Gymnázium, Praha 6, Arabská 14

Ročníková práce z programování

2018/2019 3.E

Ondřej Kuban, Otakar Kodytek, Jiří Štengl, Josef Vašička

Arabská 14, 160 00, Praha 6, Vokovice

Předmět: Programování

Téma: Kvadrokoptéra

Autoři: Ondřej Kuban, Otakar Kodytek, Jiří Štengl, Josef Vašička

Třída: 3.E

Školní rok: 2018/2019

Třídní učitel: Mrg. Jana Urzová

Prohlášení

Prohlašuji, že jsme jedinými autory tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V …… Dne ……

……

Podpis

Poděkování

Tímto bychom chtěli poděkovat Mrg. Janu Lánovi a Ing. Danielu Kahounovi za ochotu s řešením problémů této ročníkové práce. Dále Mgr. Janě Urzové za pomoc s matematikou, kterou jsme využili při programování ročníkové práce. V neposlední řadě také Ing. Tomáši Báčovi, který nám poskytl mnoho cenných rad, upozornil Nás na problémy, se kterými se můžeme potýkat a případně nastínil jejich řešení. Dále děkujeme Ing. Vítovi Kubanovi za pomoc s problematikou elektrotechniky a mechaniky.

Anotace

Tato ročníková práce se zabývá sestrojením a naprogramováním kvadrokoptéry, ovladače a mobilní aplikace pro chytré telefony. Rozebírá popis použitého hardwaru a jeho uplatnění. Dále práce popisuje funkci a vývoj PID regulátoru, komunikace mezi kvadrokoptérou a vysílačem, mezi vysílačem a chytrým telefonem. V práci jsou popsána schémata zapojení a řešení jednotlivých problémů.

Anotation

This term work applies with constraction and programming of quadcopter, remote controller and mobile application for smart phones. It analyses description of hardware and its usage and utilization. Furthermore work describes function and development of PID controller, communication between quadcopter and remote controller, between remote controller and smart phone. In the term work are described electronic schemes and solution of particular problems.

1. Obsah

2. Úvod

Motivací výběru tématu ročníkové práce bylo nastavit si cíl, který pro nás bude výzvou. Mimo to také ozkoušení programování elektronického zařízení, které interaguje s fyzickým prostředím a zlepšení elektrotechnických schopností.

Kvadrokoptéra je multirotorová helikoptéra, poháněna čtyřmi rotory. Dva rotory se točí po směru hodinových ručiček a zbylé dva proti směru hodinových ručiček. Výkon jednotlivých motorů je regulován PID regulátorem tak, aby tah všech motorů byl kolmý k požadovanému úhlu náklonu. Ovládání zprostředkovává upravený vysílač s vlastním hardwarem. Ovladač komunikuje s kvadrokoptérou a mobilním zařízením ,kterému posílá data o náklonu a stavu baterie kvadrokoptéry. Kvadrokoptéra posílá do ovladače aktuální náklon a stav baterie. Ovladač odesílá pouze požadovaný výkon, náklon a otočení pro kvadrokoptéru.

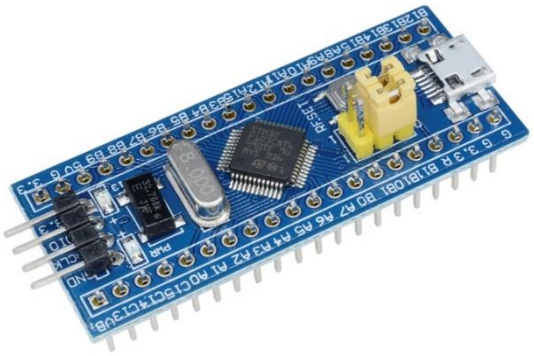
O zobrazování dat se stará aplikace spustitelná na zařízeních s operačním systémem Android. Přijímá data přes bluetooth přijímač, následně data dešifruje, vykreslí umělý horizont a zobrazí stav baterie. Pro přesnější přehled o přijatých datech má aplikace obrazovku s aktuálními daty. Aplikace načítá data o počasí v okolí z webového API, a následně uživateli zobrazí, zdali je bezpečné létat.

3. Hardware

3. 1. Komponenty

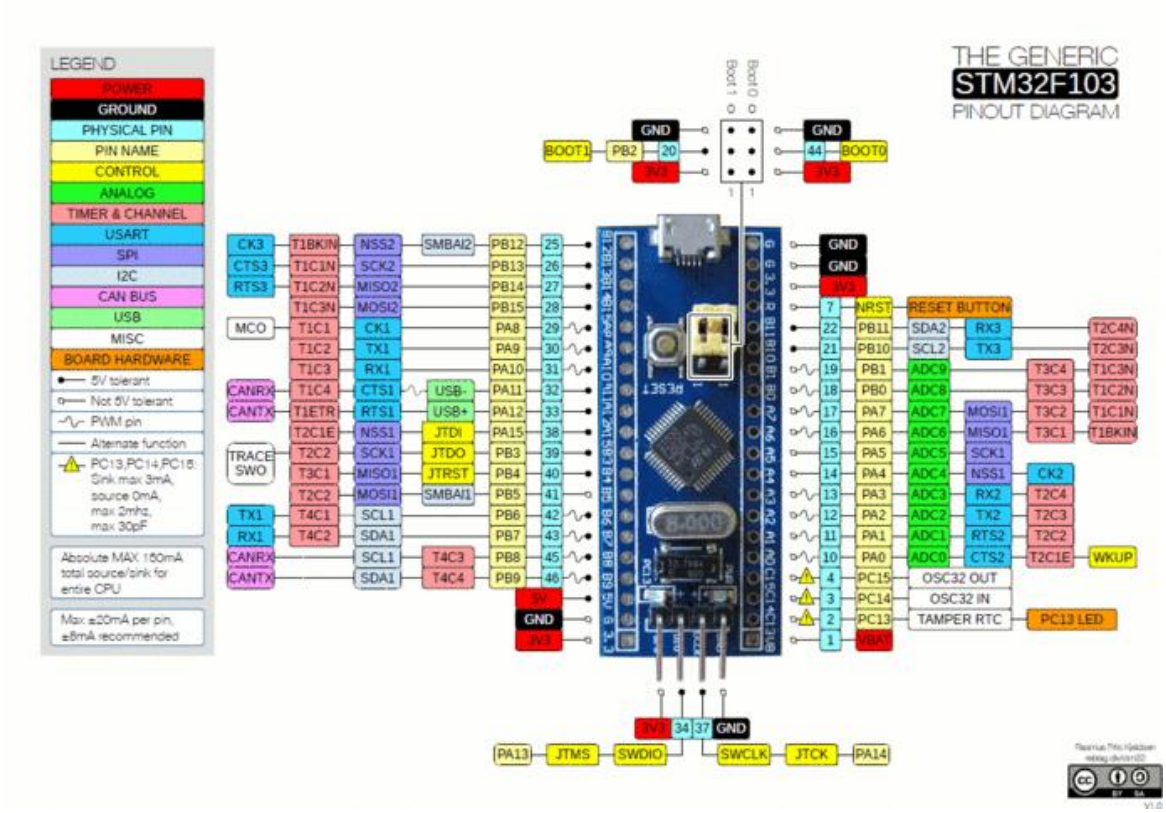
3. 1. 1. Microcontroller – STM32

Hlavní komponent ovládající celou kvadrokoptéru je mikrořadič STM32F103CT6, známý jako Blue Pill. Tento mikrořadič je vyvíjen firmou STMMicroelectronics. Je založen na 32bitovém jádru procesoru Cortex-M3. Mikrořadič pochází z rodiny STM32 F1, konkrétně z řady Performance (STM32F103). Disponuje 20kb RAM a 60kb flash paměti a pracuje na frekvenci 72MHz. Dále má STM32 10 12bitových ADC (Analog to Digital Converter), a tudíž čtení hodnot je několikanásobně přesnější než u Arduina. STM32 má 3 sériové porty oproti Arduinu, které má pouze jeden. STM32 operuje na napětí 3.3V a tudíž používá 3.3V logiku na výstupních/vstupních pinech. Některé piny podporují i 5V logiku. Disponuje také třemi sériovými linkami, dvěma I2C sběrnicemi, dvěma SPI sběrnicemi a CAN sběrnicí.



STM32

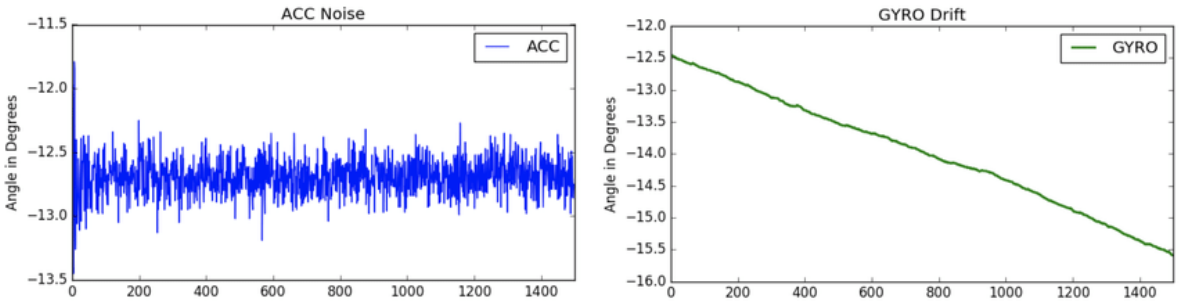
STM32 se dá programovat v Arduino IDE pomocí knihovny STMDuino (konkrétně STM32Duino). K nahrání kódu do STM32 je třeba přepojit jumper Booto z polohy 0 do polohy 1. K nahrání programu do STM32 lze využít ST-Link od firmy STMicro controllers nebo USB to Serial převodník, který připojíme do jednoho ze 3 Sériových portů na STM32. Mikrořadič STM32 jsme zvolili, protože poskytuje vyšší výkon než 8bitový mikrořadič ATMega328P, který je hlavním prvkem většiny Arduin.



STM32 pinout

3. 1. 2. Gyroskop a akcelerometr – GY-521

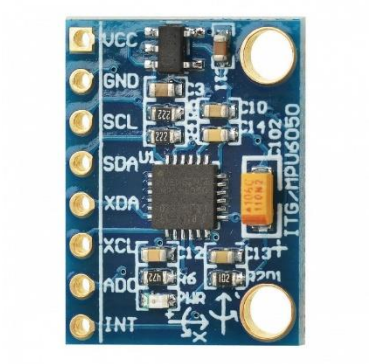
Modul GY-521 obsahuje čip MPU6050, který v sobě kombinuje gyroskop, akcelerometr a IMU. Tento modul obsahuje také teploměr, ten je ovšem velmi nepřesný, protože se nachází velmi blízko všech ostatních komponent, tudíž data z něj jsou zkreslená. Gyroskop i akcelerometr má své nevýhody, které lze vyřešit použitím obou sensorů současně.



Šum akcelerometru a drift gyroskopu

Gyroskop je zařízení měřící úhlovou rychlost pomocí Coriolisova efektu, díky čemuž je schopen udávat přesně hodnoty náklonu. U gyroskopu dochází v průběhu času k driftu, což je efekt, který zapříčiní již po krátkém čase nepřesnosti v úhlu, které by znemožnili přesné vyrovnávání.

Akcelerometr měří zrychlení ve všech osách, úhel pomocí akcelerometru lze vypočítat také, ovšem odchylka zde bude každé měření příliš velká na samostatné fungování. Akcelerometr změří vždy určitý šum.

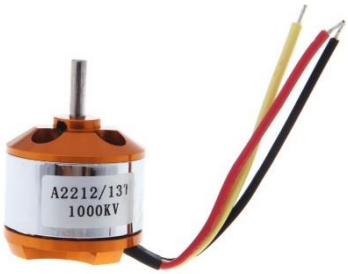


MPU60560

Velmi důležitý je DMP (IDigital Motion Processor). Pokud chceme získat přesný úhel náklonu potřebujeme sjednotit data z gyroskopu a akcelerometru a tento procesor je schopný tento náklon spočítat s použitím vhodné knihovny. Díky tomu lze vypisovat rovnou náklon všech 3 os ve stupních.

3. 1. 3. Motory – A2212/13T

Pro pohon kvadrokoptéry jsou využity bezkartáčové motory s rotačním pláštěm a statickou střední částí motoru. Výhoda outrunner motorů spočívá ve snížené výšce celého motoru, který je nižší než inrunner motory. Další výhodou je vyšší točivý moment outrunner motoru, který se hodí k rychlým manévrům kvadrokoptéry.



outrunner motor

Nevýhodu outrunner motoru je menší maximální počet otáček za minutu a horší odvod tepla ze statoru motoru. Motory použité na tomto dronu mají rychlost 1000 otáček/V. Výhoda bezkartáčových motorů oproti standartním DC

motorům je nepřítomnost komutátoru, který zařizuje mechanické střídání polarit magnetických polí vytvořených cívkami v motoru. Kontakty komutátoru jsou vyrobeny z uhlíku, který se chodem motoru vybrousí což vyústí v nefunkčnost motoru. Motory použité v této kvadrokoptéře mají cívky spojené do tvaru písmene Y. Díky využití 3-fázového ESC do motoru vedou pouze 3 přívodní kabely, protože fázový posun mezi jednotlivými fázemi je 120° a tudíž je na spoji mezi cívkami nulové napětí.

3. 1. 4. Baterie – 3300 mAh 25C tříčlánková LiPo

Lithium polymerová baterie, která je využita u této kvadrokoptéry má jmenovité napětí 12,6V. Udávaná kapacita baterie je 3,3Ah, což spolu s vybíjecím proudem 25C dovoluje odebírat z baterie teoreticky až 82,5 ampér. Výhodou lithium polymerových baterií je jejich velká kapacita při malých prostorech, dlouhá životnost, vysoký vybíjecí i nabíjecí proud.

LiPo baterie

3. 1. 5. Radiová komunikace– HC-12

Vysílací modul HC-12 slouží pro bezdrátovou komunikaci mezi kvadrokoptérou a vysílačem. Jeho maximální přenosová rychlost je 115200 baudů/sekundu. Modul umí pracovat v napájecím rozsahu 3.3V-5V a je schopný pracovat s 5V logikou. S mikrořadičem komunikuje pomocí sběrnice UART. Modul se nastavuje pomocí AT příkazů, pomocí kterých lze nastavit rychlost přenosu, nastavit jednu ze 100 vysílacích frekvencí a vysílací výkon. Modul používá integrovaný obvod SI4463, který se stará o bezdrátovou komunikaci a mikrořadič STM8, který slouží jako mezičlen mezi SI4463 a sběrnicí UART.

Výrobce slibuje dosah až 600m na volném prostranství. Odběr modulu je 20mA při běžném chodu. Při vysílání dosahuje spotřeby až 100mA.

HC-12

3. 1. 6. Bluetooth komunikace – HC-06

Bluetooth komunikace zprostředkovává propojení mezi mobilním zařízením a ovladačem kvadrokoptéry. Modul HC06 obsahuje bluetooth ve verzi 2.0. Komunikace s mikrořadičem probíhá po sériové lince rychlostí 9600 baudů/sekundu. Nevýhodou oproti HC-12 je velmi malý dosah, který dosahuje pouze 10 metrů na volném prostranství. Pracovní napětí se pohybuje od 3,3 do 6 voltů. Při napětí 5 voltů je odběr proudu asi 20 mA, v maximu až 40 mA.

H -06

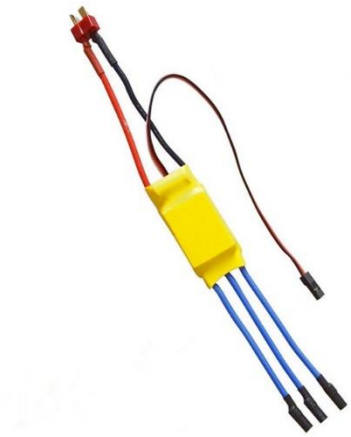
3. 1. 7. Kostra dronu – F450

Kostra kvadrokoptéry typu F450 se skládá ze čtyř ramen a dvou desek. Jedna z desek disponuje letovacími ploškami pro baterii a ESC.

Kostra F450

3. 1. 8. Electric speed controller- 30A BLDC ESC

ESC je obvod, který se skládá z 3 řídících polovičních můstků, mikrořadiče a pasivních komponent. Poloviční můstek se skládá z 2 komplementárních MOSFETů, které jsou řízeny mikrořadičem. Mikrořadič se v ESC stará o správně načasované otevírání jednotlivých polovičních můstků, tak aby došlo k vytvoření 3-fázového řídícího proudu. Každá řídící fáze je posunuta o 120° oproti zbylým dvěma fázím. Změnou frekvence fází se dociluje změna rychlosti motoru. ESC má pouze 1 řídící vodič, který přijímá signál o délce 1-2 ms (1 ms pro zastavení, 2 ms pro plný výkon). Použité ESC signalizuje pípáním abnormální napětí a nerozpoznaný signál.



BLDC ESC

3. 1. 9. Vrtule – 1045

Toto označení vrtule vychází z jejích rozměrů v palcích, které jsou 10x4,5”. Průměr vrtule je tedy 10 palců (25,4 cm). 4,5 palců je stoupání, které vyjadřuje sklon vrtule. Tuto hodnotu si lze představit jako hloubku zavrtání vrtule při jedné otáčce.

Vrtule 1045

3. 2. Montáž

3. 2. 1. Deska periferií

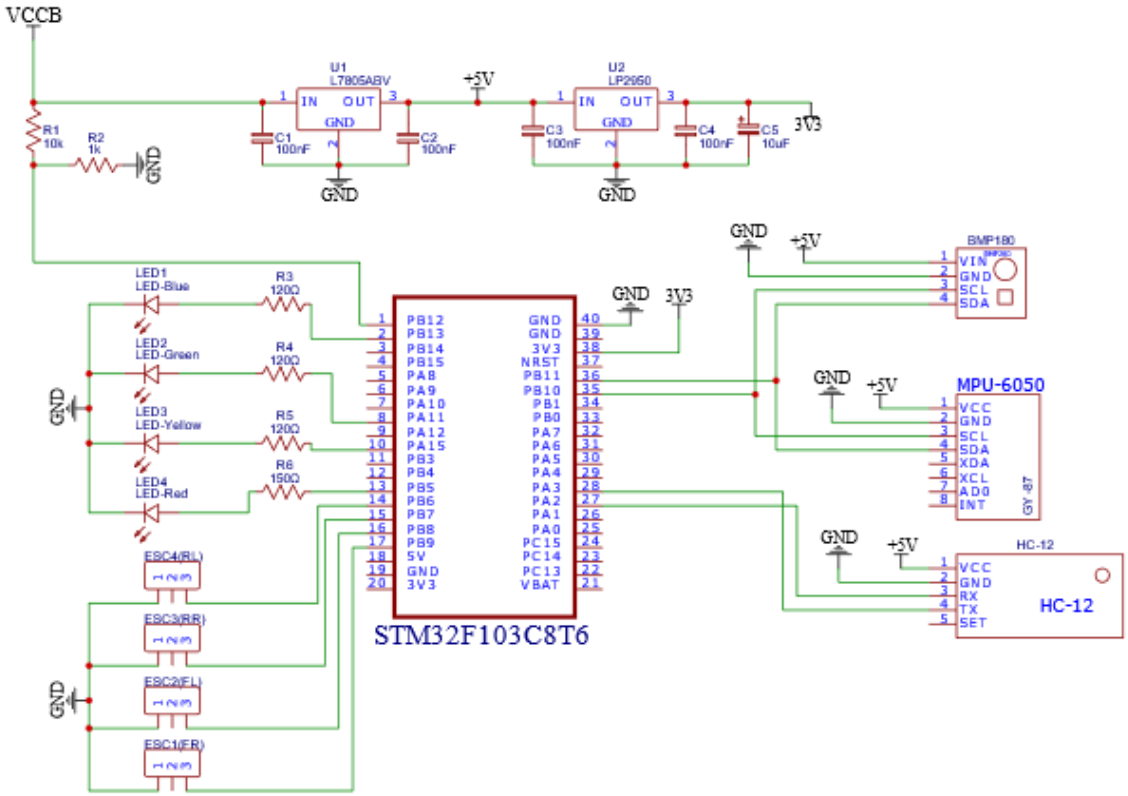


Schéma zapojení kvadrokoptéry

Na obrázku vidíme schéma zapojení desky periferií. Deska je napájena baterií s pracovním napětím 12,6- 10V (VCCB). Napětí z baterie je regulátorem L7805(U1) sníženo na 5V. Vstup a výstup regulátoru je stabilizován keramickými kondenzátory (C1, C2) o kapacitě 100 nano faradů. Druhý regulátor LP2950(U2) snižuje napětí z 5V na 3,3V. Vstup a je stabilizován opět keramickým kondenzátorem (C3) o kapacitě 100 nano faradů. Výstup je stabilizován keramickým kondenzátorem (C4) o kapacitě 100 nano faradů a elektrolytickým kondenzátorem o kapacitě 10 mikro faradů. Na pin PB12 je připojen odporový dělič napětí v poměru 1:10.

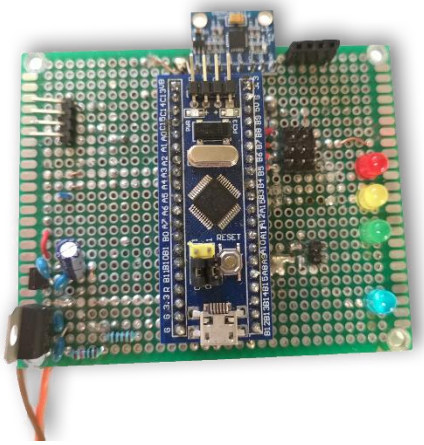


Foto zapojení 1

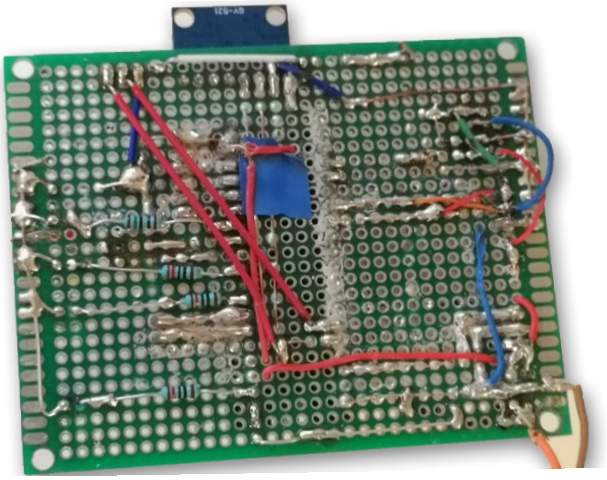


Foto zapojení 2

Barometr BME/BMP280 a akcelerometr MPU-6050 je napájen zdrojem 5V a připojen I2C sběrnicí na piny PB11 a PB10. Komunikaci s ovladačem zajišťuje vysílací modul HC-12, který je připojen serial sběrnicí k pinům PA3 a PA2.

Konektory pro ESC (electric speed controller) jsou připojené na piny PB6 až PB9. Signální LED diody jsou připojeny na piny PB5, PA15, PA11 a PB13. Před každou LED diodou je předřadný odpor.

Deska periferií je uchycena na horní desku kostry kvadrokoptéry pomocí vytisknuté redukce.

4. Software

4. 1. Kvadrokoptéra

Vývoj softwaru pro kvadroptéru probíhal ve vývojovém prostředí Arduino IDE, které je vytvořeno firmou arduino.cc. K programování mikrořadiče STM32F103C8T6, v Arduino IDE, je potřeba rozšíření stm32duino které je vyvíjeno Fredericem Pillonem a Laurentem Meunierem. Byly použity knihovny Servo.h, Wire.h a pozměněné knihovny I2Cdev.h, MPU6050_6Axis_MotionApps20.h. K programování mikrořadiče na kvadroptéře byl použit USB to Serial převaděč.

4. 1. 1. PID regulátor

PID regulátor je algoritmus používaný k přesnému řízení motorů pomocí zpracování regulační odchylky. PID regulátor se skládá ze tří složek - proporcioální, integrační a derivační. Regulátor nejprve porovná získanou hodnotu s přednastavenou hodnotou, přičemž výsledná hodnota je regulační odchylka. Tuto odchylku využívá při výpočtech všech tří složek, které po sečtení udávají s vysokou citlivostí hodnotu řídící funkce.

4. 1. 1. a) Proporcionální složka

P regulátor funguje jen jako zesilovač signálu. Regulační odchylka je vynásobena konstantou zesílení a je jí tedy přímo úměrná. Tato složka je důležitá především pro vyrovnávání velkých výkyvů, ovšem nevýhodou je její překmitávání, a především neschopnost adaptovat se na změnu situace (například změna váhy či silnější vítr). Konstanta zesílení by v takovém případě zůstal nezměněn, ačkoli na vyrovnání by bylo třeba větší změny ve výkonu. P regulátor je jediný, který se ve výjimečných případech dá použít bez dalších složek (toto ovšem neplatí pro kvadroptéry).

$$P_{\text{out}} = K_{\text{p}} e(t).$$

matematický vzorec P regulátoru

4. 1. 1. b) Integrační složka

I regulátor integruje regulační odchylku, následně výslednou hodnotu vynásobí zesílením integračního regulátoru. Integrační složka se využívá k vyrovnání dlouhodobých výkyvů, jako je například vítr, nebo nerovnoměrné

rozložení váhy, protože odchylka bude dlouhou dobu vyšší na jedné straně. Integrační složka funguje jako výpočet průměru všech dosavadních odchylek. Problémem I regulátoru je ovšem jeho nedostatečná rychlost reakce na náhlé změny, a pokud by byl samostatný, tak by stejně jako P regulátor překmitával.

$$I_{\text{out}} = K_{\text{i}} \int_0^t e(\tau)\, d\tau.$$

matematický vzorec I regulátoru

4. 1. 1. c) Derivační složka

D regulátor je složka, ve které je hodnota řídící funkce přímo úměrná derivaci regulační odchylky. Vypočtení funguje tedy jako změna odchylka za čas, přičemž výsledkem je úhlová rychlost. Derivační regulátor nemůže být použit samostatně. Prvním důvodem je, že velmi rapidně zesiluje malé rozdíly a je tedy velmi náchylný vůči šumu. Druhým důvodem je, že jeho cílem není dosáhnout úhlu blížícího se uživatelem požadované hodnotě, ale pouze vykompenzovat odchýlení od poslední pozice, což se mu v praxi nemůže podařit tak přesně, aby udržel správný náklon.

$$D_{\text{out}} = K_{\text{d}} \frac{de(t)}{dt}.$$

matematický vzorec D regulátoru

4. 1. 1. d) Implementace

O výpočet hodnot PID regulátoru se stará balíček kódu v souboru PID.ino který obsahuje metody setDesiredAngles a calculatePid.

Metoda setDesiredAngles je volaná se třemi integerovými parametry - side, forward a rotation. Parametry side a forward obsahují procentuální náklon, tyto hodnoty jsou následně vyděleny 5 a přiřazeny do proměnných desiredAngleRoll a desiredAnglePitch. Do proměné desiredAngleYaw se přičítá hodnota rotation. Jelikož je metoda setDesiredAngles volaná několikrát za sekundu je nutné dělit hodnotu rotation. Experimentálně jsme zjistili že hodnotu je optimální dělit 200. Změnou této hodnoty se reguluje rychlost otáčení kvadrokoptéry. Následně pokud hodnota proměnné desiredAngleYaw dosáhne hodnoty 360 je následně přepočítána na hodnotu 0. Toto opatření je zde kvůli gyroskopu, který není schopný naměřit větší úhel nežli je 360 stupňů.

Metoda calculatePid je volaná vždy po získání aktuálního naklonění kvadrokoptéry z gyroskopu. Metoda vypočítá aktuální chyby natočení kvadrokoptéry odečtením požadovaných úhlů od aktuálního naklonění. Pokud tato metoda proběhne poprvé, uloží si aktuální yaw natočení jako požadovaný úhel. Toto opatření je zde aby se kvadrokoptéra po startu nezačala vyrovnávat na natočení 0 stupňů.

Výpočet proporcionální části regulátoru pro všechny osy je vypočítán násobkem jejich konstanty zesílení a chyby úhlu. K výpočtu derivační části regulátoru potřebujeme získat změnu času neboli uběhlý čas od posledního měření. Tuto změnu získáme načtením vnitřního času mikrořadiče pomocí metody micros, která vrací počet mikrosekund od spuštění mikrořadiče, od které odečteme hodnotu lastTime. Změna času je převedena na milisekundy.

Výpočet derivační složky pro každou osu je rozdíl jejich aktuální a poslední chyby vydělen změnou času a následně vynásoben konstantou zesílení. Následně se sečtou hodnoty všech složek regulátoru pro každou osu. Aby nedošlo k nechtěnému překompenzování výkonu je hodnota PID regulátorů omezena. Na konec se uloží naměřené chyby do proměnných.

4. 1. 2. Gyroskop a akcelerometr

Výhodou námi použitého modulu na měření úhlu je přítomnost DMP (digital motion processor), který dokáže data z modulu rychle zpracovat a není potřeba využívat výpočetní výkon mikroprocesoru kvadroptéry.

Mikrokontroler využívá knihovnu s názvem MPU6050_6axis_motionapps20 pro správu modulu a knihovny Wire, I2Cdev pro přenos dat. Tyto knihovny musely být upraveny pro komunikaci po druhé I2C sběrnici na STM32. Tato úprava se týká především souboru I2Cdev.cpp knihovny I2Cdev kde pro použití druhé I2C sběrnice byly experimentálně odstraněny řádky č.48-82 a 89. Následně byl název objektu TwoWire předefinován z Wire2 na Wire(obrazek).

```
#include "I2Cdev.h"

    // NBWire implementation based heavily on code by Gene Knight <Gene@Telobot.com>
    // Originally posted on the Arduino forum at http://arduino.cc/forum/index.php/topic,70705.0.html
    // Originally offered to the i2cdevlib project at http://arduino.cc/forum/index.php/topic,68210.30.html
    TwoWire Wire2(2,I2C_FAST_MODE);
#define Wire Wire2
```

object TwoWire

Po detailnějším přezkoumání souboru MPU6050_6Axis_MotionApps20.h knihovny MPU6050 bylo zjištěno že po změně poslední hodnoty na řádce č.305 lze zvýšit frekvenci načítání dat z MPU6050. Hodnota byla nastavena na 100Hz, vyšší frekvence by obsahovala více chybných dat neboli šumu a nižší frekvence by ovlivnila rychlost reakce PID regulátoru.

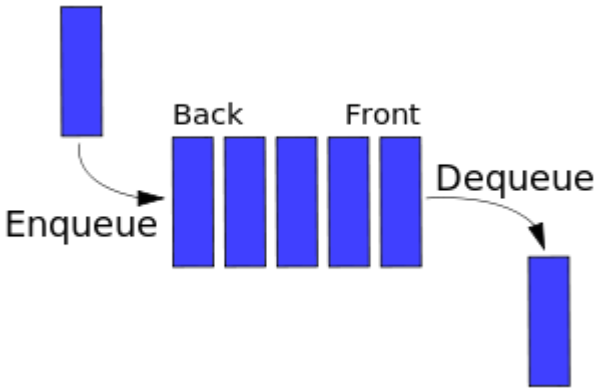
```
273  const unsigned char dmpConfig[MPU6050_DMP_CONFIG_SIZE] PROGMEM = {
274  //  BANK    OFFSET  LENGTH  [DATA]
275      0x03,   0x7B,   0x03,   0x4C, 0xCD, 0x6C,         // FCFG_1 inv_set_gyro_calibration
276      0x03,   0xAB,   0x03,   0x36, 0x56, 0x76,         // FCFG_3 inv_set_gyro_calibration
277      0x00,   0x68,   0x04,   0x02, 0xCB, 0x47, 0xA2,   // D_0_104 inv_set_gyro_calibration
278      0x02,   0x18,   0x04,   0x00, 0x05, 0x8B, 0xC1,   // D_0_24 inv_set_gyro_calibration
279      0x01,   0x0C,   0x04,   0x00, 0x00, 0x00, 0x00,   // D_1_152 inv_set_accel_calibration
280      0x03,   0x7F,   0x06,   0x0C, 0xC9, 0x2C, 0x97, 0x97, 0x97, // FCFG_2 inv_set_accel_calibration
281      0x03,   0x89,   0x03,   0x26, 0x46, 0x66,         // FCFG_7 inv_set_accel_calibration
282      0x00,   0x6C,   0x02,   0x20, 0x00,               // D_0_108 inv_set_accel_calibration
283      0x02,   0x40,   0x04,   0x00, 0x00, 0x00, 0x00,   // CPASS_MTX_00 inv_set_compass_calibration
284      0x02,   0x44,   0x04,   0x00, 0x00, 0x00, 0x00,   // CPASS_MTX_01
285      0x02,   0x48,   0x04,   0x00, 0x00, 0x00, 0x00,   // CPASS_MTX_02
286      0x02,   0x4C,   0x04,   0x00, 0x00, 0x00, 0x00,   // CPASS_MTX_10
287      0x02,   0x50,   0x04,   0x00, 0x00, 0x00, 0x00,   // CPASS_MTX_11
288      0x02,   0x54,   0x04,   0x00, 0x00, 0x00, 0x00,   // CPASS_MTX_12
289      0x02,   0x58,   0x04,   0x00, 0x00, 0x00, 0x00,   // CPASS_MTX_20
290      0x02,   0x5C,   0x04,   0x00, 0x00, 0x00, 0x00,   // CPASS_MTX_21w
291      0x02,   0xBC,   0x04,   0x00, 0x00, 0x00, 0x00,   // CPASS_MTX_22
292      0x01,   0xEC,   0x04,   0x00, 0x00, 0x40, 0x00,   // D_1_236 inv_apply_endian_accel
293      0x03,   0x7F,   0x06,   0x0C, 0xC9, 0x2C, 0x97, 0x97, 0x97, // FCFG_2 inv_set_mpu_sensors
294      0x04,   0x02,   0x03,   0x0D, 0x35, 0x5D,         // CFG_MOTION_BIAS inv_turn_on_bias_from_no_motion
295      0x04,   0x09,   0x04,   0x87, 0x2D, 0x35, 0x3D,   // FCFG_5 inv_set_bias_update
296      0x00,   0xA3,   0x01,   0x00,                     // D_0_163 inv_set_dead_zone
297                    // SPECIAL 0x01 = enable interrupts
298      0x00,   0x00,   0x00,   0x01, // SET INT_ENABLE at i=22, SPECIAL INSTRUCTION
299      0x07,   0x86,   0x01,   0xFE,                     // CFG_6 inv_set_fifo_interupt
300      0x07,   0x41,   0x05,   0xF1, 0x20, 0x28, 0x30, 0x38, // CFG_8 inv_send_quaternion
301      0x07,   0x7E,   0x01,   0x30,                     // CFG_16 inv_set_footer
302      0x07,   0x46,   0x01,   0x9A,                     // CFG_GYRO_SOURCE inv_send_gyro
303      0x07,   0x47,   0x04,   0xF1, 0x28, 0x30, 0x38,   // CFG_9 inv_send_gyro -> inv_construct3_fifo
304      0x07,   0x6C,   0x04,   0xF1, 0x28, 0x30, 0x38,   // CFG_12 inv_send_accel -> inv_construct3_fifo
305      0x02,   0x16,   0x02,   0x00, 0x01                //(0x07 -> 16Mhz) D_0_22 inv_set_fifo_rate (0x06 for first 8mhz board) (0x09 for 8Mhz board from Binoy)
306
307      // This very last 0x01 WAS a 0x09, which drops the FIFO rate down to 20 Hz. 0x07 is 25 Hz,
308      // 0x01 is 100Hz. Going faster than 100Hz (0x00=200Hz) tends to result in very noisy data.
309      // DMP output frequency is calculated easily using this equation: (200Hz / (1 + value))
310
```

výřez kódu z knihovny MPU

Gyroskop i akcelerometr je potřeba zkalibrovat. O to se postaral speciální software, ale lze to provést také ručně. Gyroskop se položí a vyrovná na stabilní podložku, v určitém časovém intervalu se zaznamenávají hodnoty a zprůměrují se. U os roll a pitch se následně otočí znaménko a číselná hodnota se vydělí číslem 4 pro gyroskop a 7,8 pro akcelerometr. U osy yaw je nutno, při kalibraci akcelerometru, načtenou hodnotu odečíst od čísla 214 a následně dělit čísly 4, nebo 7,8. Na osu yaw totiž vždy působí zemská přitažlivost, která by měla odpovídat této hodnotě. Tyto korekce se píší do funkcí set”X/Y/Z””Gyro/Accel”OffSet.

Ze začátku programu se inicializuje I2C komunikace o frekvenci 400kHZ, následně MPU, kalibruje se gyroskop a akcelerometr a nakonec proběhne inicializace DMP.

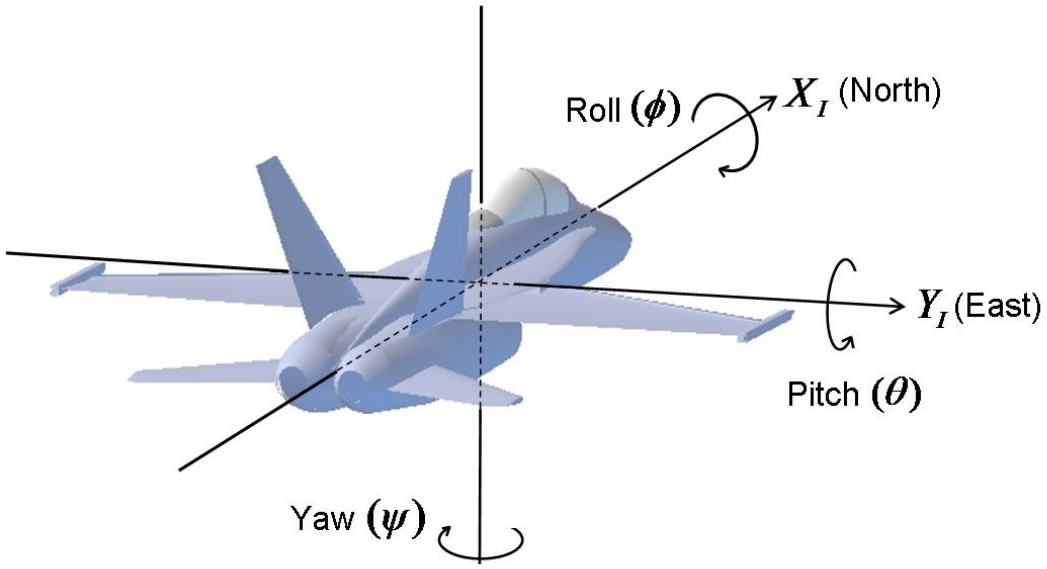
MPU ukládá data do FIFO Bufferu, což je běžný zásobník. Zpracovaná data nelze načítat vždy hned a procesor by čekal, než budou odeslána. Tato metoda



FIFO Buffer

dokáže data uchovat a posílat je ve správný moment, aniž by procesor MPU nebo procesor kvadrokoptéry čekal. V kódu je ovšem třeba kontrolovat velikost FIFO Bufferu, protože jeho přetečení by mělo za následek zaseknutí kvadrokoptéry. Toto zajišťuje funkce getFIFOCount a následující podmínka, která může využít funkci resetFIFO. Jestliže FIFO Buffer není prázdný, přidá se nová hodnota.

DMP v kódu využívá tři funkce. Jednotlivá data, která potřebuje získat se skládají z několika paketů. První funkce getQuaternion provede bitový posun u jednotlivých paketů, čímž získá určitá data o náklonu, která ovšem samotná nemají příliš velikou výpovědní hodnotu.. Druhá funkce je getGravity, která tyto



znázornění os – yaw, pitch, roll

pakety upraví podle gravitace a aktuálním nastavení citlivosti gyroskopu a akcelerometru. Výstup těchto dvou funkcí je načten třetí funkcí getYawPitchRoll, která tyto hodnoty kombinuje v matematických funkcích, které již dávají výstup přívětivější uživateli. Tuto hodnotu následně jen vynásobíme 180 a vydělíme funkcí PI, přičemž finálním výstupem je hodnota náklonu os ve stupních.

4. 1. 3. Ovládání motorů

K přesnému ovládání motorů je nutné kalibrovat ESC. Kalibrace ESC je nutná, aby ESC reagovalo v rozsahu 1000-2000 mikrosekund. Dle datasheetu (odkaz) je nutné pro kalibraci před spuštěním do ESC posílat signál který bude ESC považovat jako maximální výkon, v našem případě to byl signál dlouhý 2000 mikrosekund, poté ESC zapojit na napájení. Následně zazní zvukový signál "123" po kterém následují pípnutí jejichž počet signalizuje počet článků připojené baterie (v našem případě 3). Potvrzení uložení maximální délky signálu je indikováno dvěma pípnutími. Pro dokončení kalibrace musí být během 5 sekund do ESC poslán signál který se uloží jako signál nulového výkonu. Jedno dlouhé pípnutí signalizuje dokončení kalibrace. Použité ESC lze programovat, programovací mód se aktivuje, pokud bychom při kalibraci neposlali "nulový" signál. Programování ESC nebylo v tomto případě potřeba.

O ovládání motorů se stará balíček kódu v souboru „PropControl.ino" který obsahuje metody propInit, setThrotle a propControl.

Metoda propInit je volána po spuštění mikrořadiče bez parametru. Každému objektu Servo je přiřazen pin kterým bude knihovna ovládat motory. Pro přední pravý motor je použit pin B9, pro přední levý B8, pro zadní pravý B7 a pro zadní levý B6. Následně je na každý motor posílán inicializační signál dlouhý 1000 mikrosekund. Tento signál je nutný, aby ESC nepřešly do konfiguračního módu. Potvrzení spuštění ESC je ohlášeno třemi krátkými zvukovými signály a jedním delším. Metoda je zakončena 5 sekundami čekání.

Metoda setThrotle je volána jednou za programovou smyčku a je volána s Integerovým parametrem, který obsahuje požadovaný výkon v procentech. V metodě se proměnné throtle přiřadí 10 krát vynásobený požadovaný výkon a přičte se hodnota 1000. Tato hodnota je minimální délka signálu vysílaného do ESC.

Metoda propControl je volána jednou za programovou smyčku a je volána bez parametru. Pokud je hodnota throtle rovna 1000 posílá se do motorů nulový výkon. Toto opatření je zde aby se při přenosu kvadrokoptéry neroztočili motory kvůli PID regulátoru. Pokud se hodnota throtle nerovná 1000 tak se pravému přednímu motoru se přidělí hodnota součtu výkonu, výstupu PID regulátoru pro y osu (osa roll) a odečtou se hodnoty x osy (osa pitch) a z osy (yaw). Levému přednímu motoru se přidělí hodnota součtu výkonu, výstupu PID regulátoru pro z osu (osa yaw) a odečtou se hodnoty x osy (osa pitch) a y osy (osa roll). Pravému zadnímu motoru se přidělí hodnota součtu výkonu, výstupu PID regulátoru všech os. Levému zadnímu motoru se přidělí hodnota součtu výkonu, výstupu PID regulátoru pro x osu (osa pitch) a odečtou se hodnoty y osy (osa roll) a z osy (yaw). Následně se omezí výkony pro motory aby délka signálu byla mezi 2000 a 1030 mikrosekundami. Hodnota 1030 mikrosekund je zvolená experimentálně a určuje délku signálu při kterém se motory ještě netočí. Nakonec se do každého motoru pošle signál o požadované délce.

4. 2. Vysílač

V naší ročníkové práci probíhá komunikace mezi dronem a vysílačem, a následně mezi vysílačem a chytrým telefonem. V této kapitole se zmíním nejen o tom, jak komunikace fungují, ale i o tom, jaký byl vývoj a kde byly ve vývoji problémy a jaký hardware byl ve vysílači použit.

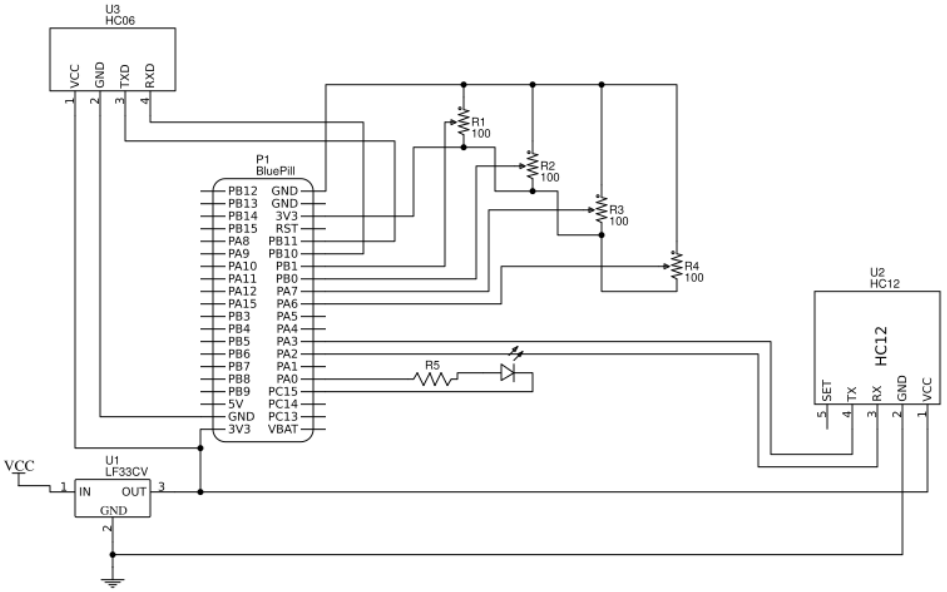


schéma zapojení vysílače

foto vysílače 1

4. 2. 1. Hardwarové zpracování vysílače

Na obrázku vidíme schéma zapojení vysílače. V zapojení vidíme dva vysílací moduly. Modul HC-12 a modul HC-06, které se starají o bezdrátový přenos informací. Dále jsou v zapojení vidět 4 potenciometry (R1-R4) v konfiguraci napěťových děličů a lineární napěťový regulátor LF33CV. Potenciometry slouží k získávání polohy páček na ovladači tím, že mění poměr napětí na jejich výstupech. LF33CV slouží ke snížení napájecího napětí ze 4 AA baterii (6V) na napětí 3.3V pro mikrořadič. Jeho využití není moc efektivní, ale pro malý proud odebíraný mikrořadičem svou funkci plní obstojně. Jako poslední na schématu zbyla LED a předřadný rezistor (R5) o velikosti 220 Ohm. LED slouží jako signální světlo pro uživatele.

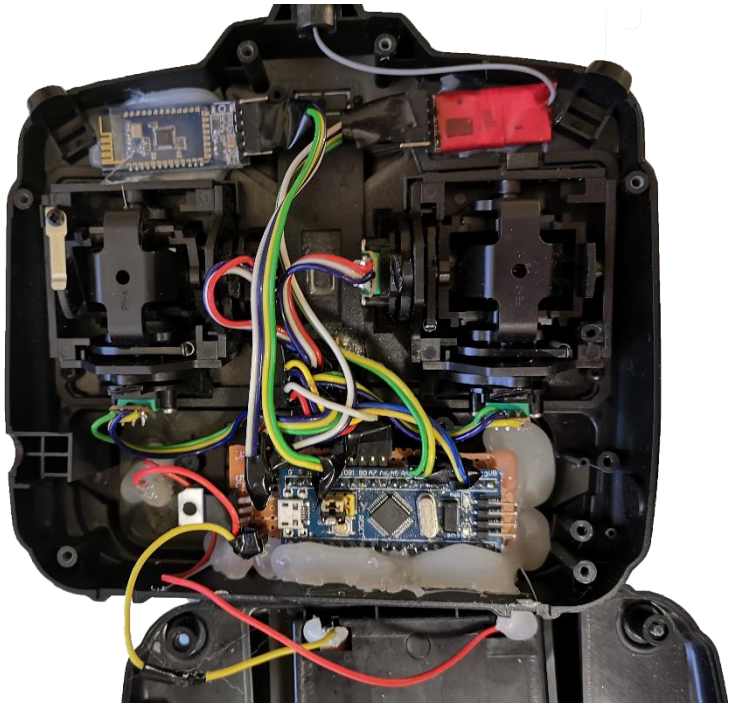


foto vysílače 2

Moduly HC-12 a HC-06 komunikují s mikrořadičem pomocí rozhraní UART přes dva datové piny (RX a TX). Modul HC-12 komunikuje rychlostí 115200 baudů/sekundu (bitů/sekundu) a modul HC-06 komunikuje pouze rychlostí 9600 b/s. HC-12 je připojeno k sériovému rozhraní 2 (A3, A2) a HC-06 je připojeno k sériovému rozhraní 3 (B11, B10). Pro změření napětí na potenciometrech (B1, B0, A7, A6) je využíván 12-bitový ADC (Analog to Digital Converter), který převede hodnotu napětí na pinech do čísla v rozsahu 0-4095.

4. 2. 2. Komunikace mezi vysílačem a dronem

V této kapitole se podíváme, jak funguje komunikace mezi vysílačem a dronem. Komunikace funguje na principu master-slave. V tomto případě je master dron, který posílá ovladači požadavky na hodnoty ovládání a zároveň mu posílá data, ze svého gyroskopu a napětí na baterii. Důvodem pro toto rozhodnutí byla série testů, kdy byl ovladač master a dron slave. Problémem při této konfiguraci je celková asynchronizace komunikace, nespolehlivost, složitost a neefektivnost celého nápadu. Asynchronizace pramenila z faktu, že ovladač se zapíná dříve než dron, a tudíž se snažil ovladač kontaktovat drona, který ještě nemusel být zapnutý. Tímto způsobem po jeho zapnutí docházelo k častým chybám, které byly způsobeny nestálým časováním mikrořadičů. Ovšem v obrácené konfiguraci k tomuto problému nedochází, protože ovladač čeká na signál od dronu, a tudíž samotné časování řídí dron.

Jak bylo zmíněno, dron slouží jako master, a tudíž posílá příkazy. Dron ovladači každých 20 milisekund pošle „R“, které je zkratka z request (požadavek). Po odeslání dron čeká na odpověď ovladače, který příkaz vyhodnotí a okamžitě pošle dronu přečtené hodnoty z páček. Po přijetí informací dron data přepočítá do použitelných hodnot a pošle ovladači „P“, které je zkratka z post (odeslat). Po odeslání „P“ pošle dron packet dat v hexadecimální soustavě. Ovladač datový packet od kvadrokoptéry převede zpět do decimální soustavy a tyto data přepošle do chytrého telefonu. V datovém packetu se nachází informace o stavu baterie a o náklonech v osách x, y a z.

Data jsou na dron posílána jako 4 po sobě jdoucí 3 ciferná čísla. Tyto čísla jsou v rozmezí 100-300 a představují procentuální náklon páček (vyjma páčky ovládající výkon, která nabývá hodnot pouze v rozsahu 200-300). Tento způsob jsem vybral z důvodu zachování konstantní délky posílaného datové řetězce, ze kterého se data parsují, protože STMDuino neobsahuje žádné knihovní funkce pro práci s řetězci znaků natož s jejich parsováním. Tyto hodnoty vznikají odečtením procentuální hodnoty páčky a následného přičtení čísla 200. V dronu se číslo 200 opět odečte, a tudíž dron získá původní procentuální hodnoty páček.

Z dronu jsou posílána data v hexadecimální soustavě. K tomuto postupu jsem se uchýlil, protože množství dat posílaných z dronu je mnohem vyšší než množství dat posílaných na dron. Konkrétně se jedná o 3 různé hodnoty v rozsahu

120k-480k a o 1 hodnotu v rozsahu 900-1200. Hodnoty 120k-480k vznikají vynásobením úhlu získaného z gyroskopu 1000 (pro zachovaní 3 desetinných míst) a přičtením hodnoty 300000. Hodnota čísla 4 je pouze vynásobena 100 (nemůže nabývat záporných hodnot). Ve vysílači jsou hodnoty převedeny zpět do decimální soustavy a jsou od nich odečteny hodnoty, které byly v dronu přičteny. Tyto hodnoty jsou následně přesměrovány přes Bluetooth do aplikace v chytrém telefonu.

4. 2. 3. Komunikace mezi vysílačem a telefonem

V této kapitole si popíšeme komunikace mezi vysílačem a chytrým telefonem, respektive s aplikací pro chytré telefony. V tomto případě opět funguje architektura master-slave. Zde je slave chytrý telefon a master představuje vysílač.

Do telefonu jsou posílány hodnoty ve tvaru písmeno (označující o jakou hodnotu se jedná) = hodnota. Písmena jsou posílána do dronu celkem 4 (x, y, z označující konkrétní osu náklonu a v pro napětí na baterii). Za každou odeslanou hodnotou je umístěn znak pro nový řádek (podle tohoto znaku poznává parser v aplikaci, kdy má přestat číst aktuální hodnotu a přesunout se k další). Po uložení řetězců dochází k třídění podle písmene a uložení do proměnných. Celá tato akce probíhá ve vedlejším vláknu aplikace.

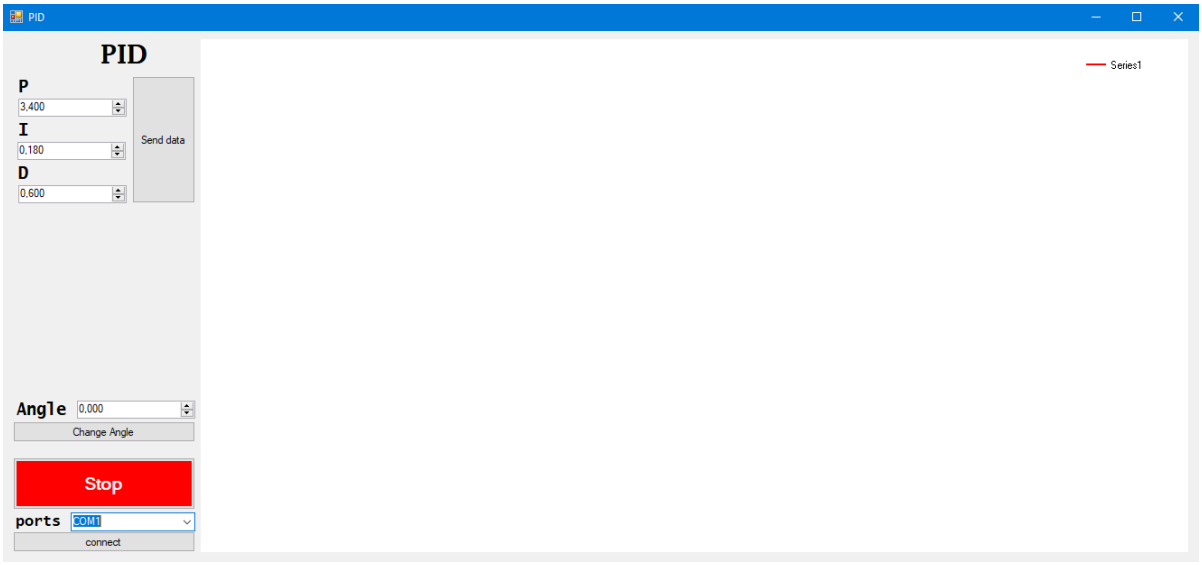
5. Vývoj ročníkové práce

5. 1. Aplikace na vývoj PID regulátoru

K vývoji PID konstant byla využita windows form aplikace napsaná ve vývojovém prostředí Visual Studio od firmy Microsoft. Pro vývoj byl použit programovací jazyk C#.

Grafická část aplikace se skládá ze tří polí pro PID konstanty a tlačítka pro odeslání. Dále je zde pole pro změnu požadovaného náklonu, tlačítko pro vypnutí motorů, seznam portů a panel pro vykreslování grafu.

Při spuštění aplikace načte seznam portů, ke kterým je možné se připojit a zobrazí je. Po připojení na port se spustí dvě vlákna pro odesílání a přijímání



software pro kalibraci PID

dat. Přijatá data jsou načtena jako řetězec a následně převedena na long. Hodnota se uloží do pole, které je následně použito jako seznam bodů pro graf.

Konstanty se převádějí do hexadecimální soustavy. Výhoda tohoto převodu je jeho konstantní délka a kompaktnost. Následně se z hodnot vytvoří paket, který je odeslán do kvadrokoptéry.

5. 2. Nákup komponent na AliExpressu

Vývoj ročníkové práce započal plánem vyrobit malou kvadrokoptéru, jejíž tělo bude vytisknuto na 3D tiskárně. Na Aliexpressu jsme nakoupili první díly potřebné pro osazení těla drona (8x micro DC motorů, 2x NRF24L01, 1x 3,7V Li-Pol Baterie 300mAh, 4x pěti palcové vrtule, 8x čtyř palcové vrtule). Problém nastal při zkoušení motorů, které byly příliš málo výkonné a nedokázali by unést kvadrokoptéru. Po tomto zjištění jsme se porozhlédli po alternativě v podobě BLDC, které se běžně používají. Po nakoupení čtyř ESC a BLDC jsme se rozhodli i pro koupi těla dronu. Ke koupi těla jsme se rozhodli hlavně z důvodu větší fyzické odolnosti oproti tělu vytisknutému na 3D tiskárně.

5. 3. Programování PID regulátoru

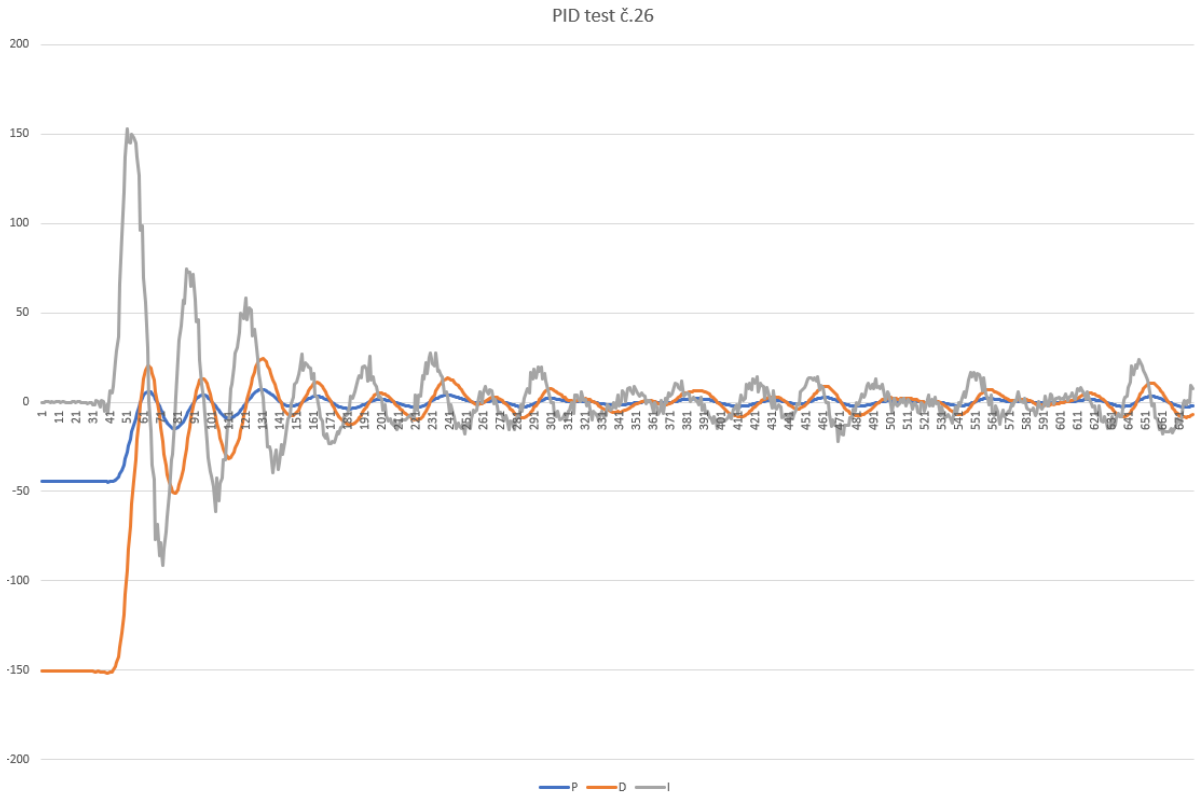
Vývoj konstant zesílení PID regulátoru započal sestavením trojúhelníkové konstrukce ze stavebnice Merkur. K této konstrukci byla připevněna kvadrokoptéra pomocí železné hřídele (obrazek xyz). Tato konstrukce

umožňovala kvadrokoptéře rotační pohyb v jedné ose. Celá konstrukce byla zatížena 10 kilogramy závaží jako zajištění. Manuální nastavování PID konstant začalo nastavením konstanty pro proporcionální složku na hodnotu jedna. Tuto hodnotu jsme poté zvyšovali. Tato metoda je známá jako Ziegler-Nicholsova metoda. Při použití této metody se zvyšuje konstanta proporcionální složky do té doby, než začne kvadrokoptéra konstantně oscilovat kolem požadovaného naklonění (v našem případě 0 stupňů). Použití této metody pro náš případ bylo nejspíš chybné, jelikož kvadrokoptéra nebyla schopná oscilace při jakékoli hodnotě konstanty proporcionální složky.

Houpačka pro kvadrokoptéru

Rozhodli jsme se proto pro manuální nastavování PID regulátoru neboli odhadování PID konstant podle vizuálního chování kvadrokoptéry. Nejdříve jsme hodnoty PID regulátoru vypisovali do sériové linky připojené k počítači. Následně jsme tato data ukládali do textových souborů, které jsme zpracovávali v Microsoft Excelu do grafů. Takto jsme udělali přes 60 měření. Tento způsob měření nebyl moc pohodlný a byl zdlouhavý. Proto jsme se rozhodli ulehčit si práci pomocí desktopové aplikace napsané v jazyce C#, která nám v reálném čase vykreslovala graf chyby PID regulátoru. Následně jsme tuto aplikaci upravili i k vzdálenému nastavování konstant, které jsme do té doby přepisovali v softwaru. Tímto

způsobem jsme postupovali, dokud jsme nedocílili stabilního vyrovnání kvadrokoptéry.



graf PID

5. 4. Problémy s komunikačními moduly

Naší první volbou pro bezdrátovou komunikaci mezi dronem a vysílačem byl modul NRF24L01, který sliboval dosah až 100m za velmi příznivou cenu kolem 1$. NRF24L01 komunikuje přes sběrnici SPI a je napájen 3.3V. Bohužel se nám, i přes několik pokusů, nepodařilo modul zprovoznit. Po tomto neúspěchu jsme začali pátrat po alternativě v podobné cenové relaci a narazili jsme na HC-12. HC-12 se osvědčila svou jednoduchostí a zároveň spolehlivostí.

5. 5. Vývoj ovladače a komunikace

Vývoj komunikace probíhal nezávisle na vývoji PID regulátoru. Vývoj jsem začal rozebráním a předěláním vysílače. Z vysílače jsem odstranil původní ovládací desku a nahradil jsem jí mikrořadičem STM32. K této desce jsem přivedl výstup z potenciometrů, které jsou připojeny k páčkám. Po jejich připojení jsem si naměřil maximální a minimální dosažitelné hodnoty, které jsem následně přepočítával do procentuálního náklonu. Procentuální náklon se posílal do kvadrokoptéry přes HC-12 připojenou k Serialu 2. Bohužel jsem si svou

nešikovností zamezil přístup k usb portu na desce, a tudíž jsem musel debugovat přes modul HC-06, který je připojený na Seriál 3. Nejdříve jsem začal programovat komunikaci s ideou, že vysílač bude master a kvadrokoptéra slave, ale od tohoto nápadu jsem brzy upustil kvůli špatné synchorinzaci komunikace, častým výpadkům a nespolehlivosti celého konceptu.

Pro vývoj komunikace jsem si postavil a naprogramoval emulátor kvadrokoptéry, který simuloval komunikaci mezi opravdovou kvadrokoptérou a vysílačem. Emulátor se skládal z mikrořadiče STM32 a komunikačního modulu HC-12.

5. 6. Jak jsme si šli poprvé zalétat, ale nic nefungovalo

Po spojení našich částí práce jsme se odebrali na Vypich, kde jsme měli otestovat letové schopnosti kvadrokoptéry. Po zapnutí vysílače se vysílač nebyl schopen připojit. Nejdříve jsme si mysleli, že jsou slabé baterie v ovladači, ale nebyly. Po tomto nepříjemném zjištění jsme se odebrali k Jiřímu domu, kde jsme pomalu začali debugovat program. Chybu jsme našli v inicializaci gyroskopu, která se zasekla, a tak nemohla proběhnout inicializace komunikace. Problém se týkal přetékajícího FIFO bufferu, který má za úkol parsovat data přijatá z gyroskopu. Po přidání resetovacího bodu pro FIFO buffer se software přestal zasekávat. Problém s připojením vysílače však přetrval. Po delším zkoumání bylo zjištěno že knihovna STM32duino má chybně nazvané seriál objekty, číslovaní pinů TX a RX neodpovídalo číslovaní objektů seriál v knihovně STM32duino.

dron na zkoušce

Po zprovoznění komunikace a ovládání kvadrokoptéry jsme se rozhodli pro první testovací let. Prvním zjištěným problémem bylo převrácení páčky pro náklon do stran, tudíž na pohyb páčky doleva kvadrokoptéra reagovala náklonem doprava. Druhotným problémem bylo zjištění že kvadrokoptéra se ve vzduchu chová úplně odlišně nežli připevněná k trojúhelníkové konstrukci. Několik po sobě jdoucích letů vyústilo k pár nehezkým pádům, při kterých se zlomily dvě vrtule.

5. 7. První úspěšný vzlet

Po měsíci upravování PID konstant a zrychlování kódu se povedl první samostatný let. Kvadrokoptéra se nejprve nad zemí potácela ale po přidání výkonu se vznesla a vyrovnala.

6. Aplikace pro chytrý telefon

V této kapitole budeme hovořit o vývoji aplikace pro chytrý telefon, jejíž úkolem je zobrazovat data přijatá přes Bluetooth z vysílače. Aplikace se dělí do tří částí. V první části nalezneme jednoduchý formulář, umožňující se připojit k vysílači a k následnému zahájení komunikace. V druhé části nalezneme log, ve kterém můžeme vidět data přijatá z dronu. Ve třetí části aplikace je grafické zpracování a vyhodnocení dat přijatých z dronu. Aplikace obsahuje jednoduché menu pro přepínání mezi jednotlivými částmi aplikace.

6. 1. Technologie použité při vývoji aplikace

Jako vývojové prostředí jsem zvolil Android Studio od firmy JetBrains. Pro vývoj byly použity programovací jazyky Kotlin a Java.

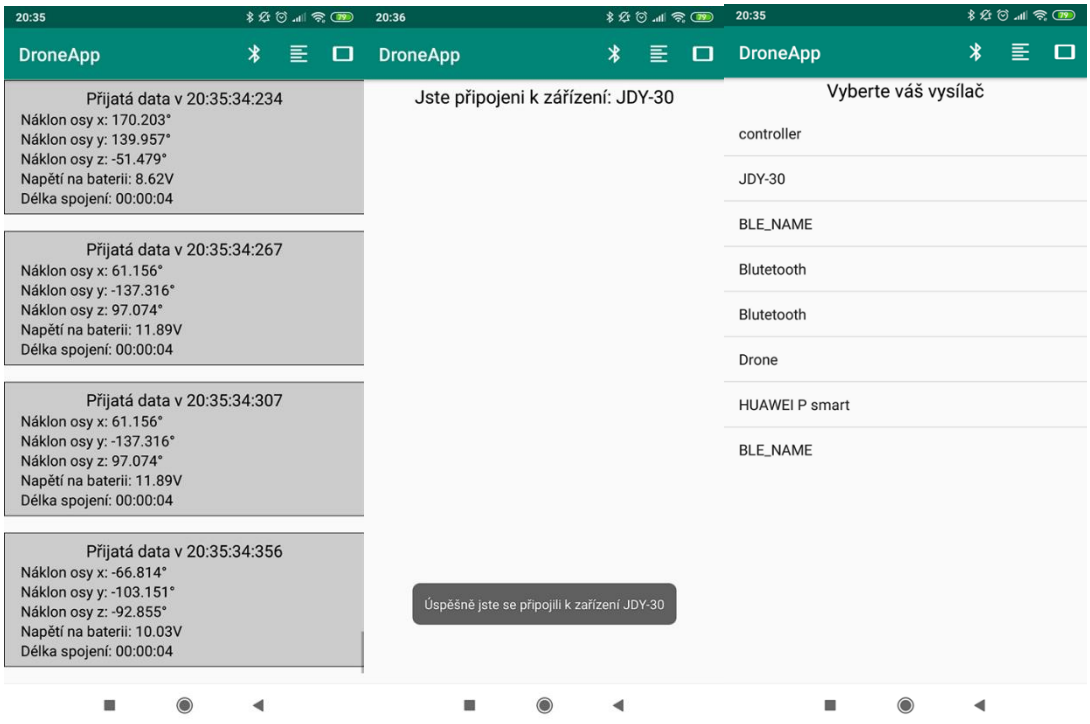
6. 2. Komunikační část aplikace

V první části aplikace se odehrává připojení k vysílači. Pokud není telefon připojen k vysílači, nemůže se uživatel odebrat do jiných částí aplikace. Uživatel si zde může vybrat ze zařízení, která jsou s jeho telefonem spárovaná. Pokud není uživatel s ovladačem spárován, musí to nejdříve provést. Pokud má uživatel svůj telefon již spárován, může vidět ovladač mezi nabízenými možnostmi připojení. Po kliknutí na vybrané zařízení se spustí druhé vlákno, které se zkusí na dané zařízení připojit. Po krátké chvilce oznámí aplikace uživateli, zdali se připojil k vybranému zařízení. Pokud k tomu nedojde, může se uživatel pokusit navázat

spojení znovu. Pokud ano, spustí se druhé vlákno, které přijímá a parsuje data z vysílače. Nyní se uživatel může odebrat i do dalších částí aplikace.

6. 3. Logovací část aplikace

V této části aplikace uživatel vidí příchozí informace z vysílače. Tyto informace jsou ukázány v přehledném boxu, ve kterém je navíc přesný čas a délka spojení s vysílačem. Každých cca 100 milisekund se přidá jeden box s aktuálními informacemi. Uživatel těchto boxů jich může vidět maximálně 2000. Neustále dochází k mazání těchto boxů, aby nedocházelo ke zpomalování aplikace. Uživatel si může jednotlivé boxy proscrollovat, nebo si zapnout autoscroll, aby viděl vždy nejaktuálnější informace. Přidávání nových boxů je v aplikaci zpomaleno na rychlost příjemnou pro uživatele.



screen aplikace 1 *screen aplikace 2* *screen aplikace 3*

6. 4. Grafická část aplikace

Tato část ročníkové práce slouží uživateli k zobrazovaní dat, která dostává aplikace pomocí bluetooth přijímače. Aplikace byla naprogramována pomocí vývojového prostředí Android Studio, které je vyvíjeno firmou JetBrains. Aplikace je kompatibilní se zařízeními, jejichž operační systém je Android verze 5.0 Lollipop a vyšší. Mobilní aplikace pomocí bluetooth přijímače dostává, a následně zpracovává náklon kvadrokoptéry a stav její baterie. Hlavní třídou

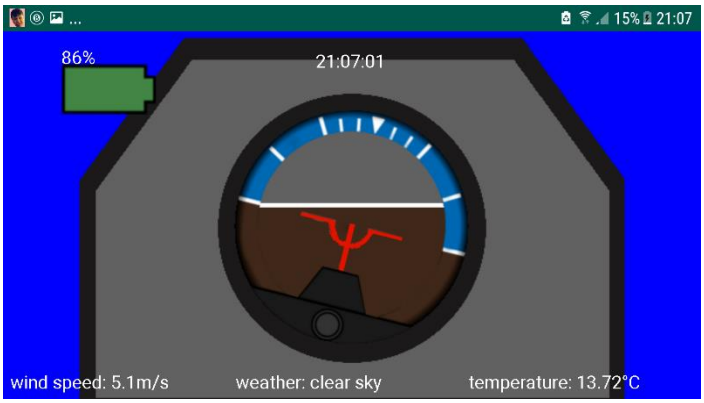
aplikace, která spravuje uživatelské rozhraní je třída GraphicActivity, která řídí celý chod aplikace. Grafické rozhraní aplikace je vykreslováno pomocí třídy MyCanvas. Třída Weather získává API z webové stránky OpenWeatherMap. V aplikaci se dále nachízí Log, s jehož pomocí jsou souhrnně vypsány všechny hodnoty. Třída MainActivity zobrazuje, zda je zařízení připojeno pomocí bluetooth s kvadrokoptérou.

6. 4. 1. GraphicActivity

Tato třída spravuje UI (User Interface – uživatelské rozhraní), zároveň spouští třídy Weather a MyCanvas. Metoda askForPermission zjišťuje, zda je aplikaci uživatelem dovoleno využívat internetové připojení.

6. 4. 2. MyCanvas

Tato třída rozšiřuje třídu View, díky tomu je zobrazitelná v uživatelském rozhraní. Grafika je složena z obrázků, které jsou načteny do jednotlivých bitmap, pomocí kterých jsou následně vykresleny. Dále se v této třídě vykreslují další údaje: stav baterie kvadrokoptéry a její náklon. Náklon je znázorňován na tzv. umělém horizontu, který je vytvořen pomocí dvou bitmap, kde bitmapa, která znázorňuje horizont, se pohybuje po ose y. Druhou částí umělého náklonu je bitmapa reprezentující kvadrokoptéru. Tato bitmapa je otáčena do daného úhlu pomocí matice. Jelikož vývoj aplikace probíhal na jednom zařízení je nutné přepočítat velikosti bitmap pro ostatní mobilní zařízení. Přepočet velikostí probíhá poměrem velikosti displeje telefonu na kterém je aplikace spuštěna a velikosti displeje telefonu na kterém aplikace byla vyvíjena (Samsung A5 2016).



6. 4. 3. Weather

Tato třída má za úkol získávat data o počasí. Třída Weather rozšiřuje třídu AsyncTask. AsyncTask umožňuje spuštění kódu na vedlejším vlákně. Rozšíření třídy AsyncTask ve třídě Weather je potřebné z důvodu ochrany před spadnutím aplikace, které by nastalo, kdyby mobilní zařízení nebylo připojené k internetu a

aplikace se pokusila na hlavním vlákně (vlákně kde běží uživatelské rozhraní) připojit k internetu. Ve třídě Weather běží funkce onBackgroud současně s hlavním vláknem, poté se zde nachází metoda onPostExecude, která výsledky metody onBackground předává třídě, ve které běží uživatelské rozhraní. Třída Weather v metodě onBackground získává API ve formátu JSON (JavaScript Object Nation), které následně parsuje a ukládá do příslušných Stringů.

7. Závěr

7. 1. Cíl ročníkové práce

Cílem naší ročníkové práce bylo sestavit a naprogramovat kvadrokoptéru, která bude vyrovnávat svou pozici a bude řízena vlastně sestaveným vysílačem. Kvadrokoptéra také bude posílat data do ovladače, který je bude posílat do ovladače. Myslíme si, že jsme splnili zadaní krom několika drobných bodů, které jsme upravili či vynechali, jako například tisknutí těla kvadrokoptéry na 3D tiskárně.

7. 2. Pozorování při vývoji

Při vývoji kvadrokoptéry jsme pozorovali proudění vzduchu která ovlivňovala let. Při příliš rychlém klesání kvadrokoptéry, víry vzduchu způsobené rotory mění směr tahu který není PID regulátor schopen změnit. Další pozorování vzdušných vírů bylo při vzletu kvadrokoptéry. Při roztočení rotorů na zemi se vzduch zaráží na zemi a tvoří další vzduchové víry. Tato situace lze vyřešit rychlým vzletem při nebo připevněním vzletových noh pod každý motor.

7. 3. Další vývoj

Další vývoj kvadrokoptéry prozatím neplánujeme, ale ani ho v budoucnu nevylučujeme. Vývoj kvadrokoptéry by se mohl zaobírat více na stabilitu a udržování polohy. Kvadrokoptéra je nyní připravena na udržování stabilní výšky které bude zajišťováno PID regulátorem který bude pracovat se sensorem na měření barometrického tlaku. Pro udržování polohy kvadrokoptéry by mohl být použit sensor s měřením optického toku obrazu. Tento sensor komunikuje po I2C sběrnici. Další možností je výpočet odchylky od původní pozice pomocí již použitého akcelerometru.

PX4FLOW sensor optického toku

8. Zdroje

- https://wiki.stm32duino.com/index.php?title=Blue_Pill – Vytvořil Racemaniac, přečteno 9. 5. 2019 (popis STM32)
- https://www.researchgate.net/figure/Accelerometer-noise-and-gyroscope-drift-in-static-conditions-y-axis-plotted-over-time_fig2_325700743 - přečteno 8. 4. 2019 (obrázek šumu a driftu)
- http://hustejvercajk.cz/2016/03/07/bezuhlikove-motory/ - vytvořil Jenda, přečteno 8. 4. 2019 (výhody a nevýhody motoru)
- https://cs.wikipedia.org/wiki/Lithium-polymerov%C3%BD_akumul%C3%A1tor – naposledy editoval Garyczek, přečteno 9. 5. 2019 (popis lithium-polymerových akumulátorů)
- https://www.rc-zoom.cz/jak-se-vyznat-v-oznaceni-vrtuli/ - přečteno 8. 5. 2019 (vysvětlení značení vrtulí)
- https://developer.android.com/docs – 4.5.2019 – dokumentace Android tříd
- https://developer.android.com/reference/android/graphics/Canvas – 1.5.2019 – dokumentace Canvas objektu
- https://developer.android.com/reference/android/os/AsyncTask – 3.5.2019 – dokumentace AsyncTask
- https://openweathermap.org/current – 8.5.2019 – dokumentace a poskytovatel API o počasí
- https://en.wikipedia.org/wiki/PID_controller- poslední edit 2. 5. 2019, přečteno 8. 5. 2019 (popis PID)
- https://cs.wikipedia.org/wiki/PID_regul%C3%A1tor - poslední edit 17. 4. 2019, přečteno 8. 5. 2019 (popis PID)
- https://www.i2cdevlib.com/forums/topic/91-how-to-decide-gyro-and-accelerometer-offsett/ - vytvořil HipT 14. prosince 2013, přečteno 8. 5. 2019 (gyro meření)
- https://forum.arduino.cc/index.php?topic=535717.0 – vytvořil nickgammon 5. 3. 2016, přečteno 8. 5. 2019 (gyro meření)
- https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)#/media/File:Data_Queue.svg – vytvořil Vegpuff 17. 8. 2009, přečteno 9. 5. 2019(obrázek FIFO buffer)
- https://github.com/jrowberg/i2cdevlib/blob/master/Arduino/MPU6050/MPU6050_6Axis_MotionApps20.h – poslední pull request – jrowberg 29. 9. 2018, přečteno 9. 5. 2019 (Otík se učil, jak vlastně funguje GYRO)
- http://www.brokking.net/ymfc-3d_main.html - vytvořil Jopp Brokking – 28.2.2019- inspirace ovládání kvadrokoptéry
- https://github.com/lobodol/drone-flight-controller - vytvořil lobodol - 10.4.2019 – ovládání kvadrokoptéry pomocí arduina
- https://github.com/jostoehr/Arduino-Copter - vytvořil jostoehr - 1.4.2019 – použití PID regulátoru na mikrořadiči arduino
- https://robotics.stackexchange.com/questions/167/what-are-good-strategies-for-tuning-pid-loops - 12.3.2019 - vývoj PID regulátoru a jeho nastavení
- https://www.crossco.com/blog/basics-tuning-pid-loops - 11.3.2019 - vývoj PID regulátoru a jeho nastavení

- https://www.omega.co.uk/technical-learning/tuning-a-pid-controller.html - 15.3.2019 - odhadování kostant PID regulátoru
- http://www.electronoobs.com/eng_robotica_tut6.php - 10.3.2019 - vývoj PID regulátoru pro Arduino
- http://www2.metso.com/MetsoTutor.aspx - 10.2.2019 - jak funguje PID regulátor
- https://www.elprocus.com/electronic-speed-control-esc-working-applications/ - 11.1.2019 – použití a ovládání ESC
- https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/ - 9.1.2019 – jak funguje bezuhlíkatý motor a ESC
- https://ae01.alicdn.com/kf/HTB19XQWJVXXXXbzXFXXq6xXFXXXO.jpg – 10. 5. 2019 (obrázek motoru)
- http://img.dxcdn.com/productimages/sku_154602_1.jpg – 10. 5. 2019 (obrázek MPU6050)
- (obrázek 1) https://s3-ap-southeast-1.amazonaws.com/a2.datacaciques.com/00/NDAy/17/12/16/9e54ifgdt2zm4pk9/6fec5ff2017d7c73.jpg – 10. 5. 2019 (obrázek STM32)
- https://ae01.alicdn.com/kf/HTB1XMqRNAPoK1RjSZKbq6x1IXXa4.jpg – 10. 5. 2019 (obrázek baterie)
- https://microtronicslab.com/wp-content/uploads/2018/03/23-600x600.jpg – 10. 5. 2019 (obrázek HC-12)
- https://img.staticbg.com/thumb/large/oaupload/banggood/images/09/2F/e7d1f570-4476-48a2-a599-45bc1d55043d.jpg – 10. 5. 2019 (obrázek HC-06)
- https://ae01.alicdn.com/kf/HTB1JNVfKFXXXXcdXVXXq6xXFXXX3/F450-ATF-Quadcopter-Frame-Kit-DJI-920KV-Brushless-Motor-DYS-Simonk-30A-ESC.jpg – 10. 5. 2019 (obrázek F450)
- https://robu.in/wp-content/uploads/2016/04/Free-Shipping-RC-BEC-30A-ESC-Motor-Speed-Controller-RC-Brushless-ESC-30-A-Hot-Sale-1.jpg_640x640-1.jpg – 10. 5. 2019 (obrázek ESC)
- https://ae01.alicdn.com/kf/HTB1wejXQVXXXXakXXXXq6xXFXXXV/1Pair-1045-10x4-5-CW-CCW-Propeller-Prop-For-RC-Multicopter-Quadcopter-F450-Black-White-Orange.jpg_640x640.jpg – 10. 5. 2019 (obrázek vrtule)
- http://www.chrobotics.com/wp-content/uploads/2012/11/Inertial-Frame.png – 10. 5. 2019 (obrázek os – pitch, roll, yaw)
- https://wikimedia.org/api/rest_v1/media/math/render/svg/69dd283bfc91f00c04a8fc80310e10cfd4bb4207 – 10. 5. 2019 (obrázek P regulátoru)
- https://wikimedia.org/api/rest_v1/media/math/render/svg/b0b74a3b7f09d0cabfa5305f4f72f215273ced5e – 10. 5. 2019(obrázek I regulátoru)
- https://wikimedia.org/api/rest_v1/media/math/render/svg/ac50845879ea867a54b5adfbb26e1962a466e6d9 – 10. 5. 2019(obrázek D regulátoru)
- http://www.sdtstore.com/WebRoot/StoreES3/Shops/82435372/5B97/E2C1/B4E8/BAEE/F484/0A0C/6D00/D60E/Captura.JPG - 10.5.2019(obrázek PX4FLOW senosor optického toku)