

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

SKUPINA INTELIGENTNÍ A MOBILNÍ ROBOTIKY

**Technický manuál
řídící deska multicopter**

TOMÁŠ BÁČA

Obsah

| | | |
|----------|---|----------|
| 1 | Hardware | 1 |
| 1.1 | Letoun | 1 |
| 1.2 | Řídicí deska | 2 |
| 1.3 | Připojitelné externí moduly | 2 |
| 1.3.1 | FlightCTRL | 3 |
| 1.3.2 | Počítač Gumstix | 3 |
| 1.4 | Příprava před letem | 3 |
| 1.5 | Aktivace k letu | 3 |
| 1.6 | Deaktivace - vypnutí | 4 |
| 2 | Software | 5 |
| 2.1 | Struktura software | 5 |
| 2.1.1 | main.c | 5 |
| 2.1.2 | controllers.c | 5 |
| 2.1.3 | system.c | 5 |
| 2.1.4 | communication.c | 7 |
| 2.1.5 | config.h | 7 |
| 2.2 | Kompilace pro ATmega164p | 8 |
| 2.3 | Upload programu do ATmega164p | 8 |
| 2.4 | Logování letových dat do Gumstixe | 9 |
| 2.4.1 | Strana řídicí desky | 9 |
| 2.4.2 | Strana Gumstixe | 9 |

1 Hardware

1.1 Letoun

Letoun, jinak běžně nazývaný kvadrukoptera, je helikoptera se čtyřmi rotory. Listy mají pevný úhel náběhu a jejich tah se řídí změnu rychlosti jejich otáčení. Díky tomu je stroj konstrukčně jednoduchý a relativně odolný proti poškození (v porovnání s běžnou helikopterou). Z toho důvodu je vhodný pro experimentální použití. Stroj je napájen z Li-Poly (Lithium-Polymerová) baterie. Doba letu se pohybuje od 5 do 15 minut v závislosti na zatížení.

Samotný letoun je nestabilní systém vyžadující stabilizaci. O tu se stará výrobcem dodaná stabilizační deska FlightCTRL (Obrázek 6). Ta je osazena 3-osými MEMS gyroskopy a akcelerometry, ze kterých syntetizuje údaj o úhlu náklonu v ose Pitch a Roll. Ty potom používá pro stabilizaci letounu. Dále se deska stará o mixování vstupních signálů (výroba virtuálních vstupů) a řízení jednotlivých vrtulí.

Vstupními řídicími signály do desky jsou:

THROTTLE - Kolektivní tah všech rotorů v procentech jejich max. otáček

ELEVATOR - Náklon letounu v ose dopředu/dozadu (pitch, výškovka), úhlový rozměr

AILERON - Náklon letounu v ose doleva/doprava (roll, křídélka), úhlový rozměr

RUDDER - Rotace kolem svislé osy (yaw, směrovka), rozměr úhlové rychlosti

Vstupní signál se do FlightCTRL zavádí po jednom vodiči pomocí Pulze Pozici Modulu (PPM). V desce je stabilizace implementována formou **úhlového regulátoru**. Všechny tři osy (yaw, pitch, roll) jsou stabilizovány na nulový úhel. Před startem je vždy třeba provést kalibraci na rovné zemi, kde si jednotka nastaví referenční nulový úhel.

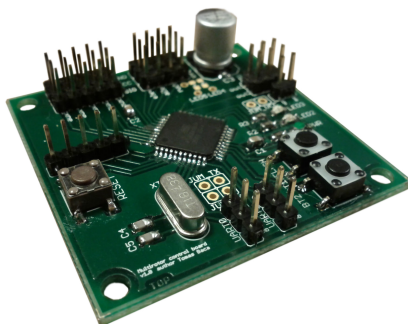
Srdcem FlightCTRL je mikrokontroler ATmega, software je napsán v C. Zdrojové kódy jsou k dispozici a je možnost je upravit a nahrát do kontroleru. Toho bylo využito pro získávání hodnot úhlů náklonu po seriové lince.



Obrázek 1: Stabilizační deska FlightCTRL

1.2 Řídicí deska

Řídicí deska plní funkci komunikačního uzlu mezi moduly a malé výpočetní jednotky pro další stabilizaci a řízení letounu. Je osazena mikrokontrolerem ATmega164p s 1kb RAM a 16kb ROM (pro program). Procesor kontroleru trpí absencí FPU (floating point unit). Ke komunikaci jsou zde 2 porty UART. Programování mikrokontroleru probíhá přes SPI.



Obrázek 2: Řídicí deska kvadrukopty

Deska má 9 vstupů pro Pulzně Šířkovou Modulaci (PWM). Použity jsou pro příjem signálů z RC přijímače. Dále jsou zde dva volitelné výstupy, z nichž jeden je použit pro PPM výstup do stabilizační desky FlightCTRL a druhý je volitelný.

Pro signalizaci jsou k dispozici 2 LED, žlutá a červená, třetí se dá případně připojit do volitelného výstupu.

Deska je osazena dvěma volitelnými tlačítky a jedním tlačítkem **reset**.

V minimální konfiguraci plní funkci konvertoru signálů z RC soupravy. Převádí PWM z přijímače na PPM pro FlightCTRL a umožňuje tedy ruční řízení letounu. Z důvodu bezpečnosti doporučuji vždy **zachovat majoritní vliv ručního řízení** a vyvíjené regulátory mixovat se signály z RC soupravy se značnou saturací. Zamezíte tím případné neovladatelnosti stroje. Více viz kapitola [2.1.2](#).

Deska je napájena z FlightCTRL pomocí komunikačního 3-žilové kabelu. Je to standardní modelářské uspořádání s **+VCC** na prostředním pinu a **GND** a **DATA** na krajních pinech. **GND** a **+VCC** jsou na celé desce propojené. Stejným způsobem se přenáší napájení do RC přijímače.

1.3 Připojitelné externí moduly

Zde uvedu další zařízení, pro která je v řídicí desce vytvořena komunikační podpora a dají se s nimi tedy relativně snadno pracovat.

1.3.1 FlightCTRL

V první řadě je to stabilizační deska FlightCTRL. V řídicí desce jsou funkce pro dekodování zpráv ve formátu Base64, které FlightCTRL posílá po seriové lince. Dají se tedy snadno přijímat údaje o úhlu náklonu helikoptéry a použít je dále pro řízení. Pro správnou funkčnost je však nutné mít ve FlightCTRL nahrán modifikovaný firmware, který automaticky posílá čerstvé údaje vždy, když jsou k dispozici.

1.3.2 Počítač Gumstix

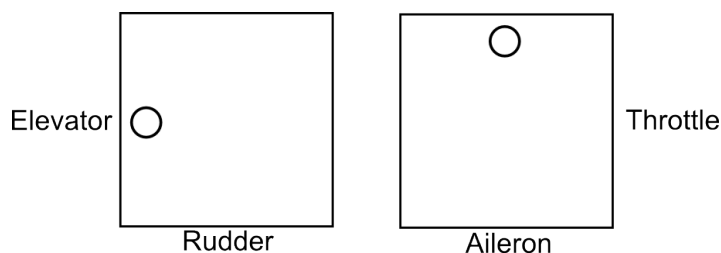
Tento modul se obvykle stará o rozpoznávání kruhového detekčního vzoru. Dále je používán pro logování dat během letu. Modul je tvořen malým počítačem s procesorem architektury ARM. Počítač disponuje wifi konektivitou, jedním UART ve funkci konzole (b.r. 115 200), druhým UART pro komunikaci s řídicí deskou (b.r. 57 600). Gumstix se automaticky připojuje na síť s SSID uKopter. V laboratoři je připraven router (bílý), který takovou síť vyrábí. IP Gumstixe je pevná - 192.168.1.103.

Připojení přes **SSH** k bílému Gumstix: `root@192.168.1.103`, heslo prázdné

O logování dat do Gumstixe se dočtete v kapitole [2.4](#).

1.4 Příprava před letem

Po připojení napájení je potřeba provést kalibrační rutinu, bez které letoun nevzlétne. Je potřeba letoun **postavit na vodorovnou zem** a poté nastavit páčky vysílače do následujících poloh:

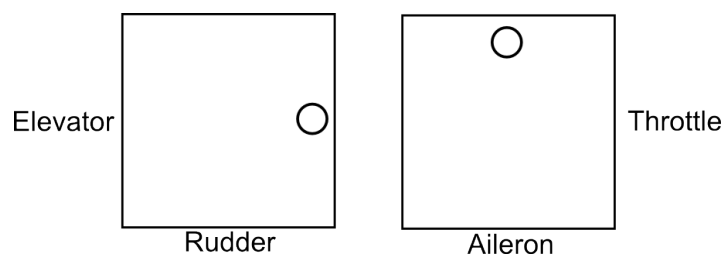


Obrázek 3: Kalibrace, část 1.

Poté je potřeba letounu říci, aby si data uložil do paměti, to se provádí obdobným "gestem":

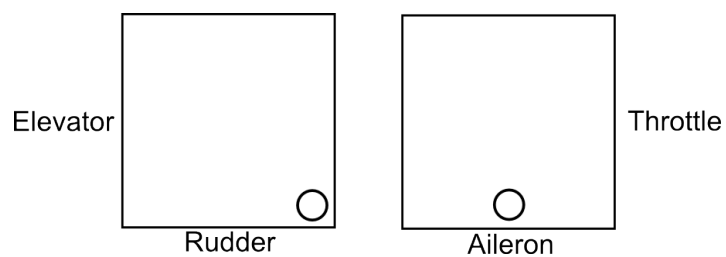
1.5 Aktivace k letu

Pokud je letoun nakalibrovaný, je potřeba ho aktivovat (lidově "naArmovat"). To se provádí obdobným gestem. Letoun poté roztočí vrtule na minimální otáčky a je připraven k



Obrázek 4: Kalibrace, část 2.

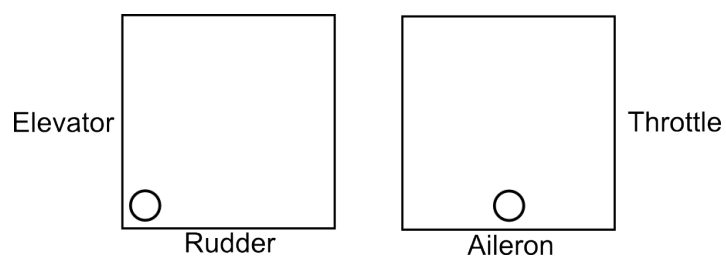
letu.



Obrázek 5: Armování letounu

1.6 Deaktivace - vypnutí

Velmi důležité gesto pro disarmování letounu. Bez něj se vrtule nevypnou. Je možné zapnout automatické volání gesta řídicí jednotkou. Např. v případě vypnutí plynu na minimum.



Obrázek 6: Disarmování letounu

2 Software

2.1 Struktura software

Software pro kontroler je napsán v programovacím jazyce C. Hlavním souborem je **main.c**. Zde je hlavní smyčka programu a předpis funkcí pro přerušení přerušeními procesoru.

Většinu času tráví procesor v nekonečné smyčce ve funkci **main()**. Zde čeká na asynchronní obsluhu komunikace, nebo periodické volání řídicích funkcí apod. Je velmi důležité, aby co možná všechny výpočty, zpracování a volání funkcí probíhaly voláním z funkce **main()**. Jinak bude procesor zablokovan (přerušení nebude vyvoláno, pokud procesor vykonává funkci jiného přerušení) a nebude zpracovávat komunikaci, což může vést k neovladatelnosti letounu.

Kód je strukturován pomocí podmínek preprocesoru. Pokud např. nebudete potřebovat kamerový modul s počítačem gumstix, lze v souboru **config.h** definovat **GUMSTIX_DATA_RECEIVE** na hodnotu **DISABLED**. Všechn kód a proměnné týkající se příjmu dat a volání funkcí, s tímto modulem spojených, bude vyřazeno z kompilace.

2.1.1 main.c

Kód **main.c** obsahuje definici globálních proměnných. Všechny proměnné, které si mají uchovat hodnotu, musí být deklarovány zde. Všechny proměnné doporučuji definovat jako **volatile**, předejdete nečekaným problémům s jejich měnícím se obsahem.

Je zde definována funkce **main()** v níž probíhá konfigurace procesoru po spuštění a následně zanoření do nekonečné smyčky.

Dále se zde nachází obsluha přerušení, viz. Tabulka 1.

V nekonečné smyčce jsou kontrolovány vlajky pro asynchronní volání funkcí. Např., poté, co přijde poslední znak zprávy z px4flow, nastaví se vlajka **px4flowDataFlag** na hodnotu 1. Ta je poté v nekonečné smyčce odchycena a jsou vykonány potřebné úkony spojené s příjmem dat - filtrace, uložení aktuálních hodnot do stavových proměnných, apod.

2.1.2 controllers.c

controllers.c obsahuje funkce pro řízení letounu.

2.1.3 system.c

system.c obsahuje funkce pro obsluhu kontroleru a letounu. Viz. Tabulka 2.

| | |
|------------------------|--|
| ISR(USART0_RX_vect) | příjem z UART0 |
| ISR(USART1_RX_vect) | příjem z UART1 |
| ISR(TIMER1_COMPA_vect) | Komparační přerušení A k 16bit čítači. Používá se pro generování vstupního PPM signálu (start pulzu). Neupravovat, pokud nevím, co dělám!! |
| ISR(TIMER1_COMPB_vect) | Komparační přerušení B k 16bit čítači. Používá se pro generování vstupního PPM signálu (konec pulzu). Neupravovat, pokud nevím, co dělám!! |
| ISR(PCINT0_vect) | Zpracování změny na vstupních PWM pinech 1..4 Neupravovat, pokud nevím, co dělám!! |
| ISR(PCINT1_vect) | Zpracování změny na vstupních PWM pinech 5..9 Neupravovat, pokud nevím, co dělám!! |
| ISR(TIMER0_OVF_vect) | Zpracování přetečení 8bit čítače (hrubé měření času). |

Tabulka 1: Použitá přerušení

| | |
|---------------------|---|
| initializeMCU() | Volá se jednou, po zapnutí. Provede nastavení mikrokontroleru. Neupravovat, pokud nevím, co dělám!! |
| enableController() | Zapne automatickou stabilizaci letounu. |
| disableController() | Vypne automatickou stabilizaci letounu. |
| armVehicle() | Provede automatické "armování" letounu. NE-POUŽÍVAT!. |
| disarmVehicle() | Provede automatické "disarmování" letounu. |
| button1check() | Kontrola stisku tlačítka 1. |
| button2check() | Kontrola stisku tlačítka 2. |

Tabulka 2: Funkce v system.c

| | |
|--------------------------|--|
| USART0_init() | inicializace UART0 |
| USART1_init() | inicializace UART1 |
| Uart0_write_char() | zapiše bajt na UART0 |
| Uart0_write_string() | zapiše string na UART1 |
| atomParseChar() | zpracuje příchozí znak z "Atomového" počítače (použití pro surfnav) |
| gumstixParseChar() | zpracuje příchozí znak z Gumstixu |
| flightCtrlParseChar() | zpracuje příchozí znak z FlightCTRL stabilizační desky |
| Decode64() | dekóduje zprávu z FlightCTRL |
| parseFlightCtrlMessage() | zpracuje zprávu z FlightCTRL stabilizační desky |
| mergeSignalsToOutput() | provádí míchání signálů regulátorů a RC vysílače. Neupravovat, pokud nevím, co dělám!! |
| px4flowParseChar() | zpracuje příchozí znak ze senzoru px4flow |
| my_mavlink_parse_char() | upravená funkce z knihovny MavLink. zpracuje znak ze senzoru px4flow. |
| capturePWMInput() | měří délku PWM pulzů z RC soupravy. Neupravovat, pokud nevím, co dělám!! |

Tabulka 3: Funkce v communication.c

2.1.4 communication.c

communication.c obsahuje funkce pro obsluhu a zpracování komunikace s externími moduly. Některé funkce zpracovávají jednotlivé příchozí bajty a jsou tedy určeny pro volání zevnitř přerušení. Jiné zpracovávají celou komunikační zprávu a musejí být volány asynchronně, zevnitř smyčky, po přijetí posledního bajtu.

2.1.5 config.h

config.h obsahuje direktivy preprocesoru pro konfiguraci celého firmwaru. Lze zde zapínat jednotlivé moduly (px4flow, gumstix, atomový PC, FlightCTRL) a nastavovat chování firmwaru. Dále se zde konfiguruje seriové linky (jejich BAUD rate) a jejich přiřazení k modulům.

Důležité je nastavení konstant pro PWM a PPM. Jsou zde hodnoty délek pulzů (min, střední, max), délka PPM rámce a délka dělicího pulzu v PPM.

Dále je zde namapování kanálů z RC na jednotlivé PWM vstupy.

| | |
|-------------------------|---|
| FRAME_ORIENTATION | Nastavení orientace letounu (PLUS.COPTER, nebo X.COPTER). Má vliv na úhly vyčítané z FlightCTRL, ty jsou vždy relativně k desce FlightCTRL. |
| GUMSTIX_CAMERA_POINTING | Kam míří kamera Gumstix počítače (FORWARD, nebo DOWNWARD), důležité pro rotaci souřadnic. |
| disableController() | Vypne automatickou stabilizaci letounu. |
| armVehicle() | Provede automatické "armování" letounu. NEPOUŽÍVAT!. |
| disarmVehicle() | Provede automatické "disarmování" letounu. |
| button1check() | Kontrola stisku tlačítka 1. |
| button2check() | Kontrola stisku tlačítka 2. |

Tabulka 4: Obsah config.h

2.2 Kompilace pro ATmega164p

Ke kompilaci je třeba mít nainstalovaný kompilátor **avr-gcc**. Pro windows je obsažen v balíčku **WinAVR**, ke stažení na adrese <http://winavr.sourceforge.net/>. Pro Linux je potřeba balíčky **gcc-avr**, **binutils-avr**, **avr-libc**, **avrdude**, **gdb-avr**.

```
sudo apt-get install gcc-avr binutils-avr gdb-avr avr-libc avrdude
```

Pro samotnou kompilaci je přítomen soubor Makefile, který obsahuje předpis pro všechny výše popsané soubory. Výstupem kompilace je soubor **main.hex**, který je vstupním parametrem pro upload do mikrokontroleru. V Linuxu je kompilace prostá - zavoláním příkazu **make all** ve složce se zdrojovými soubory. Ve Windows je postup stejný, pokud jste si nainstalovali výše zmíněný toolchain. Ten do windows doinstaloval program make, který umí interpretovat Makefile stejně, jako v Linuxu.

2.3 Upload programu do ATmega164p

Po úspěšné kompilaci máte ve složce se zdrojovými soubory soubor **main.hex**. K jeho nahrání do kontroleru potřebujete programátor (např. USBasp) a program **avrdude**. Příkaz pro upload souboru je shodný pro Windows i Linux, pouze v Linuxu je nutné volat ho s právy superuživatele.

```
avrdude -p m164P -c usbasp -U flash:w:main.hex
```

POZOR! Před nahráváním si vždy zkontrolujte, zdali je do řídicí desky řádně přivedeno napájení. Pokud je deska bez napájení, může nahrávání firmwaru nenávratně poškodit kontroler.

2.4 Logování letových dat do Gumstixe

Pro logování dat z řídicí desky do Gumstixe je potřeba dvě věci. Připravit příjem na straně Gumstixe a připravit vysílání na straně řídicí desky.

2.4.1 Strana řídicí desky

K logování slouží funkce `debug()` v souboru `main.c`. Ve funkci připravte proměnné do formátu k odeslání (lidsky čitelný text, či jiný zápis) a předejte je funkci `uart0_write_string()` která zařídí jejich předání na UART. Funkce `debug()` je pravidelně volaná, typicky po vykonání výpočtů regulátorů. Ale umístění volání je vhodné zvolit individuálně na základě požadované frekvence volání a náročnosti (množství posílaných informací). V případě velkého množství posílaných dat doporučuji rozdělit data do několika separátních funkcí, které budou volané vždy v jiném průchodu hlavní smyčky.

2.4.2 Strana Gumstixe

V Gumstixu je nutné data ukládat na Ramdisk, z důvodu pomalé hlavní paměti počítače. Ramdisk vytvoříte zavoláním skriptu `ramdisk.sh` v domovské složce.

```
root@192.168.1.103:~$ ./ramdisk.sh
```

Poté se přesuňte do složky ramdisku, která se nachází v `/tmp/ramdisk`:

```
root@192.168.1.103:~$ cd /tmp/ramdisk
```

Čtení dat ze seriové linky se prování výpisem dat ze zařízení seriové linky `ttyS0`:

```
root@192.168.1.103:/tmp/ramdisk$ cat /dev/ttyS0
```

Pokud chcete data uložit, přesměrujte výstup do souboru:

```
root@192.168.1.103:/tmp/ramdisk$ cat /dev/ttyS0 > data.txt
```

Pokud budete mít špatné wifi spojení s Gumstixem a terminál Vám bude padat, doporučuji volat poslední příkaz na pozadí:

```
root@192.168.1.103:/tmp/ramdisk$ nohup cat /dev/ttyS0 > data.txt &
```

Potom je samozřejmě nutné příkaz ukončit pomocí `kill`, číslo procesu zjistíte pomocí `ps -au`.

Ihned po skončení logování si data skopírujte z ramdisku do normálního pevného uložště! Po vypnutí Gumstixu budou data v ramdisku ztracena.