Stavba a Programování Dronu - Dokumentace

Studenti: Jan Vorel, Filip Hanzlík

Vedoucí profesor: Ing. Jindřich Koubek, Ph.D.

Datum: 04.12.2022

**Cíle a Motivace Projektu:**

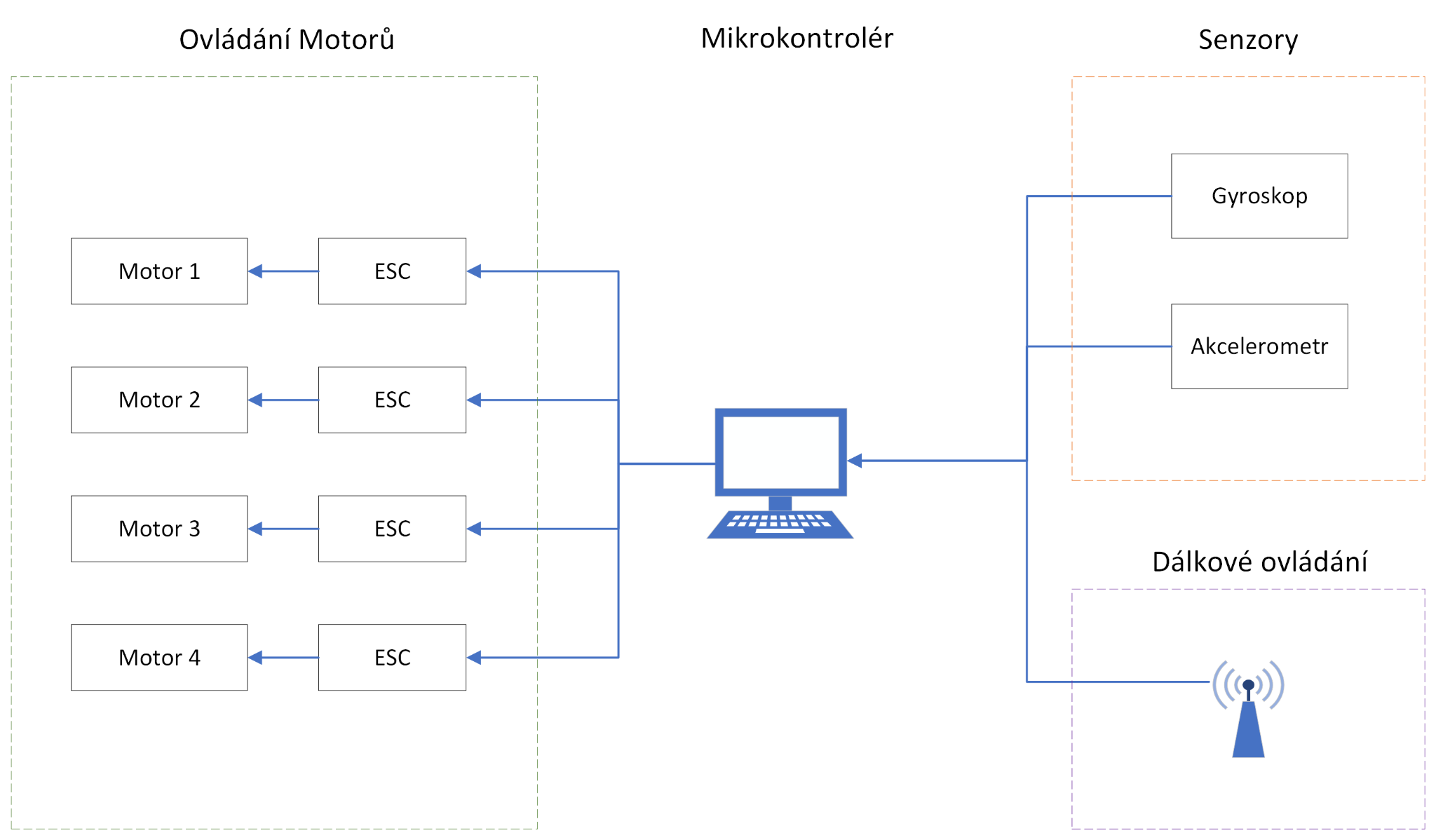
V dnešní době drony rapidně rostou na popularitě a jejich vývoj neustále přináší nové možnosti. Drony se tak stávají velmi atraktivní oblastí pro výzkum a to i na nižší úrovni, což je reflektováno i jejich častější přítomností v médiích. Co drony tvoří tak unikátními, je nejen výborná vzdušná mobilita a ovládání, ale především možnost nadstavby softwarem, jež dále výrazně rozšiřuje využití těchto přístrojů. Již aktuálně, díky implementaci systémů pro snímání obrazů, automatického předurčování trajektorie a vzájemné koordinace více dronů, je jejich uplatnění velmi široké, a to i v komerční sféře. Jedná se například o mapování a analyzování nedostupných teritorií, pořizování videozáznamů z těžko dostupných vzdušných úhlů, automatické doručování zásilek, aplikace v armádě a mnoho dalších využití. Jde tedy o technologii se zdánlivě velkým a rozvíjejícím se potenciálem.

Cílem projektu bylo tedy, jak mu název napovídá, sestrojení letu schopného dronu, a to ze základních elektronických součástek a senzorů spolu s vytvořením softwaru pro jeho ovládání. Tento projekt tedy měl důraz na programovací aspekty práce s dronem, zahrnující zejména práci s ovládáním mikrokontroléru, sběr dat ze senzorů/přijímačů, patřičné ovládání motorů a správné zpracování všech těchto dat, pro jejich finální implementaci v algoritmu, který bude provádět korekce v stavu dronu. Jako primární cíl toho projektu jsme definovali sestrojení dronu, jež bude schopný na základě dálkového ovládání, udržet konstantní vzdušnou pozici po dobu alespoň jedné minuty, protože dron schopný tohoto úkol bude schopný i letu. Takto již sestavený dron by nadále mohl být základem pro vývoj dalších funkcí, které by pak už měli aplikovatelné využití. Jedná se například o instalaci kamery, z níž by mohla být zpracována data, která by dále sloužila k různým aplikacím, jako je například vytváření 3D modelů prozkoumaných oblastí, nebo detekci objektů, či lidí. Dále, vedle samotného vývoje dronu, chceme aby tento projekt sloužil ostatním zájemcům, například studentům naší školy, pro poznání toho jak takový dron funguje a co všechno obnáší jeho sestavení, tato dokumentace by tudíž mohla sloužit jako takový vstupný bod do tohoto oboru. Budeme tedy klást důraz na přehledný popis průběhu našeho projektu tak, aby co nejlépe informoval a doprovázel zájemce, začínající zkoumat fungování dronů. Pro splnění tohoto účelu jsou všechny napsané programy a vytvořené modely k dispozici na GitHubu pod tímto odkazem: <https://github.com/FilipHanzlik/Stavba-Dronu>.

**Úvod:**

Dron je dle definice bezpilotní letadlo, které může být řízeno na dálku nebo autonomně. V našem případě půjde pouze o dálkové řízení. Drony se obecně řadí mezi tzv. nestabliní systémy, to znamená, že potřebují aktivně korigovat svojí polohu. Tohoto je dosaženo pomocí mikrokontroléru, nejdůležitější součástí dronu. Mikrokontrolér dostává data o požadovaném stavu (orientace, náklon a rychlost) z dálkového ovládání poté a je porovná s aktuálním stavem, který zjistí z přítomných senzorů. Pomocí algoritmu PID (tento algoritmus bude podrobně popsaný v budoucí sekci) mikrokontrolér určí potřebnou rychlost otáčení jednotlivých motorů a tyto instrukce poté rozešle.

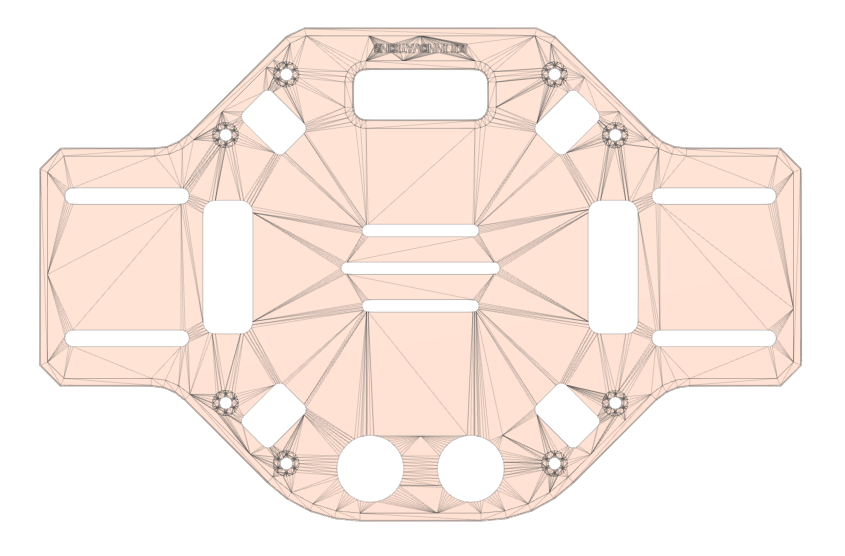
Celé fungování dronu je možné vidět na schématu níže. ESC zde značí elektronické regulátory otáček, pomocí nichž může mikrokontrolér ovládat jednotlivé motory.



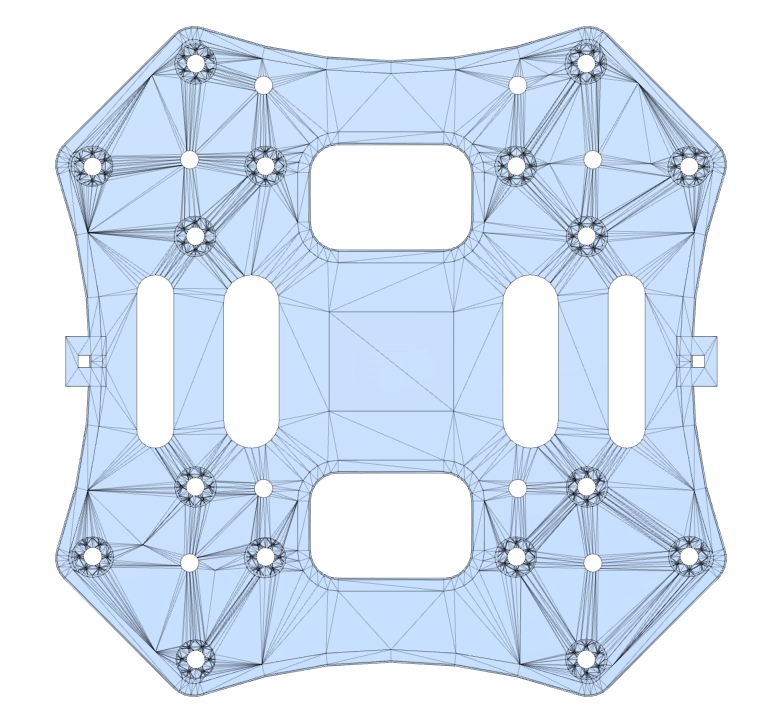
**Konstrukce:**

Celý tento systém, který je odpovědný za fungování dronu, potřebuje nosnou konstrukci. K tomuto jsme použili 3D modely, které jsme vytvořili na základě volně dostupných zdrojů z internetu. Tyto modely jsme pak vytiskli na 3D tiskárně z PETG. Celá konstrukce se tak skládá ze tří základních částí: z ramen, ze spodní desky a z horní desky.

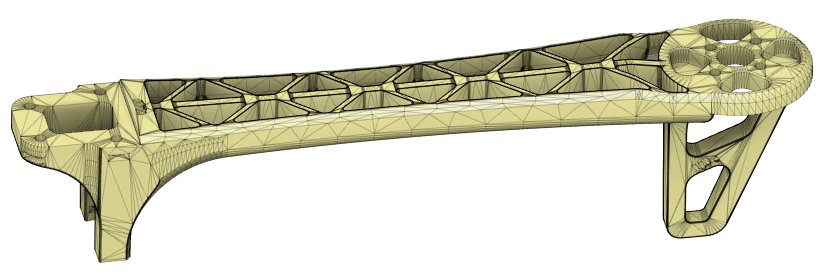
Model spodní desky je uveden níže. Na délku měří něco málo přes 18 cm a jeho šířka činí přibližně 12 cm. Její rozměry jsou nastaveny tak, aby se na tuto desku spolehlivě vešla baterka spolu s případním senzorem pro měření aktuální výšky (spodní dvě kulaté díry slouží k umístění tohoto senzoru).



Dále z vrchní strany přilehne vrchní deska o rozměrech 10.7 x 10.7 cm. Tato deska, krom toho, že umožňuje držet celou konstrukci pohromadě, také slouží jako podpora pro mikrokontrolér a další ovládací elektroniku. Je to totiž právě na této desce co je přimontovaný mikrokontrolér, spolu s dalšími senzory včetně přijímače pro dálkové ovládání. Z tohoto místa tedy vychází veškeré řízení dronu. Model horní desky je opět možné vidět níže.

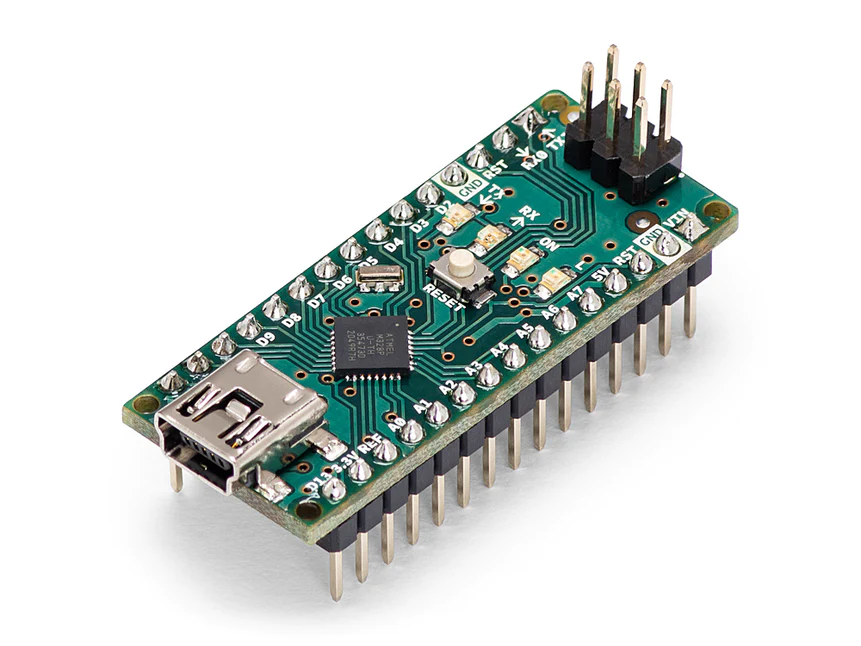


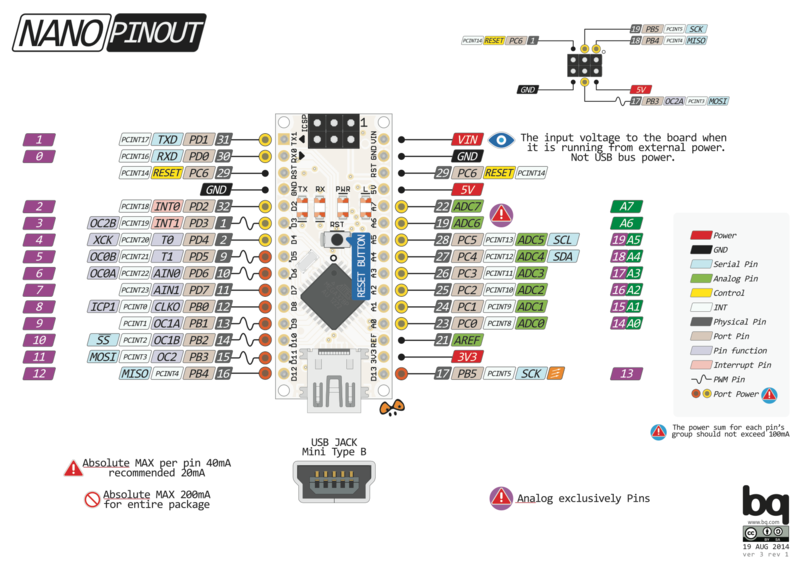
Poslední součástka, která je potřebná ve čtyřech kusech, je rameno. Ramena, s délkou 21.3 cm, jsou vždycky přišroubovaná mezi kraji horní a dolní desky. Jejich umístění je patrné z předpřipravených děr pro šrouby. Každé rameno pak na svém těle drží elektronický regulátor otáček (ESC), který je připojený na mikrokontrolér, přítomný na horní desce. Na regulátor je dál připojený motor. Motor je přišroubovaný na úplném konci ramene na patrně viditelném umístění. Model ramene je opět uveden níže.



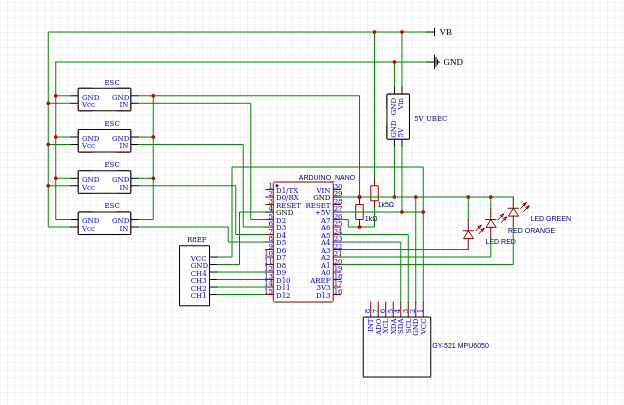
**Elektronika:**

Jak již už bylo zmíněno, nejdůležitější částí pro elektroniku je mikrokontrolér, který nese veškerý psaný software. Jako mikrokontrolér jsme si vybrali Arduino Nano a to zejména pro jeho malou velikost, nízkou cenu a hlavně velmi obsáhlou dokumentaci. Další možností by bylo Raspberry Pi Pico, to ale nemá tak rozsáhlou a přehlednou dokumentaci, a tudíž vývoj dronu by byl o něco komplikovanější a delší. Zároveň Arduino nabízí své vlastní vývojové prostředí ve kterém jsme mohli software napsat. Zde je obrázek toho, jak vypadá Arduino Nano. Spolu s označením jednotlivých pinů.





K tomu aby program na Arduinu Nanu mohl zpracovávat data ze senzorů a dálkového ovládání pro následné ovládání motorů, je nejprve zapotřebí připájet všechny komponenty k mikrokontroléru. Na tomto elektronickém schéma je znázorněné zapojení všech komponentů na našem dronu. Oproti předchozímu popisu jsou na schématu navíc přítomná led světla, která slouží jako indikátory stavu dronu (například pro skoro vybitou baterku). Dále je zde použit jeden regulátor napětí na 5V, jelikož port pro napojení na Arduinu vyžaduje 5V a v poslední řadě jsou zde ještě přítomné dva rezistory, které opět pomáhají s regulací napětí.



Jak je ze schématu patrné, kromě mikrokontroléru jsou na dronu přítomný ještě čtyři další různé komponenty, plus motory. Začněme od baterie. Celý tento systém samozřejmě potřebuje napájení a jelikož motory dokáží spotřebovat velké množství energie, vybrali jsme relativně velkou 11.1V baterii o kapacitě 4500 mah. Díky této baterii je možné létat s naším dronem po dobu přibližně patnácti minut.

Dalším komponentem je 3-osý akcelerometr a gyroskop GY-521 MPU-6050, díky němuž bude Arduino moct zpracovávat data o aktuální změně náklonu a zrychlení dronu. Vedle gyroskopu a akcelerometru dostává Arduino ještě data z přijímače na dálkové ovládání. Jedná se o přijímač R8EF pro vysílač Radiolink T8FB. Přenos dat z tohoto přijímače do Arduina probíhá pomocí metody PWM (Pulse-width modulation), která umožňuje efektivně přenášet analogový signál.

Poté co Arduino dostane a zpracuje všechna data vyšle instrukce do elektronických regulátorů otáček. V našem případě používáme jako elektronické regulátory otáček Chaos 20A. Každý tento regulátor pak obdržené instrukce zpracuje a pošle je dál motoru. Pro motory používáme bezkartáčové stejnosměrné motory XA2212/1400, které dokáží vyvinout víc než 10000 otáček za vteřinu.

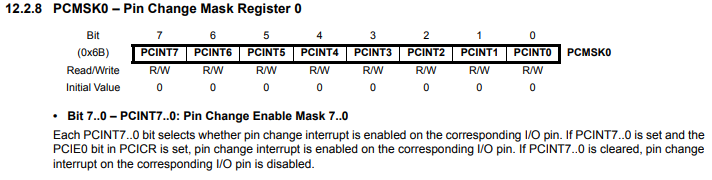
Software

Veškerý software odpovědný za ovládání dronu je napsaný v Arduino IDE, který využívá jazyk velmi podobný C++.

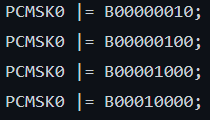
* Nastavení počátečních podmínek a čtení z přijímače

V této části programu dochází k nastavení stavu všech pinů na arduinu k tomu, aby byly schopni správně komunikovat s jednotlivými komponenty dronu. Poté dojde k nastavení počátečních parametrů pro komunikaci mezi komponentami a arduinem. K této komunikaci využíváme protokolu I2C. V případném zájmu o lepší pochopení této problematiky doporučuji shlédnout toto video: [https://www.youtube.com/watch?v=6IAkYpmA1DQ&t](https://www.youtube.com/watch?v=6IAkYpmA1DQ&t=313s).

Poté nastavíme piny, které jsou spojené s přijímačem pro dálkové ovládání. Při každém vyslání informace z přijímače pro dálkové ovládání dojde k zastavení hlavního cyklu, aby se mohla nová data uložit. Tomuto efektu se říká interrupt a při každém zavolání interuptu se spustí námi definovaná funkce, která se postará o uložení právě obdržených dat. V našem programu využíváme masku PCMSK0, která aktivuje piny PCINT0 až PCINT7. Níže je přiložen výstřižek z manuálu ATmega328P (mikročip, který Arduino Nano používá).



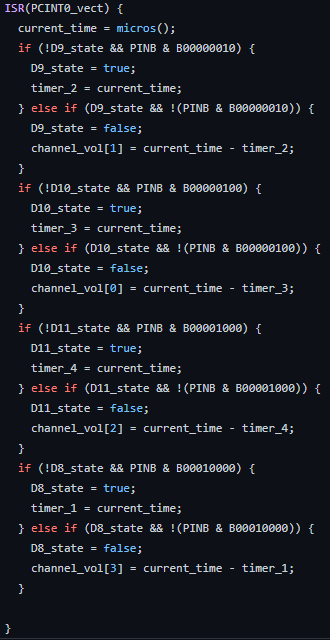
Jelikož jsme přijímač na dálkové ovládání připojili na piny PCINT1 až PCINT4, budou to právě tyto piny na něž nastavíme interrupty v programu.



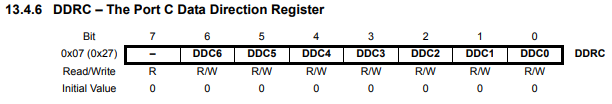
Dojde-li tedy k aktivaci interruptu, spustí se funkce, která se postará o zpracování dat z dálkového ovládání. Přijímač dálkového ovládání vysílá signál podobný Pulse-Width modulation (PWM). S frekvencí 50 Hz vysílá signál dlouhý 1000-2000 us, kde délka signálu reprezentuje to jak moc uživatel naklonil páčku na dálkovém ovládání. Tyto hodnoty pak uloží do seznamu *channel\_vol*.

Program u každého pinu postupuje následujícím způsobem:

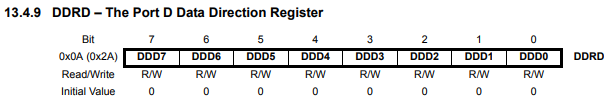
1. Zkontroluje jestli pin byl právě aktivován porovnáním hodnoty pinu a proměnné D#\_state (D#\_state zaznamenává, zda-li signál už začal), případně uloží počáteční čas signálu.
2. Nebo zkontroluje jestli signál byl ukončen, zrcadlově oproti kroku 1. Délka signálu je určena odečtením času ukončení od počátečního času, a proměnná D#\_state je nastavena zpátky na *false*, aby mohl být změřen příští signál. Délka signálu je uložena do seznamu *channel\_vol* pro budoucí využití.



Dále potřebujeme umožnit Arduinu ovládat motory a LED světla. Toho docílíme přepnutím pinů, na něž jsou tyto komponenty připojeny, do write módu. Piny pro ovládání LED žárovek jsou k nalezení v registru DDRC pod DDC1, DDC2 a DDC3.



A další čtyři piny, které potřebujeme pro ovládání motorů, můžeme najít v registru DDRD pod DDD2, DDD3, DDD4 a DDD5.

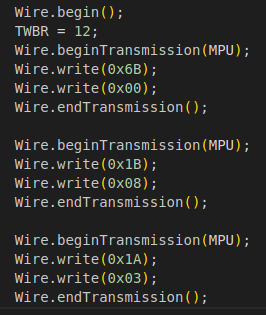


Abychom tyto piny nastavili na write mode, stačí abychom nastavili jejich hodnotu v registru z 0 na 1:

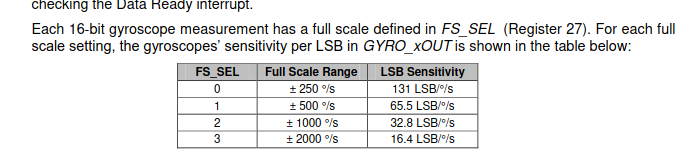


* Konfigurace I2C

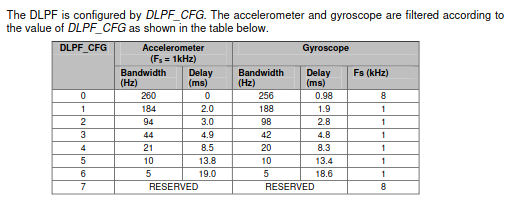
Nyní nám zbývá pouze nastavit konfiguraci pro komunikační protokol I2C. Toto zajišťuje tato část programu.



První řádek *Wire.begin()* zahájí komunikační protokol a poté *TWBR = 12* nastaví frekvenci komunikace na 400kHz. Sensor je resetován a u gyroskopu nastaví rozsah měření na ± 500 stupňů za vteřinu. Zde je výstřižek z dokumentace pro konfiguraci gyroskopu. (<https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf>)



V poslední fázi je ještě potřeba nastavit filtrování dat, abychom snížili šum z měřících přístrojů. Sensor má několik předem zpracovaných filtrů, které můžem využít. V našem případě použijeme filtr na frekvenci 42 Hz, protože je blízko k rotační rychlosti motorů.



Tímto jsou všechny potřebné konfigurace připraveny a je možné již začít psát samotný program pro ovládání dronu. V případě zájmu o podrobnější pochopení těchto konfigurací, doporučujeme se kouknout na dokumentace, které byly zmíněny v průběhu této sekce.

Hlavní loop:

Jak již už bylo zmíněno, nejpodstatnější část pro ovládání dronu probíhá v hlavním cyklu, který se neustále opakuje. Princip tohoto cyklu je následující:

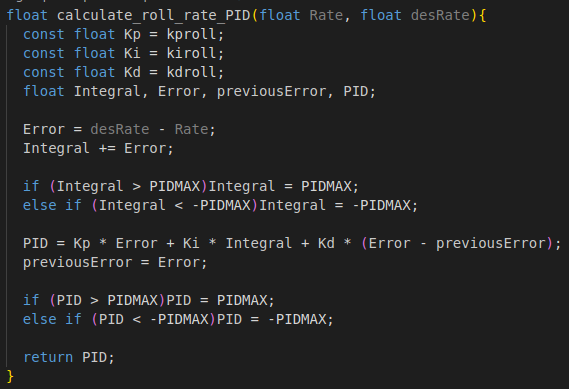
1. Data z gyroskopu a přijímače jsou sesbírána
2. Data jsou zpracována PID algoritmem, jehož výstup jsou hodnoty pro motory

PID algoritmus:

Nyní jsme se ocitli ve fázi, kde musíme data zpracovat tak, abychom mohli poslat korigující signál do motorů. PID algoritmus je velmi široce využívan díky jednoduchosti jeho implementace. Výsledná hodnota PID algoritmu se skládá ze tří složek, proporcionální, integrační a derivační (podle jejich iniciálů je utvořen název). Každá z nich zpracovává chybu v rotaci dronu oproti požadovaným hodnotám.  
 Proporcionální složka je nejjednodušší implementace. Chyba je vynásobena koeficientem a poslána do motorů. Pokud vnější působící síla překonává sílu kterou proporcionální část algoritmu vydává na základě aktuální chyby, může nastat situace, kdy systém není schopen snižovat chybu. K tomuto problému dochází jelikož proporcionální část algoritmu nedokáže kompenzovat vnější síly, ale pouze využívá vzdálenosti od referenční hodnoty.

Potřebujeme tedy složku schopnou započítat vliv těchto vnějších sil. Tohoto dosáhneme sumací celkové chyby (integrací). Pokud by se chyba nesnižovala, integrační složka algoritmu nabývá na hodnotě, což způsobuje zvýšení výkonu systému, který tak dosáhne dostatečné síly na překonání dříve zmíněných vnějších sil.

Poslední složka algoritmu se snaží předejít nadměrné korekci chyby, což by skončilo oscilací kolem referenční hodnoty, která by mohla zabránit přesnému dosažení referenční hodnoty. Zkoumáme tedy velikost změny v chybě (diferenciace). Je třeba zmínit, že tato část je velmi citlivá na šum v signálu, a proto buď musí používat filtrovaná data, anebo být odstraněna úplně (Potom by se ovládací proces nazýval pouze PI).



Kp, Ki a Kd jsou koeficienty složek. PIDMAX je nejvyšší hodnota kterou může integrační složka dosáhnout, aby nedošlo k překompenzaci a oscilacím. Zároveň je tento limit aplikován na konečnou hodnotu, aby nemohla dosáhnout příliš vysokých hodnot v