RC Ultimate

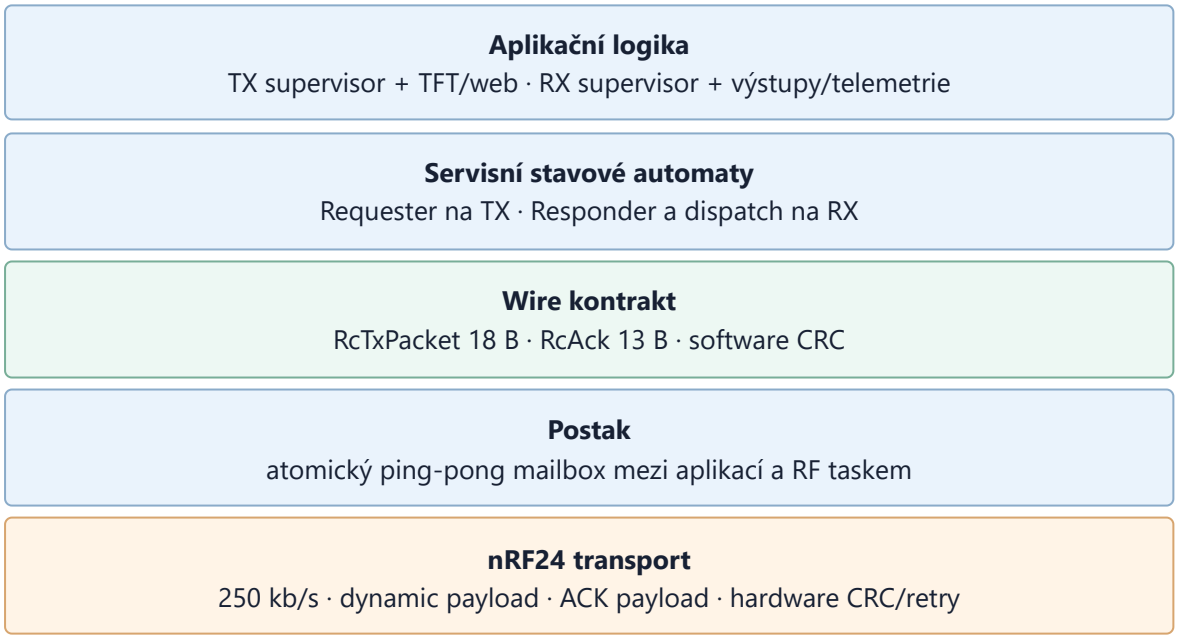
Aktuální stav hlavního RC protokolu a navázaných vrstev

Wire protocol v13 · TX Lolin S3 Mini + RX Lolin S3 Mini V2 · aktuální beta · stav k 8. 9. 2026

> RC protokol je pevný obousměrný wire kontrakt pro real-time řízení, telemetrii a potvrzenou servisní komunikaci. Každých 5 ms odešle TX jeden řídicí paket a RX ve stejném RF spojení vrátí ACK payload. Čtyřbajtový servisní podprotokol je součástí těchto paketů, ale jeho stavové automaty jsou oddělené od hlavní řídicí cesty.

Současný stav: beta TX 20260907-200420-152 + RX S3 Mini V2 20260907-200537-697, STM8 0.8. Prakticky potvrzené jsou obnova nRF, I2C hot-plug a světelná integrace, profily, reconnect, Wi-Fi konfigurace, bateriové rozhodování a OTA okraje. Otevřený zůstává thermal test, kontrola CSV a oprava OTA nabídky po pozdním příchodu RX. RF wire formát se touto integrací nemění.

1. Vrstvy systému



Každá vrstva má jednoznačný úkol. Aplikace nerozebírá rádio, RF task nerozhoduje o funkcích vozidla a Postak neinterpretuje obsah paketů. Servisní vrstva používá wire slot, ale neobchází hlavní transport.

TX cesta

```
tx_io → TX supervisor → RcTxPacket → Postak → rf_nrf24 → nRF24
```

RX cesta

```
nRF24 → rf_nrf24 → Postak → RX app/supervisor → rx_io → fyzické výstupy
                                  │
                                  └→ RcAck → Postak → ACK payload → TX
```

2. Časování a fyzický přenos

Parametr	Aktuální hodnota	Význam
RF rychlost	250 kb/s	Robustní nRF24 režim s delší dobou na jeden bit.
TX RF perioda	5000 µs / 10000 µs při ručním PA MAX	RF task vlastní nezávislý rytmus 200 Hz, případně 100 Hz; zmeškané periody nedohání burstem.
RX aplikační wake	validní RF batch / timeout 20 ms	RF task po publikaci platného batch probudí řídicí task; bez eventu pokračují safety a pomalé stavy nejpozději po 20 ms.
RX MCPWM perioda	5000 µs STEER/THROTTLE	Aktivní S3 RX vyhrazuje timer 0 pro 200Hz řízení a plyn; pomocná diff/gear/winch serva na timerech 1/2 zůstávají na 50 Hz. Běžné řízení a základní failsafe jsou prakticky ověřené end-to-end a dále sledované.
RX IRQ safety poll	20 ms	Záložní přečtení nRF24 při chybějícím IRQ; samo o sobě nevyvolává aplikační failsafe.
TX RF payload	18 B	Řízení, switches, service request a CRC16.
RX ACK payload	13 B	Telemetrie včetně RX core teploty, ACT stav, service response a CRC8.
RX link-loss detekce	1000 ms	Časové okno pro ztrátu obousměrného linku před navazující 1s rampou, 1s neutral hold a servo hard-disarmem.
TX TFT `SIGNAL LOST`	debouncovaný `LinkFs`, drop 250 ms	Prezentační hláška používá autoritativní LinkFs a v AUTO MAX/MAX režimu respektuje ještě nejvýše 250ms RF grace. Stáří jedné ACK telemetrie není trigger hlášky.
TX ACK/UI age	saturace záporného rozdílu na 0	Pokud vyšší prioritní RF task publikuje timestamp novější než dříve zachycený čas supervisoru, nevznikne unsigned podtečení ani jednorázové falešné stáří. STATS `LINK` používá autoritativní `bidirLink`; čerstvost telemetrie zůstává samostatná.
Service session	64 paketů / 1500 ms	Sdílený TX/RX limit rozpracované servisní transakce.
Service link reset	2000 ms	Kompletní reset RX responderu bez nového přijatého RF paketu; výpočet elapsed používá stejnou saturaci budoucího mezitaskového timestampu, takže nevyvolá falešný reset.
RF retry	delay 5, count 1 (fast-fail 0)	Hardwarové opakování nRF24 pod aplikačním protokolem; při fast-fail režimu TX dočasně vypne retransmise, aby neprodlužoval blokující RF pokus.

ACK payload znamená, že odpověď RX přichází jako součást potvrzení běžného TX přenosu. Není potřeba samostatné vysílací okno RX. Dynamic payload dovolí používat přesné velikosti struktur.

3. Wire paket TX → RX

0 flags 1 B	**1** seq 1 B	**2–3** steer i16	**4–5** throttle i16	**6–7** aux1 i16	**8–9** aux2 i16	**10–11** switches u16	**12–15** service 4 B	**16–17** CRC16 u16

Pole	Úloha
`flags`	Failsafe/setup/link/welcome stav a řízení PA výkonu.
`seq`	Osmibitový cyklický čítač RF paketů pro diagnostiku. Je záměrně vynechán ze software CRC16.
`steer`, `throttle`	Hlavní proporcionální řídicí kanály jako signed 16-bit hodnoty.
`aux1`, `aux2`	Další proporcionální kanály. Pouze během megatransferu v aktivním FS jsou spolu se steer/throttle dočasným osmibajtovým datovým prostorem.
`switches`	REQ bitová maska požadovaných diskrétních funkcí.
`service`	Jeden čtyřbajtový request servisního podprotokolu; nuly znamenají idle.
`crc16`	CRC16-CCITT aplikačních dat paketu; při výpočtu se `seq` nuluje.

4. Wire paket RX → TX

0 motor temp u8	1 ESC temp u8	2 RX core temp u8	3 VBAT u8	4–7 service ACK 4 B	8–9 switches ACT u16	10 flags u8	11 debug u8
12 CRC8							

Pole	Kódování a význam
Teploty motoru/ESC	Celé stupně Celsia; nula také reprezentuje nepřiřazené nebo neplatné externí čidlo.
`rx_core_temp_u8`	Interní teplota ESP v celých stupních Celsia; 255 znamená nedostupné nebo neplatné měření.
`vbat_u8`	Napětí ve voltech × 16. Rozlišení je 1/16 V.
`service_ack`	Symetrická čtyřbajtová odpověď servisního podprotokolu.
`switches_act`	ACT maska: skutečný stav funkcí na RX, ve stejném bitovém prostoru jako TX REQ.
`flags`	Stav RX linky, setupu, failsafe a PA.
`debug`	Obsahuje příznaky RX battery cutoff, thermal warning a západkového thermal tripu.
`crc8`	CRC8 všech předchozích bajtů ACK paketu.

5. Flags a diskrétní funkce

Společný byte flags

Bit	Jméno	Význam
0	FS	Failsafe stav.
1	SETUP	Setup/config režim.
2	LINK	Peer link je aktuálně platný a čerstvý.
3	WELCOME	Discovery/identifikační režim. RX přikládá CRC-validní 10B jméno nejen bez linku, ale ještě 2 s po vzniku nové link session, aby TX spolehlivě zachytil skutečnou identitu peeru i po prvním řídicím paketu.
4	Rezerva	Uvolněno přesunem navijáku do servisního kanálu.
5–6	PA level	MIN, LOW, HIGH nebo MAX.
7	PA request	Rozlišuje informativní PA status od žádosti o změnu.

REQ/ACT switches

TX odesílá požadavek `switches`, RX vrací skutečnost `switches_act`. Díky tomu TX pozná stav „požádáno, ale ještě neprovedeno" i „požadováno vypnutí, ale RX stále hlásí aktivní stav".

Bity	Definované funkce
0–2	Přední diff, zadní diff, gear high.
3–5	Světla, dálková světla, parkovací brzda.
6–7	Rezerva; nejde o proporcionální pole `aux1` / `aux2`.
8–12	Výstražná světla, maják, naviják dovnitř/ven, lightbar.
13–15	Rezerva pro budoucí real-time switch funkce.

> Setup příkazy se nemají přidávat do rezervovaných switch bitů. Patří do servisního podprotokolu, kde dostanou transaction ID, explicitní status, kontrolu parametru a ochranu proti opakovanému provedení.

6. Integrita a přijetí paketu

Transport používá dvě kontrolní úrovně:

1. **nRF24 hardware CRC a RF ACK/retry** chrání fyzický přenos.
2. **Software CRC16/CRC8** chrání přesný aplikační wire kontrakt.

TX CRC16 pokrývá celý `RcTxPacket` kromě samotného CRC pole, přičemž diagnostické `seq` se před výpočtem nuluje. RX ACK CRC8 pokrývá všech 12 bajtů před polem CRC. Service request i response jsou proto zahrnuté v aplikačním i hardwarovém CRC.

Teprve přijatý a ověřený paket je publikován aplikaci přes Postak. RF task tedy neposkytuje aplikační logice rozpracovaný nebo nevalidní frame. Po vyprázdnění celého RX FIFO pošle za batch s alespoň jedním platným paketem jednu task notification; aplikace zpracuje poslední atomický snapshot.

7. Postak: hranice mezi RF a aplikací

Postak je atomický ping-pong mailbox. Pro směr IN i OUT drží stabilní publikovanou kopii a oddělený buffer pro právě připravovaná data.

```
RF task zapisuje nový paket
            ↓
   ověření CRC a přijetí
            ↓
     atomické publish
            ↓
aplikační task čte stabilní snapshot
```

Stejně opačně aplikace připraví další výstupní paket, provede commit a RF task si převezme kompletní stabilní kopii. Tím se omezuje riziko, že jedno jádro nebo task čte strukturu ve chvíli, kdy ji jiné místo zrovna mění.

Role	Postak IN	Postak OUT
TX	`RcAck`	`RcTxPacket`
RX	`RcTxPacket`	`RcAck`

8. Čtyřbajtový servisní podprotokol

Servisní slot má v obou směrech stejný footprint:

Byte	Request TX → RX	Response RX → TX
0	horní nibble TID, dolní nibble operation	horní nibble stejné TID, dolní nibble status
1	task ID	potvrzený task ID
2	parameter ID	potvrzený parameter ID
3	hodnota nebo řízení streamu	vrácená hodnota nebo řízení streamu

Nulový frame znamená idle. Transaction ID používá hodnoty 1 až 15 a cyklicky se protáčí. TX přijme odpověď pouze při shodě TID, tasku a parametru.

Stop-and-wait pravidla

1. TX zveřejní jeden request.
2. Stejný request zůstává v každém hlavním RF paketu.
3. RX jej provede a vrátí odpověď.
4. TX čeká na přesně odpovídající terminální status.
5. Po dokončení zveřejní nulový idle request.

RX si pamatuje poslední request a odpověď. Opakovaný identický RF frame proto neprovede povel znovu, pouze vrátí cache. To zajišťuje idempotenci například při NVS zápisu.

Recovery po ztrátě TX

RX zpracuje service request jen při skutečně nové publikaci vstupu z Postaka; mezi novými pakety drží poslední ACK. Opuštěný upload nebo download autonomně expiruje po 1500 ms nebo po rozpětí 64 hlavních packet sequence čísel. Expirace smaže stream i duplicate-response cache. Po 2000 ms bez nového RF paketu proběhne kompletní reset responderu. Restartovaný TX proto může otevřít novou transakci bez restartu RX a správnost nezávisí na tom, zda stihl odeslat wire `CANCEL`.

Limity transakce

- současně běží nejvýše jedna transakce;
- sdílený TX/RX limit je 1500 ms nebo rozpětí 64 hlavních RF packet sequence čísel;
- po 2000 ms bez nového RX paketu se resetuje celý responder i replay cache;
- `BUSY` dovoluje RX handleru pokračovat v asynchronní práci;
- změna obsahu aktivního TID je odmítnuta;
- neúspěšné terminální výsledky se propisují do diagnostiky/blackboxu.

9. Operace servisního podprotokolu

Operace	Účel
`GET`	Načtení skutečné aktuální hodnoty z RX.
`WRITE`	Změna runtime/RAM hodnoty.
`WRITE_SAVE`	Změna s persistentním uložením, pokud ji konkrétní task povoluje.
`EXECUTE`	Jednorázová akce, například ping nebo explicitní save.
`STREAM_BEGIN`	Zahájení logického vícebajtového přenosu.
`STREAM_DATA_0..3`	Čtyři možné pozice fragmentu.
`CANCEL`	Zrušení rozpracovaného streamu.

Response status rozlišuje mimo jiné `OK`, `ERROR`, `BUSY`, `UNKNOWN_TASK`, `BAD_OPERATION`, `BAD_VALUE`, `FORBIDDEN`, `NOT_SUPPORTED` a `TRANSACTION_MISMATCH`. `TIMEOUT` a `RETRY_LIMIT` vznikají lokálně na TX.

10. Vícebajtový přenos v pevném 4B slotu

Jedna fyzická výměna má jen jeden datový byte, ale vrstva umí logickou hodnotu dlouhou až čtyři bajty. Jednobajtové hodnoty používají rychlou kompaktní cestu. Širší hodnota se rozdělí na potvrzované fragmenty.

WRITE dvoubajtové hodnoty

```
TX: STREAM_BEGIN (semantická operace WRITE, délka 2)
RX: STREAM_BEGIN potvrzen

TX: STREAM_DATA_0 + první byte
RX: fragment 0 potvrzen

TX: STREAM_DATA_1 + druhý byte
RX: OK; teprve nyní RX validuje a aplikuje celou hodnotu
```

GET dvoubajtové hodnoty

```
TX: GET
RX: STREAM_BEGIN, délka odpovědi 2

TX: žádost o fragment 0
RX: první byte

TX: žádost o fragment 1
RX: druhý byte + OK

TX: sestavení výsledné little-endian hodnoty
```

Celý stream drží stejné TID, task a parameter. Existuje vždy jen jeden nepotvrzený fragment. Při opožděném nebo špatně seřazeném fragmentu se vrátí `BUSY` a očekávaný fragment se zopakuje.

> Teoretický prostor čtyřbajtového slotu při 200 Hz je 800 B/s v každém směru. Skutečný aplikační datový tok je nižší kvůli hlavičkám, ACK krokům a stop-and-wait. To je záměrné: prioritou je jednoduchost a spolehlivost pomalých příkazů, ne maximální throughput.

11. Aktuální servisní tasky

Task	ID	Aktuální použití
System	`0x00`	Ping, výběr TX AP nebo společné client Wi-Fi pro další RX setup.
Smart drive	`0x10`	Automatické blinkry a jejich střed od steering trimu.
Light control	`0x11`	Pomocné světelné povely a GET/WRITE runtime LIGHTBAR PWM.
Servo setup	`0x12`	GET/WRITE pozic předního/zadního diffu a převodovky, explicitní NVS save.
Winch control	`0x13`	GET/WRITE podepsané runtime rychlosti navijáku −100 až +100 % a potvrzovaný arm/disarm jeho servo signálu.
Battery control	`0x14`	GET/WRITE_SAVE stavu RX battery guardu a potvrzený sleep během kritického cutoff okna.
Brake ABS	`0x15`	GET/WRITE základních parametrů pulzní brzdy v RX RAM a explicitní NVS save.
Log control	`0x16`	GET/WRITE_SAVE persistentního RX loggeru a potvrzené smazání RX incident logů.
Megatransfer	`0x20`	Potvrzovaný velký přenos; nyní client Wi-Fi profil pro prázdné RX.

Megatransfer a Wi-Fi provisioning

`MEGATRANSFER` (`0x20`) je vyhrazená výjimka z obecného čtyřbajtového service payloadu. Čtyřbajtový service rámec zůstává řídicí částí stop-and-wait přenosu; během `MEGA_DATA` nesou pole `steer`, `throttle`, `aux1` a `aux2` osm datových bajtů. Velikost hlavního paketu zůstává 18 B.

Operace	Kód	Hodnota požadavku	Podmínka úspěšného ACK
`MEGA_BEGIN`	`0xB`	Celkový počet bajtů payloadu	`READY (2)` až po fyzickém servo LOW
`MEGA_DATA`	`0xC`	Index 8B chunku od nuly	ACK vrátí přesně přijatý index
`MEGA_END`	`0xD`	Celkový počet chunků	`COMMITTED (6)` až po aplikaci, získání IP a NVS save
`CANCEL`	`0xA`	`0`	Uvolnění session; rollback pouze před commitem

Celý přenos drží stejné transaction ID, task `0x20` a parameter. ACK rámec neobsahuje operaci, proto TX každou nově publikovanou ACK zpracuje pouze jednou a navíc kontroluje hodnotu očekávanou v aktuální fázi: `READY`, index chunku nebo `COMMITTED`. Staré `OK` ponechané v Postaku z předchozí fáze proto nemůže vyvolat falešný stav `SYNCED`.

Přenos je povolen pouze z TFT Wi-Fi setupu a RX jej přijme jen s aktivním `RC_FLAG_FS`. RX nejprve přes stávající servo-disarm politiku dokončí rampu, neutral hold a hard-disarm signálů LOW. Teprve potom potvrdí připravenost. Prvním objektem je client Wi-Fi profil o maximálně 103 B: verze, délky, SSID, heslo a CRC32, rozdělené do nejvýše 13 osmibajtových chunků.

RX přijatý profil nejprve zařadí pouze do RAM a počká, až se deferred konfigurace skutečně stane živým client profilem. Potom bez spuštění HTTP serveru ověří připojení a přidělení IP a teprve po úspěchu uloží NVS. Apply, Wi-Fi probe a commit posouvá asynchronní poll RX aplikačního tasku, takže dokončení nezávisí na přijetí dalšího RF požadavku. Krátký RF výpadek způsobený Wi-Fi asociací drží již disarmovanou session; chyba, cancel před commitem nebo vypršení 45s lease vrátí předchozí RAM konfiguraci. Potvrzeně uložený profil se již nevrací.

Datové požadavky odcházejí podle 5ms TX send tiku. RX asynchronní stav apply/connect/save kontroluje při každém validním RF batch eventu a bez RF nejpozději po 20ms safety timeoutu; timeout není rychlostí přenosu chunků. Receive-progress timeout je 3 s, celková lease 45 s a Wi-Fi probe timeout 20 s. Payload má vlastní CRC32, navíc jej chrání stávající packet CRC a hardwarové CRC nRF24. Záměrně není šifrovaný.

Servo setup

Šest pozic OFF/ON používá rozsah −150 až +150 % kolem středu 1500 µs:

```
pulse_us = 1500 + percent × 5
```

Rozsah −120 až +120 se vejde do kompaktního jednoho bajtu. Hodnoty mimo něj používají dvoubajtový little-endian `int16`. `WRITE` mění pouze RAM a fyzický výstup. Long-click na TFT odešle potvrzený `EXECUTE/SAVE_CONFIG`; `SAVED` se zobrazí až po úspěšném NVS zápisu na RX.

Winch control

TFT mění rychlost navijáku po 5 % a odesílá absolutní podepsanou hodnotu. Záporná hodnota znamená OUT, kladná IN a nula STOP. RX převádí procenta přímo na nakonfigurovaný `WINCH_PWM` servo pulz pouze v době, kdy je otevřené TFT menu navijáku. Otevření odešle potvrzovaný arm při nulové rychlosti. Zavření nejprve dokončí potvrzený STOP a potom potvrzený disarm, který drží pouze tento signál trvale LOW. Failsafe, setup, restart RX nebo aplikace konfigurace rychlost vynuluje a výstup disarmuje. Proporcionální AUX pole hlavního paketu zůstávají volná.

Battery control

`BATTERY_CONTROL` (`0x14`) odděluje autonomní RX ochranu od potvrzeného rozhodnutí uživatele. `GET/GUARD_ENABLED` čte uložený stav guardu, `WRITE_SAVE/GUARD_ENABLED` jej mění a ukládá a `EXECUTE/SLEEP_NOW` potvrzuje deep sleep pouze během aktivního cutoff rozhodovacího okna.

Po potvrzeném kritickém podpětí RX drží failsafe a servo-disarm sekvenci nejvýše 15 s. TX může potvrdit okamžitý sleep nebo po vědomém dlouhém držení guard trvale vypnout. Bez rozhodnutí, po ztrátě service linku nebo při nouzovém napěťovém flooru RX usne autonomně. Service odpověď nikdy nenahrazuje lokální RX ochranné rozhodnutí.

Brake ABS

`BRAKE_ABS` (`0x15`) zpřístupňuje přes TFT `Adjust/ABS` režim `OFF/LINEAR/FIXED`, frekvenci 5–50 Hz, lineární duty min/max a pevné duty 0–100 %. Otevření karty automaticky sekvenčně provede `GET` všech pěti skutečných RX hodnot do existující TX RAM cache; klik na načtený řádek proto může rovnou otevřít editaci bez druhého GET. Při první chybě se dávka zastaví, aby nedostupný RX nevyvolal řetězec timeoutů. Potvrzení používá RAM-only `WRITE`; dlouhý stisk odešle jediný `EXECUTE/SAVE_CONFIG` a dirty stav se smaže až po úspěšném NVS ACK. Změna parametru resetuje ABS stavový automat do bezpečného čekání na uživatelský neutrál. Čtyřbajtový service slot a velikosti paketů 18/13 B se nemění.

Log control

`LOG_CONTROL` (`0x16`) obsluhuje TFT kartu `Setup/Logs`. `GET/ENABLED` načte skutečný uložený stav RX loggeru, `WRITE_SAVE/ENABLED` jej persistentně zapne nebo vypne a `EXECUTE/ERASE` smaže uložené RX incident logy. Stav TX loggeru je lokální; společné `Erase all` nejprve čeká na `OK` z RX a až potom smaže TX failsafe a crash logy. `BUSY`, timeout nebo nedostupný RX proto nezpůsobí jednostranné smazání TX. Celý read/toggle/ erase průchod byl prakticky potvrzen na aktuální stable v3 dvojici. Wire rámce ani velikosti paketů se nezměnily.

12. Oddělení real-time a servisní cesty

Real-time část	Servisní část
steer, throttle, AUX	setup a konfigurace
switch REQ/ACT	GET skutečného nastavení
failsafe a link flags	potvrzené RAM změny
základní telemetrie	explicitní NVS save
každý RF cyklus	jedna stop-and-wait transakce

Čekání, retry nebo více fragmentů běžné servisní transakce nepozastaví hlavní ovládání. Vyhrazený megatransfer je vědomá výjimka: běží pouze v aktivním FS po servo hard-disarmu, nuluje switches a osová pole dočasně používá jako data.

RX profile backup/restore není další RF service objekt. Velký verzovaný JSON jde v setup režimu cestou prohlížeč → TX WebSetup → RX HTTP API. Schema-v2 přidalo parametrický battery profil a schema-v3 top-level konfiguraci pulzní brzdy; Wi-Fi hesla, API klíč, live telemetrie, logy a OTA stav se záměrně neexportují.

13. Typická úplná servisní cesta

14. Kompatibilita a pravidla dalšího růstu

- `RcTxPacket` musí zůstat přesně 18 B a `RcAck` 13 B, dokud není záměrně změněn wire protokol na obou stranách.
- Pořadí polí, packing, endianita, flags a CRC jsou kompatibilitní hranice.
- `RC_PROTO_VER = 13` je diagnostická verze; není serializovaná v každém RF paketu a sama neprovádí runtime vyjednávání kompatibility.
- Změna velikosti ACK nebo významu polí proto vyžaduje společný build a nasazení TX i RX.
- Nové rychlé diskrétní funkce mohou využít rezervované switch bity.
- Nové setup parametry mají dostat task/parameter v servisním protokolu.
- Obecná logická service hodnota má limit čtyři bajty. Vyhrazený megatransfer používá service jako řízení a v aktivním FS dočasně přenáší osm datových bajtů v osových polích.
- Persistenci je vhodné oddělit od runtime změny a provádět ji pouze po explicitním potvrzení uživatele.

TX a RX se nesmějí aktualizovat nesourodými wire verzemi. Současný systém nemá handshake, který by bezpečně přeložil starší ACK a 13B aktuální ACK. Pro změnu wire formátu se proto nasazují oba firmware společně.

15. Lokální I2C linka a světelný modul

RF kontrakt zůstává 18 B TX / 13 B základní ACK / 4 B service. I2C je samostatná lokální HW hranice na RX. Výsledný logický stav světel vlastní RX supervisor, adresovou dostupnost a Wire recovery vlastní `i2c_devices`, protokol modulu přenáší `rx_light_i2c`. Obě strany používají sdílený `i2c_light_module/lib/light_i2c_protocol`.

```
RX supervisor → konfigurace + latest-wins světelný stav
  → rx_light_i2c → LightI2cMaster → Wire / I2C 100 kHz → STM8
i2c_devices → dostupnost adres + generace obnovy sběrnice
```

Hranice	Současný kontrakt
Elektrická vrstva RX V2	GPIO35 SDA, GPIO36 SCL; STM8 PB5 SDA, PB4 SCL; společná zem, 3,3V pull-up, adresa 0x30.
Discovery	STM8 kontrola 250 ms, odpojení po dvou NACK; jedna MPU6050 na 0x68 nebo 0x69, kontrola dostupné/nedostupné 500/1500 ms, tři NACK.
Recovery sběrnice	Opakovaný bus error/timeout → odložené Wire.end(), nejvýše devět SCL pulzů a STOP, Wire.begin(). Samotný address NACK ani CRC/short-read klienta zdravou sběrnici neresetuje.
INFO protokol v3	CRC, magic, verze, schema, 1–16 kanálů, velikost konfigurace a dvě 16bitové PWM masky. STM8 0.8: 11 kanálů, config 82 B, HW 0x0607, PWM 0x07FF.
Konfigurace	Velikost 3 + N × 6 + 13 B; bloky dat nejvýše 16 B, atomický CONFIG_COMMIT a potvrzení CONFIGURED se správnou generací.
Runtime	I2C task s 20ms průchody; STATE po 150 ms a STATUS přibližně po 1 s. Výběr čteného registru a čtení jsou oddělené STOPem.
Změna / návrat modulu	Nulový STATE před reconfig; živá data až po potvrzení generace. Reconnect, reset modulu či generace bus recovery opakují handshake. Modul autonomně hlídá timeout a failsafe duty.
Profil / UI	16 External pozic v RX NVS v11, JSON schema v6 (import v1–v6); skutečné kanály a funkce řídí INFO. FRONT/REAR jen HW PWM, LIGHTBAR PWM, digitální kanály jen 0/255 bez fade.

Lokální bindingy se při připojení modulu nepřepisují. Shodná lokální/externí funkce sdílí autoritativní parametry. Žádný I2C krok neblokuje RX control task; adresa s ACK sama o sobě nepotvrzuje kompatibilitu ani aktivní konfiguraci modulu.

Obnova místního nRF

Po 200 ms bez platného řídicího paketu (RX) nebo ACK payloadu (TX) RF task ověří STATUS a aktuální CONFIG, kanál, rychlost, PA, retry, ACK/dynamic payload registry a adresy. Zdravý čip pokračuje bez reinitu. Při chybě proběhne begin(&SPI), obnova konfigurace a readback bez restartu SPI, objektu nebo tasku. Další kontrola/pokus je nejdříve 1 s od dokončení předchozího. Pošťák, čas posledních platných dat a safety se neresetují. Diagnostika: [RF] [health] a čítače kontrol/pokusů/úspěchů.

Současný výsledek

Aktuální RC protokol kombinuje tři vlastnosti:

1. **Deterministické real-time řízení** ve 200Hz pevně strukturované výměně.
2. **Skutečný stav RX** přes ACK telemetrii a REQ/ACT masky.
3. **Spolehlivé pomalé služby** přes symetrický 4B stop-and-wait podprotokol s fragmentací, deduplikací a explicitní persistencí.

Výsledkem není druhá paralelní radiová linka. Je to jedna RC výměna, nad kterou běží dvě logické cesty: nepřetržité řízení a potvrzovaná servisní komunikace.

Stav dokumentu: 8. 9. 2026 · současná beta a I2C protokol v3
Zdrojový wire kontrakt: `lib/common/include/rc_protocol.h`
Servisní protokol: `lib/rc_service/README.md` a `lib/rc_service/`
Mailbox: `lib/postak/` · RF transport: `lib/rf/`