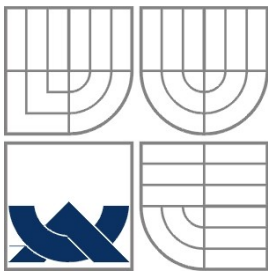


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

NÁVRH ŘÍDÍCÍHO MODULU UAV ROBOTU

THE DESIGN OF CONTROL UNIT FOR UAV ROBOT

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE **BC. PETR ARNOŠT**
AUTHOR

VEDOUCÍ PRÁCE **ING. STANISLAV VĚCHET, PH.D.**
SUPERVISOR

BRNO 2012

ZADÁNÍ ZÁVĚREČNÉ PRÁCE

(na místo tohoto listu vložte originál a nebo kopii zadání Vaš práce)

ABSTRAKT

Tato diplomová práce je zaměřena na návrh řídícího softwaru pro bezpilotní letadlo. Jako zástupce bezpilotních letadel byla zvolena kvadrikoptéra Parrot Ar.Drone. Práce popisuje způsob ovládání a komunikace. Na základě těchto informací je vytvořen program pro řízení letu kvadrikoptéry Ar.Drone.

ABSTRACT

This diploma thesis is focused on the design of the control software for the unmanned aerial vehicle. Parrot Ar.Drone quadrocopter was a representative of the unmanned aerial vehicle. This thesis describes the way of control and communication with the unmanned aerial vehicle. Based of this information the flight control software for Ar.Drone quadrocopter is created.

KLÍČOVÁ SLOVA

UAV, Parrot, Ar.Drone, UDP, AT příkazy, navdata, csharp, c#.

KEYWORDS

UAV, Parrot, Ar.Drone, UDP, AT commands, navdata, csharp, c#.

PODĚKOVÁNÍ

Touto cestou bych rád poděkoval mému vedoucímu Ing. Stanislavu Věchetovi Ph.D. za věnovaný čas, připomínky a cenné rady při tvorbě této práce.

Také bych chtěl poděkovat rodičům za umožnění studia a podporu při dokončování této práce.

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem tuto práci vypracoval samostatně pod vedením Ing. Stanislava Věcheta Ph.D. a použil jsem literaturu uvedenou v bibliografii

V Brně, 23.5.2012

….....

Bc. Petr Arnošt

BIBLIOGRAFICKÁ CITACE

ARNOŠT, P. Návrh řídicího modulu UAV robotu.. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012. 61 s. Vedoucí diplomové práce Ing. Stanislav Věchet, Ph.D..

Obsah:

1 ÚVOD

UAV je zkratka anglického *unmanned aerial vehicle*, což se dá do českého jazyka přeložit jako bezpilotní letadlo. Již podle názvu je patrné, že se jedná o létající stroje, které na své palubě nemají lidskou posádku. Nejčastěji se s UAV můžeme setkat ve vojenské oblasti, kde jsou pro posádku největší rizika. V dnešní době, díky výkonným počítačům a stálému vylepšování dalších používaných komponent, dokáže UAV nahradit mnoho činností v této oblasti. Jsou veřejně dostupné informace o bezpilotních letadlech, která dokáží i zaútočit. Určitě v této oblasti existuje i několik tajných projektů, které budou využívat daleko vyspělejší technologie. Na druhou stranu existují i UAV, které jsou běžně dostupné širší veřejnosti a slouží pro větší pohodlí života, nebo pro zábavu. Touto problematikou se zabývá kapitola 2.

Hlavní zaměření této diplomové práce je na konkrétní model UAV, kterým je výrobek od firmy Parrot s názvem Ar.Drone. Jedná se o kvadrikoptéru, kterou je možné ovládat pomocí mobilního telefonu. Kvadrikoptéra je velmi zajímavý stroj, který by se dal způsobem létání vzdáleně přirovnat k vrtulníku. Na rozdíl od něj má ale 4 vrtule umístěné v jedné rovině. Jednotlivých směrů letu je dosahováno specifickým způsobem. Nejenom způsob létání, ale i jednotlivé komponenty Ar.Drone, jsou podrobně popsány ve 3. kapitole této práce.

Protože cílem práce je návrh řídicího systému tuto kvadrikoptéru, je nutné ujasnit si jakým způsobem probíhá komunikace mezi ovládacím zařízením a kvadrikoptérou Ar.Drone. O záležitostech týkajících se této problematiky pojednává kapitola 4. Kapitola také obsahuje stručný popis programu WireShark.

Kapitola 5. se zabývá podrobným popisem a vysvětlením formátů dat, které je nutné posílat z ovládacího zařízení do Ar.Drone. Při striktním dodržení pravidel popsaných v této kapitole bude Ar.Drone létat podle požadavků.

V kapitole 6 je podrobně popsaná opačná komunikace, a to z Ar.Drone do ovládacího zařízení. Čtenář zde má možnost jednoduchým a jasným způsobem zjistit, jak získat navigační data z Ar.Drone a jak s nimi naložit, aby se dozvěděl požadované aktuální letové informace z kvadrikoptéry.

Hlavním cílem práce byl návrh řídicího systému pro Ar.Drone. Tento program využívá informací popsaných v předchozích kapitolách a umožňuje ovládání kvadrikoptéry Ar.Drone z počítače.

2 UAV

UAV je zkratka z anglického Unmanned Aerial Vehicle, česky známé jako bezpilotní letadlo. Toto zařízení může být pilotem ovládané ze země, může létat automaticky na základě předem stanovené trasy, nebo zcela samostatně díky autonomnímu řídicímu systému. Při ovládání ze země sedí pilot u řídícího pultu podobného obyčejnému letadlu. Letadlo ovládá zejména na základě obrazu na monitoru a letových údajů posílaných z letadla. Létání podle předem stanovené trasy se v praxi často nepoužívá. Když už je použito, tak většinou v kombinaci buď s pilotem na zemi, nebo autonomním systémem. Tento způsob ovládání totiž není schopný zareagovat a nečekanou změnu okolní situace. Autonomní řídicí systém dokáže letadlo řídit zcela sám bez zásahu obsluhy a dokáže řešit i případné nečekané situace.

Oficiálně byl termín Unmanned Aerial Vehicle změněn na Unmanned Aircraft System (UAS), což lépe vystihuje skutečnost, že tyto systémy obsahují kromě vlastních letadel i různé pozemní stanice a další prvky. Pojem UAS se nicméně tolik nerozšířil, takže je nadále ve většině případů používán termín UAV.

I když se pojem UAV objevuje čím dál častěji ve spojení s civilní oblastí, největší zastoupení a vývoj je stále v oblasti vojenské.

Stupně autonomie:

Některé UAV jsou nazývány Drone, protože jsou složitější, než jednoduché dálkově řízené RC modely, ale stále je kompletně ovládá pilot.

Sofistikovanější modely mohou mít vestavěné kontroly a navigační systémy sledující rychlost, dráhu letu a další údaje, ze kterých je řízena například stabilita stroje, nebo jednoduché navigační funkce.

Plně autonomní UAV dokáže létat sám bez zásahu lidského faktoru. Dokáže řešit nečekané situace. [1]

Bezpilotní letadla můžeme rozdělit do následujících kategorií, podle oboru v jakém se vyskytují, nebo plní úkoly. U každé kapitoly je uveden příklad UAV s podrobnějším popisem.

První tři kategorie se týkají zejména vojenského průmyslu. UAV se v této kategorii používají kvůli několika důvodům. Nejvýznamnějším z nich je určitě lidský život. Pilot zůstává v bezpečí základny a při jakékoliv nehodě je zničen pouze stroj, který se dá podle plánů vždy zkonstruovat znovu. Pilotovy zkušenosti jsou také jedním z důvodu proč pilota chránit. S vývojem technologií se objevuje ještě další důvod. Je jím manévrovatelnost letadel vůči střelám ve vzduchu. V dnešní době existují střely, které zvládnou daleko větší přetížení, než je schopný zvládnout člověk. A pokud je stroj bez lidské posádky, jeho odolnost vůči přetížení se tím zvyšuje. Technologie umožňují vyrobit letadlo, které vydrží přetížení i přes 30 G. Kdežto člověk, který vydrží maximálně 9 G by proti střele schopné manévrovat s přetížením až 100 G, neměl žádnou šanci. [2]

2.1 Cíle a návnady

Do této části spadají letadla, jejichž hlavním úkolem je vydávat se za skutečná letadla. Akorát nemají pilota a jsou řízeny dálkově. Jsou používány jednak v bojových podmínkách a také pro výcvik obrany.

Jako zástupce v této kategorii byl vybrán model *BQM-34* (viz. Obr. 1). Tento stroj je používaný pro trénink obranné pohotovosti a vývoj zbraní. Může letět ve výšce od 3 m do 18 km. [3]

Obr. 1: UAV BQM-34 [4]

2.2 Bojové

Základní vlastností UAV v této kategorii je, že jsou vybaveny zbraněmi. Většinou se jedná o rakety. Bojové UAV jsou daleko menší, než stíhačky vybavené raketami a tím pádem hůř zpozorovatelné. Nepřítel tedy nemusí o útoku do poslední chvíle vědět.

Příkladem v této oblasti je UAV *MQ-9 Reaper* (viz. Obr 2). Tento letoun může být vybaven až 14-ti raketami. Některé z těchto raket můžou být laserově řízeny a tím pádem jsou vysoce přesné. [5]

Obr. 2: MQ-9 Reaper [6]

2.3 Průzkumné

V této kategorii je kladen důraz na to, aby UAV bylo vybaveno kamerami, radary a dalšími snímacími zařízeními. Důležité také je, aby letadlo vydrželo co nejdéle v provozu. Některé modely se můžou přiblížit k době až 40-ti hodin ve vzduchu.

Asi nejznámějším v této kategorii je *RQ-4 Global Hawk* (viz. Obr. 3). Vybaven

je elektrooptickými senzory a infračervenými senzory. Dokáže za den zmapovat prostor až 4 km^2, nebo rozpoznat pohybující se objekty rychleji, než 7,5 km/h. Global Hawk je při startu a přistávání ovládán ze základny. Cena tohoto UAV se pohybuje okolo $ 123 000 000. [7]

Obr. 3: RQ-4 Global Hawk [8]

2.4 Ostatní

Tato poslední kategorie by se ještě dala dělit na další podkategorie UAV, například podle vykonávané činnosti, nebo oboru působnosti. Bylo by možné sem zařadit UAV, které jsou zatím pouze ve vývoji, ale využívají zajímavé technologie. Jeden z nich by mohl být *Pathfinder Plus* (viz. obr. 4) vyvinutý v NASA. Jedná se o jedno křídlo, které má 8 vrtulí. Zajímavý je zdroj energie. Toto křídlo je celé po své vrchní straně pokryto solárními články, které napájí motory vrtulí. Rozpětí tohoto křídla je okolo 30 m.

Obr. 4: Pathfinder Plus [9]

Do této kategorie by se daly zařadit i civilní UAV, které můžou sloužit například pro sledování dopravní situace, měření metrologických hodnot ve vyšších polohách, monitorovat situace na hromadných akcích (koncerty, demonstrační akce a podobně). Také zde můžeme zařadit různé dálkově ovládané RC modely. Do této kategorie se řadí i kvadrikoptéra Parrot Ar.Drone (obr. 5), o které pojednává tato práce. [1] [5]

Obr. 5: Ar.Drone od výrobce Parrot [10]

3 AR.DRONE

Parrot Ar.Drone je kvadrikoptéra ovládaná z mobilních zařízení Apple (iPhone, iPad, iPod) a nově i ze zařízení s operačním systémem Android, Bada a Symbian. [11]

Kvadrikoptéra je létající stroj, který má čtyři vrtule v jedné rovině a jejich nezávislým otáčením se dokáže pohybovat. Lze najít různá označení těchto strojů. Např.: *kvadrokoptéra*, *quadrotor*, *quodrocopter*, *„čtyřtulka“*. V této práci bude používáno označení *kvadrikoptéra*, jelikož výrobce Parrot takto nazývá Ar.Drone na svých oficiálních webových stránkách www.parrot-ardrone.cz.

Firma Parrot se zabývá zejména Bluetooth bezdrátovými handsfree sadami do aut a mimo ně. [12]

Po prozkoumání dostupných možností byla kvadrikoptéra Ar.Drone pro tuto práci zvolena z několika hledisek. Jeden z důvodů pro výběr byla cena, která v době začátku prací na této práci byla pod 10 000 Kč. Další důvod s cenou úzce spojený byla snadná dostupnost na českém trhu. Neméně důležitým důvodem pro výběr byla také okamžitá použitelnost modelu, bez nutnosti sestavování modelu apod.

3.1 Obsah balení

Obr. 6: Obsah balení Parrot Ar.Drone

Balení kvadrikoptéry Parrot Ar.Drone obsahuje hlavně samotnou kvadrikoptéru, která je okamžitě připravená k použití (obr. 6).

Dále jsou v krabici dva kryty z tvrzeného polystyrenu (obr. 7). Větší z krytů při nasazení překryje konce rotujících vrtulí a zabraňuje tak jejich kontaktu s překážkami. Tento kryt je určen hlavně pro létání ve vnitřních prostorách, kde je překážek mnoho. Ochraňuje jednak samotné vrtule před poškozením při kontaktu s cizím tělesem a také překážky (nábytek, stěny atd.) proti případnému poškození nebo poškrábání od rotujících vrtulí. Druhý, menší, z krytů je určený zejména pro létání venku. Jednak zde není tolik překážek, které by mohli přijít do kontaktu s vrtulemi, ale také tím, že překrývá pouze tělo kvadrikoptéry, je létání snadnější při mírném větru.

Obr. 7: Použití krytu pro venkovní a vnitřní prostředí

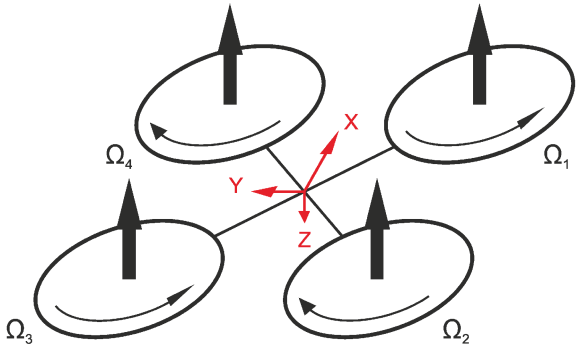
V balení je dále akumulátor, který je zdrojem energie pro kvadrikoptéru a speciální nabíječka pro tento akumulátor. Jsou zde i různé koncovky na české a zahraniční elektrické zásuvky.

Balení obsahuje samozřejmě návod k použití a také barevné samolepky použitelné pro hru dvou kvadrikoptér Ar.Drone proti sobě. Tohle je zatím podporováno pouze oficiálními aplikacemi.

3.2 Princip létání

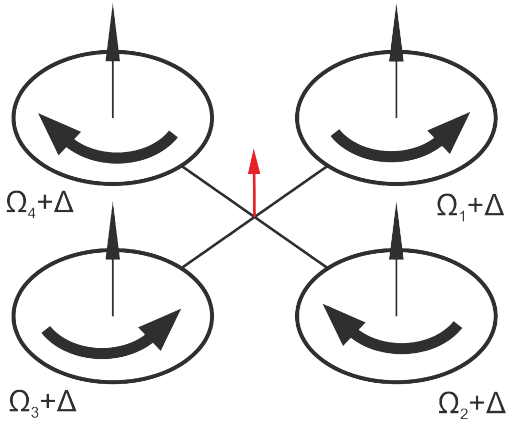
Jak již bylo napsáno v úvodu této kapitoly, kvadrikoptéra je stroj se čtyřmi vrtulemi v jedné rovině. Princip létání je u těchto strojů velmi specifický a relativně složitý. Z pohledu dynamiky má kvadrikoptéra 6 stupňů volnosti (posuvy po třech osách a rotace okolo tří os) a pouze 4 vrtule, navíc umístěné v jedné rovině.

Kvůli vyrovnání momentů se dvě vrtule točí po směru hodinových ručiček a zbylé dvě opačně. Samozřejmě dvě vrtule jsou pravé a dvě levé, aby se i přes nutnost opačného otáčení vytvářel tah jedním směrem. Viz. Obr. 8



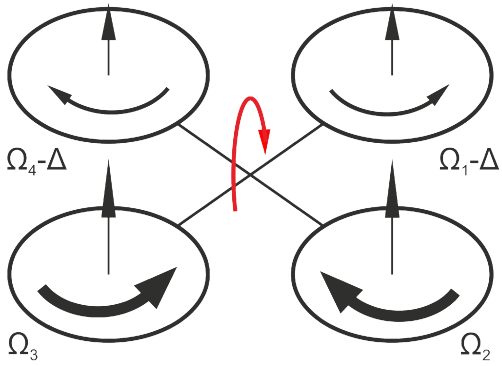
Obr. 8: Znázornění směru otáčení vrtulí

Veškeré pohyby kvadrikoptéry se odvíjí od změny otáček vrtulí (motorů), čímž se změní úhlová rychlost vrtulí , která je značena Ω. Pohybu nahoru případně dolů (podél osy Z) se dosáhne snížením případně zvýšením otáček všech vrtulí o stejnou hodnotu Δ. Viz. Obr. 9



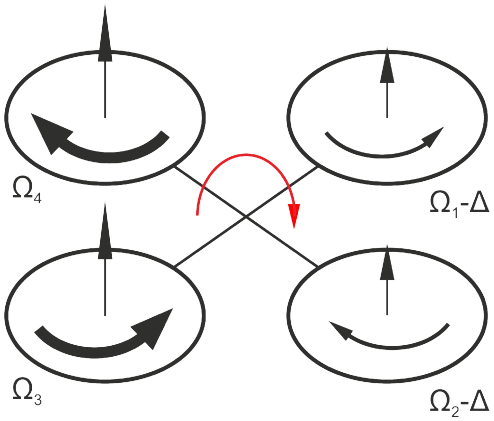
Obr. 9: Otáčení vrtulí při letu vzhůru

Pro let dopředu či dozadu (podél osy *X*) se musí kvadrikoptéra naklonit okolo osy *Y*. Tento úhel náklonu se anglicky označuje *pitch*, nebo znakem θ. Náklonu se dosáhne snížením otáček předního či zadního páru vrtulí o hodnotu Δ. Viz. Obr. 10



Obr. 10: Pitch – létání dopřetu (širší šipky značí vyšší otáčky motorů)

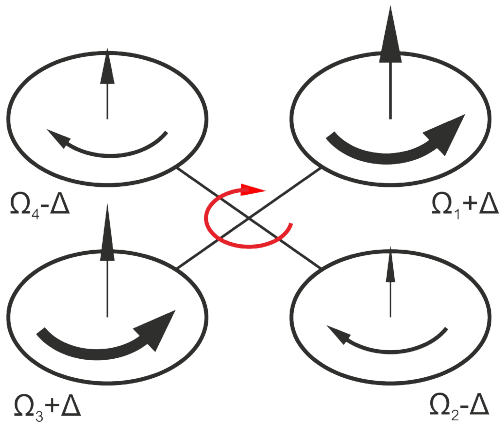
Letu doleva nebo doprava (podél osy *Y*) je možné docílit náklonem kolem osy *X*. Takový náklon se anglicky nazývá *roll* a značí se φ. Náklonu se dosáhne opět snížením otáček a to levé, nebo pravé dvojice vrtulí o určitou hodnotu Δ. Viz. Obr. 11



Obr. 11: Roll – létání do boku

Poslední dvojice pohybů je rotace kolem osy *Z*. Tento úhel natočení se anglicky značí *yaw* a náleží mu řecké písmeno ψ. Tohoto pohybu docílíme změnou otáček všech čtyř vrtulí. Změny při rotaci kolem osy ve směru hodinových ručiček jsou následující. Dvojice vrtulí, otáčející se také ve směru hodinových ručiček, sníží své otáčky o hodnotu Δ_A. Zbylá dvojice,

otáčející se proti směru hodinových ručiček své otáčky zvýší o hodnotu Δ_B. Viz. Obr. 12



Obr. 12: Yaw – rotace kolem svislé osy

Tato kapitola čerpá ze zdrojů [13] a [14].

3.3 Vybavení kvadrokoptéry

Tato podkapitola se zabývá jednotlivými částmi kvadrikoptéry Ar.Drone a jejich specifickými parametry.

3.3.1 Akumulátor a nabíječka

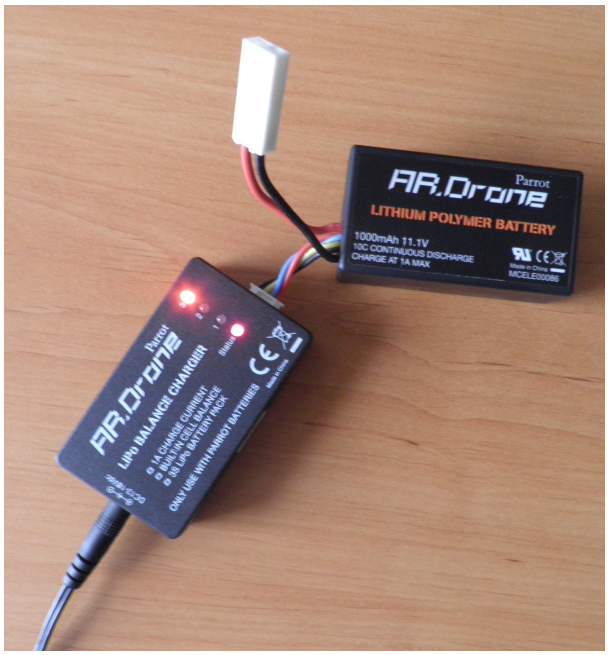
Jediným zdrojem energie je lithium-polymerový akumulátor (viz. obr. 13), značený Li-Pol, nebo Li-Po. Technologie Li-Pol vychýzí z technologie lithio-iontových akumulátorů. U obou typů vzniká elektrická energie chemickou reakcí.

Napětí Li-Pol článků se může pohybovat od 2,7 V do 4,23 V. Tyto hodnoty nesmí být překročeny, protože by se akumulátor velmi rychle zničil, nebo v krajním případě mohl i ohrozit uživatele. Z toho důvodu je nutné, aby každý článek byl hlídán elektronickým obvodem, který jej, v případě hrozícího překročení některé hranice, odpojí. Akumulátory je dovoleno nabíjet pouze originální nabíječkou dodávanou výrobcem. [15]

Akumulátor použitý v Ar.Drone je tříčlánkový o celkové kapacitě 1000 mAh. Udávané napětí je 11,1 V. Měřením bylo zjištěno, že nabitý akumulátor poskytuje napětí okolo 12,4 V, vybitý pak necelých 9 V. Průměrná doba létání s nabitým akumulátorem byla 10 – 12 minut. Při práci s kvadrikoptérou bez použití motorů (testování senzorů apod.) je výdrž více než 5 hodin.

Obr. 13: Akumulátor Ar.Drone

Nabíjení probíhá přes speciální nabíječku dodávanou v balení (viz. Obr. 14). K nabíjení slouží tzv. balanční konektor se čtyřmi kontakty. Díky tomu se nabíječka stará o nabíjení každého článku v akumulátoru zvlášť. Stav nabití každého článku je signalizován příslušnou kontrolkou na nabíječce. Čtvrtá kontrolka informuje o celkovém stavu nabíjení. Celý proces nabíjení trvá okolo 90 minut při vybitém akumulátoru.



Obr. 14: Nabíječka a akumulátor Ar.Drone

3.3.2 Motory a vrtule

Jak již bylo uvedeno, kvadrikoptéra má 4 vrtule (2 pravé, 2 levé). Vrtule mají průměr 20 cm a stoupání 19,5 mm. Na základě těchto údajů společně s požadovanými otáčkami jsou voleny dostatečně výkonné motory.

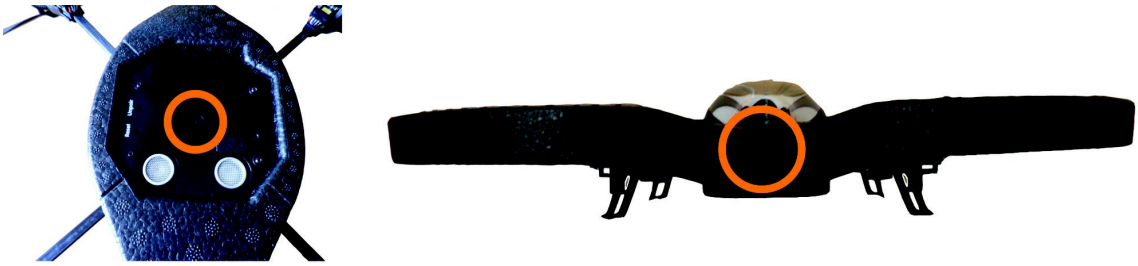
Kvůli požadavku na řízení otáček každé vrtule zvlášť, otáčí každou vrtulí samostatný elektromotor (viz. Obr. 15). Ar.Drone používá čtyři střídavé motory o výkonu 15 W. Motor má otáčky 28 000 RPM (otáček za minutu) a při největším zrychlení kvadrikoptéry dosahuje až 41 400 RPM. Vrtule navíc nejsou připevněny přímo na výstupní hřídeli motoru, ale jsou poháněny přes ozubená kola (viz. obr. 15). Převodový poměr tohoto soukolí je 8:75 (tedy 4:35).

Rychlost motorů je řízena elektronikou, která se sestává zejména z 8bitového mikrokontroleru a 10-bitového ADC (analogově-digitálního) převodníku. Deska s řídící elektronikou obsahuje také dvoubarevnou informační LED pro signalizaci různých situací a režimů. [16]

Obr. 15: Převod mezi vrtulí a motorem; detail motoru a řídící elektroniky [16]

3.3.3 Přední a spodní kamera

Kvadrikoptéra Ar.Drone je vybavena dvěma kamerami (viz. Obr. 16). Kamera, která směřuje dopředu, snímá na čip typu CMOS v rozlišení VGA (tzn. 640x480 pixelů) v zorném úhlu 93°. Druhá kamera je umístěna zespodu kvadrikoptéry a při letu snímá podlahu. Její maximální rozlišení čipu CMOS je QCIF (176x144 pixelů) a zorný úhel je 64°. Přední kamera snímá s frekvencí 15 fps (snímků za sekundu) a spodní kamera s frekvencí 60 fps. [16] [17]

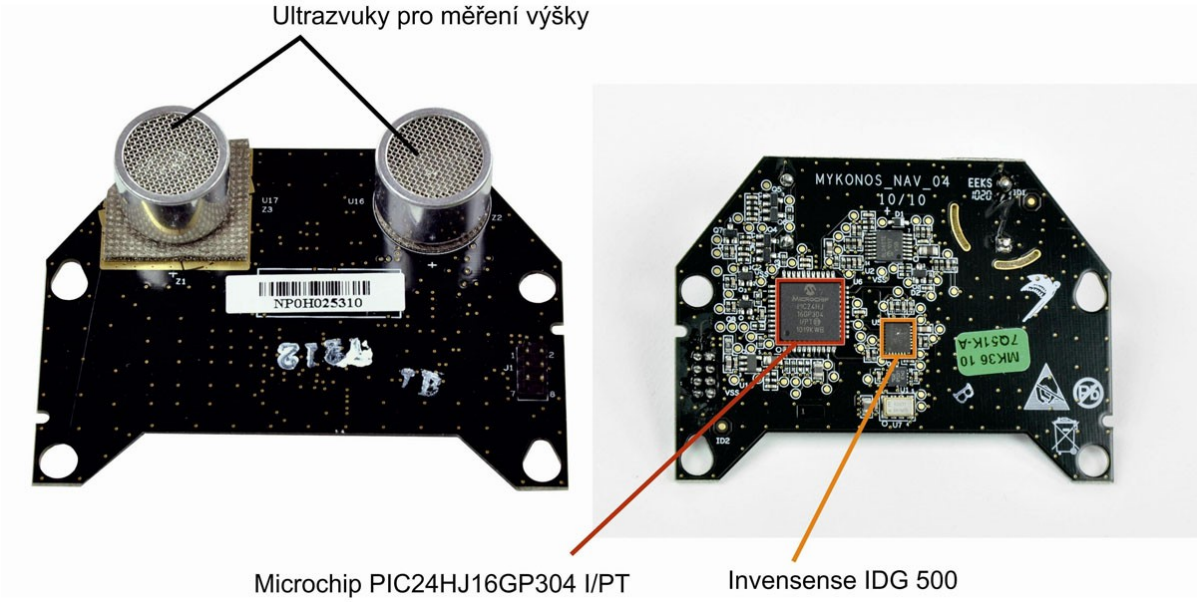


Obr. 16: Umístění spodní a přední kamery na Ar.Drone

3.3.4 Hardware a software

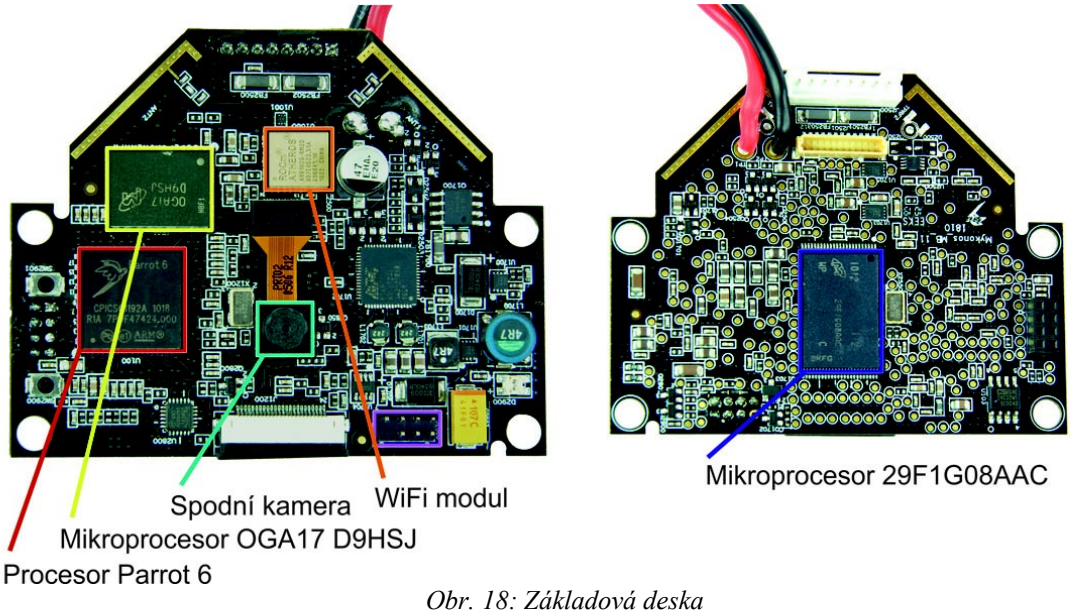
Pod spodním krytem kvadrikoptéry Ar.Drone najdeme dvě desky plošných spojů.

Menší desku můžeme označit jako navigační (viz. Obr. 17), protože obsahuje součástky poskytující údaje o letu. Jedná se zejména o gyroskop Invensense IDG 500 a výškoměr. Údaje z těchto senzorů zpracovává mikroprocesor Microchip PIC24HJ16GP304 I/PT taktovaný na frekvenci 40 MHz se šířkou sběrnice 16bitů.



Obr. 17: Navigační deska [16]

Druhá deska je označovaná jako základová (viz. obr. 18) a obsahuje procesor, operační paměť, WiFi modul a další. Procesor Parrot 6 postavený na technologii ARM9 s taktem 468 MHz slouží pro běh systému s Linuxovým jádrem. Systém se stará o veškerou komunikaci, zpracování informací ze senzorů a stabilizaci. Systém ke své práci používá operační paměť DDR o velikosti 128 MB s časováním 200 MHz. Dále jsou na této desce mikroprocesory Micron OGA17 D9HSJ a Micron 29F1G08AAC. K této desce jsou připojeny kamery, napájení a ovládání motorů, navigační deska, zdroj energie a obsahuje 2 tlačítka (restart a zrušení párování s WiFi zařízením) a informační LED. [16] [17]



Obr. 18: Základová deska

3.3.5 WiFi připojení

Ovládání kvadrikoptéry je samozřejmě bezdrátové, ale na rozdíl od podobných RC modelů to není pomocí rádiového ovládání. Ar.Drone obsahuje WiFi modul, pomocí kterého se dá propojit se zařízením podporujícím standard WiFi.

Připojení zajišťuje ROCm Atheros AR6102G-BM2D b/g WiFi modul. [17]

3.3.6 Ultrazvuk

Na bezkontaktní měření vzdálenosti se v robotice používá buď optického paprsku, nebo zvukového vlnění. Optický paprsek se odrazí od překážky a dopadne na snímač. Na základě doby od vyslání do návratu paprsku se spočítá vzdálenost překážky od senzoru. Metoda s použitím zvukového signálu funguje na stejném principu. Používá se signálu o frekvenci 40 kHz, který je pro lidský sluch neslyšitelný, takže při provozu robota člověk není rušen nepříjemným pískáním. Součástka používaná v robotice pro tento účel může vypadat podobně jako na obrázku 19. Vzdálenost se potom vypočítá z rychlosti šíření světla (příp. zvuku) a časového intervalu od vyslání signálu do jeho přijetí.

Ar.Drone používá podobný senzor k určení letové výšky. Ultrazvuk je umístěn na spodní straně kvadrikoptéry a odrazem od země se měří letová výška.

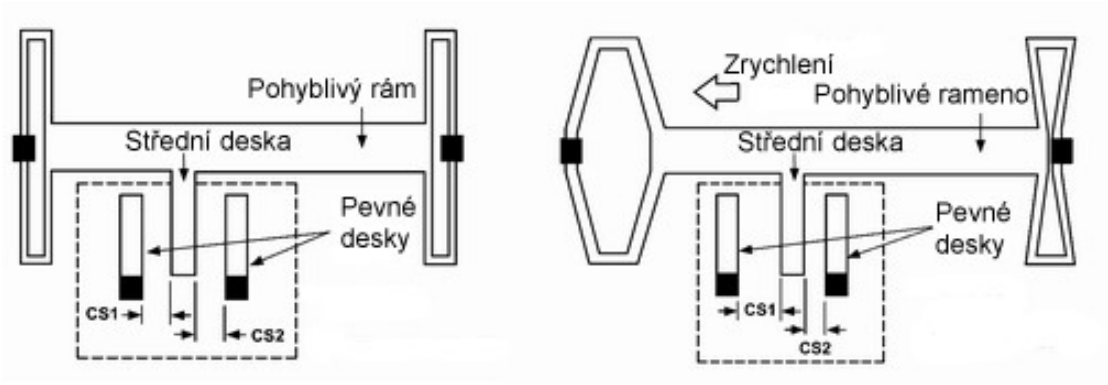
Obr. 19: Ultrazvuk SRF05 [18]

3.3.7 Snímače polohy a natočení

Při letu je nutné znát nejen výšku, ale i polohu v jaké se stroj nachází. Pro tyto účely v praxi slouží akcelerometry a gyroskopy.

Akcelerometry jsou senzory, které měří dynamické zrychlení (akceleraci), nebo statické zrychlení (poloha v gravitačním poli) předmětu respektive senzoru. Gyroskopy jsou využívány k zjištění změny natočení nebo úhlové rychlosti předmětu, ke kterému jsou připevněny. V dnešní době jsou tyto senzory, navíc s vyhodnocovacími obvody, zabudovány do jednoho integrovaného obvodu, který má velikost jen pár milimetrů. Tato technologie, která v sobě spojuje polovodičové a mechanické části se nazývá MEMS (Micro-Electro-Mechanical Systems).

Akcelerometry pracují na principu změny kapacity vnitřního proměnného kondenzátoru vlivem působící síly vzniklé zrychlením pouzdra senzoru. Samotný senzor je povrchová mikrochemická polykřemíková struktura plovoucí na povrchu křemíkového monokrystalu. Křemíkové pružiny umožňují pohyb monokrystalu a poskytují mechanický odpor. Pohyb této struktury je převeden na změnu kapacity diferenciálního kondenzátoru. Kondenzátor se skládá z pohyblivé desky, která je pevně spojena s plovoucí strukturou a ze dvou pevných desek spojených s rámem součástky (viz. obr. 20).



Obr. 20: Akcelerometr v klidu; akcelerometr při pohybu vlevo [19]

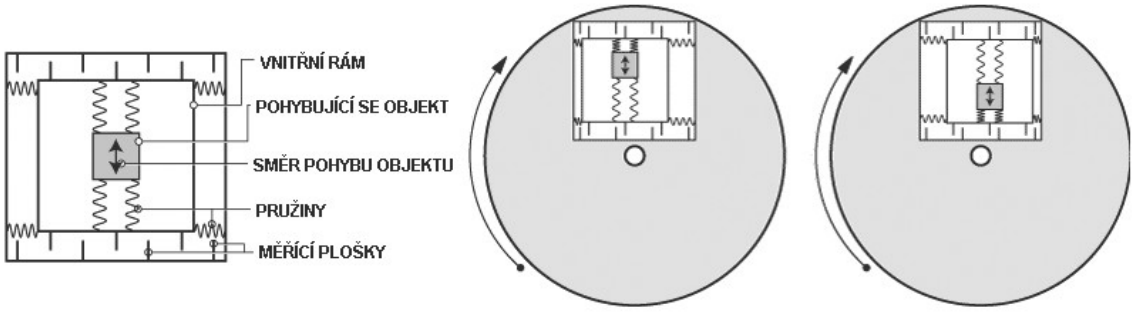
Diferenciální kondenzátor tvoří dělič pro dva obdélníkové signály stejné amplitudy vzájemně posunuté ve fázi o 180°, které budí jeho pevné desky. Působícím zrychlením dojde k posunutí prostřední pohyblivé desky a tím ke změně dělícího poměru kondenzátoru. Na výstupu se tím pádem objeví obdélníkový signál o amplitudě úměrné hodnotě zrychlení a fázi, která nese informaci o směru pohybu nosníku, tedy o směru působícího zrychlení.

Vyhodnocení obdélníkového signálu provádí demodulátor s použitím hodinového signálu z generátoru. Demodulátor potlačí všechny, které nejsou synchronní s hodinovým signálem. Signál ve fázi s hodinovým je vyhodnocen jako kladné napětí na výstupu, signál posunutý o 180° jako záporné. Výstup z demodulátoru je přiveden na vstup předzesilovače k provádění kalibrace při zrychlení 0 g. Interní zpětná vazba pomocí elektrostatické síly vrací pohyblivý nosník zpátky do původní pozice. [19]

Gyroskopy poskytují údaje o tom, jak rychle se měřený objekt otáčí. V běžně používaném kartézském souřadném systému je možné objektem rotovat kolem tří os. Gyroskopy vyráběné technologií MEMS pracují na principu Coriolisovy síly.

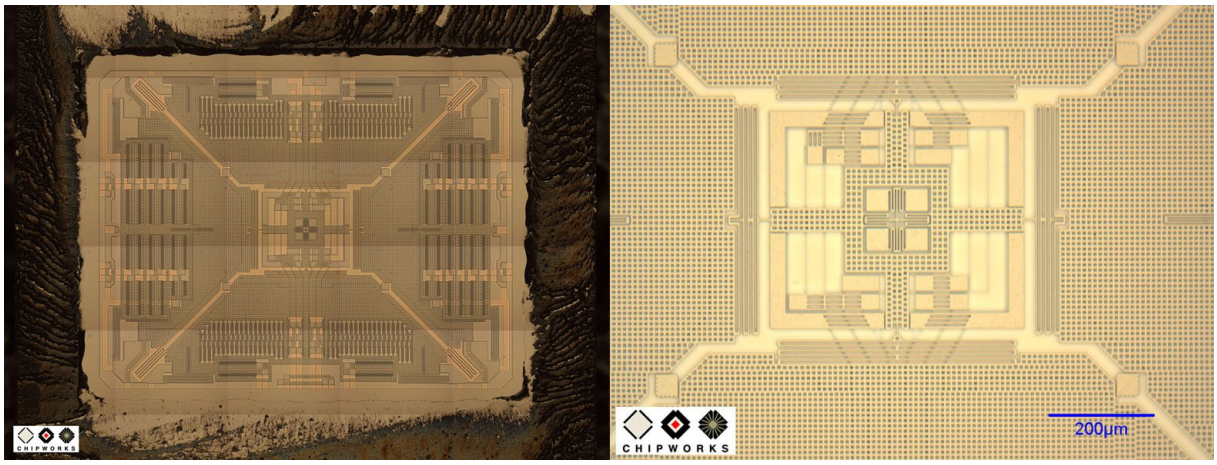
Coriolisova síla je síla působící na hmotný bod pohybující se rychlostí v v soustavě rotující úhlovou rychlostí ω.

V MEMS gyroskopech je na pružinách uvnitř rámu upevněn objekt o přesně dané hmotnosti (viz. Obr. 21). Objektem je elektronicky pohybováno v přesných intervalech. Protože Coriolisova síla je úměrná rychlosti a směru otáčení, můžeme určit úhlovou rychlost. Je nutné zajistit, aby pohyb objektu byl kolmý na osu rotace. Coriolisova síla způsobí stlačení vnějších pružin rámu a posun měřících plošek, které změní kapacitu vzduchových kondenzátorů. Na základě těchto hodnot jsme schopni určit úhlovou rychlost. [20]



Obr. 21: Znázornění principu gyroskopu [20]

Ar.Drone v době má jeden gyroskop a to Invensense IDG 500. Tento MEMS gyroskop dokáže snímat pohyb rychlostí až 500°/s pokud nám záleží na rychlosti, nebo 110°/s pokud nám jde o velkou přesnost. Rozměry integrovaného obvodu jsou 4x5x1,2 mm. Gyroskop obsahuje funkci automatického vynulování a teplotní čidlo. Na obrázku 22 je zobrazen reálný gyroskop při zvětšení pod mikroskopem. [21]



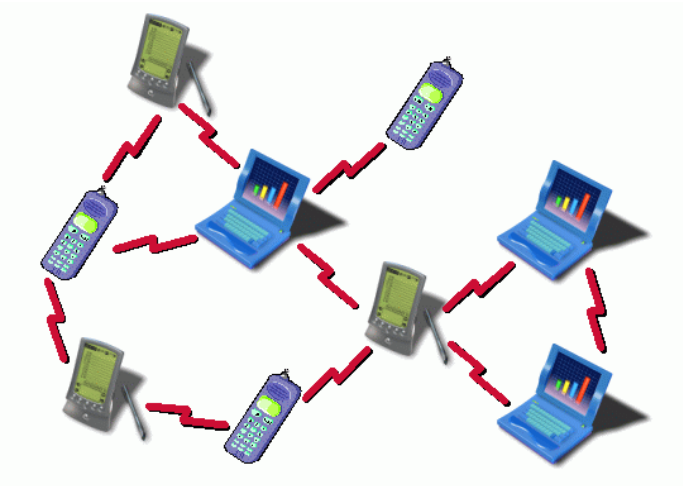
Obr. 22: Fotografie podobného gyroskopu pod mikroskopem [22]

4 KOMUNIKACE

Jak již bylo zmíněno v předchozí kapitole, veškerá komunikace mezi Ar.Drone a ovládacím zařízením probíhá přes WiFi. V této kapitole bude podrobně probráno, jak tato komunikace probíhá.

4.1 Ar.Drone jako přístupový bod

Původně Ar.Drone fungoval pouze v režimu Ad-hoc (schéma sítě na obr. 23). Ad-hoc mezi počítači je jednoduché propojení, kdy na jednom počítači vytvoříme síť, tento počítač se v tuto chvíli stane pro ostatní přístupový bod (Access Point – AP) a ostatní se připojí na novou síť. V některých operačních systémech označovaná jako rovnocenné počítače. V případě odpojení prvního počítače se řízení ujme náhodně vybraný jiný účastník.



Obr. 23: Schéma Ad-hoc sítě [23]

Do verze firmwaru Ar.Drone 1.7.4 byl umožněn pouze režim Ad-Hoc. Verze 1.7.4 přišla s důležitou změnou a to z režimu Ad-Hoc na AP. S touto změnou se vlastně z Ar.Drone stane přístupový bod, který je možné vyhledat a připojit se k němu. Právě od této doby je umožněno používat zařízení s OS Android, který bez zásahů do systému Ad-hoc sítě vyhledat neumí. [24]

V základním nastavení má Ar.Drone IP adresu 192.168.1.1 a zařízení které se k němu připojí, dostane přidělenou adresu 192.168.1.2.

4.2 Protokoly UDP a TCP

Protokol TCP (Transmision Control Protocol) je protokolem transportní vrstvy. TCP je spojovaná služba. To znamená, že se naváže spojení a to je udržováno po celou dobu komunikace. Spojení je plně duplexní, to znamená, že se můžou data přenášet současně oběma směry nezávisle na sobě. Ztracená, nebo poškozená data jsou vyžádána znovu. Data obsahují kontrolní součet pro ověření jejich správného a úplného doručení. Ar.Drone používá protokol TCP pro data, u kterých je důležité jejich doručení (např. konfigurační data).

UDP (User Datagram Protocol) je také protokol na transportní vrstvě a je alternativou k protokolu TCP. UDP je služba nespojovaná – nenaváže se spojení. Odesílatel odešle data a už se nestará o to, zda se doručí v pořádku, nebo zda se vůbec doručí. Data také nemusí přijít v pořadí, v jakém byly odeslány. Může se to zdát jako nevýhoda, ale jsou určité případy, kdy není nutné tyto věci kontrolovat. Velká výhoda oproti TCP je nižší režie. Protokol UDP je u Ar.Drone použit k odesílání letových povelů a k přijímání navigačních dat a videa.

Komunikace na těchto protokolech může být synchronní nebo asynchronní.

U synchronní se po odeslání dat pouze čeká na odpověď. Asynchronní komunikace umožňuje, aby systém po odeslání dat, prováděl jinou práci, než druhá strana odpoví. Data jsou při přenášení rozdělena na tzv. pakety.

4.3 Porty pro komunikaci

Porty používá systém pro přiřazení datům ze síťové komunikace jednotlivým aplikacím. Port se označuje číslem. Existuje dohoda, na základě které se některé čísla portů používají jen pro předem dohodnuté aplikace. Např. port 80 pro http. Kompletní seznam obsazených portů je možné najít na stránkách organizace IANA (Internet Assigned Numbers Authority).

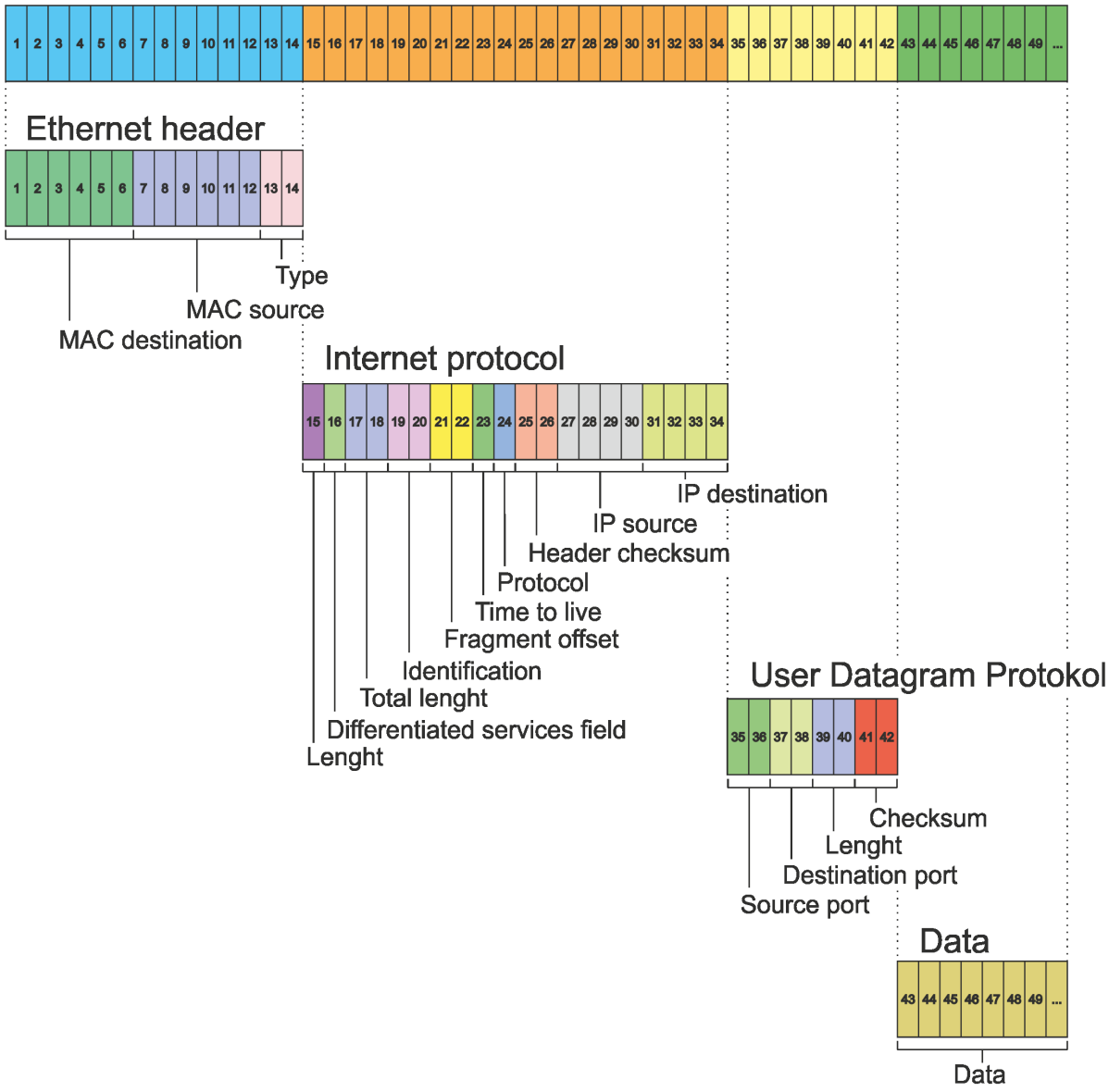
Ar.Drone pro svou komunikaci využívá porty 5554, 5555, 5556 a 5559. Čtyři porty jsou použity pro oddělení dat podle druhu. Použití je uvedeno v tabulce 1.

Port	Protokol	Použití
5554	UDP	Navigační data
5555	UDP	Video
5556	UDP	Letové příkazy
5559	TCP	Kritická data

Tabulka 1: Porty a jejich využití

4.4 Pakety

Paket je základní jednotkou síťové komunikace. Pakety se skládají ze záhlaví, těla a zápatí. Tato práce se zabývá nejčastěji pakety protokolu UDP. Na obrázku 24 je zobrazeno z čeho se takový UDP paket skládá. Velikost dat se může paket od paketu lišit. [25]



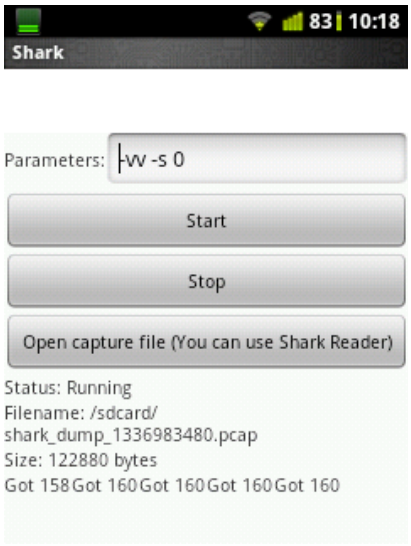
Obr. 24: Struktura UDP paketu

4.5 Program WireShark

Pro lepší pochopení komunikace v síti byl používán program WireShark. Díky datům, která je schopný tento program získat, bylo možné zkoumat pakety nutné pro komunikaci Ar.Drone a ovládacího zařízení.

WireShark je program pro sledování síťové komunikace. Umožňuje zachytávání a prohlížení provozu v počítačové síti. V této práci významně pomohl pro pochopení komunikace mezi ovládacím zařízením a kvadrikoptérou. Program běží na mnoha platformách, jako je např. OS Windows, OS Linux, OS X a další.

Velkou roli sehrála i verze pro mobilní telefony se systémem Android. Aplikace je běžně veřejně dostupná.

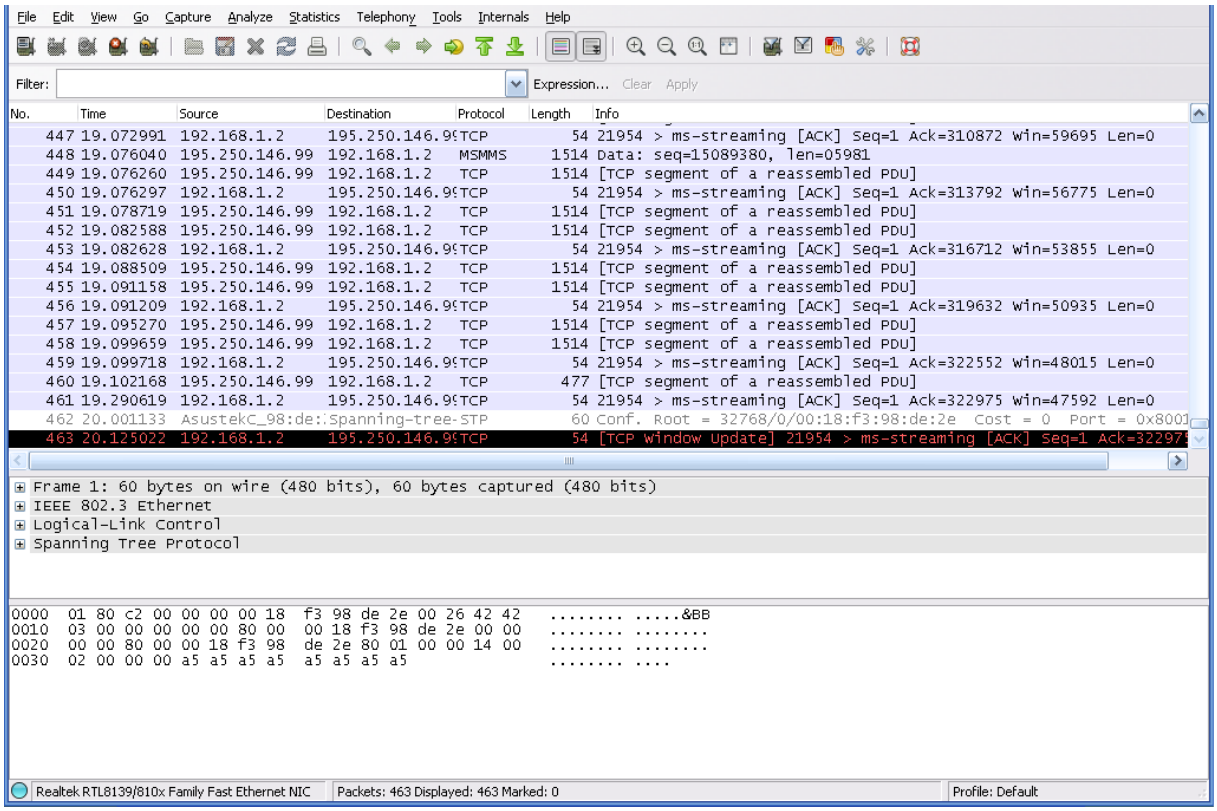


Obr. 25: Shark pro OS Android

Ovládání aplikace je velmi snadné. Stačí telefon připojit k síti a spustit aplikaci Shark (viz. Obr. 25). Parametry není vůbec nutné nějak měnit. Pro spuštění záznamu komunikace mezi telefonem a AP stačí kliknout na tlačítko Start. Ve spodní části obrazovky se objeví základní údaje o vytvářeném souboru, do kterého je komunikace ukládána. Nyní je možné spustit sledovanou aplikaci (pokud již neběží) a pracovat s ní. V našem případě jsme spustili aplikaci Androne Freeflight.AR a začali létat s kvadrikoptérou. Po určité době se můžeme vrátit opět do aplikace Shark a kliknutím na tlačítko Stop ukončíme záznam komunikace, který se nám následně uloží do souboru na paměťové kartě. Tento soubor můžeme prohlížet buď přímo v telefonu pomocí aplikace Shark Reader (viz. Obr. 26), nebo si ho otevřít na počítači, kde je prohlížení pohodlnější a jsou zde větší možnosti filtrace dat.

Obr. 26: Aplikace Shark Reader (Android)

Program WireShark pro operační systémy osobních počítačů disponuje více nástroji, než mobilní aplikace. Základní princip činnosti je však stejný. Prohlížení výsledního souboru je pak pohodlnější zejména kvůli větší obrazovce a lepším možnostem filtrace, případně barevného odlišení podle různých kritérií.



Obr. 27: Program WireShark pro osobní počítače

Na obrázku 27 je program v režimu naslouchání a záznamu komunikace. Režim prohlížení odchycených paketů vypadá téměř stejně. V horní části okna jsou prvky pro ovládání programu a filtraci zaznamenané komunikace. Hned pod lištou s textovým polem pro zadávání parametrů filtrování jsou zobrazeny pakety vyhovující filtračním parametrům (v tomto případě všechny) přijaté nebo odeslané z počítače. Každý jednotlivý paket můžeme vybrat a v dolních dvou polích se nám zobrazí podrobně. V první části je paket rozdělen podle částí, které obsahuje, zde si můžeme vybrat část, která nás zajímá. Ve spodní části jsou už samotné byty paketu. Při výběru části paketu v prostřední části se nám tato část dat zvýrazní v části dolní. Data jsou zde zobrazena ve dvou formách. Vlevo jsou data ve formátu hex (jde přepnout na binární formát) a v pravé časti, jsou tyto byty převedeny na znaky (char). Byty, které nejsou zastoupeny v tabulce znaků, jsou zobrazeny jako tečky.

5 PŘÍKAZY

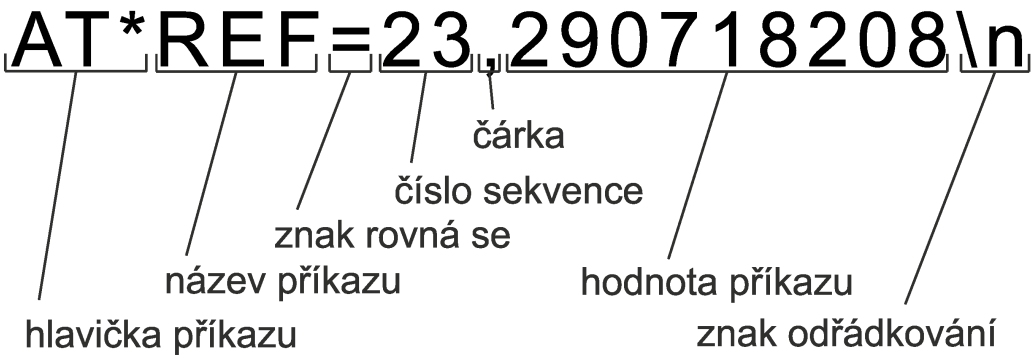
Předchozí kapitoly popisovaly co to Ar.Drone je a co všechno má k dispozici. Tato kapitola pojednává už o konkrétních příkazech, které způsobí mimo jiné létání Ar.Drone.

5.1 Formát příkazů

Aby Ar.Drone příkazy rozpoznalo, je nutné, aby splňovaly všechny náležitosti popsané v této podkapitole. Pokud příkaz není správný, Ar.Drone na něj nereaguje, nebo může reagovat nepředvídatelně.

Příkazy se posílají přes protokol UDP pouze na port 5556 (viz. kapitola 4.3). Při odeslání na jiný port Ar.Drone nereaguje.

Příkazy vypadají jako textové řetězce (viz. obr. 28), ale ve skutečnosti se odesílají jako jednotlivé znaky (char). Například název příkazu (v tomto případě REF) bude tedy vypadat tato: $52_{(16)}$ $45_{(16)}$ $46_{(16)}$, nebo $01010010_{(2)}$ $01000101_{(2)}$ $01000110_{(2)}$. Čísla se také převádějí na textové znaky (char).



Obr. 28: Příklad příkazu s popisem

Tento ukázkový příkaz odeslaný na správný port způsobí vzlétnutí Ar.Drone (pokud je Ar.Drone ve správném režimu). Datová část UDP paketu odesílaného na port 5556 bude tedy vypadat následovně (v hex): *41 54 2a 52 45 46 3d 31 2c 32 39 30 37 31 38 32 30 38 0d*.

Další důležitá věc, na kterou je potřeba klást stejný důraz, je perioda odesílání. Pakety s řídícími příkazy musí Ar.Drone obdržet minimálně jednu za 2 s. Ar.Drone se v tomto případě stabilizuje a po určité době přistane. Pokud během 2 s nepřijde žádný příkaz, je potřeba provést znovu inicializaci.

Jak je z obrázku 28 patrné, příkaz má 7 základních částí. Postupně si každou rozebereme.

5.1.1 Hlavička příkazu

Je v příkazu povinná a je vždy stejná. Skládá se ze tří znaků. Velké A, velké T a hvězdička. Začíná jí datová část paketu. V hexadecimální soustavě bude tedy AT* vypadat takto:

- znak A: $41_{(16)}$
- znak T: $54_{(16)}$
- znak *: $2a_{(16)}$

5.1.2 Název příkazu

Název je také povinná hodnota. Na rozdíl od hlavičky už ale nebude v každém příkazu stejný. Příkazy jsou různé pro různé činnosti Ar.Drone různé. Všechny příkazy budou přesně

rozepsány dále v této kapitole.

5.1.3 Znak rovná se

Tento znak odděluje hlavu příkazu od těla, které obsahuje hodnoty příkazu. Opět se jedná o povinný znak. Znak rovná se převedený do hexadecimální soustavy je vyjádřen jako $3d_{(16)}$, případně v binární soustavě jej vyjadřuje hodnota $00111000_{(2)}$.

5.1.4 Číslo sekvence

Touto částí začíná hodnotová část sekvence. Číslo sekvence je opět povinná část příkazu. Důležité u tohoto čísla je, aby pořád rostlo. S každým doručeným příkazem do Ar.Drone musí být číslo alespoň o 1 větší. Pokud přijde příkaz s nižším číslem sekvence, Ar.Drone tento příkaz nevykoná. Pokud přijde příkaz, kde je číslo větší o víc než 1 než v předchozím příkazu, nevadí to. Je to z důvodu použitého protokolu. Jak je uvedeno v kapitole 4.2 protokol UDP nekontroluje doručení paketů a tímto způsobem je zajištěno dodržení posloupnosti příkazů i při ztrátě paketu, nebo nesprávném doručení.

Výjimka: číslo sekvence nemusí být vyšší než v předchozím příkazu v jediném případě. Pokud je odesílaný příkaz AT*COMWDG, který nuluje číslo sekvence, může být číslo sekvence 1.

5.1.5 Čárka

Čárka v hodnotové části příkazu slouží pro oddělení čísla sekvence a vlastní hodnoty příkazu. Je to opět povinný znak (kromě výjimky uvedené v 5.1.4). V hexadecimální soustavě je čárka představovaná hodnotou 2c. Stejný znak se používá v některých příkazech na oddělení jednotlivých částí hodnot příkazu (např. příkaz AT*PCMD)

5.1.6 Hodnota příkazu

Tato část se odvíjí hlavně od názvu příkazu (5.1.2). Každý příkaz má jinou syntaxi hodnot. Některé příkazy hodnotu nemají vůbec. Názvy a syntaxe příkazů budou probrány dále v této kapitole.

5.1.7 Znak odřádkování

Jedná se opět o povinný znak. Ar.Drone tím dostane najevo, že příkaz končí. Může tímto znakem končit celý paket. Případně může následovat další příkaz v tom stejném paketu opět svou hlavičkou (AT*).

Pozor: nejedná se o 2 znaky (zpětné lomítko \ a *n*), ale pouze o jeden. Znak odřádkování má v hexadecimální soustavě hodnotu 0d.

5.2 Příkaz AT*REF

Syntaxe: **AT*REF={0},{1}\n**

{0} – číslo sekvence

{1} – hodnota příkazu: celočíselná hodnota (int), délka 32 bitů, ovládání základních činností: vzlétnutí, přistání, nouzové vypnutí

Pro číslo posílané na místo {1} jsou určitá pravidla k jeho vytvoření. Protože se jedná o 32 bitové číslo, může nabývat hodnot 0 až 2^{32}-1. Po aplikování následujících pravidel nás budou zajímat pouze 3 čísla z tohoto rozsahu. Je totiž přesně dané na jakou hodnotu má být

který bit nastavený.

- Bity 18, 20, 22, 24 a 28 budou vždy nastaveny na 1.
- Bit 8 nastavený na 1 je nouzové zastavení (vysvětleno dále), nastavený na 0 je běžný provoz
- Bit 9 nastavený na 1 znamená vzlétnutí, nastavený na 0 je přistání (vysvětleno dále)
- Všechny ostatní bity jsou nastaveny na 0

Nouzové zastavení

Jedná se o situaci, kdy se stane nepředvídatelná věc a je potřeba rychle ukončit let. Při odeslání tohoto příkazu se všechny motory okamžitě zastaví a kvadrikoptéra začne nekontrolovatelně padat. Není proto rozumné provádět toto ve velké výšce, kde by pád mohl mít vážné následky pro kvadrikoptéru. Do nouzového zastavení se kvadrikoptéra dostane i při kontaktu vrtule s překážkou, nebo při opravdu velkém naklonění (okolo 80°). Návrat z tohoto stavu se provádí odesláním příkazu pro přistání.

Vzlétnutí

Tento příkaz zajistí roztočení motorů a automatické vzlétnutí kvadrikoptéry do cca 80 cm. Po ustálení (1 – 2 s) se přesune do předem nastavené maximální výšky. Do výšky 80 cm vyletí i když je maximální letová výška nastavená nižší (např. 50 cm).

Přistání

Tímto příkazem se spustí přistávací manévr, který spočívá v rychlém klesání do výšky cca 1 m nad zemí, kde začne kvadrikoptéra brzdit (při klesání z opravdu velké výšky se vrtule skoro zastaví a rychlost klesání je docela velká), ustálí se kousek nad zemí a přibližně ve výšce 20 cm nad zemí vypne motory a dopadne na zem. Pád z 20 cm se může v některých situacích zdát hlučný a nebezpečný, ale Ar.Drone je na tento pád konstruováno. Je to způsobeno tím, že ultrazvukové čidlo na měření výšky nedokáže změřit menší vzdálenost než cca 20 cm (viz. kap. 3.3.6).

Z uvedených pravidel pro nastavení hodnoty příkazu tedy plyne, že hodnoty, které se budou měnit, jsou pouze na pozici 8 a 9. V binární soustavě můžou být pouze hodnoty 0 (false) nebo 1 (true), takže máme 4 různé možnosti nastavení tohoto příkazu. Při logickém zamyšlení ještě jednu z nich eliminujeme, protože při nouzovém zastavení bude samozřejmě bit 9 nastavený na 0 (nechceme vzlétnout, když potřebujeme kvadrikoptéru co nejrychleji vypnout).

Čísla, které se tedy budou posílat v hodnotě příkazu, budou:

- **290717952** pro nouzové zastavení
- **290718208** pro vzlétnutí
- **290717696** pro přistání

Výsledné čísla jsou na začátku této podkapitoly sestavovány pomocí nastavování jednotlivých bitů v 32bitovém čísle a poté převedeny z binární soustavy do desítkové. Ovšem pozor opět na to jak budou odeslána. Jako celý příkaz, tak i tohle číslo se převede na jednotlivé hodnoty typu char, tyto hodnoty dále na typ byte a teprve v této formě se můžou zakomponovat do paketu a přenést po síti.

Příklad: **AT*REF=9,290718208\n** *(vzlétnutí Ar.Drone)*

5.3 Příkaz AT*PCMD

Syntaxe: **AT*PCMD={0},{1},{2},{3},{4},{5}\n**

{0} – číslo sekvence
{1} – nastavení progresivních příkazů (používáme číslici 1)
{2} – hodnota naklonění vlevo (–) nebo vpravo (+) (roll)
{3} – hodnota naklonění dopředu (–) nebo dozadu (+) (pitch)
{4} – rychlost stoupání (+) nebo klesání (–)
{5} – úhlová rychlost otáčení kolem svislé osy – vlevo (–), doprava (+)

Ar.Drone má nastavený maximální náklon (ve výchozím nastavení okolo 30° – jde změnit jedním z dalších příkazů). Hodnoty naklonění a rychlostí jsou reprezentovány celým číslem o velikosti 32 bitů. Toto číslo je získáno následujícím postupem:

- Úhel o který chceme aby se Ar.Drone naklonilo jsou procenta z této maximální hodnoty v desetinném tvaru (např. 0.23 znamená 23% => při výchozím nastavení tedy 6.9°)
- Tato procentuální hodnota se převede na 32bitový binární datový typ s plovoucí desetinnou čárkou (float) (v našem případě $3E6B851F_{(16)}$)
- Binární hodnota se poté převede na celočíselný datový typ o velikosti 32bitů (int) a tato hodnota se teprve vloží do parametru příkazu

Po doplnění hodnot do příkazu se tento opět převede na jednotlivé znaky (char), které se ve formě bytů pošlou UDP paketem do Ar.Drone.

Příklad: **AT*PCMD=14,1,0,-1102263091,0,0\n** *(let pouze dopředu)*
AT*PCMD=21,1,1063843267,0,0,0\n *(let pouze doprava)*
AT*PCMD=28,1,-1088170230,-1050924810,0,-1043878380\n
(zatáčení doleva při letu dopředu)

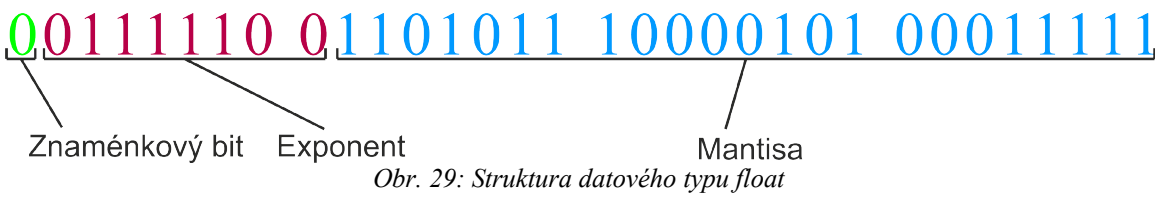
5.3.1 Převod reálných čísel na float

Převod desetinného čísla představujícího procenta z maximální hodnoty náklonu kvadrikoptéry na typ s plovoucí desetinnou čárkou probíhá podle normy IEEE 754, která popisuje nejenom formáty uložení hodnot v systému pohyblivé řádové čárky.

Podle této normy se desetinné číslo reprezentuje vztahem: $X_{FP}=(-1)^S * 2^{exp - b} * m$, kde:

- X_{FP} – reprezentovaná numerická hodnota z podmnožiny racionálních čísel
- 2 – báze (někdy radix)
- exp – vždy kladná hodnota exponentu posunutého o hodnotu *b*
- b – hodnota, díky které je exponent vždy kladný ($b = 2^{eb - 1} - 1$, kde *eb* je počet bitů vyhrazených pro exponent)
- m – mantisa
- s – znaménkový bit

Číslo uložené v paměti se tedy skládá ze 4 bytů. Ty mají tuto strukturu jako na obrázku 29.



Obr. 29: Struktura datového typu float

Znaménko (1 bit) – představuje znaménko reprezentovaného čísla. Pokud je bit 0, je znaménko kladné, pokud je hodnota bitu 1, znaménko je záporné.

Exponent (8 bitů) – hodnota exponentu, která je vždy kladná díky posunutí o hodnotu *b*. Podle normy IEEE 754 je $b = 127$. Jelikož byte může nabývat až 255 různých hodnot, při posunutí může být exponent hodnota z intervalu -127..128. Vychází to z použití dříve uvedeného vzorce, kde $ep = 8$.

Mantisa (23 bitů) – hodnota reprezentující desetinné číslo, která chceme vyjádřit. Bez exponentu a znaménkového bitu nepoužitelná.

[26]

5.4 Příkaz AT*CONFIG

Syntaxe: **AT*CONFIG={0},{1},{2}**

{0} – číslo sekvence
{1} – název nastavované vlastnosti
{2} – hodnota nastavované vlastnosti

Části parametru *název vlastnosti* a *hodnota vlastnosti* jsou textové řetězce, které jsou vždy uzavřeny v dvojitých horních uvozovkách ("). Tento znak je v ASCII tabulce znaků reprezentován hodnotou $22_{(16)}$.

Seznam nejpoužívanějších vlastností a jejich hodnot nastavenení je uveden v tabulce 2.

Název vlastnosti	Nastavitelné vlastnosti	Datový typ vlastnosti
general:ardrone_name	jméno kvadrikoptéry	string
general:navdata_demo	posílání navigačních dat	bool
control:euler_angle_max	maximální úhel náklonu [°]	float
control:altitude_max	maximální výška letu [mm]	int
control:altitude_min	minimální výška letu [mm]	int
control:control_vz_max	maximální rychlost svislého letu	float
control:control_yaw	maximální velikost úhlové rychlosti při otáčení okolo osy z	float
network:owner_mac	MAC adresa uživatele, při nastavení se jiný nepřipojí	string
pic:ultrasound_freq	nastavení frekvence ultrazvuku	int

*Tabulka 2: Nejpoužívanější hodnoty příkazu AT*CONFIG*

Příklad: **AT*CONFIG=32,"general:navdata_demo","TRUE"\n**
(zapne posílání navigačních dat)
AT*CONFIG=35,"control:altitude_max","500"\n
(nastavení maximální výšky letu na 50 cm)

5.5 Příkaz AT*FTRIM

Syntaxe: **AT*FTRIM={0}\n**

{0} – číslo sekvence

Tento příkaz nemá žádnou hodnotu (kromě čísla sekvence, které je v každém příkazu, viz. 5.1.4). Příkaz se používá většinou na začátku komunikace, při inicializaci. Tento příkaz vynuluje hodnoty gyroskopu. Neboli odečte aktuální hodnoty přicházející z gyroskopu, uloží do paměti a tyto hodnoty potom odečítá od dalších hodnot přicházejících z gyroskopu. Příkaz by se měl používat, jen když kvadrikoptéra leží na vodorovné ploše, protože použití při náklonu by ovlivnilo stabilizaci letu.

Příklad: **AT*FTRIM=45\n** *(vynuluje hodnoty z gyroskopu)*

5.6 Příkaz AT*COMWDG

Syntaxe: **AT*COMWDG={0}\n**

{0} – číslo sekvence

Jediný z příkazů, u kterého nezáleží, zda číslo sekvence je větší než u předcházejícího. Tento příkaz totiž nuluje číslo sekvence. Následující příkaz musí mít číslo sekvence opět vyšší než tento (tzn. 2). Příkazy do kvadrikoptéry musí chodit nejpozději jednou za 2 s (viz. 5.1). Tato doba je na plynulé ovládání relativně dlouhá. Ve výsledném programu této práce je interval mezi příkazy okolo 200 ms. Při létání číslo sekvence tím pádem docela narůstá a je nutné jej jednou za čas vynulovat a začít počítat znovu.

Protože tento příkaz nijak neovlivňuje let, nebo nastavení kvadrikoptéry je vhodné jej využít na udržení komunikace, když není potřeba posílat žádný jiný příkaz.

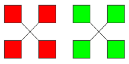
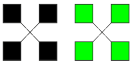
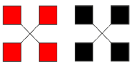
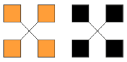
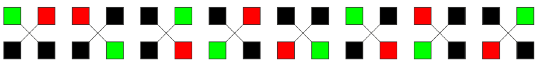
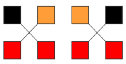
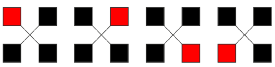
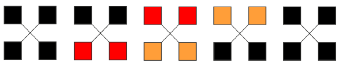
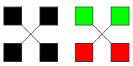
Příklad: **AT*COMWDG=1\n** *(vynuluje počítání sekvence)*

5.7 Příkaz AT*LED

Syntaxe: **AT*LED={0},{1},{2},{3}\n**

{0} – číslo sekvence
{1} – sekvence blikání animací, která se spustí
{2} – frekvence animace [Hz]
{3} – doba, po kterou bude animace probíhat [s]

Tímto příkazem jsme schopni rozblikat signalizační LED, které jsou umístěny vždy u každého motoru. Diody můžou svítit červenou, nebo zelenou barvou. Kombinací obou barev získáme oranžovou. Do režimu blikání zasahovat nemůžeme, je možné pouze spouštět předem definované sekvence, které jsou uvedené v tabulce 3. Sekvencí je celkem 21 a je možné je použít k signalizaci určitých stavů Ar.Drone, protože je to jediná možnost signalizace (kromě točících se vrtulí).

Číslo	Název	Pořadí stavů LED
0	Blikání zelená – červená	
1	Blikání zelená	
2	Blikání červená	
3	Blikání oranžová	
4	Had zelená - červená	
5	Oheň	
6	Standard	
7	Červená	
8	Zelená	
9	Červený had	
10	Nesvítí	
11	Střela vpravo	
12	Střela vlevo	
13	Dvojitá střela	
14	Přední levý zelený	
15	Přední pravý zelený	
16	Zadní pravý zelený	
17	Zadní levý zelený	
18	Levé zelené, pravé červené	
19	Levé červené, pravé zelené	
20	Standardní blikání	

*Tabulka 3: Sekvence blikání jednotlivých LED v příkazu AT*LED*

Další parametr příkazu je frekvence animace. Hodnota je v Hz a do příkazu ji zadáváme opět jako typ float. Postup při tvoření této hodnoty je obdobný jako v příkazu AT*PCMD (viz. 5.3), akorát nemusíme přepočítávat procenta z určité maximální hodnoty.

Zadáváme přímo počet Hertzů.

Poslední hodnotou v příkazu AT*LED je doba v sekundách po kterou bude animace probíhat. Hodnota musí být celé kladné číslo.

Příklad: **AT*LED=8,5,1073741824,3\n**
(blikání LED ve stylu Oheň, s frekvencí 2 Hz po dobu 3 s)

5.8 Příkaz AT*ANIM

Syntaxe: **AT*ANIM={0},{1},{2}\n**

{0} – číslo sekvence
{1} – číslo přednastaveného letového pohybu
{2} – doba, po kterou bude sekvence probíhat [s]

Tento příkaz v běžném létání využití nejspíš nenajde. Jeho funkce spočívá v tom, že podle zvoleného parametru {1} Ar.Drone provede určitou sekvenci pohybů. Definovaných pohybů je celkem 16 a jsou uvedeny v následující tabulce 4.

Třetí vstupující parametr je, stejně jako u předchozího příkazu, doba v sekundách, po kterou se tento pohybu bude provádět.

Číslo	Definované pohyby
0	Let vpravo pod úhlem 30°
1	Let vlevo pod úhlem 30°
2	Let dozadu pod úhlem 30°
3	Let dopředu pod úhlem 30°
4	Let dopředu pod úhlem 20° a rotace kolem svislé osy ve směru hodinových ručiček
5	Let dopředu pod úhlem 20° a rotace kolem svislé osy proti směru hodinových ručiček
6	Otočení dokola
7	Otočení dokola a klesání
8	Střídavé točení kolem svislé osy na obě strany (třesení)
9	„Tanec“ kolem svislé osy
10	„Tanec“ v bočním směru
11	„Tanec“ v přímém směru
12	„Tanec“ ve svislém směru
13	Vlnění
14	Kombinace náklonu v přímém a bočním směru
15	Zdvojená kombinace náklonu v přímém a bočním směru

*Tabulka 4: Význam hodnot v parametru příkazu AT*ANIM*

Příklad: **AT*ANIM=4,3,5\n**
(let pod úhlem 30° po dobu 5 s)

Tato kapitola čerpá z zdroje [13]

6 NAVIGAČNÍ DATA

Jako navigační data jsou označeny pakety odesílané z Ar.Drone řídící aplikaci. V těchto paketech jsou obsaženy jednak informace o stavu kvadrikoptéry a hodnoty ze senzorů. Z těchto údajů jsme schopni zjistit aktuální letovou situaci, nebo předejít různým nečekaným situacím. Tato kapitola popisuje jak tyto data získat a jak je zpracovat, abychom získali požadované informace.

6.1 Posílání navigačních dat z Ar.Drone

Po zapnutí kvadrikoptéry a připojení klienta se ještě žádná data neposílají. Je nutné dodržet určité kroky k získání těchto dat.

Jak je uvedeno v kapitole 4.3, Ar.Drone posílá navigační data přes protokol UDP na portu 5554. Komunikaci na tomto portu aktivujeme tak, že na něj pošleme UDP paket, jehož datová část ponese tyto 4 byty: $00\ 00\ 00\ 01_{(16)}$. Tohle se většinou provádí na začátku komunikace, při inicializaci. Ar.Drone poté začne z tohoto portu odesílat pakety. V tuto chvíli datová část paketu moc užitečných informací ještě neobsahuje. Data obsahují akorát hlavičku, informaci o velikosti a část *Stav ARDrone*, která bude popsaná dále.

Pro získání podrobnějších dat musíme toto aktivovat příkazem AT*CONFIG s parametry: "general:navdata_demo" a "TRUE" jak je popsáno v kapitole 5.4. Tento příkaz se už ale neodesílá na port 5554, ale na port 5556 jako standardní příkaz. Tím se aktivuje odesílání plnohodnotných navigačních dat a my je můžeme začít přijímat, číst a zpracovávat.

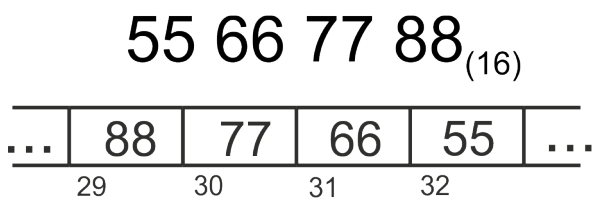
6.2 Formát dat

Než se dostaneme přímo ke konkrétním hodnotám obsaženým v paketech odeslaných z Ar.Drone, je nutné si ujasnit způsob, jakým jsou tyto data poskládána.

6.2.1 Little endian a big endian

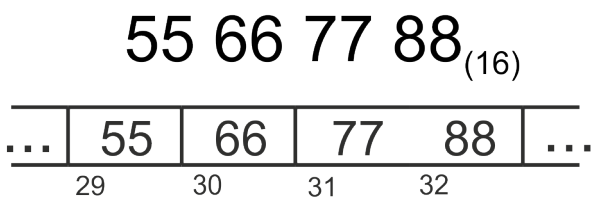
Existují dva způsoby, jakými se ukládají data do paměti. Používanější se nazývá *little endian*, méně používaný *big endian*.

Little endian – je používanější díky rozšířenosti procesorů Intel, které jsou založeny na tomto způsobu. Princip je takový, že byte s nejnižším významem je uložený na nejnižší adresu. Na obrázku 30 je zobrazeno uložení 32bitového čísla do paměti na adresu 29 tímto způsobem.[27] [28]



Obr. 30: Uložení hodnoty způsobem Little endian

Big endian – je méně používaný způsob. Využívají jej například mikroprocesory Motorla. Způsob ukládání dat je tu opačný než v předchozím případě. Nejméně významný byte je uložen na nejnižší adresu. Na obrázku 31 je opět příklad uložení stejného čísla, akorát způsobem *big endian*. [27] [28]



Obr. 31: Uložení hodnoty způsobem Big endian

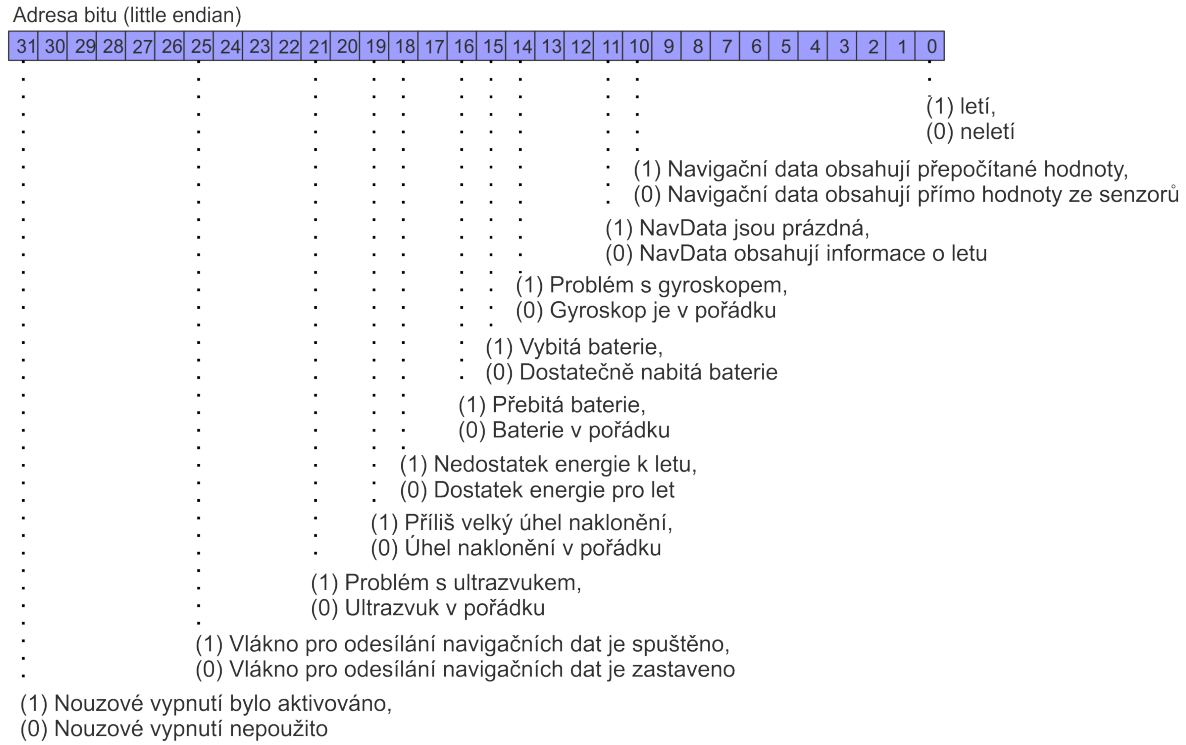
6.2.2 Struktura navigačních dat

Po předchozím ujasnění pojmů můžeme uvést, že navigační data, která přichází z kvadrikoptéry Ar.Drone jsou poskládány způsobem *Little endian*. Pod obrázkem 32 jsou popsány nejdůležitější údaje, které se z těchto dat dají vyčíst.

Obr. 32: Rozdělení navigačních dat

Hlava – těmito čtyřmi byty navigační data vždy začínají. Tuto část obsahují navigační data vždy, i když jsou prázdná (viz. 6.2). Jedná se o 32bitové číslo, které má vždy tuto hodnotu: $55\ 66\ 77\ 88_{(16)}$. V paketu, který je uložen podle způsobu Little endian jsou data uložena opačně ($88\ 77\ 66\ 55_{(16)}$).

Stav Ar.Drone – tuto část navigační data také obsahují vždy. Je to opět 32bitové číslo. Každý bit v tomto čísle má určitý význam. Zde na obrázku 33 jsou uvedeny jen ty, které mají pro tuto práci význam. Kompletní popsání nalezneme v příloze 1 (v angličtině).



Obr. 33: Význam bitů v bytu Stav Ar.Dorne

Sekvence – pořadové číslo paketu navigačních dat. Protože jsou pakety posílány protokolem UDP, není zaručeno mimo jiné pořadí doručení (viz. 4.2). Tyto 4 byty reprezentují celé kladné číslo.

Stav – dává nám aktuální informace o tom, co Ar.Drone právě provádí za příkazy. Jedná se opět o 4 byty, které představují celé kladné číslo. Stavy, které můžou nastat, jsou uvedeny v následující tabulce 5.

Číslo	Činnost Ar.Drone
0	Výchozí hodnota
1	Inicializace
2	Ar.Drone leží na zemi (připraveno vzlétnout, po přistání)
3	Ar.Drone právě letí některým směrem.
4	Ar.Drone je ve vzduchu a setrvává na místě
5	Testovací hodnota
6	Vzlétání
7	Let na určenou pozici
8	Pristávání

Tabulka 5: Stavy Ar.Drone z navigačních dat

Baterie – jedná se o 4 bytové celé kladné číslo, které nám dává informaci o hodnotě

nabití akumulátoru. Tato hodnota je v procentech z celkové kapacity. Při nízké hodnotě nabití Ar.Drone nevzlétne, ale funguje systém, signalizační diody a senzory. Při ještě nižší hodnotě systém i ostatní funkce běží, ale při inicializaci se kvadrikoptéra vypne.

Úhly pitch, roll a yaw – tyto tři úhly jsou každý reprezentován 32bitovým číslem, které představuje hodnotu float. Tato hodnota je u úhlů pitch (kolem osy y) a roll (kolem osy x) z intervalu -90° až 90°. Úhel yaw (kolem osy z) nabývá hodnot od -180° do 180°.

Rychlosti X, Y, Z – rychlosti ve třech souřadných osách. Opět je každá reprezentovaná 32bitovým číslem, které po převedení představuje hodnotu s plovoucí desetinnou čárkou float.

Tato kapitola čerpá ze zdroje [13]

7 PROGRAM NA OVLÁDÁNÍ

Tato kapitola se zabývá popisem programu, který byl v rámci práce vytvořen. Uvádí rozdělení zdrojového kódu do jednotlivých tříd a jejich propojení. Popisuje ovládání programu a jednotlivé prvky v uživatelském rozhranní programu.

Pro tvorbu programu bylo použito programovacího jazyku C#. Pro psaní a kompilaci kódu bylo využíváno Microsoft Visual Studio 2010. C# je objektově orientovaný jazyk a toho bylo při psaní kódu využito. Program je rozdělen do několika tříd. Základní vlastnosti a metody některých tříd jsou popsány v této kapitole.

Program bylo nutné, kvůli síťové komunikaci, rozdělit na tři současně běžící vlákna. V další části této kapitoly jsou popsány činnosti jednotlivých vláken.

7.1 Třídy programu

7.1.1 GlobPrm

Jedná se o statickou třídu, obsahující statické vlastnosti. Statická vlastnost zajistí, že je hodnota této vlastnosti uložena v paměti pouze jednou a pokud ji změníme na jenom místě programu a přistupujeme k ní z jiné části, získáme již změněnou hodnotu. Vlastnosti této třídy by se daly nazvat *globálními proměnnými* programu.

Tyto proměnné jsou v programu využívány hlavně pro sdílení dat mezi třídami a metodami, které používají různá vlákna. Dále jsou používány pro kontroly již proběhlých procesů (např. inicializace) a pro kontrolu spuštěných vláken.

C*ekajiciPrikaz* je proměnná, do které se uloží vygenerovaný příkaz na základě požadavku uživatele. Přistupuje k ní metoda, která se stará o odesílání příkazů. Může se stát, že posloupnost příkazů je generovaná rychleji, než se z této proměnné příkazy načítají a odesílají. Příkazy jsou přepisovány a tím pádem se může stát, že některý příkaz se neodešle. Tuto funkci má program záměrně, protože metoda odesílající data proměnnou kontroluje relativně často, a kdyby uživatel tak rychle měnil příkazy, bylo by to nejspíš v situaci (např. nebezpečí střetu s překážkou), kdy by tuto změnu požadoval skoro okamžitě a je důležité, aby kvadrikoptéra zareagovala ihned. Ne až po nějaké době, kdy se odešlou všechny příkazy z bufferu, které navíc můžou obsahovat povely ještě z doby správného letu.

NavData je statická proměnná pracující na podobném principu, jako *CekajiciPrikaz*. Vlákno pracující s navigačními daty je po zpravování uloží do této proměnné a začne zpracovávat další data, kterými po zpracování přepíše data v této proměnné. K proměnné přistupují metody, které tyto data využívají ke svým činnostem. Myšlenka je opět taková, že i když se nestihne přečíst každá nová data v této proměnné uložená, nic se neděje, protože při příštím přístupu tam budou data nová. Sice už se nemusí jednat o stejný paket, ale data se aktualizují tak rychle, že rozdíly mezi několika nejbližšími sousedními pakety jsou nepozorovatelné i při seberychlejším letu.

7.1.2 Commands

Tato třída slouží pro vytváření struktur příkazů. Příkazy vystupující z metod této třídy mají formu textového řetězce, který neobsahuje žádné konkrétní hodnoty příkazů. Prakticky vypadají jako syntaxe příkazů uvedených v této práci v kapitole 5. Hodnoty se do těchto příkazů doplňují až v dalších třídách, které používají metody z této třídy.

S touto třídou úzce souvisí i výčtové typy uvedené v tomto souboru, které se týkají zejména LED animací a příkazu AT*REF. Je pohodlnější pracovat s názvem LED animace, než s pořadovým číslem, stejně jako s povelem např. *Přistát* místo čísla 290717696.

7.1.3 Control

Tato třída používá metody třídy *Commands* a do generovaných struktur příkazů doplňuje většinu hodnot podle požadavků získaných od uživatele. Do příkazu už chybí doplnit akorát číslo sekvence, které se doplňuje až těsně před odesláním. Příkaz je v tomto stavu uložen do globální proměnné, kde si jej může vyzvednout jiná metoda.

Tato třída obsahuje i metodu, která se spouští pouze na začátku, při připojování kvadrikoptéry. Tato metoda obsahuje přesně danou posloupnost příkazů, které se postupně odesílají a zajišťuje tak inicializaci komunikace a nastavení kvadrikoptéry. Obsahuje mimo jiné příkaz na nastavení maximální letové výšky, příkaz pro vynulování gyroskopu, sled příkazů nutných pro spuštění navigačních dat. Na závěr je použita LED animace pro signalizaci, že inicializace proběhla úspěšně.

7.1.4 NavData

Tato třída se kompletně stará o zpracování a reprezentaci navigačních dat. Metoda, z jiné třídy, pověřená příjmem dat, předá pole bytu do konstruktoru této třídy. Voláním metod v této třídě se pole bytů rozdělí do vlastností této třídy, ke kterým je poté možné přistupovat. Třída obsahuje jednak metody pro rozdělení pole a zpracování hodnot podle jejich datových typů. Ve třídě jsou také metody na zpracování 32bitových hodnot *Stav Ar.Drone* a *Stav* (viz. 6.2.2).

7.1.5 Communication

Tato třída se stará o hlavní provoz programu. Obsahuje smyčky, běžící ve vláknech pro odesílání příkazů a přijímání navigačních dat. V této třídě také najdeme metody pro práci s číslem sekvence. Jsou volány až těsně před odesláním příkazu, aby bylo opravdu zajištěno, že toto číslo bude jenom růst. Protože tato třída zajišťuje veškerou síťovou komunikaci, jsou v ní definovány jednotlivé porty pro přijímání a odesílání dat a také IP adresa Ar.Drone.

Pro odesílání dat je použita metoda `BeginSendTo`. Její syntaxe vypadá následovně:

```
Socket.BeginSendTo(byte[] buffer,
                   int offset,
                   int size,
                   SocketFlags socketFlags,
                   EndPoint remoteEP,
                   AsyncCallback callback,
                   Object state)
```

Kde:

- `buffer` – pole bytů obsahující data k odeslání
- `offset` – místo v bufferu, od kterého se začne odesílat
- `size` – počet bitů, které se odešlou
- `socketFlags` – chování soketu
- `remoteEP` – koncový bod příjemce (IP a port)
- `callback` – fuknce, která se zavolá po odeslání dat
- `state` – informace funkci callback

Funkce, která se zavolá po odeslání dat, v tomto případě obsahuje pouze zavolání metody na uzavření probíhajícího spojení.

Pro příjem dat je použita funkce `BeginReceiveFrom`, jejíž syntaxe je taková:

```
Socket.BeginReceiveFrom(byte[] buffer,
                        int offset,
```

```
                                        int size,
                                        SocketFlags socketFlags,
                                        ref EndPoint remoteEP,
                                        AsyncCallback callback,
                                        Object state)
```

Kde:

- `buffer` – pole bytů, do kterého se uloží přijatá data
- `offset` – místo v bufferu, od kterého se začne ukládat
- `size` – počet bitů, které se přijmou
- `socketFlags` – chování soketu
- `remoteEP` – koncový bod ze kterého se odesílá (IP a port)
- `callback` – funkce, která se zavolá po přijmutí dat
- `state` – informace funkci callback

Funkce pro příjem dat se používá k přijímání navigačních dat. Po přijetí těchto dat je zavolaná funkce `callback`, která uzavře spojení a do globální proměnné n*avData* uloží instanci třídy *NavData*, ve které jsou již data zpracovány a naplněny vlastnosti v této instanci. [29]

7.1.6 Gyroskop, Natoceni, Vyskomer

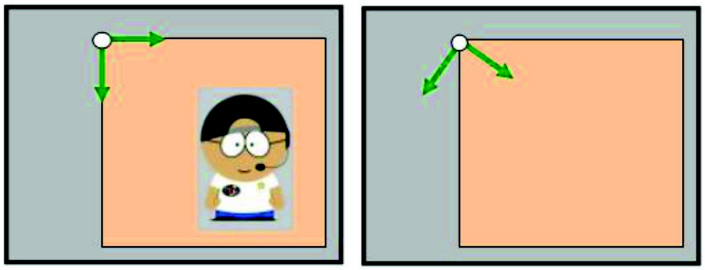
Tyto třídy jsou používány v druhém okně aplikace. Zpracovávají údaje z navigačních dat a na jejich základě vykreslují ve formuláři ukazatele vypadající jako „budíky“ ve skutečném letadle.

Každá z těchto tříd má stejnou strukturu. V konstruktoru se nejdříve vloží instance bitmap do proměnných. V další části se podle potřeby nastaví průhlednost některých barev v bitmapách. Konstruktor končí naplněním proměnných bodů, které jsou důležité pro rotaci jednotlivých bitmap. Další metoda, kterou mají všechny třídy, je metoda pro překreslení bitmapy podle vstupních hodnot.

Metoda pro překreslení obsahuje nejprve výpočet měřítka. To je dáno poměrem mezi rozměrem bitmapy a rozměrem plochy, ve které budeme chtít výslednou grafiku vykreslit. V dalším kroku si vytvoříme z bitmapy grafiku typu `Graphics` se kterou budeme moci pracovat. V dalších krocích se postupně vykreslují jednotlivé bitmapy, ke kterým se výsledná grafika skládá. Některé bitmapy je potřeba natočit a posunout, aby působily pohyblivým dojmem. Jedná se o různé ukazatele jako ručičky, nebo číslice. Otočení bitmapy okolo daného bodu je vysvětleno dále. Nakonec se všechny bitmapy spojí do jedné výsledné bitmapy a to je výstupní hodnota této metody.

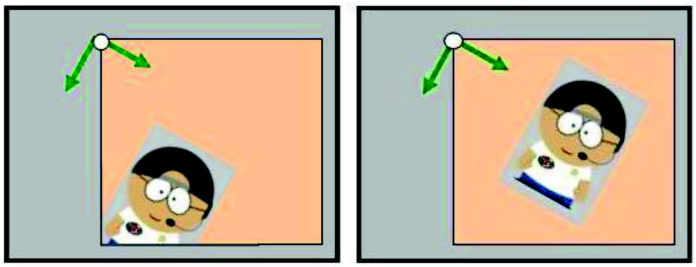
Otáčení bitmapy okolo určitého bodu

Tento pohyb není úplně tak snadný jak by se mohlo na první pohled zdát. Na rotování souřadného systému grafiky v C# je samozřejmě funkce. Jmenuje se `RotateTransform` a v našem případě je použitelná funkce s jedním vstupním parametrem datového typu float, který představuje úhel, o který se má souřadný systém natočit. Bohužel otáčí se kolem přesně daného bodu. Tím bodem je počátek souřadného systému, který leží v levém horním rohu grafiky. Kladná část osy *X* směřuje doprava, kladná část osy *Y* směřuje dolů. Natočení je znázorněno na následujících obrázcích 34 a 35.



Obr. 34: Rotace kolem počátku s. s. (v druhé části je postava mimo viditelnou oblast) [31]

Po natočení souřadného systému můžeme pomocí funkce DrawImage vykreslit bitmapu, která bude ve výsledku natočená jako souřadný systém. Funkce DrawImage je 30x přetížená. Nás zajímá zejména volání funkce s parametry tohoto typu a v tomto pořadí: `Image image, int x, int y, int width, int height`. První parametr požaduje obrázek (bitmapu), který se bude vykreslovat. Druhý a třetí parametr jsou souřadnice nulového bodu obrázku. Poslední dva parametry určují rozměry výsledného obrázku. Pro nás jsou důležité právě druhý a třetí parametr, pomocí nichž můžeme vykreslení bitmapy posunout. Rotace potom probíhá ve dvou krocích: natočení souřadného systému a vykreslení posunuté bitmapy. [31]



Obr. 35: Posunutí natočeného obrazu [31]

7.2 Vlákna programu

Vlákna jsou při běhu programů užitečná, protože nám zajistí fungování například okna programu, i když bude třeba provádět náročný výpočet, nebo čekat na připojení určité síťové komunikace. Vlákno je možné si představit jako další běžící proces, který ale může s hlavním procesem sdílet data a části programu. Systém poté rozděluje čas procesoru mezi tyto vlákna. Jakmile vlákno dostane k dispozici procesor, začne provádět svou práci. Po ukončení časového intervalu (v řádech 10 ms) se práce vlákna uloží a výpočetní výkon procesoru se přidělí dalšímu vláknu. Tak je například možné zobrazovat průběh výpočtu, nebo provádět další činnosti, aniž by byla aplikace po dobu výpočtu „zamrzlá“. V dnešní době, kdy jsou běžné vícejádrové procesory se přidělují výpočetní výkony jednotlivých jader. Pokud programátor neuvede jinak, má program jedno hlavní vlákno, ve kterém běží vše. Z hlavního vlákna potom může programátor spouštět vlákna další.

Jak již bylo uvedeno dříve v této kapitole, při plném používání programu běží souběžně 3 vlákna. Běh dalších dvou vláken se spouští při inicializaci připojení a nastavení kvadrikoptéry.

7.2.1 Hlavní vlákno

V tomto vlákně běží oba formuláře aplikace a jejich vykreslování. Přes toto vlákno se celá aplikace ovládá a generují se příkazy pro Ar.Drone. Tyto příkazy se ukládají do globální proměnné (viz. 7.1.1). Také se v určitých intervalech (50 ms) obnovují hodnoty, zobrazující na formuláři letové informace. V druhém formuláři na tomto vlákně zároveň běží

zobrazování a překreslování grafických ukazatelů naklonění, natočení, výšky a nabití baterie (také jednou za 50 ms).

7.2.2 Vlákno CMD

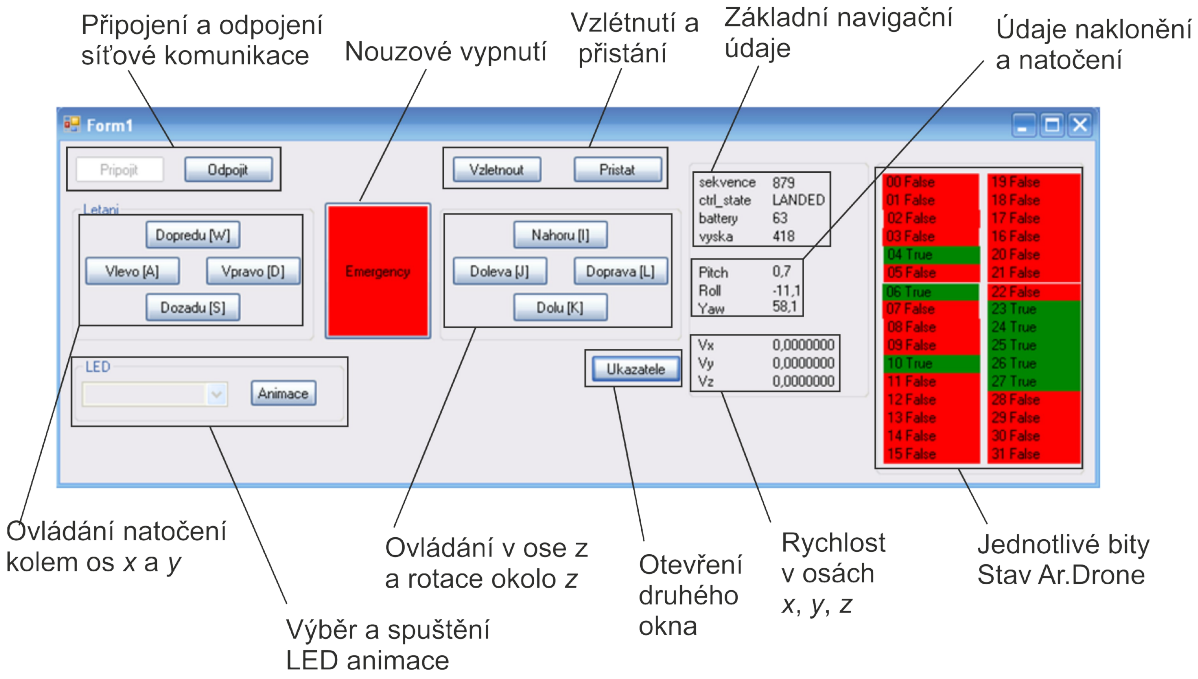
V tomto vlákně běží „nekonečná“ smyčka `while`, kontrolující svou globální proměnnou, přes kterou je možné smyčku v případě potřeby zastavit. Tato smyčka přistupuje do globální proměnné, ve které by měl být uložený příkaz připravený k odeslání do Ar.Drone. Ověří, zda tam příkaz opravdu je, odešle jej a obsah globální proměnné vymaže. Pokud není žádný příkaz čekající na odeslání, odesílá se příkaz na nulování čísla sekvence (AT*COMWDG). Tím se udržuje aktivní komunikace, mezi aplikací a kvadrikoptérou Ar.Drone. Protože stačí příkazy odesílat jednou za 2 s, je tohle vlákno vždy po každém cyklu na určitou dobu pozastaveno, aby se ušetřil výpočetní výkon a vznikl prostor pro běh dalších vláken.

7.2.3 Vlákno NAV

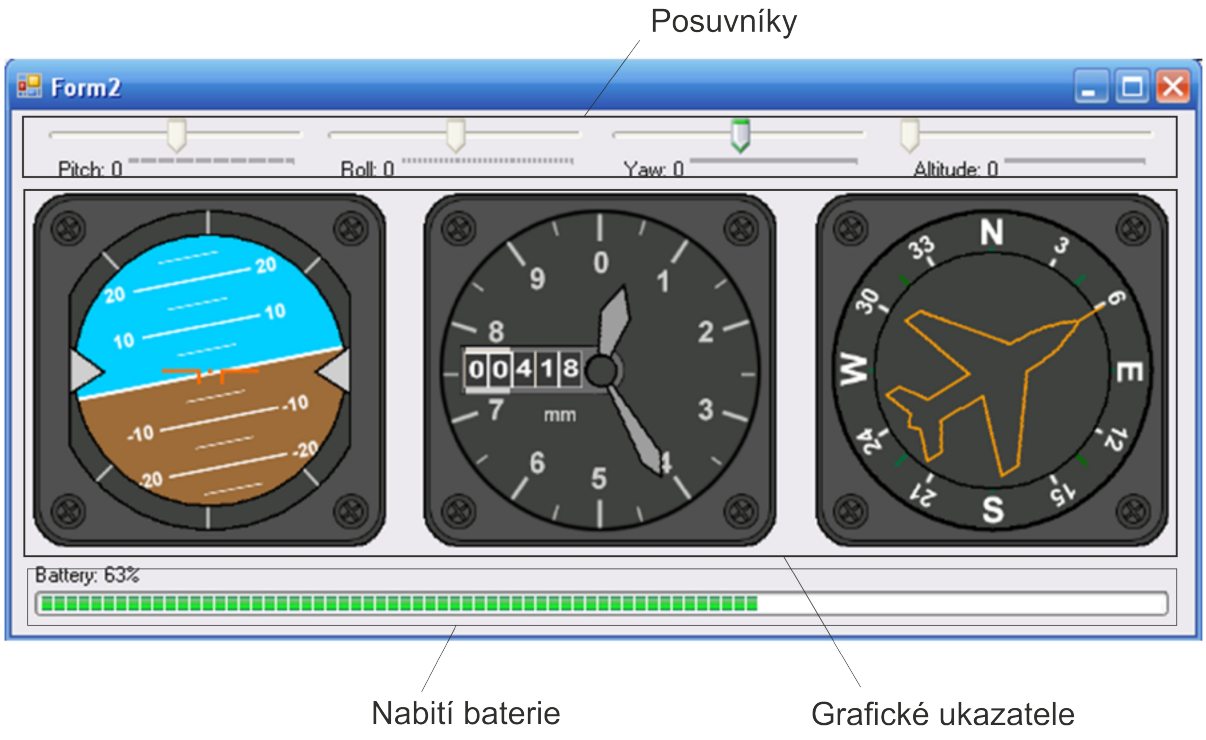
Toto vlákno se stará o příjem a zpracování navigačních dat. Opět běží v kontrolované „nekonečné“ smyčce `while`. Zde se data voláním určitých metod přijmou, zpracují, rozdělí a převedou na konkrétní hodnoty a uloží se do globální proměnné. Poté se proměnné určené k příjmu dat vymažou a připraví na další průběh cyklu. Vlákno se na nějakou dobu uspí, aby se opět šetřil výpočetní výkon. Ke globální proměnné, s již uloženými navigačními daty, může přistupovat kterékoliv vlákno – zejména hlavní, načítat z ní hodnoty a zobrazovat na formulářích, případně s nimi jinak pracovat.

7.3 Používání programu

Program je rozdělen na dvě oddělená okna. Hlavní okno (viz. obr. 36) slouží pro ovládání kvadrikoptéry. Obsahuje tlačítka pro připojení a odpojení síťové komunikace, vzlétnutí a přistání a dále se v něm nachází tlačítka pro ovládání směrů letu a natočení. V pravé části okna jsou signalizační a informační prvky, které na základě navigačních dat zobrazují určené hodnoty.



Obr. 36: Hlavní okno programu



Obr. 37: Okno programu s ukazateli informací o letu

Druhé okno aplikace (viz. obr. 37) obsahuje grafické prvky, které simulují ukazatele z letadel. Tyto prvky jsou zobrazeny v komponentách zvaných *PictureBox*. *PictureBox* je určený k vykreslování obrázků a grafiky. Program obsahuje tři *PictureBoxy*. První simuluje ukazatel umělého horizontu. Je z něj možné pohodlně odečítat náklon jak kolem osy *y*, tak současně náklon kolem osy *x* (viz. obr. 38).

Obr. 38: Umělý horizont

Dalším ukazatelem je grafika představující výškoměr (viz. obr. 39). Tento ukazatel zobrazuje výšku dvěma způsoby. Prvním je zobrazení pomocí ručiček, které se otáčí okolo středu a na stupnici na obvodu je možné odečítat hodnoty. Dalším zobrazením výšky jsou pohyblivé číslice, které se jako kdyby otáčejí na válečcích (použito například u tachometrů v autech, nebo dříve jako počítadla na benzínových pumpách).

Obr. 39: Výškoměr

Posledním grafickým ukazatelem v tomto okně je ukazatel natočení kolem osy *z* (viz. obr. 40). Tento údaj se může zdát jako kompas, ale jsou to pouze údaje z gyroskopu. Nula, tedy ukazatel směřuje nahoru, je v poloze, ve které byla kvadrikoptéra zapnuta. Třída *Natoceni* obsahuje funkci pro „vynulování“ této hodnoty. Ve skutečnosti se pouze zjistí aktuální hodnota natočení a odečítáním této hodnoty od nové se docílí nastavení ukazatele na nulovou hodnotu. Tato funkce je užitečná, neboť kvadrikoptéru většinou nezapínáme v poloze, ve které budeme chtít vzlétnout a tak si před letem, nebo během letu můžeme tuto hodnotu vynulovat podle sebe.

Obr. 40: Ukazatel natočení

V horní části druhého okna programu jsou posuvníky, které slouží pro simulování hodnot přicházejících z Ar.Drone. Tyto posuvníky jsou zejména kvůli testování vykreslování jednotlivých grafik. Při připojení kvadrikoptéry a zapnutí posílání navigačních se tyto posuvníky deaktivují, takže s nimi nelze ovlivňovat zobrazovaná data.

Okno obsahuje v dolní části ještě ukazatel nabití baterie. Hodnota je uvedena jednak číselně nad ukazatelem a také znázorněna graficky, jako vyplnění signalizačního pruhu odpovídající procentům nabití baterie.

7.3.1 Konfigurace počítače

Program byl tvořen a testován na notebooku s operačním systémem Microsoft Windows XP obsahující Service Pack 3. Jak již bylo uvedeno, na psaní a kompilaci kódu bylo použit software Microsoft Visual Studio 2010. Program byl psán v programovacím jazyce C#.

Notebook má jednojádrový procesor Intel Pentium M taktovaný na frekvenci 1,73 GHz, který pro výsledný program naprosto dostačuje. Další z komponent důležitých pro funkci programu je přítomnost WiFi karty. Před připojením programu ke kvadrikoptéře je nutné zkontrolovat, zda je počítač připojen k přístupovému bodu Ar.Drone. Ar.Drone používaný pro tuto práci ve výchozí konfiguraci vytváří WiFi přístupový bod s názvem *ardrone_048993*. Nejjednodušší ověření tohoto připojení a funkce komunikace je příkazem *ping* z příkazové řádky systému.

8 ZÁVĚR

Práce se ve své první části zabývá rešerší na téma UAV. Je možné zde dohledat základní rozdělení těchto zařízení a příklady z jednotlivých kategorií.

Dalším cílem práce bylo prostudování možností řídícího systému platformy Ar.Drone. Touto problematikou se podrobně zabývá kapitola 3, kde je popsáno ujasnění pojmů, princip fungování kvadrikoptér a podrobný popis jednotlivých součástí kvadrikoptéry Ar.Drone. Právě princip létání, popisovaný v kapitole 3.2 je velmi zajímavý, protože v praxi se často nevyskytuje. Za zmínku stojí i podkapitoly 3.3.6 a 3.3.7, které jsou zaměřeny na senzory, které Ar.Drone využívá.

Pro splnění dalšího cíle práce bylo nutné ujasnění a upřesnění pojmů síťové komunikace v rámci protokolů UDP a TCP. Jsou zde popsány používané protokoly a specifikovány porty, na kterých komunikace probíhá. Dále tato kapitola obsahuje přehledný popis datového paketu, který je sítí přenášen. V poslední části kapitoly je stručně popsaná obsluha programu WireShark. Tento program velkou mírou přispěl k pochopení komunikace v síti a velmi usnadnil porozumění principů v následujících kapitolách tím, že nám umožnil zkoumat jednotlivé pakety pohybující se v síti.

Kapitoly 5 a 6 se zabývají odesíláním a příjmem dat z ovládacího zařízení. Odesílají se příkazy, které mají přesně daný formát. Ovládací příkazy jsou velmi podrobně popsány v kapitole 5. V první části kapitoly jsou popsány jednotlivé úseky příkazu. Druhá část se věnuje rozboru jednotlivých příkazů a poskytuje rady pro jejich používání. Následující kapitola pak popisuje způsob příjmu dat z kvadrikoptéry. Jsou zde uvedeny kroky nutné pro začátek komunikace a nastavení kvadrikoptéry tak, aby odesílala požadované data. Dále tato kapitola popisuje strukturu těchto dat a způsob jak z nich získat hodnoty, které jsou pro provoz stěžejní.

Poslední 7. kapitola je zaměřena na software, který byl v rámci této práce vytvořen. Tato kapitola popisuje jednotlivé třídy a základní principy metod daných tříd. Pro vývoj softwaru bylo využito poznatků z předchozích kapitol a aplikace generuje a odesílá příkazy nutné pro provoz kvadrikoptéry. Aplikace je dále schopna přijímat data z kvadrikoptéry a přijatá data zobrazovat v uživatelském rozhraní.

Další vývoj do budoucna by se mohl ubírat směrem k využití kamer umístěných na kvadrikoptéře, nebo použití Ar.Drone v kombinaci s GPS přijímačem.

Přínosem této práce je souhrnné uspořádání informací užitečných pro další vývoj aplikací využívající platformu Ar.Drone, která je dostupná v jedné z laboratoří na ústavu automatizace a informatiky VUT FSI.

SEZNAM POUŽITÉ LITERATURY

[1] The UAV. *The UAV* [online]. [cit. 2012-05-21]. Dostupné z: http://www.theuav.com/

[2] Bezpilotní bojové letouny. *Military* [online]. © 2012 [cit. 2012-05-21]. Dostupné z: http://www.military.cz/usa/air/in_service/unmanned/uvod/ucav.htm

[3] BQM-34 Firebee High Performance Aerial Target System. *Northrop Grumman* [online]. © 2012 [cit. 2012-05-21]. Dostupné z: http://www.as.northropgrumman.com/products/targets_bqm34/index.html

[4] Firebee BQM-34A Drone. *White Sands Missile Range Museum* [online]. © 2010 [cit. 2012-05-22]. Dostupné z: http://www.wsmr-history.org/BQM-34A.htm

[5] The UAV Drone. *Hooked On RC Airplanes* [online]. © 2007-2012 [cit. 2012-05-22]. Dostupné z: http://www.hooked-on-rc-airplanes.com/uav-drone.html#1

[6] Britain's RAF Requests 10 MQ-9 Reapers for over $1B. *Defense lindustry daily*[online]. 12.12. [cit. 2012-05-22]. Dostupné z: http://www.defenseindustrydaily.com/britain-requests-10-mq-9-reapers-for-over-1b-04536/

[7] RQ-4 Global Hawk. *Vojsko.net* [online]. 3.3.2005 [cit. 2012-05-22]. Dostupné z: http://www.vojsko.net/index.php?clanek=letecka/uav/rq4

[8] RQ-4 Global Hawk. *U. S. Air Force* [online]. 27.1.2012 [cit. 2012-05-22]. Dostupné z: http://www.af.mil/information/factsheets/factsheet.asp?id=13225

[9] NASA Pathfinder Plus- Solar Powered Flight. *Fiddlers green* [online]. © 1994-2012 [cit. 2012-05-22]. Dostupné z: http://www.fiddlersgreen.net/models/aircraft/NASA-Pathfinder.html

[10] Parrot AR.Drone RC Helicopter Review. *Bit-Tech* [online]. 18.6.2010 [cit. 2012-05-22]. Dostupné z: http://www.bit-tech.net/bits/2010/07/18/parrot-ar-drone-review/1

[11] AR.Drone.cz - První kvadrikoptéra ovládaná pomocí iPhone/iPad/iPod Touch. *AR.Drone.cz - První kvadrikoptéra ovládaná pomocí iPhone/iPad/iPod Touch* [online]. © 2010 [cit. 2012-05-22]. Dostupné z: http://www.parrot-ardrone.cz

[12] Produkty Parrot. *Parrot Products* [online]. 2010-2012 [cit. 2012-05-22]. Dostupné z: http://www.parrotklub.cz/index.php/cs/produkty-parrot

[13] A.R.Drone Developer Guide. In: *A.R.Drone Developer Guide* [online]. 2011 [cit. 2012-05-22]. Dostupné z: https://projects.ardrone.org/attachments/download/365/ARDrone_SDK_1_7_Developer_Guide.pdf

[14] MIAN, Ashfaq Ahmad; DAOBO, Wang. Modeling and Backstepping-based Nonlinear Control Strategy for a 6 DOF Quadrotor Helicopter. [online]. 2008, [cit. 2012-05-22]. Dostupný z: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B8H0X-4SSRK5P-B-1&_cdi=42506&_user=640830&_pii=S1000936108600345&_orig=search&_coverDate=06/30/2008&_sk=999789996&view=c&wchp=dGLbVlWzSkWA&md5=53a53ff6f7ca912ef71d958de889df75&ie=/sdarticle.pdf

[15] Li-Pol akumulátory. *Notebook.cz* [online]. 22.06.2011 [cit. 2012-05-22]. Dostupné z: http://notebook.cz/clanky/technologie/2011/Li-Pol-akumulatory

[16] Parrot AR.Drone Teardown. *IFixit* [online]. © 2012 [cit. 2012-05-22]. Dostupné z: http://www.ifixit.com/Teardown/Parrot-AR-Drone-Teardown/3984/1

[17] AR Drone Hardware. *Drone-RK* [online]. 6.8.2011 [cit. 2012-05-22]. Dostupné z: http://drone-rk.org/wiki/drone

[18] Devantech SRF05. *HVW Technologies* [online]. ©1997-2010 [cit. 2012-05-22]. Dostupné z: http://www.hvwtech.com/products_view.asp?ProductID=943

[19] Akcelerometry - integrované snímače od AD. *Automatizace.HW.cz* [online]. 6.2.2005 [cit. 2012-05-22]. Dostupné z: http://automatizace.hw.cz/clanek/2005020601

[20] Integrované MEMS GYROSKOPY. *Automatizace.HW.cz* [online]. 11.10.2009 [cit. 2012-05-22]. Dostupné z: http://automatizace.hw.cz/integrovane-mems-gyroskopy

[21] IDG-500 Integrated Dual-Axis Gyroscope. *InvenSense* [online]. © 2012 [cit. 2012-05-22]. Dostupné z: http://invensense.com/mems/gyro/idg500.html

[22] IPhone 4 Gyroscope Teardown. *IFixit* [online]. © 2012 [cit. 2012-05-22]. Dostupné z: http://www.ifixit.com/Teardown/iPhone-4-Gyroscope-Teardown/3156/1#.T7vuVeh1B-g

[23]Mobile Ad Hoc Networks. *Mobile Ad Hoc Networks* [online]. © 2012 [cit. 2012-05-22]. Dostupné z: http://www.ece.iupui.edu/~dskim/manet/
[24]FreeFlight Android. *Parrto Ar.Drone* [online]. © 2010 [cit. 2012-05-22]. Dostupné z: http://www.parrot-ardrone.cz/support-android
[25]Principy síťových architektur. *Počítačové sítě* [online]. 20.7.2000 [cit. 2012-05-22]. Dostupné z: http://www.gybon.cz/~rusek/vyuka/site/site003.html
[26]Norma IEEE 754 a příbuzní: formáty plovoucí řádové tečky. *Root.cz* [online]. 31.5.2006 [cit. 2012-05-23]. Dostupné z: http://www.root.cz/clanky/norma-ieee-754-a-pribuzni-formaty-plovouci-radove-tecky/
[27]Big a Little endian - způsoby ukládání dat v paměti počítačů. *Big a Little endian* [online]. 30.09.2009 [cit. 2012-05-22]. Dostupné z: http://jankoweb.tode.cz/blog/programovani/big-a-little-endian-zpusoby-ukladani-dat-v-pameti-pocitacu/
[28]Little Endian vs. Big Endian. *EArchiv.cz* [online]. © 2011 [cit. 2012-05-22]. Dostupné z: http://www.earchiv.cz/a92/a237c120.php3
[29]Socket.BeginReceiveFrom Method. *MSDN* [online]. © 2012 [cit. 2012-05-22]. Dostupné z: http://msdn.microsoft.com/cs-cz/library/system.net.sockets.socket.beginreceivefrom.aspx
[30]Socket.BeginSendTo Method. *MSDN* [online]. © 2012 [cit. 2012-05-22]. Dostupné z: http://msdn.microsoft.com/cs-cz/library/system.net.sockets.socket.beginsendto.aspx
[31]C# Avionic Instrument Controls. *The code project* [online]. 17.6.2008 [cit. 2012-05-22]. Dostupné z: http://www.codeproject.com/Articles/27411/C-Avionic-Instrument-Controls
[32]SDK 1.8 package. *ARDRONE open API platform* [online]. © 2006-2009 [cit. 2012-05-23]. Dostupné z: https://projects.ardrone.org/attachments/download/373/ARDrone_SDK_Version_1_8_20110726.tar.gz

SEZNAM PŘÍLOH

CD-R obsahující: Tento dokument ve formátu PDF
Archiv se zdrojovými kódy softwaru

Zobrazení významu všech bitů v bytu *Stav Ar.Drone*

PŘÍLOHY

Příloha 1: Význam bitů v *Stav Ar.Drone* [32]

```
31                                                              0
x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x x
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FLY MASK : (0) ardrone is landed, (1) ardrone is flying
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | VIDEO MASK : (0) video disable, (1) video enable
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | VISION MASK : (0) vision disable, (1) vision enable
| | | | | | | | | | | | | | | | | | | | | | | | | | | | CONTROL ALGO : (0) euler angles control, (1) angular speed control
| | | | | | | | | | | | | | | | | | | | | | | | | | | ALTITUDE CONTROL ALGO : (0) altitude control inactive (1) altitude control active
| | | | | | | | | | | | | | | | | | | | | | | | | | USER feedback : Start button state
| | | | | | | | | | | | | | | | | | | | | | | | | Control command ACK : (0) None, (1) one received
| | | | | | | | | | | | | | | | | | | | | | | | Trim command ACK : (0) None, (1) one received
| | | | | | | | | | | | | | | | | | | | | | | Trim running : (0) none, (1) running
| | | | | | | | | | | | | | | | | | | | | | Trim result : (0) failed, (1) succeeded
| | | | | | | | | | | | | | | | | | | | | Navdata demo : (0) All navdata, (1) only navdata demo
| | | | | | | | | | | | | | | | | | | | Navdata bootstrap : (0) options sent in all or demo mode, (1) no navdata options sent
| | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | Bit means that there's an hardware problem with gyrometers
| | | | | | | | | | | | | | | | VBat low : (1) too low, (0) Ok
| | | | | | | | | | | | | | | VBat high (US mad) : (1) too high, (0) Ok
| | | | | | | | | | | | | | Timer elapsed : (1) elapsed, (0) not elapsed
| | | | | | | | | | | | | Power : (0) Ok, (1) not enough to fly
| | | | | | | | | | | | Angles : (0) Ok, (1) out of range
| | | | | | | | | | | Wind : (0) Ok, (1) too much to fly
| | | | | | | | | | Ultrasonic sensor : (0) Ok, (1) deaf
| | | | | | | | | Cutout system detection : (0) Not detected, (1) detected
| | | | | | | | PIC Version number OK : (0) a bad version number, (1) version number is OK
| | | | | | | ATCodec thread ON : (0) thread OFF (1) thread ON
| | | | | | Navdata thread ON : (0) thread OFF (1) thread ON
| | | | | Video thread ON : (0) thread OFF (1) thread ON
| | | | Acquisition thread ON : (0) thread OFF (1) thread ON
| | | CTRL watchdog : (1) delay in control execution (> 5ms), (0) control is well scheduled  [Check frequency of control loop]
| | ADC Watchdog : (1) delay in uart2 dsr (> 5ms), (0) uart2 is good [Check frequency of uart2 dsr (com with adc)]
| Communication Watchdog : (1) com problem, (0) Com is ok // Check if we have an active connection with a client
Emergency landing : (0) no emergency, (1) emergency
```