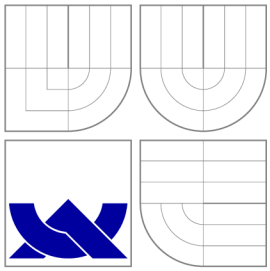
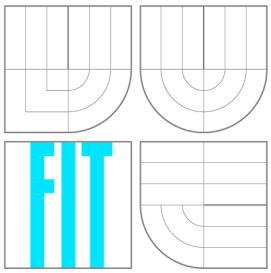


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# PROTINÁRAZOVÝ SYSTÉM KVADROKOPTÉRY POMOCÍ SADY SONARŮ

ANTI-CRASH SYSTEM FOR QUADCOPTER WITH SET OF SONARS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADIM HRAZDIL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍTĚZSLAV BERAN, Ph.D.

BRNO 2015

# Abstrakt

Tato bakalářské práce se zabývá osazením kvadrokoptéry ultrazvukovými senzory, jejich připojením k mikrokontroléru Arduino Micro a implementací ovladače pro řízení sonarů. Je navržena a implementována demonstrační aplikace využívající ROS framework, která v reálném čase detekuje překážky na základě výsledků měření. Při použití na kvadrokoptéře je od těchto překážek schopna udržovat minimální vzdálenost nebo snížit rychlost pohybu směrem ke překážce. Při návrhu systému byl kladen hlavní důraz hlavně na frekvenci měření instalovaných sonarů a pokrytí co největšího prostoru okolo kvadrokoptéry.



# Abstract

This bachelor's thesis is focused on installing ultrasonic sonars on a quadcopter, connecting them to the Arduino Micro microcontroller and implementing a program for controlling connected sonars. A demonstrational application using ROS framework was designed and implemented. Based on data received from sonars this application detects nearby obstacles and is able to maintain safe distance as well as slow down when getting too close to an obstacle. The design of the collision avoidance system is focused on high frequency of measurements and maximal space coverage.



# Klíčová slova

Protinárazový systém, kvadrokoptéra, senzory, ultrazvukové sonary, detekce překážek, ROS, Arduino Micro

# Keywords

Collision avoidance system, quadcopter, sensors, ultrasonic sonars, obstacle detection, ROS, Arduino Micro


# Citace

Radim Hrazdil: Protinárazový systém kvadrokoptéry
pomocí sady sonarů, bakalářská práce, Brno, FIT VUT v Brně, 2015

# Protinárazový systém kvadrokoptéry pomocí sady sonarů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing, Vítězslava Berana, Ph.D.

.....
Radim Hrazdil
19. května 2015

## Poděkování

Chtěl bych poděkovat panu Ing. Vítězslavu Beranovi, Ph.D. za odbornou pomoc, vedení správným směrem a cenné rady, které mi pomáhaly v průběhu realizace celé práce.

# Obsah

# Kapitola 1

# Úvod

Účelem této práce je vytvořit měřící zařízení s využitím ultrazvukových sonarů a protinárazový systém pro kvadrokoptéru. Kvadrokoptéry jsou vrtulníky se čtyřmi rotory, jejichž výhoda oproti klasickým vrtulníkům spočívá v jednodušší mechanické konstrukci. Na druhé straně však vyžadují vysokou přesnost řízení jednotlivých rotorů. Výkonné modely jsou schopny nést až několik kilogramů zátěže, což umožňuje osazení kamerovými systémy, senzory nebo dokonce výkonnými miniaturními počítači. Proto není divu, že začínají nacházet uplatnění ve vojenské i komerční sféře. Nicméně uživatelé, kteří nemají s létáním s kvadrokoptérou žádné nebo minimální zkušenosti, mohou mít s ovládáním i profesionálních strojů velké problémy. Situaci by měl významně ulehčit právě tento systém, který za uživatele hlídá vzdálenost od okolních překážek. V případě, že začátečník špatně odhadne směr natočení kvadrokoptéry, nebo její vzdálenost od překážky, protinárazový systém nedovolí uživateli narazit. Vzhledem k pořizovacím cenám profesionálních letounů může tento systém zamezit významným peněžním ztrátám.

Cílem je vytvořit protinárazový systém s využitím ultrazvukových sonarů. S tím souvisí navržení rozmístění a zapojení sonarů. K sonarům je třeba připojit mikrokontrolér, který je bude schopen řídit a získávat z nich údaje o měření. Naměřená data je pak třeba analyzovat. K tomu by měly sloužit uzly implementované pro Robotický operační systém (ROS). Tyto uzly budou data získaná ze senzorů zpracovávat, a v případě nutnosti zabraňovat uživateli vrazit s kvadrokoptérou do překážky.

# Kapitola 2

# Technologie

Tato kapitola seznamuje čtenáře s technologiemi, principy a nástroji použitými při návrhu a vývoji samotného HW zařízení i SW nástrojů. Jsou představeny existující projekty zabývající se podobnou problematikou – využitím ultrazvukových sonarů pro detekci a vyhýbání se překážkám. Dále je vysvětlen vlastní princip funkcionality ultrazvukových senzorů a jejich vlastností. Jsou také popsány situace, kdy ultrazvukové senzory selhávají a na co je třeba dávat pozor při jejich využití na kvadrokoptéře. Následující podkapitola čtenáři představí platformu AscTec Pelican, na které je školní kvadrokoptéra – Tyra[1] – postavena, a na kterou mají být sonary instalovány. Další část je věnována robotickému operačnímu sytému, který je na Tyře nainstalován, a který ji umožňuje ovládat. V závěrečné části je popsán princip filtru typu dolní propusť, využitého při zapojování sonarů pro stabilizaci napětí a zpětnovazební regulace, využité při implementaci přistávacího regulátoru na mikrokontrolér Arduino.

## 2.1 Existující řešení

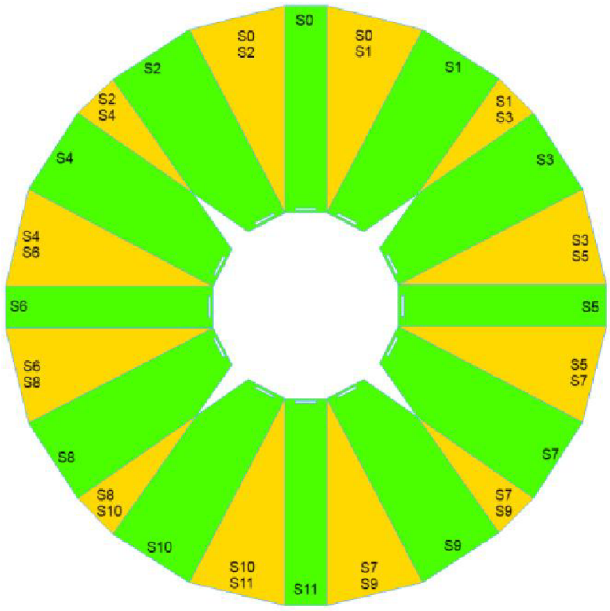
V současnosti je na trhu mnoho modelů multikoptér, které nabízejí různé úrovně autonomie s využitím různých technologií. V této podkapitole představuji existující projekty, které se zabývají podobnou problematikou jako tato práce. Tyto projekty vznikly na německých univerzitách ve městech Würnzburg a Bonn. Autoři prvního projektu se spolehli téměř výhradně na využití ultrazvukových sonarů, zatímco autoři druhého projektu osadili jejich letoun několika různými typy senzorů.

### Detekce překážek s využitím optických a ultrazvukových sonarů

První projekt vznikl na německé univerzitě ve Würnzburgu [3]. Byly použity ultrazvukové sonary v kombinaci s infračervenými senzory. Na kvadrokoptéru bylo umístěno celkem 14 senzorů, z toho 12 sonarů po straně kvadrokoptéry a 2 senzory pro měření výšky (jeden ultrazvukový a jeden infračervený).

Z obrázku 2.1 je vidět, že díky většímu množství sonarů se signály jednotlivých sonarů překrývají, což umožňuje až dvojnásobně zvýšit přesnost detekce pozice překážky. Je-li například detekována překážka sonarem S1, ale sonary S0 a S3 žádnou překážku nedetekují,

[1] http://www.fit.vutbr.cz/research/groups/robo/eqp.php.cs




Obrázek 2.1: Intervaly vzniklé navzájem se překrývajícími signály sonarů. Zdroj:[3]

znamená to, že se překážka nachází pouze v prostoru sonaru S1. Umístěním sonarů blízko k sobě ale samozřejmě vznikl problém s přeslechy, který byl řešen rozdělením sonarů do třech skupin, které se při současném měření neovlivňovaly. Vyhýbání se překážkám je řešeno vytvořením tří zón v každém směru pohybu:

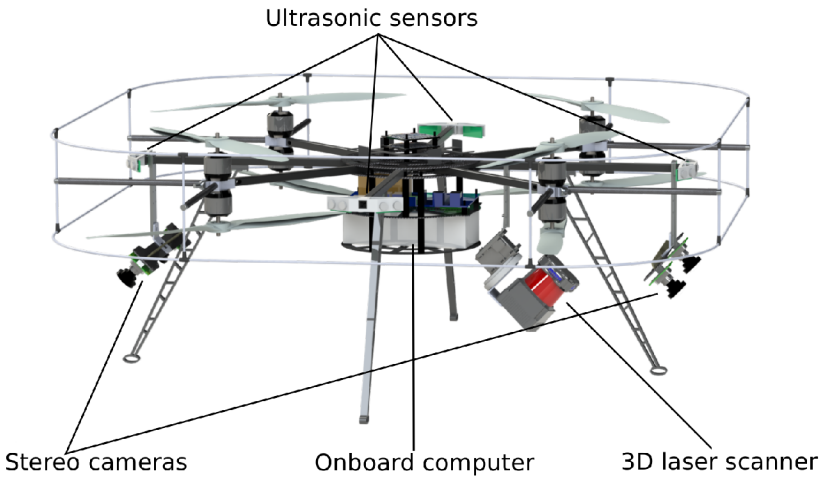
- bezpečná zóna,
- zóna s detekovanou překážkou,
- nebezpečná zóna.

Chování modulu, který zajišťuje vyhýbání se překážkám, může být popsáno konečným automatem se třemi zmíněnými stavy. Je-li kvadrokoptéra ve stavu, kdy je poblíž detekována překážka, jsou aplikovány rychlostní limity ve směru k této překážce. Při vstupu do nebezpečné zóny je pak vzdálenost od překážky kontrolována PID kontrolérem, který neumožní další přiblížení směrem k překážce.

Z výsledků experimentů tohoto projektu vyplývá, že za určitých okolností jsou ultrazvukové sonary velmi přesné, s odchylkami pouze 1–2 cm. Slabinou jsou ale měkké povrchy, jako například oblečení, pěna, tráva a podobně. Ultrazvukové sonary jsou ale na druhou stranu schopny měřit vzdálenost zdi skrz kouř, detekovat sklo či vodu, což jsou situace, kde optické senzory selhávají. Pro co nejlepší výsledky a co nejvyšší spolehlivost je tak nutné kombinovat několik typů senzorů.

## Všesměrová detekce překážek

Autoři této práce využili kromě osmi ultrazvukových sonarů ještě dva páry stereokamer, jednu klasickou kameru, 3D rotující skener pro snímání aktuálních vzdáleností ve všech směrech a GPS modul. Rozmístění zmíněných senzorů je naznačeno na obrázku 2.2.




Obrázek 2.2: Model oktokoptéry s osazenými senzory. Zdroj: [5]

Ke komunikaci mezi jednotlivými komponentami výsledného software byl využit ROS framework. Ve výsledku je pokryt téměř celý prostor kolem oktokoptéry, výjimkou je pouze malý prostor nad letounem [5]. Zpracování dat ze senzorů a kamer obstarává výkonný palubní počítač, umístěný ve středu letounu, vybavený procesorem Intel Core i7 [2] a 8 GB operační paměti.

Pro řízení ultrazvukových sonarů je použit mikrokontrolér Crumbuino-Mega s procesorem ATmega2560[3], kompatibilní s Arduinem. Sonary jsou umístěny po obvodu oktokoptéry a mají za úkol detekovat malé překážky, jako například větve stromů nebo dráty elektrického vedení. Konkrétně se jedná o model Devantech SRF10[4] s minimální operační vzdáleností 4 cm a maximálním dosahem 6 m.

Chybovost měření ultrazvukových sonarů je řešena tak, že v úvahu jsou brány jen ty hodnoty, které jsou naměřeny opakovaně. Zde by bylo možné řešení vylepšit použitím sofistikovanějšího filtru, například Kalmanova filtru[5].

Autoři se v této práci nicméně nespoléhají pouze na sonary, ale na kombinaci údajů z více typů senzorů, kterými je jejich letoun osazen. Například pro měření absolutní výšky je využíváno ultrazvukového senzorů pouze do vzdálenosti 5 m nad zemí. Při dosažení vyšší výšky je pak údaj o měření získáván z 3D rotujícího laseru, který je schopen přesně měřit až do vzdálenosti 30 m. Nevýhodou je však znatelně nižší frekvence měření a to 2 Hz.

Výsledky experimentů ukázaly, že výsledné řešení antikolizního systému je schopné se vyhnout i pohyblivým překážkám. Na demonstračním videu[6] je letounu zadána GPS pozice a výška nad zemí, kterou má udržovat. Při průchodu chodce pak oktokoptéra opustí zadanou pozici a umožní chodci bezpečně projít. Po bezpečném průchodu chodce pak letoun opětovně zaujme zadanou pozici.

[2] http://www.intel.com/content/www/us/en/processors/core/core-i7-processor.html
[3] https://www.chip45.com/products/crumbuino-mega_arduino_compatible_atmega2560_module_board_usb.php
[4] http://www.robot-electronics.co.uk/htm/srf10tech.htm
[5] http://www.uamt.feec.vutbr.cz/~richter/vyuka/0910_mpov/tmp/kalman_filter.html.cs
[6] http://www.ais.uni-bonn.de/MoD

## 2.2 Ultrazvukové sonary

Ultrazvukové senzory jsou bezdotykové detektory objektů v rámci určité oblasti, jejíž rozsah a tvar je dán vlastnostmi sonaru. Tyto senzory využívají pro detekci objektů akustické vlny o vysoké frekvenci, nejsou tedy ovlivněny například barvou objektu. Princip těchto senzorů spočívá ve měření doby letu zvuku k překážce a zpět.

Pro Tyru byly vybrány sonary MB1220 XL[7] od firmy Maxbotix. Jedná se o sonary pro obecné použití, vhodné jak pro vnitřní, tak i venkovní prostory. Nabízí vyvážený poměr mezi detekcí malých a velkých předmětů, toleranci proti rušení a pro kvadrokoptéru výhodný rozsah minimální a maximální vzdálenosti detekce překážky.

Vybraný model sonaru měří objekty vzdálené 20–765 cm s výrobcem udávanou odchylkou 1 cm. Objekty umístěné blíže než 20 cm nebo dále než 765 cm, sonar naměří jako vzdálené 20, respektive 765 cm. Prakticky tak nemá žádnou mrtvou zónu. Naměřenou hodnotu je možné získat měřením délky pulzu nebo hodnoty napětí. Kromě toho sonary ještě nabízí i sériovou linku RS232. Další vlastnosti:

- požadované napětí napájení je 3,3 nebo 5 V,
- možnost zřetězení sonarů pro aplikace, kde hrozí přeslechy,
- frekvence měření až desetkrát za sekundu.

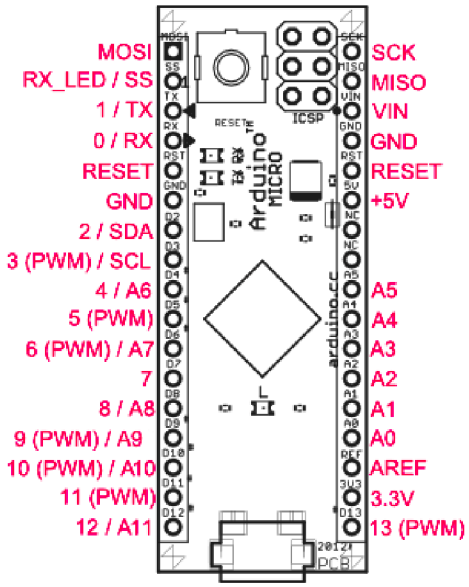
Význam jednotlivých pinů použitých sonarů je detailně popsán v příloze B. Další podrobnější informace je možné nalézt v technické dokumentaci sonarů [7].

Pro ovládání a kontrolu sonarů byl zvolen mikrokontrolér Arduino Micro, a to především pro jeho minimalistické provedení. Arduino je navíc otevřená platforma zahrnující samotný mikrokontrolér i vývojové prostředí. Arduino Micro mikrokontrolér je založený na architektuře ATmega32u4, vyvíjen ve spolupráci s firmou Adafruit[8]. Nabízí 20 digitálních a 12 analogových pinů, jejichž rozložení a funkce jsou zobrazeny na obrázku 2.3. Frekvence jádra je 16 MHz a pro připojení k počítači je možné využít mikro USB konektor, který je schopen mikrokontrolér bez problémů napájet. Je nicméně možné využít i externího zdroje, v tom případě je doporučené napětí 7 až 20 voltů. Pro komunikaci s počítačem nebo s jiným mikrokontrolérem je možné využít sériové periferní rozhraní (SPI), I2C nebo USB. Jak pro SPI, tak pro I2C jsou připraveny knihovny[9], které využití těchto rozhraní významně usnadňují.

Důležitou roli při využití ultrazvukových senzorů nejen na kvadrokoptéře hraje jejich rozmístění. Rozmístění sonarů totiž přímo ovlivňuje jejich schopnost spolehlivě pracovat. Ultrazvukové senzory jsou během činnosti ovlivňovány okolními zdroji rušení, jako například proudem vzduchu nebo výkyvy elektrického proudu. Dalším problémem mohou být vibrace samotného rámu kvadrokoptéry, na kterém jsou sonary připevněny.

Vzdušné proudy vzniklé v důsledku rotace vrtulí mohou změnit směr a intenzitu zvukových vln generovaných sonarem a výrazně tak ovlivnit výsledek měření. Důsledkem vzdušných proudů je totiž rozptýlení části energie generovaného akustického impulzu. Sonar pak naměří větší vzdálenost k překážce, než jaká je skutečná vzdálenost překážky. Je proto třeba umístit sonary tak, aby se směr zvukových impulzů neprotínal s oblastmi nad a pod

[7] http://www.maxbotix.com/documents/XL-MaxSonar-EZ_Datasheet.pdf
[8] https://www.adafruit.com/
[9] http://arduino.cc/en/Reference/Libraries
[10] http://www.arduino.cc/en/Main/arduinoBoardMicro




Obrázek 2.3: Schéma znázorňující rozložení a funkcionalitu pinů Arduina Micro[10].

vrtulemi. Pro sonary měřící výšku a prostor nad kvadrokoptérou je typické využít místo co nejblíže ke středu kvadrokoptéry.

Akustický hluk vydávaný motory a rotací samotných vrtulí má podobný efekt jako vzdušné proudy. Důsledek je ale opačný. Pokud je sonar umístěný směrem k takovému zdroji rušení, tak registrují část této energie, což může vést k naměření kratší vzdálenosti.

V aplikacích, kde je využito většího počtu sonarů je třeba brát v ohled způsob, jakým budou sonary napájeny, uzemněny a jak z něj budou čteny samotná data. Je-li zdroj napájení rušen nebo nestabilní, pak mohou být čtená data degradována, i když výstup samotného sonaru není ničím ovlivněn. Nabízí se dva způsoby zapojení, a to zapojení do hvězdy nebo do série [2].

Při zapojení do hvězdy nehrozí, že by se jednotlivé sonary ovlivňovaly mezi sebou. Každý sonar je v tomto případě připojen vlastním párem kroucených vodičů. Pokud pak má jeden sonar zvýšený odběr, snížené napětí se nedotkne ostatních sonarů v tomto systému. Nevýhodou je ale velké množství kabeláže. Opačným extrémem je zapojení do série. Představa, že by byly všechny sonary zapojeny do série, je ale prakticky neproveditelná. Použité sonary mají ve špičce odběr 100 mA, což v praxi znamená výkyv napětí. Takové výkyvy by pak měly vliv na všechny zapojené sonary v sérii a výsledky měření by byly znehodnoceny. Je proto třeba dosáhnout kompromisu mezi množstvím kabeláže a počtem sonarů zapojených do série. Vhodným řešením by mohlo být omezit počet sonarů zapojených v sérii a přidat do obvodu RC filtry (dolní propusť) pro eliminaci krátkých výkyvů napětí [4].

Je-li v rámci aplikace využito většího počtu sonarů, je třeba navrhnout bezpečný způsob, jakým mají sonary měřit. Pokud by všechny zahájily měření ve stejnou chvíli, s největší pravděpodobností by docházelo k přeslechům (zachycení akustických vln ostatních sonarů). K tomu jsou sonary vybaveny možností zřetězeného zapojení, kdy je mikrokontrolérem aktivován jeden sonar, který po dokončení měřícího cyklu aktivuje další sonar v pořadí. Tímto způsobem je zajištěno, že žádné dva sonary nebudou měřit ve stejnou chvíli.

I přes všechnu snahu o správné rozmístění sonarů na kvadrokoptéru jsou však situace, kdy ultrazvukové senzory selhávají. Na obrázku 2.4 jsou tyto situace znázorněny.

V prvním případě na obrázku 2.4 *(a)* je sonar namířen kolmo na překážku, nedochází tak k žádné chybě a naměřená vzdálenost je korektní. V případě *(b)* je však před měřenou překážkou umístěn malý objekt, který způsobí odraz akustické vlny. Naměřená vzdálenost je tak menší. Na obrázku *(c)* je sonar na překážku namířen z úhlu. Vzhledem k charakteristice signálu sonaru nedojde k přijetí odrazu akustické vlny z místa, kam je sonar namířen, ale z místa blíže k sonaru. To má za následek naměření kratší než měřené vzdálenosti. Naopak je tomu na obrázku *(d)*, kde úhel, pod kterým je sonar namířen na překážku, je příliš velký na to, aby se akustická vlna odrazila zpět. Sonar pak naměří výrazně větší vzdálenost, obvykle maximální možnou, protože akustická vlna se již nemá jak odrazit zpátky. Podobně je tomu na obrázku *(e)*, kde je sonar namířen na hranu překážky. Na obrázku *(f)* dochází k druhotným odrazům, v důsledku čehož je trajektorie akustické vlny delší a tím pádem naměřená vzdálenost větší než skutečná vzdálenost překážky.

Těmto situacím se lze za použití pouze ultrazvukových senzorů jen stěží vyhnout. Vzniklé nepřesnosti je možná částečně korigovat využitím různých filtrů, například Kalmanova filtru.

## 2.3 AscTec Pelican

Sestava AscTec Pelican[11] zahrnuje desku AutoPilot[12], která je osazena dvěma mikroprocesory ARM7 a několika rozhraními pro komunikaci. AutoPilot je připojen k Pico-ITX základní desce s výkonným procesorem Intel Atom (Atomboard), schopným provádět náročnější výpočty, zpracovávat video nebo plánovat trasu s využitím GPS. Schéma znázorňující komunikaci a rozhraní těchto komponent je na obrázku 2.5.

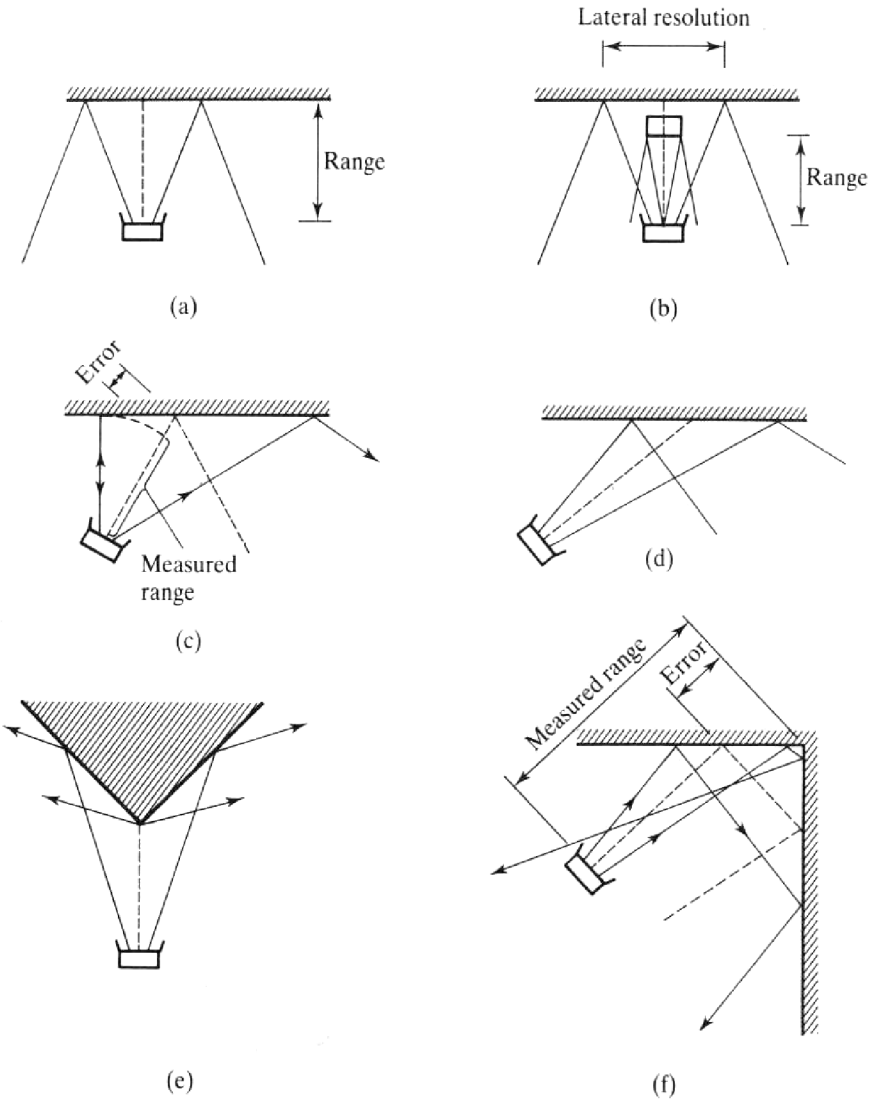
Low Level Procesor (dále jen LLP) zpracovává data z integrovaných senzorů a kontroluje stabilitu s frekvencí až 1 GHz. Také zpracovává údaje o aktuální pozici z integrovaného magnetometru a GPS modulu. Data ze senzorů lze získat pomocí sériového rozhraní, prostřednictvím kterého lze také zasílat příkazy pro pohyb letounu. V závislosti na letovém módu je možné zadávat i GPS souřadnice. Jsou dostupné tři letové módy:

- navigace pomocí GPS,
- mód držení výšky,
- manuální mód.

Dále mají všechny bezpilotní letouny AscTec záchranný mód pro případ ztráty spojení s dálkovým ovládáním. Smyslem záchranného módu je zabránit nekontrolovanému letu spuštěním záchranné procedury. K dispozici je několik záchranných procedur:

- okamžité přistání - klesání stálou rychlostí 1 m/s. Je-li k dispozici signál GPS, je udržována pozice, kde došlo ke ztrátě spojení s dálkovým ovládáním,
- počkej a přistaň - letoun vyčká 5 sekund v aktuální pozici pro případné navázání signálu, pak začne klesat,
- návrat domů - automatické přepnutí do GPS módu a návrat do místa, kde byly nastartovány motory. Pokud v momentě roztočení motorů není k dispozici GPS signál, nemůže být návrat realizován a dojde k přepnutí do módu okamžitého přistání.

[11] http://www.asctec.de/uav-uas-drohnen-produkte/asctec-pelican/
[12] http://wiki.asctec.de/display/AR/AscTec+AutoPilot



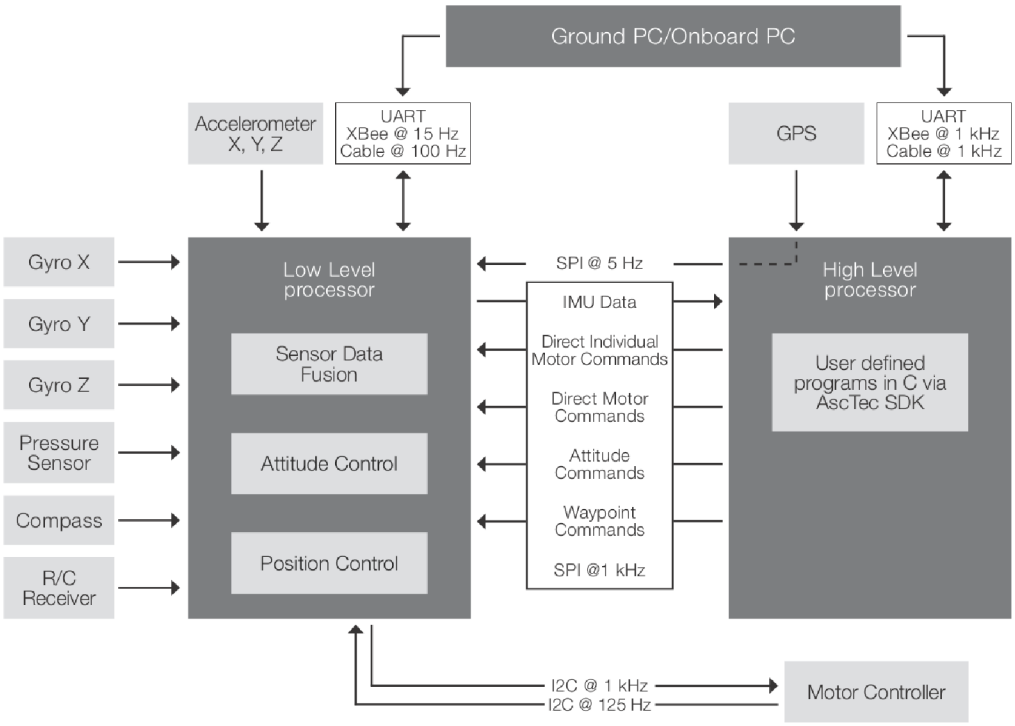

Obrázek 2.4: Situace, ve kterých ultrazvukové senzory selhávají. Zdroj: [9]

Na High Level Processor (dále jen HLP) může být nahrán uživatelský kód pro ovládání letu kvadrokoptéry. Uživatel má možnost kdykoli během testování přepnout zpět na LLP tlačítkem na dálkovém ovladači. To je užitečné například v situaci, kdy se uživatelský kód zacyklí nebo jiným způsobem neumožní kvadrokoptéru bezpečně ovládat. Komunikace mezi oběma mikroprocesory je dostatečně rychlá na to, aby měl HLP k dispozici data ze všech integrovaných senzorů, stejně jako LLP. HLP navíc nabízí rozhraní typu UART, I2C a SPI. Programování HLP zajišťuje AscTec Mav Framework pro ROS.

Tento framework obsahuje ovladače, nástroje a kontrolér pozice pro kvadrokoptéry vybavené deskou AutoPilot. Framework nezprostředkovává pouze komunikaci s HLP, ale umožňuje HLP také přímo programovat. Framework se skládá ze třech balíčků:

- asctec_hl_comm: definuje podobu zpráv,

[13] http://wiki.asctec.de/display/AR/AscTec+AutoPilot




Obrázek 2.5: Schéma rozhraní desek Autopilot a Atomboard[13].

- asctec_hl_interface[14]: rozhraní pro komunikaci s HLP, skripty pro ladění a monitorování,
- asctec_hl_fimrware: firmware, který musí být nahrán na HLP, aby mohl framework správně fungovat.

Velmi významným balíčkem je asctec_hl_interface, který obsahuje uzel hl_interface pro komunikaci s HLP. Tento uzel naslouchá na tématech (pojmy uzel a téma souvisí s ROSem a jsou vysvětleny v následující podkapitole):

- fcu/control: příkazy pro ovládání letu,
- fcu/pose a fcu/state: naslouchá informace o stavu a pozici kvadrokoptéry,
- fcu/ekf_state in: naslouchá informace od ethzasl_sensor_fusion[15] (framework pro vizuální odometrii).

AscTec Autopilot je pomocí UART připojen k miniaturnímu PC s procesorem Intel Atom. Na desce je nainstalován operační systém Ubuntu ve verzi 12.10 a již zmíněný ROS ve verzi Hydro. Atomboard nabízí množství rozhraní pro připojení periferních zařízení, jako například USB kamery nebo Arduina. Dále jsou k dispozici: WiFi modul, GPIO piny, VGA výstup, Ethernet a I2C. Na Tyře není použit přímo AscTec Atomboard, ale levnější řešení od firmy Advantech – MIO-2261, které nabízí velmi podobnou výbavu. Formát desky je Pico-ITX, což je po Mobile-ITX nejmenší dostupný formát na trhu, a je tak vhodný pro platformu Pelican.

[14] http://wiki.ros.org/asctec_hl_interface
[15] http://wiki.ros.org/ethzasl_sensor_fusion

## 2.4 Robotický operační systém

Robotický operační systém (dále jen ROS) je flexibilní open-source meta-operační systém určen pro UNIX systémy. Jde o kolekci knihoven a nástrojů pro vývoj robotického software. ROS poskytuje podobné služby jako standardní operační systém – abstrakce nad hardware, ovládání HW zařízení na nízké úrovni, zasílání zpráv mezi procesy (uzly), správa balíčků atd. Hlavní myšlenkou a důvodem ke vzniku ROS byla potřeba spolupráce na komplexních problémech. Vývoj software pro robotiku není jednoduchý, už jen kvůli velkému množství různých robotů složených z různého HW, což v mnoha případech omezuje možnosti znovupoužití zdrojových kódů.

Architektura ROS frameworku se podobá architektuře grafu. Jde v principu o množinu uzlů, které mezi sebou komunikují [8]. Základní komponenty jsou:

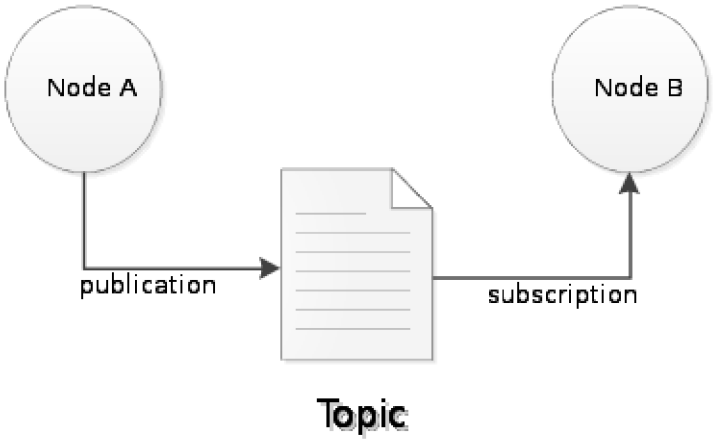
- uzly neboli procesy. Jde o klasické programy v C++ nebo Pythonu využívající ROS client library provádějící výpočet. Kontrolu celého robota obstarává obvykle mnoho uzlů,
- uzel Master poskytující registraci uzlů a umožňující komunikaci (zasílání zpráv) mezi jednotlivými uzly. V principu se jedná o podobnou službu jako je služba DNS[16],
- parameter Server (součást uzlu Master), který umožňuje uzlům ukládat data podle klíče,
- zprávy – datové struktury – pomocí kterých mezi sebou jednotlivé uzly komunikují. Každá zpráva má definovaný počet proměnných předem určeného datového typu,
- témata. Zprávy nejsou zasílány způsobem point-to-point (mezi jednotlivými uzly), ale jsou publikovány na dané téma. Tyto témata jsou jednoznačně identifikována jménem. Pokud si některý uzel, nebo několik uzlů, přejí získávat data určitého typu, zaregistrují se k odběru daného tématu,
- služby sloužící pro komunikaci typu dotaz-odpověď. Uzel, poskytující službu pod daným jménem, přijme dotaz od jiného uzlu, tento dotaz zpracuje a odešle odpověď.

Jednoduchý příklad komunikace mezi uzly je představen na obrázku 2.6, kde uzel A publikuje data na téma Topic, ze kterého data odebírá uzel B. Množství uzlů, které na Topic publikuje nebo z něj odebírá není omezené.

Výhodou tohoto principu zasílání zpráv je, že uzel B neví, jakým způsobem zprávy, které přebírá, vznikají. Správně navržený a implementovaný uzel by měl pracovat správně kdykoli je publikována zpráva, kterou se tento uzel zavázal odebírat.

Jedním z důvodů, proč by tak měly být uzly navrhovány, je nástroj `rosbag`[17], který ROS framework nabízí. Pomocí něj je možné zaznamenávat do souboru zprávy publikované na jedno nebo více témat, a později tyto zprávy opakovaně přehrávat. Tímto způsobem je možné velmi efektivně testovat určitý problém, protože s robotem stačí pouze jednou nahrát potřebná data, například údaje ze senzorů, a ty pak následně přehrávat a sledovat, zda testovaný uzel reaguje správně.

[16] http://cs.wikipedia.org/wiki/Domain_Name_System
[17] http://wiki.ros.org/rosbag



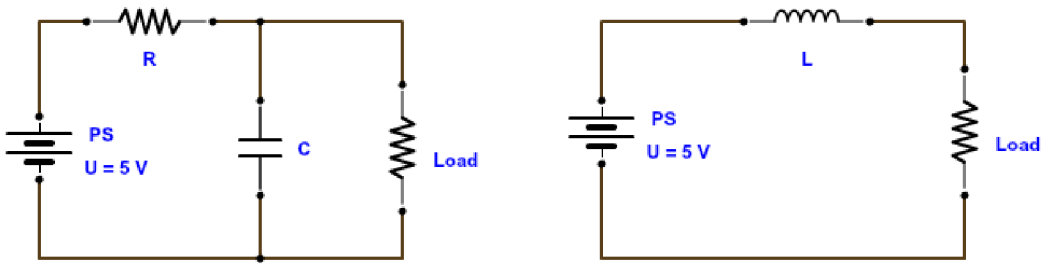

Obrázek 2.6: Příklad jednosměrné komunikace mezi uzly A a B.

## 2.5 Dolní propusť

Dolní propusť, neboli také horní zádrž, je obvod, který propouští nízké frekvence a eliminuje frekvence vysoké [6]. Konkrétně propouští frekvence nižší než mezní frekvence $f_0$[18] a naopak nepropouští frekvence vyšší. Existují dva typy obvodů, které jsou schopny plnit tento úkol – kapacitní a induktivní dolní propusť. Induktivní dolní propusť je možné realizovat pouhým zapojením cívky do série se zátěží. Při zvýšení frekvence se zvýší i impedance cívky, která pak zabraňuje průchod vysokofrekvenčním kmitům k zátěži. Příklad takového obvodu je na obrázku 2.7, kde zátěž je představována rezistorem.

Nevýhodou induktivního filtru jsou rušivé vlivy cívky. Proto se velmi často používá druhého typu dolní propusti, který je složen z kondenzátoru a rezistoru (obr. 2.7). V tomto typu obvodu je rezistor zapojen do série se zátěží a kondenzátor je zapojen paralelně se zátěží. Při nízkých frekvencích kondenzátor projevuje vysokou kapacitanci a blokuje nízkofrekvenční signály, což má za následek, že proud do zátěže teče přes rezistor. Při vyšších frekvencích kapacitance kondenzátoru klesá. Kondenzátor pak kompenzuje poklesy napětí napájecího zdroje. Mezní frekvence pro RC článek je dána následujícím vztahem [6].

$$f_0 = \frac{1}{2\pi RC}$$




Obrázek 2.7: RC a LC filtr eliminující krátké výkyvy napětí (dolní propusť). Zdroj [6]

[18] http://dynsys.uml.edu/tutorials/1st_Order_Systems/Cutoff%20Frequency.pdf

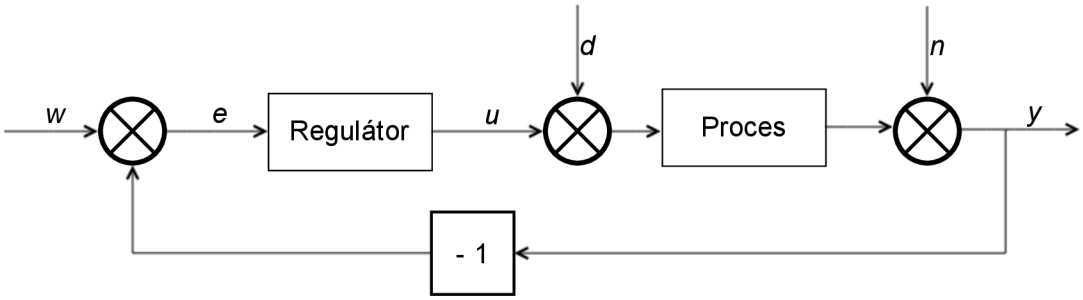
## 2.6 Zpětnovazební regulace

Proporcionálně-integračně-derivační (dále jen PID) regulátory jsou nejpoužívanějšími zpětnovazebními regulátory v průmyslu. Většina zpětnovazebních smyček je ovládána právě tímto algoritmem nebo jeho variací [10]. Uvažujeme-li proces, jehož výstupní veličina se zvýší, pokud se zvýší i jeho vstupní veličina, pak lze princip zpětné vazby popsat následujícími pravidly [1]:

1. Zvyš vstupní hodnotu systému, pokud je aktuální výstupní hodnota systému menší než požadovaná výstupní hodnota.

2. Sniž vstupní hodnotu systému, pokud je aktuální výstupní hodnota systému větší než požadovaná výstupní hodnota.

Tento typ zpětné vazby někdy také bývá nazýván záporná nebo negativní zpětná vazba, protože regulovaný vstup systému se pohybuje v opačném směru než výstup. Blokové schéma takového regulátoru je znázorněno na obrázku 2.8. Přítomnost negujícího bloku indikuje negativní vazbu. Hlavním požadavkem je, aby výstupní veličina $y$ (reprezentující aktuální stav) co nejpřesněji napodobovala vstupní veličinu $w$ (reprezentující požadovanou hodnotu výstupní veličiny) [11]. Přitom je důležité, aby tak činila bez ohledu na působení dynamických chybových veličin $d$,$n$ a dynamické změny systému. To regulátor zajišťuje generováním akční veličiny $u$ na základě chybové veličiny $e$. Chybová veličina je vypočítávána na základě předchozího výstupu a požadovaného výstupu regulovaného systému.




Obrázek 2.8: Blokové schéma regulační smyčky s negativní zpětnou vazbou. Zdroj: [10]

### Dvoustavový regulátor

Nejjednodušším zpětnovazebním regulátorem je dvoustavový regulátor. V tomto případě regulátor funguje tak, že nastaví maximální akční veličinu, pokud je chyba kladná a minimální akční veličinu, pokud je chyba záporná. Matematicky může být takový mechanismus popsán následovně:

$$u = \begin{cases} u_{max} & \text{pokud} \quad e > 0 \\ u_{min} & \text{pokud} \quad e < 0 \end{cases}$$

kde $e = w - y$ [1], tedy požadovaná hodnota minus naměřená hodnota. Výhodou tohoto regulátoru je, že není třeba nastavovat žádné parametry. I přes svou jednoduchost ale dokáže držet řízenou veličinu $y$ blízko k požadované hodnotě u většiny stabilních i nestabilních systémů. Nevýhodou však je generování maximální veličiny i při minimální změně

odchylky. Výsledkem jsou silné oscilace, které znemožňují použití tohoto regulátoru pro regulaci pohybu, ať už pozemního robota či kvadrokoptéry.

### P regulátor

Vylepšenou verzí dvoustavového regulátoru je proporcionální regulátor, který odstraňuje nedostatek generování maximální akční veličiny při minimální změně odchylky. Matematický popis proporcionálního regulátoru je následující:

$$u = \begin{cases} u_{max} & \text{pokud} \quad e > 0 \\ Ke + u_b & \text{pokud} \quad Ke + u_b \in \langle u_{min}, u_{max} \rangle \\ u_{min} & \text{pokud} \quad e < 0 \end{cases}$$

kde $K$ je zesílení regulátoru. Za hodnotu $u_b$ se obvykle volí polovina intervalu $\langle u_{min}, u_{max} \rangle$ [1]. Při odchylkách větších než $u_{max}$ a $u_{min}$ se však P regulátor chová stejným způsobem jako dvoustavový regulátor. Tento nedostatek řeší PI a PID algoritmy zpětnovazebného regulátoru.

### PID regulátor

PID regulátor má oproti P regulátoru navíc integrační (I) a derivační složku (D). Integrační složka zajišťuje (v ideálním případě) nulovou regulační odchylku v ustáleném stavu za předpokladu, že se nemění požadovaná hodnota $w$ a hodnoty poruch $d$ a $n$. Derivační složka je výsledkem další snahy vylepšit stabilitu smyčky s PI regulátorem. Derivační složka totiž umožňuje předvídat chování procesu a tuto znalost využívat pro výpočet akční veličiny. Chování PID regulátoru lze popsat vzorcem

$$u(t) = Ke(t) + \frac{1}{T_i} e(t)dt + T_d \frac{de(t)}{dt} \tag{2.1}$$

kde $e(t)$ je chyba v čase $t$, $T_i$ se nazývá integrační časová konstanta a $T_d$ derivační časová konstanta.

### Konfigurace PID regulátoru

Důležitou částí implementace zpětnovazebních algoritmů je nalézt správné konstanty $K$, $T_i$ a $T_d$. Nejčastěji se tak děje přímým experimentováním s uzavřenou smyčkou. Metodou pokus-omyl jsou voleny hodnoty parametrů PID regulátoru a podle tvaru odezvy na změnu požadované hodnoty, nebo uměle zavedený skok v poruše, je subjektivně usuzována vhodnost zvolených parametrů. Existují postupy, které tento proces mají zefektivnit. Jedním z nejznámějších a nejjednodušších postupů je následující [10].

1. Vypni integrační a derivační složku ($T_i$ a $T_d$ nastav na hodnotu 0). Postupně zvyšuj zesílené proporcionální složky až vzniknou trvalé kmity. Pak zmenši zesílení na polovinu.

2. Postupně zmenšuj integrační časovou konstantu ($T_i$) až vzniknou trvalé kmity. Poté ji zvětši třikrát.

3. Postupně zvětšuj derivační časovou konstantu ($T_d$) až nastanou trvalé kmity. Pak ji zmenši třikrát

# Kapitola 3

# Protinárazový systém

V této kapitole je čtenář seznámen s detaily zadání projektu a požadavky na výsledné řešení. Dále jsou zde na základě znalostí, získaných z předchozí kapitoly, prezentovány návrhy řešení dílčích částí práce. Tyto návrhy jsou seřazeny tak, aby na sebe navzájem navazovaly a aby důsledky předchozích úvah ovlivňovaly úvahy následující. Nejprve je řešen způsob rozmístění sonarů na Tyru, následně jejich zapojení, kalibrace a ovládání mikrokontrolérem Arduino Micro. Nakonec je představena možná architektura aplikace využívající nainstalovaných sonarů.

## 3.1 Analýza zadání

Řízení bezpilotního letounu dálkovým ovládáním je činnost, která na uživatele klade nemalé nároky. Při řízení má pilot obvykle omezený kontext určený zorným polem kamery nebo vzdáleností pilotovaného letounu. Kamera je často připevněna napevno a není tak možné jednoduše sledovat prostor a překážky po bocích řízeného stroje. Při přímém řízení kvadrokoptéry (bez použití kamery) je na větší vzdálenost problém určit, jakým směrem je natočena. V takovém případě může pilot ve snaze vzdálit se od překážky naopak do překážky narazit. Při řízení multikoptéry je navíc situace složitější tím, že na rozdíl od pozemních robotů je třeba navíc kontrolovat výšku a prostor nad letounem.

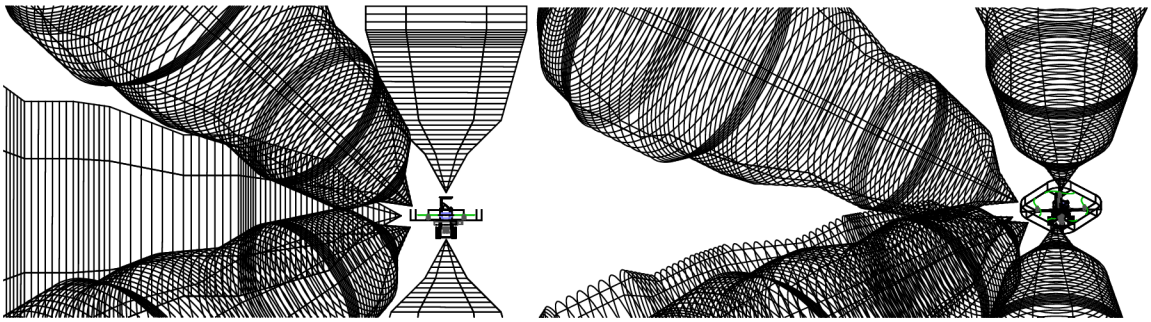
Cílem této práce je navrhnout způsob instalace sonarů na Tyru (kvadrokoptéra AscTec Pelican), vyrobit prototyp měřícího zařízení s využitím Arduina a implementovat balíčky pro ROS, které budou zpracovávat data ze sonarů a tvořit jádro protinárazového systému. tento protinárazový systém bude umožňovat jednodušší ovládání Tyry ve stísněných prostorách. Tyra by měla být schopná interpretovat příkazy od uživatele takovým způsobem, aby nedošlo ke kolizi s okolními překážkami. K tomu by mělo sloužit až 14 sonarů, které je třeba vhodně umístit na Tyru tak, aby okolo Tyry tvořily jakýsi ochranný obal. Zároveň by mělo být dosaženo co nejnižší hmotnosti celého zařízení.

Důležitou součástí této práce je navrhnout způsob, jak zmíněné množství sonarů na Tyru připevnit. Musí být rozmístěny na takových místech, aby se mezi sebou ovlivňovaly co možná nejméně. Tím pak bude možné dosáhnout vyšší frekvence měření. Je také třeba navrhnout způsob napájení, aby nebylo nutné vést ke každému sonaru samostatný pár vodičů. Dále také způsob komunikace mezi Arduinem a Atomboardem a nakonec architekturu ROS uzlů, které budou zpracovávat data naměřená Arduinem a na základě těchto dat reagovat na detekované objekty okolo Tyry.

## 3.2 Rozmístění sonarů

Způsob rozmístění sonarů na kvadrokoptéře ovlivňuje několik důležitých parametrů – pokrytí prostoru okolo kvadrokoptéry a frekvenci měření. Cílem je dosáhnout co nejvyššího pokrytí prostoru okolo Tyry a co nejvyšší možné frekvence měření.

Pro ověření správného pokrytí prostoru sonary byl v programu AutoCAD[1] na základě charakteristiky vybraných sonarů[2] vytvořen model signálu. Model kvadrokoptéry AscTec Pelican je volně k dispozici na GrabCad.com, jehož autorem je Daniel Lucena[3]. S vytvořenými modely sonarů jsem pak hledal jejich optimální rozmístění. Výsledek je znázorněn na obrázku 3.1, kde je pro lepší přehled zobrazeno pouze 5 sonarů. Sonary nejvíce vpravo jsou namířeny kolmo vzhůru a kolmo dolů. Sonar umístěný ve středu rámu Tyry je namířen vodorovně. Zbylé dva sonary umístěné v rohu jsou nakloněny pod úhlem 35° vzhledem k vodorovné rovině, což je úhel, při kterém se signály těchto sonarů ještě protínají. Navržené rozmístění pokrývá odhadem 90% prostoru okolo kvadrokoptéry a počítá dohromady s využitím 14 sonarů (4 po bocích, 8 v rozích, jeden v dolní části namířen kolmo dolů a jeden v horní části namířen kolmo vzhůru. Navíc umožňuje nezávislé měření sonarů umístěných na boku na sonarech umístěných v rozích Tyry.



Obrázek 3.1: Dva pohledy na Tyru s umístěnými modely sonarů. Pro lepší přehled je zobrazeno pouze 5 sonarů – horní, dolní, boční a dva rohové.

Po nalezení vhodného rozmístění sonarů bylo třeba stanovit, jakým způsobem vnímat prostor okolo Tyry a jakým způsobem modelovat Tyru samotnou. Na obrázcích je vidět, že sonary jsou od pomyslného středu Tyry téměř stejně daleko, a proto je výhodné okolo Tyry vytvořit jakýsi ochranný obal ve tvaru koule.

Vzhledem k vlastnostem ultrazvukových senzorů se jako vhodný způsob vnímání okolí jeví prostor, kde jsou všechny objekty navzájem pravoúhlé. Ke každé překážce, která bude detekována, pak bude přistupováno, jako by na daném místě byla zeď orientovaná kolmo k Tyře.

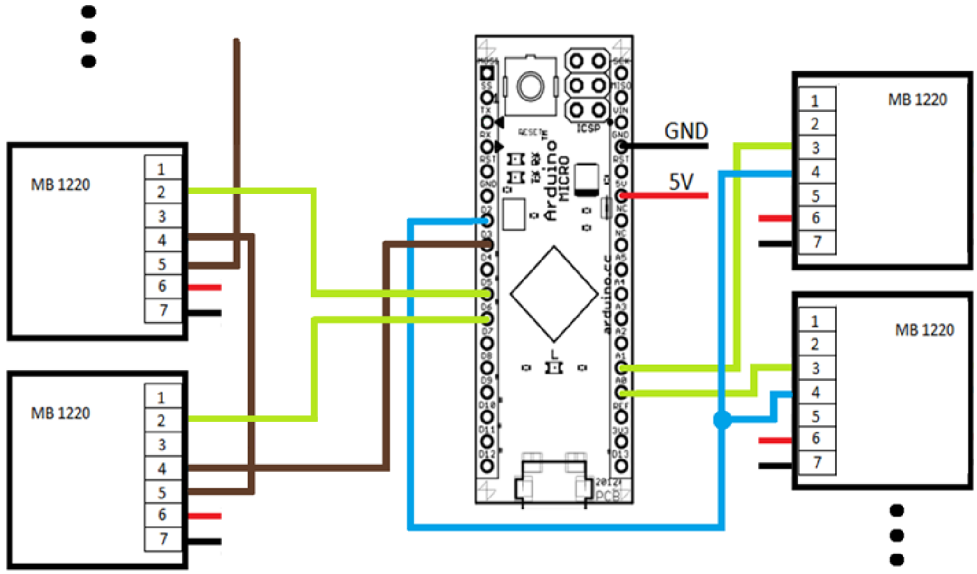
## 3.3 Zapojení sonarů

Vzhledem k tomu, že na Tyře by mělo být dohromady až 14 sonarů, je potřeba navrhnout zapojení s ohledem na omezený počet pinů Arduina. Pro ovládání sonarů je možné na Arduinu využít 12 digitálních a 6 analogových pinů. Je zřejmé, že pokud by sonary měly být zapojeny samostatně, tzn. 2 piny pro každý sonar, nebude na Arduinu dostatečný počet

[1] http://www.autodesk.com/education/free-software/autocad
[2] http://www.maxbotix.com/documents/XL-MaxSonar-EZ_Datasheet.pdf
[3] https://grabcad.com/library/asctec-pelican-quadrotor-1

pinů. Tento problém by šlo řešit i pomocí multiplexorů a analogových přepínačů. To by ale vyžadovalo další plošný spoj a množství kabeláže, což pro potřeby kvadrokoptéry není vhodné.

Lepším řešením by mohlo být využití jak analogového, tak i digitálního čtení výsledků, přičemž sonary po bocích kvadrokoptéry by byly spouštěny současně, protože mezi nimi nehrozí přeslechy a sonary umístěny v rozích kvadrokoptéry by byly zapojeny zřetězeně. Zároveň je třeba si uvědomit, že boční sonary jsou nejvíce kritické, a je třeba dosáhnout co nejvyšší frekvence měření. Pro spodní sonar by také měla být zajištěna co nejvyšší frekvence, aby bylo možné spolehlivě udržovat výšku nebo implementovat rozšiřující funkce, jako například automatické přistání. Informace ze sonaru umístěného v horní části Tyry a sonarů, které budou připevněny v rozích kvadrokoptéry, nejsou natolik důležité, a proto mohou být zapojeny zřetězeně a měřit po jednom.




Obrázek 3.2: Znázornění zapojení sonarů.

Výsledkem předchozích úvah je schéma zapojení znázorněné na obrázku 3.2. Sonary na levé straně jsou zapojeny zřetězeně. Startovací vodič (hnědý) vede z digitálního pinu Arduina D3 do pinu 4 levého spodního sonaru. Výsledek měření je Arduinu poslán vodičem (zelený) ve formě šířky pulzu. Po dokončení měření tento sonar ještě vygeneruje startovací pulz na pinu 5, který je připojen opět hnědým vodičem k dalšímu sonaru na startovací pin 4 a celý cyklus se opakuje až k poslednímu sonaru.

Na pravé straně obrázku 3.2 jsou sonary startovány současně vodičem zapojeného na digitálním pinu Arduina D2 (modrý). Tyto sonary jsou připevněny po bocích Tyry, proto je možné je spouštět současně. Výsledky měření jsou postupně čteny analogovými piny Arduina. Sonary po naměření nastaví příslušnou úroveň napětí na pinu 3, přičemž tato hodnota je pak držena až dokud není naměřen nový výsledek.

Arduino po přečtení naměřeného výsledku z některého z připojených sonarů vytvoří zprávu, kterou ihned pošle sériovou linkou počítači, ke kterému je Arduino připojeno USB kabelem. Zpráva je zasílána v následujícím formátu:

```
<cislo_sonaru>-<namerena_vzdalenost>\n
```

kde jsou sonary číslovány od 1 do 14 a naměřená vzdálenost je v centimetrech.

## 3.4 Kalibrace sonarů

Po zapojení sonarů k Arduinu a několika experimentech se ukázalo, že výsledky měření čtené analogově se liší od výsledků měřených digitálně. V technické dokumentaci jsem nenalezl informaci o tom, že by se hodnoty přečtené šířkou pulzu (digitálně) a analogově měly lišit. Bylo proto třeba sonary zkalibrovat tak, aby se výsledky sjednotily. K tomu byla provedena série měření, jejíž výsledky jsou v tabulce 3.1. Každá vzdálenost byla měřena pětkrát, v tabulce je uvedena průměrná hodnota.

| Měřená vzdálenost [cm] | naměřeno digitálně [cm] | naměřeno analogově [cm] |
|---|---|---|
| 20 | 25 | 19 |
| 30 | 38 | 27 |
| 40 | 49 | 35 |
| 50 | 62 | 44 |
| 60 | 75 | 52 |
| 70 | 88 | 62 |
| 80 | 100 | 71 |
| 90 | 113 | 80 |
| 100 | 126 | 89 |
| 110 | 139 | 98 |
| 120 | 152 | 109 |
| 130 | 165 | 117 |
| 140 | 178 | 126 |
| 150 | 191 | 135 |
| 160 | 204 | 145 |
| 170 | 217 | 156 |
| 180 | 230 | 164 |
| 190 | 243 | 174 |

Tabulka 3.1: Tabulka naměřených hodnot pro kalibraci analogového a digitálního čtení ze sonarů.

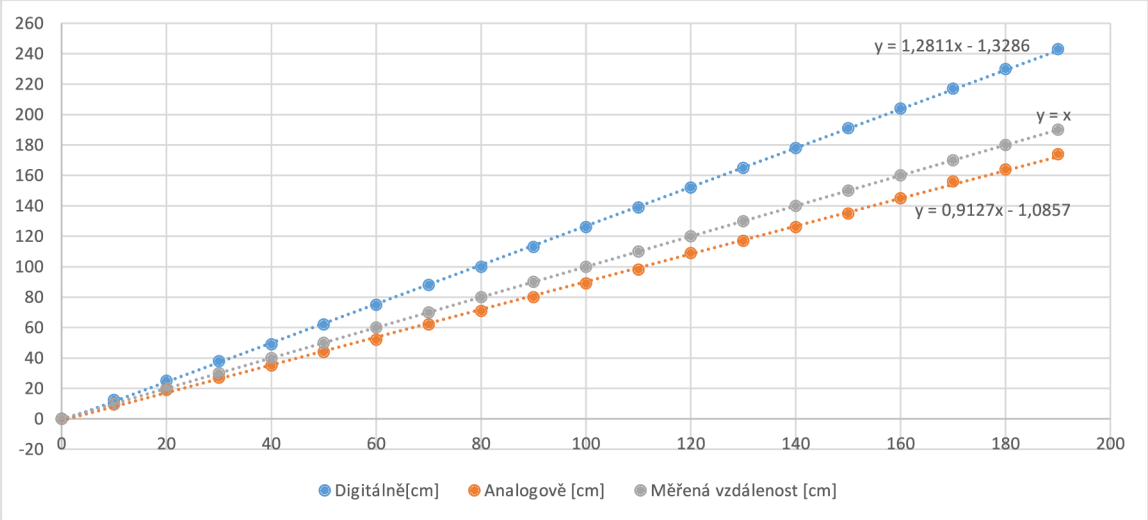
Tyto výsledky, zanesené i do grafu na obrázku 3.3, prokázaly, že chyba je lineárně závislá na měřené vzdálenosti, což je možné modelovat lineární rovnicí ve tvaru:

$$y = ax + b \tag{3.1}$$

Protože je z grafu zřejmé, že chyba čtených hodnot je lineární, byla pro získání koeficientů $a$ a $b$ zvolena metoda lineární regrese[4], tedy proložení přímek analogově a digitálně naměřenými body. S využitím tabulkového procesoru pak byly získány rovnice těchto přímek a tím i koeficienty $a$ a $b$. Výsledné koeficienty jsou uvedeny v tabulce 3.2.

Ze získaných rovnic je třeba vyjádřit inverzní funkce, pomocí kterých pak lze přepočítat změřenou hodnotu na správný výsledek. Pro analogové a digitální čtení pak platí vztahy 3.2, respektive 3.3, kde $x$ představuje hodnotu v centimetrech změřenou Arduinem. Po aplikaci

[4] http://en.wikipedia.org/wiki/Linear_regression




Obrázek 3.3: Graf znázorňující lineární chybu měření.

| | koeficient $a$ | koeficient $b$ |
|---|---|---|
| analogově | 0,9127 | 1,0857 |
| digitálně | 1,2811 | 1,3286 |

Tabulka 3.2: Výsledné koeficienty pro přepočet měřené vzdálenosti.

spočtených koeficientů je odchylka měření cca 2 cm od měřené vzdálenosti, což je oproti původním výsledkům velký posun.

$$y = \frac{x + 1,0857}{0,9127} \tag{3.2}$$

$$y = \frac{x + 1,3286}{1,2811} \tag{3.3}$$

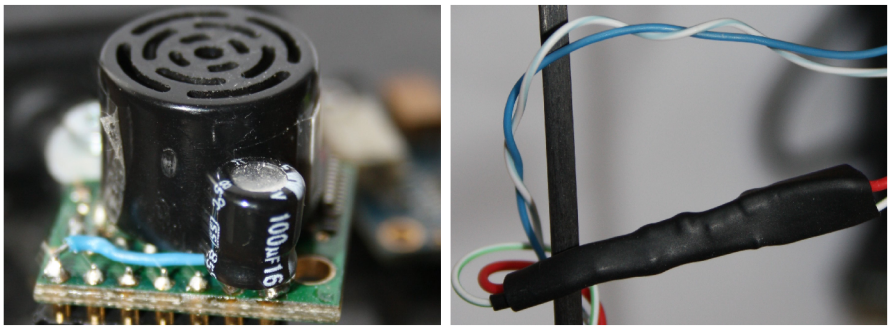
Pro příklad uvažujme, že digitálně čtený sonar naměřil vzdálenost 100 cm. Pro přepočet na správnou vzdálenost stačí dosadit hodnotu 100 do rovnice 3.3. Výsledek je po zaokrouhlení roven 79,1 cm, což odpovídá měřené vzdálenosti 80 cm.

## 3.5 RC filtr

Během provádění pokusů s instalovanými sonary byl zjištěn výskyt nepravidelných nestabilit v naměřených hodnotách. Problém se projevoval tak, že jednou za náhodný počet měření se u některého ze sonarů stalo, že vrátil maximální vzdálenost, kterou je sonar schopen měřit, tedy 765 cm, jak je popsáno v podkapitole 2.2.

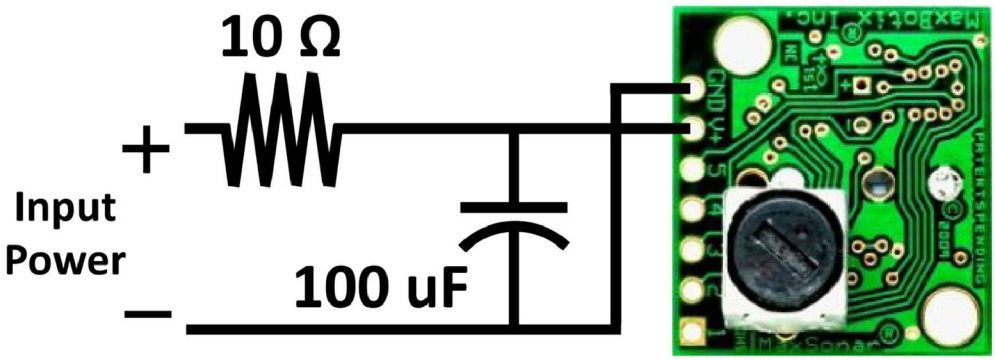
Pravděpodobnou příčinou těchto nestabilit se zdály být poklesy napětí způsobené zahájením měřícího cyklu skupiny pěti sonarů ve stejnou chvíli. Vzhledem k tomu, že každý sonar má ve špičce odběr 100 mA, náhlý pokles napětí mohl být natolik významný, že způsobil naprosté znehodnocení naměřené vzdálenosti. Pro odstranění poklesů napětí byl pro každý sonar sestaven filtr typu dolní propusť s využitím rezistoru a kondenzátoru. Výrobcem doporučená hodnota odporu rezistoru je 10 ohmu, kapacita kondenzátoru pak 100 $\mu$F [4]. Rezistory byly připájeny k vodiči připojenému ke zdroji napětí 5 V a izolovány

tenkou stahovací bužírkou. Kondenzátory vzhledem k větší velikosti nemohly být připájeny stejným způsobem na všechny sonary, protože držáky pro senzory po stranách kvadrokoptéry kvůli nedostatku místa nenabízí možnost připájet kondenzátor přímo na sonar. U takto umístěných sonarů proto byly umístěny mezi napájecí vodiče a schovány do stahovací bužírky společně s rezistorem (obr. 3.4).



Obrázek 3.4: Výsledek realizace RC filtru – připájený kondenzátor na napájecích pinech sonaru (vlevo) a kondenzátor schovaný ve smršťovací bužírce společně s rezistorem (vpravo).

Přesný způsob zapojení kondenzátorů a rezistorů je znázorněn na obrázku 3.5

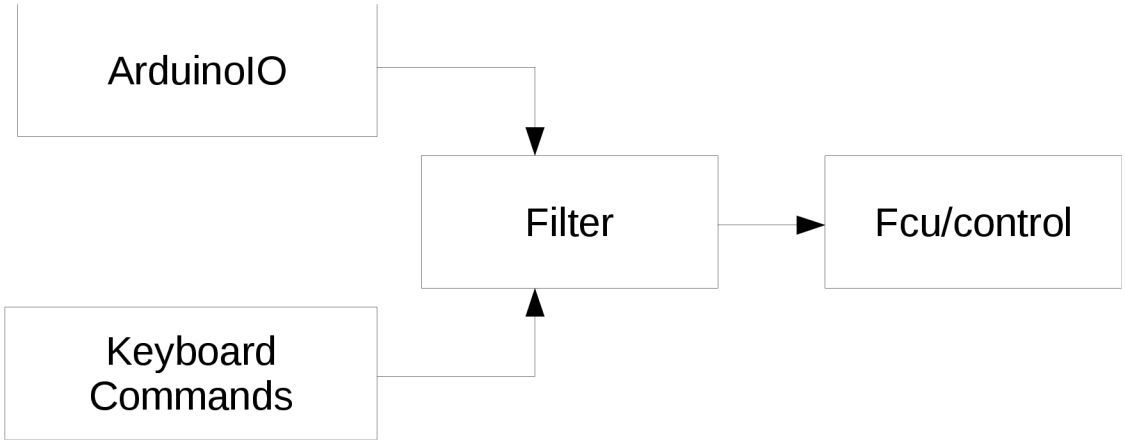



Obrázek 3.5: Schéma zapojení kondenzátoru a rezistoru k sonaru. Zdroj: [4]

## 3.6 Architektura ROS aplikace

Výsledná ROS aplikace by měla být složena z několika modulů – uzlů. Jeden uzel je potřeba pro čtení a zpracování dat z Arduina. Tato zpracovaná data jsou pak publikována na téma, ze kterého si je mohou odebírat jiné uzly. Jedna zpráva obsahuje číslo sonaru, a naměřenou hodnotu. Dále uzel pro detekci překážek ze zpracovaných dat, který zároveň představuje samotný filtr uživatelských příkazů. Zde je možné řešit i případné udržování konstantní vzdálenosti od zdi a podobně. Zároveň tento uzel konfrontuje s detekovanými překážkami i uživatelské příkazy. Protože s aktuální verzí firmwaru pro HLP procesor ale není možné v rámci ROSu odchytávat příkazy uživatele z dálkového ovládání, je nutné ovládat Tyru jiným způsobem – klávesnicí. K tomu je zapotřebí další uzel, který uživatelské příkazy publikuje na téma, ze kterého si je může zmíněný filtr přebírat. Ten je schopen určit,

zda uživateli hrozí kolize, a na základě toho provést patřičnou akci – varování bzučením, provedení úhybného manévru či ignorováním příkazu nebo omezením jeho vlivu na pohyb kvadrokoptéry. Způsob toku zpráv je znázorněn na obrázku 3.6.




Obrázek 3.6: Architektura uzlů v ROS frameworku.

Z obrázku je zřejmé, že uzel ArduinoIO, který čte data z Arduina, poskytuje zpracovaná data pro uzel Filter, který zároveň odebírá data od uzlu umožňujícím uživateli ovládat Tyru pomocí klávesnice. Na základě dat ze sonarů pak Filter rozhoduje, zda uživatelské příkazy přepošle na téma `Fcu/control` upravené nebo neupravené.

# Kapitola 4

# Realizace

Tato kapitola čtenáře seznámí s postupem výroby a instalací držáků pro skupinu rohových sonarů a výsledným zapojením všech instalovaných sonarů k Arduinu. Dále je představena jednoduchá aplikace pro otestování funkčnosti zapojených sonarů a popis implementace programu pro Arduino a ROS framework. Závěr kapitoly je věnován testování frekvence měření sonarů a reakční doby implementovaných uzlů.

## 4.1 Připevnění sonarů na Tyru

Držáky pro boční sonary byly vymodelovány a vytisknuty na 3D tiskárně skupinou studentů magisterského studia z předmětu Robotika – Tomášem Černíkem, Štěpánem Karáskem a Martou Čudovou[1]. K držákům ale ještě bylo třeba vyrobit další pro 8 sonarů, které mají být umístěny v rozích Tyry (viz podkapitola 3.2). Jako materiál byl zvolen hliník s tloušťkou 0,6 mm, pro jeho nízkou hmotnost, snadnou manipulaci a také relativně nízkou cenu. Zbývající 2 sonary (horní a spodní) nevyžadovaly žádný speciální držák, protože je bylo možné připevnit pouze pomocí distančních sloupků[2].

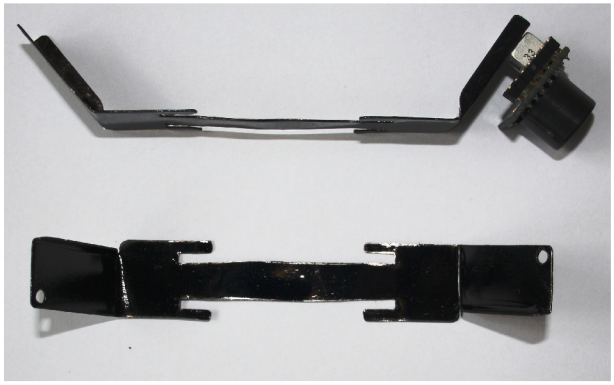
Držáky byly navrhovány tak, aby je bylo jednoduché na Tyru připevnit, aby nepřekážely při manipulaci s Tyrou, a také aby nehrozil kontakt držáků či vodičů připojených k sonarům s vrtulemi. Model držáku byl vytvořen v programu AutoCAD a je uložen na přiloženém CD.

Při výrobě byl nejprve vystřižen hrubý tvar těla držáku, který byl pak dotvarován pilníkem. Obzvláště tenké zobáčky na přichycení k rámu Tyry bylo potřeba vybrousit. Nakonec byly držáky nabarveny černým lakem. Konečná podoba výsledných držáků je zobrazena na obrázku 4.1. Vodiče vedoucí k připevněným sonarům jsou umístěny v svazkovací spirále, a stahovací páskou upevněny k rámu, aby nehrozilo jejich přeseknutí vrtulemi. K připevnění samotných sonarů k držákům byly využity distanční sloupky, pouze kratší než v případě horního sonaru, a plastové šrouby[3], aby bylo dosaženo co nejnižší hmotnosti.

## 4.2 Kabeláž

Pro napájení sonarů jsou využity piny Arduina, které je napájeno přes USB z Atomboardu, a je rozděleno do čtyř větví. Každá z těchto větví napájí 3 sonary. Zbylé dva sonary (horní

[1] http://merlin.fit.vutbr.cz/wiki/index.php/Beran-Cernik-Cudova-Karasek
[2] http://www.gme.cz/kda6m3x25-p623-063
[3] http://www.gme.cz/spc306-p662-025



Obrázek 4.1: Dvojice vyrobených držáků pro sonary umístěné v rozích Tyry.

a dolní) jsou připevněny příliš daleko od ostatních sonarů, ale relativně blízko k Arduinu, a proto jsou připojeny samostatně přímo k Arduinu. K analogově připojeným sonarům vede z Arduina jeden startovací vodič a jeden datový vodič. K digitálně připojeným sonarům také vede jeden datový vodič z Arduina, ale startovací vodič je potřeba jen pro první ze skupiny zřetězených sonarů. Další sonar je pak vždy odstartován předchozím sonarem po dokončení jeho měřícího cyklu.

## 4.3 Aplikace pro ověření funkčnosti sonarů

Protože již při využití 6 sonarů není jednoduché z terminálových výpisů určit, zda všechny sonary pracují správně nebo ne, byla vytvořena jednoduchá testovací aplikace s grafickým uživatelským rozhraním, která přehledně zobrazuje data zasílaná Arduinem (obr. 4.2). Pro vytvoření aplikace bylo využito open-source frameworku Qt[4]. Aplikace po spuštění začne čte data ze sériového portu Arduina (/dev/ttyAMC0). Pokud Arduino v době spuštění aplikace není připojeno, aplikace v pravidelných intervalech opakuje pokus o připojení. Není proto třeba brát ohled na to, zda je v době spouštění této aplikace již Arduino připojeno k počítači nebo ne.

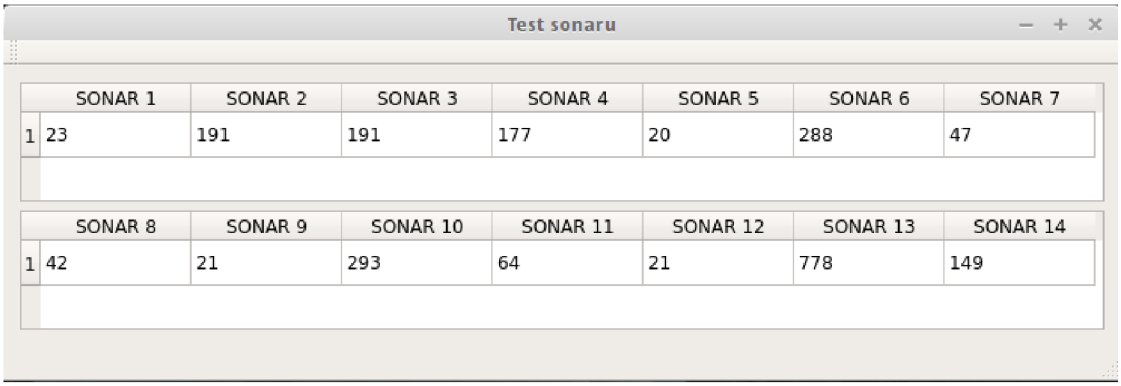
Nabízí se několik možností, jak aplikaci dále vylepšit. Například při spuštění by mohly být pole, kde se zobrazují hodnoty měření, vybarvené červeně. Ke změně vybarvení na zelenou by pak došlo až při významné změně hodnoty. Pozorováním bylo totiž zjištěno, že při problému s kontaktem Arduino přečte předem nedefinovanou hodnotu, která se při opakovaném čtení ale již příliš neliší. Dále by pak aplikace mohla být rozšířena o vizualizaci prostoru a překážek okolo Tyry.

## 4.4 Implementace

V rámci této práce byl v jazyce C++ implementován program pro Arduino, který čte naměřená data z připojených sonarů a odesílá je sériovou linkou. Zároveň obsahuje PID regulátor, který je schopný na základě naměřených dat spodního sonaru počítat výkon motorů potřebný pro přistání.

Pro ROS framework pak byly implementovány v jazyce C/C++ uzly pro čtení dat z Arduina a uzel pro detekci a uhýbání překážkám. Uzel pro čtení dat z Arduina byl implemen-

[4] https://www.qt.io/qt-framework/




Obrázek 4.2: Aplikace pro ověření funkčnosti zapojených sonarů. Výsledky měření jsou zobrazovány do přehledné tabulky.

tován ve spolupráci se skupinou studentů z magisterského studia z předmětu Robotika[5]. Uzel, který umožňuje ovládat Tyru z klávesnice, již byl hotový. Jeho autorem je Adam Crha[6]. Program však bylo třeba upravit a přesměrovat zprávy tak, aby byly publikovány na téma, ze kterého si je může odebírat uzel pro detekci překážek. Zprávy s uživatelskými příkazy z klávesnice jsou posílány na téma `keyboard_commands`.

### Program pro Arduino

Programy pro mikrokontroléry Arduino se skládají ze dvou základních funkcí – `void setup()`[7] a `void loop()`[8]. Funkce `void setup()` se zavolá pouze jednou na začátku programu a slouží pro inicializaci proměnných, nastavení vstupních a výstupních portů nebo konfiguraci sériového rozhraní (například baud rate) a tak dále. Po jejím skončení je volána hlavní funkce `Void loop()`. Při vývoji jsem zjistil, že slabinou Arduina jsou poněkud omezené možnosti přerušení – je dostupné pouze na 2 pinech. Vzhledem k počtu 14 sonarů jsem se tedy musel obejít bez přerušení, které by umožnilo elegantnější zápis programu, a sonary postupně číst v předem daném pořadí.

Aby bylo dosaženo vyšší frekvence, je využito toho, že sonary po naměření udržují hodnotu napětí na analogovém výstupu až do doby, než dojde k dalšímu měření. Nezáleží tedy, v jakém momentu je výsledek přečten. V případě digitálních sonarů je problém v tom, že po naměření vzdálenosti odešlou pulz dané šířky a dále již nic neposílají až do doby, než znovu naměří další výsledek. Pokud by tedy Arduino nečekalo na pulz ještě před jeho příchodem, mohlo by se stát, že pulz již nezastihne, a tím pádem nepřečte ani žádný výsledek. V důsledku by pak Arduino nezastihlo ani další pulzy následujících sonarů. Experimentálně bylo ověřeno, že v době mezi výsledky digitálních sonarů je možné bezpečně přečíst hodnoty ze dvou analogových sonarů, aniž by docházelo k chybám v důsledku nezastižení pulzu. Experimenty byly prováděny tak, že byly před sonary v krátké vzdálenosti umístěny překážky. Doba měřícího cyklu totiž záleží na aktuální vzdálenosti překážky. Čím blíže překážka je, tím kratší dobu měření trvá.

Na Arduinu je kromě ovládání a čtení sonarů ještě implementován PID regulátor. Pro

[5] http://merlin.fit.vutbr.cz/wiki/index.php/Beran-Cernik-Cudova-Karasek
[6] http://merlin.fit.vutbr.cz/wiki/index.php/Beran-Crha
[7] http://www.arduino.cc/en/Reference/setup
[8] http://www.arduino.cc/en/Reference/loop

implementaci byla využita volně šiřitelná knihovna[9], nabízející základní funkce pro PID algoritmus. Regulátor je implementován tak, že v průběžně počítá potřebný výkon motorů pro dosažení výšky 25 cm. Z této výšky by již mělo být pro uživatele bezpečné přistát. Protože PID regulátor bere v úvahu i historii, je potřeba v momentě, kdy si uživatel přeje zahájit přistávání, tuto historii smazat. Program na Arduinu s tímto počítá a historii PID regulátoru smaže po přijetí znaku „R“ na sériovém rozhraní. Vypočtená data PID regulátorem jsou zasílána ve stejném tvaru jako data ze sonarů. Identifikována jsou číslem 15, tedy číslem o jedničku vyšším než má poslední sonar. Hodnoty vypočtené regulátorem jsou v intervalu 0–255, kde 255 je maximální výkon. Tyto hodnoty by pro využití ROS frameworkem bylo třeba přepočítat do intervalu 0–1, protože s tímto intervalem pracuje Mav Framework. V ROSu již nicméně nebyly implementovány uzly, který by s PID regulátorem pracovaly. Regulátor tak slouží jako základ pro případnou budoucí implementaci funkce automatického přistávání.

### Uzel pro zpracování zpráv z Arduina

Uzel který zpracovává zprávy z Arduina byl implementován ve spolupráci se studenty předmětu Robotika. Funguje tak, že v cyklu načítá data ze sériového rozhraní, která pak znak po znaku prochází. Zpráva je Arduinem zasílána ve tvaru zmíněném předchozí kapitole. Protože v momentě, kdy uzel přečte ze sériové linky přijaté znaky, nemusí být poslední z přijatých zpráv kompletní, je třeba, aby funkce, která zpracování zpráv provádí, vrátila část zprávy, která nebyla přijata celá. K této části se v dalším cyklu připojí zbylá část zprávy, která pak může být zpracována standardním způsobem. Zprávy jsou publikovány na téma `sonar_data`.

### Uzel pro detekci a vyhýbání překážkám

Tento uzel, na obrázku 3.6 znázorněn jako Filter, je přihlášen k odběru dat z témat `sonar_data` a `keyboard_commands`. Data ze sonarů si ukládá do dvojrozměrného pole, přičemž zároveň vypočítává koeficient maximální povolené rychlosti pro každý směr. Vzhledem k tomu, že uzel pro ovládání z klávesnice umožňuje pohyb pouze konstantní rychlostí, je maximální povolená rychlost v každém směru vypočítávána a případně limitována lineárně. Kontrolovány jsou směry doprava, doleva, dopředu a dozadu. Aby bylo možné kontrolovat všechny směry z dat všech 14 sonarů, je potřeba navrhnout efektivnější metodu jak tato data zpracovat a vyvodit potřebný směr pohybu, například operacemi s vektory. Aktuálně implementovaný způsob je totiž výpočetně příliš náročný pro zpracování dat ze všech sonarů v reálném čase.

Uzel zároveň kontroluje minimální vzdálenost od překážek. Po přiblížení na méně než 30 cm provede pohyb směrem od překážky. Aby však uzel nezasahoval do řízení v případech, kdy dojde jen k chybě měření některého ze sonarů, dojde k pohybu pouze pokud bude naměřena vzdálenost kratší než 30 cm dvakrát po sobě. Hodnota 30 cm byla stanovena na základě experimentů, kdy Tyra byla schopná reagovat s dostatečnou rezervou.

## 4.5 Testování rychlosti odezvy sonarů

Při testování jsem se zaměřil na frekvenci měření sonarů. Dále také na dobu, za kterou je Tyra schopná zareagovat na překážku umístěnou před některý z bočních sonarů a jak

[9] http://playground.arduino.cc/Code/PIDLibrary

rychle zaregistruje odstranění této překážky.

Před vlastním testováním a létáním je třeba zajistit, aby bylo dálkové ovládání správně nastaveno a na Tyře spustit potřebné ROS uzly. Následující postup by měl zajistit, aby vše fungovalo správně.

1. Zapojit baterii a počkat na nabootování do systému.
2. Zapnout ovládání.
3. Přepínač B na dálkovém ovládání přepnout do spodní polohy (blíže k sobě). Tím je vybrán procesor vyšší úrovně HLP.
4. Zapnout AscTec Autopilot.
5. Spustit následující uzly:
   - `roscore` – Master, zajišťuje komunikaci a správnou funkci ostatních uzlů (viz kapitola 2.4),
   - `fcu.launch` – AscTec Mav Framework. Zajišťuje vytvoření témat pro ovládání Tyry (fcu/control). Často dochází k problému se spuštěním, pokud skončí chybou „rx timeout“, je třeba se ujistit že je přepínač B na ovládání opravdu ve správné pozici. Dálkovým ovládáním by nemělo být možné roztočit motory. Pokud problém přetrvá, může pomoci změnit pozici USB kabelu do druhého USB konektoru. Před změnou je však vhodné nejprve vypnout AscTec Autopilot,
   - `arduinoIO` – Uzel pro zpracování zpráv z Arduina. Pokud po spuštění skončí chybou, je třeba se ujistit, že není spuštěna grafická aplikace pro testování sonarů,
   - `kb_sonar_control` – Upravená verze uzlu `kb_control`,
   - `kb_sonar_flight_control` – Uzel pro detekci překážek zamezující vzniku kolizí.

Po úspěšném nastavení dálkového ovládání a spuštění potřebných uzlů lze začít s experimenty a měřením. Nejprve jsem zjišťoval frekvenci čtení analogových a následně i digitálních. K tomu účelu jsem vytvořil soubor, do kterého byly uloženy výsledky všech měření Arduina za dobu 30 sekund. Podle množství výsledků z analogově a digitálně čtených sonarů pak bylo možné spočítat, kolikrát jsou sonary schopny měřit za sekundu. Výsledky experimentu jsou zaneseny do tabulky 4.1.

| | Počet měření | Frekvence [Hz] |
|---|---|---|
| Analogově | 138 | 4,6 |
| Digitálně | 35 | 1,2 |

Tabulka 4.1: Výsledky počtů naměřených výsledků analogově a digitálně čtených sonarů za časový úsek 30 sekund a vypočtená frekvence.

Z tabulky 4.1 je zřejmé, že z digitálně čtených sonarů jsou výsledky získávány mnohem vyšší frekvencí. To je způsobené tím, že digitálně čtené sonary jsou spouštěny postupně jeden po druhém, zatímco analogově čtené sonary jsou odstartovány současně.

Dále jsem zjišťoval, jak rychle je uzel pro detekci překážek schopen detekovat a reagovat na objekt umístěný do bezprostřední vzdálenosti před boční sonar, čtený analogově. V tomto případě tedy hraje roli kromě frekvence měření i rychlost zasílání a zpracování zpráv

| Doba detekce překážky [ms] | Doba detekce odstranění překážky [ms] |
|---|---|
| 825 | 627 |
| 859 | 715 |
| 977 | 608 |
| 817 | 673 |
| 971 | 779 |

Tabulka 4.2: Výsledky měření reakční doby na překážku.

z Arduina. Společně s tím byla měřena i doba, za kterou uzel rozpoznal, že byl objekt odstraněn. Výsledky jsou zaneseny do tabulky 4.2.

Z naměřených hodnot vychází průměrná doba reakce na překážku 890 ms. Doba, za kterou uzel zjistil, že překážka již byla odstraněna byla v průměru zhruba o 200 ms kratší, z naměřených hodnot vychází na 680 ms. Důvodem, proč je doba detekce odstranění překážky kratší, je že uzel na překážku reaguje až v případě, že dvě měření po sobě překročí určitou hranici. Čeká tedy na jakési potvrzení, aby nedocházelo k zásahům do řízení v případech, kdy došlo pouze k chybě měření některého ze sonarů.

# Kapitola 5

# Závěr

Cílem této práce bylo osadit kvadrokoptéru Tyra sadou 14 sonarů, navrhnout způsob jejich rozmístění, zapojení a implementovat balíčky pro ROS framework, které mají společně se sonary tvořit protinárazový systém. Samotnému návrhu předcházelo studium existujících projektů, zaměřených na podobnou problematiku. Dále studium technologií potřebných pro vývoj měřícího zařízení, tedy mikrokontroléru Arduino Micro a samotných ultrazvukových senzorů, a studium ROS frameworku pro vývoj robotického software. Následně byla provedena analýza zadání, kde bylo stanoveno, na které parametry by měl být při návrhu kladen důraz – prostor pokrytý sonary, co nejvyšší frekvence měření a nízká hmotnost.

Návrh rozmístění se postupně vyvíjel, než byl nalezen kompromis mezi pokrytím prostoru a obtížností připevnění sonarů na vybraná místa. K výslednému návrhu byl vytvořen model s vizualizací rozsahů signálů sonarů pro ověření dostatečného pokrytí prostoru. Držáky pro sonary byly vyrobeny na základě modelu vytvořeném v programu AutoCAD. Zapojení sonarů bylo navrženo s ohledem na omezený počet pinů Arduina, část sonarů je proto zapojena zřetězeně, i když to znamená nižší frekvenci měření. Zřetězeně jsou proto zapojeny ty sonary, které jsou méně kritické – rohové a sonar umístěný v horní části.

Pro ROS framework byla implementována sada nástrojů pro čtení a zpracování dat z Arduina, detekci objektů a asistenci při řízení Tyry. Na základě získaných a zpracovaných dat je pak Tyra schopna udržovat minimální odstup od překážek nebo zpomalit pohyb proti překážce v závislosti na vzdálenosti od ní.

Během testování byl kladen důraz na frekvenci měření sonarů a rychlost reakce protinárazového systému. Z naměřených výsledků byla vypočítána frekvence analogově čtených sonarů na 4,6 Hz a frekvence digitálních sonarů 1,2 Hz. Dále byla měřena i doba reakce na umístění a odebrání překážky bezprostředně před jeden z bočních sonarů. Zde výsledky měření přinesly hodnoty 680 ms, respektive 890 ms.

Výsledkem práce je tedy funkční prototyp měřícího zařízení, které je díky Arduinu možné použít bez ohledu na platformu, a protinárazový systém pro ROS framework určen pro systém Ubuntu ve verzi 12.04. Protinárazový systém je schopen rychle reagovat na překážky po bocích kvadrokoptéry. Systém by bylo vhodné dále vylepšit tak, aby byl výpočet zpracování dat efektivnější, a aby bylo možné reagovat na překážky detekované všemi instalovanými sonary. Toho by mohlo být dosaženo například využitím OpenCV knihovny a efektivních operací s vektory, nebo se vydat jiným směrem a implementovat protinárazový systém fuzzy regulátorem. Výhoda tohoto řešení by mohla spočívat v jednodušší konfiguraci pro jiné typy kvadrokoptér s rozdílnou hmotností, velikostí a jiným výkonem motorů.

# Literatura


[1] Aström, K. J.; Hägglund, T.: *PID Controllers: Theory, Design, and Tuning*. Research Triangle Park, N.C.: International Society for Measurement and Control, druhé vydání, 1995, ISBN 1-55617-516-7.

[2] Bonar, T.: *Using Multiple MaxSonar®Sensors*. MaxBotix, Listopad 2012. URL http://www.maxbotix.com/articles/031.htm

[3] Gageik, N.; Müller, T.; Montenegro, S.: Obstacle detection and collision avoidance using ultrasonic distance sensors for autonomous quadcopter. Technická zpráva, University of Würzburg, Aerospace Information Technology(Germany), Září 2012, [Online; Citováno 03-04-2015]. Dostupné z http://www8.informatik.uni-wuerzburg.de/fileadmin/10030800/user_upload/quadcopter/Paper/Gageik_Mueller_Montenegro_2012_OBSTACLE_DETECTION_AND_COLLISION_AVOIDANCE_USING_ULTRASONIC_DISTANCE_SENSORS_FOR_AN_AUTONOMOUS_QUADROCOPTER.pdf.

[4] Gross, B.: *MaxSonar Operation on a Multi-Copter*. MaxBotix, Únor 2013. URL http://www.maxbotix.com/articles/067.htm

[5] Holz, D.; Nieuwenhuisen, M.; Droeschel, D.; aj.: Towards multimodal omnidirectional obstacle detection for autonomous unmanned aerial vehicles. Technická zpráva, Institute for Computer Science VI, University of Bonn, Září 2013. URL http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-1-W2/201/2013/isprsarchives-XL-1-W2-201-2013.pdf

[6] Kuphaldt, T. R.: *Lessons In Electric Circuits*. Independently published, druhé vydání, July 2007. URL http://www.ibiblio.org/kuphaldt/electricCircuits/AC/index.html

[7] Maxbotix: *XL-MaxSonar®- EZ$^{TM}$Series*. High Performance Sonar Range Finder. URL http://www.maxbotix.com/documents/XL-MaxSonar-EZ_Datasheet.pdf

[8] O'Kane, J. M.: *A Gentle Introduction to ROS*. Independently published, Říjen 2013, ISBN 978-1492143239, dostupné z http://www.cse.sc.edu/~jokane/agitr/.

[9] R. Siegwart and D. Scaramuzza: Range sensors. http://www1.cs.columbia.edu/~allen/F14/NOTES/rangesensing.pdf.

[10] Schlegel, M.: Průmyslové PID regulátory: teorie pro praxi. Technická zpráva, Fakulta aplikovaných věd ZČU v Plzni, katedra kybernetiky, ZAT a.s. Příbram, 2001, dostupné z http://zcu.arcao.com/kky/zky/Prago1.pdf.

[11] WWW stránky: PID for Dummies.
http://www.csimn.com/CSI_pages/PIDforDummies.html.

# Příloha A

# Obsah CD

- Zdrojový kód k Arduinu Micro spolu s PID knihovnou ve složce /src/Arduino
- Zdrojové kódy k ROS uzlům ve složce /src/tyra_sonar/src
- Definice zprávy pro zpracovaná data z Arduina ve složce /src/tyra_sonar/msg
- Aplikace pro test funkcionality sonarů ve složce /src/test_sonaru
- Data ze sonarů pro testovací účely ve složce /src/rosbag
- Model držáků pro sonary a model Tyry s rozmístěnými sonary ve složce /modely
- Tato práce ve formátu pdf s názvem Technická zpráva ve složce /tz
- Zdrojové kódy této technické zprávy ve složce /tz/src
- Demonstrační video ve složce /video
- Plakát ve složce /plakat.

# Příloha B

# Funkčnost pinů sonaru MB1220

- Pin 1 – BW: pro sériový výstup na 5. pinu nechat nepřipojen nebo nastavit na logickou 1. Pokud je nastaven na logickou 0, pin 5 po dokončení měřícího cyklu posílá pulz pro aktivaci následujícího sonaru
- Pin 2 – PW: na tomto pinu sonar po ukončení měřícího cyklu vysílá pulz o šířce reprezentující naměřenou vzdálenost. Vzdálenosti 1 cm odpovídá pulz o šířce 58 uS[1]
- Pin 3 – AN: na tomto pinu sonar nastavuje napětí odpovídající naposledy naměřené vzdálenosti. Slouží pro analogové čtení. Je-li sonar připojen na 5 V zdroj, pak 1 cm odpovídá 4.9 mV, pokud je připojen na 3.3 V, pak 2 cm odpovídá 3.2 mV
- Pin 4 – PX: je-li pin nastaven na logickou 1 po dobu alespoň 20 uS, sonar spustí měřící cyklus. Pokud pin není připojen, pak sonar neustále opakuje měření
- Pin 5 – RX: Je-li nastaven pin BW na logickou 1 nebo nepřipojen, pak tento pin asynchronně vysílá sériová data ve formátu daným standardem RS232[2]. Výstup je zasílán v podobě jednoho znaku "R"následovaným třemi číslicemi (vzdálenost v centimetrech) a znakem konce řádku. Je-li pin BW držen na logické 0, pak 5. pin vysílá pulz pro aktivaci následujícího sonaru
- Pin 6 – +5: přivádí napětí 3.3 nebo 5 V. Maximální proud při 5 V je 100 mA. Tento proud je odebírán v momentě, kdy sonar generuje akustickou vlnu.
- Pin 7 – GND: uzemnění.

[1] http://www.maxbotix.com/articles/033.htm
[2] http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html