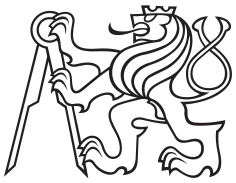
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Vzduchová levitace dvou ping-pongových míčků

Jaroslav Klapálek

Vedoucí: Ing. Jiří Zemánek
Obor: Systémy a řízení
Studijní program: Kybernetika a robotika
Květen 2017

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Jiřímu Zemánkovi za jeho cenné rady a odborné konzultace, které mi ve vypracování této práce velmi pomohly.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 26. května 2017

Abstrakt

Obsahem bakalářské práce je návrh a realizace systému pro vzduchovou levitaci dvou ping-pongových míčků v trubici. Tento systém se skládá z trubice, větráku, senzorů polohy a řídicího systému. Pro výběr parametrů jednotlivých součástí a pro účely návrhu řídicího systému byl vytvořen matematický model, jehož parametry byly vypočteny, popř. identifikovány experimentálně. Řídicí systém ovládá rychlost otáčení větráku a polohu míčku. Jeho implementace je provedena na platformě Arduino. Součástí práce je i dokumentace experimentálního ověření funkčnosti celého systému.

Klíčová slova: vzduchová levitace, levitace, trubice, větrák, senzor polohy, ultrazvuk, arduino, řízení

Vedoucí: Ing. Jiří Zemánek

Abstract

This thesis contains designing and assembling a system capable of air levitation of multiple ping-pong balls in a single tube. System is composed of an air fan, a tube, range sensors and a control system. Using mathematical model of system parameters for control system design are defined. Fan speed and ball position are manipulated by control system which is implemented on Arduino platform. Experimental verification of whole system is documented.

Keywords: air levitation, levitation, tube, fan, range sensor, ultrasonic, arduino, control

Title translation: Air levitation of two ping-pong balls

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Tato práce je zaměřena na vytvoření systému pro vzduchovou levitaci ping-pongových míčků v trubici. Jeho součástí není jen vytvoření reálného modelu, ale také návrh řídicího systému.

Reálný model se skládá z větráku, trubice a dvou senzorů polohy, které jsou umístěny na koncích trubice tak, aby měřily polohu míčků uvnitř. Větrák a senzory polohy jsou v práci podrobněji rozebírány, včetně identifikace a modelace.

Řídicí systém, který je implementován na platformě Arduino, je vytvořen pro regulaci polohy míčků a je také schopen řízení rychlosti otáčení větráku.

Cílem práce je řízení polohy jednoho (popř. dvou) ping-pongových míčků. Výsledný model by mohl být rozšířen na větší počet trubic a sloužit jako exponát na chodbách školy k motivaci budoucích studentů, popř. jako další úloha pro předmět Automatické řízení [Če].

Kapitola 2
Problematika

Problém vzduchové levitace je všeobecně známý a lze na něj (v různých obměnách) najít velké množství prací. Žádná z nich neřeší levitaci dvou ping-pongových míčků v trubici. Nejblíže mému zadání je [CQ], což je zároveň práce, ze které vycházím při vytváření matematického modelu, protože tam se modelování věnují. Práce [EOR05] modelování pouze naznačuje, ale dále ho nerozvádí.

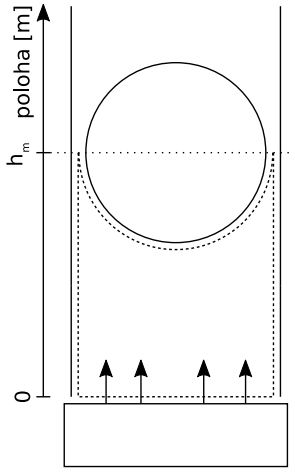
Vzduchová levitace míčku v trubici se také využívá během vyučování. Příkladem může být dynamika tekutin [HBB15] nebo řízení systémů [Ver] a [But].

Výsledky projektů jsou pak k dohledání na serveru YouTube, kam je dávají sami studenti – např. [Eli], [Ucu], [Smi] a [Gue].

Kapitola 3

Matematický model

Situaci v trubici popíši dvěma rovnicemi - jedna bude vztažena k silám působících na míček, druhá k silám působícím na vzduchový válec pod míčkem. Situace je znázorněna na obrázku (3.1). Rozbor sil působících v trubici je založen na [CQ].



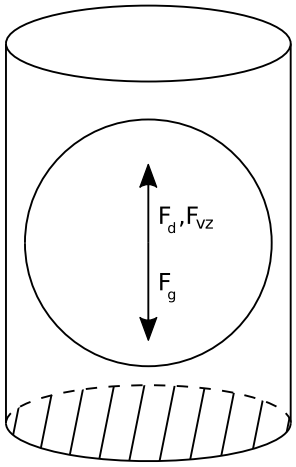
Obrázek 3.1: Situace v trubici

Míček

Na míček v trubici působí tři síly:

1. gravitační síla $F_g = m_m g$,
2. odporová síla $F_d = \frac{1}{2}\rho(v - v_m)^2 C_d S_m$,
3. vztlaková síla $F_{vz} = V_m \rho g$,

kde m_m [kg] je hmotnost míčku, g [m/s^2] gravitační zrychlení, ρ [kg/m^3] hustota vzduchu, v [m/s] rychlost proudění okolí vzhledem k míčku, v_m [m/s] rychlost míčku, C_d [-] součinitel odporu prostředí, S_m [m^2] plocha průřezu míčku a V_m [m^3] objem míčku. Síly jsou znázorněny na obrázku 3.2. Dále předpokládám směr pohybu nahoru za kladný.



Obrázek 3.2: Znázornění sil působících na míček

Odporová síla. Odporová síla je aerodynamickou mechanickou silou, která působí proti pohybu objektu ve vzduchu. Tato síla je generována rozdílem rychlosti mezi objektem a jeho okolím. Její velikost je závislá na součiniteli odporu prostředí, jehož hodnota je určena Reynoldsovým číslem (viz [Halb]). Pro míček lze použít součinitel platný pro ideální kouli (viz [Eleb]):

$$C_d \doteq 0{,}5. \tag{3.1}$$

Výsledná síla. Výslednou silou působící na míček je součet gravitační, odporové a vztlakové síly:

$$F_m = \sum F = F_d + F_{vz} - F_g. \tag{3.2}$$

Vzduchový válec

Na vzduchový válec pod míčkem působí tyto síly:

1. tlaková síla $F_t = S_t p$,
2. hydrostatická tlaková síla $F_h = h_m g \rho S_t$ neboli síla způsobena hydrostatickým tlakem,
3. síla způsobená míčkem F_m podle rovnice (3.2),
4. síla způsobená otáčením větráku F_v,

kde $S_t\,[\mathrm{m}^2]$ je plocha průřezu trubice (vnitřní) a $h_m\,[\mathrm{m}]$ poloha míčku. V rámci zjednodušení zanedbávám prostor mezi míčkem a trubicí při výpočtu síly.

Tlaková síla. Pod pojmem tlaková síla se myslí především síla způsobena rozdílným tlakem mezi jednotlivými konci válce. Proto budu dále za tlakovou sílu považovat:

$$F_t = S_v p_v - S_t p_m, \tag{3.3}$$

kde p_v [Pa] je tlak u větráku, p_m [Pa] tlak pod míčkem a S_v [m^2] je plocha větráku.

Síla způsobena větrákem. Pro vyjádření síly, kterou působí do trubice větrák, využiji rovnici kontinuity, resp. zákon o zachování hmoty pro mechaniku tekutin (viz [Hala]):

$$\dot{m} = \rho v S = \text{konst.}, \tag{3.4}$$

kde $\dot{m}$ [kg/s] je hmotnostní tok, ρ [kg/m^3] hustota tekutiny, v [m/s] rychlost toku tekutiny a S [m^2] plocha průřezu prostoru, kterým se tekutina pohybuje.

Síla se dá vyjádřit jako derivace hybnosti, tedy:

$$F = \frac{dp}{dt} = \frac{d(mv)}{dt} = \frac{dm}{dt}v + m\frac{dv}{dt}, \tag{3.5}$$

přičemž za předpokladu, že v daném okamžiku bude rychlost toku konstantní, lze rovnici (3.5) přepsat:

$$F = \frac{dm}{dt}v = \dot{m}\, v. \tag{3.6}$$

Dosazením rovnice (3.6) do rovnice (3.4) a vyjádřením síly získám rovnici pro F_v – sílu, kterou větrák působí na vzduchový válec:

$$F_v = \rho v^2 S_t. \tag{3.7}$$

Výsledná síla. Síla, kterou působí větrák na vzduchový válec, se rozloží do ostatních sil:

$$F_v = \sum F = F_h - F_t - F_m. \tag{3.8}$$

Celý systém

V momentě, kdy se míček nehýbe, jsou síly působící na míček a síly působící na vzduchový válec v rovnováze. Dosazením rovnice (3.2) do rovnice (3.8) vznikne rovnice:

$$F_d + F_{vz} - F_g = F_h - F_t - F_v, \tag{3.9}$$

po dosazení příslušných rovnic za jednotlivé síly:

$$\frac{1}{2}\rho(v - v_m)^2 C_d S_m + V_m \rho g - m_m g = h_m g \rho S_t - (S_v p_v - S_t p_m) - \rho v^2 S_t, \tag{3.10}$$

kde jsou jedinými proměnnými rychlost vzduchu v, rychlost míčku v_m, tlaky p_v a p_m a nakonec poloha míčku h_m.

Za předpokladu, že rozdíl mezi plochou průřezu trubice a plochou větráku je minimální, mohu tlakovou sílu F_t vyjádřit jako:

$$F_t = S_t(p_v - p_m) = S_t \Delta p, \tag{3.11}$$

kde Δp [Pa] je tlakový rozdíl mezi začátkem trubice a místem, kde se nachází míček. Z rovnice (3.10) si vyjádřím rychlost v, kterou musí mít tok vzduchu, aby se míček nehýbal (rychlost míčku je tedy nulová):

$$v = \sqrt{\frac{h_m g \rho S_t - S_t \Delta p - V_m \rho g + m_m g}{0{,}5 \rho C_d S_m + \rho S_t}}. \tag{3.12}$$

Poloha míčku. V rovnici (3.12) se ve jmenovateli objevuje rozdíl sil F_t a F_h. S rostoucí výškou se zvyšuje i tento člen, tedy rychlost potřebná k udržení míčku se snižuje. Z tohoto důvodu je pro samotné zvednutí míčku potřeba vyšší rychlost vzduchu než pro samotné udržení v trubici.

Z rovnice (3.12) lze vypočítat rychlost vzduchu, která je potřebná pro zvednutí míčku. Protože se tato rychlost s rostoucí výškou snižuje, jedná se zároveň o její potřebné maximum. Za předpokladu, že je míček v nejspodnější poloze (tedy síly F_t a F_h jsou nulové), lze rovnici upravit a tím zjednodušit výpočet:

$$v = \sqrt{\frac{m_m g - V_m \rho g}{0{,}5 \rho C_d S_m + \rho S_t}} \doteq 3{,}353\,\mathrm{m/s} \tag{3.13}$$

Z rovnice (3.13) se dá určit požadovaná objemová rychlost, kterou musí větrák vygenerovat:

$$Q = v S_t \doteq 5{,}098 \cdot 10^{-3}\,\mathrm{m^3/s} = 18{,}352\,\mathrm{m^3/h}. \tag{3.14}$$

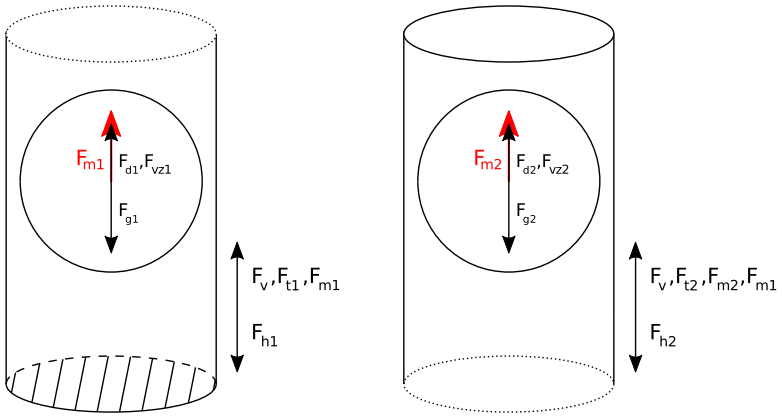
Dva míčky

V případě, kdy by v trubici byly dva míčky, se dá silové působení rozdělit na čtyři části:

- síly působící ve vzduchovém válci pod prvním míčkem,
- síly působící na první míček (ten níže),
- síly působící ve vzduchovém válci mezi míčky,
- síly působící na druhý míček.

V následujících rovnicích jsou číselnými indexy rozlišeny síly a parametry jednotlivých válců a míčků – spodní mají index 1, horní 2 (viz obrázek 3.3).

První vzduchový válec s míčkem. Zde jsou rovnice stejné jako v předchozím případě. Síly působící na míček jsou v rovnici (3.2), síly působící ve vzduchovém válci v rovnici (3.8).



Obrázek 3.3: Znázornění sil působících na míčky

Druhý vzduchový válec. Síly, které působí na první vzduchový válec, působí i na válec druhý. Navíc přibyla síla druhého míčku F_{m2}:

$$F_v = F_{h2} - F_{t2} - F_{m1} - F_{m2}, \tag{3.15}$$
$$F_{h2} = h_{m2} g \rho S_t, \tag{3.16}$$
$$F_{t2} = (p_v - p_{m2}) S_t = S_t \Delta p_2, \tag{3.17}$$

kde síla F_v způsobena otáčením větráku je stejná jako pro první vzduchový válec. Toto tvrzení platí za předpokladu, že míčky jsou od sebe dostatečně vzdáleny – pak je tedy i podle rovnice (3.4) rychlost stejná. O tom se zmiňuje i [Sou06].

Druhý míček. Rovnice pro druhý míček obsahuje stejné síly jako první míček s tím rozdílem, že jsou vztaženy k druhému vzduchovému válci:

$$F_{m2} = F_{d2} + F_{vz2} - F_{g2}, \tag{3.18}$$
$$F_{d2} = \frac{1}{2} \rho (v - v_{m2})^2 C_{d2} S_{m2}, \tag{3.19}$$
$$F_{vz2} = V_{m2} \rho g, \tag{3.20}$$
$$F_{g2} = m_{m2} g, \tag{3.21}$$

kde pro sílu F_{d2}, resp. pro rychlost prostředí v platí stejný předpoklad jako u rovnice (3.15).

Ustálení druhého míčku. V momentě, kdy se druhý míček nehýbe, musí být (stejně jako v případě pro jeden míček) síly působící na druhý míček v rovnováze se silami ve vzduchovém válci – pro jeden míček je situace popsána rovnicí (3.9), pro druhý je velice podobná. Vznikne spojením rovnice (3.15) a rovnice (3.18):

$$F_{d2} + F_{vz2} - F_{g2} = F_{h2} - F_{t2} - F_v - F_{m1}, \tag{3.22}$$

po dosazení síly F_{m1}:

$$F_{d2} + F_{vz2} - F_{g2} = F_{h2} - F_{t2} - F_v - F_{d1} - F_{vz1} + F_{g1}, \tag{3.23}$$

ze které vyjádřím člen v – tedy rychlost prostředí, kterou by musel proud vzduchu dosáhnout, aby byly síly v rovnováze a míček držel svou polohu:

$$v = \sqrt{\frac{F_{h2} - F_{t2} - F_{vz1} + F_{g1} - F_{vz2} + F_{g2}}{0{,}5\rho C_{d1} S_{m1} + 0{,}5\rho C_{d2} S_{m2} + \rho S_t}}. \tag{3.24}$$

Polohy míčků. Aby mohly být oba dva míčky ustáleny (tedy držely stabilní polohu), musí se rovnice (3.12) a (3.24) rovnat:

$$\sqrt{\frac{F_{h1} - F_{t1} - F_{vz1} + F_{g1}}{0{,}5\rho C_{d1} S_{m1} + \rho S_t}} = \sqrt{\frac{F_{h2} - F_{t2} - F_{vz1} + F_{g1} - F_{vz2} + F_{g2}}{0{,}5\rho C_{d1} S_{m1} + 0{,}5\rho C_{d2} S_{m2} + \rho S_t}}, \tag{3.25}$$

kde pro zjednodušení budu předpokládat, že oba míčky jsou pingpongové – mají tedy kulovitý tvar, stejný poloměr a tudíž i obsah průřezu a objem:

$$C_{d1} = C_{d2} = C_d, \tag{3.26}$$
$$S_{m1} = S_{m2} = S_m, \tag{3.27}$$
$$V_{m1} = V_{m2} = V_m, \tag{3.28}$$

z čeho je zřejmá i rovnost vztlakových sil $F_{vz1} = F_{vz2} = F_{vz}$. Upravením rovnice (3.25) získám:

$$\frac{F_{h1} - F_{t1} - F_{vz} + F_{g1}}{0{,}5\rho C_d S_m + \rho S_t} = \frac{F_{h2} - F_{t2} - 2F_{vz} + F_{g1} + F_{g2}}{\rho C_d S_m + \rho S_t}. \tag{3.29}$$

Nyní zavedu substituci za některé členy:

$$K_1 = 0{,}5\rho C_d S_m + \rho S_t, \tag{3.30}$$
$$K_2 = \rho C_d S_m + \rho S_t \tag{3.31}$$

a tyto konstanty dosadím do rovnice (3.29):

$$\frac{F_{h1} - F_{t1} - F_{vz} + F_{g1}}{K_1} = \frac{F_{h2} - F_{t2} - 2F_{vz} + F_{g1} + F_{g2}}{K_2}. \tag{3.32}$$

ze které nakonec vyjádřím sílu F_{g2}:

$$F_{g2} = (F_{h1} - F_{t1} - F_{h2} + F_{t2})K_{21} + (2 - K_{21})F_{vz} + (K_{21} - 1)F_{g1}, \tag{3.33}$$

kde $K_{21} = \frac{K_2}{K_1}$. Aby byly oba míčky ustáleny, musí hmotnost druhého míčku splňovat podmínku v rovnici (3.33). Z této rovnice plyne, že hmotnost je různá pro rozdílné výšky, rozdílné tlaky a nakonec závisí i na hmotnosti prvního míčku.

Za předpokladu, že účinek větráku na změnu tlaku v trubici je konstantní, jsou tlakové rozdíly obsažené v síle F_t závislé pouze na poloze (resp. rozdílu atmosférických tlaků v daných polohách). Rovnici (3.33) lze poté přepsat takto:

$$F_{g2} = (2 - K_{21})F_{vz} + (K_{21} - 1)F_{g1}, \tag{3.34}$$

protože síly F_h a F_t se odečtou:

$$F_h - F_t = hg\rho S - S\Delta p = hg\rho S - Shg\rho = 0. \tag{3.35}$$

Podle rovnice (3.34) musí pro hmotnost m_{m2} platit:

$$m_{m2} = (2 - K_{21})V_m\rho + (K_{21} - 1)m_{m1}, \tag{3.36}$$

což při hmotnosti prvního míčku $m_{m1} = 2{,}2 \cdot 10^{-3}\,\mathrm{kg}$ vychází:

$$m_{m2} \doteq 4{,}722 \cdot 10^{-4}\,\mathrm{kg} = 0{,}472\,\mathrm{g}, \tag{3.37}$$

což ping-pongový míček nemá. Zároveň se jedná o maximální hmotnost pro druhý míček, protože bez zmíněného předpokladu by se projevila i změna tlaku způsobená větrákem a výsledná hmotnost by byla tudíž nižší. Z tohoto důvodu předpokládám, že řízená levitace dvou ping-pongových míčků v trubici za použití jednoho větráku není možná.

Kapitola 4
Součásti modelu

Tato kapitola popisuje jednotlivé součásti, ze kterých se model systému skládá.

4.1 Míček

Typ míčku	ping-pongový
Poloměr	20 mm
Hmotnost	2,2 g
Materiál	plast
Barva	oranžová

Tabulka 4.1: Parametry použitého míčku

4.2 Trubice

Délka	2 m
Vnitřní poloměr	22 mm
Vnější poloměr	25 mm
Materiál	plexislo
Barva	čirá

Tabulka 4.2: Parametry použité trubice

4.3 Arduino Uno

Mikrokontrolérová vývojová deska Arduino Uno zajišťuje měření veličin v systému a jeho regulaci. Deska je postavena na procesoru Atmel ATMega328P o taktu 16 MHz. Pro komunikaci s okolím obsahuje deska 14 digitálních vstupních/výstupních pinů (z toho jich 6 podporuje PWM signály) a 6 vstupních analogových pinů.

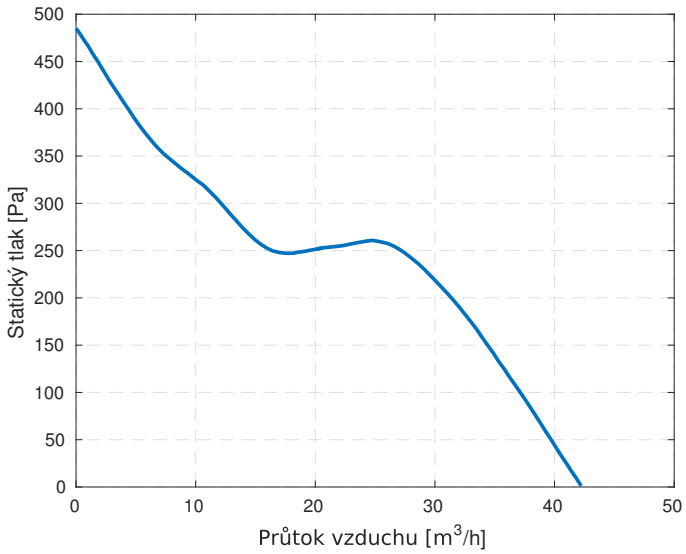
Ve výsledném modelu systému je použit klon této desky. Jeho vlastnosti jsou totožné s originálem.

4.4 Větrák

Větrák SUNON PF40281B1-000U-S99 použitý v modelu systému má tyto vlastnosti (viz [SUN12]):

Typ větráku	DC
Druh větráku	axiální
Rozměry	$40 \times 40 \times 28\,\text{mm}$
Počet vývodů	4
Napájecí napětí	12 V
Pracovní napětí	10,2 13,2 V
Jmenovitý proud	510 mA
Rychlost otáčení	17600($\pm$10%) ot./min
Průtok vzduchu	24,9 CFM (42,3 m^3/h)
Statický tlak	1,95 Inch-H_2O (485,7 Pa)
Akustický hluk	56 dB(A)

Tabulka 4.3: Parametry použitého větráku



Obrázek 4.1: Závislost statického tlaku na průtoku vzduchu (převzato z [SUN12, str. 5])

4.5 Senzory polohy

Pro měření polohy míčku (míčků) jsou použity ultrazvukové senzory HC-SR04, které mají tyto parametry (viz [ITe10] a [Elea]):

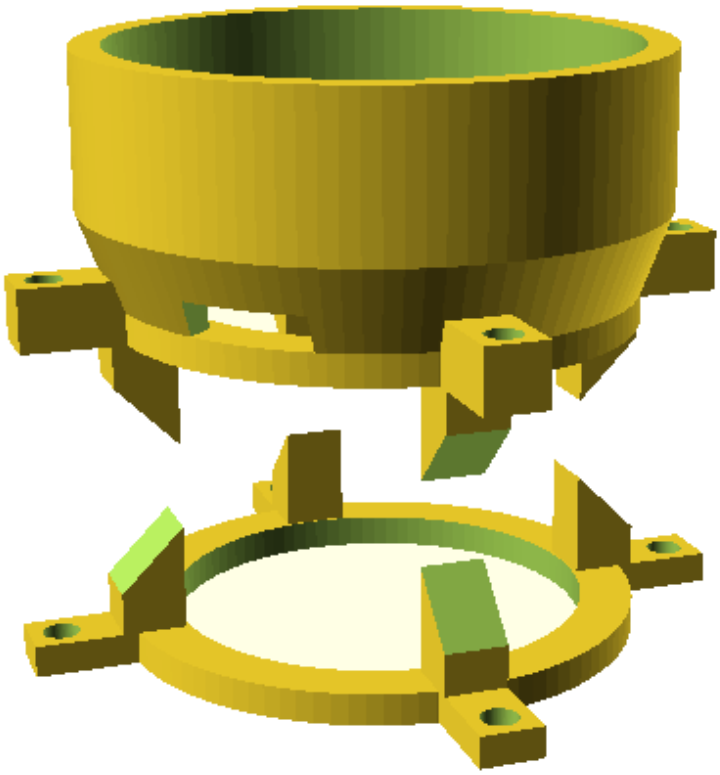
Rozměry	$45 \times 20 \times 15\,\text{mm}$
Pracovní napětí	5 V DC
Klidový proud	$< 2\,\text{mA}$
Jmenovitý proud	15 mA
Minimální vzdálenost	2 cm
Maximální vzdálenost	4–5 m
Měřící úhel	$< 15°$

Tabulka 4.4: Parametry použitého ultrazvukového senzoru

Maximální vzdálenosti se mezi jednotlivými dokumentacemi liší. Protože je ale maximální měřená vzdálenost v soustavě do 2 m, což je hodnota, kterou ani jedna z dokumentací nevylučuje, tak na reálné velikosti tohoto parametru nezáleží.

4.6 Spojovací materiály

Veškeré spojovací díly pro model byly vytištěny na 3D tiskárně ve škole. Pro návrh byl použit program OpenSCAD (viz [K+]). Výkresy spojovacích dílů jsou v příloze B.



Obrázek 4.2: Nástavec pro větrák s otvorem pro ultrazvukový senzor

Kapitola 5
Větrák

Větrák je stejnosměrný motor s lopatkovým kolem.Otáčením větráku vzniká proud vzduchu, který nadnáší míček.

5.1 Dělení větráků

Větráky se dělí do několika skupin podle směru průtoku vzduchu na: axiální (*axial*), radiální (*radial/centrifugal*), „smíšené“ (*mixed*) a tangenciální (*cross-flow/tangential*) [Mar07, kap. 14.5]. Běžně dostupné jsou první dva zmíněné – axiální a radiální.

5.2 Zapojení

Minimální počet přívodních vodičů větráku je dva. Tyto vodiče jsou využity pro napájení (standardně 12 V stejnosměrných). Třípinový větrák má vývod na měření otáček, čtyřpinový větrák má ještě navíc vývod pro řízení rychlosti otáčení.

5.2.1 Měření rychlosti otáčení

Třípinové a čtyřpinové větráky mají pin, který se dá využít pro měření otáček. Dvoupinové větráky touto funkcí nedisponují, proto je dále nebudu uvažovat. Avšak kdyby byly dvoupinové větráky jedinou možností, lze měření otáček dosáhnout přidaným čidlem.

Větrák vyšle na třetí pin pulz každou půlotáčku. Na měřící straně musí být pull-up rezistor na 12 V (viz [Int04]).

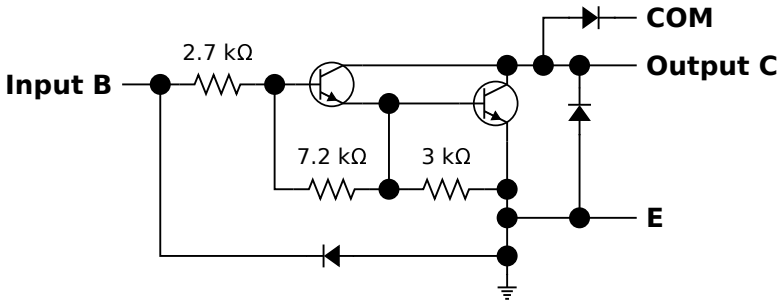
5.2.2 Řízení rychlosti otáčení

Možnosti řízení rychlosti otáčení větráku jsou závislé na počtu pinů.

Třípinové větráky

U třípinových větráků se dá rychlost otáčení řídit „přerušováním“ obvodu za pomocí tranzistoru.

Řízení tranzistorem. Zapojení s tranzistorem umožňuje PWM signálem (přivedeným na jeho bázi) postupně spínat a rozepínat obvod a tím regulovat rychlost otáčení větráku. Při realizaci zapojení jsem využil tzv. Darlingotovo zapojení dvou tranzistorů, které je obsaženo v integrovaném obvodu řady ULN2803.



Obrázek 5.1: Darlingtonovo zapojení tranzistorů v IO ULN2803A (podle [Tex04, str. 2]).

Na obrázku 5.1 je zobrazeno zapojení v integrovaném obvodu ULN2803A. Napájení pro větrák se zapojí na výstupy **COM** a **E** (GND). Přívodní vodiče větráku se zapojí na výstupy **C** a **COM**. Ovládací signál PWM se připojí na vstup **B**.

Čtyřpinové větráky

Čtyřpinové větráky obsahují vestavěné tranzistory pro ovládání rychlosti otáčení. Čtvrtý vodič je zapojen do báze tranzistoru.

Čtyřpinový větrák se řídí PWM signálem o frekvenci 25 kHz s maximálním napětím 5.25 V a proudem 5 mA (viz [Int04]).

5.3 Výběr větráku

Podle znalostí z 5.1 a 5.2.1 jsem se rozhodl vybírat mezi axiálními větráky se čtyřmi přívodními vodiči.

Během hledání vhodného větráku jsem zjistil, že běžně dostupné větráky nejsou pro danou aplikaci dostačující – důvodem byl nízký statický tlak. Experimentálně jsem zvolil větrák PF40281B1-000U-S99 od společnosti SUNON, jehož statický tlak je 485,7 Pa.

5.4 Charakteristika

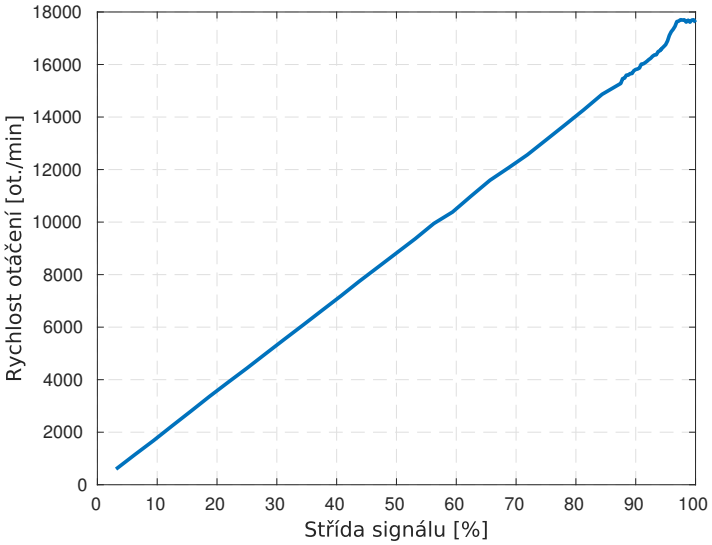
S dostupným měřením a ovládáním rychlosti otáčení mohu změřit charakteristiku větráku.

V dokumentaci se uvádí, že procentuání střída signálu PWM by měla odpovídat rychlosti otáčení, tedy jejich vzájemná závislost by měla být lineární (viz [SUN12]).

Způsob měření charakteristiky. Postup pro měření charakteristiky větráku:

1. Nastavení požadované střídy PWM signálu
2. Ustálení rychlosti otáčení
3. Zaznamenání rychlosti (v ot./min)

Tento postup se opakuje pro všechny vybrané hodnoty plnění střídy. Po ukončení měření se jednotlivé hodnoty zanesou do společného grafu a jejich spojením je získána charakteristika větráku (obr. 5.2).



Obrázek 5.2: Experimentálně získaná charakteristika větráku

Z obrázku 5.2 je patrné, že výsledná charakteristika obsahuje statické nelinearity.

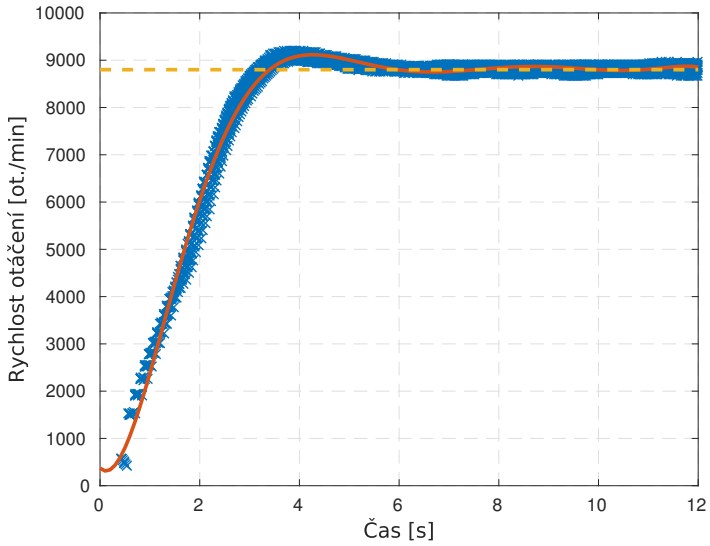
Zóna necitlivosti. Zóna necitlivosti je rozsah vstupního napětí (v tomto případě střídy signálu PWM), při které se větrák neroztočí. Tato nelinearita se zde objevuje do střídy s plněním 3 % (včetně). Při určování zóny necitlivosti se v některých chvílích větrák neočekávaně roztočil a pomalu dotáčel. O tomto jevu se zmiňuje i dokumentace (viz [SUN12, str. 9]).

Saturace. Saturace je jev, při kterém se zvyšováním vstupního napájení (střídy signálu PWM) nezmění ustálená hodnota výstupu. Tato statická nelinearita se objevuje při střídě s plněním nad 97 %.

Podle lineární části charakteristiky lze závislost veličin vyjádřit rovnicí:

$$\omega = 17600 \frac{D_{PWM}}{100}. \tag{5.1}$$

Vzhledem ke statickým nelinearitám platí rovnice (5.1) pro činitel plnění D_{PWM} 3 %–97 %.



Obrázek 5.3: Aproximace naměřených hodnot spolu s vyznačenou ustálenou hodnotou

5.5 Identifikace

Pro vytvoření modelu je nutné větrák identifikovat. Identifikaci jsem se rozhodl provést při nastavení střídy 50%, protože při této hodnotě (a jejím okolí) je charakteristika větráku lineární.

Způsob měření pro identifikaci. Postup pro měření dat, která se použijí při identifikaci systému:

1. Spuštění záznamu rychlosti otáčení (v ot./min)
2. Nastavení požadované střídy PWM signálu
3. Ustálení rychlosti otáčení
4. Vypnutí záznamu

Tento postup se opakuje pro eliminaci náhodné chyby, která vzniká při měření. Naměřené hodnoty se poté vloží do společného grafu a proloží křivkou (obr. 5.3).

Především podle tvaru odezvy na skok (obr. 5.3) se dá větrák identifikovat jako systém druhého řádu. Lze tedy předpokládat, že bude odpovídat obecnému přenosu (viz [Še17, sl. 5]):

$$G(s) = k\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \tag{5.2}$$

Rovnice (5.2) obsahuje tyto proměnné:

Ustálená hodnota $y(\infty)$. Ustálená hodnota je velikost výstupu pro odeznění všech přechodových jevů. Její hodnotu lze vypočítat na základě změřené charakteristiky (viz rovnici 5.1). Pro ustálenou hodnotu tedy platí:

$$y(\infty) = 8800\,\text{ot./min.} \tag{5.3}$$

Doba ustálení T_s. Doba ustálení odpovídá času, při kterém se průběh dostane do určitého okolí ustálené hodnoty. Pro 2 % okolí platí:

$$T_s \doteq 5{,}1\,\text{s.} \tag{5.4}$$

Procentuální překmit $\%OS$. Tato hodnota označuje o kolik procent průběh maximálně překmitl ustálenou hodnotu – v tomto případě platí:

$$\%OS \doteq 3{,}585\,\%. \tag{5.5}$$

Poměrné tlumení ζ. Pro tlumení platí následující vztah:

$$\zeta = \frac{-\ln\left(\%OS/100\right)}{\sqrt{\pi^2 + \ln^2\left(\%OS/100\right)}} \doteq 0{,}727. \tag{5.6}$$

Přirozená frekvence ω_n. Pro přirozenou frekvenci ω_n platí vztah:

$$\omega_n \doteq \frac{4}{\zeta T_s} \doteq 1{,}079, \tag{5.7}$$

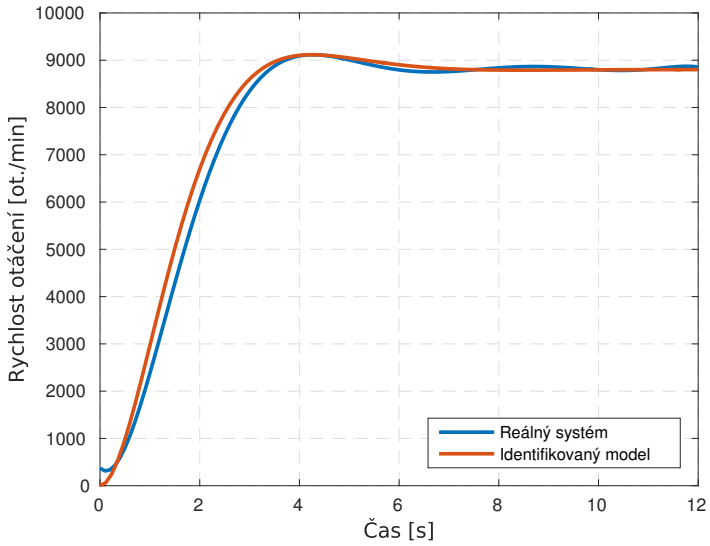
Zesílení k. Zesílení udává, kolikrát je ustálená hodnota na výstupu vyšší než hodnota vstupní. Lze ho tedy vyjádřit rovnicí:

$$k = \frac{y(\infty)}{u(\infty)} = 176. \tag{5.8}$$

Výsledná přenosová funkce – po dosazení rovnice 5.3 až rovnice 5.8 do rovnice 5.2 je:

$$G(s) = \frac{204{,}717}{s^2 + 1{,}569s + 1{,}163}. \tag{5.9}$$

Ověření modelu. Pro ověření identifikace porovnám skokovou odezvu systému, který je popsán přenosovou funkcí v rovnici (5.9) s aproximací skokové odezvy větráku z naměřených dat (obr. 5.3).



Obrázek 5.4: Porovnání odezev na jednotkový skok

Porovnání původního systému s identifikovaným modelem je na obrázku 5.4. Strmost náběhu, maximální překmit a doba ustálení modelu odpovídají původnímu systému.

5.6 Regulace

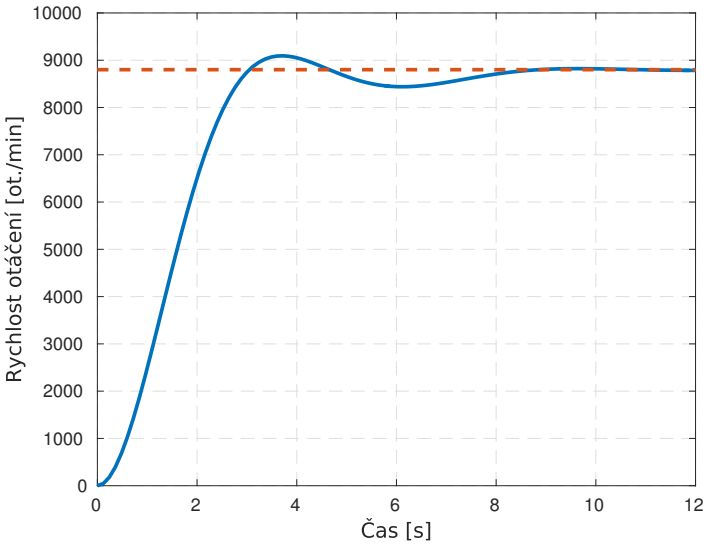
Pro vytvořený model větráku (viz rovnici 5.9) lze navrhnout určitými postupy zpětnovazební regulátor. Způsob, který jsem si vybral, využívá aplikaci „PID tuner“ pro *MATLAB* (viz [Mat]). Tuto metodu jsem si vybral především kvůli své jednoduchosti a rychlému zobrazení účinků regulace.

PID tuner. V „PID tuneru“ stačí zvolit systém pro regulaci, vybrat typ regulátoru a pak dvěma posuvníky měnit vlastnosti tvořeného kompenzátoru. Jeho vliv na systém (resp. na jednotkový skok reference) je vidět v reálném čase. Při manipulaci s proměnnými jsem se pokoušel přiblížit k vlastnostem systému zjištěným během měření, mezi které patří doba ustálení, strmost náběhu a překmit. Regulátor, který nebere v úvahu dynamiku původního systému, může mít vyšší a rychlejší akční zásahy, na které větrák nemusí stíhat reagovat.

Měření otáček větráku není příliš stabilní a průběh je zatížen velkým šumem. Z tohoto důvodu je navržený regulátor pouze typu PI. Hodnoty parametrů jsou:

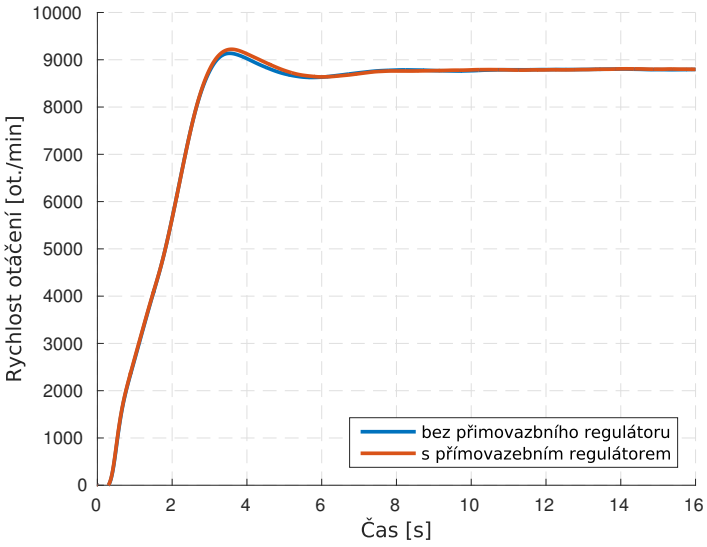
$$K_p = 3{,}654 \cdot 10^{-3}, K_i = 3{,}657 \cdot 10^{-3}. \tag{5.10}$$

Ověření regulátoru. Vytvořený kompenzátor zapojím do zpětné vazby k modelu větráku a změřím jeho odezvu na skok reference (obr. 5.5).



Obrázek 5.5: Odezva modelové soustavy s regulátorem na skok reference

Ověření regulátoru na reálném systému. Kromě ověření navrženého kompenzátoru (viz rovnici 5.10) s modelem, je důležité ověřit jeho funkci i s reálným systémem. Implementaci PID regulátoru na Arduinu se věnuje sekce 7.3.

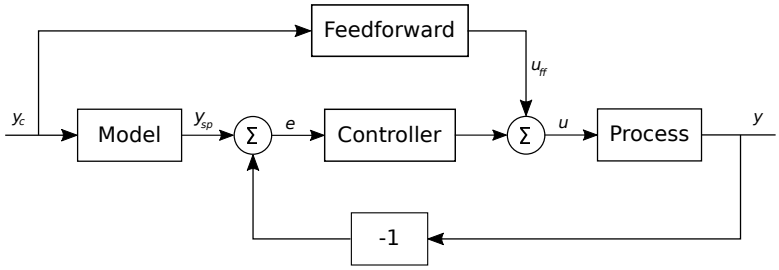


Obrázek 5.6: Odezva reálné soustavy se zpětnovazebním a bez/s přímovazebním regulátorem

5.6.1 Dopředný regulátor

Sledování reference soustavy se dá zlepšit použitím dopředného regulátoru (Feedforward), který je připojen z reference přímo na vstup systému (Process)

mimo regulátor (obr. 5.7).



Obrázek 5.7: Zapojení dopředného regulátoru pro referenci (podle [AH95, str. 285])

Aby bylo sledování reference dokonalé, musel by přenos regulátoru odpovídat inverzi modelu (viz [Še13, sl. 2]). Aproximaci inverze se dá dosáhnout za pomoci lineární charakteristiky (viz 5.1). Vyjádřením D_{PWM} se získá hodnota P složky dopředného regulátoru:

$$K_p = \frac{100}{17600} \tag{5.11}$$

Účinek přímovazebního regulátoru. Pro ověření funkčnosti přímovazebního regulátoru a jeho porovnání se soustavou bez tohoto kompenzátoru naměřím průběhy podle stejného způsobu uvedeném v sekci 5.5.

Na obrázku 5.6 jsou porovnány průběhy aproximované z naměřených dat pro obě soustavy. Průběh pro soustavu se zapojeným přímovazebním regulátorem má podle grafu o 0,969 % větší překmit a o 0,288 s delší dobu ustálení.

Kapitola 6
Senzory polohy

Ve finálním zapojení je zapotřebí měřit polohu jednoho, popř. dvou míčků. Tato kapitola porovnává vybrané způsoby snímání polohy při měření v trubici.

6.1 Infračervený senzor

Infračervený triangulační senzor využívá infračervené diody, jejíž světlo dopadá na měřený objekt a snímáním odrazu dochází ke změření vzdálenosti. K výpočtu se používá metoda triangulace. Senzor se skládá ze dvou částí – „vysílací“ diody a „přijímací“ fotodiody (viz [RDKN05, str. 34–35]).

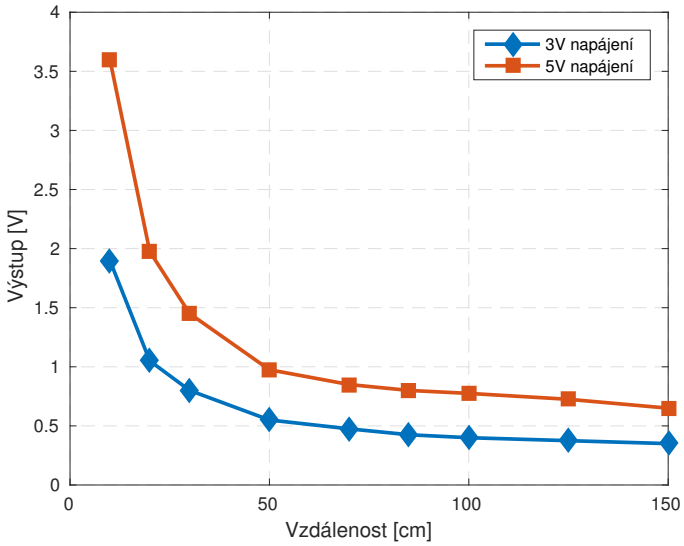
Senzor vybraný pro experimenty byl *Sharp GP2Y0A60SZLF* s modulem od firmy Pololu (viz [Pol]). Samotný senzor by měl dokázat měřit vzdálenosti $10-150\,\mathrm{cm}$ (viz [SHA]).

6.1.1 Zapojení

Přídavný modul zjednoduší zapojení senzoru na pouhé 4 piny – zem (GND), 5 V stejnosměrné napájení (VCC), zapnutí měření (EN) a výstup senzoru (OUT). Změřená vzdálenost se na výstupu senzoru projeví jako velikost napětí v rozmezí $0-5V$.

Po zapnutí senzoru (přivedením logické 1 na pin EN) dochází periodicky k měření vzdálenosti a změně napětí na pinu OUT. Perioda tohoto měření je $16{,}5\,\mathrm{ms}\pm3{,}7\,\mathrm{ms}$ (viz [SHA]).

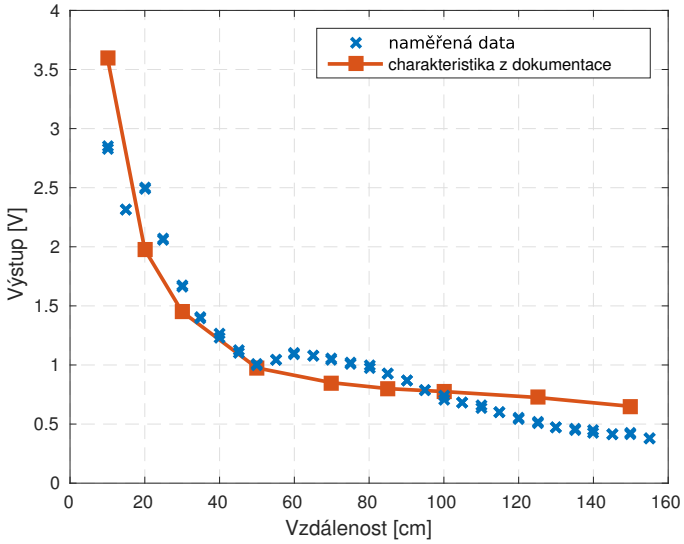
6.1.2 Charakteristika



Obrázek 6.1: Charakteristika senzoru podle dokumentace (viz [SHA, str. 6])

Charakteristika senzoru (viz obr. 6.1) předpokládá nelineární závislost mezi změřenou vzdáleností a výstupním napětím. Abych zjistil chování senzoru v prostředí trubice, provedu vlastní měření charakteristiky.

Měření charakteristiky senzoru je provedeno po 5 cm úsecích v intervalu 10−155 cm. Pro zachycení případné hystereze jsou měření provedena dvě, v opačném směru posouvání předmětu.



Obrázek 6.2: Naměřená charakteristika senzoru

Z grafu (obr. 6.2) je patrné, že charakteristika infračerveného senzoru umístěného v trubici neodpovídá charakteristice z dokumentace. V oblasti okolo vzdálenosti 50 cm od senzoru dochází k náhlému poklesu výstupního napětí. Opakovaným měřením bylo zjištěno, že to nebyla náhlá chyba v měření, ale že je toto skutečný výstup senzoru. Z tohoto důvodu není tento senzor pro měření polohy míčku vhodný.

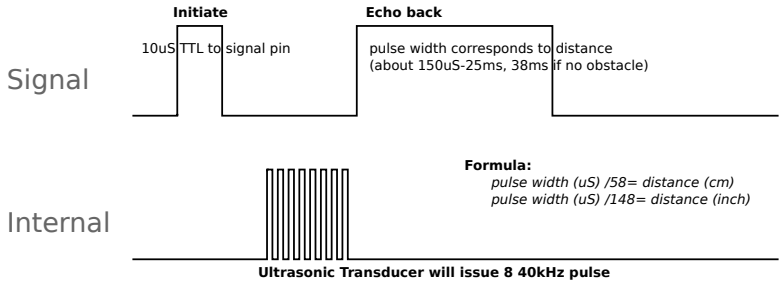
6.2 Ultrazvukový senzor

Ultrazvukový senzor polohy využívá odrazu vyslané zvukové vlny od předmětu. Doba zpoždění mezi vysláním vlny a jejím přijetí je za pomoci rychlosti zvuku přepočítána na vzdálenost. Senzor se skládá ze dvou částí – vysílače a přijímače zvukové vlny (viz [Mar04, str. 90]).

Ultrazvukový senzor vybraný pro experimenty nese označení HC-SR04 a podle dokumentace dokáže změřit vzdálenosti 2 cm−5 m (viz [ITe10]).

6.2.1 Zapojení

Tento ultrazvukový senzor má 4 piny - zem (GND), 5 V stejnosměrné napájení (VCC), signál k provedení měření (Trig) a výstup senzoru (Echo).



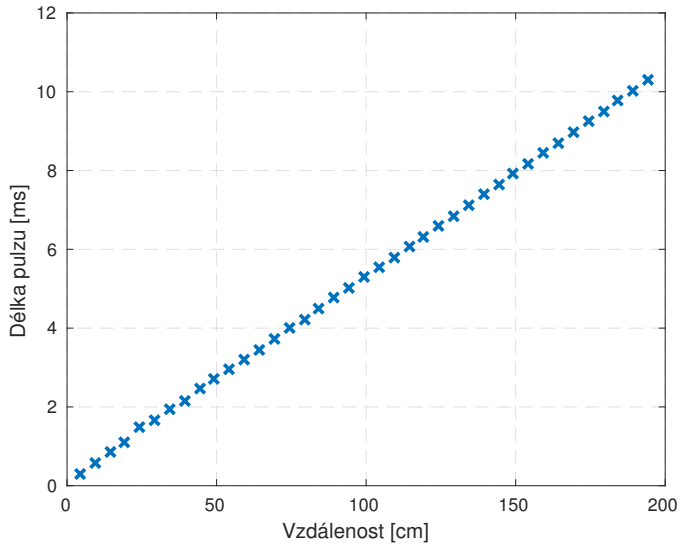
Obrázek 6.3: Funkce ultrazvukového senzoru (podle [ITe14])

Funkce je znázorněna na obrázku 6.3 – měření je provedeno po vyslání pulzu na pin *Trig*. Jeho šířka má být 10 µs (experimenty ukázaly, že senzor měření provede i s pulzem šířky 5 µs). Senzor po obdržení toho signálu vyšle sérii osmi zvukových pulzů o frekvenci 40 kHz a začne vysílat pulz na pinu *Echo*. Jeho šířka je závislá na vzdálenosti od překážky (až se vrátí odražená zvuková vlna, senzor pulz ukončí).

6.2.2 Charakteristika

Podle obrázku 6.3 z dokumentace se dá očekávat lineární závislost mezi vzdáleností předmětu a délkou pulzu vyslaného senzorem. Protože ani tento případ nepočítá s umístěním senzoru do trubice, provedu proměření závislosti délky pulzu na vzdálenosti objektu.

Měření bylo provedeno v rozsahu délky trubice.



Obrázek 6.4: Změřená charakteristika ultrazvukového senzoru

Průběh vynesený do grafu 6.4 dokazuje, že je charakteristika ultrazvukového senzoru skutečně lineární. Nyní je třeba senzor kalibrovat. Kalibrace se provede proložením naměřených dat přímkou a vyjádřením jejího předpisu:

$$x = \frac{y - 104{,}895}{52{,}321}, \tag{6.1}$$

kde proměnná x má význam vzdálenosti v cm, proměnná y délku pulzu v µs.

Protože rovnice (6.1) neodpovídá rovnici na obrázku 6.3 byla kalibrace oprávněná.

Měření dvou míčků. Pro měření polohy dvou míčků jsou potřeba minimálně dva senzory – pro každý míček jeden. V trubici by pak tyto senzory byly umístěny proti sobě, každý na jednom konci trubice. Nevýhodou tohoto umístění senzorů je jejich případné ovlivňování měření. Tento problém se eliminuje, pokud nebudou senzory měřit ve stejný čas a bude mezi jejich spouštěcími pulzy dostatečná prodleva. Je doporučeno provádět opakovaná měření minimálně po 60 ms (viz [Elea]). Tento čas je ale uvažován pro otevřené prostranství a pro maximální měřitelnou vzdálenost. V případě trubice může být prodleva menší.

6.3 Rozpoznávání obrazu

Metoda rozpoznávání obrazu předpokládá kameru umístěnou tak, aby její obraz obsahoval celou délku trubice. Pomocí speciálního softwaru se v obrazu hledají specifické body – v tomto případě určitá barva míčku. Výhodou je, že by stačila jedna kamera na snímání obou míčků.

Pro zjednodušení pokusů s rozpoznáváním obrazu jsem si vybral kameru „PixyCam“, která má tuto funkcionalitu již implementovanou. Tlačítkem

umístěným na kameře je možné program „učit“ barvy. Přes připojenou sériovou linku (např. SPI) posílá kamera informace o poloze jednotlivých objektů (viz [Car]).

Experimenty s kamerou a míčkem umístěným v trubici ukázaly tuto metodu snímání polohy jako zcela nevhodnou. Kamera byla schopna míček detekovat pří jejím přiblížení k trubici na 30 cm úseku. Její případné využití by tedy mohlo být pouze ve zpřesnění polohy určené jiným senzorem.

Je možné, že s jinou kamerou, která má odlišné parametry (například rozlišení snímaného obrazu), by mohl mít experiment s touto metodou lepší výsledky.

6.4 Výběr senzoru

Vyzkoušeny byly tři způsoby snímání polohy míčku v trubici. Infračervený senzor se ukázal nevhodný kvůli své charakteristice, kde v určité vzdálenosti míčku od senzoru nebylo měření jednoznačné (obr. 6.2). Nevhodná se ukázala i metoda rozpoznávání obrazu, která i přes svou přesnost v určování polohy nebyla schopná monitorovat míček po celou délku trubice. Pro měření vzdálenosti míčku byl proto vybrán ultrazvukový senzor, jehož charakteristika je lineární (obr. 6.4).

Kapitola 7
Arduino Uno

Pro měření a regulaci soustavy je použita mikrokontrolérová vývojová deska Arduino Uno. Tato deska má mimo jiné 14 digitálních vstupních/výstupních pinů, 6 analogových vstupů a 3 timery (viz [Ardg]).

7.1 Zapojení

K Arduinu je zapotřebí připojit měření a ovládání větráku a jeden (popř. dva) senzory.

Větrák. Větrák je k Arduinu připojen pomocí dvou vodičů – jedním se měří rychlost otáčení (viz subsekci 5.2.1), druhým se rychlost otáčení ovládá (viz subsekci 5.2.2). Vodič pro měření otáčení je připojen na pin 2 (viz sekci 7.4), vodič pro ovládání rychlosti je připojen na pin 9 (viz sekci 7.5).

Senzory. Ultrazvukový senzor je připojen k Arduinu dvěma vodiči – jedním se spouští měření, druhým se získává informace o vzdálenosti překážky (viz sekci 6.2). Vodič pro spouštění měření je připojen na pin 4, druhý na pin 5 (viz sekci 7.7). Jejich umístění nezáleží na žádných dalších podmínkách a mohou být tudíž připojeny na jakékoliv digitální piny. Pro druhý ultrazvukový senzor je rezervován pin 6, respektive pin 7.

Rozšířené zapojení

V rámci lepší prezentovatelnosti a rozšiřitelnosti je možné k Arduinu připojit další zařízení. Tato část je teoretická a nebyla implementována.

Podsvícení. Pro zpestření efektu levitace je možné míček podsvítit. Mezi zdroje světla se nabízí buď LED diody nebo laserové diody, jejichž efekt by měl být mnohem výraznější (viz [Fer]). Model systému počítá i s místem na diodu – např. mezi přijímačem a vysílačem ultrazvukových senzorů je prostor $9 \times 12\,\text{mm}$. Svícení jednobarevné diody by šlo ovládat jedním pinem podobně jako větráky (viz sekci 5.2.2).

Komunikační kanál. Při použití několika modelů systému najednou je nevýhodné každý ovládat zvlášť – z toho důvodu je možné Arduina pro ovládání systému podřídit jinému (centrálnímu) počítači. Pro komunikaci mezi centrálním počítačem a Arduiny lze použít například PWM signál po jednom vodiči (tedy využití jednoho pinu na každém Arduinu) nebo sériovou linku – hardwarovou nebo softwarovou (viz [Ardf]), která by zabrala dva piny na každém Arduinu. Přes vytvořený komunikační kanál by se přenášela informace o požadované výšce míčku (popř. míčků).

V případě sériové linky by se pravděpodobně jednalo o přenos řetězce, v případě signálu PWM by roli hrály především šířky pulzů.

7.2 Struktura programu

Program je strukturován podle časové závislosti operací na časově závislé a časově nezávislé.

Časově závislé. Mezi časově závislé operace patří měření otáček a regulace rychlosti otáčení větráku. Tyto operace musí být přednostně vykonány, proto jsou využity přerušení v programu. Pro měření větráku se jedná o přerušení vyvolané změnou logické hodnoty na pinu (viz sekci 7.4). Pro regulaci rychlosti větráku se jedná o přerušení, které vytváří časovač (viz sekci 7.5).

Časově nezávislé. Mezi časově nezávislé operace řadím především kontrolní výpisy programu – tedy například informace o rychlosti otáčení větráku a poloze míčku. Jejich umístění je v hlavní smyčce programu – *loop()*, která má nejnižší prioritu. Dále mezi časově nezávislé řadím i měření vzdálenosti míčku a regulaci podle polohy. Ač se jedná o operaci, která se dá zařadit mezi časově závislé, stejně jako větrák, rozhodl jsem se pro tento krok především z toho důvodu, že délka přerušení měření polohy je až 40x delší než perioda spouštění regulace větráku. Arduino je jednoprocesorové, tudíž nemůže zpracovat více přerušení najednou, a tím se zhoršuje regulace větráku.

7.3 PID regulátor

Pro PID regulaci za pomocí Arduina je použita knihovna PIDLibrary (viz [Bea]).

7.3.1 Vytvoření regulátoru

Regulátor je objekt, jehož instance se vytvoří konstruktorem:

```
double Input, Output, Setpoint;
double kp, ki, kd;

PID myPID(&Input, &Output, &Setpoint, kp, ki, kd, DIRECT);
```

&Input. Konstruktoru se předává reference na proměnnou Input, která obsahuje výstup ze systému.

&Output. Objektu se předává reference na proměnnou Output, do které se zapisuje výstup regulátoru.

&Setpoint. Regulátoru se předává i reference na proměnnou Setpoint, která obsahuje referenci.

kp, ki, kd. Další parametry jsou předány v proměnných kp, ki a kd, které obsahují konstanty jednotlivých částí PID regulátoru – proporcionální, integrační a derivační.

DIRECT. Poslední parametr udává směr regulace (resp. zda mají být konstanty kladné či záporné) – DIRECT znamená kladné konstanty, REVERSE záporné.

7.3.2 Funkce regulátoru

PID regulátor z knihovny PIDLibrary obsahuje množství metod pro práci s objektem:

void SetOutputLimits(int, int). Funkce *SetOutputLimits()* nastaví omezení výstupu regulátoru – prvním parametrem je spodní hranice, druhým hranice horní.

void SetSampleTime(int). Funkcí *SetSampleTime()* se nastaví vzorkovací periodu pro regulátor. Pokud by byl regulátor spuštěn předčasně, podmínka uvnitř mu nedovolí proběhnout.

void SetMode(int). Funkcí *SetMode()* s parametrem 1 (také AUTOMATIC) se PID regulátor aktivuje.

bool Compute(). Zavoláním funkce *Compute()* se provede regulace – ze zadaného vstupu Input, reference Setpoint a parametrů kp, ki a kd se určí akční zásah regulátoru Output.

void FlushVariables(). Funkce *FlushVariables()* byla do knihovny přidána. Jejím zavoláním se pročistí vnitřní proměnné PID regulátoru – další výpočet akčního zásahu nebude závislý na předchozích hodnotách. Tato funkce má své využití při cyklickém měření, kdy je zapotřebí systém „zresetovat“. Samotná funkce vypadá takto:

```
void PID::FlushVariables(void)
{
 ITerm = 0;
 lastInput = 0;
}
```

7.4 Měření otáček

Pro měření rychlosti otáčení větráku je použita funkce *rpm_fun()*:

```
// Mereni otacek
#define RPM_COUNT_MAX 4

volatile byte rpm_count;
volatile unsigned long rpm;
volatile unsigned long rpm_last_time;


// Funkce pro pocitani otacek
void rpm_fun()
{
  rpm_count++;

  if (rpm_count >= RPM_COUNT_MAX) {
    unsigned long rpm_time = micros();
    rpm = 30000000/(rpm_time-rpm_last_time)*RPM_COUNT_MAX;
    rpm_last_time = rpm_time;
    rpm_count = 0;
  }
}
```

Tato funkce je zavolána pokaždé, když se na pinu 2 objeví pulz. Toho je dosaženo funkcí *attachInterrupt()*, která na Arduinu Unu umožňuje přiřadit funkci pinům 2 a 3 (viz [Ardc]).

Počítací funkce průměruje časový rozdíl po určitém počtu pulzů. Experimentálně byl počet pulzů stanoven na 4. Při menším počtu pulzů se viditelně projevovala chyba způsobená zaokrouhlováním při počítání časového rozdílu – po sobě jdoucí měření rychlosti byla diametrálně odlišná.

Výpočet rychlosti otáčení. Rovnice pro přepočet časového rozdílu na rychlost otáčení vychází z následujícího vztahu:

$$N_{\mathrm{RPM}} = \frac{60{\cdot}10^6}{\Delta t}\frac{N}{2}, \tag{7.1}$$

přičemž Δt je časová diference v µs, N počet půlotáček a výsledné N_{RPM} rychlost otáčení v ot./min.

7.5 Řízení rychlosti otáčení

Pro řízení rychlosti otáčení větráku (tedy pro generování PWM signálu o určité střídě) se dá použít funkce *analogWrite()* (viz [Ardb]). V základním nastavení funkce generuje PWM signál při frekvenci 490 Hz nebo 980 Hz,

avšak podle dokumentace [SUN12] je třeba pro ovládání větráku frekvence v rozmezí 21 kHz–28 kHz s cílovou frekvencí (tedy preferovanou) 25 kHz. Pro navýšení frekvence je nutné upravit registry některého z časovačů.

Veškeré další informace o časovačích v této sekci jsou (pokud není jinak uvedeno) převzaty z dokumentace [Atm15].

7.5.1 Výběr časovače

Arduino Uno disponuje třemi časovači – čísly označenými 0 až 2. Protože úprava registrů změní frekvenci na které časovač generuje přerušení, je třeba vybrat takový, který neovládá nic jiného – tím je vyloučen časovač s číslem 0. Ten je zodpovědný za určování času v systému – funkce *millis()* a *delay()* (viz [Arda]). Proto pro úpravu použiji časovač 1, který obsluhuje piny 9 (`OC1A`) a 10 (`OC1B`).

7.5.2 Registry časovače

Funkci časovače určují jeho registry, mezi které patří:

TCCR1A, TCCR1B. Osmibitové registry `TCCR1A` a `TCCR1B` určují chování čítače, resp. způsob určování výstupu (mód) – tomu se věnuje subsekce 7.5.3. V těchto registrech se dá nastavit i hodnota tzv. prescaleru, což je dělička, která omezuje frekvenci vstupního signálu.

TCNT1. Šestnáctibitový registr `TCNT1` představuje vnitřní čítač časovače.

ICR1. Hodnota v šestnáctibitovém registru `ICR1` shora omezuje čítač `TCNT1`.

OCR1A, OCR1B. Hodnoty v šestnáctibitových registrech `OCR1A` a `OCR1B` jsou stále porovnávány s hodnotou v interním čítači `TCNT1`. Pokud se čísla rovnají, je možné vygenerovat přerušení nebo změnit hodnotu na výstupu pinů `OC1A` (pro registr `OCR1A`) a `OC1B` (pro registr `OCR1B`).

7.5.3 Módy časovače

Každý časovač umožňuje úpravou registrů *TCCR1A* a *TCCR1B* nastavit výstupní chování, které může být:

- normální/bez PWM (non-PWM),
- „rychlé“ PWM (fast PWM),
- fázově správné PWM (Phase Correct PWM),
- fázově a frekvenčně správné PWM (Phase and Frequency Correct PWM).

Pro řízení rychlosti otáčení jsem si vybral „fázově správné PWM“ (též označované jako středově zarovnané PWM), protože tento mód se běžně pro řízení stejnosměrných motorů používá. Důvodem je především elektromagnetická kompatibilta a větší odolnost vůči elektromagnetickému rušení – „hrany“

PWM signálu nejsou na okraji periody, ale jsou rozprostřeny přes celou periodu (viz [LG11, sl. 27]).

Fázově správné PWM

Při tomto módu časovač čítá hodnotu v registru `ICNT1` obousměrně – nejprve od 0 do `ICR1`, poté od `ICR1` do 0 a v každém kroku porovnává hodnotu `ICNT1` s hodnotu v registru `OCR1x`. Toto porovnávání má dvě možnosti nastavení – neinvertující a invertující.

Pokud se hodnoty v registrech `ICNT1` a `OCR1x` rovnají, neinvertující nastavení tohoto módu přiřadí na pin `OC1A` logickou 0, pokud čítač čítá směrem k `ICR1`, v opačném případě logickou 1. Obdobně pro pin `OC1B` s registrem `OCR1B`. Invertující nastavení je stejné až na obrácené logické hodnoty.

Z funkce „překlápění“ PWM signálu na výstupním pinu je zřejmé, že pro vygenerování jedné periody je zapotřebí provést celý cyklus počítání ve vnitřním čítači (tedy směrem nahoru i dolu). Tím je maximální možná frekvence – oproti „rychlému“ PWM – poloviční.

Nastavení frekvence časovače. Frekvence časovače se pro vybraný PWM mód nastavuje hodnotou v registru `ICR1` a velikostí prescaleru. Aby bylo pro nastavení PWM signálu co nejvíce různých hodnot, nastavím hodnotu prescaleru na 1. Hodnota registru `ICR1` se dá vypočítat z rovnice (7.2) – (viz [Atm15, str. 126]):

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}, \tag{7.2}$$

přičemž $f_{OCnxPCPWM}$ je výsledná frekvence PWM v Hz, $f_{clk\_I/O} = 16\,\text{MHz}$ frekvence procesoru, $N = 1$ velikost prescaleru a TOP označuje hodnotu v registru `ICR1`. Vyjádřením TOP a dosazením (rovnice 7.3) získám hodnotu, která patří do daného registru:

$$TOP = \frac{f_{clk\_I/O}}{2 \cdot N \cdot f_{OCnxPCPWM}} = \frac{16000000}{2 \cdot 1 \cdot 25000} = 320. \tag{7.3}$$

7.5.4 Nastavení registrů

Podle vybraného módu a vypočtených hodnot registrů (subsekce 7.5.3) lze nyní přistoupit k úpravě registrů, která se provede ve funkci *setup()* (převzato z [Bon16]):

```
// Configure Timer 1 for PWM @ 25 kHz.
TCCR1A = 0;           // undo the configuration done by...
TCCR1B = 0;           // ...the Arduino core library
TCNT1  = 0;           // reset timer
TCCR1A = _BV(COM1A1)  // non-inverted PWM on ch. A
       | _BV(COM1B1)  // same on ch; B
       | _BV(WGM11);  // ph. correct PWM, TOP = ICR1
```

```
TCCR1B = _BV(WGM13)    // ditto
       | _BV(CS10);    // prescaler = 1
ICR1   = 320;          // TOP = 320
```

7.5.5 Nastavení střídy PWM

S upravenými registry časovače 1 je zapotřebí také správně nastavovat střídu PWM. Podle teorie zmíněné v subsekci 7.5.3 je vyjádření střídy obsaženo v registru `OCR1x`. Podle nastavení maximální hodnoty čítače `ICR1` musí být hodnota `OCR1x` v rozmezí 0−320. Tato hodnota odpovídá plnění 0−100 % PWM. Přepočet z plnění střídy na hodnotu do registru se dá vyjádřit rovnicí (7.4):

$$\mathrm{OCR1x} = \frac{320}{100} D_{\mathrm{PWM}}. \tag{7.4}$$

V kódu programu je nutné vytvořit funkci, která bude přepisovat registr `OCR1x` (převzato z [Bon16]):

```
void analogWrite25k(int pin, int value)
{
  switch (pin) {
  case 9:
    OCR1A = value;
    break;
  case 10:
    OCR1B = value;
    break;
  default:
    // no other pin will work
    break;
  }
}
```

7.6 Regulace větráku

Větrák využívá regulátor, který pro realizaci musí být diskrétní. Jedná se tedy o časově závislou operaci. O určení parametrů regulátoru je sekce 5.6.

Vzorkovací perioda T_s. Určení vzorkovací periody se odráží od maximální rychlosti větráku 17600 ot./min. Otáčení větráku generuje pulz každou půlotáčku. To znamená, že Arduino přijme maximálně 35200 pulzů/min, což se dá vyjádřit frekvencí 586,67 Hz – frekvence regulátoru by měla být vyšší.

Implementace. Pro periodické volání regulátoru se dá například využít Timer0, jehož výchozí frekvence je 976,5625 Hz (viz [Arda]). Pro nastavení automatického spuštění funkce je potřeba přepsat dva registry (ve funkci *setup()*):

```
// Nastaveni hodnoty OCR0A
OCR0A = 0xAF;
TIMSK0 |= _BV(OCIE0A);
```

Nastavením bitu `OCIE0A` v registru `TIMSK0` se zapne generování přerušení `TIMER0_COMPA` v momentě, kdy se číslo ve vnitřním čítači bude rovnat číslu v registru `OCR0A`. Toto přerušení se dá zachytit funkcí `SIGNAL` s parametrem, který odpovídá danému přerušení – v tomto případě `TIMER0_COMPA_vect`.

```
// Funkce pro provedeni regulace
SIGNAL(TIMER0_COMPA_vect) {
    Input = rpm;
    myPID.Compute();
    analogWrite25k(PIN_FAN_PWM, int((320/100)*Output));
}
```

Funkce pro regulaci nejprve přiřadí aktuální rychlost otáčení do vstupu regulátoru, poté provede výpočet akčního zásahu regulace a tuto hodnotu zapíše na řídící pin větráku. Použitá funkce *analogWrite25k()* předpokládá zadanou hodnotu PWM v rozmezí 0−320. Pro přepočet se použije funkce v rovnici (7.4).

Přímovazební regulátor. Subsekce 5.6.1 uvažuje při regulaci větráku také využití přímovazebního regulátoru. Jeho implementace spočívá v úpravě jednoho řádku v regulační funkci:

```
// Funkce pro provedeni regulace
SIGNAL(TIMER0_COMPA_vect) {
    Input = rpm;
    myPID.Compute();
    analogWrite25k(PIN_FAN_PWM, int(constrain((320 / 100)
       * Output + (Setpoint * 320 / 17600), 0, 320)));
}
```

Do kódu přibyla kromě členu, který odpovídá přímovazebnímu regulátoru, také funkce *constrain()*. Tato funkce zajistí, že vypočtený akční zásah do větráku bude v intervalu $\langle 0, 320 \rangle$ (viz [Ardd]).

7.7 Měření vzdálenosti

Implementace měření vzdálenosti (tedy obsluhy ultrazvukového senzoru) se skládá ze dvou částí (viz sekci 6.2). První částí je vyslání signálu senzoru, aby začal měřit:

```
digitalWrite(PIN_UZ_TRIG, LOW);
delayMicroseconds(2);
digitalWrite(PIN_UZ_TRIG, HIGH);
delayMicroseconds(5);
digitalWrite(PIN_UZ_TRIG, LOW);
```

Kombinací funkcí *digitalWrite()* a *delayMicroseconds()* docílím požadovaného spouštěcího pulzu. Po jeho vyslání je potřeba změřit délku pulzu, který senzor posílá zpět:

```
long range = pulseIn(PIN_UZ_ECHO, HIGH);
```

Funkce *pulseIn()* vyčká, až se logická hodnota na příslušném pinu zvýší na logickou 1 a pak začne počítat čas. V momentě, kdy logická hodnota poklesne, funkce vrátí délku pulzu v mikrosekundách (viz [Arde]).

7.7.1 Průměrování hodnoty

Během experimentování s ultrazvukovým senzorem jsem pozoroval občasné „výchylky“ v měřených hodnotách – naměřená hodnota se diametrálně lišila od ostatních. Pro zmírnění účinku této hodnoty na regulaci jsem se rozhodl pro implementaci klouzavého průměru, jehož výsledkem je „efektivní“ poloha míčku, se kterou se dále počítá. Implementace může vypadat následovně:

```
ef_vzd[ef_vzd_index++] = aktualni_vzd;

if (ef_vzd_index >= EF_VZD_MAX)
  ef_vzd_index = 0;

unsigned long efektivni_vzd = 0;

for (int i = 0; i < EF_VZD_MAX; i++) {
  efektivni_vzd += ef_vzd[i];
}

efektivni_vzd /= EF_VZD_MAX;
```

7.8 Kontrolní výpisy

Pro záznam měření a pro uchování informace o nastavení v systému používám kontrolní výpisy, které přijímám v počítači přes sériový terminál, který je připojen na sériovou linku k Arduinu. Jednotlivé výpisy jsou odděleny koncem řádky. Výpisy mohou vypadat například takto:

```
#Mereni prubehu
5261364:sta
5261784:rpm:5400
5262872:cms:164
5263864:stp
```

Komentáře. Komentáře začínají znakem # (křížek). V záznamech měření většinou obsahují informace o typu měření, nastavení PID regulátorů, referenční hodnota a další.

Události. Události (nebo také měřená data) jsou všechny ostatní řádky, které nejsou komentáře. Skládají se ze dvou nebo třech částí s použitým oddělovačem **:** (dvojtečka). První údaj obsahuje aktuální čas systému (tedy Arduina) v mikrosekundách. Druhý údaj je třípísmenný a slouží jako reference – označuje, co daná řádka reprezentuje. Nepovinná třetí část pak obsahuje hodnotu pro danou referenci.

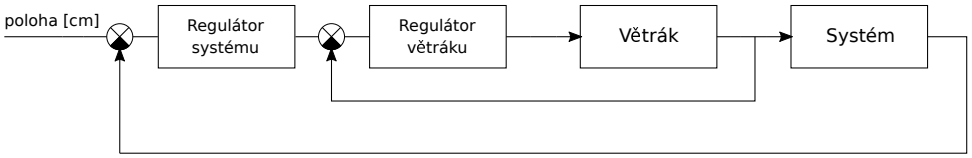
Ref.	Význam
sta	začátek měření
stp	konec měření
set	požadovaná rychlost otáčení [ot./min]
scm	požadovaná poloha míčku [cm]
rpm	rychlost otáčení [ot./min]
pwm	akční zásah do větráku [0−320 PWM]
cms	aktuální poloha míčku [cm]
cmx	efektivní poloha míčku [cm]

Tabulka 7.1: Použité reference pro kontrolní výpisy

Kapitola 8

Regulace systému

K regulování větráku jsem přistupoval na základě identifikace a modelace. Kromě metod založených právě na identifikaci existují i další – především experimentální – způsoby regulace. K tomuto kroku jsem se rozhodl především kvůli složitosti celého systému. U všech zmíněných způsobů předpokládám použití regulátoru pro větrák, navrhovaný kompenzátor je v rámci schématu před větrákem (viz obrázek 8.1). Bylo by možné větrák jako podsystém vypustit a vytvářet regulátor pro celý systém – v tom případě by výstupem regulátoru byl činitel plnění signálu PWM.



Obrázek 8.1: Blokové schéma soustavy

8.1 Vícestavová regulace

Vícestavová regulace rozřazuje naměřené (vstupní) hodnoty do určitých skupin/stavů, podle kterých se určí výstup regulátoru. Složitější verze vícestavové regulace se označuje jako regulace založená na „fuzzy množinách“ (z anglického termínu „fuzzy control“ [PY98]). Tato regulace „. . . spočívá v zavedení tzv. stupně příslušnosti prvku k fuzzy množině, který může nabývat hodnot z intervalu $\langle 0,1 \rangle$ na rozdíl od klasické teorie množin, kdy každý prvek do množiny buď patří nebo nepatří.“ – [Mod02, str. 2].

Tato metoda je zcela experimentální, tudíž výsledky její regulace nemusí být srovnatelné s návrhy podle jiných metod. Na druhou stranu k jejímu použití není třeba znát o soustavě žádné informace - lze k ní přistupovat jako k tzv. „blackboxu“.

U této metody používám dva termíny, které nejprve definuji:

Základní rychlost ω_0. Základní (popř. nulovou) rychlostí označuji rychlost otáčení, při které je míček nejblíže rovnovážné poloze – tedy jeho poloha v trubici se nemění.

Cílová poloha h**.** Cílovou polohou (někdy označovanou jako „setpoint“) označuji polohu, na které chci, aby se míček ustálil.

8.1.1 Postup experimentu

S postupným zvyšováním nastavené rychlosti otáčení větráku sleduji polohu míčku. Rychlost otáčení, při které se míček začne zvedat, označím jako nulovou rychlost.

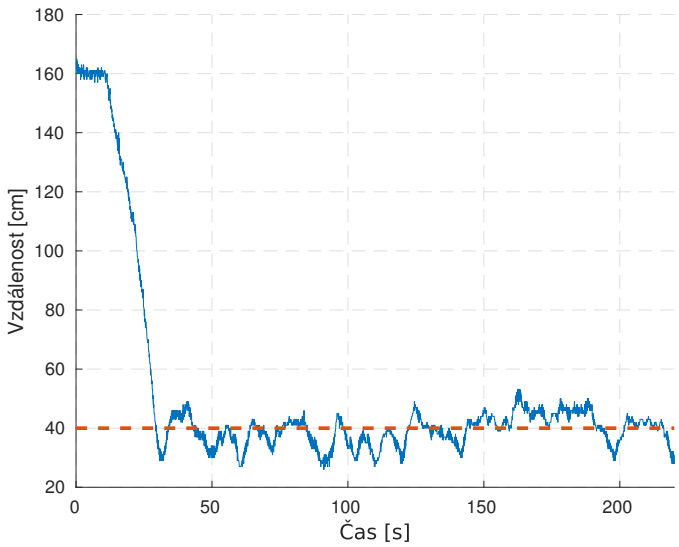
Nyní si zvolím počet stavů, kterých může regulátor nabývat, a tyto stavy rovnoměrně rozmístím okolo cílové polohy. V případě pěti stavů je mohu označit čísly: -2, -1, 0 (což je stav, kdy je míček v požadované poloze), $+1$ a $+2$. Každý ze stavů pak mohu popsat například zleva otevřeným intervalem (tedy až na poslední, který je otevřený). Tímto způsobem bude pro každou polohu jednoznačně přiřazen jeden stav.

Každý stav obsahuje rychlost větráku, kterou přiřadí na svůj výstup, pokud bude daný stav vybrán. Ve stavu označeném 0 je základní rychlost. V ostatních stavech je hodnota mírně odlišná od okolních stavů. Tyto ostatní hodnoty se dají například určit zavedením fixní odchylky ε_{rpm}, která se vynásobí číslem stavu a přičte k základní rychlosti.

8.1.2 Implementace regulátoru

Stavový regulátor lze implementovat například takto:

```
if (aktualni_poloha < poloha_21)      // stav -2
    vystup = omega_0 + 2 * eps;
else if (aktualni_poloha < poloha_10) // stav -1
```



Obrázek 8.2: Poloha míčku při zapojení vícestavového regulátoru s vyznačenou referenční hodnotou

```
    vystup = omega_0 + eps;
else if (aktualni_poloha < poloha_01) // stav 0
    vystup = omega_0;
else if (aktualni_poloha < poloha_12) // stav 1
    vystup = omega_0 - eps;
else                                  // stav 2
    vystup = omega_0 - 2 * eps;
```

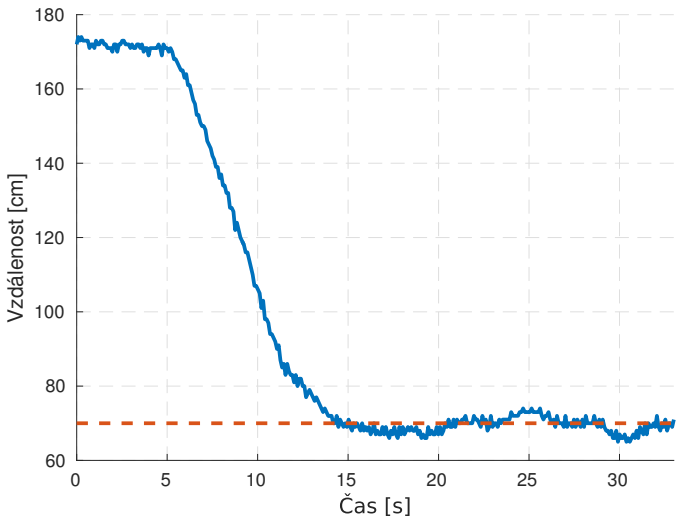
V kódu je použito označení ze subsekce (8.1.1) – (`poloha_xx`) a fixní ochylka ε_{rpm} (`eps`). Tato odchylka se po přenásobení číslem stavu odečítá, protože při nižší poloze je potřeba vyšší rychlost otáčení.

8.1.3 Ověření regulátoru

Vliv regulátoru na soustavu je ovlivněn nejen počtem stavů, ale i jim přiřazenými rychlostmi. Účinek regulátoru o sedmi stavech je na obrázku (8.2). Pro měření byl použit senzor polohy na horní straně trubce, proto je vzdálenost obrácená.

8.2 PID regulace

Další použitou metodou pro regulaci polohy míčku je experimentálně navržený PID regulátor, který se skládá ze tří složek – proporcionální (P), integrační (I) a derivační (D). Protože regulace vychází z klidového stavu (danou základní rychlostí – viz sekci 8.1), je k regulátoru přidán „offset“. Implementací se zabývá sekce 7.3.



Obrázek 8.3: Poloha míčku při zapojení PID regulátoru s vyznačenou referenční hodnotou

8.2.1 Postup experimentu

Po určení základní rychlosti (viz sekci 8.1) nastavím všechny složky PID regulátoru na 0. S postupným zvyšováním proporcionální složky sleduji chování míčku v trubici. Zvyšování hodnoty P složky zastavím v momentě, kdy jsou amplitudy oscilací přibližně stejně velké. Nastavenou hodnotu P složky zmenším přibližně na polovinu a postupně začnu upravovat hodnotu I složky, dokud se poloha míčku neustálí. Derivační složku jsem v tomto případě nepoužil, protože s její implementací je výsledný regulátor náchylný na rychlé změny (tedy nepřesnosti v měření) polohy.

8.2.2 Ověření regulátoru

Vliv regulátoru na soustavu je ovlivněn nastavením jeho jednotlivých složek. Účinek regulátoru s nastavením $K_p = 4{,}5, K_i = 0{,}0005, K_d = 0$ je na obrázku (8.3). Pro měření byl použit senzor polohy na horní straně trubce, proto je vzdálenost obrácená.

Kapitola 9

Závěr

V této práci jsem sestavil reálný model pro levitaci dvou ping-pongových míčků v trubici pomocí větráku – a to včetně řídicího systému, který je implementován na platformě Arduino.

Za využití vytvořeného matematického modelu pro jeden a poté i pro dva míčky jsem určil podmínky, které musí míčky splňovat, aby byla řízená vzduchová levitace možná. Výpočty ukázaly, že pro dva ping-pongové míčky není tímto způsobem regulovaná levitace uskutečnitelná.

Řídící systém je připraven pro řízení a regulaci rychlosti otáčení větráku a pro měření polohy dvou míčků. Další volné piny na Arduinu umožňují připojení dalších součástí, které jsou v práci zmíněny – například podsvícení nebo komunikační kanál pro vzdálené ovládání z centrálního počítače.

Návrhy spojovacích dílů společně s vytvořeným řídicím systémem umožňují snadné rozšíření systému.

Příloha A

Literatura

[AH95] K. Aström and T. Hägglund, *Pid controllers: Theory, design, and tuning*, 2nd ed., Instrument Society of America, 1995.

[Arda] Arduino AG, *Arduino - adjusting pwm frequencies*, http://playground.arduino.cc/Main/TimerPWMCheatsheet.

[Ardb] Arduino AG, *Arduino - analogwrite()*, https://www.arduino.cc/en/Reference/analogWrite.

[Ardc] Arduino AG, *Arduino - attachinterrupt()*, https://www.arduino.cc/en/Reference/attachInterrupt.

[Ardd] Arduino AG, *Arduino - constrain()*, https://www.arduino.cc/en/reference/constrain.

[Arde] Arduino AG, *Arduino - pulsein()*, https://www.arduino.cc/en/Reference/pulseIn.

[Ardf] Arduino AG, *Arduino - softwareserial library*, https://www.arduino.cc/en/Reference/softwareSerial.

[Ardg] Arduino AG, *Arduino uno & genuino uno*, https://www.arduino.cc/en/main/arduinoBoardUno.

[Atm15] Atmel, *Atmega48a/pa/88a/pa/168a/pa/328/p*, November 2015, http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf.

[Bea] B. Beauregard, *Arduino pid library*, http://playground.arduino.cc/Code/PIDLibrary.

[Bon16] E. Bonet, *Need help to set pwm frequency to 25khz*, June 2016, https://arduino.stackexchange.com/questions/25609/need-help-to-set-pwm-frequency-to-25khz.

[But] A. Butterfield, *Drag and control*, https://www.che.utah.edu/outreach/module?p_id=10.

[Car] Carnegie-Mellonova univerzita a Charmed Labs, *Cmucam5 pixy*, http://cmucam.org/projects/cmucam5.

[CQ] C. Caro and N. Quijano, *Low cost experiment for control systems*, http://www.engr.sjsu.edu/bjfurman/courses/ME190/Research_paper_assignment_related/Low%20Cost%20Experiment%20for%20Control%20Systems.pdf.

[Elea] ElecFreaks, *Ultrasonic ranging module hc - sr04*, https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.

[Eleb] G. Elert, *Aerodynamic drag*, http://physics.info/drag/.

[Eli] P. Elias, *Ping-pong ball in tube - student lab prototype with pid contoller*, https://www.youtube.com/watch?v=425P9TLMg9Y.

[EOR05] J. M. Escano, M. G. Ortega, and F. R. Rubio, *Position control of a pneumatic levitation system*, 10th IEEE Conference on Emerging Technologies and Factory Automation 2005 ETFA 2005 **1** (2005).

[Fer] L. Fernekes, *Laser pong one*, https://www.youtube.com/watch?v=zwG5c8IyJMI.

[Gue] R. Guenette, *Levitating ping pong ball*, https://www.youtube.com/watch?v=Bb_LYKEbkf8.

[Hala] N. Hall, *Mass flow rate*, https://www.grc.nasa.gov/www/K-12/airplane/mflow.html.

[Halb] ———, *What is drag?*, https://www.grc.nasa.gov/WWW/K-12/airplane/drag1.html.

[HBB15] J. Hillard, K. Branch, and A. Butterfield, *Teaching fluid dynamics with the ball-in-tube device*, International Journal of Mechanical Engineering Education **43** (2015), 15–22.

[Int04] Intel Corporation, *4-wire pulse width modulation (pwm) controlled fans specification*, 1.2 ed., July 2004, http://www.formfactors.org/developer/specs/rev1_2_public.pdf.

[ITe10] ITead Studio, *Ultrasonic ranging module: Hc-sr04*, November 2010, ftp://imall.iteadstudio.com/Modules/IM120628012_HC_SR04/DS_IM120628012_HC_SR04.pdf.

[ITe14] ITead Studio, *Ultrasonic ranging module: Hc-sr04*, June 2014, https://www.itead.cc/wiki/Ultrasonic_Ranging_Module_HC-SR04.

[K+] M. Kintel et al., *Openscad the programmers solid 3d cad modeller*, http://www.openscad.org/.

[LG11] J. Lepka and P. Grasblum, *Mikroprocesorová technika v aplikacích řízení elektrických pohonů*, http://www.crr.vutbr.cz/system/files/prezentace_09_1106.pdf, June 2011.

[Mar04] R. Martinek, *Senzory v průmyslové praxi*, 1. ed., Nakladatelství BEN, 2004.

[Mar07] L. S. Marks, *Marks' standard handbook for mechanical engineers*, eleventh ed., McGraw-Hill Education, 2007.

[Mat] MathWorks, *Pid tuner*, https://www.mathworks.com/help/control/ref/pidtuner-app.html.

[Mod02] O. Modrlák, *Fuzzy řízení a regulace*, https://www.kirp.chtf.stuba.sk/~bakosova/wwwRTP/tar2fuz.pdf, 2002.

[Pol] Pololu Corporation, *Pololu carrier with sharp gp2y0a60szlf analog distance sensor 10-150cm, 5v*, https://www.pololu.com/product/2474.

[PY98] K. M. Passino and S. Yurkovich, *Fuzzy control*, 1. ed., Addison-Wesley Longman, Inc., 1998.

[RDKN05] P. Ripka, S. Daďo, M. Kreidl, and J. Novák, *Senzory a převodníky*, Vydavatelství ČVUT, 2005.

[SHA] SHARP, *Gp2y0a60sz0f/gp2y0a60szlf*, http://www.sharp.co.jp/products/device/doc/opto/gp2y0a60szxf_e.pdf.

[Smi] M. Smids, *Ping pong ball levitation*, https://www.youtube.com/watch?v=jd00UtIx0pA.

[Sou06] J. Southard, *12.090 introduction to fluid motions, sediment transport, and current-generated sedimentary structures, chapter 3*, Massachusetts Institute of Technology, 2006, MIT OpenCourseWare, https://ocw.mit.edu. Licence: Creative Commons BY-NC-SA.

[SUN12] SUNON, *Pf40281b1-000u-s99*, 5 ed., July 2012, http://www.tme.eu/cz/Document/04ecbbafd5c3a23dfb7428f2eeb8581e/PF40281B1-S99-DTE.pdf.

[Tex04] Texas Instruments, *Uln2803a darlington transistor array*, revised ed., August 2004, https://www.sparkfun.com/datasheets/IC/uln2803a.pdf.

[Ucu] H. Ucuncu, *Levitating ping pong ball control system*, https://www.youtube.com/watch?v=NdCptwUShdQ.

[Ver] Vernier Software & Technology, *Pid ping-pong ball levitation*, http://www2.vernier.com/sample_labs/EPV-12-pid_ping_pong_ball.pdf.

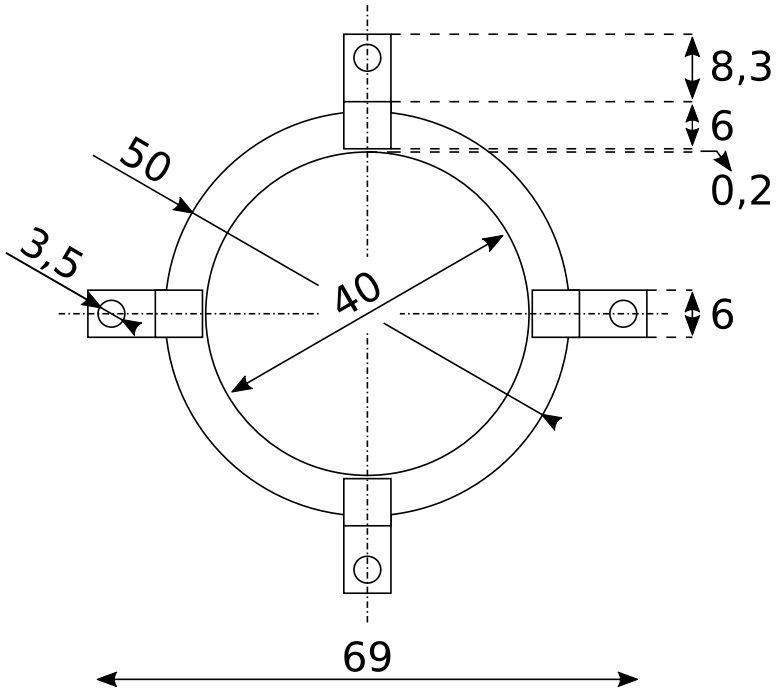
[Če] České vysoké učení technické, Fakulta elektrotechnická, *A3b35ari - automatické řízení*, http://www.fel.cvut.cz/cz/education/bk_peo/predmety/12/53/p12538304.

[Še13] M. Šebek, *Automatické řízení 10 - přímá vazba, feedforward*, http://www.polyx.com/_ari/slajdy/Bas-ARI-10-Feedforward.pdf, 2013.

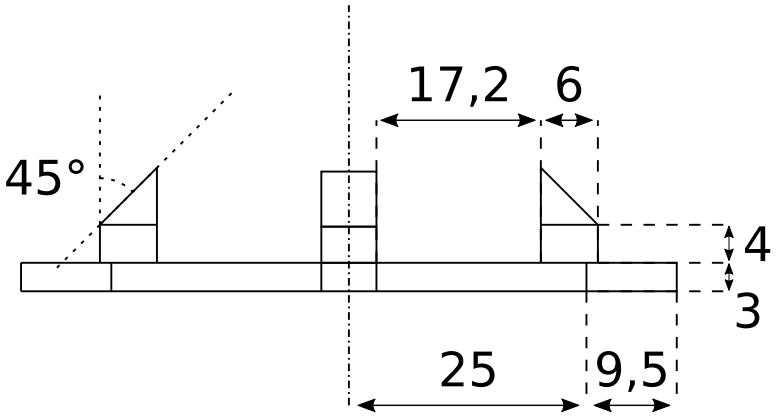
[Še17] ———, *Automatické řízení 5 - identifikace*, http://www.polyx.com/_ari/slajdy/Bas-ARI-05-Identification.pdf, 2017.

Příloha B

Výkresy spojovacích dílů

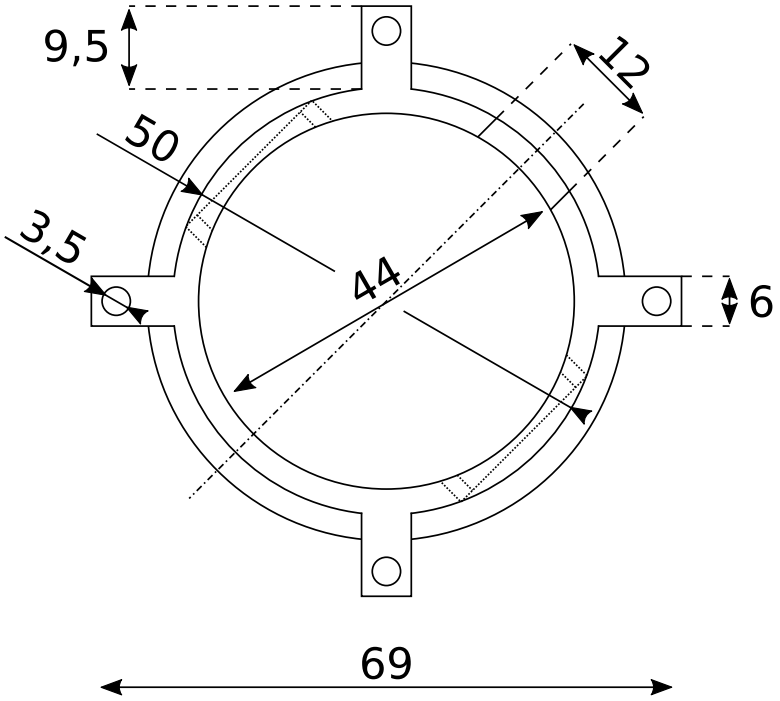


(a) : Pohled shora

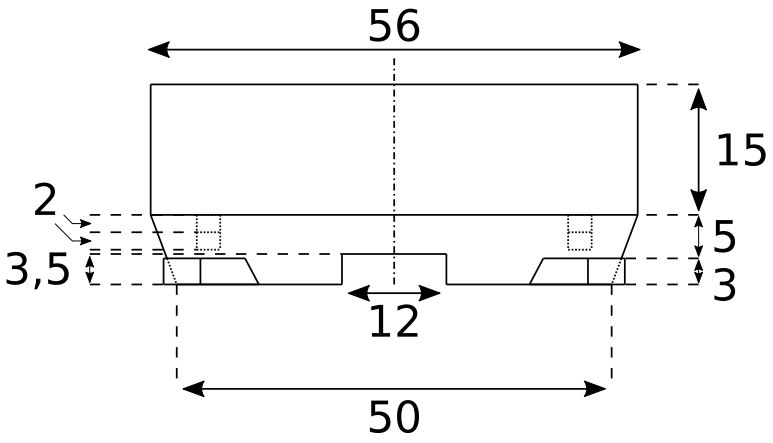


(b) : Pohled ze strany

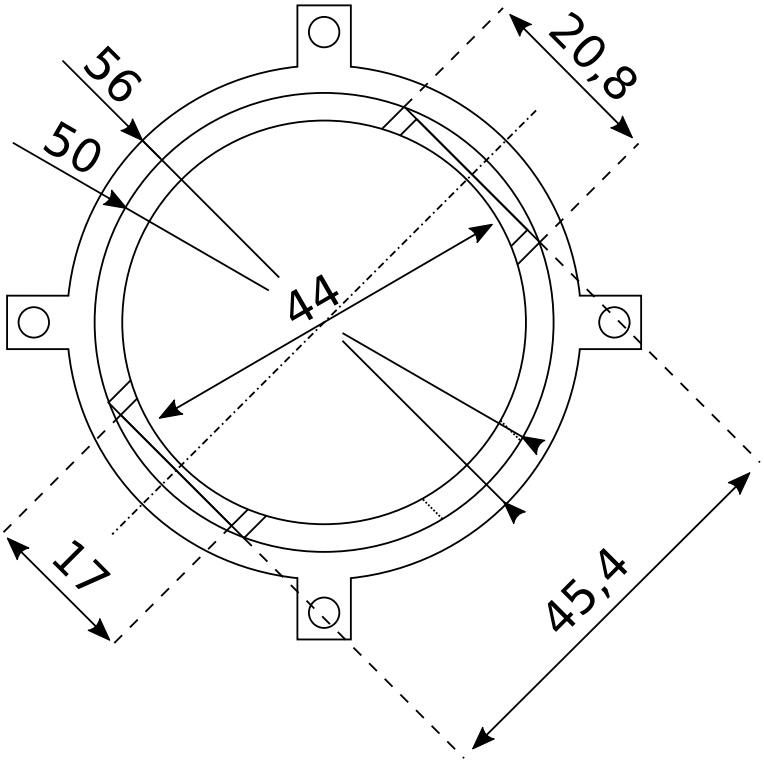
Obrázek B.1: Úchyt pro větrák



(a) : Pohled zdola



(b) : Pohled ze strany



(c) : Pohled shora

Obrázek B.2: Násada na trubku s místem na senzor

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Klapálek Jaroslav**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Vzduchová levitace dvou ping-pongových míčků**

Pokyny pro vypracování:

1. Dokončete a zdokumentujte systém pro vzduchovou levitaci dvou ping-pongových míčků v trubici. Systém musí být schopný měřit a řídit otáčky větráku a polohu obou míčku.
2. Vytvořte matematický model systému pro jeden a dva levitující míčky. Experimentálně identifikujte neznámé parametry systému.
3. Navrhněte řídicí systém pro řízení polohy jednoho míčku a analyzujte možnosti řízení polohy dvou míčků.
4. Systém upravte tak, aby se dal snadno rozšiřovat a byl vhodný pro prezentační účely. Navrhněte demonstrační a interaktivní režimy provozu.

Seznam odborné literatury:

[1] Franklin, Gene F., et al. Feedback control of dynamic systems. Vol. 3. Reading, MA: Addison-Wesley, 1994.
[2] Brown, Forbes T. Engineering system dynamics: a unified graph-centered approach. CRC press, 2006.
[3] Ping Pong Spectrum Analyzer. Hackaday, April 28, 2016. http://hackaday.com/2016/04/28/ping-pong-spectrum-analyzer/ [online]

Vedoucí: Ing. Jiří Zemánek

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 21. 2. 2017