SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

MSP knihovna pro MicroPython

Autor: Tobiáš Hopp

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2025/26

Poděkování

Chtěl bych poděkovat Mgr. Marcelovi Godovskému za obstarání mikrořadiče Raspberry Pi Pico.

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 6. 1. 2026

. . . . . .

Podpis autora

Abstrakt

Tato práce popisuje implementaci softwarové knihovny v jazyce MicroPython, která zajišťuje komunikaci s letovými kontroléry (např. Betaflight, iNAV) prostřednictvím protokolu Multiwii Serial Protocol (MSP) v1. Řešení využívá rozhraní UART a bylo ověřeno na platformě Raspberry Pi Pico W. Vyvinuté API umožňuje zasílání příkazů a zpracování telemetrických dat. Knihovna obsahuje parsery pro čtení orientace stroje, stavu baterie, dat z IMU senzorů, GPS souřadnic a RC kanálů. Text dokumentuje strukturu kódu, hardwarové zapojení a integraci do uživatelských skriptů pro účely externího monitoringu dronů.

Klíčová slova

MicroPython, Multiwii Serial Protocol, UART, Raspberry Pi Pico W, Betaflight, telemetrie

Abstract

This thesis describes the implementation of a MicroPython software library that facilitates communication with flight controllers (e.g., Betaflight, iNAV) using the Multiwii Serial Protocol (MSP) v1. The solution utilizes the UART interface and was verified on the Raspberry Pi Pico W platform. The developed API enables command transmission and telemetry data processing. The library includes parsers for reading vehicle attitude, battery status, IMU sensor data, GPS coordinates, and RC channels. The text documents the code structure, hardware wiring, and integration into user scripts for external drone monitoring purposes.

Keywords

MicroPython, Multiwii Serial Protocol, UART, Raspberry Pi Pico W, Betaflight, telemetry

Obsah

Úvod

Hlavní motivací této práce byla snaha realizovat projekt, který by propojil oblast softwarového inženýrství s hardwarovou implementací. Jako ideální nápad mi přišlo projekt propojit s FPV (first person view) dronem, kterého jsem si v té době plánoval sestavit.

Původní záměr práce počítal s využitím již existujících softwarových řešení pro komunikaci s letovým kontrolérem. Důkladná rešerše dostupných knihoven v jazycích C a Python však odhalila, že většina veřejně dostupných projektů je zastaralá, postrádá potřebnou dokumentaci nebo již není autory aktivně udržována. Specificky pro prostředí MicroPython, které jsem plánoval použít, neexistovalo žádné komplexní řešení, které by umožňovalo spolehlivou integraci.

Na základě těchto zjištění byl cíl práce přeformulován. Místo pouhé aplikace existujících nástrojů se práce zaměřuje na návrh a implementaci vlastní, plně dokumentované knihovny pro komunikaci pomocí protokolu MSP. Součástí výstupu jsou rovněž praktické příklady použití, které slouží k ověření funkčnosti a usnadňují implementaci dalším uživatelům.

1 POUŽITÝ HARDWARE

Realizace této práce vyžadovala integraci několika hardwarových celků. Systém je rozdělen na dvě hlavní části: řízený objekt (FPV dron) a řídicí/monitorovací jednotku (mikrokontrolér s klientskou knihovnou). Tato kapitola podrobně popisuje technické parametry jednotlivých komponent, které jsou klíčové pro správnou funkci implementovaného komunikačního rozhraní MSP (Multiwii Serial Protocol).

1.1 FPV DRON

Pro účely testování a ověření softwarové knihovny byl zvolen bezpilotní letoun kategorie FPV (First Person View). Na rozdíl od běžných komerčních dronů, které jsou primárně určeny pro stabilizované natáčení videa a disponují pokročilou autonomií, jsou FPV drony konstruovány s důrazem na vysokou manévrovatelnost, nízkou latenci a manuální řízení.

Pilot ovládá stroj z pohledu „první osoby“ prostřednictvím videobrylí, do kterých je v reálném čase přenášen obraz z palubní kamery. Tyto stroje obvykle létají v tzv. „Acro“ režimu, který nestabilizuje náklon dronu automaticky, ale vyžaduje neustálé korekce ze strany pilota. Tato charakteristika klade vysoké nároky na rychlost odezvy řídicí elektroniky a efektivitu komunikačních protokolů.

Z hlediska této práce je FPV dron ideální platformou díky své otevřené architektuře. Většina moderních FPV dronů využívá firmware Betaflight (nebo jeho deriváty), který nativně podporuje protokol MSP pro externí konfiguraci a telemetrii.

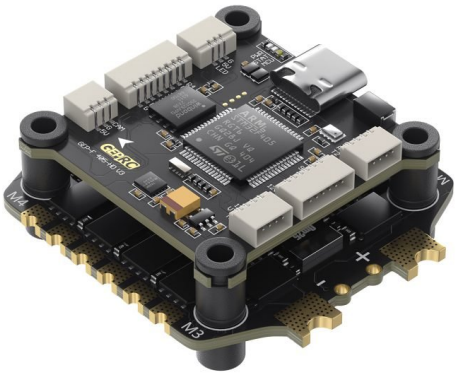
1.2 LETOVÝ KONTROLÉR

Jádrem avioniky použitého dronu je stack (kombinovaná jednotka řízení letu a regulace motorů) **GEPRC TAKER F405 BLS 65A V2**. Tato komponenta je zodpovědná za sběr dat ze senzorů, výpočet PID smyčky a generování řídicích signálů pro motory.

Pro komunikaci s externími zařízeními je rozhodující použitý mikrokontrolér a dostupnost sériových rozhraní. Deska je osazena čipem **STM32F405**, který poskytuje dostatečný výpočetní výkon pro běh letového firmwaru i obsluhu periférií.

Technická specifikace kontroléru

- **MCU (Microcontroller Unit):** STM32F405RGT6. Jedná se o výkonný 32-bitový ARM Cortex-M4 procesor, který zajišťuje logiku celého systému.
- **IMU (Inertial Measurement Unit):** ICM-42688-P. Tento čip kombinuje tříosý gyroskop a tříosý akcelerometr, jejichž data jsou nezbytná pro stabilizaci letu a jsou rovněž vyčítatelná pomocí vyvíjené knihovny.
- **Komunikační rozhraní:** Deska disponuje šesti hardwarovými UART porty (UART1 až UART6). To umožňuje připojení přijímače dálkového ovládání, VTX (video vysílače), GPS modulu a v našem případě i externího mikrokontroléru Raspberry Pi Pico.
- **Regulátor (ESC):** Integrovaný 4-v-1 regulátor s trvalým proudovým zatížením 65 A a firmwarem BLHeli_S, který přijímá povely z letového kontroléru prostřednictvím protokolu DSHOT.



Obrázek 1.1: GEPRC TAKER F405 BLS 65A V2

1.3 RASPBERRY PI PICO W

Jako hardwarová platforma pro běh vyvinuté MicroPython knihovny (klienta MSP) byl zvolen mikrokontrolér **Raspberry Pi Pico W**. Jedná se o kompaktní vývojovou desku postavenou na čipu RP2040, který byl vyvinut nadací Raspberry Pi.

Volba této platformy byla motivována několika faktory: nativní podporou jazyka MicroPython, dostatečným počtem hardwarových UART rozhraní a kompatibilitou logických úrovní s letovým kontrolérem. Verze „W“ navíc obsahuje bezdrátový modul, což do budoucna otevírá možnost bezdrátového přenosu telemetrických dat z dronu do PC bez nutnosti proprietárních rádiových linek.

Specifikace a propojení

- **GPIO a UART:** Deska operuje s logickou úrovní 3,3 V. To je klíčová vlastnost, protože mikrokontrolér STM32F405 na letovém kontroléru rovněž pracuje s logikou 3,3 V. Díky tomu je možné propojit komunikační linky (TX a RX) přímo drátově, bez nutnosti použití převodníků logických úrovní (level shifters), což zjednodušuje hardwarový návrh.
- **Konektivita:** Čip Infineon CYW43439 zajišťuje podporu bezdrátových sítí Wi-Fi (802.11n) v pásmu 2,4 GHz.

Propojení systému je realizováno křížovým zapojením sériových linek, kde vysílací pin (TX) Raspberry Pi Pico W je připojen na přijímací pin (RX) volného UART portu letového kontroléru a naopak.

2 POUŽITÉ SOFTWAROVÉ TECHNOLOGIE

Tato kapitola se věnuje softwarovým komponentám, které tvoří základ projektu. Propojuje vysokoúrovňové programování v jazyce Python s nízkoúrovňovým řízením letového kontroléru prostřednictvím standardizovaných komunikačních protokolů.

2.1 MICROPYTHON

MicroPython je implementace programovacího jazyka Python 3, která je optimalizována pro běh na mikrokontrolérech a v prostředích s omezenými zdroji.

V rámci tohoto projektu MicroPython umožňuje:

- **Rychlé prototypování:** Díky interpretovanému charakteru jazyka lze snadno testovat algoritmy pro ovládání dronu bez nutnosti zdlouhavé kompilace.
- **Přímý přístup k hardwaru:** Knihovna poskytuje nízkoúrovňové moduly pro obsluhu sběrnic UART, I2C a SPI, které jsou nezbytné pro komunikaci s letovým kontrolérem.
- **Správu paměti:** Automatický Garbage Collector (GC) pomáhá efektivně spravovat omezenou RAM mikrokontroléru během zpracování datových paketů.

2.2 BETAFLIGHT

Betaflight je open-source firmware pro řízení letu (flight control firmware), který se zaměřuje na výkon, stabilitu a moderní řídicí algoritmy. V tomto projektu Betaflight funguje jako nízkoúrovňový procesor, který obstarává:

- **PID regulaci:** Zajišťuje stabilizaci dronu v reálném čase na základě dat z gyroskopu a akcelerometru.
- **Mixování motorů:** Převádí abstraktní příkazy pro plyn, klonění, klopení a zatáčení na konkrétní otáčky jednotlivých motorů.
- **Konfigurační rozhraní:** Umožňuje nastavení parametrů pomocí příkazového řádku (CLI) nebo grafického konfigurátoru.

2.3 MultiWii Serial Protocol (MSP)

MSP protocol představuje páteřní komunikační vrstvu mezi MicroPython skriptem a firmwarem letového kontroléru. Jedná se o binární, paketově orientovaný protokol, který pracuje na principu *Master–Slave* (nebo také *Request–Response*).

2.3.1 Historie a evoluce

Protokol byl původně navržen pro projekt *MultiWii* (odtud název), který využíval senzory z herních ovladačů Nintendo Wii pro stabilizaci multikoptér na platformě Arduino. Díky své jednoduchosti a nízké režii jej adoptovaly následné projekty jako Baseflight, Cleanflight a nakonec **Betaflight**.

V průběhu let se protokol vyvinul do dvou hlavních verzí:

- **MSP V1:** Původní verze s 8-bitovou adresací příkazů (max. 255 ID) a 8-bitovou délkou datového pole. Tato verze je dodnes nejrozšířenější pro běžné telemetrické dotazy.
- **MSP V2:** Představena s nástupem komplexnějších systémů (jako iNav a novější Betaflight). V2 zavádí 16-bitová ID příkazů a umožňuje přenos mnohem větších datových objemů, což je nezbytné pro pokročilou konfiguraci moderních senzorů.

2.3.2 Struktura paketu

Každý paket má striktně definovanou strukturu, která zajišťuje integritu dat při přenosu přes nespolehlivé sériové linky (UART):

- **Hlavička ($M):** Identifikuje začátek MSP rámce.
- **Směr:** < pro příkazy jdoucí do kontroléru, > pro odpovědi z kontroléru.
- **Payload size:** Definice délky následujících dat.
- **Command ID:** Specifikuje typ zprávy (např. 105 pro `MSP_IDENT` nebo 200 pro `MSP_SET_RAW_RC`).
- **Checksum:** XOR součet všech bajtů (kromě hlavičky), sloužící k detekci chyb při přenosu.

2.3.3 Nedostatky a omezení

I přes svou popularitu má MSP protokol několik nevýhod, které musí vývojář knihovny pro MicroPython brát v potaz:

1. **Režie synchronní komunikace:** Standardní MSP je založen na dotazování. Pokud Master (MicroPython) nepošle dotaz, Slave (Betaflight) nepošle data. To může být limitující u aplikací vyžadujících extrémně vysokou obnovovací frekvenci (např. blesková reakce na překážku).

2. **Binární nekompatibilita:** Při přechodu mezi verzemi firmwaru se občas mění význam jednotlivých bajtů v datovém poli (payload), což vyžaduje neustálou údržbu knihovny a ověřování verzí API.

3. **Jednosměrný Checksum:** Jednoduchý XOR checksum není tak robustní jako CRC16 nebo CRC32, což v prostředí s vysokým elektromagnetickým rušením (blízkost regulátorů motorů) může vést k propuštění poškozeného paketu.

3 Principy fungování knihovny

Knihovna je navržena jako lehký a bezstavový klient pro protokol MSP v1. Její hlavní charakteristikou je, že interně neukládá žádná vyčtená data do dlouhodobé paměti mikrokontroléru, ale slouží výhradně jako komunikační rozhraní a nástroj pro dekódování příchozích zpráv.

3.1 Bezstavová architektura

Důležitým aspektem knihovny je její efektivita z hlediska využití paměti RAM. Třída `MicroPythonMSP` po své inicializaci neudržuje žádné vyrovnávací paměti (buffery) pro telemetrická data. Jakmile je zpráva přijata a zpracována, jsou výsledné hodnoty předány hlavnímu programu a knihovna je dále neeviduje. Tento přístup zajišťuje, že uživatel má vždy k dispozici aktuální data přímo z letového kontroléru bez rizika práce se zastaralými hodnotami.

3.2 Komunikační cyklus

Komunikace s letovým kontrolérem probíhá v synchronním režimu na principu dotaz–odpověď. Po vytvoření instance třídy a přiřazení UART portu je standardní pracovní postup následující:

1. **Odeslání požadavku:** Pomocí funkce `send_request()` se sestaví a odešle MSP paket. Funkce automaticky vypočítá kontrolní součet (XOR) a přidá hlavičku `$M<`.
2. **Čtení odpovědi:** Funkce `read_response()` naslouchá na sériové lince a hledá hlavičku odpovědi `$M>`. Pokud je nalezen platný paket se správným kontrolním součtem, vrátí funkce kód příkazu a neupravená data (payload).

3.3 Zpracování dat a parsování

Knihovna odděluje proces příjmu dat od jejich interpretace. Pro uživatelské zjednodušení obsahuje specializované metody pro parsování konkrétních typů zpráv, které převádějí binární data na srozumitelné datové typy.

- **Sémantické parsování:** Metody jako `parse_attitude()` nebo `parse_analog()` automaticky provádějí převody jednotek (např. dělení deseti pro získání stupňů nebo napětí), aby výstup odpovídal reálným fyzikálním hodnotám.
- **Flexibilita formátů:** Parsovací funkce počítají s drobnými rozdíly v implementaci MSP mezi firmwarem Betaflight a iNAV, což umožňuje širší nasazení knihovny bez nutnosti měnit zdrojový kód.

4 Implementace a použití knihovny

Tato kapitola popisuje proces instalace knihovny na cílový hardware a základní metodiku jejího použití pro řízení letového kontroléru.

4.1 Příprava hardwaru a nahrání knihovny

Pro běh knihovny je vyžadován mikrokontrolér s podporou MicroPythonu (např. Raspberry Pi Pico, ESP32) a letový kontrolér s firmwarem Betaflight, který má volný alespoň jeden UART port.

4.1.1 Instalace MicroPythonu

Před použitím knihovny je nutné do mikrokontroléru nahrát interpret MicroPython. Pro Raspberry Pi Pico se doporučuje použít nejnovější stabilní verzi `.uf2` souboru z oficiálních stránek micropython.org.

4.1.2 Nahrání souborů knihovny

Celá knihovna se všemi potřebnými soubory, které implementují nízkoúrovňovou komunikaci, se nachází ve složce `micropython_msp/` . Pro nahrání do zařízení postupujte následovně:

1. Připojte mikrokontrolér k PC pomocí USB kabelu.
2. Použijte IDE podporující MicroPython (např. VS Code se micropython pluginem).
3. Nahrajte složku s knihovnou `micropython_msp/` přímo do kořenového adresáře zařízení nebo do složky `lib/`.

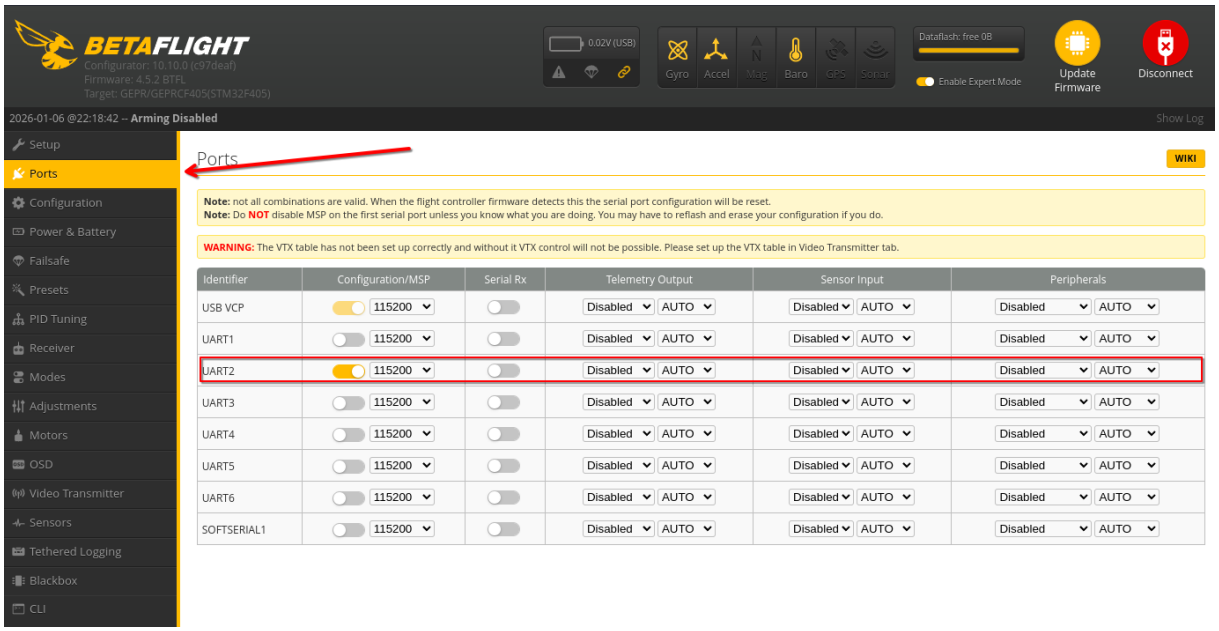
4.2 Konfigurace Betaflightu

Aby mohl letový kontrolér komunikovat s MicroPythonem, je nutné správně nastavit sériový port (UART) v aplikaci Betaflight Configurator.

4.2.1 Nastavení portů

V záložce **Ports** (Porty) identifikujte UART, ke kterému jste mikrokontrolér fyzicky připojili. Pro tento port je nutné:

1. Aktivovat přepínač v prvním sloupci s názvem **Configuration/MSP**.
2. Nastavit rychlost přenosu (**Baudrate**) na hodnotu `115200`. Tato rychlost musí odpovídat nastavení v MicroPython skriptu.
3. Kliknout na tlačítko **Save and Reboot** v pravém dolním rohu.

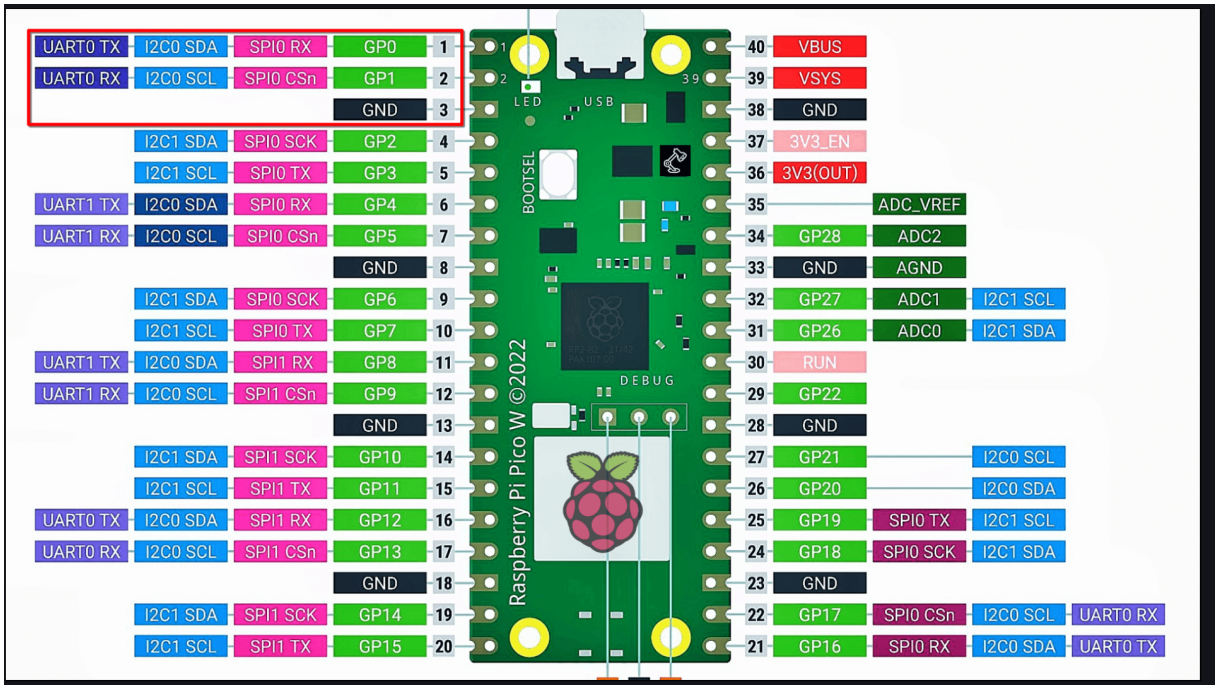


Obrázek 4.1: Nastavení protokolu MSP na portu UART2 v rozhraní Betaflight.

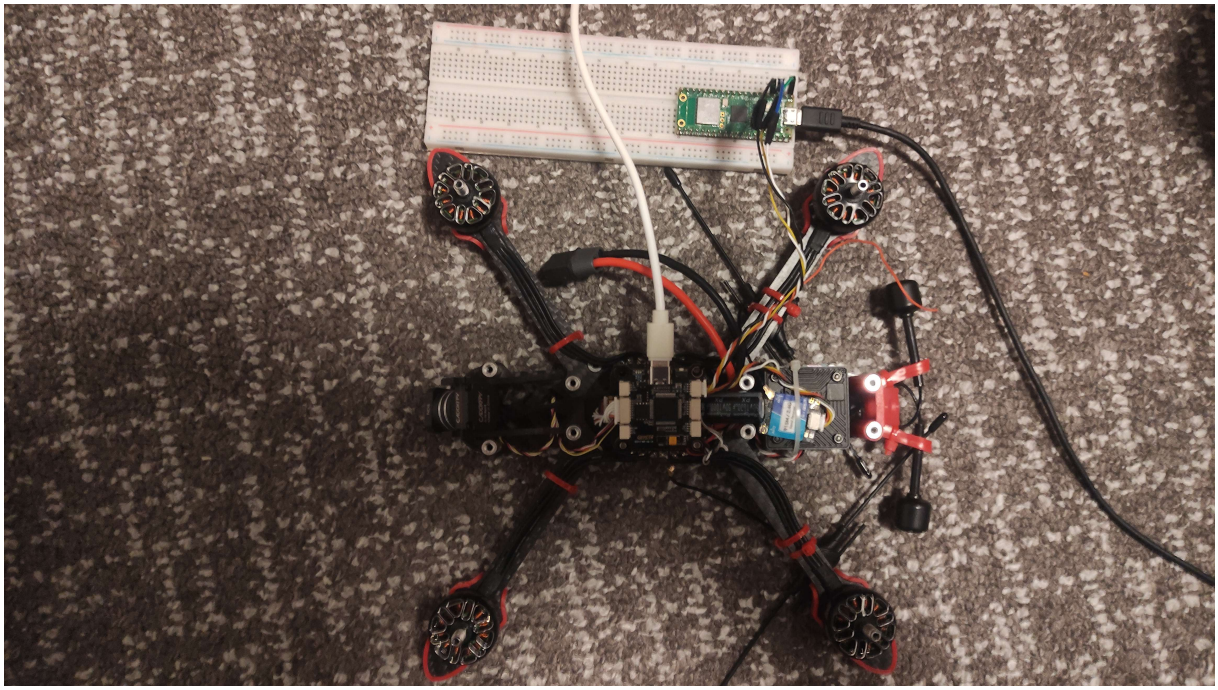
4.3 Zapojení

Komunikace probíhá přes sériové rozhraní UART. Je nezbytné propojit:

- **TX** mikrokontroléru na **RX** letového kontroléru.
- **RX** mikrokontroléru na **TX** letového kontroléru.
- **GND** obou zařízení (společná zem je kritická pro integritu signálu).



Obrázek 4.2: Raspberry Pi Pico pinout s vyznačením používaných pinů



Obrázek 4.3: FPV dron připojený k Raspberry Pi Picu

4.4 Základní příklad použití

Následující ukázka kódu demonstruje inicializaci knihovny a vyžádání základních informací o stavu dronu.

```
from machine import UART
from micropython_msp import MicroPythonMSP, MSP_ATTITUDE, MSP_STATUS

uart = UART(uart_id, baudrate=baudrate, timeout=10)
msp = MicroPythonMSP(uart)

msp.send_request(MSP_ATTITUDE)
        cmd, payload = msp.read_response(timeout_ms=10)

        if cmd is None:
            print("No valid response or checksum error")
        else:
            print("Received cmd:", cmd)
```

Kód 4.1: Ukázka implementace knihovny

4.5 Bezpečnostní doporučení

Při práci s touto knihovnou a reálným hardwarem je nutné dodržovat následující zásady:

- **Demontáž vrtulí:** Při testování komunikace a nastavování MSP příkazů **vždy** sundejte vrtule. Nezamýšlený příkaz `SET_RAW_RC` může aktivovat motory.
- **Failsafe:** Ujistěte se, že v Betaflightu máte nastavený korektní Failsafe pro případ ztráty spojení s MicroPythonem.
- **Baudrate:** Standardní rychlost pro MSP je 115200 baudů. Pokud nastavíte jinou v Betaflightu, musí se shodovat i v MicroPython skriptu.

ZÁVĚR

Předložená knihovna implementuje základní vrstvu protokolu MSP v prostředí MicroPython a umožňuje efektivní komunikaci mezi mikrokontrolérem a letovým kontrolérem s firmwarem Betaflight. Projekt v praxi ověřil funkčnost čtení telemetrických dat i základní programové řízení stroje. Existuje však značný prostor pro budoucí rozšíření, zejména v oblasti abstrakce složitějších funkcí pro zjednodušení uživatelského rozhraní, integrace asynchronního zpracování dat nebo podpory novějších verzí protokolu. Projekt tak tvoří funkční základ pro další vývoj autonomních systémů využívajících jazyk Python.

LITERATURA

[1] THECOGNIFLY. *YAMSPy: Yet Another Multiwii Serial Protocol (MSP) Python library* [online]. GitHub, 2026 [cit. 2026-01-06]. Dostupné z: https://github.com/thecognifly/YAMSPy

[2] DEEPWIKI. *MSP Serial Protocol: MultiWii Firmware* [online]. 2026 [cit. 2026-01-06]. Dostupné z: https://deepwiki.com/multiwii/multiwii-firmware/7.1-msp-serial-protocol

[3] GOOGLE. *Gemini* [online]. 2026 [cit. 2026-01-06]. Dostupné z: https://gemini.google.com

[4] GEPRC. *GEPRC TAKER F405 BLS 65A V2 STACK Specifications* [online]. 2026 [cit. 2026-01-12]. Dostupné z: https://geprc.com/product/geprc-taker-f405-bls-65a-v2-stack/

Seznam obrázků