TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Řízení pneumatického systému pomocí mikroprocesoru s použitím vývojové desky Arduino

Diplomová práce

Studijní program: N2301 – Strojní inženýrství
Studijní obor: 2301T049 – Výrobní systémy a procesy

Autor práce: **Bc. Vladyslav Myroshnychenko**
Vedoucí práce: Ing. Radek Votrubec, Ph.D.

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechanical Engineering

Control of pneumatic system using Arduino board.

Master thesis

Study programme: N2301 – Mechanical Engineering
Study branch: 2301T049 – Manufacturing Systems and Processes

Author: **Bc. Vladyslav Myroshnychenko**
Supervisor: Ing. Radek Votrubec, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta strojní
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Vladyslav Myroshnychenko**
Osobní číslo:	**S15000509**
Studijní program:	**N2301 Strojní inženýrství**
Studijní obor:	**Výrobní systémy a procesy**
Název tématu:	**Řízení pneumatického systému pomocí mikroprocesoru s použitím vývojové desky Arduino**
Zadávající katedra:	**Katedra výrobních systémů a automatizace**

Zásady pro vypracování:

1. Seznamte se s řídicím obvodem pneumatické pružiny.
2. Na základě použitých čidel a akčních členů proveďte výběr vyhovující řídicí desky Arduino.
3. Vytvořte potřebné převodníky pro připojení akčních členů na vstupy a výstupy mikrokontroléru.
4. Realizujte řídicí obvod pomocí desky Arduino a naprogramujte algoritmus řízení.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **50-60 stran**

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

Seznam odborné literatury:

[1] Začínáme s Arduinem: příručka. In: Snail Instruments: www.hobbyrobot.cz [online]. 2014 [cit. 2015-01-09]. Dostupné z: http://www.snailshop.cz/literatura/1537-zaciname-s-arduinem-prirucka.html
[2] Arduino Learning: Getting Started with Arduino. In: 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, 663 s. ISBN 978-80-247-4116-1.
[4] OLEHLA, M. a NĚMEČEK S. Automatické řízení. Vyd. 1. Liberec: Technická univerzita v Liberci, 2013. ISBN 978-807-3729-721.

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

Datum zadání diplomové práce: **1. listopadu 2016**

Termín odevzdání diplomové práce: **1. února 2018**

prof. Dr. Ing. Petr Lenfeld
děkan

L.S.

Ing. Petr Zelený, Ph.D.
vedoucí katedry

V Liberci dne 1. listopadu 2016

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou 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 diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou 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.

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

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

Datum: 3. 5. 2017

Podpis:

Poděkování

Chtěl bych poděkovat vedoucímu své diplomové práce Ing. Radku Votrubcovi, Ph.D. za poskytnutí možnosti pracovat na zajímavém tématu, kterým se zabývala tato diplomová práce. Děkuju za profesionální vedení, věnování času na konzultacích, vzácné rady a připomínky.

Anotace

Diplomová práce se zabývá realizací procesu regulace pneumatického systému pomocí desky Arduino, pneumatického ventilu a dalších zařízení. Jsou tady popsány nejpoužívanější v dnešní době desky Arduino, odůvodněno proč byla zvolena deska Arduino M0 Pro, charakterizované programovací prostředí Arduino IDE, jeho funkce a možnosti. V další části jsou uvedeny existující druhy pneumatických ventilů, volba potřebného ventilu pro účely této diplomové práce a další zařízení, nezbytné pro sestavení systému jako jsou čidla, kompresor, zdroj napájení atd. Taky jsou zde uvedeny možné způsoby realizace procesu spojité regulace pomocí různých druhů regulátorů. V praktické části je popsány způsob zapojení všech prvků, sestaveny elektrický a pneumatický obvod systému a algoritmus regulaci, který je zrealizován v podobě programu.

Klíčová slova

Deska Arduino, Arduino IDE, pneumatický solenoidový ventil, spojitá regulace, programování.

Annotation

The target of master thesis is implementation of pneumatic system's control process using Arduino board, pneumatic valve and additional equipment. There has been described most used nowadays Arduino boards, given reasons, why the particular board Arduino M0 Pro was chosen, also there is a description of Arduino IDE development environment, its functions and capabilities. In the next part of master thesis there is a description of existing pneumatic valves' types, choice of the valve required by the specific system and additional equipment needed in order to assemble the complete system such as sensors, compressor, power supply etc. Also there is a mention about existing ways of continuous control process's implementation using different types of regulators. Practical part of master thesis describes the way components connected to each other, also there has been created electrical and pneumatic circuits and control algorithm, implemented in form of a program.

Key Words

Arduino board, Arduino IDE, pneumatic solenoid valve, continuous control, programming.

Obsah

Seznam použitých zkratek

IDE	Integrované vývojové prostředí (Integrated Development Environment)
USB	Univerzální sériová sběrnice (Universal Serial Bus)
OS	Operační systém (Operational System)
3D	3-rozměrný (3-Dimensional)
I/O	Vstupní/Výstupní (Input/Output)
LED	Dioda emitující světlo (Light-Emitting Diode)
AC	Střídavý proud (Alternating Current)
DC	Stejnosměrný proud (Direct Current)
Wi-Fi	Bezdrátové připojení k sítí (Wireless Fidelity)
PCB	Deska plošných spojů (Printed Circuit Board)
PWM	Pulsně šířková modulace (Pulse Width Modulation)
SRAM	Statická operační paměť (Static Random-Access Memory)
NO	Ve výchozím nastavení otevřený (Normally Opened)
NC	Ve výchozím nastavení uzavřený (Normally Closed)
P-regulátor	Proporcionální regulátor (Proportional)
I-regulátor	Integrační regulátor (Integral)
D-regulátor	Derivační regulátor (Derivative)
PD-regulátor	Proporcionální regulátor (Proportional-Derivative)
PI-regulátor	Proporcionální regulátor (Proportional-Integral)
PID-regulátor	Proporcionálně-integračně-derivační regulátor (Proportional-Integral-Derivative)
PLX-DAQ	Nástroj pro získání dat do Excelu od výrobce Parallax (Parallax Data Acquisition Tool)

1. Úvod

Pneumatické systémy jsou součástí velkého množství různých zařízení, se kterými se často setkáváme v reálném životě. Tyto systémy jsou řízeny pomocí ventilů a dalších zařízení s použitím stlačeného vzduchu jako zdroje potenciální energie.

V dnešní době jsou široce využívány, zejména v průmyslových odvětvích. Příklady použití pneumatických systémů můžou být automatizované výrobní linky, řízení automaticky se otevírajících dveří, vzduchové brzdy na autobusy a kamiony, cvičební stroje, přistávací zařízení letadel atd. Dalšími příklady aplikací pneumatických systémů jsou zařízení vytvořené na katedře KSA v rámci bakalářských a diplomových prací. Jsou to systémy pro regulaci tuhosti autosedačky a ortopedické matrace, popsané v pracích [14] a [15]. Pneumatické systémy mají mnoho výhod, mezi kterými jsou vysoká účinnost a spolehlivost, jednoduchý design, bezpečnost, šetrnost k životnímu prostředí a hospodárnost. Jsou typicky flexibilnější, méně nákladné a spolehlivější než mnoho typů elektromotorů.

K řízení pneumatických systémů se dost často používá elektronické zařízení, u kterého je hlavním prvkem mikroprocesor. Deska Arduino je takovým mikroprocesorem s digitálními a analogovými vstupy a výstupy, na které lze připojit různá zařízení a komunikovat s nimi prostřednictvím posílání signálů z desky a na desku. To znamená, že deska Arduino splňuje úkol řídící jednotky pro systém.

Výhodami Arduina, v porovnání s jinými mikroprocesory, jsou nízké ceny desek, oficiální software, sloužící k programování desky, nabízený zdarma na webové stránce výrobce a dobře naladěná uživatelská podpora ve formě dokumentace programovacího jazyku a velkého množství hotových projektů volně dostupných k prohlížení na těch samých na webových stránkách výrobce. Vzhledem ke všem těmto výhodám vývojový kit Arduino může být jednoduchým, avšak efektivním nástrojem pro realizaci úlohy řízení pneumatického systému.

1.1. Cíl práce

K úlohám této práce patří výběr vyhovujících komponent pro sestavení systému, jako jsou řídící jednotka Arduino, pneumatické ventily, čidla atd., seznámení s jejich

parametry a možnostmi. Dále vytvoření samotného systému s použitím všech vybraných prvků, k čemuž patří sestavení pneumatického a elektrického obvodu, potřebných převodníků, regulátoru a programu pro realizaci algoritmu řízení. Globálním cílem práce je vytvoření výukového modelu, který by sloužil jako jednoduchý příklad činnosti pneumatického systému.

2. Arduino

Arduino je volně dostupná vývojová platforma zahrnující hardwarové a softwarové komponenty. Běžně se skládá z desky a programovacího prostředí s názvem Arduino IDE, který se používá k psaní kódu a jeho následnému nahrání na desku. Chování desky Arduino je ovlivňováno zasláním instrukcí na mikroprocesor, který je umístěn na desce. Kvůli tomu softwaru Arduino IDE ještě se říká "nahrávací software" (nahrává příkazy na desku). K tomuto nahrávání není potřeba žádného dodatečného hardwaru. Komunikace mezi Arduino IDE a mikroprocesorem probíhá jednoduše pomocí USB kabelu. Zvláštností desek Arduino je, že jsou schopné číst analogové a digitální vstupní signály z různých dodatečných zařízení a přeměnit ty data na výstupní signály. Desky Arduino jsou v dnešní době velmi populárním nástrojem pro různé typy projektů hlavně z důvodu jednoduchosti použití a relativně malé ceny v porovnání s jinými mikroprocesory a taky kvůli dostupnosti programovacího prostředí Arduino IDE pro každý dnes používaný operační systém jako jsou Windows, Mac OS, Linux.

2.1. Základní typy desek Arduino

Na webových stránkách Arduina je vidět celou řadu desek, které lze použit pro svůj projekt. Produkty jsou rozděleny do několika skupin:

- produkty pro začátečníky
- produkty s rozšířenými funkcemi
- produkty pro 3D tisk
- produkty, které se uplatňují v oblasti Internet of Things
- další skupiny

Desky se mohou lišit jedna od druhé použitým uvnitř desky vestavěným mikroprocesorem, vstupním napětím na desce, operačním napětím, počtem analogových a digitálních vstupních a výstupních pinů, množstvím flash-paměti, spotřebou energie, frekvencí, rozměry a dalšími charakteristikami.

Co se týká mikroprocesorů, skoro všechny desky mají ho od výrobce Atmel. Pouze deska Arduino 101 má mikrořadič od výrobce Intel. Doporučené vstupní napětí

na většině desek je 7 až 12 voltů s několika výjimkami jako u desek Arduino Yún, Arduino M0, Arduino Pro, Arduino M0 Pro a pracovní napětí může být buď 5 nebo 3,3 voltů. Co se týká digitálních a analogových pinu, jejich počet na různých deskách může být úplně různý, platí jenom, že na všech deskách počet digitálních pinů je větší než analogových. Digitální piny jsou konfigurovatelné, totiž mohou být buď vstupními nebo výstupními. Analogové piny jsou většinou pouze vstupní a jenom několik desek ještě k tomu poskytují jeden nebo dva analogových výstupů. K těmto deskám patří Arduino Due, Arduino Zero, Arduino M0, Arduino M0 Pro. Taky důležitou charakteristikou je flash-paměť. Nedostatek flash-paměti může omezovat rozměr kódu, použitého v projektu. Jinak desky vcelku mají 32 nebo 256 kB flash-paměti, a pouze deska Arduino Due má 512 kB. Dalšími charakteristikami, kterými se liší desky, jsou frekvence, spotřeba energie a fyzické rozměry, které také mohou být zcela různé, a všechny jsou popsány v Tabulce 1, kde také jsou uvedeny všechny charakteristiky pro běžně používané v dnešní době desky.

Tabulka 1: Technické charakteristiky nejpoužívanějších desek Arduino

Deska	***Arduino Uno***	***Arduino Leonardo***	***Arduino Mega 2560***
Mikroprocesor	ATmega328P	ATmega32u4	ATmega2560
Operační napětí, V	5	5	5
Vstupní napětí, V	7 až 12	7 až 12	7 až 12
Počet digitálních I/O pinů	14	20	54
Počet analogových vstupních pinů	6	12	16
Stejnosměrný proud, mA	20	40	20
Flash-paměť, kB	32	32	256

Frekvence, MHz	16	16	48
Rozměry desky, mm	68.6 x 53.4	68.6 x 53.4	101.52 x 53.4
Deska	***Arduino Due***	***Arduino Mega AD***	***Arduino Pro***
Mikroprocesor	AT91SAM3X8E	ATmega2560	ATmega328
Operační napětí, V	3,3	5	5
Vstupní napětí, V	7 až 12	7 až 12	5 až 12
Počet digitálních I/O pinů	54	54	14
Počet analogových vstupních pinů	12+2 výstupy	16	6
Stejnosměrný proud, mA	130	40	40
Flash-paměť, kB	512	256	32
Frekvence, MHz	84	16	16
Rozměry desky, mm	101.52 x 53.4	101.52 x 53.4	52 x 53.4
Deska	***Arduino Yún***	***Arduino Ethetnet***	***Arduino Zero***
Mikroprocesor	ATmega32u4	ATmega328	ATSAMD21G18
Operační napětí, V	5	5	3,3
Vstupní napětí, V	5	7 až 12	7 až 12
Počet digitálních I/O pinů	20	14	20
Počet analogových vstupních pinů	12	6	6+1 výstup

Stejnosměrný proud, mA	40	40	7
Flash-paměť, kB	32	32	256
Frekvence, MHz	16	16	48
Rozměry desky, mm	68.6 x 53.4	68.6 x 53.4	68.6 x 53.4
Deska	***Arduino 101***	***Arduino M0***	***Arduino M0 Pro***
Mikroprocesor	Intel Curie	ATSAMD21G18	ATSAMD21G18
Operační napětí, V	3,3	3,3	3,3
Vstupní napětí, V	7 až 12	5 až 15	5 až 15
Počet digitálních I/O pinů	14	20	20
Počet analogových vstupních pinů	6	6+1 výstup	6+1 výstup
Stejnosměrný proud, mA	20	7	44
Flash-paměť, kB	196	256	256
Frekvence, MHz	32	48	48
Rozměry desky, mm	68.6 x 53.4	68.6 x 53.4	68.6 x 53.4

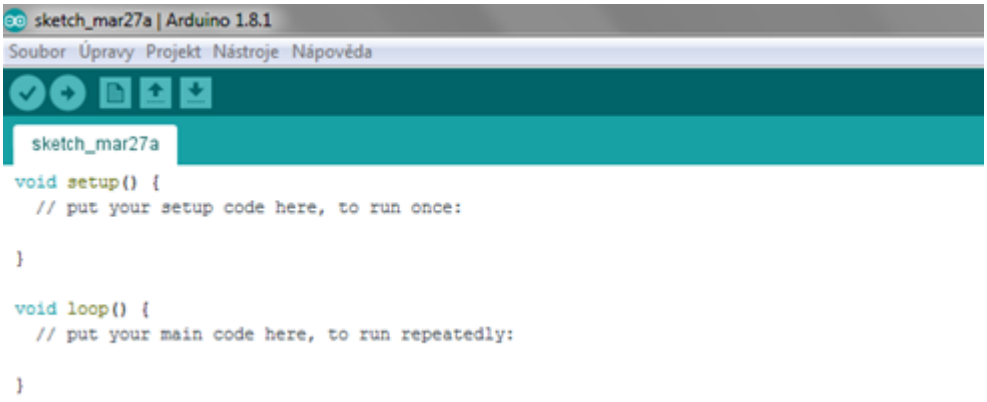
Společnou vlastností desek je, že všechny mohou být naprogramovány pomocí vývojového prostředí Arduino IDE a taky, že mají víceméně stejně prvky umístěné na nich. Dál popsány prvky, společné pro skoro všechny typy desek:

- USB-port: slouží k napájení desky
- Barrel Jack: pro externí zdroj napájení
- Reset: slouží pro restartování v případě, že uživatel chce spustit program ze začátku

- 3.3V: dodává napětí velikosti 3.3 V
- 5V: dodává napětí velikosti 5 V
- GND: uzemnění
- Vin: taky lze použit pro externí zdroj napájení
- Digitální piny: lze nakonfigurovat jako vstupní nebo výstupní
- Analogové piny: tyto piny mohou přečíst signál z analogových zařízení a přeměnit ten signál v digitální podobu, která může být zpracována mikroprocesorem
- Mikroprocesor: může se považovat za mozek desky. Integrovaný obvod na každé desce může být lehce odlišný, ale všechny desky mají v sobě mikroprocesor od výrobce Atmel
- LED-indikátor: rozsvítí se, když Arduino je napájen nějakým zdrojem, totiž napovídá, bud' je Arduino napájeno nebo je něco špatně s připojením.

2.2. Instalace a popis programovacího prostředí Arduino IDE

Dalším krokem k osvojení Arduina je instalace softwaru, nabízeného od Arduina, Arduino IDE. V tomto softwaru se píšou programy, které pak ovlivňují chování desky. Kód psány v prostředí Arduino IDE je v podstatě půjčen od programovacího jazyku C a lidí, kteří mají zkušenosti s psaním programu v populárních v dnešní době jazycích, jako jsou C, C++, Java neměli by mít problém se sestavením jednoduchých programu v Arduino IDE pro svoje projekty. Arduino IDE, na rozdíl od samotné desky, je nabízen zdarma na webových stránkách Arduina a běží na všech operačních systémech používaných dnes. Po instalaci softwaru se spustí okno programovacího prostředí a je možné uvidět velmi jednoduché rozhrání s jen několika nástroji, záložkami a polem pro samotný kód.



Obrázek 1: Zobrazení vývojového prostředí Arduino IDE při nastartování

Pro začátek komunikace mezi Arduino IDE a deskou je třeba v záložce ”Nástroje” zvolit název používané desky, aby při spouštění programu nedošlo k žádným chybám. Taky je potřeba zvolit sériový port pro Arduino desku, což lze udělat taky v záložce ”Nástroje - Port”. Když tyto dva kroky jsou splněny, je možné začít psát program. Pod záložkami umístěno šest tlačítek. První tlačítko ”Ověřit” se používá pro účely kompilace programu, totiž ověření zda program neobsahuje žádné chyby a může být spuštěn. Další tlačítko ”Nahrát” slouží pro spuštění programu nebo jinými slovy pro začátek komunikace mezi deskou a softwarem. Pokud program je sestavený správně, deska na základě instrukcí udělá určité činnosti. Další tlačítko ”Nový” slouží pro vytvoření nové skici pro napsání programu. Tlačítko ”Otevřít” je určené k otevření již existujícího programu, uloženého v paměti a tlačítko ”Uložit” uloží poslední změny v programu, aby se data neztratily při třeba nečekaném zavření prostředí Arduino IDE. No a poslední tlačítko má název ”Sériový monitor” a slouží k obdržení sériových dat z desky nebo jejich zaslání na desku. Sériový monitor dovoluje uživatelovi jednoduše rozumět co se děje a jakých výsledků momentálně dosáhl.

2.3. Nejčastěji používané prvky v programech

Program napsaný v Arduino IDE jmenuje se “sketch”. Všechny “sketchy” mají podobnou struktura v nezávislosti na tom, jaké instrukce jsou uvnitř programu.

setup() a loop()

Struktura “sketchu” se skládá z dvou zásadních funkcí *setup()* a *loop()*. Před těmito funkcemi *setup()* a *loop()* se obvykle provádí definování proměnných. Funkce

setup() spustí se jenom jednou na začátku programu a uvnitř ní se definují režimy pinů, knihovny a všechno co pak bude použité v další části programu. Funkce *loop()* vykonává určité činnosti, které po ní chce uživatel, a, na rozdíl od funkce *setup()*, dělá to opakovaně.

```
// definování proměnných
void setup() {
      // příkazy, které se provedou jenom jednou
}
void loop() {
      // příkazy, které se stále opakují
}
```

Programy se skládají s příkazů, které vykonávají určité činnosti. Na konci každého z nich se píše středník, který označuje konec tohoto příkazu. Příkazy, které se píšou uvnitř nějaké určité funkce, se dávají do složených závorek. Na konci funkce, na rozdíl od příkazů, středník se nepoužívá.

Text, psány za dvěma lomítky Arduino IDE nevnímá. Tomu se říká "komentáře". Komentáře se píšou, aby program byl přehledný a uživatel mohl si jednoduše pamatovat co se děje v každém úseku kódu, ale tyto komentáře nemají žádný vliv na průběh programu. Dva lomítka dělají komentářem pouze ten řádek, před kterým byly použity. Pokud je potřeba okomentovat několik řádku, používá se na začátku komentáře "/*" a na konci " */".

```
/*
text komentáře
text komentáře
text komentáře
*/
```

Programovací jazyk v Arduino IDE má všechny atributy jazyku C, totiž taky má datové typy, operátory, konstanty, cykly, podmínky, funkce atd.

Datové typy

byte

Tento typ je určen pro celá čísla s rozsahem hodnot od 0 do 255. Každá proměnná typu *byte* zabírá 1 byte paměti.

```
byte promenna = 254;
```

int

Určen pro celá čísla s rozsahem hodnot od -32 768 do 32 767. Každá proměnná typu *int* zabírá 2 bytů paměti.

```
int promenna = 10456;
```

long

Určen pro celá čísla s rozsahem hodnot od -2 147 483 648 do 2 147 483 647. Každá proměnná typu *long* zabírá 4 bytů paměti.

```
long promenna = 1000123456;
```

unsigned long

Určen pro celá čísla s rozsahem hodnot od -2 147 483 648 do 2 147 483 647. Každá proměnná typu *unsigned long* zabírá, stejně jako u typu long, 4 bytů paměti. Rozdílem je, že typ *unsigned long* má v svém rozsahu pouze kladné hodnoty od 0 do 4 294 967 295.

```
unsigned long promenna = 3000123456;
```

float

Určen pro čísla s desetinnými místy s rozsahem hodnot od -3.4028235E+38 do 3.4028235E+38. Každá proměnná typu *float* zabírá 4 bytů paměti.

```
float promenna = 8.53664;
```

double

Určen pro čísla s desetinnými místy většinou s rozsahem hodnot stejným jako u typu *float*, ale při použití některých desek může mít až 8 bytů rozlišení.

```
double promenna = 9.2442;
```

boolean

Tento typ může mít pouze dvě hodnoty *true* (1) nebo *false* (0). Každá proměnná typu boolean zabírá 1 byte paměti.

```
boolean promenna = true;
```

Operátory

Nejčastěji používanými operátory jsou aritmetické, porovnávací a logické.

Aritmetické operátory

Slouží pro vykonávání matematických operací. Příklady: ” = ”, “ + ”, “ - “, “ * ”, “ / ” atd.

Porovnávací operátory

Slouží pro porovnávání hodnot a proměnných. Příklady: ” > ”, “ < ”, “ == “ (rovná se), “ >= ”, “ <= ”, “ != ” (nerovná se) atd. Rozdíl mezi operátory “ = “ a “ == “ je v tom, že “ = “ označuje přiřazení hodnoty k proměnné a operátor “ == “ ověřuje, zda dvě hodnoty jsou stejné.

Logické operátory

Slouží pro logické operace. Příklady: ” operand && operand ” (logický operátor AND), “ operand || operand ” (logický operátor OR), “ !operand” (logický operátor NOT) atd. Výraz s logickým operátorem AND bude pravdivý pouze v případě, že oba dva (nebo více) operandů jsou pravdivé. Výraz s logickým operátorem OR bude pravdivý pouze v případě, že alespoň jeden z dvou (nebo více) operandů je pravdivý. Výraz s logickým operátorem NOT bude pravdivý pouze v případě, jestli operand je nepravdivý.

Konstanty

Nejčastěji používanými konstanty jsou *HIGH*, *LOW*, *INPUT*, *OUTPUT*.

HIGH

Při použití jako parametru funkce *digitalWrite()*, pošle digitální signál “1” na digitální pin.

```
digitalWrite(promennaProPin, HIGH);
```

LOW

Při použití jako parametru funkce *digitalWrite()*, pošle digitální signál “0” na digitální pin.

```
digitalWrite(promennaProPin, LOW;
```

INPUT

Při použití jako parametru funkce *pinMode()*, nastaví digitální pin jako vstupní.

```
pinMode(promennaProPin, INPUT);
```

OUTPUT

Při použití jako parametru funkce *pinMode()*, nastaví digitální pin jako výstupní.

```
pinMode(promennaProPin, OUTPUT);
```

Podmínky if … else …

Podmínkami se nazývají prvky, které regulují směr toku programu. Podmínky lze rozdělit na *if … else …* a *switch … case ….* Podmínka *switch … case …* se vyskytuje v praxi nečasto, na rozdíl od *if … else …*, která může být aplikována ve třech základních variacích:

Podmínka if …

Příkazy 1 a 2 atd. se splní, pokud výraz v závorkách bude totožný booleanové hodnotě *true*. Pokud výraz v kulatých závorkách bude *false*, program nebude brát v úvahu, jaké jsou výrazy v složených závorkách. Tento typ podmínky *if … else …* se používá, pokud je potřeba ověřit jeden výraz a není nutné dělat nějaké činnosti v případě, že tento výraz není pravdivý.

```
if (výraz){
      // příkaz 1
      // příkaz 2
      // ...
}
```

Podmínka if … else …

Pokud v případě nesplnění podmínky *if …* je potřeba realizace jiných příkazů, používá se blok *else …*, a tím vzniká další variace podmínky *if … else ….*

```
if (výraz) {
      // příkaz 1
      // příkaz 2
      // ...
}
else {
      // příkaz 3
      // příkaz 4
      // ...
}
```

Podmínka if … else if … else …

Pokud variant je víc než dvě, lze využit bloky *else … if …*, které se budou postupně ověřovat v případě nesplnění první podmínky *if …*. Totiž na začátku se ověří výraz 1. Pokud bude totožný true, proběhnou příkazy 1 a 2, a další bloky *else … if …* a *else* budou ignorovány programem. Pokud výraz 1 je totožný *false*, ověří se výraz 2 v kulatých závorkách a pokud je *true* realizují se příkazy 3 a 4. V jiném případě se ověří další výraz v kulatých závorkách a tak dál. Pokud všechny výrazy bloků *else … if …* jsou totožné false, program přeskočí na blok *else …* a konečně ověří ho. Základní ideou je, že pokud program narazí na pravdivý výraz, splní se všechny příkazy daného bloku a další podmínky už se ignorují.

```
if (výraz) {
      // příkaz 1
      // příkaz 2
      // ...
}
else if (výraz 2) {
      // příkaz 3
      // příkaz 4
      // ...
}
// tady mohou být další bloky else if ...
else {
      // příkaz 5
      // příkaz 6
      // ...
}
```

Cykly

Dovolují vykonávat příkaz nebo skupinu příkazů opakovaně. Existuje několik základních druhů cyklů. Nejčastěji používanými jsou *while …*, *do … while …* a *for …*.

Cyklus while …

Tento cyklus dělá příkazy 1, 2 do té doby, pokud se splňuje výraz 1 v kulatých závorkách. Jakmile se podmínka přestane splňovat, cyklus se ukončí.

```
while (výraz 1) {
      // příkaz 1
      // příkaz 2
      // ...
}
```

Cyklus do … while…

Cyklus *do … while…* se liší od cyklu *while …* tím, že blok *do …* s příkazy 1 a 2 se splní ještě před ověřením pravdivosti výrazu 1 v kulatých závorkách. Totiž v podstatě se jedná o stejných cyklech pouze s tím rozdílem, že v cyklu *do … while…* příkazy 1 a 2 se splní alespoň jednou ještě před ověřením podmínky. Poté už funguje stejně jako cyklus *while ….*

```
do {
      // příkaz 1
      // příkaz 2
      // ...
}
while (výraz 1);
```

Cyklus for …

V kulatých závorkách cyklu *for …* jsou tři parametry. První z nich je inicializace proměnné, která bude sloužit jako počáteční hodnota cyklu. Třetí parametr je sčítač, sloužící k zvýšení nebo snížení počáteční hodnoty (prvního parametru). No a druhý parametr je podmínka, která ověří, zda stále zvyšována (nebo snižována) hodnota proměnné nepřekročila mez. Totiž základní ideou je, že příkazy 1 a 2 se vyplní pouze tolikrát, kolikrát je předem určeno.

```
for (inicializace proměnné s počáteční hodnotou; podmínka; sčítač)
{
      // příkaz 1
      // příkaz 2
      // ...
```

```
}
```

I/O funkce

Piny na desce Arduino mohou být konfigurovány jako vstupní nebo výstupní. Arduino IDE má v sobě pět základních funkcí pro práci s vstupy a výstupy. Těmito funkcemi jsou *pinMode()*, *digitalRead()*, *digitalWrite()*, *analogRead()* a *analogWrite()*.

pinMode()

Slouží pro nastavení pinu jako vstupního nebo výstupního. Uvnitř kulatých závorek má dva parametry. Prvním parametrem je číslo pinu, který se bude konfigurovat, a druhým parametrem je režim pinu, který může být buď *INPUT* nebo *OUTPUT* (vstupní nebo výstupní).

```
pinMode(5, OUTPUT);
```

digitalRead()

Slouží pro sečtení dat z digitálních pinů. Uvnitř kulatých závorek má jeden parametr, kterým je číslo pinu, ze kterého se sečítají data. Tato funkce na výstupu vrací hodnotu *HIGH* nebo *LOW* (přítomnost nebo absence napětí na pinu).

```
digitalRead(3);
```

digitalWrite()

Slouží pro zápis dat na digitální piny. Uvnitř kulatých závorek má dva parametry, První parametr je číslo pinu, na který se zapisují data, a druhým parametrem je hodnota, kterou je třeba poslat na tento pin. Hodnota může být buď *HIGH* nebo *LOW* (přítomnost nebo absence napětí na pinu).

```
digitalWrite(4, HIGH);
```

analogRead()

Pomocí této funkce můžeme přesně zjistit jak velké napětí je na analogovém pinu. Hodnotám napětí odpovídají digitální hodnoty z určitého rozsahu. Funkce má jeden parametr, a to je číslo pinu, s kterého je potřeba sečíst data. Funkce *analogRead()* na výstupu vrací digitální hodnotu z určitého rozsahu, kterou následně je možné převést na hodnotu napětí.

```
analogRead(A2);
```

analogWrite()

Funkce *analogWrite()* zapisuje digitální hodnotu na určitý pin. V kulatých závorkách má dva parametry. Prvním parametrem je číslo pinu, na který se zapisuje digitální hodnota, a druhým parametrem je samotná digitální hodnota z určitého rozsahu.

```
analogWrite(A0, 1000);
// při rozsahu digitálních hodnot třeba od 0 do 1023
```

Dodatečná funkce

delay()

Slouží pro zastavení programu na určitý čas. Má jeden parametr, který označuje čas v milisekundách, na který se program "zamrzne". Po uplynutí času prodlevy program bude pokračovat vykonávat příkazy, následující po funkci *delay()*.

millis()

Funkce vrací čas uplynutý od spouštění programu do toho okamžiku, když tato funkce byla použitá. Čas je vracen v milisekundách. Funkce nemá žádné parametry.

map()

Mění rozsah hodnot pro určitou veličinu v programu. Má pět parametrů. Prvním parametrem je veličina, pro kterou je potřeba změnit rozsah hodnot. Druhým a třetím parametry je současný rozsah hodnot, a čtvrtým a pátým parametry je nový rozsah hodnot.

```
map(velicina, 0, 100, -50, 150);
```

Knihovny

Knihovny rozšiřují možnosti programů v prostředí Arduino IDE. Pokud je potřeba použití dodatečných funkcí, je možné importovat knihovnu, poskytující tyto funkce. Aby importovat knihovnu do svého programu, je nutné v oblasti definování proměnných, před funkcí *setup(),* použit příkaz *#include <nazevKnihovny.h>;.*

```
#include <WiFI.h>;
```

2.4. Arduino M0 Pro

Pro účely této diplomové práce byla zvolená deska Arduino M0 Pro.

Obrázek 2: Deska Arduino M0 Pro [5]

Na této desce je umístěn mikroprocesor od výrobce Atmel. Má 256 kB flash-paměti, které je dost i pro docela rozsáhlé projekty a pro účely této diplomové práce by mělo víc než stačit. Taky na desce je k dispozici dva USB porty, každý, z kterých se dá použit pro napájení desky. Dalším druhem napájení může být externí zdroj třeba z AC/DC převodníku nebo baterie. Pro externí zdroj napájení používá se pin "Vin", pro který povolený rozsah napětí je 6 až 20 V. Každý z čtrnácti digitálních pinů na desce Arduino M0 Pro lze použit jako vstupní nebo výstupní. Zda pin bude vstupní nebo výstupní lze určit nastavením parametrů funkce pinMode(). V případě, že uživatel chce, aby pin byl vstupní, parametr funkce musí být "IN", v jiném případě parametr bude "OUT". Digitální piny operují při napětí 3,3 V a mají maximální stejnosměrný proud 7mA. Do pinu číslo 13 je zapojena vestavěná LED dioda. Každý z šesti analogových pinů A0-A5 ve výchozím nastavení má rozsah hodnot od 0 do 3,3 V a poskytuje 12 bitů rozlišení nebo 4096 různých hodnot. Pin A0 taky poskytuje analogové výstupy s 10-bitovým rozlišením nebo 1024 různých hodnot. Přítomnost analogových výstupů na pinu A0 je jedním z hlavních důvodů, proč deska Arduino M0 Pro byla zvolená pro tuto diplomovou práci. Vzhledem k tomu, že je potřeba řídit systém prostřednictvím spojité regulace, možnost použití analogových výstupů je velmi užitečná a eliminuje potřebu v přídavných modulech, které rozšiřují možnosti Arduina a jmenují se "shieldy". Na rozdíl od skokové regulace, kde ventil má pouze dva stavy (otevřený nebo uzavřený), spojitá regulace umožní přepnout ventil do více poloh.

Technical specs

Arduino Microcontroller

Microcontroller	ATSAMD21G18, 48pins LQFP
Architecture	ARM Cortex-M0+
Operating Voltage	3.3V
Flash memory	256 KB
SRAM	32Kb
Clock Speed	48 MHz
Analog I/O Pins	6 + 1 DAC
DC Current per I/O Pins	7 mA (I/O Pins)

General

Input Voltage	5-15 V
Digital I/O Pins	20
PWM Output	12
Power Consumption	44 mA
PCB Size	53.34 x 68.58 mm

Obrázek 3: Technické charakteristiky desky Arduino M0 Pro [5]

3. Pneumatické ventily

Pneumatický ventil je přístrojem pro regulaci průtoku plynu, který je generovaný z kompresoru, vakuové pumpy nebo podobného zařízení. Ventily regulují systém vykonáváním jedné nebo více z následujících funkcí: zastavením nebo nastartováním průtoku vzduchu, změnou průtokového množství, změnou směru pohybu vzduchu, regulací tlaku v procesu. Existuje celá řada ventilů, které se splňují tyto funkce.

3.1. Klasifikace ventilů

Ventily, používané v pneumatické regulaci lze rozdělit do čtyř základních skupin:

- Ventily pro regulaci směru
- Ventily pro regulaci průtoku
- Ventily pro regulaci tlaku
- Nevratné ventily

Ventily pro regulaci směru určují cestu, kterou se bude vzduch pohybovat. V závislosti na požadavku vzduch může se pohybovat určitým směrem, zároveň jiné směry budou uzavřený pro průchod nebo vzduch půjde do atmosféry přes vypouštěcí port. Ventily pro regulaci směru mohou být posouzeny podle počtu portů a poloh. Množství portů a poloh se obvykle píše přes lomítko. V případě nápisu "5/2" jedná se o ventil s pěti porty a dvěma polohami. Další skupinou ventilů jsou ventily pro regulaci průtoku. Totiž tady se jedná o průtokovém množství vzduchu. Tyto ventily určují, nakolik velký bude objem protékajícího vzduchu. Ventily pro regulaci tlaku určují velikost tlaku vzduchu uvnitř něho. Poslední druh ventilů je nevratný ventil, zvláštností kterého je protékání vzduchu pouze jedním směrem. Druhý směr ventilu je zablokovaný pro průtok vzduchu.

Taky ventily mohou mít různé způsoby ovládání. Rozlišují se ruční, mechanické, pneumatické a elektrické druhy ovládání. Ručně ovládané ventily mohou se ovládat pomocí tlačítka, páky, pedále atd. Mechanicky ovládané ventily se ovládají pístem, pružinou, válcem atd. Co se tyká pneumaticky ovládaných ventilů, ovládání probíhá

prostřednictvím stlačeného vzduchu působícího na ventil. No a v případě elektricky ovládaných ventilů, funkci ovládání splňuje solenoid.

Taky ventily lze rozdělit na monostabilní a bistabilní. Monostabilní ventily mají preferovanou polohu, do které se pokaždé automaticky vracejí po vypnutí signálu. Bistabilní ventily naopak nemají preferovanou polohu, a při zastavení signálu mohou se ocitnout v kterékoli z nich.

3.2. Solenoidový proporcionální ventil

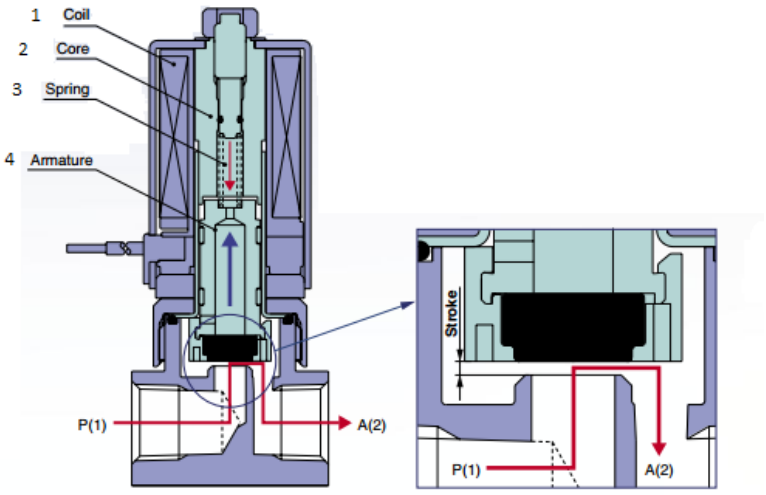
Vzhledem k tomu, že ventily v této úloze by se měly ovládat elektricky pomocí desky Arduino, byla potřeba si zvolit ventil ze skupiny elektricky ovládaných. U těchto ventilů v konstrukci se převážně používá solenoid. Většina solenoidových ventilů operuje na digitálním principu. To znamená, že ventil může mít dva stavy: zapnuto, když cívka je aktivována elektrickým proudem, a vypnuto když ventilem neprobíhá elektrický proud. Pro tuto úlohu byla potřeba proporcionálního ventilu oproti dvoustavovému, protože regulace pneumatického systému by měla probíhat spojitě, totiž ventil by se měl přepínat do více než dvou poloh a přesněji řečeno do všech hodnot z rozsahu povolených.

U solenoidových ventilů se často setkáš s nápisy “NC“ a “NO“. První znamená “Normally Closed“, což se překládá z angličtiny jako ventil, který je ve výchozím stavu uzavřený. Pod výchozím stavem rozumíme stav, při kterém na ventil nepůsobí signál. Druhý nápis znamená “Normally Opened“ nebo ventil, který je otevřený ve výchozím nastavení. Ventily s nápisem “NO“ umožňují průtok vzduchu ventilem při absenci působícího signálu.

Ventil PVQ13-6M-03-M5-A od výrobce SMC, který je světovou jedničkou na trhu v oblasti pneumatiky, je právě solenoidovým proporcionálním ventilem, kvůli tomu by měl vyhovovat této úloze

Obrázek 4: Solenoidový proporcionální ventil SMC PVQ13-6M-03-M5-A [10]

Tento ventil patří k sérii ventilů PVQ10, které fungují na podobném principu, znázorněnému na Obrázku 5. Tento princip spočívá v tom, že armatura se přitahuje k jádru, v případě působení elektrického proudu na cívku. V závislosti na tom, jak velký je aplikovaný proud, proporcionálně tomu se armatura přitahuje k jádru a mezera, určená pro průchod vzduchu, se zvětšuje. Totiž čím větší je aplikovaný elektricky proud, tím větší je mezera pro průchod vzduchu. Pokud elektrický proud nepůsobí na cívku vůbec, pružina, umístěná mezi armaturou a jádrem, vrací armaturu do výchozího stavu a uzavírá mezeru pro průtok vzduchu. Vstup, jak je vidět z obrázku, je označen jako P (1), a výstup - jako A (2). Z toho se dá říct, že ventil SMC PVQ13-6M-03-M5-A je "NC" nebo ve výchozím stavu uzavřený.



Obrázek 5: Konstrukce ventilu SMC PVQ13-6M-03-M5-A [9]

1. Cívka

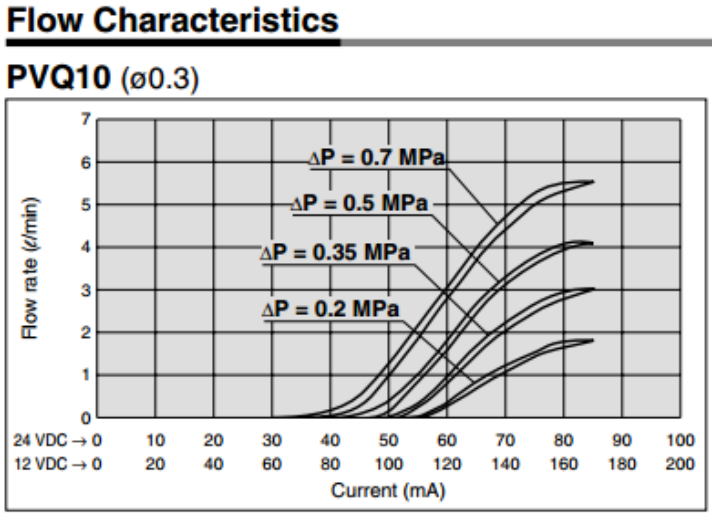
2. Jádro

3. Pružina

4. Armatura

Podle všeho, co bylo řečeno, se dá konstatovat, že míra průchodu vzduchu ventilem je proporcionální velikosti aplikovaného elektrického proudu. To je důvod, proč se těmto ventilům říká “proporcionální”.

Tělo ventilu je vyrobené z nerezové ocele. Životní doba ventilu je 25 milionů cyklů. Únikové množství vzduchu ve vypnutém stavu je 5 cm^3 na minutu. V samotném celém názvu ventilu se kryje důležitá informace o něm. ”13“ znamená, že, jak už bylo řečeno, ventil je uzavřený ve výchozím stavu nebo ”NC“, ”6“ označuje provozní napětí 12 VDC. ”03“ znamená, že průměr ventilu se rovná 0,3 mm a maximální provozní tlak – 0,7 MPa. Na Obrázku 6 vidíme průtokové charakteristiky toho ventilu, kde je znázorněna závislost míry průtoku vzduchu ventilem na aplikovaném na ventil proudu.



Obrázek 6: Průtoková charakteristika ventilu SMC PVQ13-6M-03-M5-A [9]

4. Další zařízení

4.1. Kompresor

Pro tuto úlohu byl zvolen dostupný na katedře kompresor AS-176. V podstatě se jedná o mini kompresor s vestavěným vzdušníkem objemem 0,3 litrů, který by měl splňovat úkol zdroje vzduchu pro pneumatický systém.

Obrázek 7: Kompresor AS-176 [13]

Kompresor dodává 20-23 litrů vzduchu za minutu. Tlak, při kterém se automatický nastartuje, je 2,1 barů nebo 210 kPa, a automaticky se zastaví – 3,1 barů nebo 310 kPa. Fyzické rozměry kompresoru 160 x 165 x 185 mm a hmotnost 2,7 kg. Ke svému provozu vyžaduje napětí ve velikosti 220 – 240 V, které je možné jednoduše obdržet z jakékoli zásuvky. Taky jeho užitečnou vlastností je přenosnost, což je velmi jednoduchou úlohou vzhledem k jeho malým rozměrům a hmotnosti. Dalšími vlastnostmi jsou jeho nízká hlučnost (47 dB) a ochrana před přehřátím. Na povrchu kompresoru jsou umístěný regulátor tlaku a měřidlo tlaku, které poskytuje informaci o tom, jaký tlak je momentálně aplikován.

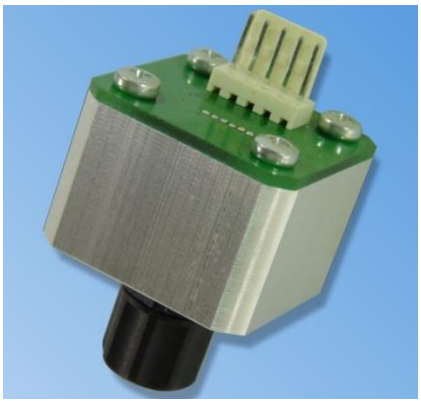
4.2. Čidla

Čidla jsou velmi důležitými prostředky automatického řízení. Vykonávají určité funkce, ke kterým patří získání informace o nějakém procesu, přenos této informace a její následné zpracování a vyhodnocení. Volba čidla pro tu nebo jinou úlohu je závislá

na několika faktorech. Za prvé, samozřejmě, záleží na tom, jaký je druh měřené veličiny a rozsah hodnot této veličiny. Pak může být celá řada dalších faktorů, které ovlivňují volbu snímače, jako jsou přesnost měření, pracovní prostředí, rychlost odezvy na změnu atd. Podle prvního a zásadního faktoru, kterým je druh měřené veličiny, dělí se na čidla: síly, zrychlení, polohy, hladiny, teploty, průtoku, ionizujícího záření, vlhkosti, vodivosti, hustoty atd. Pro tuto úlohu byla potřeba si zvolit dva snímače: jeden ze skupiny čidel tlaku a druhý ze skupiny čidel polohy (nebo vzdálenosti). Čidla tlaku podle způsobu realizace procesu měření se dělí na deformační, kapalinové, odporové, kapacitní, piezoelektrické, rezonanční, indukční a optické. Snímače vzdálenosti stejným způsobem jako snímače tlaku taky se dělí do několika skupin. Snímače vzdáleností bývají odporové, kapacitní, kontaktní, indukční, zvukové a optické.

4.2.1. Čidlo tlaku

Čidlo tlaku DRMOD-I2C-R1B6 patří ke skupině čidel ze série DRMOD-I2C-R od výrobce Hygrosens Instruments, které pracují na stejném principu, ale rozdílem mezi nimi je to, že mají různé rozsahy měřených hodnot tlaku.



Obrázek 8: Čidlo tlaku DRMOD-I2C-R1B6 [12]

Princip spočívá v tom, že tlak se měří na základě deformaci keramické membrány. Tento senzor měří tlak na keramické membráně relativně k tlaku vnějšího okolí. Čidlo má pouze jeden kontakt pro měření tlaku a druhá strana membrány je odkrytá vnějšímu okolí. Rozsah hodnot tlaku pro této čidlo může být od 0 do 1,6 barů nebo od 0 do 160 kPa. Tyto hodnoty jsou proporcionální výstupnímu napětí, které generuje čidlo. Tento rozsah výstupního napětí je od 0 do 5 V. To znamená, že hodnotě 0 V odpovídá tlak 0 kPa, a hodnotě 5 V – 160 kPa. Vstupní operující napětí je v rozsahu

od 6 do 15 V a spotřeba energie nesmí překročit 5 mA (rekomandováno 3 mA). Povolený pracovní rozsah teploty je -40 – 100 °C. Je důležitým nepřesahovat meze všech těchto parametrů, aby nedošlo k poškození čidla.

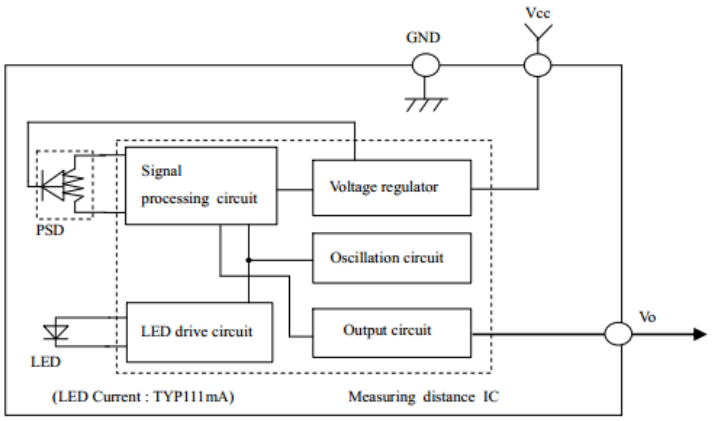
Čidlo DRMOD-I2C-R1B6 má pět pinů, každý z kterých má určitý úkol. Pin číslo jedna má název "OUT" a slouží pro výstupní napětí. Druhý a třetí mají značku "SDA" a "SCL" a slouží pro komunikaci digitálního I2C-rozhrání s mikroprocesorem. Tuto funkci není potřeba používat pro naši úlohu. Čtvrtý pin má název "GND" a jeho účelem je uzemnění. No a poslední pátý se značí "VDD" a je určen pro vstupní operující napětí 6 až 12 V.

4.2.2. Čidlo vzdálenosti

Výrobcem čidla vzdálenosti GP2Y0A41SK0F je známá společnost Sharp.

Obrázek 9: Čidlo vzdálenosti GP2Y0A41SK0F [11]

Patří ke skupině čidel, založených na optickém principu měření. Jeho fyzické rozměry 29,5 x 13 x 13,5. Fyzický rozsah měření je 4 - 30 cm a tomu proporcionálně odpovídá hodnota analogového výstupu, který poskytuje snímač. Snímač generuje analogový výstup v podobě výstupního napětí jako odezvu na změnu polohy objektu. Využití metody triangulace přispívá tomu, že výsledky měření nemohou být snadno ovlivněné působením činitelů vnějšího okolí, jako je třeba teplota vzduchu.



Obrázek 10: Obvod pro zpracování signálu čidla GP2Y0A41SK0F [11]

Snímač má tři piny. První pin má název ”V0” a slouží pro výstup z čidla, který je generován v podobě analogového napěťového signálu. Druhý je určen pro uzemnění a má značku ”GND”. No a poslední třetí pin odpovídá za vstupní operující napětí, u kterého doporučovaný rozsah je 4,5 – 5,5 V a značí se ”Vcc”.

5. Spojitá regulace

Řízením se nazývá působení jednoho systému na druhý takovým způsobem, aby bylo dosaženo nějakého výsledku. V takovém případě první systém se jmenuje řídícím a druhý – řízeným. Řízení se dá rozdělit na dva druhy. První druh je ovládání. Jedná se o takový způsob přímého řízení, kde neexistuje zpětná vazba, totiž systém se ovládá pouze pomocí vstupních veličin. Druhý druh řízení je regulace, kde se jedná o řízení se zpětnou vazbou, a důležitým krokem tohoto procesu je výpočet regulační odchylky. Existence zpětné vazby v regulačním druhu řízení znamená, že se systém reguluje nejenom pomocí vstupních veličin, ale taky pomocí výstupních, které při následné transformaci v akční veličinu se stávají zároveň vstupem do systému. Regulace je samozřejmě mnohem lepším způsobem řízení než ovládání z toho důvodu, že výstupní hodnoty jsou bližší k žádanému výsledku než u ovládání. Proces regulace se realizuje pomocí nástrojů, kterým se říká regulátory. Regulátorem se obvykle nazývá souhrn všech prvků, které se zúčastňují v procesu regulace. Důležitou etapou činnosti regulátoru je výpočet regulační odchylky:

$$e(t) = w(t) - y(t) \qquad (5.1)$$

Kde $w(t)$ – žádaná hodnota, a $y(t)$ – regulovaná veličina.

Z této rovnice je vidět, že regulační odchylka je rozdíl mezi požadovanou hodnotou a výstupem s regulátoru. Samozřejmě platí, že čím menší je hodnota regulační odchylky, tím lépe regulátor splňuje svůj úkol. Regulátor zpracovává regulační odchylku a na základě toho vypočítá akční veličinu. Úkolem akční veličiny je zajistit zmenšení hodnoty regulační odchylky nebo případně její odstranění vůbec.

Regulátory lze rozdělit do skupin podle několika hledisek. Podle průběhu výstupní veličiny regulátory bývají spojité a nespojité (diskrétní). Diskrétní nebo nespojitá regulace je jednoduchý druh regulace, kde výstupem z regulátoru může být několik diskrétních hodnot z povoleného rozsahu. Totiž výstup z těchto regulátoru může mít pouze několik stavů. U spojitých regulátorů výstupní veličina může nabývat všech hodnot z povoleného rozsahu.

5.1. Druhy spojitých regulátorů

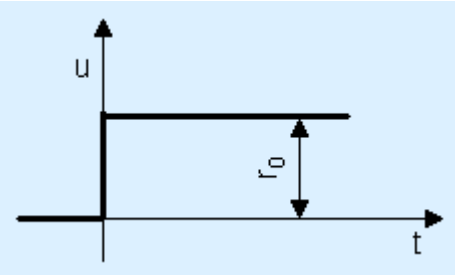
Spojité regulátory je možné rozdělit na několik dalších druhů. Obecně regulátory mohou v sobě mít proporcionální, integrační a derivační složky. Podle absence nebo přítomnosti těchto prvků se dá říct, o který druh regulátoru se jedná. Pokud regulátor má jenom proporcionální prvek, můžeme říct, že je to P-regulátor. Pokud jenom integrační prvek – je to I-regulátor. Kombinace proporcionálního a derivačního prvků udělá PD-regulátor, a kombinace proporcionálního a integračního – PI-regulátor. No a použití všech třech prvků znamená, že byl vytvořen PID-regulátor.

5.1.1. P-regulátor

Jedná se o proporcionální regulátor. Má takový název, protože hodnota akční veličiny se mění proporcionálně hodnotě regulační odchylky. V rovnici akční veličiny je přítomná pouze proporcionální složka:

$$u = K_p e \qquad (5.2)$$

Kde u – akční veličina, K_p – proporcionální člen, e – regulační odchylka. Na Obrázku 11 znázorněná přechodová charakteristika P-regulátoru, kde $r_0 = K_p$.



Obrázek 11: Přechodová charakteristika P-regulátoru [4]

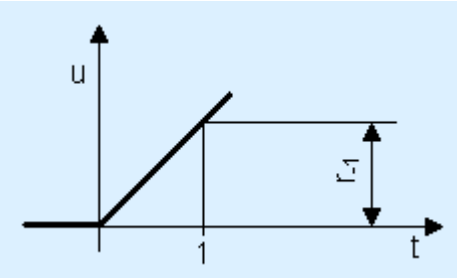
Výhodou proporcionálního regulátoru je rychlá odezva na změny v procesu regulace, což je výsledkem okamžité korekční reakce v případě výskytu odchylky. Další výhodou je velmi stabilní proces regulace, v případě, že proporcionální člen je správně zvolený. Nevýhodou těchto regulátoru je ustálená regulační odchylka, která se nikdy nebude rovnat nule. Při regulaci je také třeba si dávat pozor na veličinu hodnoty proporcionálního členu, protože velké hodnoty způsobují klesání stability procesu.

5.1.2. I-regulátor

Jedná se o integrační regulátor. Používá se pro úplnou korekci deviace systému. Pokud hodnota regulační odchylky není nulová, integrační regulátor přinutí akční veličinu změnit situaci. Jenom když hodnoty žádané a regulované veličin budou stejné, systém se může považovat za vyvážený. V rovnici akční veličiny je přítomná pouze integrační složka a je proporcionální integrálu od regulační odchylky:

$$u = K_i \int_0^t e(\tau)d\tau, \qquad K_i = \frac{1}{T_n} \qquad (5.3)$$

Kde u – akční veličina, K_i – integrační člen, e – regulační odchylka závislá na čase, T_n – integrační časová konstanta. Na Obrázku 12 je znázorněná přechodová charakteristika I-regulátoru, kde $r_{-1} = K_i$.

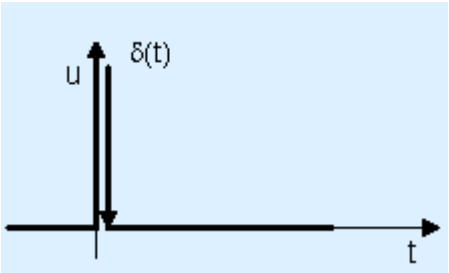


Obrázek 12: Přechodová charakteristika I-regulátoru [4]

Jak rychle se akční veličina bude zvedat nebo klesat záleží na regulační odchylce a integrační časové konstantě. Pokud regulátor má krátký integrační čas, akční veličina se zvedá rychleji. Výhodou integračního regulátoru je absence odchylky v ustáleném stavu. Ale má taky dvě nevýhody. Má pomalou odezvu při velkých hodnotách integrační časové konstanty. Druhou nevýhodou je náchylnost ke kmitání při malých hodnotách integrační časové konstanty, totiž v takovém případě proces se stává nestabilní.

5.1.3. PD-regulátor

Jedná se o proporcionálně-derivační regulátor. D-regulátory reagují ještě rychleji na změny v procesu regulace než P-regulátory. I když odchylka je malá, D-regulátor generuje velkou amplitudu, jakmile se nějaká změna vyskytuje. Ale D-regulátory nerozpoznávají signály ustálené chyby, protože bez ohledu na to, jak velká je odchylka, rychlost změny je nulová. Z toho důvodu D-regulátory se nečasto vyskytují v praxi. Na Obrázku 13 je znázorněná přechodová charakteristika D-členu.

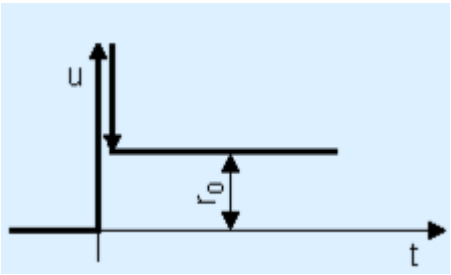


Obrázek 13: Přechodová charakteristika D-členu [4]

Většinou se používají v kombinaci s jinými regulačními prvky a nejčastěji s proporcionálním. Takovým způsobem vzniká PD-regulátor. Akční veličina u PD-regulátoru vzniká sečtením proporcionální a derivační složek:

$$u = K_p e + K_d \frac{de}{dt}, \qquad K_d = \frac{K_p}{T_v} \tag{5.4}$$

Kde u – akční veličina, K_p – proporcionální člen, K_d – derivační člen, e – regulační odchylka, T_v – derivační časová konstanta. Na Obrázku 14 je znázorněná přechodová charakteristika PD-regulátoru, kde $r_0 = K_p$.



Obrázek 14: Přechodová charakteristika PD-regulátoru [4]

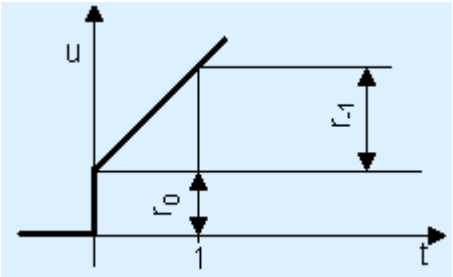
PD-regulátory se uplatňují v těch případech, když P-regulátory jsou nedostatečně efektivní.

5.1.4. PI-regulátor

Jedná se o proporcionálně-integrační regulátor. Tento typ regulátora se často vyskytuje v praxi. P-regulátor a I-regulátor jsou zapojený paralelně a tím vzniká PI-regulátor. PI-regulátor kombinuje v sobě výhody obou samotných regulátorů – stabilitu, rychlost odezvy, a absence ustálené odchylky. Rovnice akční veličiny má v sobě proporcionální a integrační složky:

$$u = K_p e + K_i \int_0^t e(\tau) d\tau, \qquad K_i = \frac{K_p}{T_n} \tag{5.5}$$

Kde u – akční veličina, K_p – proporcionální člen, K_i – integrační člen, e – regulační odchylka, T_n – integrační časová konstanta. Na Obrázku 15 je znázorněná přechodová charakteristika PI-regulátoru, kde $r_0 = K_p,\ r_{-1} = K_i$.



Obrázek 15: Přechodová charakteristika PI-regulátoru [4]

Proporcionální složka nutí akční veličinu okamžitě reagovat na signál o odchylce, zatímco integrační složka začíná působit jenom s odstupem v čase. Integrační časová konstanta reprezentuje čas, za který integrační prvek začíná generovat stejné působení jako proporcionální prvek na začátku.

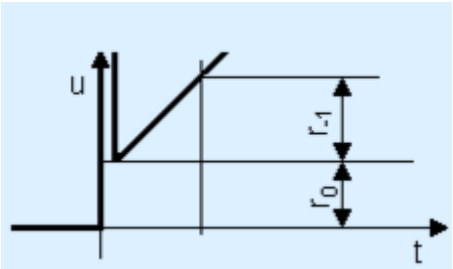
5.1.5. PID-regulátor

Jedná se o proporcionálně-integračně-derivační regulátor. Pokud k PI-regulátoru přidat derivační prvek, výsledkem bude extrémně univerzální PID-regulátor. Přidání derivačního prvku způsobí to, že regulovaná veličina dosáhne ustáleného stavu mnohem rychleji. Proporcionální, integrační a derivační prvky jsou zapojený paralelně. Rovnice akční veličiny má v sobě tři složky:

$$u = K_p e + K_d \frac{de}{dt} + K_i \int_0^t e(\tau) d\tau, \quad (5.6)$$

$$K_i = \frac{K_p}{T_n}, \qquad K_d = \frac{K_p}{T_v}$$

Kde u – akční veličina, K_p – proporcionální člen, K_i – integrační člen, K_d – derivační člen, e – regulační odchylka, T_n – integrační časová konstanta, T_v – derivační časová konstanta. Na Obrázku 16 je znázorněná přechodová charakteristika PID-regulátoru, kde $r_0 = K_p,\ r_{-1} = K_i$.



Obrázek 16: Přechodová charakteristika PID-regulátoru [4]

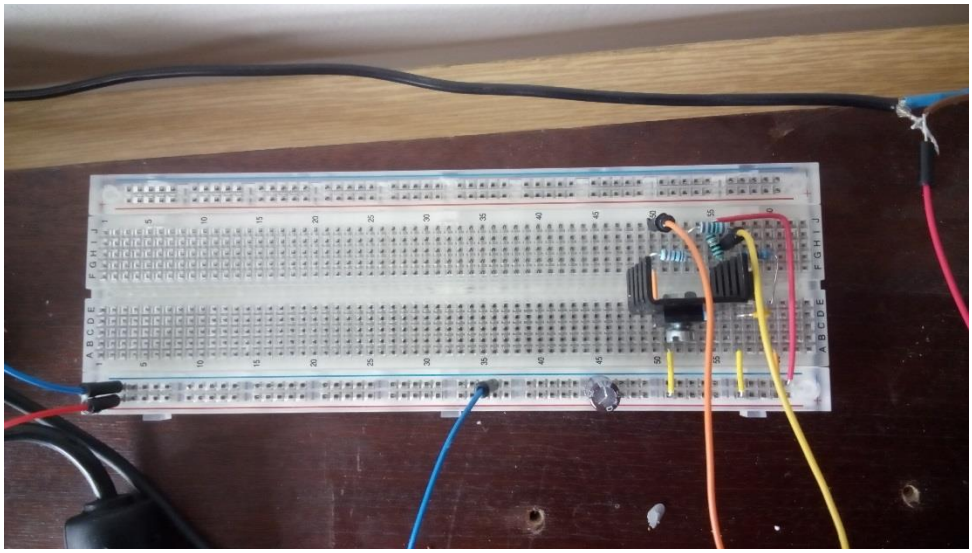
V porovnání s jinými regulátory PID-regulátor projevuje nejvíc sofistikovanou regulaci. Regulovaná veličina dosahuje žádaného stavu velmi rychle, taky rychle se stabilizuje a kmitá jen nepatrně kolem požadované hodnoty. Jedinou nevýhodou PID-regulátoru je docela složité nastavení parametrů regulátoru. Ale pokud ty parametry nastaveny správně, PID-regulátor je nejlepší volbou.

6. Realizace procesu regulace systému

V praktické části bude popsáno, jakým způsobem byly sestavený elektrický a pneumatický obvody systému, regulace těchto obvodů pomocí řídící jednotky Arduino a programu sestaveného v jeho programovacím prostředí Arduino IDE. Základním prvkem algoritmu řízení bude jeden z popsaných dřív regulátorů, které realizují proces spojité regulace.

6.1. Elektrický obvod operačního zesilovače

Arduino M0 Pro poskytuje napájení jiných komponent napětím velikosti 3,3 nebo 5 V. Avšak toto napětí není dostačující pro napájení ventilu, který vyžaduje 12 V. Z toho důvodu pro ventil byl použit externí zdroj napájení s napětím na jeho výstupu 15 V. Co se tyká velikosti řídícího signálu přiváděného na ventil, tak ten je založen na velikosti proudu aplikovaného na vstupu ventilu a měl by být v rozsahu od 0 do 330 mA. Arduino nemůže poskytnout tuto velikost proudu a z toho plyne, že nelze připojit Arduino přímo na ventil. Kvůli tomu bylo rozhodnuto vytvořit operační obvod znázorněny na Obrázku 17, který by umožnil řízení ventilu přes výstupní pin Arduino.



Obrázek 17: Obvod s operačním zesilovačem

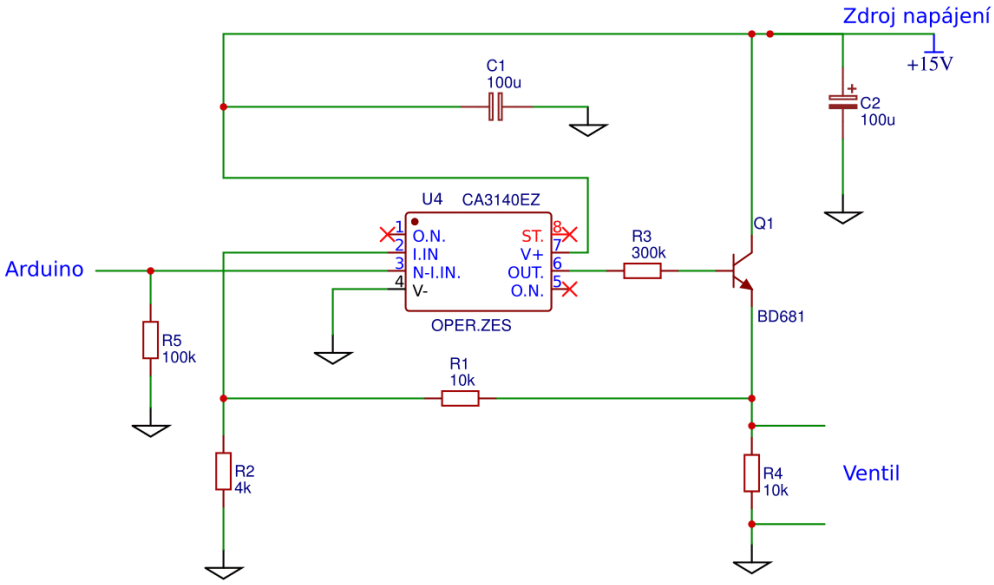
Schéma je realizována na operačním zesilovači CA3140EZ a výkonném duálním tranzistoru BD681, který se používá jako koneční zesilovač. Operační zesilovač je vybrán ze série zesilovačů rail-to-rail, které se liší od konvenčních vysokou vstupní

impedanci, širokým rozsahem výstupního signálu (od negativního do pozitivního napájení) bez zkreslení dynamických charakteristik. Z dokumentace pro operační zesilovač je vidět, že má osm pinů, ze kterých v tomto případě byly použity pouze pět. Piny číslo 1 a 4 s označením "OFFSET NULL" a pin 8 s označením "STROBE", z důvodu jejich zbytečnosti pro danou úlohu použity nebyly. Operační zesilovače běžně mají invertní a neinvertní vstupy a CA3140EZ není výjimkou. Neinvertní vstup má pin číslo 3 a označen "V+", invertní – pin číslo 2 a označen jako "V-". Piny číslo 4 a 7 jsou určeny pro napájení. Pin 4 má značku "Vs-" a je v tomto obvodu pro uzemnění, a na pin číslo 7 se značkou "Vs+" je přiváděno napětí velikosti 15 V z externího zdroje napájení. No a pin 6 je určen pro výstup z operačního zesilovače. CA3140EZ je zapojen do obvodu s negativní zpětnou vazbou, kde se koeficient zesílení spočítá podle vzorce:

$$Au = \frac{R_1+R_2}{R_2} \qquad (6.1)$$

Rezistory tady slouží jako děliče napětí. Tranzistor na výstupu poskytuje velké proudové zesílení. Zvláštnosti tohoto tranzistoru je nízký odpor přechodu "kolektor - emitor" v otevřeném stavu, což způsobuje vznik napětí blízkého k napájecímu.

Kompletní schéma zapojení všech komponent operačního obvodu je zobrazená na Obrázku 18:



Obrázek 18: Schéma obvodu operačního zesilovače

6.2. Kompletní elektrický obvod

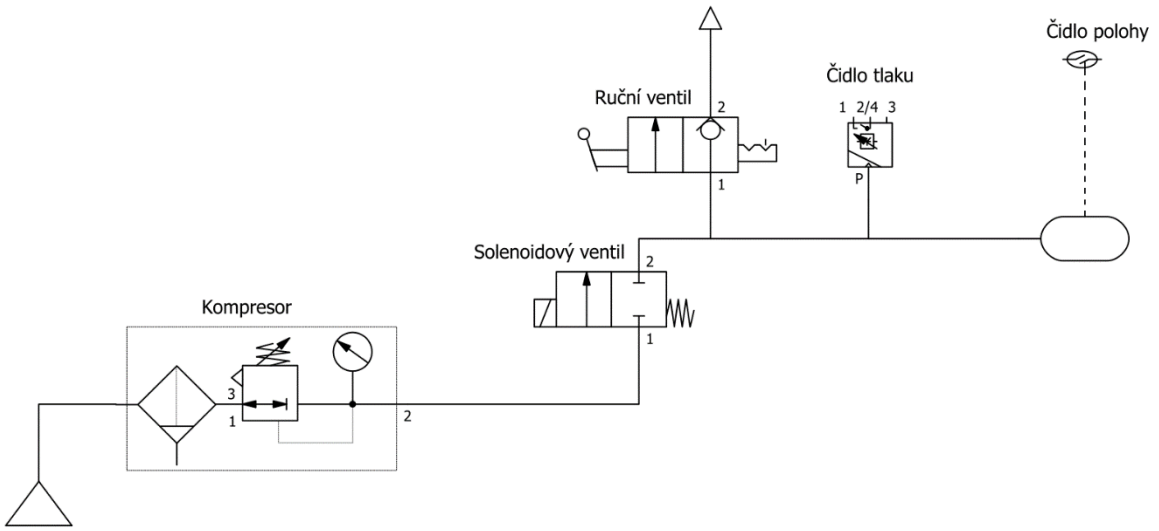
V této kapitole bude popsáno, jakým způsobem byl sestaven systém pro regulaci z elektrického hlediska. V elektrickém obvodu na Obrázku 19 je vidět všechny elektrické prvky, které se zúčastňují procesu regulace systému, propojený mezi sebou určitým způsobem. K těmto prvkům patří deska Arduino M0 Pro, solenoidový proporcionální ventil PVQ13, čidla tlaku a vzdálenosti a operační zesilovač. Zdrojem energie pro všechny prvky v elektrickém obvodu slouží obyčejný napájecí zdroj z největší části určený pro převod napětí, dodávaného ze zásuvky na malé napětí vhodné k napájení elektronických komponent počítače nebo jiných elektronických zařízení. Velikost napětí na výstupu ze zdroje bylo nastaveno na 15 V.

Jak už bylo řečeno v kapitole o Arduino, deska má analogové vstupní a výstupní piny, do kterých mohou být zapojena další zařízení, kterými jsou v tomto případě čidla a solenoidový ventil. Čidlo tlaku je zapojeno na vstup Arduina přes analogovy pin číslo 5, který je označen na desce “A5“, čidlo vzdálenosti – na analogový pin číslo 2, který je označen na desce “A2“. Prostřednictvím tohoto zapojení čidel tlaku a vzdálenosti na vstupy desky získáváme informaci o velikosti digitálních hodnot na výstupu z čidel a vidíme tuto informaci v programovacím prostředí Arduino IDE. Tři piny s označením “GND“ na desce slouží jako uzemnění pro sestaveny obvod. Pin s označením “A0“ je jediný pin určený pro výstup z Arduina, z kterého se posílá řídící signál na solenoidový ventil. Avšak tento signál nejde z Arduina přímo na ventil. Signál z výstupního pinu desky prochází operačním zesilovačem a až po zesílení může být použit jako řídící signál pro ventil. Všechno co se tyče propojení prvků v elektrickém obvodu je dobře znázorněno na schématu, které je z důvodu velkých rozměrů umístěné v příloze č. 1.

6.3. Pneumatický obvod

Schéma pneumatického obvodu je určená pro zobrazení toku vzduchu mezi jednotlivými komponenty pneumatického systému. Prvním prvkem v pneumatickém obvodu je kompresor, který má funkci zdroje vzduchu. Uvnitř kompresoru jsou vestavěny filtr vzduchu, regulátor tlaku a manometr. Dál po obvodu stlačený vzduch se pohybuje k solenoidovému ventilu, u kterého míra otevření závisí na řídícím signálu z desky Arduino. Pro účely vypouštění vzduchu ze systému slouží ručně ovládaný ventil.

No a posledními dvěma komponenty pneumatického systému jsou čidla tlaku a polohy, které zaznamenávají hodnoty tlaku a změnu výšky pneumatické pružiny. Všechny popsané komponenty pneumatického obvodu jsou znázorněny na Obrázku 20.

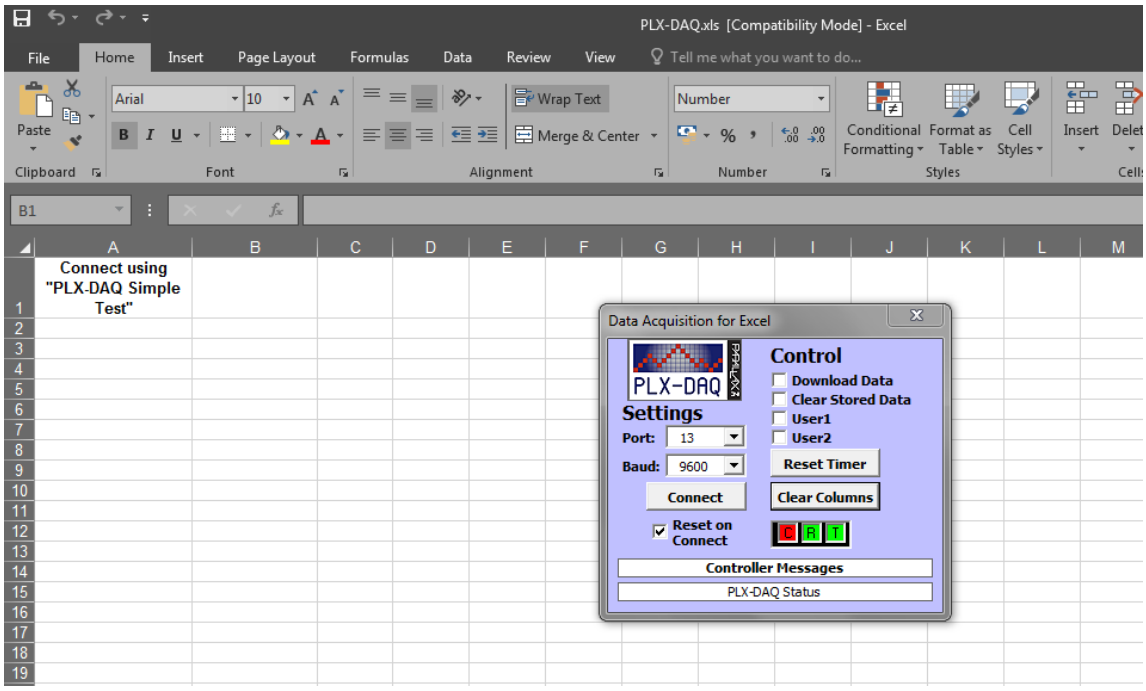


Obrázek 19: Schéma zapojení pneumatických prvků systému

6.4. Kontrola přesnosti měření čidel tlaku a vzdálenosti

Čidla tlaku a vzdálenosti splňují úkol zpětné vazby v systému, totiž pomocí těchto čidel získáváme informaci o tlaku v pružině a její zvednutí nad počáteční polohou. Abychom mohli říct, že čidla jsou příhodné pro měření, byla potřeba udělat kontrolní měření, kde by se porovnávaly skutečné fyzikální hodnoty tlaku (v kPa) a výšky (v milimetrech) pružiny s digitálními hodnotami na výstupech z čidel. Skutečné digitální hodnoty by měly narůstat nebo snižovat se proporcionálně jedna druhé. Jak už bylo řečeno v kapitolách 4.2.1. a 4.2.2., kde byly popsány čidla tlaku a vzdálenosti, tyto čidla generují výstupní napětí v rozsahu od 0 do 5 V. Těmto hodnotám výstupního napětí proporcionálně odpovídají digitální hodnoty a skutečné hodnoty tlaku v pružině a její zvednutí nad počáteční polohou v milimetrech. Signál z čidel v podobě digitální hodnoty přichází na analogové vstupy desky Arduino. Dál následuje znázornění těchto dat obdržených z desky, což se obvykle dosahuje pomocí sériového monitoru dostupného uvnitř programovacího prostředí Arduino IDE. Sériový monitor je podobou terminálů, používaných v populárních programovacích prostředích jako jsou Eclipse,

Intellij IDEA, Netbeans atd., sloužících pro komunikaci mezi uživatelem a programovacím prostředím. V daném případě sériový monitor slouží pro komunikaci mezi uživatelem a deskou, která probíhá zasláním dat na desku a zpětně obdržením dat z desky. Avšak data, které je vidět v sériovém monitoru jsou nepříhodné k dalšímu zpracování (třeba vytvoření grafů na základě těch dat). Z toho důvodu bylo rozhodnuto najít způsob jak dostat digitální hodnoty z čidel tlaku a vzdálenosti do obyčejného excelového souboru, který by umožnil další zpracování těchto dat. Aby toto bylo dosaženo, za prvé byla potřeba nainstalovat bezplatný software s názvem PLX-DAQ. Tento software v podstatě pouze přidává další funkčnost k Excelu, kterou je obdržení reálných dat z desky. Po instalaci softwaru PLX-DAQ na ploše počítače se objeví obyčejný excelový soubor s názvem PLX-DAQ. Po otevření tohoto souboru na obrazovce vidíme dobře známou excelovou tabulku s jediným rozdílem, že uprostřed se objeví malé okno s názvem “Data Acquisition for Excel”, což se překládá jako “Získání dat do Excelu” (Obrázek 21).



Obrázek 20: Úvodní zobrazení softwaru PLX-DAQ

Aby se data zapisovala z desky do excelu je potřeba nastavit číslo portu, přes který probíhá komunikace mezi deskou a Arduino IDE a rychlost sériového připojení. V tomto případě to byl port číslo 13, což je uvedeno v poli “Port“ pod nadpisem “Settings“.

Rychlost sériového připojení se nastaví v poli “Baud“. Hodnota “Baud“ by měla souhlasit s rychlostí sériové komunikace nastavené v programu. Pro začátek zapisování dat z desky Arduino je potřeba stisknout tlačítko “Connect“ a dolů místo nadpisu “PLX-DAQ Status“ se objeví nadpis “Connected“, podle kterého se dá poznat, že se připojení k desce podařilo. Dál už by se měly zobrazovat určité data v horním levém rohu pod buňkou “Connect using PLX-DAQ Simple Test“.

Na straně Arduina, přesněji řečeno v Arduino IDE, je potřeba vytvořit několik příkazů. První příkaz v sekci setup() smaže všechna předchozí data. Tento příkaz je nezbytný stejně jako první dva parametry druhého příkazu LABEL a Time, další parametry druhého příkazu určují název sloupce dat a můžou se pojmenovat podle požadavku uživatele. V mém případě to byly tlak a výška nafukování.

V sekci *setup()*:

```
Serial.println("CLEARDATA");
Serial.println("LABEL, Time, NazevSloupce1, NazevSloupce2,");
// je možné pojmenovat další sloupce
```

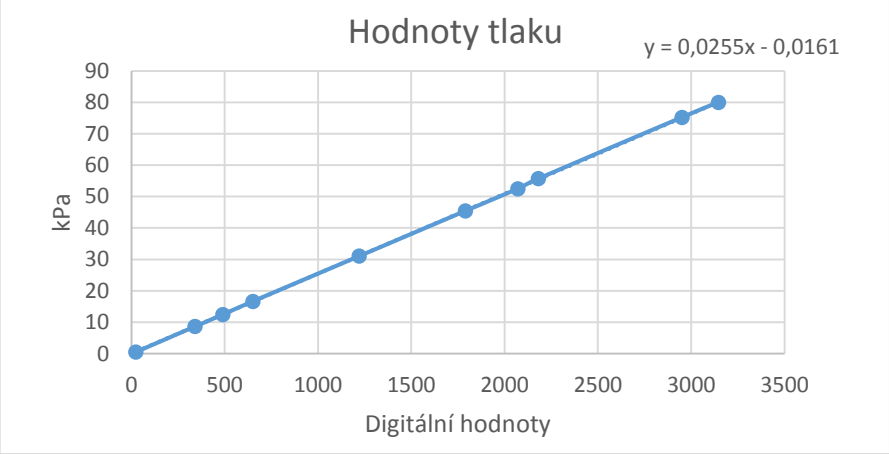
První příkaz v sekci loop() taky je nezbytné napsat právě v tom formátu, který je uvedený vyš. Parametry “DATA“ a “TIME“ zaznamenají datum a čas zapsání každého řádku dat. Proměnné promennaProData1, promennaProData2 atd. mají v sobě umístěné určité hodnoty, které by uživatel chtěl zobrazovat v excelu při každém kroku cyklu loop(). V mém případě to byly digitální hodnoty tlaku a výšky nafukování, umístěné do proměnných analogPressureValue a analogDistanceValue.

V sekci *loop()*:

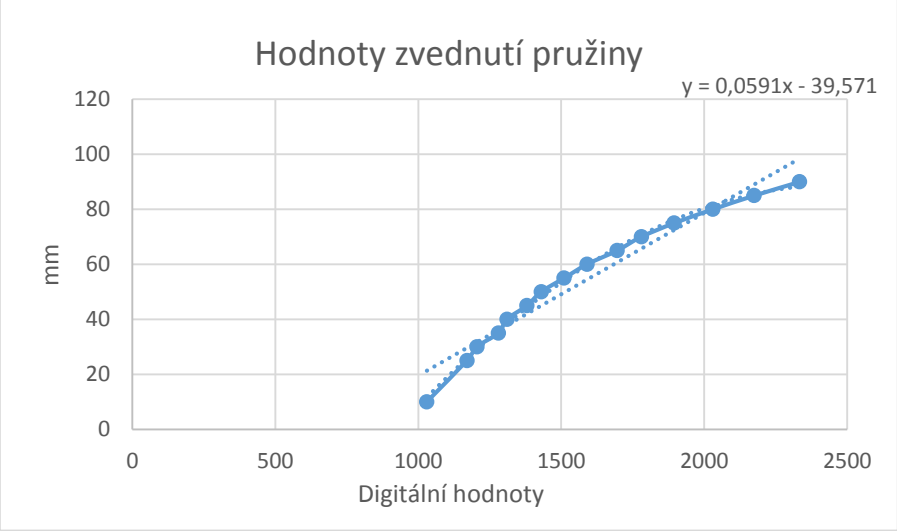
```
Serial.print("DATA, TIME,");
Serial.print(promennaProData1);
Serial.print(",");
Serial.println(promennaProData2);
// tady můžou být další data přes čárku
```

Takže takovým způsobem jsem získal digitální hodnoty tlaku a vzdálenosti do excelového souboru. Zároveň jsem naměřil pomocí obyčejného pravítka hrubé fyzické hodnoty výšky zvednutí pružiny vztažně k počátečnímu stavu a pomocí tlakoměru – hodnoty tlaku v pružině v různých okamžicích času. Na základě těchto dat se mi

podařilo vytvořit grafy, kde je vidět vztah mezi digitálními hodnotami a reálnými fyzickými veličinami. Jak je vidět z grafu na Obrázcích 22 a 23 digitální hodnoty z čidel tlaku a vzdálenosti stoupají proporcionálně k narůstání fyzikálních hodnot tlaku a zvednutí.



Obrázek 21: Tlak v pneumatické pružině v závislosti na digitálních hodnotách z čidla tlaku



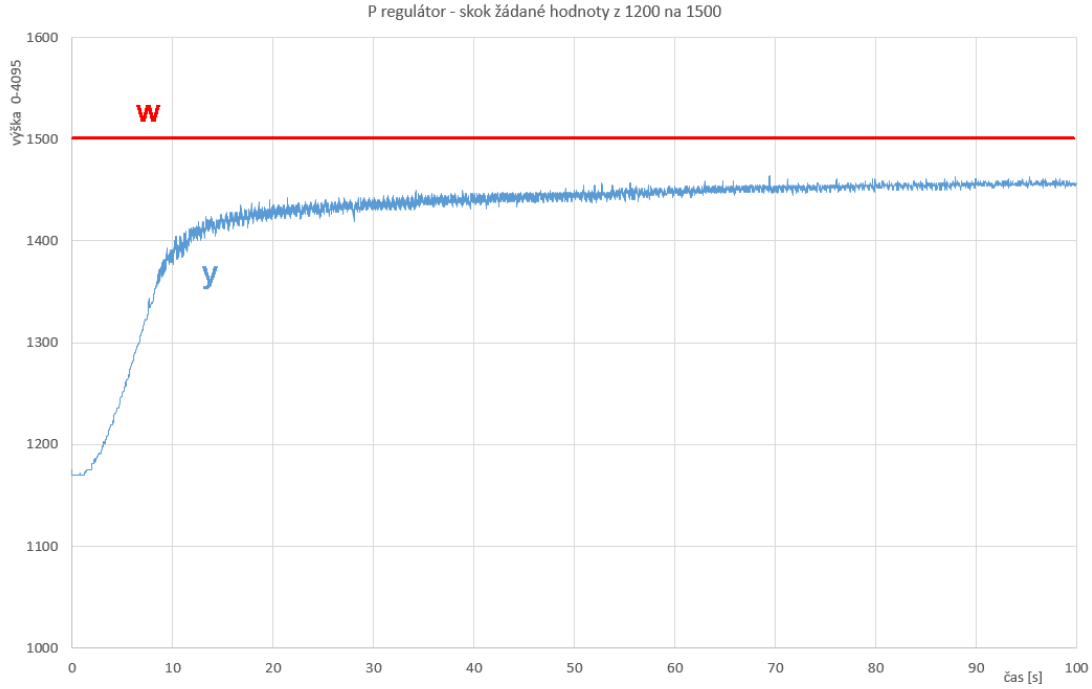
Obrázek 22: Zvednutí pneumatické pružiny v závislosti na digitálních hodnotách z čidla polohy

Jak je vidět z grafů závislost mezi digitálními hodnotami a tlakem v kPa je daná rovnicí (6.2) a závislost mezi digitálními hodnotami a výškou pružiny v milimetrech je daná rovnici (6.3).

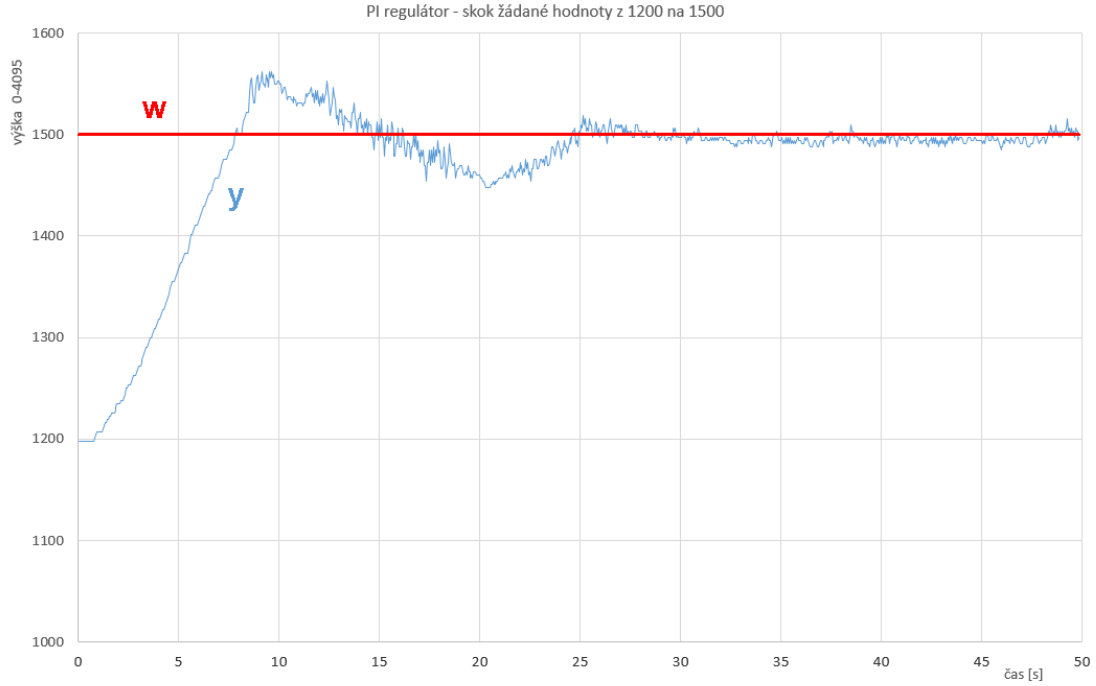
$$y = 0.0255x - 0.0161 \quad (6.2)$$

$$y = 0.0591x - 39.571 \quad (6.3)$$

Také se v rámcích měření podařilo vyzkoušet, jak se systém chová při skokové změně žádané veličiny. Na Obrázcích 23 a 24 je zobrazený průběh regulované veličiny při takové skokové změně digitální hodnoty žádané veličiny z 1200 na 1500.



Obrázek 23 Průběh regulované veličiny při skokové změně s použitím P-regulátoru



Obrázek 24 Průběh regulované veličiny při skokové změně s použitím PI-regulátoru

Obrázek 23 ukazuje průběh regulované veličiny při aplikaci P-regulátoru s konstantou $K_p = 6.4$, a Obrázek 24 – při aplikaci PI-regulátoru s konstantami $K_p = 8.2$ a $K_i = 5.4$. V prvním případě vidíme trvalou regulační odchylku velikosti 45, která je dána odečtením ustálené hodnoty regulované veličiny y od žádané hodnoty w. V druhém případě v počáteční fázi hodnota regulační odchylky je přibližně stejná jako u P-regulátoru, ale postupem času se regulační odchylka ustálí na nule, což znamená, že pomocí PI-regulátoru by se dalo dosáhnout mnohem přesnějšího výsledku.

6.5. Program

Pro účel regulace systému byl zvolen PI-regulátor, vzhledem k tomu, že poskytuje lepší výsledky než P-, I-, a PD-regulátory na jednou strany a seřízení konstant regulátoru je jednodušší než u PID-regulátoru na druhou stranu.

V první části programu jsou definovány všechny proměnné, které budou použity v sekcích setup() a loop(). V první proměnné je uložené číslo pinu, který bude použit k zaslání řídících příkazů na ventil.

```
// pin pro ovládání ventilu pro přivádění vzduchu
byte valvePin = A0;
```

Další dvě proměnné určují, z kterých pinů se informace nadchází na desku z čidel tlaku a vzdálenosti.

```
byte distanceSensorPin = A2; // pin pro výstup z čidla vzdálenosti
byte pressureSensorPin = A5; // pin pro výstup z čidla tlaku
```

Dál následuje definice proměnných pro regulační odchylku, žádanou veličinu, regulovanou veličinu a nakonec akční veličinu. Každá tato proměnná je definována dvakrát zvlášť pro tlak a pro výšku pružiny. To je uděláno z toho důvodu, že teoreticky by se dalo regulovat pneumatický systém podle obou z těchto parametrů. Avšak v tomto programu regulovanou veličinou je výška zvednutí pružiny. Hodnota žádané veličiny výšky se určuje uživatelem a zapisuje se do proměnné *distanceWinmm*. V tomto případě rovná se 50 mm. V případě regulace tlaku v pružině program by měl mít stejnou strukturu jen s několika nepatrnými odlišnostmi.

```
double pressureE; // digitální hodnota regulační odchylky pro tlak
```

```
double distanceE; // digitální hodnota regulační odchylky pro
výšku
double pressureW; // digitální hodnota žádané veličiny pro tlak
double distanceW; // digitální hodnota žádané veličiny pro výšku
double distanceWinmm = 50; // žádaná hodnota výšky v milimetrech
double pressureWinkPa = 0; // žádaná hodnota tlaku v kPa
// digitální hodnota regulované veličiny pro tlak
double pressureY;
// digitální hodnota regulované veličiny pro výšku
double distanceY;
// hodnota regulované veličiny pro tlak v kPa
double pressureYinkPa;
// hodnota regulované veličiny pro výšku v milimetrech
double distanceYinmm;
double pressureU; // digitální hodnota akční veličiny pro tlak
double distanceU; // digitální  hodnota akční veličiny pro výšku
```

Další dvě proměnné slouží pro uchovávání hodnot proporcionální a integrační konstant, a pomocná proměnná *integral* představuje integrál, který je spolu s integrační konstantou součásti integrační složky PI-regulátoru. Hodnota proporcionálního a integračního členů se zjišťuje experimentálním způsobem pomocí Wadeho metody. Všechny tyto proměnné budeme potřebovat pro výpočet akční veličiny.

```
const double Kp = 8.2; // proporcionální člen
const double Ki = 5.4; // integrační člen
double integral = 0; // integrační složka regulátora
```

V dalším úseku definice proměnných vidíme proměnné času. První z nich slouží pro nastavení frekvence vyhodnocení regulátoru, která je na hodnotě 30 milisekund. To znamená, že regulátor bude vyhodnocovat stav systému každých 30 milisekund. Další proměnné jsou určeny pro zaznamenávání času začátku regulace, času posledního vyhodnocení regulátoru a času mezi jeho dvěma vyhodnoceními. Čas mezi dvěma vyhodnoceními regulátoru je potřeba uchovávat zvlášť v milisekundách a v sekundách.

```
int samplingFrequency = 30; // frekvence vzorkování
```

```
unsigned long currentTime; // zaznamenání času začátku regulace
// zaznamenání času posledního vyhodnocení regulátoru
unsigned long lastEvaluationTime;
// čas mezi dvěma vyhodnoceními v milisekundách
int timeBetweenEvaluations;
// čas mezi dvěma vyhodnoceními v sekundách
int timeBetweenEvaluationsInSec;
// konstanta pro převod času z milisekund na sekundy
const int constTimeToSeconds = 0.001;
```

Dalším úsekem kódu je funkce *setup()*, kde se nastavuje frekvence sériové komunikace na hodnotu 9600 bitů za sekundu. Také jsou tady nastaveny rozlišení vstupních a výstupních digitálních signálu na hodnotu 12 bitů, což znamená, že hodnota vstupního nebo výstupního signálu bude v rozmezí od 0 do 4095. Dál následují nápis pro uživatele, aby viděl, že se regulace začala. Poslední dva příkazy slouží pro převod fyzikálních veličin na digitální hodnoty.

```
void setup() {
      // nastaví frekvenci sériové komunikace na hodnotu 9600 bitů
      // za sekundu
      Serial.begin(9600);
      // nastavení rozlišení vstupů Arduina na 12 bitů (0-4095)
      analogReadResolution(12);
      // nastavení rozlišení výstupů Arduina na 12 bitů (0-4095)
      analogWriteResolution(12);
      // nápis v sériovém monitoru
      Serial.println("Začátek regulace pneumatické pružiny");
      // převod žadané hodnoty v milimetrech na digitální hodnoty
      distanceW = round((distanceWinmm + 39,571) / 0.0591);
      // převod žadané hodnoty v kPa na digitální hodnoty
      pressureW = round((pressureWinkPa + 0,0161) / 0,0255);
}
```

V sekci *loop()* jsou zapsány příkazy, které se stále budou opakovat. Hodnotu regulované veličiny dostáváme z čidla vzdálenosti při každém kroku cyklu *loop()* pomocí funkce *analogRead()*. Parametrem této funkce je číslo pinu, ze kterého se nadchází digitální hodnota.

```
void loop() {
      // změna rozsahu digitálních hodnot na výstupu z čidla
      // vzdálenosti
      distanceY = analogRead(distanceSensorPin);
```

V dalším úseku kódu první příkaz slouží pro zaznamenávání nynějšího času. Dál se spočítá čas mezi dvěma vyhodnoceními odečtením času posledního vyhodnocení od aktuálního v daný okamžik. Třetí příkaz převede čas mezi vyhodnoceními regulátoru do sekund, protože tento čas bude potřeba použit právě v sekundách při výpočtu integrační složky PI-regulátoru.

```
      currentTime = millis(); // aktuální čas
      // výpočet času mezi dvěma vyhodnoceními regulátoru
      timeBetweenEvaluations = currentTime - lastEvaluationTime;
      // převod času mezi dvěma vyhodnoceními v sekundy
      timeBetweenEvaluationsInSec = timeBetweenEvaluations *
      constTimeToSeconds;
```

Dál následuje podmínka, která sleduje, zda čas mezi dvěma vyhodnoceními regulátoru přesahuje čas vzorkovací frekvence. Jinými slovy, pokud čas, uplynutý od posledního vyhodnocení, přesahuje 30 milisekund, což je časem vzorkovací frekvence, je potřeba udělat další výpočty. Jestli čas ještě neuplynul, program neudělá žádné výpočty a hned přeskočí na konec této podmínky. Uvnitř podmínky udělány výpočty pro regulační odchylku podle vzorce (5.1) a integrál, který pak bude použitý v integrační složce při výpočtu akční veličiny.

```
      // proces regulace se spustí, pokud čas mezi dvěma
      // vyhodnoceními je větší nebo rovná se vzorkovací frekvenci
if (samplingFrequency <= timeBetweenEvaluations) {
      // výpočet regulační odchylky
```

```
      distanceE = distanceW - distanceY;
      // výpočet integrálu, který je součástí integračního členu
      integral = integral + (distanceE *
      timeBetweenEvaluationsInSec);
```

Další dvě podmínky sledují, aby integrační složka nepřekročila meze -765 a 765. Pokud integrační složka přesahuje tyto meze, hodnota proměnné *integral* se nastaví na maximálně možnou v případě překročení horní meze nebo na minimálně možnou, v případě překročení dolní meze.

```
// kontrola aby integrační složka nepřekročila mez 765
if (integral * Ki > 765){
      // v případě překročení integrační složka se nastaví na
      // maximálně možnou hodnotu 765
      integral = 765/Ki;
}
      // kontrola aby integrační složka nepřekročila mez -765
if (integral * Ki < -765){
      // v případě překročení integrační složka se nastaví na
      // minimálně možnou hodnotu -765
      integral = -765/Ki;
}
```

Po ověření, zda integrační složka nepřesahuje meze, následuje výpočet akční veličiny podle vzorce (5.5) a její převod do rozsahu 0 – 4095.

```
// výpočet akční veličiny
distanceU = Kp * distanceE + Ki * integral;
// převod akční veličiny do rozsahu 0-4095
distanceU = round(distanceU * 5.35);
```

Když akční veličina je spočítaná, je také potřeba ověřit, zda nepřesahuje povolené meze. Zase, stejně jako v případě kontroly integrační složky, pokud akční veličina překročí horní mez, nastaví se na maximálně možnou hodnotu, pokud překročí dolní mez – na minimálně možnou. Po podmínkách do sériového monitoru se vypisují

nynější čas, hodnota regulované veličiny v milimetrech a digitální hodnota regulované veličiny.

```
// kontrola aby akční veličina nepřekročila mez 4095
if (distanceU > 4095){
      // v případě překročení akční veličina se nastaví na
      // maximálně možnou hodnotu 4095
      distanceU = 4095;
}
// kontrola aby akční veličina nepřekročila mez -4095
if (distanceU < -4095){
      // v případě překročení akční veličina se nastaví na
      // minimálně možnou hodnotu -4095
      distanceU = -4095;
}
// převod regulované veličiny z digitální hodnoty na milimetry
distanceYinmm = 0.0591 * distanceY - 39,571;
// vypisování nynějšího času, hodnoty regulované veličiny
// v milimetrech a digitální hodnoty regulované veličiny do
// sériového monitoru
Serial.print(currentTime);
Serial.print(",");
Serial.print(distanceY);
Serial.print(",");
Serial.print(distanceYinmm);
Serial.print(",");
Serial.println(distanceU);
```

No a v posledním úseku kódu se vyhodnocuje hodnota akční veličiny. V případě že je kladná, posílá se z Arduina na ventil jako druhý parametr funkce *analogWrite()*. Na místo prvního parametru funkce se píše proměnná pinu, ze kterého se tento signál posílá. Pokud hodnota akční veličiny je záporná, na ventil se pošle nula, totiž příkaz k

uzavření. Po podmínce poslední příkaz zaznamená čas vyhodnocení regulátoru zapsáním nynějšího času do proměnné *lastEvaluationTime* a program se skončí uzavřením složených závorek první podmínky, kontrolující zda uplynul čas pro další vyhodnocení, a cyklu *loop()*.

```
// zapisování hodnoty akční veličiny na výstup Arduina
if (distanceU >= 0){
      // pokud hodnota akční veličiny je větší nebo rovná se nule,
      // na ventil se zapíše její hodnota
      analogWrite(valvePin, distanceU);
}
else{
      // v jiném případě ventil se zavře
      analogWrite(valvePin, 0);
}
// uložení aktuálního času do proměnné pro záznam času posledního
// vyhodnocení regulátoru
lastEvaluationTime = currentTime;
}
}
```

7. Závěr

První část diplomové práce se zabývala popisem platformy Arduino, k čemuž patřilo seznámení se základními typy desek dostupnými od výrobce, instalace potřebného softwaru k programování desky – Arduino IDE, nastudování nejvíce používaných struktur programovacího jazyku [1], [2]. Jako řídící jednotka pro potřeby této práce byla zvolená deska Arduino M0 Pro, hlavně z toho důvodu, že je to jedna z několika desek, které poskytují analogové výstupy, což je zásadní, vzhledem k tomu, že byla potřeba řízení systému pomocí spojité regulace.

Dále následovala rešerše existujících typů ventilů a volba vyhovujícího pro tuto úlohu. Takovým ventilem se stal solenoidový proporcionální ventil ze série PVQ13 od výrobce SMC. Klíčovou charakteristikou tohoto ventilu je jeho proporcionalita, totiž přepínání do všech poloh z rozsahu povolených.

V následující části byly popsané další zařízení potřebné pro sestavení obvodů. K nim patří kompresor, splňující úkol zdroje vzduchu pro pneumatickou pružinu, a zvolené čidla tlaku a vzdálenosti, sloužící pro účely zpětné vazby v systému.

Dál následovalo nastudování základů teorie řízení, zejména její časti o spojité regulaci [3]. Seznámil jsem se s nejpoužívanějšími regulátory [4], a zvolil jsem PI-regulátor vzhledem k jeho dostatečné přesnosti a jednoduchému nastavení parametrů.

V poslední části diplomové práce byl realizován kompletní systém prostřednictvím zapojení všech prvků pneumatického a elektrického obvodu. Toto zapojení je znázorněné a popsané v jednotlivých kapitolách. Vzhledem k tomu, že ventil nebylo možné připojit přímo na analogový výstup desky Arduino, byl vytvořen převodník, sloužící pro zesílení signálu posílaného z desky. Posledním úkolem této části práce bylo sestavení programu na základě zvoleného PI-regulátoru, který by zrealizoval algoritmus řízení systému, čehož se taky úspěšně podařilo dosáhnout.

Tím pádem se podařilo vytvořit jednoduchý výukový model, sestavený ze všech komponent a softwaru uvedeného výše, znázorňující princip práce pneumatického systému, přesněji řečeno regulovaný proces nafukování pneumatické pružiny. Řídicí systém je po nepatrných úpravách možno aplikovat na řízení tlaku v pneumatickém

prvku sedačky s proměnnou tuhostí, řízení tlaku v pneumatických členech ortopedické matrace nebo jiném pneumatickém systému.

Použitá literatura

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

[2] NUSSEY, John. Arduino for dummies. West Sussex, England: Wiley, c2013. ISBN 978-1-118-44643-0.

[3] BALÁTĚ, Jaroslav. Automatické řízení. Praha: BEN - technická literatura, 2003. ISBN 80-7300-020-2.

[4] Spojité typy regulátorů [online]. [cit. 2017-04-21]. Dostupné z: http://195.178.94.43/CAAC_PHP/CAAC/cesky/synteza/s_reg/s_reg.php

[5] Arduino M0 PRO [online]. [cit. 2017-04-21]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoBoardM0PRO

[6] Arduino Products [online]. [cit. 2017-04-21]. Dostupné z: https://www.arduino.cc/en/Main/Products

[7] BD6xxx Transistors Datasheet [online]. [cit. 2017-04-21]. Dostupné z: http://www.st.com/content/ccc/resource/technical/document/datasheet/1f/51/a4/6a/68/84/45/a6/CD00000939.pdf/files/CD00000939.pdf/jcr:content/translations/en.CD00000939.pdf

[8] CA3140, CA3140A Operational Amplifiers Datasheet [online]. [cit. 2017-04-21]. Dostupné z: https://www.intersil.com/content/dam/Intersil/documents/ca31/ca3140-a.pdf

[9] PVQ Series Valves Datasheet [online]. [cit. 2017-04-21]. Dostupné z: http://www.smclt.lt/failai/Proportional_Valve_Series_PVQ1222728689__1.pdf

[10] PVQ13-6M-03-M5-A Solenoid Valve [online]. [cit. 2017-04-21]. Dostupné z: http://it.rs-online.com/web/p/elettrovalvole/2550745970/

[11] GP2Y0A41SK0F Distance Sensor Datasheet [online]. [cit. 2017-04-21]. Dostupné z: https://www.pololu.com/file/download/GP2Y0A41SK0F.pdf?file_id=0J713

[12] DRMOD-I2C Pressure Sensors Datasheet [online]. [cit. 2017-04-21]. Dostupné z: https://www.rapidonline.com/pdf/502447_da_en_01.pdf

[13] AIRBRUSH COMPRESSOR SALON AIR [online]. [cit. 2017-04-21]. Dostupné z: http://www.tiendaracingcolors.com/Airbrush-Compressor-Salon-Air

[14] MAROUSEK, Petr. Řízení pneumatických ventilů pomocí vývojového kitu Arduino. Liberec: TU v Liberci - KVS, 2016.

[15] NOVOTNÝ, Tomáš. Řízení tuhosti ortopedické matrace. Liberec: TU v Liberci - KVS, 2016.

Seznam obrázků

Seznam tabulek

Seznam příloh

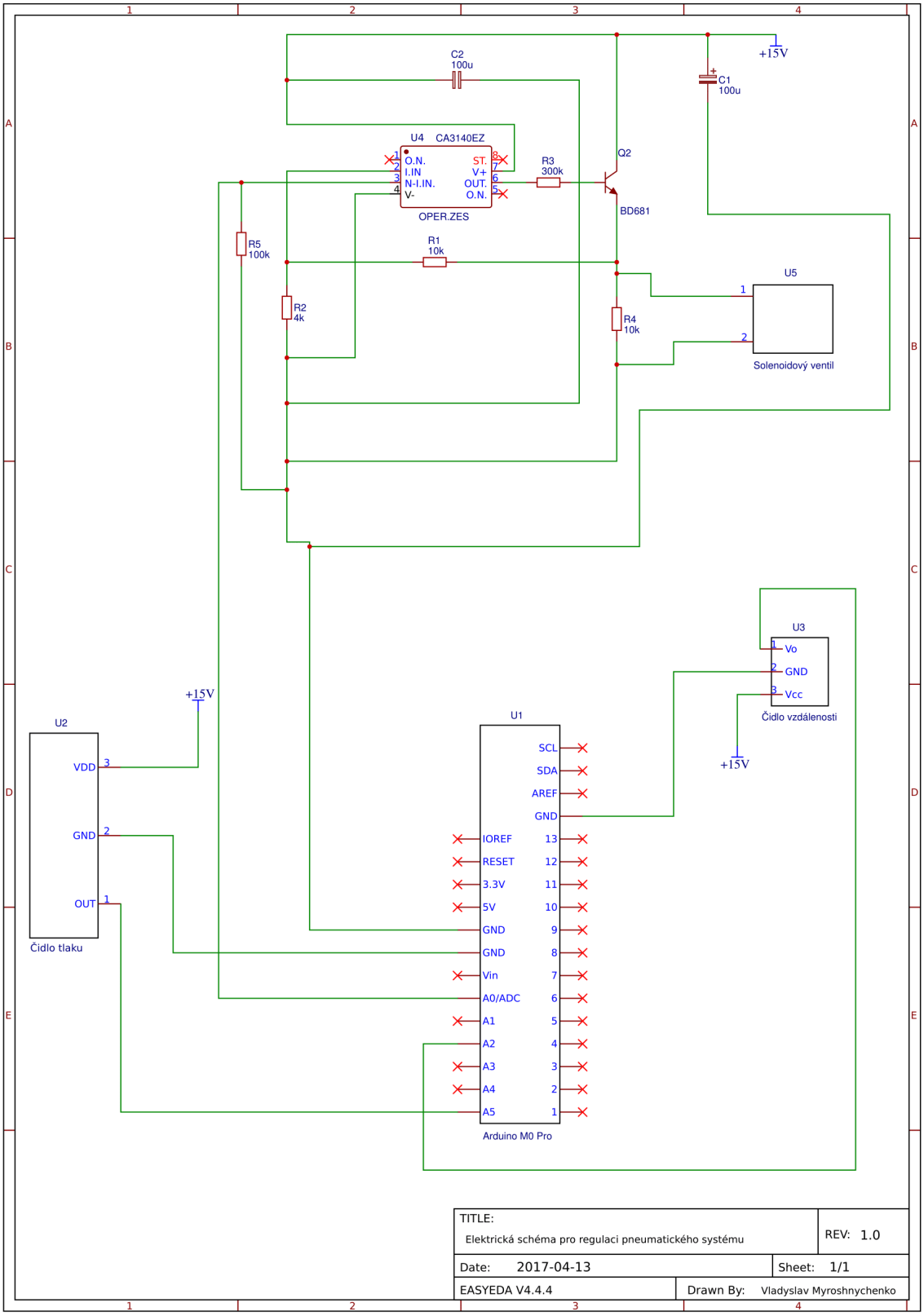
Příloha č. 1: Kompletní elektrické schéma zapojení všech prvků

Příloha č. 2: Kód programu pro regulaci systému

Obsah CD:

- ✓ Text práce (Diplomova_Prace_Vladyslav_Myroshnychenko_2017.pdf)
- ✓ Kód programu ve formátu .ino a .txt

Příloha č. 1: Kompletní elektrické schéma zapojení všech prvků



Příloha č. 2: Kód programu pro regulaci systému

```
// pin pro ovládání ventilu pro přivádění vzduchu
byte valvePin = A0;
byte distanceSensorPin = A2; // pin pro výstup z čidla vzdálenosti
byte pressureSensorPin = A5; // pin pro výstup z čidla tlaku
double pressureE; // digitální hodnota regulační odchylky pro tlak
double distanceE; // digitální hodnota regulační odchylky pro
výšku
double pressureW; // digitální hodnota žádané veličiny pro tlak
double distanceW; // digitální hodnota žádané veličiny pro výšku
double distanceWinmm = 50; // žádaná hodnota výšky v milimetrech
double pressureWinkPa = 0; // žádaná hodnota tlaku v kPa
// digitální hodnota regulované veličiny pro tlak
double pressureY;
// digitální hodnota regulované veličiny pro výšku
double distanceY;
// hodnota regulované veličiny pro tlak v kPa
double pressureYinkPa;
// hodnota regulované veličiny pro výšku v milimetrech
double distanceYinmm;
double pressureU; // digitální hodnota akční veličiny pro tlak
double distanceU; // digitální  hodnota akční veličiny pro výšku
const double Kp = 8.2; // proporcionální člen
const double Ki = 5.4; // integrační člen
double integral = 0; // integrační složka regulátora
int samplingFrequency = 30; // frekvence vzorkování
unsigned long currentTime; // zaznamenání času začátku regulace
// zaznamenání času posledního vyhodnocení regulátoru
unsigned long lastEvaluationTime;
// čas mezi dvěma vyhodnoceními v milisekundách
int timeBetweenEvaluations;
```

```
// čas mezi dvěma vyhodnoceními v sekundách
int timeBetweenEvaluationsInSec;
// konstanta pro převod času z milisekund na sekundy
const int constTimeToSeconds = 0.001;
void setup() {
      // nastaví frekvenci sériové komunikace na hodnotu 9600 bitů
      // za sekundu
      Serial.begin(9600);
      // nastavení rozlišení vstupů Arduina na 12 bitů (0-4095)
      analogReadResolution(12);
      // nastavení rozlišení výstupů Arduina na 12 bitů (0-4095)
      analogWriteResolution(12);
      // nápis v sériovém monitoru
      Serial.println("Začátek regulace pneumatické pružiny");
      // převod žadané hodnoty v milimetrech na digitální hodnoty
      distanceW = round((distanceWinmm + 39,571) / 0.0591);
      // převod žadané hodnoty v kPa na digitální hodnoty
      pressureW = round((pressureWinkPa + 0,0161) / 0,0255);
}
void loop() {
      // změna rozsahu digitálních hodnot na výstupu z čidla
      // vzdálenosti
      distanceY = analogRead(distanceSensorPin);
      currentTime = millis(); // aktuální čas
      // výpočet času mezi dvěma vyhodnoceními regulátoru
      timeBetweenEvaluations = currentTime - lastEvaluationTime;
      // převod času mezi dvěma vyhodnoceními v sekundy
      timeBetweenEvaluationsInSec = timeBetweenEvaluations *
      constTimeToSeconds;
      // proces regulace se spustí, pokud čas mezi dvěma
      // vyhodnoceními je větší nebo rovná se vzorkovací frekvenci
```

```
if (samplingFrequency <= timeBetweenEvaluations) {
      // výpočet regulační odchylky
      distanceE = distanceW - distanceY;
      // výpočet integrálu, který je součástí integračního členu
      integral = integral + (distanceE *
      timeBetweenEvaluationsInSec);
      // kontrola aby integrační složka nepřekročila mez 765
if (integral * Ki > 765){
      // v případě překročení integrační složka se nastaví na
      // maximálně možnou hodnotu 765
      integral = 765/Ki;
}
// kontrola aby integrační složka nepřekročila mez -765
if (integral * Ki < -765){
      // v případě překročení integrační složka se nastaví na
      // minimálně možnou hodnotu -765
      integral = -765/Ki;
}
// výpočet akční veličiny
distanceU = Kp * distanceE + Ki * integral;
// převod akční veličiny do rozsahu 0-4095
distanceU = round(distanceU * 5.35);
// kontrola aby akční veličina nepřekročila mez 4095
if (distanceU > 4095){
      // v případě překročení akční veličina se nastaví na
      // maximálně možnou hodnotu 4095
      distanceU = 4095;
}
// kontrola aby akční veličina nepřekročila mez -4095
if (distanceU < -4095){
```

```
      // v případě překročení akční veličina se nastaví na
      // minimálně možnou hodnotu -4095
      distanceU = -4095;
}
// převod regulované veličiny z digitální hodnoty na milimetry
distanceYinmm = 0.0591 * distanceY - 39,571;
// vypisování nynějšího času, hodnoty regulované veličiny
// v milimetrech a digitální hodnoty regulované veličiny do
// sériového monitoru
Serial.print(currentTime);
Serial.print(",");
Serial.print(distanceY);
Serial.print(",");
Serial.print(distanceYinmm);
Serial.print(",");
Serial.println(distanceU);
// zapisování hodnoty akční veličiny na výstup Arduina
if (distanceU >= 0){
      // pokud hodnota akční veličiny je větší nebo rovná se nule,
      // na ventil se zapíše její hodnota
      analogWrite(valvePin, distanceU);
}
else{
      // v jiném případě ventil se zavře
      analogWrite(valvePin, 0);
}
// uložení aktuálního času do proměnné pro záznam času posledního
// vyhodnocení regulátoru
lastEvaluationTime = currentTime;
}}
```