SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE FAKULTA ELEKTROTECHNIKY A INFORMATIKY

KVADROKOPTÉRA

ZÁVEREČNÝ PROJEKT K PREDMETU VNORENÉ RIADIACE SYSTÉMY

(Programová dokumentácia)

Ústav: Robotika a kybernetika

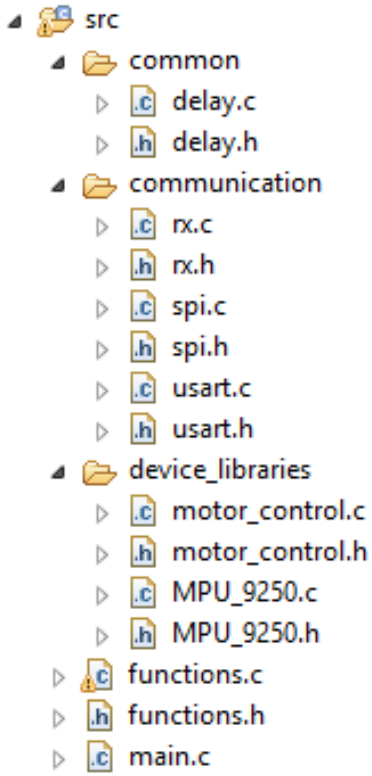
Vypracoval: Miroslav Kohút, Viktor Dluhoš, Jaromír Stanko, Peter Kmeť

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 1 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 (roll, pitch, yaw) a kompletné riadenie zariadenia.

```
void PID_rate_control();
```

Ostatné funkcie je potrebné kalibrovať a lepšie otestovať no sú pripravené

```
void PID_stabilization_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 -** knižnica slúžiaca na generovanie zdržania programu (delay) požadovanej dĺžky.

`void delay_ms(uint32_t t)` – vstupným parametrom je čas v milisekundách udávajúci dĺžku pozastavenia programu. Funckia využiva „System tick timer“ a jeho periodické generovanie prerušení (každú milisekundu). Perióda generovania prerušenia sa nastavuje vo funkcii:

```
uint32_t SysTick_Config(uint32_t ticks);
```

Vstupným parametrom je perióda s akou sa generuje prerušenie (každých **n** tikov procesora).

Priečinok communication

1. **rx.c / rx.h –** knižnica pre spracovanie riadiacich signálov z vysielačky. Z vysielačky sa prijíma signál od 4 kanálov – **roll, pitch, yaw a throttle**.

   `void rx_init();` - hlavná inicializačná funkcia, ktorá inicializuje timer, GPIO a prerušenia používané na spracovanie signálu z prijímača:

   ```
   void NVIC_init(void);
   void Timer_init(void);
   void GPIO_init(void);
   ```

   Funkcie `void TIM3_IRQHandler(void)`, `void TIM4_IRQHandler(void)` a `void TIM9_IRQHandler(void)` obsluhujú prerušenia od časovačov (TIM3, TIM4, TIM9) vyvolané prijatím signálu z vysielačky. Vo vnútri funkcie sa vypočíta šírka a frekvencia prijímaného PWM signálu z vysielačky, ktorý sa ďalej spracováva pri riadení. Vypočítané hodnoty sa zapisujú do globálnych premenných:

   ```
   __IO float frequency_throttle;
   __IO float frequency_yaw;
   __IO float frequency_pitch;
   __IO float frequency_roll;
   __IO int16_t pulse_length_throttle;
   __IO int16_t pulse_length_yaw;
   __IO int16_t pulse_length_putch;
   __IO int16_t pulse_length_roll;
   ```

2. **spi.c / spi.h -** Knižnica pre komunikáciu cez spi rozhranie. Je inicializovaná a používaná 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();
   ```

3. **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 –** knižnica/driver pre ovládanie motorov kvadrokptéry.

   `void motor_init()` – hlavná inicializačná funkcia pre ovládanie motorov kvadrokoptéry. Vo vnútri sa volajú funkcie na inicializáciu timera pre generovanie PWM signálu, inicializáciu GPIO ku ktorým sú pripojené BLDC motory a inicializácia motorov privedením po zapnutí kvadrokoptéry minimálnu hodnotu šírky riadiaceho impulzu na všetky motory:

   ```
   void TIM2_PWM_init(void);
   void PWM_init(void);
   void GPIO_PWM_init(void);
   void set_throttle(uint8_t motor, int8_t data);
   ```

   Vstupnými parametrami funkcie **set_throttle** sú číslo motora, na ktorý sa má priviesť riadiaci signál(uint8_t motor) a percentuálna hodnota ako rýchlo sa má motor točiť(int8_t data v rozsahu od 0% - 100%) teda šírka riadiaceho impulzu. V tejto funkcii sa taktiež nachádza parameter **MAX_THROTTLE**, ktorý je definovaný v **motor_control.h** a obmedzuje maximálnu výstupnú rýchlosť na motoroch. Jeho rozsah je 0-100 a pre účely testu sme ho zvolili 30.

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();
   ```

Záver: Softvér je ešte stále len v štádiu vývoja a preto treba so zariadením aj jeho úpravami narábať s maximálnou opatrnosťou a pozornosťou. Z hľadiska vlastného bezpečia neodporúčame náhodne meniť jednotlivé parametre programu.