časového rozhraní
  }
```

Grafy zobrazující průběh regulace jsou vloženy v příloze č. 8 a v příloze č. 4 je vložen kompletní kód PI-regulátoru.

5.3 Regulování tuhosti sedáku autosedačky

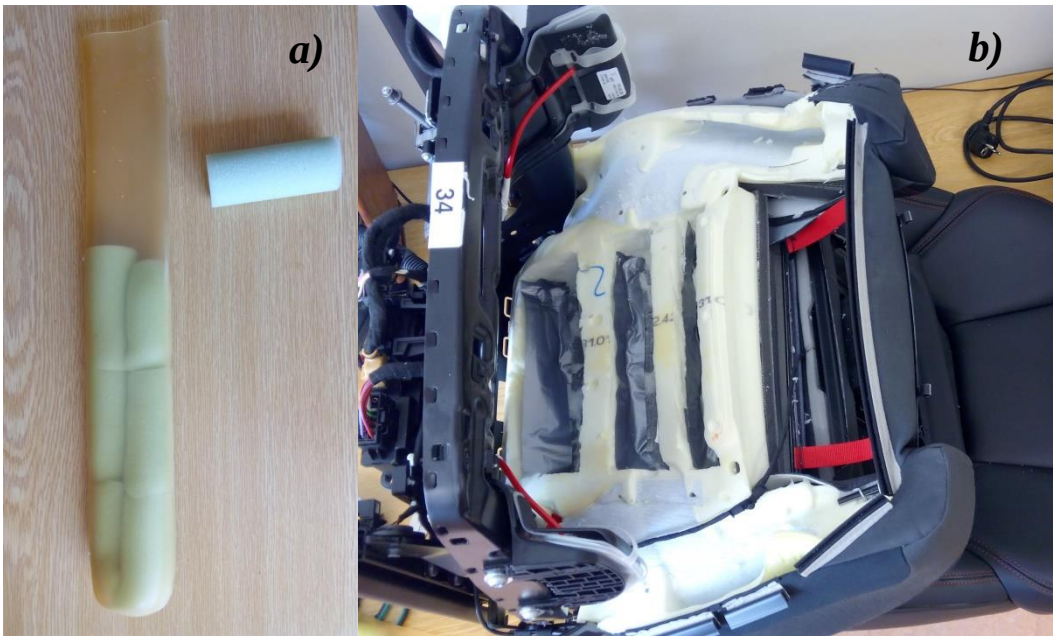
V této kapitole je uveden jeden ze způsobů praktického využití regulátoru tvořeného Arduinem, který pomocí pneumatických ventilů reguluje tlak ve vzduchových vacích v sedáku autosedačky. Je zde popsán postup úpravy autosedačky, zabudování vzduchových vaků do sedáku a připevnění ventilů i Arduina.

Autosedačka, na které byl celý proces prováděn, je z automobilu Porsche Cayenne viz Obr 15 a). Prvním úkolem bylo zabudování pryžových vaků na vzduch do sedáku sedačky. To obnášelo částečné rozebrání čalounění sedáku, čímž byl umožněn přístup k molitanovému korpusu. Na vrchní části molitanu, hned pod čalouněním, byla nalepena elektroinstalace na vyhřívání sedačky. Z tohoto důvodu bylo rozhodnuto, že vaky na vzduch budou do molitanového korpusu umisťovány ze spodní strany, která se dá odklopit od plechové konstrukce pod sedákem, jak je vidět na Obr 15 b).

Obr.15 *a) sedadlo z Porsche Cayenne*
b) odklopený sedák sedadla od plechové konstrukce

Po odklopení sedáku byly do jeho spodní části vyříznuty tři komory. Ty zaujímaly svojí pozici tak, že byly umístěny přesně mezi prošíváním čalounění, a tedy nezpůsobily tvarovou deformaci sedáku. Hloubka komor sahala těsně k elektrickému vyhřívání. Mezi vyhříváním a komorou zůstala asi půl centimetru tenká vrstva molitanu. Tato hloubka byla volena tak, aby regulování tuhosti sedáku bylo co nejznatelnější. Mezi jednotlivými komorami vedly molitanem kovové dráty, které držely tvar korpusu sedáku. Aby nedošlo ke kontaktu vaku s kovovým drátem, byly boky komor vylepeny lepicí páskou (Obr. 16 b)).

Dále následovala příprava pryžových vaků. Jejich tvar je válcový, ovšem ve vyfouklém jsou ploché a úzké a sedák by ve vyfouklém stavu vypadal značně zdeformovaně. Bylo tedy nutné je naplnit molitanem (Obr. 16 a)). K dispozici byly molitanové válečky, jejichž tuhost byla nižší, než tuhost odebraného molitanu ze sedáku. Díky tomu byly vaky měkčí, než původní molitan sedáku a zároveň bylo zabráněno možnému zdeformováni sedačky ve vyfouklém stavu.



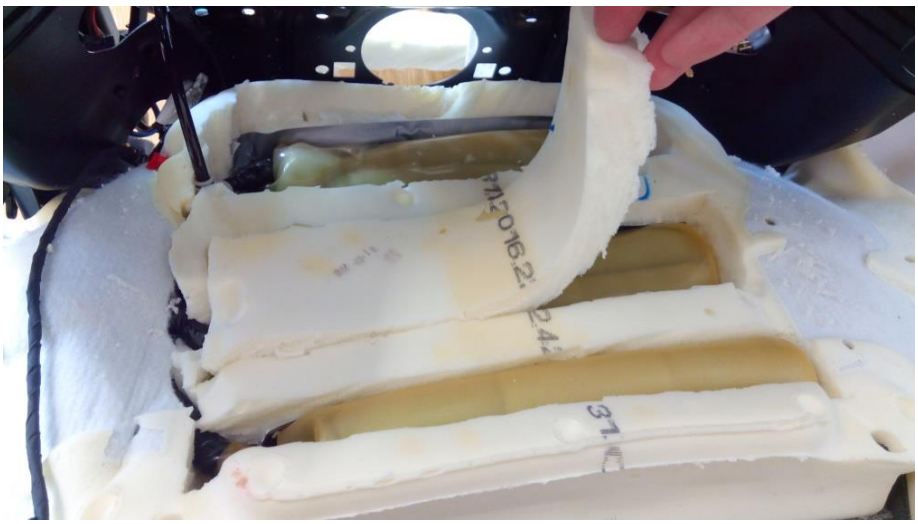
Obr.16 *a) pryžový vak*
b) komory vyřezané do spodní části sedáku

Vaky naplněné molitanovými válečky byly umístěny do připravených komor. Jejich otevřené konce byly zakráceny, aby vyhovovaly délce komory a následně byla na otevřený konec připojena hadička. Připojení bylo provedeno obmotáním volného konce vaku okolo hadičky a přelepením těsnící páskou. Následně byla provedena kontrola těsnosti nafouknutím vaku.

Všechny hadičky pro přívod tlakového vzduchu do vaků byly pomocí T-spojek propojeny. Tak byla vytvořena jedna přívodní hadička, která byla vyvedena ze spodní strany sedačky.

Nyní bylo vše nachystáno a následovalo zakrývání a zajišťování vaků. Takže molitany, které byly ze sedáku vyjmuty pro přípravu komor, byly následně seříznuty a posloužily pro překrytí pryžových vaků na spodní straně sedáku (obr 17).

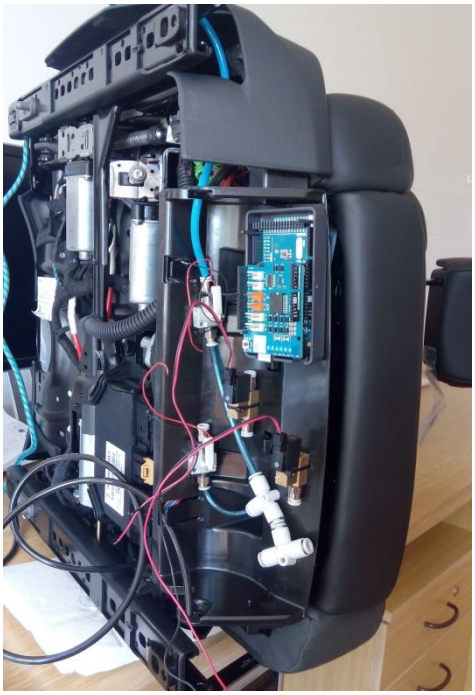
Celá spodní strana byla přelepena lepicí páskou. Sedák byl přiklopen zpět k plechové konstrukci a zajištěn. Čalounění bylo napnuto a uchyceno na původních místech.



Obr.17 *Vaky uložené v sedáku*

Místo přihrádky ve spodní části sedadla bylo připevněno Arduino s ventily (Obr. 18). Po připojení elektrického a pneumatického obvodu a připojení na přívod tlakového vzduchu, bylo vše připraveno k regulování tuhosti sedáku.

Jako první byl nahrán program pro dvoustavovou regulaci. Využívaly se tedy



Obr.18 *Umístění Arduina a ventilů na sedadle*

dvoustavové ventily. Systém tvořen pryžovými vaky se chová jinak než systém tvořen tlakovou lahví. Bylo tedy nutné nastavit vyhovující rozsah hystereze. Experimentováním byl zvolen rozsah 2,5 kPa. Při menším rozsahu bylo způsobeno příliš časté přepínání ventilů, naopak při zvolení většího rozsahu se na sedáku projevovaly znatelné tlakové pulzy.

Následovalo testování spojitých regulátorů, a tedy v pneumatickém obvodu byly zapojeny proporcionální ventily. Prvním testovaným byl proporcionální regulátor. Seřízení probíhalo nastavěním proporcionální konstanty na malou hodnotu, která byla 0,3. Postupným zvyšováním se docílilo vhodného průběhu regulované veličiny. Proporcionální konstanta byla navýšena až na hodnotu šestnáct, kdy systém vykazoval vhodný průběh. V ustáleném stavu byla zřejmá trvalá regulační odchylka.

Hned po nahrání programu integračního regulátoru se mohla nastavovat integrační konstanta. Nastavovala se obdobným způsobem jako u P-regulátoru. Soustava měla nejlepší průběh regulované veličiny s hodnotou integrační konstanty rovné čtrnácti. Při této hodnotě byl zaznamenán malý překmit a celkem rychlé ustálení regulované veličiny na požadované hodnotě. Menší hodnoty integrační konstanty měly za následek pomalejší ustálení regulované veličiny a větší hodnoty způsobovaly větší překmit a větší šum.

Na závěr byl program integračního regulátoru nahrazen programem s proporcionálně-integračním regulátorem. U něj je zapotřebí nastavit dvě konstanty: proporcionální a integrační. Proporcionální konstantu máme nastavenou z případu samotného P-regulátoru. Podle Wadeho metody, byla následně snížena na 70 % původní hodnoty, tedy z šestnácti na jedenáct. Potom byla navyšována integrační konstanta až na hodnotu osm, kdy soustava vykazovala nejmenší šum, rychlé ustálení regulační veličiny a ne tolik výrazný překmit. Navrácením proporcionální konstanty na 100 % původní hodnoty bylo seřizování dokončeno.

Pryžové vaky způsobovaly dynamické a nestálé chování soustavy. Z hlediska přesnosti a rychlosti se v této soustavě nejvíce osvědčil I-regulátor. P- regulátor pracoval bez překmitu, ale nevýhodou byla trvalá regulační odchylka a poměrně vysoký šum. Takovou prostřední variantou je PI-regulátor. Nemá trvalou regulační odchylku, efekt šumu je menší než u P-regulátoru, a překmit je menší, než u I-regulátoru, nicméně ustálení regulované veličiny trvá déle. U dvoustavové regulace bylo nestálým chováním systému způsobeno nepravidelné kmitání regulované veličiny v oblasti hystereze.

6. Závěr

V této práci se povedlo dosáhnout stanovených cílů. Nastudováním příručky pro programování Arduina [1] jsem získal schopnosti vytvořit jednoduché programy a tím jsem se naučil ovládat vstupy a výstupy Arduina. Vytvořil jsem mnoho programů, které ovládaly například jen LED diody a podobně. To mi pomohlo nasbírat zkušenosti a poznatky, které jsem mohl uplatnit jak při psaní teoretické části této práce, tak i při psaní složitějších programů.

Dalším úkolem bylo seznámit se s parametry pneumatických ventilů. To bylo velmi důležité pro budoucí sestavení obvodů a práci s nimi. Proudové a napěťové omezení Arduina totiž nedovolovalo připojit pneumatické ventily pro dvoustavovou regulaci přímo k Arduinu. Problém byl vyřešen náhradním zdrojem a galvanickým oddělením obvodů Arduina a náhradního zdroje. Řešení vyžadovalo základní znalosti elektrotechniky, které jsem v průběhu vypracovávání práce získával.

Následovaly kroky k vytvoření programů pro spojitou a nespojitou regulaci. Pochopení funkce dvoustavové regulace bylo předpokladem pro úspěšné vytvoření programu. Povedlo se mi vytvořit funkční program, který splňuje požadavky, které jsem si na začátku stanovil, jako je například nastavení hystereze, přehlednost, jednoduchost.
K vytvoření kódů spojitých regulátorů bylo opět klíčovým podrobně se seznámit s principem jejich funkce.

Po vytvoření regulátorů následovalo jejich zkoušení. Prvním postřehem při testování jednotlivých regulací bylo, že regulace spojitými regulátory jsou oproti nespojité regulaci o trochu pomalejší. Na vině jsou proporcionální ventily. Jejich průřez je při plném otevření 0,3 mm a to způsobuje pomalejší zvyšování a snižování tlaku.

I přesto, že spojité regulace byly pomalejší, tak jejich průběh v čase byl pořád vyhovující a jejich funkčnost byla správná. Mohl jsem tedy uskutečnit poslední bod této práce a tím byla regulace tuhosti sedáku autosedačky. Do sedáku byly úspěšně nainstalovány všechny komponenty potřebné pro řízení jeho tuhosti a to tak, že na něm nejsou patrné téměř žádné známky deformace tvaru. Regulování tuhosti je rychlé přesné a dá se uplatnit v reálných podmínkách.

Použitá literatura

[1] *Arduino: Příručka programátora* [online]. [cit. 2016-06-18]. Dostupné z: http://readgur.com/doc/131826/arduino-%E2%80%93-p%C5%99%C3%ADru%C4%8Dka-program%C3%A1tora

[2] VODA, Zbyšek. *Průvodce světem Arduina*. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

[3] *What is an Arduino?* [online]. [cit. 2016-06-18]. Dostupné z: https://learn.sparkfun.com/tutorials/what-is-an-arduino

[4] *GPS Shield - Arduino Uno* [online]. [cit. 2016-06-18]. Dostupné z: https://www.bananarobotics.com/shop/image/cache/data/sku/BR/0/1/0/1/3/BR010130-ITEAD-GPS-Shield/ITEAD-GPS-Shield-on-an-Arduino-600x600.jpg

[5] *Robotstore.cz* [online]. [cit. 2016-06-18]. Dostupné z: http://robotstore.cz/obchod/arduino/atmel-atmega328p-ch340g-arduino-uno-r3-2/

[6] *Knihovny Arduino* [online]. [cit. 2016-06-18]. Dostupné z: https://www.arduino.cc/en/Reference/Libraries

[7] SCHMID, Dietmar. *Řízení a regulace pro strojírenství a mechatroniku*. Překlad Jiří Handlíř. Praha: Europa - Sobotáles, 2005. ISBN 80-86706-10-9.

[8] HLAVA, Jaroslav. *Prostředky automatického řízení II: analogové a číselné regulátory, elektrické pohony, průmyslové komunikační systémy*. Praha: České vysoké učení technické, 2000. ISBN 80-01-02221-8.

[9] *Spojité regulátory* [online]. [cit. 2016-06-18]. Dostupné z: http://moodle.hradebni.cz/pluginfile.php/9100/mod_resource/content/0/spojite_regulatory.pdf

[10] *Mechatronika: Inovace ve výuce odborných předmětů – aplikace RVP do ŠVP* [online]. [cit. 2016-06-18]. Dostupné z: http://www.spssou-pe.cz/soubory/projekty/opvk/vystupy/mechatronika.pdf

[11] *Port Solenoid Valve Series S070* [online]. [cit. 2016-06-18]. Dostupné z: http://www.smcpneumatics.com/pdfs/S070.pdf

[12] *Compact Proportional Solenoid Valve, Series PVQ* [online]. [cit. 2016-06-18]. Dostupné z: http://stevenengineering.com/Tech_Support/PDFs/70PCPVQ.pdf

Seznam obrázků

Seznam tabulek

Seznam příloh

Obsah přiloženého CD

- text bakalářské práce
  - bakalarska_prace_2016_Petr_Marousek.pdf
- zdrojové kódy programů
  - dvoustavovy_regulator
  - P-regulator
  - I-regulator
  - PI-regulator

Příloha č. 1: Program dvoustavového regulátoru

```
//////////DEKLARACE PROMĚNNÝCH
byte A = 5;    // Ovládací pin prvního ventilu
byte B = 6;    // Ovládací pin druhého ventilu
byte senz = A1;   // Pin pro výstup z tlakového senzoru
byte pot = A2;   // pin pro výstup z potenciometru
double val;    // proměnná pro hodnotu tlakového senzoru
double set;   // proměnná pro hodnotu potenciometru

const float a = (50.)/(207); // konstanta pro převod odečtené hodnoty
                            //z tlakového senzoru na hodnotu tlaku [kPa]

const float b = 608 ; //konstanta posouvá rozsah do nuly

//nastavení hystereze:
float hkPa = 3; // rozsah hystereze v kPa
//Pozn.: pracovní tlaky čidla: min tlak = -100 kPa, max tlak = 100 kPa
float h = hkPa / a; // rozsah hystereze [0 - 415]

unsigned long cas; //proměnná pro záznam času

////////////////////////////////////////////////////////////////////
///////
void setup() {
  //Nastavení pinů, jako výstupních
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);

  Serial.begin(9600); // Zahájení sériové komunikace
  Serial.println("ZAČÁTEK REGULACE");
}

////////////////////////////////////////////////////////////////////
///////
void loop() {
  // Načtení řídící a regulované veličiny
  val = analogRead(senz)-b;
  set = map(analogRead(pot), 0, 1023, 0, 415); //načtení a změna
měřítka

  //výpis stavu tlaku [kPa] a požadované hodnoty tlaku [kPa]
  cas = millis();
  Serial.print(cas);
  Serial.print(";");
  Serial.print(set * a);
  Serial.print(";");
  Serial.println(val * a);

  // Zvyšování regulované veličiny až na horní
  // hodnotu hystereze yh, pokud už nebyla dosažena
  while (val < (set + (h / 2))) {
    digitalWrite(A, HIGH);
```

```
    val = analogRead(senz)-b;
     set = map(analogRead(pot), 0, 1023, 0, 415); //načtení a změna
měřítka

    //výpis stavu tlaku [kPa] a požadované hodnoty tlaku [kPa]
     cas = millis();
  Serial.print(cas);
  Serial.print(";");
  Serial.print(set * a);
  Serial.print(";");
  Serial.println(val * a);
  }
  digitalWrite(A, LOW); // Po dosažení yh uzavře plnící ventil

  // Snižování regulované veličiny na hodnotu
  // spodní hranice hystereze yd, pokud už nebyla dosažena
  while (val > (set - (h / 2)) && (val > 0)) {
    digitalWrite(B, HIGH);
    val = analogRead(senz)-b;
    set = map(analogRead(pot), 0, 1023, 0, 415); //načtení a změna
měřítka

    //výpis stavu tlaku [kPa] a požadované hodnoty tlaku [kPa]
      cas = millis();
  Serial.print(cas);
  Serial.print(";");
  Serial.print(set * a);
  Serial.print(";");
  Serial.println(val * a);
  }

  digitalWrite(B, LOW);//Po dosažení yd uzavře vypouštěcí ventil

}
```

Příloha č. 2: Program proporcionálního regulátoru

```
//////////Vstupy
byte cidlo = A12; // hodnota regulované veličiny
byte pot = A11; // hodnota žádané veličiny (potenciometr)

//////////Výstupy na ventily - pwmA přidá tlak, pwmB ubere tlak
byte pwm_A = 3; // rychlost A
byte dir_A = 12; // polarizace A
byte pwm_B = 11; // rychlost B
byte dir_B = 13; // polarizace B

double w, y, u, e;
//y - regulovaná veličina
//w - řídící veličina
//u - Akční veličina
//e - regulační odchylka

//P - proporcionální konstanta
const double P = 10; //nastavení proporcionální konstanty

const float a = (20. / 51); //převodní konstanta na kPa
const float b = (1. / 51); //převodní konstanta akční veličiny na volty

unsigned long cas; //proměnná pro záznam času
////////////////////////////////////////////////////////////

void setup() {

  Serial.begin(9600); //Start sériové komunikace
  Serial.println("Zaciname"); // Vypíše nápis

  //Nastavení pinů jako výstupních
  pinMode(dir_A, OUTPUT);
  pinMode(dir_B, OUTPUT);
}
////////////////////////////////////////////////////////////
void loop() {

  digitalWrite(dir_A, HIGH); //napětí na kladný pól vstupního ventilu
  digitalWrite(dir_B, HIGH); //napětí na kladný pól vypouštěcího ventilu

  y = map(analogRead(cidlo) - 608, 0, 415, 0, 255);
  w = map(analogRead(pot), 0, 1023, 0, 255);

  e = w - y; //výpočet regulační odchylky
    u = P * e; // výpočet P-regulátoru

    if (u > 255)
    {
      u = 255;
    }
  if (u < -255)
```

```
    }
      u = -255;
    }

  cas = millis(); //záznam času

  /////vypsání hodnot///
  Serial.print(cas);
  Serial.print("; ");
  Serial.print(w * a);
  Serial.print("; ");
  Serial.print(u * b);
  Serial.print("; ");
  Serial.println(y * a);

///rozhodnutí o zvyšování, nebo snižování tlaku/////
  if (u >= 0)
  {
    analogWrite(pwm_B, 0);
    analogWrite(pwm_A, u);
  }
  else
  {
    analogWrite(pwm_A, 0);
    analogWrite(pwm_B, -u);
  }
}
```

Příloha č. 3: Program integračního regulátoru

```
//////////Vstupy//////////////
byte cidlo = A12; // hodnota regulované veličiny
byte pot = A11; // hodnota žádané veličiny (potenciometr)

//////////Výstupy na ventily - pwmA přidá tlak, pwmB ubere tlak
byte pwm_A = 3; // rychlost
byte dir_A = 12; //směr polarizace (směr otáčení)
byte pwm_B = 11; // rychlost
byte dir_B = 13; //směr polarizace (směr otáčení)

double w, y, u, e;
//y - regulovaná veličina
//w - požadovaná hodnota SP
//u - Akční veličina
//e - regulační odchylka

//I - integrační konstanta
const double I = 16; // nastavení integrační konstanty
double yI = 0; //pomocná integrační proměnná

///////časové proměnné//////
const int frekvence = 50; // vzorkovací frekvence [ms]
int dt; //proměnná pro čas mezi dvěma kroky
unsigned long lasttime; //proměnná pro čas posledního vypočítání
unsigned long now; //proměnná pro čas od spuštění programu

float x; //proměnná pro čas dt v sekundách
float g; //pomocná proměnná pro hlídání hodnoty akční veličiny

const float a = (20. / 51); //převodní konstanta na kPa
const float b = (1. / 51); //převodní konstanta akční veličiny na volty
////////////////////////////////////////////////////////////

void setup() {

  Serial.begin(9600); //Start sériové komunikace
  Serial.println("Zaciname"); // Vypíše "začátek"

  //Nastavení pinů jako výstupních
  pinMode(dir_A, OUTPUT);
  pinMode(dir_B, OUTPUT);
}
////////////////////////////////////////////////////////////
void loop() {

  digitalWrite(dir_A, HIGH); //napětí na kladný pól vstupního ventilu
  digitalWrite(dir_B, HIGH);  //napětí na kladný pól vypouštěcího ventilu

  //načtení senzoru a změna měřítka
  y = map(analogRead(cidlo) - 609, 0, 415, 0, 255);
```

```
  //načtení žádané hodnoty a změna měřítka
  w = map(analogRead(pot), 0, 1023, 0, 255);

  now = millis(); //záznam času (čas od spuštění programu)
  dt = (now - lasttime); // čas od posledního výpočtu regulátoru

  if (frekvence <= dt) //rozhoduje se, jestli už uplynul čas pro vzorek
  {
    e = w - y; //výpočet regulační odchylky
    x = (dt * 0.001); // převedení dt na sekundy
    yI = (yI + e * (x)); //yI pomocná integrační proměnná

    g = 255 / I;
    if (yI > g) // nestane se, překročení hodnoty 255
    {
      yI = g;
    }
    if (yI < -g) // nestane se, překročení hodnoty -255
    {
      yI = -g;
    }
    u = I * yI; //výpočet I-regulátoru

    ///////////výpis hodnot//////////////
    Serial.print(now); // čas
    Serial.print("; ");
    Serial.print(w * a);
    Serial.print("; ");
    Serial.print(u * b);
    Serial.print("; ");
    Serial.println(y * a);

    ///rozhodnutí o zvyšování, nebo snižování tlaku/////
    if (u >= 0)
    {
      analogWrite(pwm_B, 0);
      analogWrite(pwm_A, u);
    }
    else
    {
      analogWrite(pwm_A, 0);
      analogWrite(pwm_B, -u);
    }
    lasttime = now; // aktualizace časového rozhraní
  }
}
```

Příloha č. 4: Program proporcionálně-integračního regulátoru

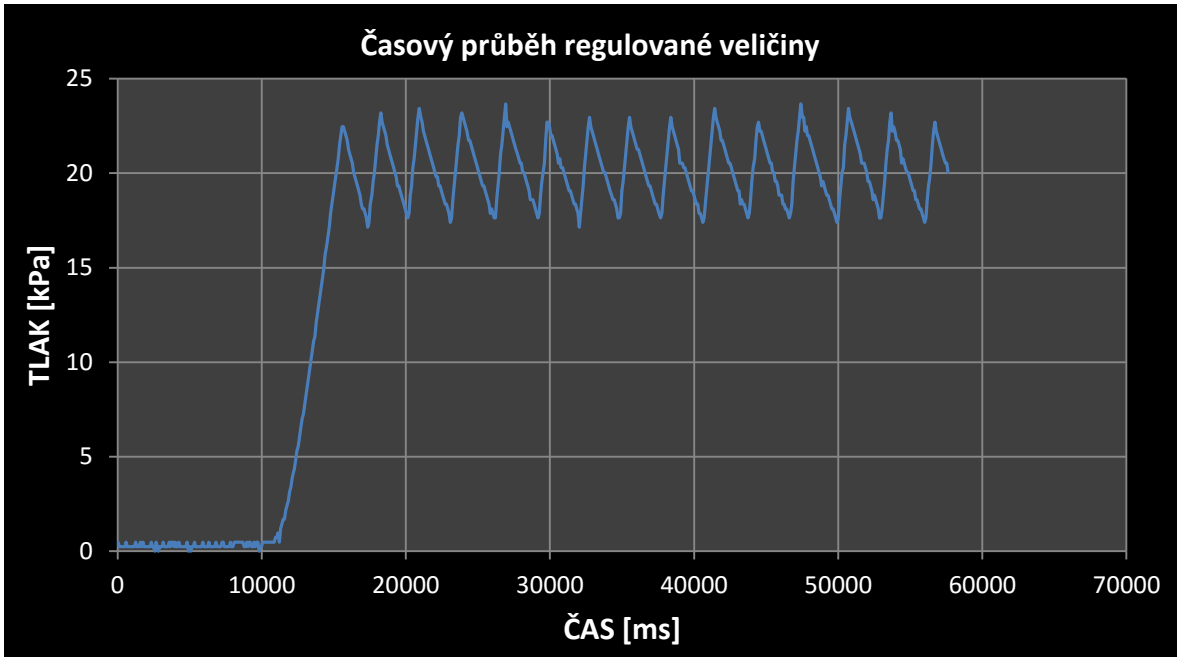
```
//////////Vstupy
byte cidlo = A12; // hodnota regulované veličiny
byte pot = A11; // hodnota žádané veličiny (potenciometr)

//////////Výstupy na ventily - pwmA přidá tlak, pwmB ubere tlak
byte pwm_A = 3; // rychlost
byte dir_A = 12; //směr polarizace (směr otáčení)
byte pwm_B = 11; // rychlost
byte dir_B = 13; //směr polarizace (směr otáčení)

double w, y, u, e;
//y - regulovaná veličina
//w - požadovaná hodnota SP
//u - Akční veličina
//e - regulační odchylka

//P - proporcionální konstanta
//I - integrační konstanta
const double P = 7; //nastavení proporcionální konstanty
const double I = 16; // nastavení integrační konstanty
double yI = 0; //pomocná integrační proměnná

///////sledování času
int frekvence = 50; // vzorkovací frekvence [ms]
int dt; // čas mezi dvěma kroky
unsigned long lasttime; // čas posledního vyhodnocení regulátoru
unsigned long now; // čas od spuštění programu

float x;//čas dt v sekundách
float g; //hlídá rozsah yI aby nepřekročil 255

const float a = (20. / 51); //převodní konstanta na kPa
const float b = (1. / 51); //převodní konstanta akční veličiny na volty
////////////////////////////////////////////////////////////

void setup() {

  Serial.begin(9600); //Start sériové komunikace
  Serial.println("Zaciname"); // Vypíše "začátek"

  //Nastavení pinů jako výstupních
  pinMode(dir_A, OUTPUT);
  pinMode(dir_B, OUTPUT);
}
////////////////////////////////////////////////////////////
void loop() {

  digitalWrite(dir_A, HIGH); // na kladný pól vstupního ventilu
  digitalWrite(dir_B, HIGH);  //napětí na kladný pól vypouštěcího ventilu
```

```
//načtení senzoru a změna měřítka
y = map(analogRead(cidlo) - 609, 0, 415, 0, 255);

//načtení žádané hodnoty a změna měřítka
w = map(analogRead(pot), 0, 1023, 0, 255);

now = millis();  //záznam času
dt = (now - lasttime); // čas od posledního výpočtu regulátoru
x = (dt * 0.001);// převedení dt na sekundy

if (frekvence <= dt) //rozhoduje se jestli už uplynul čas pro vzorek
{
  e = w - y; //výpočet regulační odchylky

  yI = (yI + e * (x)); //yI pomocná integrační proměnná

  g = 255 / I;

  if (yI > g)
  {
    yI = g;
  }

  if (yI < -g)
  {
    yI = -g;
  }

  u = P * e + I * yI; //výpočet I-regulátoru

  if (u > 255)
  {
    u = 255;
  }

  if (u < -255)
  {
    u = -255;
  }
  ///////////výpis hodnot//////////////
  Serial.print(now);
  Serial.print("; ");
  Serial.print(w * a);
  Serial.print("; ");
  Serial.print(u * b);
  Serial.print("; ");
  Serial.println(y * a);

  ///rozhodnutí o zvyšování, nebo snižování tlaku/////
  if (u >= 0)
  {
    analogWrite(pwm_B, 0);
    analogWrite(pwm_A, u);
  }
```

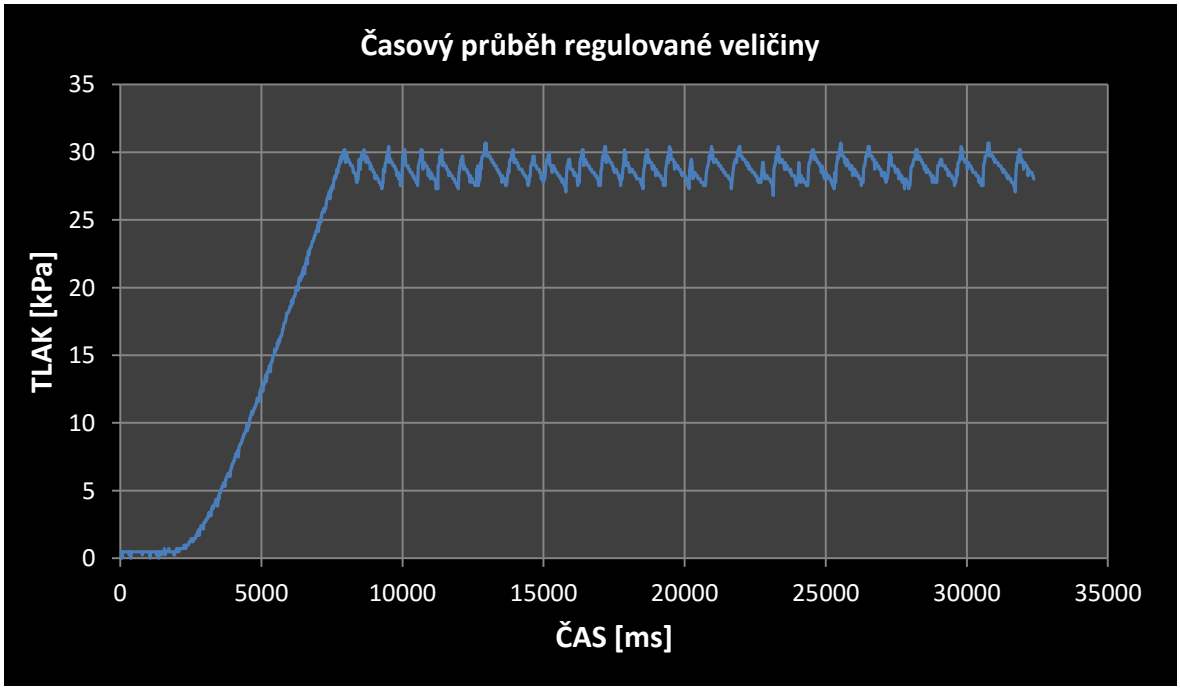
```
      else
      {
        analogWrite(pwm_A, 0);
        analogWrite(pwm_B, -u);
      }
      lasttime = now; // aktualizace časového rozhraní
    }
}
```

Příloha č. 5: Grafy průběhů regulace dvoustavovým regulátorem

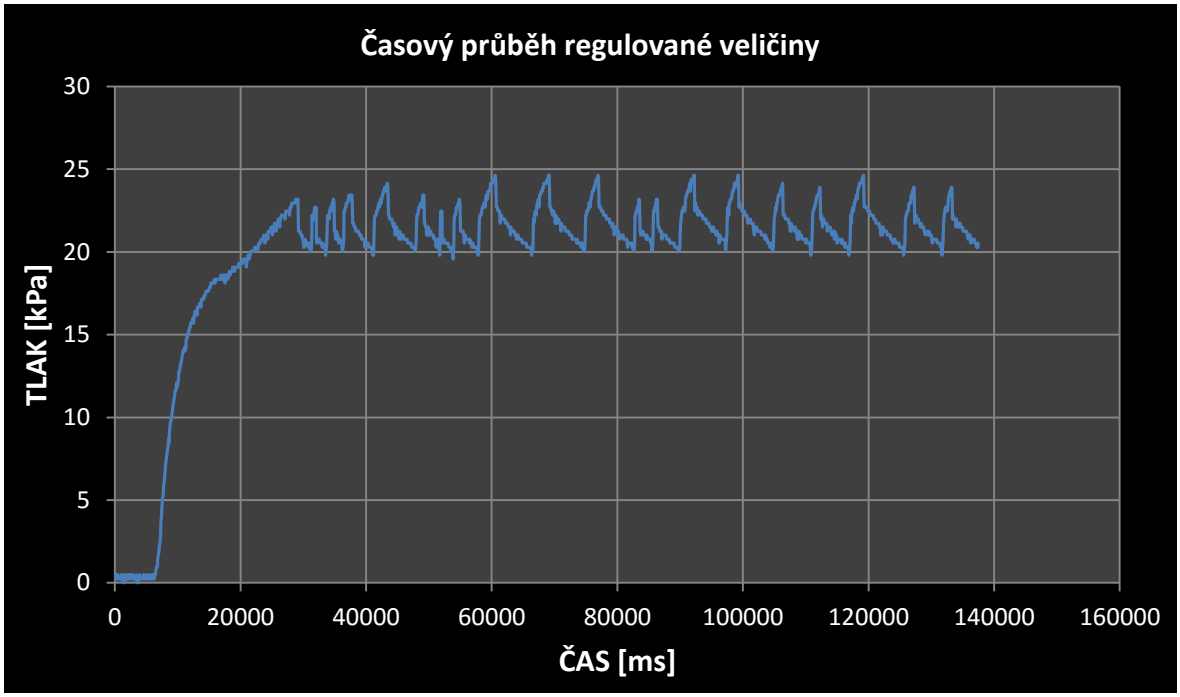
Systémem je tlaková lahev, rozsah hystereze je 5 kPa.



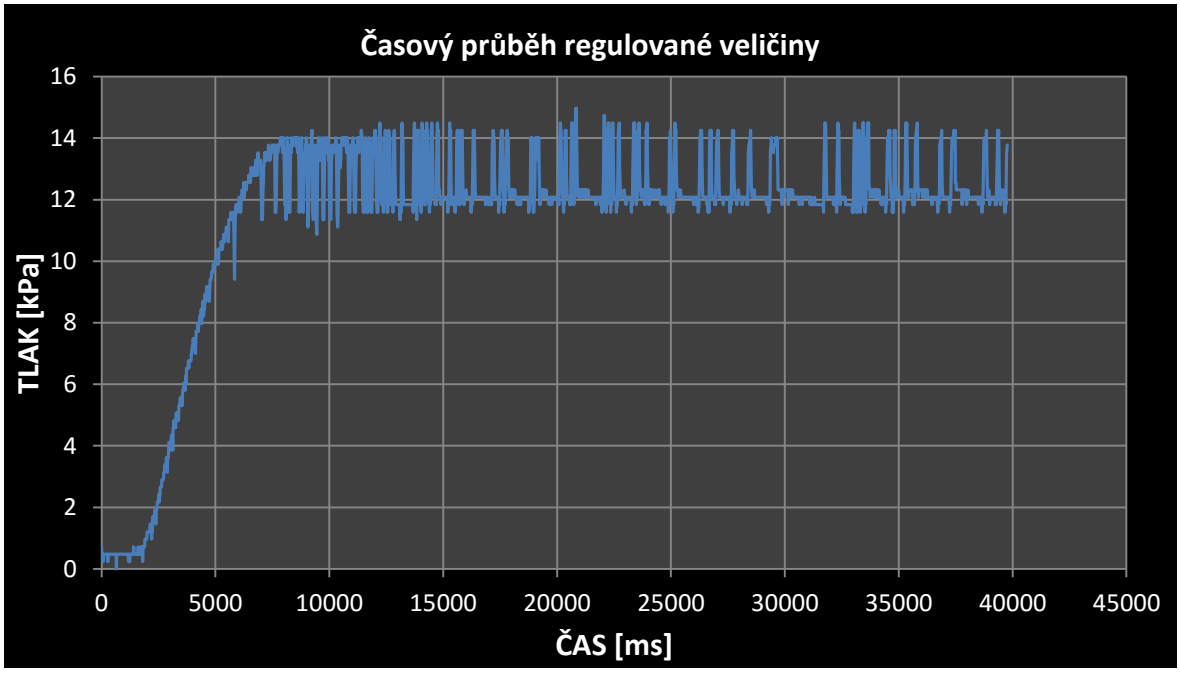
Systémem je tlaková lahev, rozsah hystereze je 2,5 kPa.



Systémem jsou pryžové vaky v sedáku, rozsah hystereze je 4 kPa.

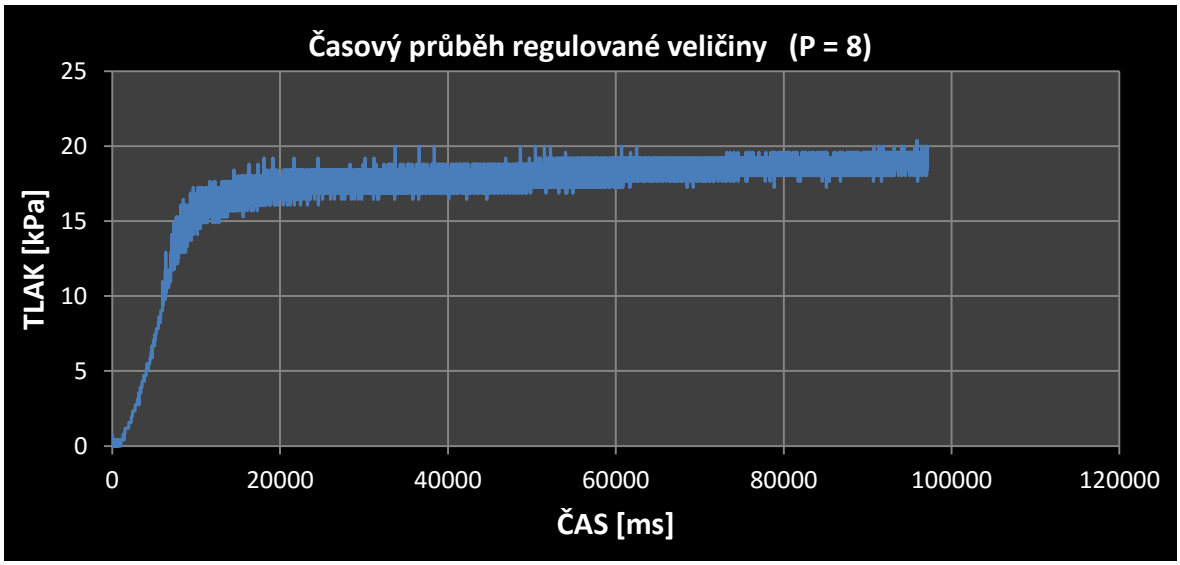


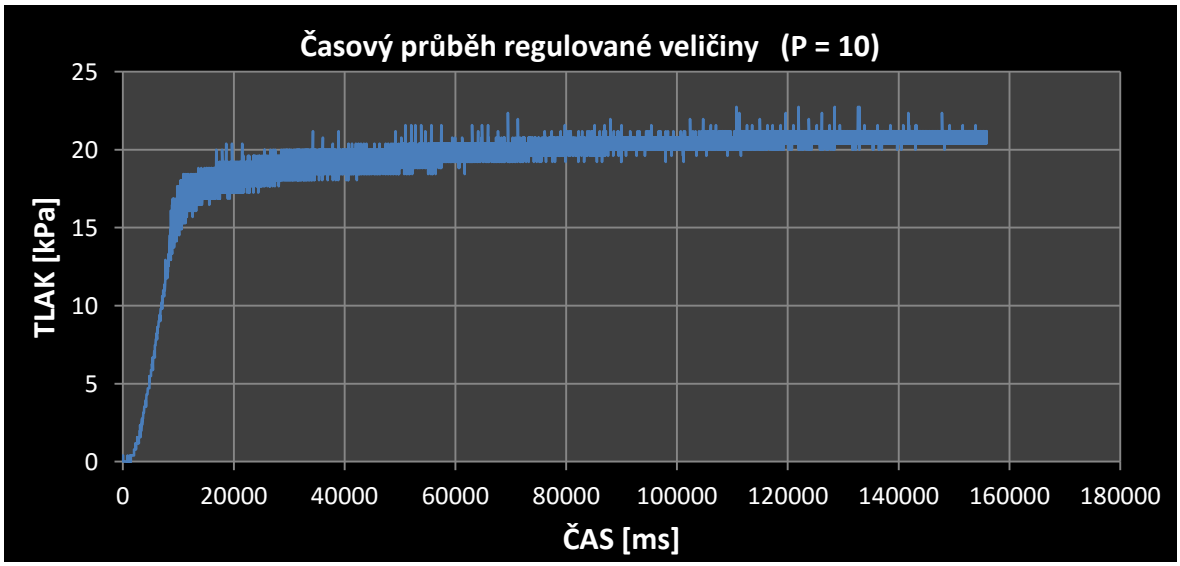
Systémem jsou pryžové vaky v sedáku, rozsah hystereze je 2 kPa.

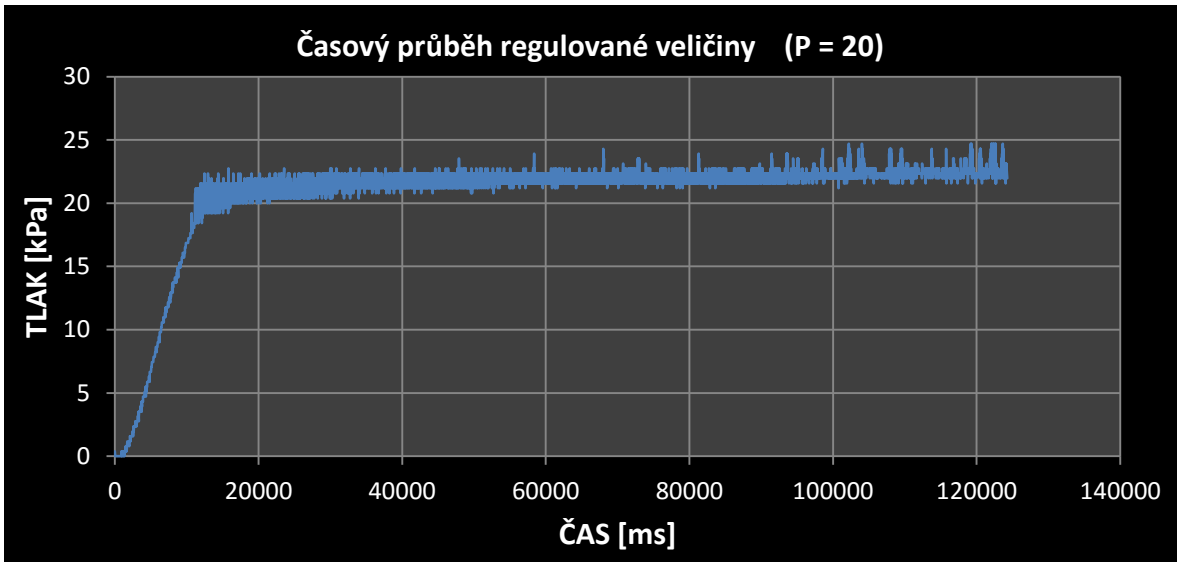


Příloha č. 6: Grafy průběhů regulace proporcionálním regulátorem

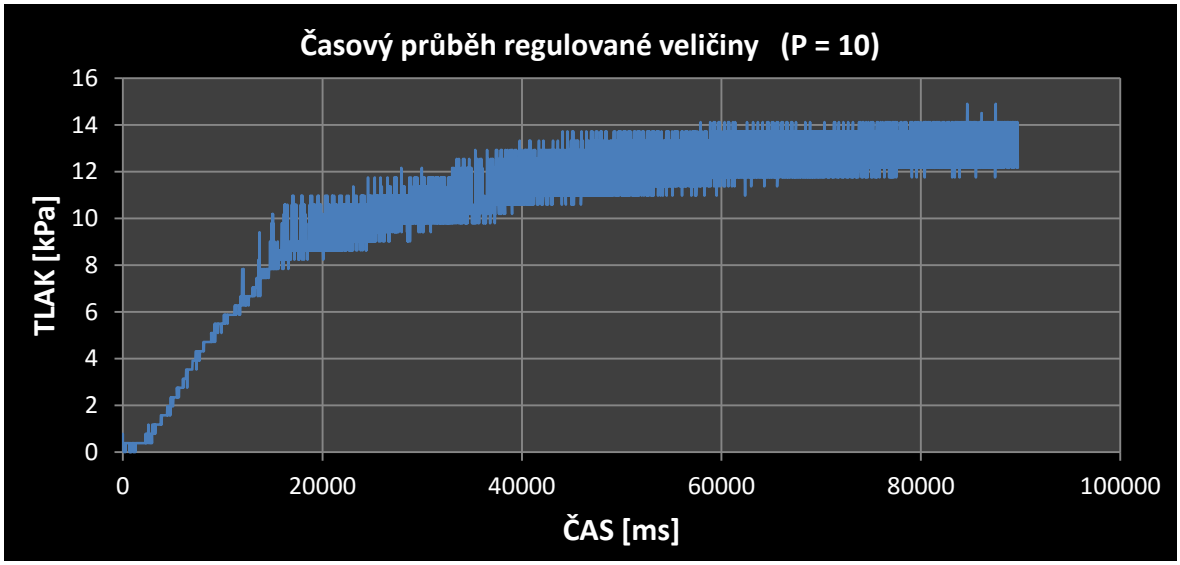
Systémem je tlaková lahev, proporcionální konstanta má hodnoty 8, 10, 20.

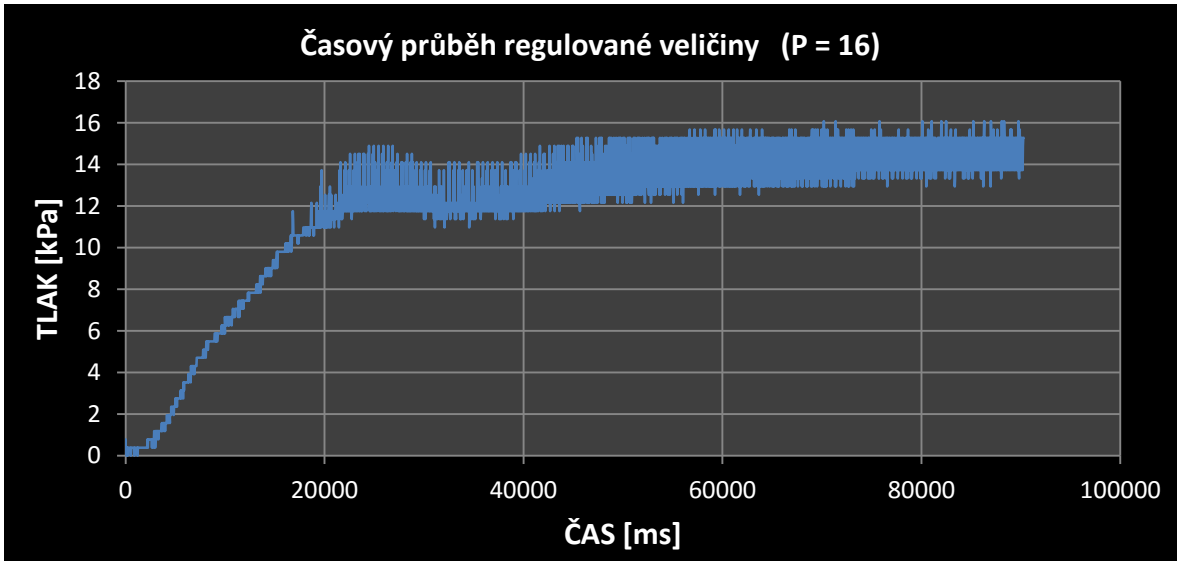


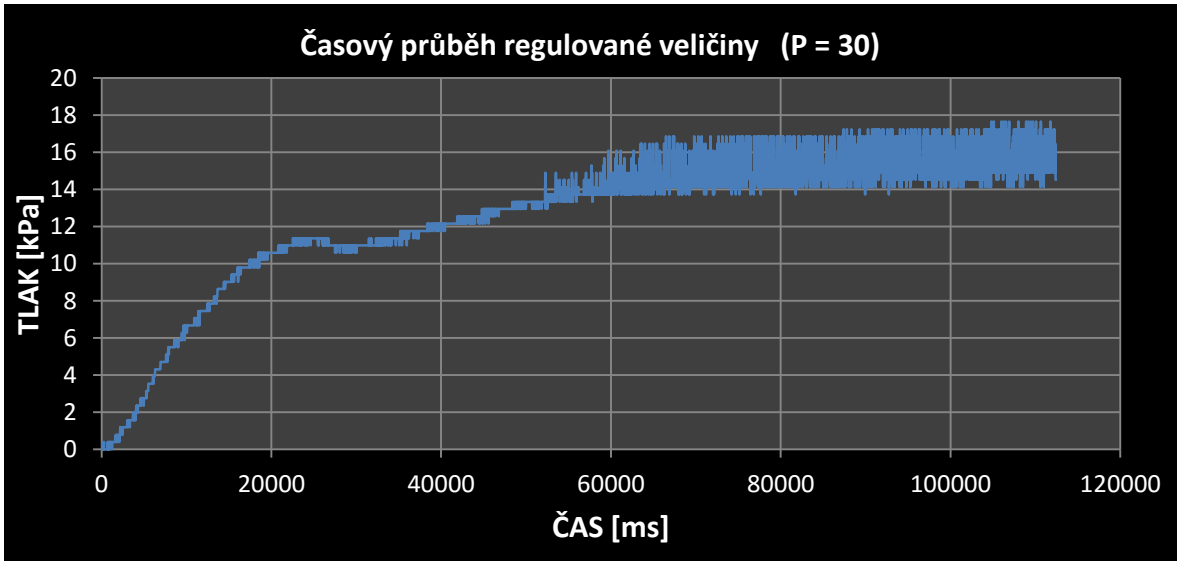




Systémem jsou pryžové vaky v sedáku, proporcionální konstanta má hodnoty 10, 16, 30.

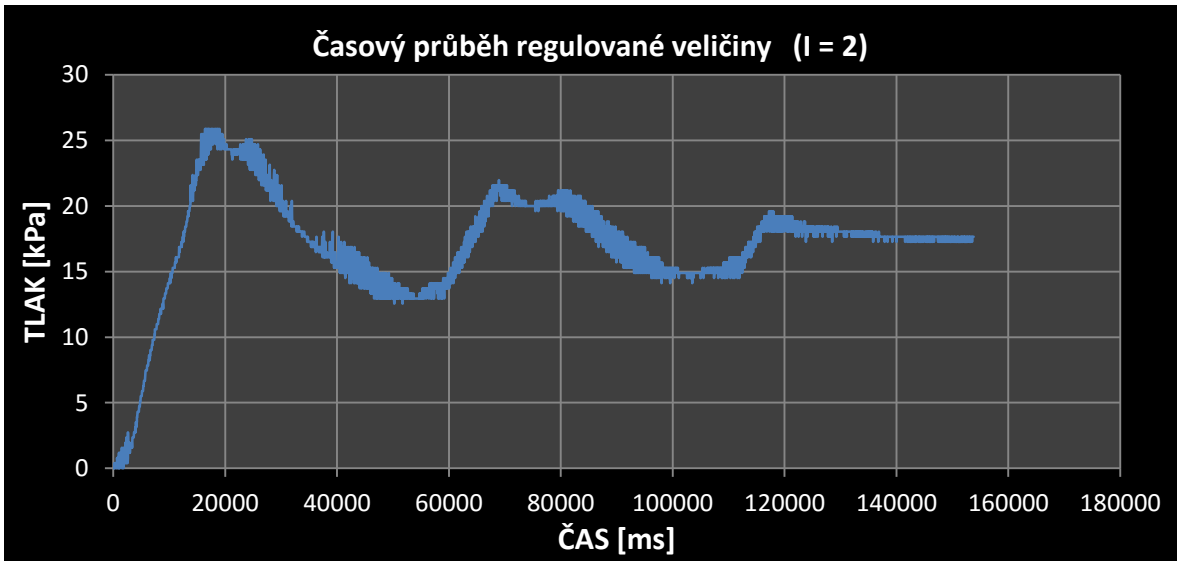


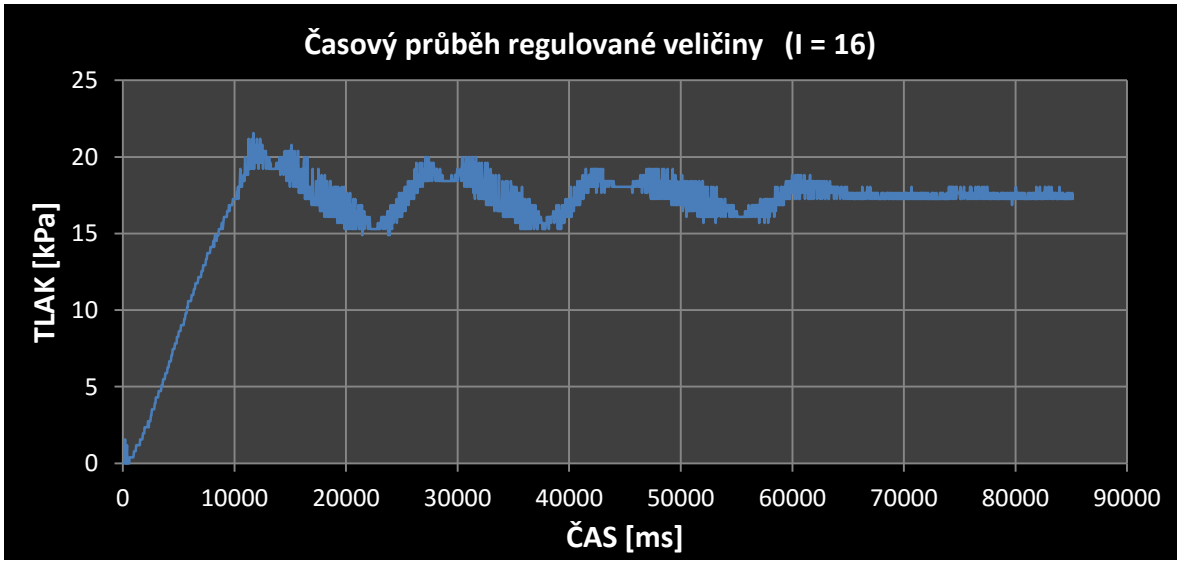


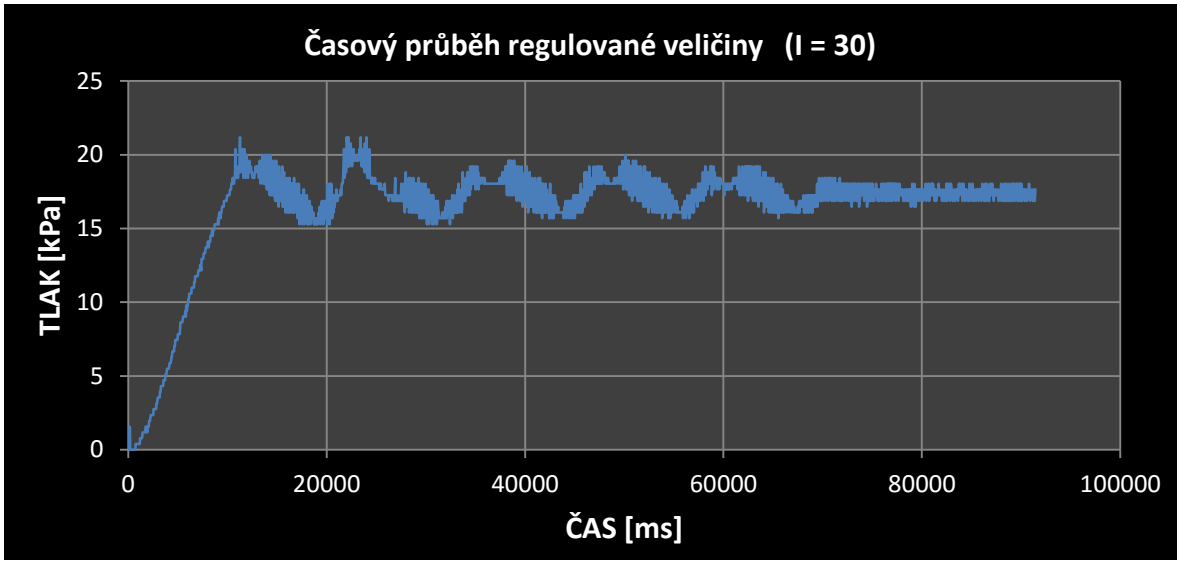


Příloha č. 7: Grafy průběhů regulace integračním regulátorem

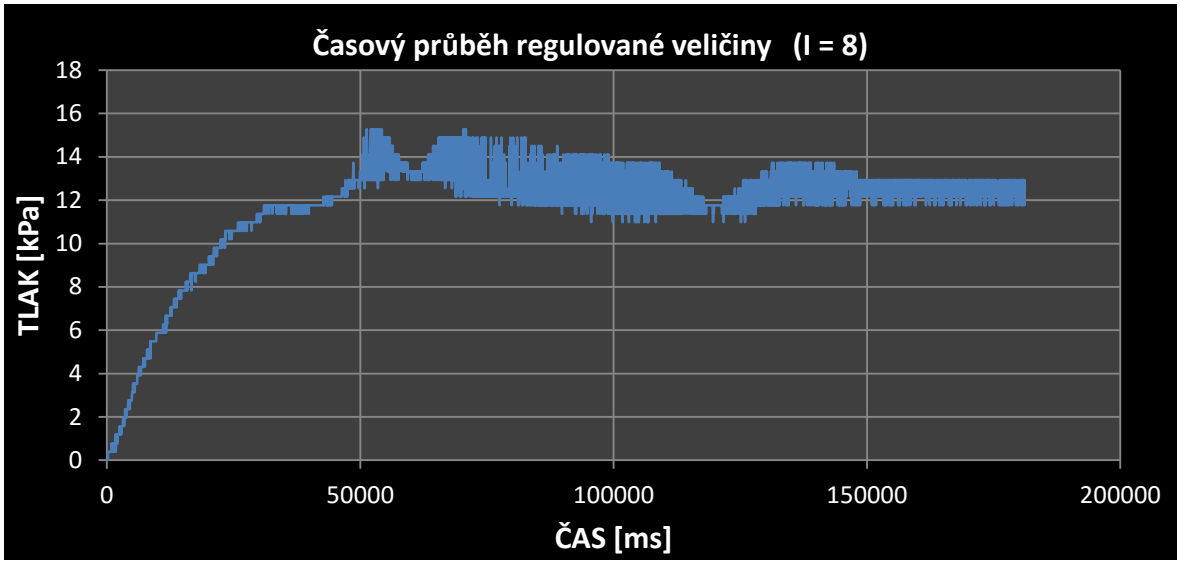
Systémem je tlaková lahev, integrační konstanta má hodnoty 2, 16, 30.

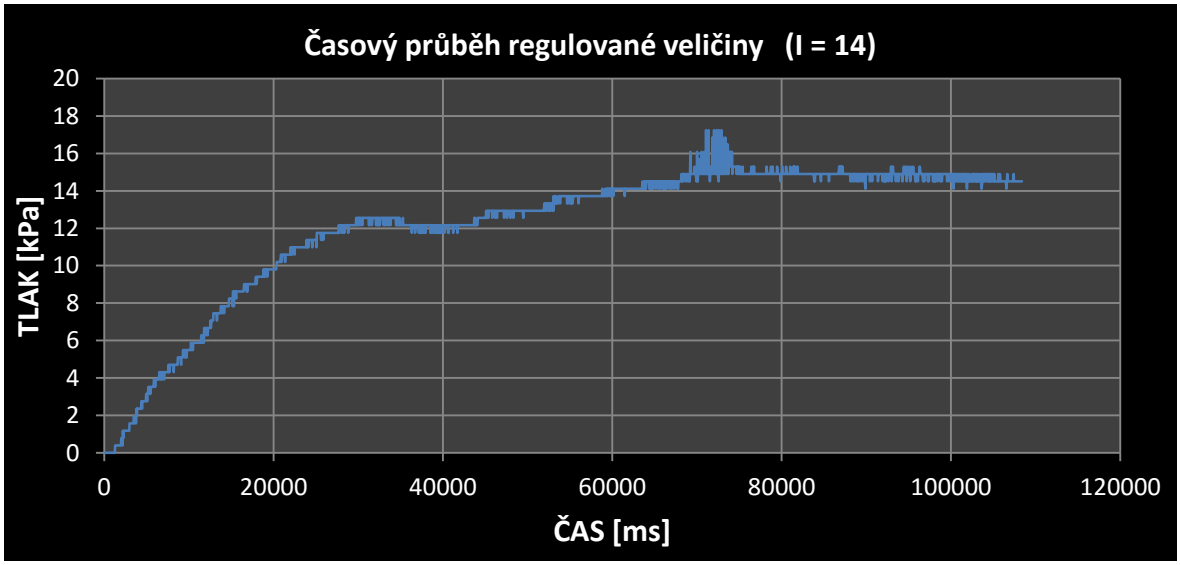


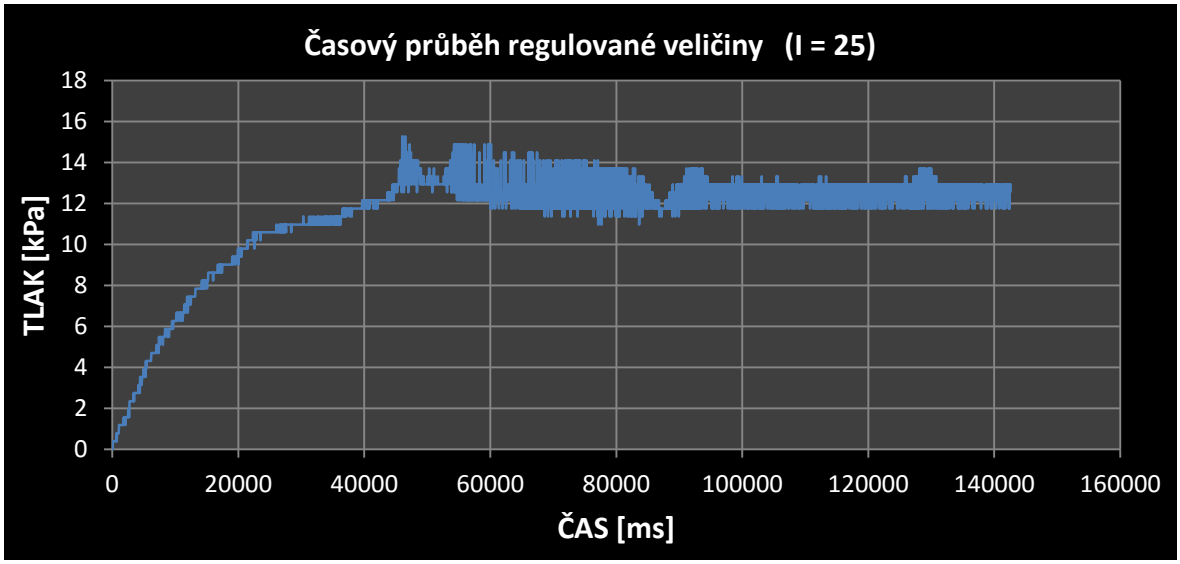




Systémem jsou pryžové vaky v sedáku, integrační konstanta má hodnoty 8, 14, 25.

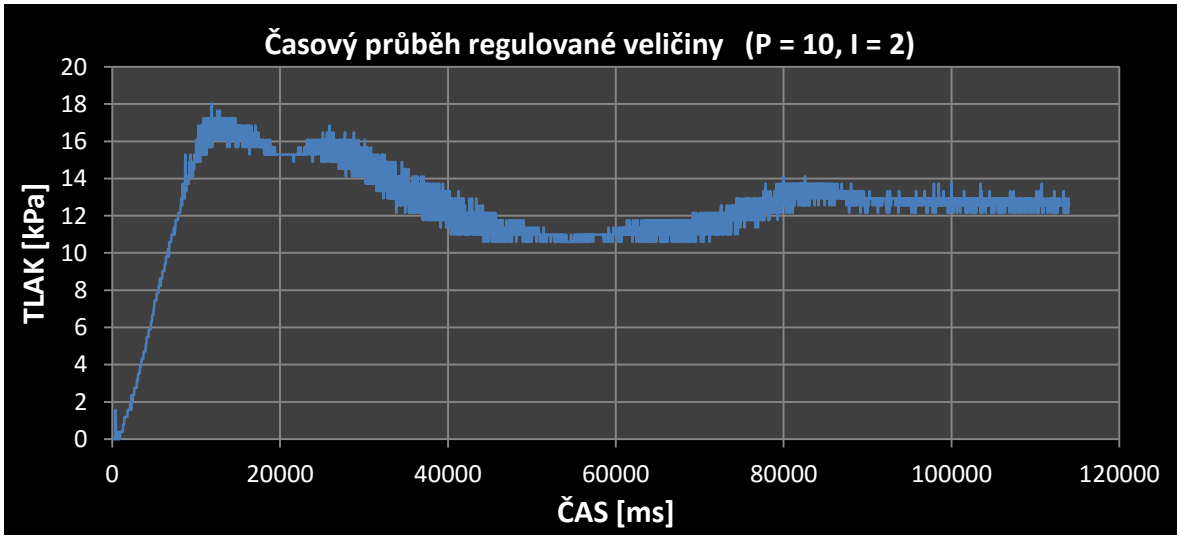


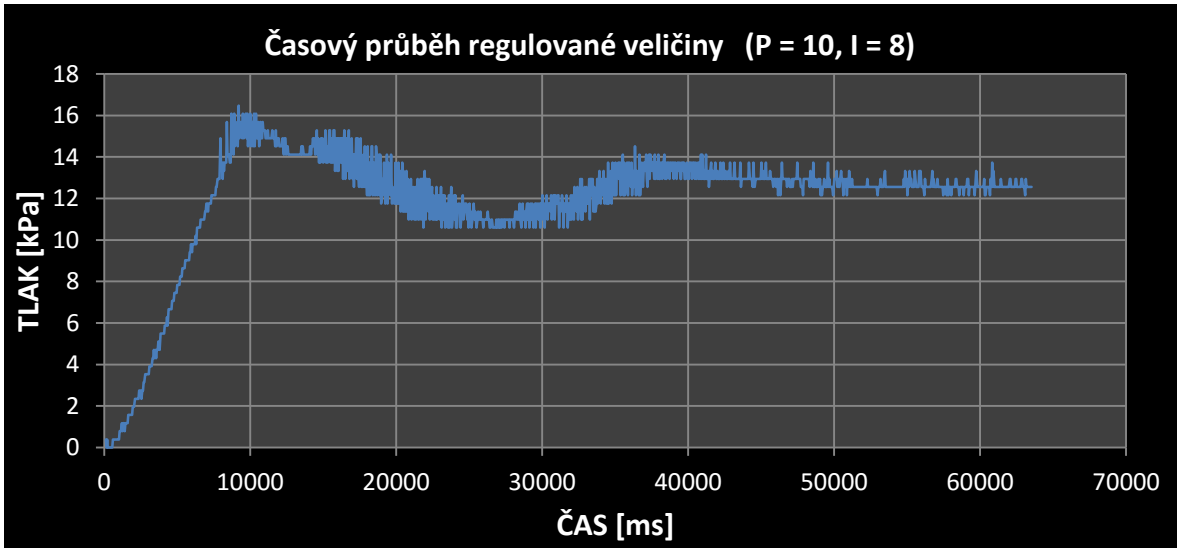


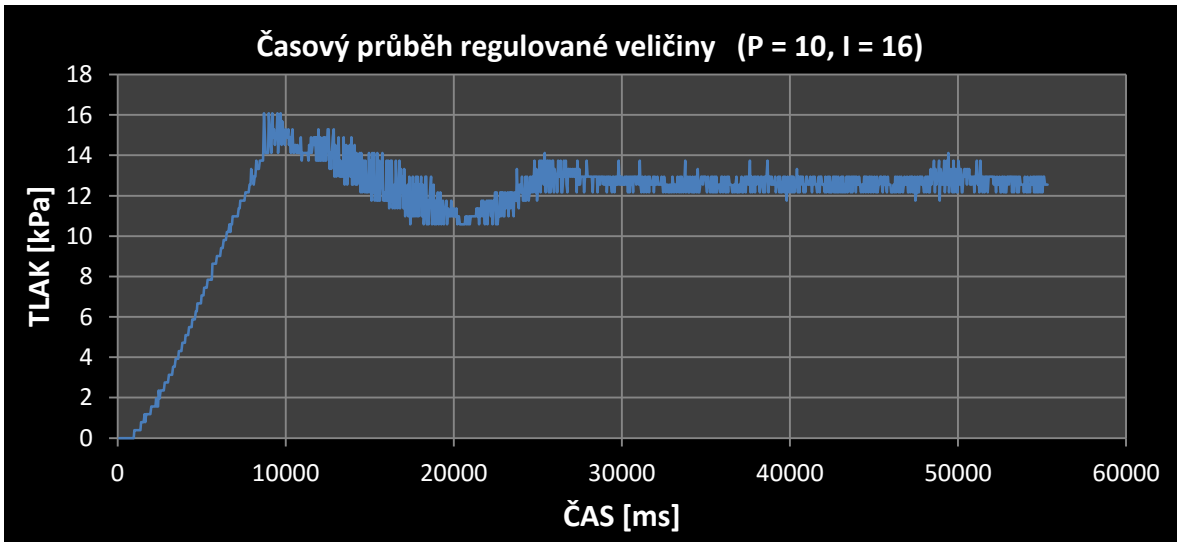


Příloha č. 8: Grafy průběhů regulace proporcionálně-integračním regulátorem

Systémem je tlaková lahev, hodnoty proporcionální a integrační konstanty jsou:
P = 10, I = 2; P = 10, I = 8; P = 10, I = 16;







Systémem jsou pryžové vaky v sedáku, hodnoty proporcionální a integrační konstanty jsou: P = 16, I = 8; P = 16, I = 16; P = 18, I = 8;