**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

**KVADROKOPTÉRA**

**ZÁVEREČNÝ PROJEKT K PREDMETU VNORENÉ RIADIACE SYSTÉMY**

**(Programová dokumentácia)**

**Ústav**: Robotika a kybernetika

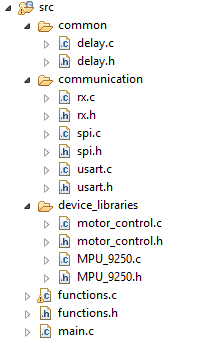
**Vypracoval:** Miroslav Kohút, Viktor Dluhoš, Jaroslav Stanko, Peter Kmeť

**Github:** <https://github.com/MiroslavKohut/quadrocopter_driver/tree/master>

**Programátorská príručka**

V tomto dokumente nájdete podrobne popísane jednotlivé programové časti. Na základe nich je možné lepšie pochopiť štruktúru celého programu a funkčnosť riadenia kvadrokoptéry.

**Štruktúra softvéru:**



Obrázok Štruktúra softvéru

Na **obrázku 1** je vidieť základnú štruktúru zdrojových kódov, ktoré boli nami vytvorené a doplnené do vopred vytvoreného projektu v Atollic studiu.

# Priečinok src

1. **main.c -** v hlavnom priečinku môžeme nájsť main.c program, ktorý v sebe obsahuje prázdny nekonečný cyklus, ktorý môže byť určený na debug posielanie dát cez Usart alebo na pomocné výpočty, ktoré nemusia byť synchronizované. V našom prípade však nemá žiadnu úlohu. Okrem hlavného cyklu sa tu nachádzajú ešte inicializácie celého programu. Sú volané z knižnice functions.h, ktorá celá beží celá na "pozadí" main programu.
2. **functions.c / functions.h** - Táto knižnica predstavuje najdôležitejšiu časť celého programu. Využíva ju main program a je previazaná so všetkými ostatnými zariadeniami a periferiálnymi knižnicami.

Inicializačné funkcie všetkých periférii a zariadení volané v maine. Vo funkcii **global\_init** sa inicializujú zariadenia a potrebné periférie. Vo funkcii **timers\_init** sú inicializované časovače pre meranie dát z imu jednotky a pre vzorkovanie regulátora akčného zásahu:

**void** **global\_init**();

**void** **timers\_init**();

Inicializačné funkcie časovača na riadenie kvadrokoptéry a na čítanie vzoriek z gyroskopu. Ako vstup do funkcie berú periódu v akej sa majú vykonávať.

**void** **TIM4\_controller\_timer**(**int** period\_in\_miliseconds);

**void** **TIM5\_sampling\_timer**(**int** period\_in\_miliseconds);

Funkcia pre komplementárny filter na kombinovanie dát gyroskopu a akcelerometra pre riadenie roll a pitch (nebolo súčasťou úlohy)

**void** **complementary\_filter**();

Funkcie pre riadenie jednotlivých osí kvadrokoptéry a kompletné riadenie zariadenia. My používame len

**void** **PID\_yaw\_control**();

Ostatné funkcie je potrebné kalibrovať a lepšie otestovať no sú pripravené

**void** **PID\_stabilization\_control**();

**void** **PID\_pitch\_control**();

**void** **PID\_roll\_control**();

Na záver sú v súbore functions.c aj handlery pre inicializovaný timre. TIM4 číta dáta z gyra a akcelerometra v inicializačnej perióde. Taktiež vie vykonávať moving average filter z počtu vzoriek **moveing\_average\_samples**, ktorý je definovaný vo functions.h. Prečítané dáta ukladá do globálnych polí a tie potom môžu byť spracované inými funkciami.

**void TIM4\_IRQHandler**()

TIM5 vykonáva kompletnú reguláciu a akčný zásah riadenia. Je možné v ňom určiť, ktorá funkcia regulátora sa má použiť. My pre účely nášho zadania používame **PID\_yaw\_control**. Okrem toho tu aj komplementárny filter spracováva údaje z MPU9250 jednotky (nie je potrebné pre funkčnosť zadania).

**void** **TIM5\_IRQHandler**()

# Priečinok common

1. **delay.c / delay.h - JARO :)**

# Priečinok communication

1. **rx.c / rx.h - JARO :)**
2. **spi.c / spi.h -** Knižnica pre komunikáciu cez spi rozhranie. Je inicializovaná a použivána knižnicou pre zariadenie MPU9250. Slúži na čítanie dát z tohoto zariadenia a taktiež na zápis potrebných nastavení.

Nasledujúca funkcia inicializuje SPI zbernicu a umožní cez ňu komunikovať s pripojeným zariadením.

**void** **init\_SPI1**();

Funkcie na zápis a čítanie cez zbernici. Umožňujú zápis buď do konkrétneho registra alebo čítanie z jedného alebo viacero po sebe idúcich registrov:

**void** **write\_reg**( uint8\_t WriteAddr, uint8\_t WriteData);

uint8\_t **read\_reg**( uint8\_t WriteAddr);

**void** **read\_regs**( uint8\_t ReadAddr, uint8\_t \*ReadBuf, **unsigned** **int** Bytes );

Pomocné funkcie určené na setnutie a resetnutie chipselectu, ktorý vlastne spúšťa komunikáciu a je umiestnený na pine **PA08**.

**void** **chip\_select**();

**void** **chip\_deselect**();

1. **usart.c / usart.h -** jednoduchá komunikačná knižnica určená pra komunikáciu z pc pomocou **USART2** určená hlavne na debug a testing.

Inicializáciu sériovej komunikácie. V súbore usart.h je možné definovať požadovanú baudrate, s ktorou chcete pracovať. Usart je taktiež inicializovaný s NVIC prerušením pre príjmané dáta, ktoré sa vo funkcii **USART2\_IRQHandler** ukladajú do premennej **rec\_data.**

**void** **usart\_init**();

Základné funkcie na odosielanie reťazcov textového poľa alebo ľubovoľného desatinného čísla:

**void** **USART\_send\_function**(char text[]);

**void** **USART\_send\_function\_number**(**float** number);

# Priečinok device\_libraries

1. **motor\_control.c / motor\_control.h - JARO :)**
2. **MPU\_9250.c / MPU\_9250.h -** knižnica pre komunikáciu s imu jednotkou a čítanie potrebných dát z akcelerometra a gyroskopu.

Inicializačné funkcie pre inicializovanie zariadenia a nastaveniach príslušných registrov. Taktiež sú tu funkcie pre nastavenie mierky gyroskopu a akcelerometra:

uint8\_t **mpu9250\_init**(**int** sample\_rate\_div,**int** low\_pass\_filter);

uint32\_t **set\_acc\_scale**(**int** scale);

uint32\_t **set\_gyro\_scale**(**int** scale);

Ďalšia skupina funkcií, ktorá slúži na kalibráciu akcelerometra a čítanie jednotlivých dát tie sú potom ukladané do globálnych premenných: **accelerometer\_data** a **gyroscope\_data**, ktoré obsahujú hrubé prečítané dáta a je možné ich používať v ostatných funkciách

**void** **calib\_acc**();

**void** **read\_acc**();

**void** **read\_rot**();