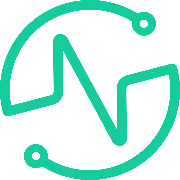
****

**Vyšší odborná škola**

**a Střední průmyslová škola elektrotechnická**

**Plzeň, Koterovská 85**

**praktická práce s obhajobou**

**Téma: CZECHMATE**

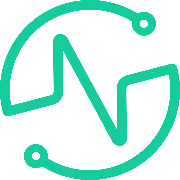
**Autoři práce: Krutina Alfréd, Jager Matěj**

**Třída: 2.D**

**Vedoucí práce: Miroslav SOUKUP**

**Dne: . . 2026**

**Hodnocení:**

**Vyšší odborná škola**

**a Střední průmyslová škola elektrotechnická**

**Plzeň, Koterovská 85**

**Zadání ročníkové práce**

**Žáci:** Krutina Alfréd, Jäger Matěj

**Třída:** 2.D

**Studijní obor:** 26-41-M/01 Elektrotechnika

**Zaměření:**IoT

**Školní rok:** 2025–2026

**Téma práce:** chytrá šachovnice

**Pokyny k obsahu a rozsahu práce:**

Návrh a realizace inteligentní šachovnice (e-board), která detekuje pohyby figurek, vizualizuje tahy pomocí LED

**Určení částí tématu zpracovávaných jednotlivými žáky:**

1. HW – Elektrotechnický návrh zařízení: Matěj Jäger

* Detekce figurek je založena na principu maticového skenování řádků a sloupců pro rychlé určení polohy, zatímco vizualizace tahů probíhá pomocí adresovatelných RGB LED diod zapojených v sériovém řetězci, který umožňuje ovládání všech polí jediným datovým signálem.

1. FW – Vývoj programu pro zařízení: Alfréd Krutina

* Firmware implementuje šachovou logiku, zpracovává a validuje tahy, koordinuje komunikaci

mezi systémovými komponentami a spravuje stav hry v reálném čase.

1. SW – Vývoj aplikace/webové stránky pro zařízení

* WIFI Access Point (AP) s HTTP serverem pro přímé připojení a WiFi Station (STA) pro

připojení k domácí síti, webové rozhraní s REST API poskytuje vzdálené ovládání a sledování

hry

1. Návrh mechanických částí zařízení

* Celá konstrukce byla navržena v softwaru Autodesk Fusion 360 jako spojení vnějšího designového šasi s integrovanými funkčními prvky a vnitřního modulárního systému, který zajišťuje přesné uložení a ochranu všech vnitřních součástek.

***Požadavek na počet vyhotovení ročníkové práce:*** *1 výtisk*

Termín odevzdání: **09. června 2026**

Čas obhajoby: **5-8 minut**

Vedoucí práce: **Miroslav Soukup**

Projednáno a schváleno učitelem předmětu **IoT**.

V Plzni dne: 01. listopadu 2025

**Anotace:** Cílem této práce bylo navrhnout chytrou šachovnici, která by pomocí světelné signalizace reagovala na hru. Jelikož zadání neobsahovalo konkrétní technologii snímání, součástí práce byl výzkum a vývoj vlastního řešení. Výsledný prototyp využívá magnetické senzory a mikrokontrolér ESP32-C6 k detekci pohybu (systém "Pick & Drop") a nahrazuje původně zamýšlené standardní diody moderním adresovatelnými LED páskem pro komplexní vizualizaci hry.

**Poděkování:** Rádi bychom poděkovali našim učitelům, a to zejména panu Miroslavu Soukupovi, za zasvěcení do oboru a motivaci, která v nás probudila touhu objevovat nové možnosti a věnovat se projektu i nad rámec školních povinností. Děkujeme také vývojářům platformy ESP-IDF **[3]** za přípravu frameworku, na kterém je náš kód postaven.

Velký dík patří tvůrcům online vzdělávacího obsahu, z jejichž práce jsme čerpali klíčovou inspiraci. Obrovským přínosem pro logiku hry nám bylo video „Coding Adventure: Chess AI“ od Sebastiana Laguea **[6]** . Zásadním zdrojem pro nás byl Shawn Hymel **[5]** , jehož detailní tutoriály o systému FreeRTOS, WIFI driverech a mnoho dalších tématech, nám poskytly nezbytný teoretický základ pro návrh firmwaru našeho zařízení.

Závěrem děkujeme našim rodinám za materiální podporu a toleranci během dlouhých hodin strávených vývojem a testováním.

**Klíčová slova**: IoT, Embedded, ESP32-C6, Bluetooth LE, maticové snímání, WS2812B, Fusion 360, Li-ion, chytrá šachovnice.

**Čestné prohlášení:**

„Prohlašujeme, že jsme tuto práci vypracovali samostatně a použili literárních pramenů a informací, které citujeme a uvádíme v seznamu použité literatury a zdrojů informací.“

„Souhlasíme s využitím naší práce učiteli VOŠ a SPŠE Plzeň k výuce.“

V Plzni dne: 09.06. 2026

Podpisy:

……………………………………… …………………………………………

Alfréd Krutina Matěj Jäger

Obsah

[1. Úvod 4](#_Toc219572383)

[2. HW – Elektrotechnický návrh zařízení 5](#_Toc219572384)

[2.1 Výběr vhodných součástek 5](#_Toc219572385)

[2.1.1 Magnetický jazýčkový spínač (Reed switch) 5](#_Toc219572386)

[2.1.2 Schottkyho diody 5](#_Toc219572387)

[2.1.3 ESP32-C6 6](#_Toc219572388)

[2.1.4 UPS Li-ion Boost modul s nabíječkou, 5V 6](#_Toc219572389)

[2.1.5 GeB Li-Ion Baterie 3x18650 1S3P 3.7V 9600mAh 7](#_Toc219572390)

[2.1.6 inteligentní RGB LED NeoPixel, WS2812B, 10x10, 5050, 5V 7](#_Toc219572391)

[2.1.7 Tlačítko 6x6x5mm 8](#_Toc219572392)

[2.1.8 Spínač kovový s aretací – Power, stříbrný 16mm 3-6V 8](#_Toc219572393)

[2.1.10 Nabíječka Li-ion článku TP4056 s ochranou USB-C 9](#_Toc219572394)

[2.2 Blokové schéma 9](#_Toc219572395)

[2.3 Schéma zapojení 11](#_Toc219572396)

[2.4 Postup a realizace elektrického rozvodu 15](#_Toc219572397)

[3 FW – Vývoj programu pro zařízení 16](#_Toc219572398)

[3.1 Textový popis programu 16](#_Toc219572399)

[3.1.1 Volba platformy a architektura systému 16](#_Toc219572400)

[3.1.2 Časování pomocí scheduleru 17](#_Toc219572401)

[3.1.3 Detailní popis hlavních úloh (Tasků) 17](#_Toc219572402)

[3.1.4 Technická řešení problémů (Paměť a bezpečnost) 19](#_Toc219572403)

[3.1.5 Průběh vývoje a překážky 20](#_Toc219572404)

[3.2 Diagram toku dat a komunikace mezi úlohami 21](#_Toc219572405)

[3.3 Postup vývoje programu 22](#_Toc219572406)

[3.3.1 Historie a evoluce projektu 22](#_Toc219572407)

[3.3.2 Vývojové nástroje a zálohování 22](#_Toc219572408)

[3.3.3 Známá omezení programu 22](#_Toc219572409)

[3.3.4 Plány do budoucna a rozšíření 23](#_Toc219572410)

[4 Návrh mechanických částí zařízení 24](#_Toc219572411)

[4.1 Výběr / návrh mechanických částí zařízení 24](#_Toc219572412)

[4.1.1 Rendery modelu: 24](#_Toc219572413)

[5 Odzkoušení hotové ročníkové práce v praxi 26](#_Toc219572414)

[6 Závěr 30](#_Toc219572415)

[7 Seznam obrázků 31](#_Toc219572416)

[8 Reference 32](#_Toc219572417)

# Úvod

Tato práce řeší návrh a implementaci elektronické šachovnice na platformě ESP32-C6. Hlavním podnětem pro vznik tohoto projektu byla naše osobní záliba v šachové hře a potřeba pořídit si vlastní šachovnici, kterou jsem se rozhodl rovnou vylepšit o moderní „chytré“ funkce. Společně s kolegou jsme se rozhodli vytvořit zařízení, které bude konstrukčně jednoduché, uživatelsky přívětivé a zároveň cenově dostupné.

Cílem je vytvořit funkční systém schopný automaticky detekovat tahy hráčů na fyzické šachovnici, validovat je podle pravidel FIDE **[7]** , poskytovat vizuální zpětnou vazbu prostřednictvím LED osvětlení a umožňovat vzdálené sledování hry přes webové rozhraní.

Práce se zaměřuje na implementaci šachové logiky včetně všech standardních i speciálních tahů, detekci konce hry, návrh senzorického systému pro detekci figurek, bezdrátovou komunikaci přes Wi-Fi a integraci časovače. Technické řešení je postaveno na operačním systému FreeRTOS pro paralelní zpracování úloh a režimu Wi-Fi APSTA pro flexibilní síťové připojení.

# HW – Elektrotechnický návrh zařízení

V rámci hardwarové realizace byl kladen důraz na funkční a kompaktní propojení všech modulů do jednoho celku bez použití vlastního plošného spoje. Jako řídicí platformu jsme zvolili mikrokontroler ESP32**-**C6, ke kterému jsou přímo napájeny jednotlivé komponenty a periférie.

Všechny klíčové součástky, včetně 64 magnetických jazýčkových spínačů a příslušných Schottkyho diod, byly ručně propojeny tak, aby tvořily funkční matici. Stejným způsobem byly vnitřním prostorem rozvedeny datové a napájecí vodiče k jednotlivým RGB LED čipům, které jsou pájeny do sériového řetězce. Tento přístup vyžadoval precizní manuální práci při pájení, aby bylo zajištěno spolehlivé vedení signálu a zároveň byly všechny součástky v šasi umístěny na svých přesně definovaných pozicích.

## Výběr vhodných součástek

V této kapitole jsou popsány klíčové komponenty zvolené pro realizaci chytré šachovnice. Výběr byl podřízen požadavkům na spolehlivou detekci figurek, vizuální signalizaci a dlouhou výdrž při bezdrátovém provozu.

### Magnetický jazýčkový spínač (Reed switch)

Tyto spínače slouží k detekci přítomnosti figurky na konkrétním poli šachovnice. Jazýčkový kontakt je umístěn pod každým polem a sepne v okamžiku, kdy se k němu přiblíží magnet zabudovaný v podstavci šachové figurky. Celkem je v projektu využito 64 těchto spínačů.



Obrázek - Magnetický jazýčkový spínač (Reed switch)

### Schottkyho diody

V zapojení jsou tyto diody využity výhradně v rámci snímací matice 8x8. Jsou zapojeny v sérii s každým jazýčkovým spínačem, aby se zabránilo vzniku falešných stisků (tzv. ghostingu), ke kterému by docházelo, pokud by bylo sepnuto více polí současně. Schottkyho diody byly zvoleny pro svůj velmi nízký úbytek napětí v propustném směru.

Obsah obrázku elektronka

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - Schottkyho diody

### ESP32-C6

Řídicím centrem je mikrokontroler ESP32-C6 **[1]** . Tento moderní čip podporuje Wi-Fi 6 a Bluetooth 5 LE, což umožňuje komunikaci šachovnice s externími aplikacemi. Zajišťuje skenování matice spínačů, vyhodnocování šachových pravidel a adresování RGB LED diod.

Obsah obrázku elektronika, Elektronické inženýrství, Elektronická součástka, Obvodoví součástka

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - ESP32-C6

### UPS Li-ion Boost modul s nabíječkou, 5V

Jelikož inteligentní RGB LED diody vyžadují pro plný jas napětí 5 V, zatímco baterie poskytují nižší napětí, je v obvodu zařazen zvyšující (boost) měnič. Ten udržuje stabilní napájecí hladinu i při postupném vybíjení akumulátoru.

Obsah obrázku elektronika, Elektronická součástka, Obvodoví součástka, Pasivní součástka

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - UPS Li-ion Boost modul s nabíječkou, 5V

### GeB Li-Ion Baterie 3x18650 1S3P 3.7V 9600mAh

Napájení zajišťuje trojice akumulátorů zapojených paralelně. Toto zapojení zvyšuje celkovou kapacitu zdroje na **9600 mAh**, což zaručuje poměrně dlouhou výdrž zařízení i při aktivním podsvícení všech polí. Napětí sestavy zůstává na jmenovitých 3,7 V.

Obsah obrázku umělá hmota

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - GeB Li-Ion Baterie 3x18650 1S3P 3.7V 9600mAh

### 2.1.6 inteligentní RGB LED NeoPixel, WS2812B, 10x10, 5050, 5V

Pro vizuální signalizaci jsou pod každým polem použity **inteligentní RGB LED NeoPixel** typu **WS2812B [2]**  v pouzdře **5050** o rozměrech **10x10 mm**, které jsou napájeny napětím **5V** a díky integrovanému řídicímu kontroléru umožňují nezávislé ovládání barvy i jasu každého pole pomocí jediného datového pinu z ESP32.



Obrázek - nteligentní RGB LED NeoPixel, WS2812B, 10x10, 5050, 5V

### Tlačítko 6x6x5mm

V šasi je umístěno celkem 9 ovládacích mikrospínačů (tact switches), které jsou ručně propojeny s řídicí jednotkou. Jedno tlačítko slouží jako systémový reset pro restartování hry, zatímco zbývajících 8 tlačítek je rozděleno po stranách šachovnice (pro každého hráče), kde slouží ke specifické funkci proměny pěšce v jinou figuru při dosažení poslední řady.

Obsah obrázku sníh, páčka

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - Tlačítko 6x6x5mm

### Spínač kovový s aretací – Power, stříbrný 16mm 3-6V

Mechanický spínač slouží k hardwarovému odpojení baterie od hlavního obvodu, čímž se minimalizuje samovybíjení a zajišťuje se bezpečné vypnutí zařízení.



Obrázek - Spínač kovový s aretací – Power, stříbrný 16mm 3-6V

### 2.1.9 Nabíječka Li-ion článku TP4056 s ochranou USB-C

Pro pohodlné dobíjení integrovaných akumulátorů je zařízení vybaveno nabíjecím modulem s konektorem USB-C. Tento modul zajišťuje bezpečný proces nabíjení (CC/CV algoritmus) a chrání Li-ion články před přebitím. Díky modernímu USB-C standardu lze šachovnici nabíjet běžnými adaptéry od mobilních telefonů nebo z USB portu počítače

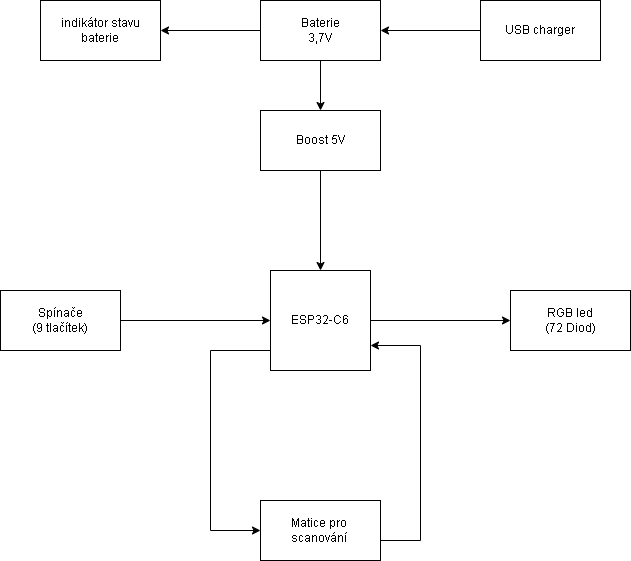
**Obsah obrázku Elektronická součástka, Obvodoví součástka, Pasivní součástka, Elektronické inženýrství

Obsah generovaný pomocí AI může být nesprávný.**

Obrázek - Nabíječka Li-ion článku TP4056 s ochranou USB-C

## Blokové schéma

Blokové schéma znázorňuje tok energie a signálů v celém systému. Napájecí část začíná **USB-C konektorem**, který přes nabíjecí modul dobíjí tři paralelně zapojené **Li-ion články**. Napětí z baterií prochází přes hlavní vypínač do **Boost měniče**, který jej stabilizuje na **5 V** pro napájení celého zařízení. Centrální jednotka **ESP32-C6** přijímá vstupy z **matice 64 jazýčkových spínačů** (ošetřených diodami) a **9 ovládacích tlačítek**. Na základě těchto dat pak mikrokontroler po jedné datové lince řídí sériově zapojený řetězec **64 RGB LED NeoPixel** a odesílá informace o hře přes **Wi-Fi/Bluetooth**.

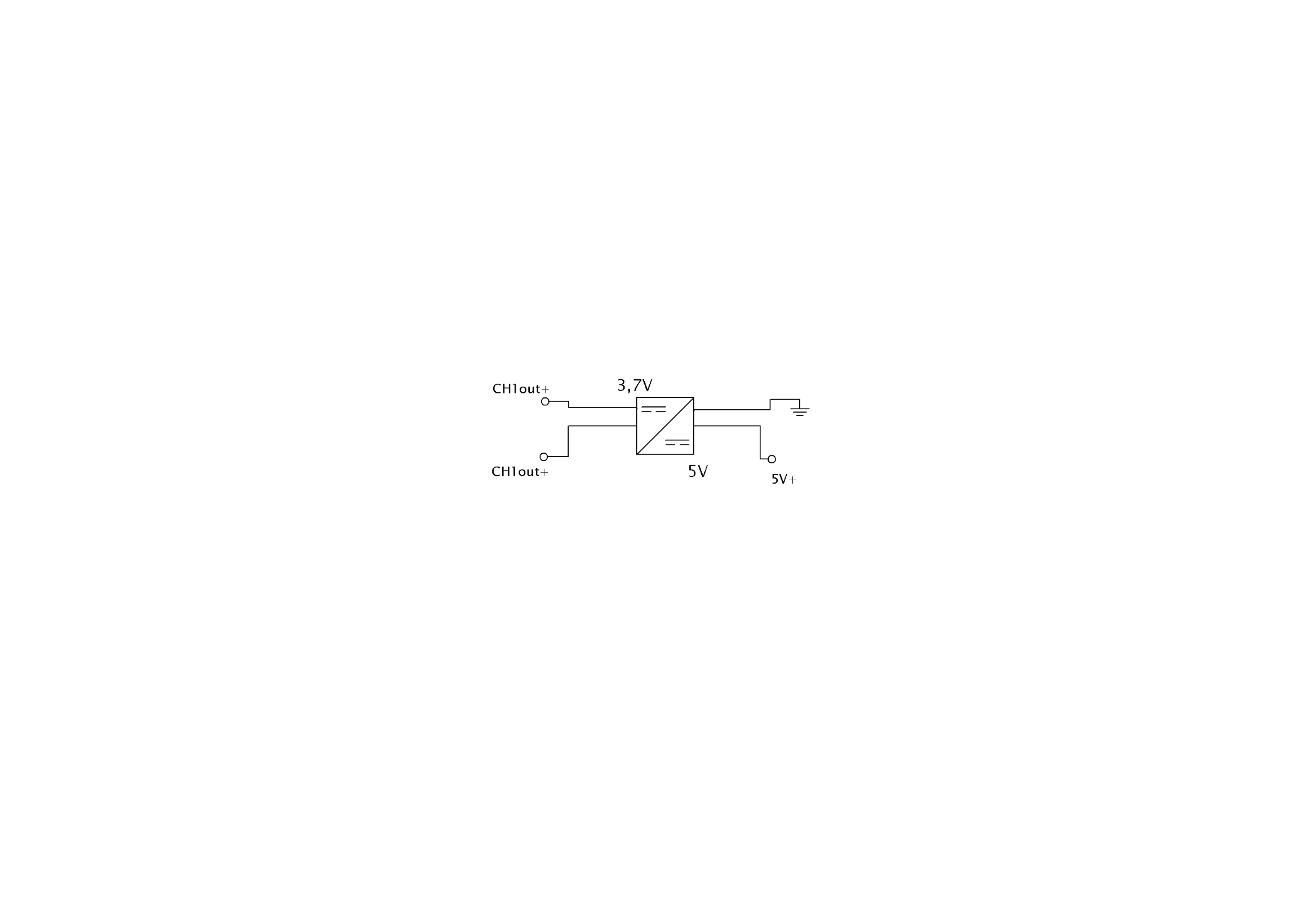


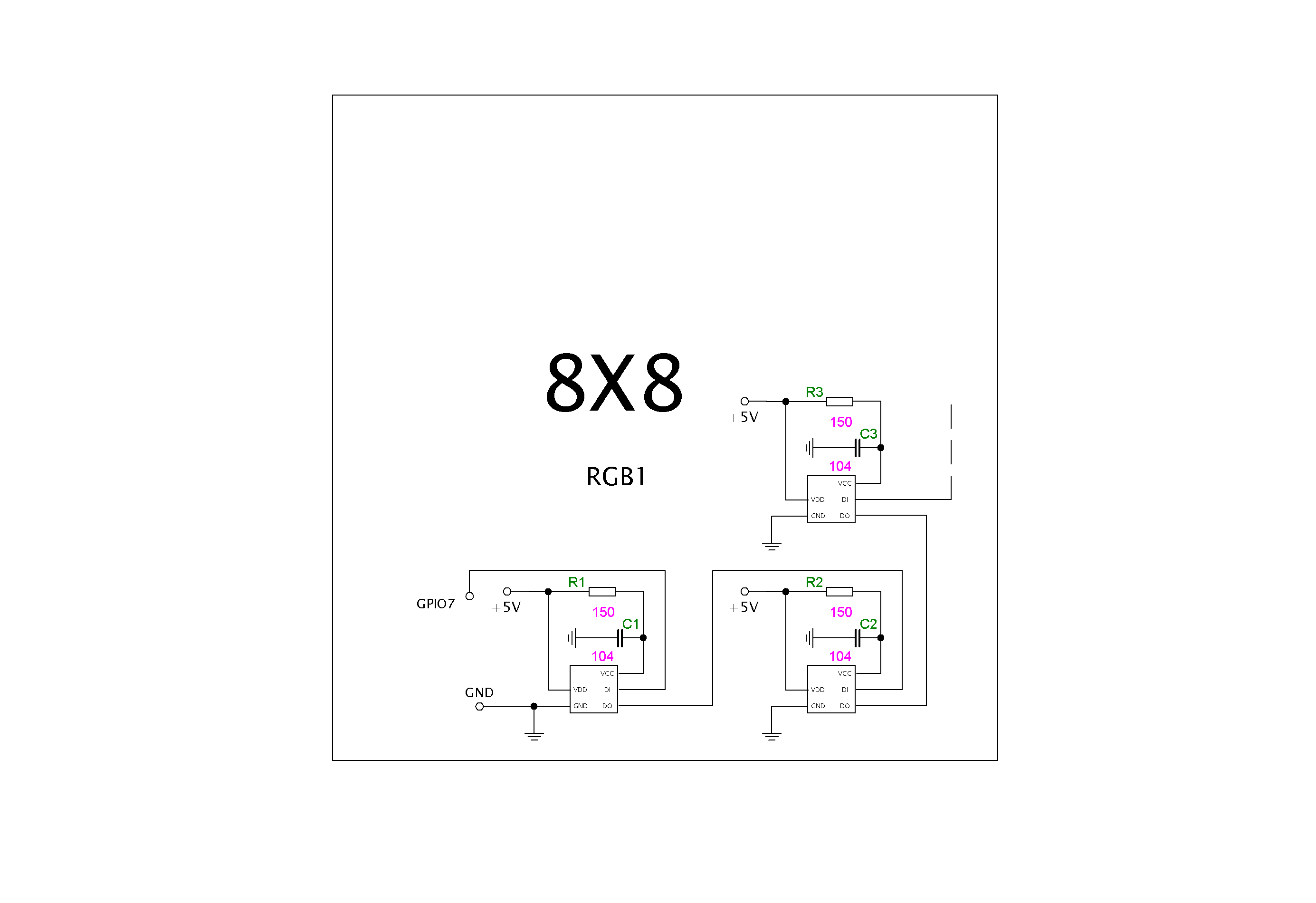
Obrázek - blokové schéma

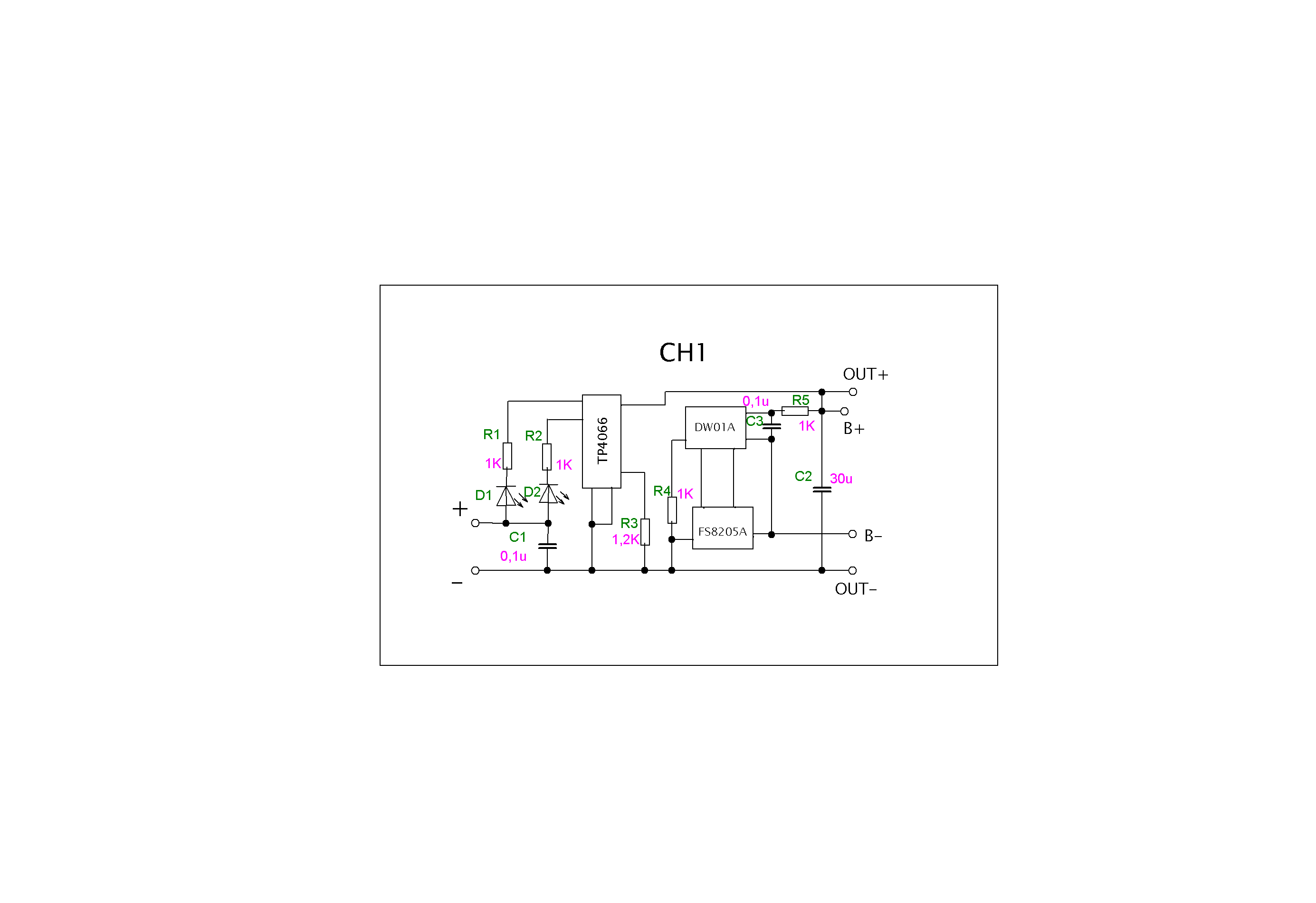
## Schéma zapojení

Schéma lze rozdělit do několika logických celků, které na sebe navazují:

* **Napájecí větev:** Začíná vstupním konektorem USB-C, který je připojen k nabíjecímu modulu. Ten je spojen se sestavou tří Li-ion článků. Za bateriemi následuje hlavní vypínač, který pouští napětí do Boost měniče. Ten převádí kolísavé napětí baterie na stabilních 5 V pro napájení LED a mikrokontroleru.
* **Řídicí část (ESP32-C6):** Tvoří střed schématu. Jsou zde zakresleny vývody (piny), které obsluhují sběr dat z matice, ovládání tlačítek a odesílání signálu pro LED diody. (viz:)
* **Snímací matice 8x8:** Tato část obsahuje 8 řádkových a 8 sloupcových vodičů. V každém křížení je zakreslen jazýčkový spínač v sérii se Schottkyho diodou. Diody směřují tak, aby umožnily průchod proudu pouze jedním směrem a zabránily falešným signálům. (viz:)
* **Zobrazovací větev (NeoPixel):** Je znázorněna jako řetězec 64 symbolů LED diod. Každá má tři piny: 5V, GND a DATA. Datový výstup jedné LED je propojen s datovým vstupem (DIN) následující, čímž vzniká sériový "had".(viz:)
* **Uživatelské vstupy:** Obsahuje 9 tlačítek zapojených proti zemi (GND) mikrokontroleru. Osm tlačítek je vyhrazeno pro herní volby (proměna figurky) a jedno pro reset systému. (viz:)



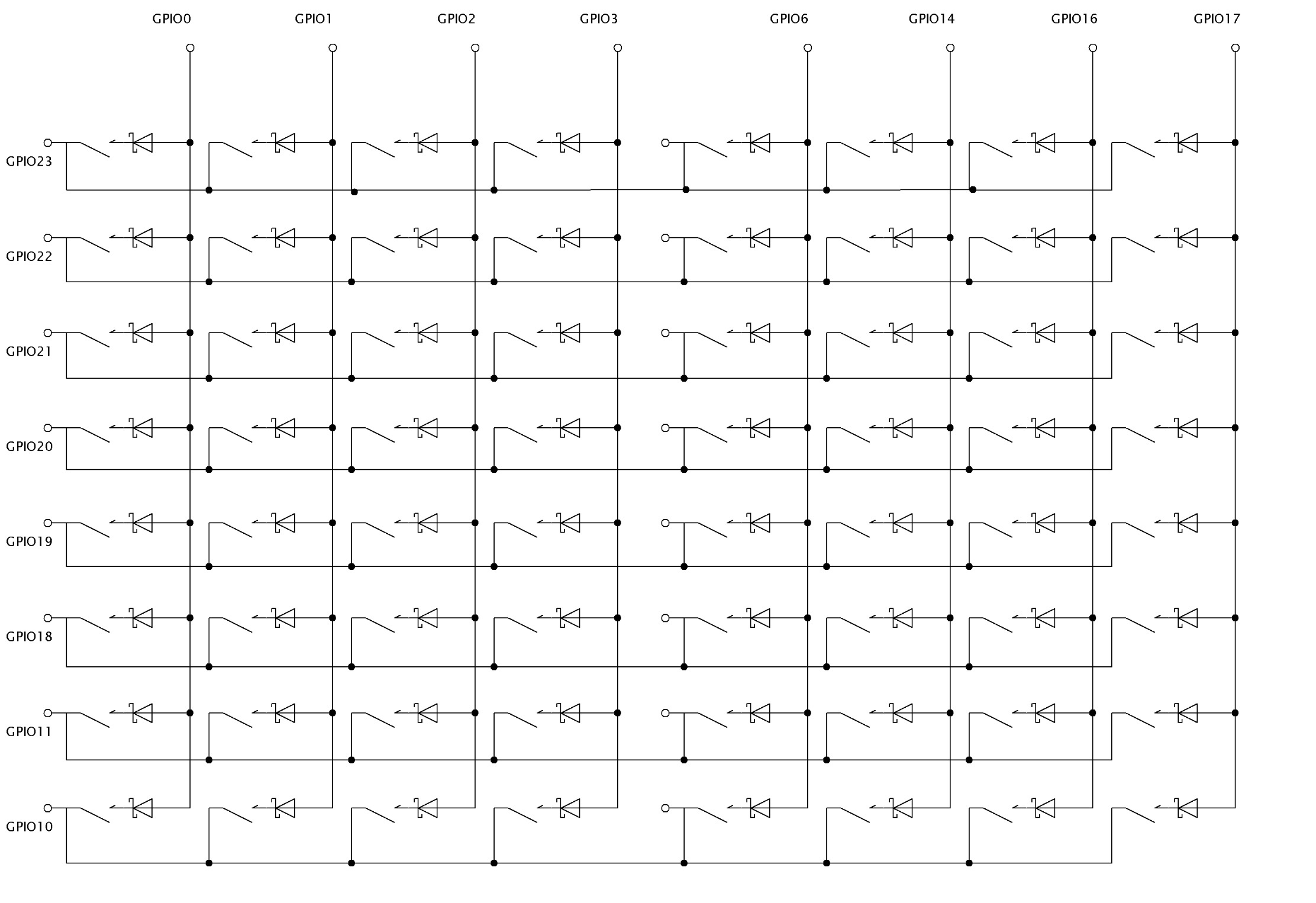




Obrázek - Schéma zapojení boost modulu

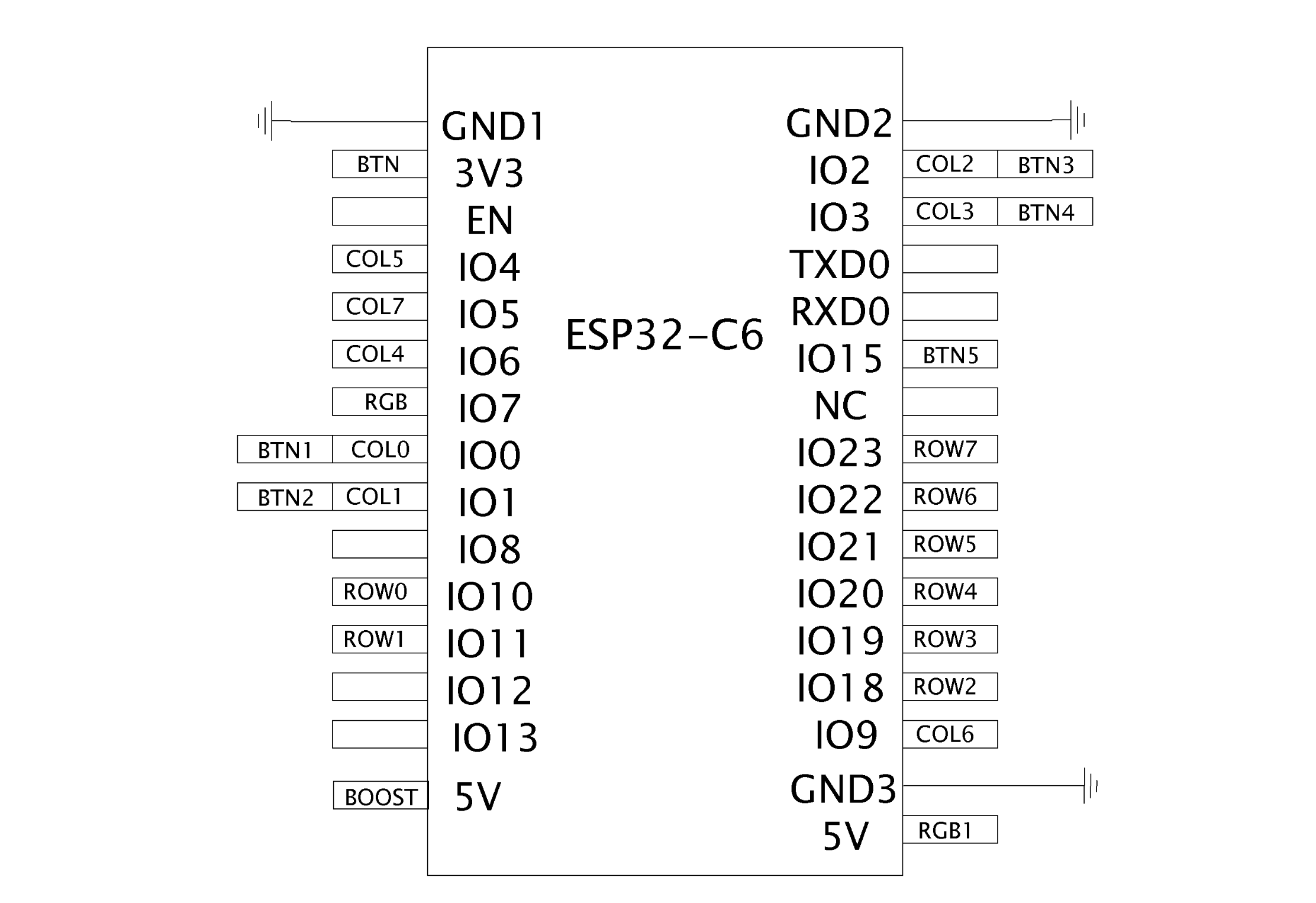
Obrázek - Schéma zapojení nabíjecího modulu

Obrázek - Schéma zapojení ledek WS2812B



[Magnetický jazýčkový spínač (Reed switch)](#_Magnetický_jazýčkový_spínač) [Schottkyho diody](#_Schottkyho_diody)

Obrázek - Schéma zapojení matice skenující přítomnost figurek



Obrázek - Schéma zapojení komponentů k MCU

## Postup a realizace elektrického rozvodu

Vzhledem k velkým rozměrům herní plochy (8x8 polí) a snaze o zachování co nejnižších nákladů nebylo pro projekt navrženo klasické pevné PCB. Výroba takto rozměrného plošného spoje by byla technicky náročná a ekonomicky neefektivní.

Realizace proto probíhala formou přímého pájení a propojování jednotlivých komponent vodiči:

Příprava a fixace: Jednotlivé jazýčkové spínače a RGB LED čipy byly nejprve přesně umístěny a zafixovány do připravených pozic v šasi navrženém ve Fusionu 360.

Pájení matice a LED řetězce: Spínače byly následně v sérii se Schottkyho diodami propojeny do matice 8x8 pomocí tenkých vodičů. Současně byly propojeny napájecí a datové kontakty všech 64 LED čipů do sériového „hada“ (daisy-chain).

Propojení s řídicí jednotkou: Všechny větve matice, ovládací tlačítka a datový vodič LED byly svedeny k mikrokontroleru ESP32-C6.

Oživení: Po dokončení pájení proběhla kontrola spojitosti cest multimetrem, aby se předešlo zkratům, a následně byl nahrán testovací software pro ověření funkčnosti všech 64 polí a všech 9 ovládacích tlačítek.

# FW – Vývoj programu pro zařízení

Firmware pro zařízení je implementován v jazyce C s využitím frameworku ESP-IDF a real-time operačního systému FreeRTOS. Jako řídicí jednotku jsme po počátečních experimentech s platformou AVR a zvažování čipu PIC32JH01 zvolili moderní ESP32-C6. Hlavním důvodem byla rozšiřitelnost, silná podpora komunity a integrované periferie. Programování a debugging probíhal přímo přes rozhraní USB Serial JTAG, což eliminovalo potřebu externího programátoru.

Hlavní úkoly a technické výzvy Cílem bylo vytvořit systém schopný v reálném čase detekovat pohyby na šachovnici (pomocí matice senzorů 8x8), řídit 73 adresovatelných LED WS2812B a současně obsluhovat Web Server. Při vývoji jsme museli řešit několik zásadních problémů:

Snímání matice: První návrhy na starších platformách selhávaly na problému s "duchy" (ghosting) při stisku více tlačítek. Finální řešení na ESP32-C6 využívá diodovou matici navrženou kolegou, což umožnilo spolehlivou detekci.

Memory Management (RAM): Vzhledem k omezené velikosti SRAM (256 KB) a riziku fragmentace heapu jsme implementovali Shared Buffer Pool. Místo dynamické alokace paměti (malloc) využíváme staticky předalokované buffery, což zajistilo stabilitu systému.

Thread-safety: Kritickým problémem bylo "přetahování" se o ovládání LED stripu mezi více tasky, což způsobovalo vizuální artefakty (glitche). Řešením bylo zavedení mutexů (led\_mutex, game\_mutex), které garantují, že ke sdíleným prostředkům přistupuje v daný okamžik vždy jen jeden task.

Debugging: Ladění multi-threadové aplikace bylo komplikované. Tradiční debugging často selhával, proto jsme přešli na detailní logování pomocí ESP\_LOG, které nám umožnilo identifikovat příčiny pádů (např. Stack Overflow nebo race conditions).

Softwarová architektura Finální aplikace běží v 8 nezávislých úlohách s celkovou alokací zásobníku (stack size) cca 73 KB. Díky scheduleru FreeRTOS má každá funkce (LED, Game logic, Web, Matrix scan) svou prioritu a nedochází k blokování kritických procesů. Firmware zabírá přibližně 1,5 MB z dostupné 2 MB Flash paměti. Pro zajištění stability je systém hlídán Watchdog Timerem (WDT) s timeoutem 10 sekund, který v případě zaseknutí provede systémový reset.

## Textový popis programu

### 3.1.1 Volba platformy a architektura systému

Jádrem celého zařízení je mikrokontrolér ESP32-C6 postavený na architektuře RISC-V. Tuto platformu jsme zvolili po úvodních pokusech s čipy AVR a zvažování platformy PIC32. Hlavním důvodem pro ESP32-C6 byla jeho vysoká rychlost, integrovaná podpora WiFi 6 dostatek periferií pro řízení LED pásku (RMT) a jednoduché rozšiřování projektu v budoucnu.

Program není napsán jako jedna dlouhá smyčka (while(1)), jak je běžné u jednodušších projektů, ale využívá operační systém reálného času FreeRTOS [**4**]. Ten nám umožnil rozdělit složitý program na 8 nezávislých částí – vlákno (Threads). Každý task řeší konkrétní problém (např. jeden ovládá LEDky, druhý počítá šachovou logiku, třetí řeší WiFi) a operační systém mezi nimi rychle přepíná. Díky tomu se nám nestává, že by se hra zasekla ve chvíli, kdy procesor počítá validitu tahu nebo čeká na síťovou komunikaci.

Celý systém je navržen jako Event-Driven (řízený událostmi). Tasky neběží zbytečně naprázdno, ale čekají, až se něco stane – například až přijde zpráva ve frontě (Queue) nebo se uvolní sdílený prostředek chráněný Mutexem.

### 3.1.2 Časování pomocí scheduleru

Během vývoje jsme museli zásadně změnit přístup k časování. V prostředí multitaskingu nelze používat klasickou funkci delay(), protože tato funkce tzv. "blokuje" procesor (je synchronní). Kdybychom ji použili například při čekání na animaci LED, celý procesor by se zastavil a přestala by fungovat detekce figurek i webový server.

#### Místo toho využíváme systémové funkce FreeRTOS [4]:

* **vTaskDelay():** Tato funkce neblokuje procesor, ale řekne plánovači (scheduleru), že daný task teď po určitou dobu nepotřebuje běžet. Plánovač tak může okamžitě přidělit výkon jinému tasku, který má práci.
* **vTaskDelayUntil()**: Tuto funkci používáme u kritických úloh (např. LED Task nebo Matrix Task). Zajišťuje, že se task spustí v naprosto přesných intervalech (např. přesně každých 33 ms), bez ohledu na to, jak dlouho trval výpočet v předchozím kroku. To je nezbytné pro plynulé animace a stabilní snímání senzorů.

### 3.1.3 Detailní popis hlavních úloh (Tasků)

Celý systém je rozdělen do osmi nezávislých úloh (Tasků), které běží paralelně pod správou operačního systému FreeRTOS. Každá úloha má přidělenou prioritu (čím vyšší číslo, tím důležitější) a vyhrazenou velikost zásobníku (Stack). Kromě našich vlastních úloh běží na pozadí systému i další procesy spravované frameworkem ESP-IDF (tzv. Managed Components), například pro WiFi stack nebo USB komunikaci.

#### Zde je popis klíčových úloh, které jsme implementovali:

**1. LED Task (Priorita 7 – Kritická)**

Funkce: Řídí všech 73 adresovatelných LED WS2812B **[2]**  (64 na šachovnici + 9 u ovládacích tlačítek) připojených na pin GPIO7.

Proč má nejvyšší prioritu: Protokol WS2812B **[2]**  vyžaduje extrémně přesné časování (v řádu stovek nanosekund). Jakékoli přerušení procesu odesílání dat by způsobilo chyby v barvách nebo blikání.

Princip fungování: Task běží v cyklu 33 ms (což odpovídá cca 30 FPS pro plynulé animace). Využívá systém "Batch Update" – změny barev se nejprve ukládají do vyrovnávací paměti (led\_pending\_changes) a teprve na konci cyklu se hromadně odešlou do pásku pomocí funkce led\_privileged\_batch\_commit. Přístup k LED bufferu je chráněn mutexem led\_unified\_mutex, aby nedocházelo ke kolizím s ostatními tasky.

**2. Matrix Task (Priorita 6 – Vysoká)**

Funkce: Zajišťuje skenování stavu šachovnice (matice 8×8 Reed switchů) a detekuje zvednutí nebo položení figurky.

Princip fungování: Skenování probíhá každých 10 ms. Task postupně aktivuje jednotlivé řádky (nastavením na LOW aktuálního řádku) a čte stavy sloupců. Aby se předešlo falešným detekcím způsobeným zákmitu mechanických kontaktů, je implementován softwarový debouncing – změna stavu je potvrzena až po třech stabilních měřeních.

Komunikace: Pokud je detekován pohyb (přechod z 1 na 0 nebo opačně), task vygeneruje událost PICKUP nebo DROP a odešle ji do fronty game\_command\_queue.

**3. Button Task (Priorita 5)**

Funkce: Obsluhuje 5 fyzických tlačítek (4 – jsou spojena pro bílého a černého pro jejich promoci pěšce a 1 pro reset hry).

Princip fungování: Běží v rychlém cyklu 5 ms pro okamžitou reakci na stisk. Dokáže rozlišit krátký stisk, dlouhý stisk (> 2 sekundy) a dvojklik. Stejně jako Matrix Task využívá pokročilý debouncing pro filtraci zákmitů tlačítek.

**4. Game Task (Priorita 4 – Mozek systému, Stack 10 KB)**

Tento task představuje centrální logiku celého systému. Zatímco ostatní tasky řeší "pouze" vstupy a výstupy (čtení senzorů, blikání LED), Game Task rozhoduje o tom, co se ve hře skutečně děje. Implementuje kompletní pravidla šachu (FIDE **[7]** ) a řídí průběh partie od prvního tahu až po mat.

#### Princip fungování (Smyčka):

Task běží v nekonečné smyčce s periodou 100 ms. V každém cyklu nejprve resetuje Watchdog Timer (WDT), aby potvrdil, že systém nezamrzl. Následně kontroluje frontu game\_command\_queue, kam mu chodí příkazy z ostatních částí systému – například zpráva z Matrix Tasku, že hráč zvedl figurku, nebo příkaz z Web Serveru pro novou hru.

#### Validace tahů (Chess Engine):

Jádrem tasku je validační systém. Šachovnice si pamatuje aktuální rozložení figur v poli board[8][8]. Když hráč zvedne figurku a někam ji položí, task si tento tah "nanečisto" vyzkouší v paměti (simulace).

* Ověří, zda tah odpovídá pohybu dané figury (např. střelec jen diagonálně).
* Zkontroluje, zda v cestě nestojí jiná figura (kromě jezdce).
* Nejdůležitější je kontrola šachu – pokud by tah vystavil vlastního krále šachu, je okamžitě zamítnut.

#### Logika "Pick & Drop" a Error Recovery:

Unikátní vlastností našeho firmwaru je systém ovládání. Hra reaguje na dvě události:

* PICKUP (Zvednutí): Task zjistí, kterou figurku hráč vzal. Pokud je to jeho figura, vypočítá všechny možné tahy a přes LED Task rozsvítí cílová pole zeleně.
* DROP (Položení): Po položení figury se provede validace. Pokud je tah platný, hra pokračuje. Pokud hráč udělal chybu (neplatný tah), spustí se systém Smart Error Recovery. Cílové pole začne blikat červeně a původní pole (odkud byla figurka vzata) svítí modře. Hra se "zastaví" a čeká, dokud hráč figurku nevrátí zpět na správné místo.

#### Speciální tahy a pravidla: Task zvládá i složitější šachová pravidla:

* Rošáda (Castling): Kontroluje, zda se král ani věž ještě nepohnuli a zda je cesta volná.
* En Passant (Braní mimochodem): Pamatuje si poslední tah pěšcem o dvě pole a umožňuje jeho vyhození v následujícím tahu.
* Promoce: Pokud pěšec dojde na konec desky, task hru pozastaví a čeká na stisk jednoho ze 4 tlačítek (Dáma, Věž, Střelec, Jezdec) pro výběr nové figury.
* Detekce konce hry: Po každém tahu kontroluje 5 podmínek: Mat, Pat, Nedostatek materiálu, Pravidlo 50 tahů a Třikrát opakovanou pozici.

Synchronizace dat:

Protože stav hry (kde stojí jaká figurka) potřebuje číst i Web Server (aby ji ukázal v prohlížeči) a UART Task (pro výpis do konzole), chráníme tato data pomocí Mutexu (game\_mutex). To zajišťuje, že se data nepomíchají, když k nim přistupuje více tasků najednou. Pro velké výpisy (např. historie tahů) používáme tzv. "Streaming Output", abychom nezahlcovali operační paměť RAM velkými textovými řetězci.paměť RAM.

**5. UART Task (Priorita 3 – Komunikace)**

Funkce: Zajišťuje komunikaci přes sériovou linku (USB Serial JTAG). Slouží jako příkazová řádka (CLI) pro ladění a ovládání.

Princip fungování: Task neustále čte vstupní buffer. Podporuje historii příkazů a barevný výstup (ANSI kódy). Parsuje textové příkazy (např. NEW\_GAME, STATUS) a posílá je ke zpracování do Game Tasku. Výstup do konzole je chráněn mutexem uart\_mutex, aby se texty z různých částí programu nemíchaly.

**6. Web Server Task (Priorita 3 – Konektivita)**

Funkce: Provozuje WiFi Access Point ("ESP32-CzechMate") a HTTP server s webovým rozhraním.

Princip fungování: Webová stránka je uložena přímo v paměti mikrokontroléru. Task obsluhuje REST API endpointy (např. /api/status, /api/board), přes které frontend získává data o hře ve formátu JSON. Pro úsporu paměti při odesílání velké HTML stránky využívá metodu "Chunked Transfer Encoding" – posílá data po malých částech.

**7. Animation Task (Priorita 3 – Neaktivní/Legacy, Stack 2 KB)**

Funkce: Původně navržen pro složité světelné efekty.

Aktuální stav: Během vývoje jsme zjistili, že je efektivnější řídit animace přímo v rámci LED Tasku nebo pomocí centrálního manažera animací. Tento task je v kódu stále přítomen (běží naprázdno), ale v současné verzi firmwaru nevykonává žádnou práci a slouží jako rezerva pro budoucí rozšíření.

**8. Test Task (Priorita 1 – Diagnostika, Stack 4 KB)**

Funkce: Slouží k automatické kontrole hardwaru a systému. Má nejnižší prioritu, takže běží jen tehdy, když procesor nemá nic jiného na práci.

Princip: Obsahuje sady testů (Hardware, System, Performance). Můžeme přes něj například spustit test všech LED diod, ověřit rychlost zápisu do paměti nebo zkontrolovat funkčnost všech tlačítek, aniž bychom museli hrát celou šachovou partii.

### 3.1.4 Technická řešení problémů (Paměť a bezpečnost)

Aby byl systém stabilní i při dlouhodobém běhu, museli jsme vyřešit několik pokročilých problémů:

#### Správa paměti (Shared Buffer Pool):

ESP32-C6 má omezenou paměť RAM. Zjistili jsme, že časté dynamické přidělování paměti (malloc) způsobuje její fragmentaci ("rozbití" na malé kousky) a následný pád systému. Proto jsme implementovali vlastní Shared Buffer Pool. Vytvořili jsme staticky alokované bloky paměti, které si jednotlivé tasky půjčují jen na dobu potřeby a pak je vrací. To vyřešilo problémy s nedostatkem paměti.

#### Thread-Safety a Mutexy:

V systému, kde běží více věcí najednou, se stávalo, že dva tasky chtěly zapisovat do LED pásku současně. To způsobovalo náhodné červené problikávání. Řešením bylo zavedení Mutexů (led\_unified\_mutex, game\_mutex). Mutex funguje jako zámek – když jeden task pracuje s LEDkami, zamkne si je a ostatní musí počkat, dokud neskončí.

Watchdog Timer (WDT): Pro jistotu jsme aktivovali bezpečnostní časovač. Každý důležitý task musí pravidelně hlásit, že "žije". Pokud by se program zasekl v nekonečné smyčce, Watchdog by to po 10 sekundách poznal a automaticky restartoval celý čip.

### 3.1.5 Průběh vývoje a překážky

Vývoj firmwaru nešel přímočaře a museli jsme překonat několik slepých uliček:

#### Začátky na AVR:

První pokusy probíhaly na čipu AVR s jednoduchým ovládáním LED matice MAX7219. Zde jsme narazili na limity výkonu a paměti, což vedlo k přechodu na ESP32.

#### Problém s "duchy" na matici:

Při prvním návrhu maticového skenování jsme neměli u senzorů diody. To způsobovalo, že při stisku více tlačítek najednou systém detekoval sepnutí i tam, kde nebylo (tzv. ghosting). Po konzultaci a úpravě hardwaru (přidání diod) se nám podařilo napsat spolehlivý skenovací kód.

#### Ladění pádů (Panic):

Ladění (debugging) multi-thread aplikace bylo složité, protože standardní debugger často selhával. Museli jsme zavést detailní logování pomocí ESP\_LOG do konzole. Díky tomu jsme odhalili například přetékání zásobníku (Stack Overflow) u náročnějších tasků a mohli jsme jim přesně nastavit velikost paměti.

#### Bootovací piny:

V jednu chvíli deska přestala startovat. Ukázalo se, že jsme omylem použili tzv. "strapping piny" (GPIO 0-5) pro tlačítka. Tyto piny ovlivňují start procesoru, takže jsme museli v kódu přemapovat ovládání na bezpečné piny a kvůli tomu musíme využívat time multiplexning.

## A screenshot of a diagram AI-generated content may be incorrect. Diagram toku dat a komunikace mezi úlohami

Obrázek - Diagram toku dat a komunikace mezi tasky

A qr code with blue squares and black squares

AI-generated content may be incorrect.

Podrobnější diagramy jsou v HTML dokumentaci, jenž můžete najít na [Gitu](https://github.com/AlfredKrutina/chess_esp32_c6_devkit.git).

*Poznámka: pro přístup na Git musíte být přihlášeni.*

## Postup vývoje programu

### 3.3.1 Historie a evoluce projektu

Vývoj softwaru nezačal přímo na čipu ESP32, ale předcházela mu fáze experimentování. První pokusy probíhaly už o prázdninách na platformě AVR (Microchip). V prostředí MPLAB jsem naprogramoval základní logiku, kde se šachy ovládaly přes UART a stav hry se zobrazoval na matici 8x8 řízené čipem MAX7219.

V této fázi jsme narazili na první zásadní hardwarový problém. Při pokusu o skenování matice senzorů docházelo k detekci tzv. "duchů" (falešných sepnutí), protože v návrhu chyběly diody. Matěj (kolega přes hardware) navrhl úpravu zapojení s diodami, což problém vyřešilo a umožnilo mi pokračovat v psaní kódu pro skenování.

Když jsme se dozvěděli, že bychom ve škole měli pracovat s procesorem PIC32JH01, rozhodl jsem se jít vlastní cestou a zvolil jsem ESP32-C6. Důvody byly praktické: moderní architektura a dostatečný výkon (160MHz), pokročilá konektivita (Wi-Fi 6 & Bluetooth 5 LE), poměr cena/výkon, jednoduché rozšíření a vylepšování do budoucna.

Samotná migrace kódu z AVR na ESP32 byla poměrně náročná jelikož kód změnil strukturu (stavový automat -> event driven s Freertos), ale díky tomu, že jsem měl kód rozdělený na vrstvy (použil jsem HAL – Hardware Abstraction Layer), i když ne dokonale, nemusel jsem přepisovat úplně celou logiku hry, ale jen tu část, která přímo ovládá piny procesoru, některé fce a hlavně rozdělit hlavní soubor z AVR na Tasky v Freertos. Postupně jsem k základu (UART + Matrix + Game) přidal Web Server, časovače a

animace LED pásku. Hodně mi pomohlo studium hotových kódů na internetu a využívání AI nástrojů pro generování "hrubých" částí kódu, které jsem pak upravoval a propojoval.

### 3.3.2 Vývojové nástroje a zálohování

Program jsem psal v editoru VS Code s rozšířením pro ESP-IDF. Co se týče zálohování a verzování, ze začátku jsem nepoužíval žádný pokročilý systém. Prostě jsem si dělal zálohy kopírováním celé složky projektu na disk v počítači, když něco fungovalo. Až v pozdější fázi, když byl projekt větší a složitější, jsem začal používat Git **[8]** pro ukládání funkčních verzí (když mi začlo něco fungovat), abych se měl k čemu vrátit, když se něco rozbije.

### 3.3.3 Známá omezení programu

I když je systém funkční, má několik omezení, o kterých víme a která vycházejí buď z hardwaru, nebo z nastavení softwaru:

* Omezení webového serveru: Na webové rozhraní se mohou připojit maximálně 4 uživatelé najednou. Je to dáno nastavením síťové TCP/IP knihovny (LWIP – light weight IP stack), která má vyhrazený omezený počet socketů. Roli taky hraje nedostatek RAM, protože každé Web socket připojení vyžaduje značnou velikost RAM paměti. Přes station mód lze ale připojit 10 zařízení.
* WiFi reconnect: Pokud šachovnice ztratí připojení k domácí WiFi síti (režim Station), neumí se sama znovu připojit. Je nutné zařízení restartovat.
* Paměť při dlouhých hrách: Přestože používáme Shared Buffer Pool, při extrémně dlouhých partiích může dojít k fragmentaci paměti (Heap). Tento stav ale hlídá Watchdog, který by v kritickém případě systém restartoval.
* Spolehlivost senzorů: Jazýčkové kontakty (Reed switches) jsou mechanické součástky a časem se mohou opotřebovat, nalomit, nebo "slepit", což by vedlo k chybám v detekci.
* Start systému: Program nekontroluje, jestli je šachovnice při zapnutí prázdná. Hra se spustí i s rozestavěnými figurkami (což je vlastně výhoda, pokud chceme pokračovat v rozehrané partii), ale matice začne skenovat okamžitě.

### 3.3.4 Plány do budoucna a rozšíření

Projekt je navržen tak, aby se dal dále rozvíjet. Příklady možného rozšíření:

* Analýza hry: Plánuji propojit šachovnici s externím počítačem (např. Raspberry Pi), na kterém poběží silný šachový engine Stockfish. Ten by mohl v reálném čase hodnotit kvalitu tahů, podobně jako to dělá web Chess.com.
* Chytrá domácnost (Smart Home): Chtěl bych do kódu přidat protokol MQTT. Díky tomu by se šachovnice mohla připojit do systému Home Assistant. To by umožnilo ovládat ambientní osvětlení šachovnice nebo ji integrovat do Apple HomeKitu jako chytré zařízení.

# Návrh mechanických částí zařízení

Celá konstrukce šachovnice byla od základu navržena v CAD softwaru **Autodesk Fusion 360**. Toto řešení bylo zvoleno kvůli možnosti přesného lícování vnitřních komponent s vnějším šasi. Finální model byl následně realizován pomocí **3D tisku**, což umožnilo vytvořit specifické držáky a šachty pro senzory přímo na míru.

**Vnější a vnitřní konstrukce:** Vnější část zařízení slouží jako designové šasi s integrovanými otvory pro USB-C konektor, vypínač a ovládací tlačítka. Vnitřní prostor je rozdělen na několik sekcí, které oddělují napájecí část (akumulátory) od citlivé řídicí elektroniky a matice senzorů. Kompletace probíhala postupným vkládáním komponent do vytištěných lůžek a jejich následným zajištěním.

**Bezpečnost a odolnost:** Krabička poskytuje základní ochranu proti prachu a mechanickému poškození vnitřních spojů. Vzhledem k tomu, že je zařízení určeno pro interiérové použití, nebyla vyžadována vysoká voděodolnost (standard IP). Materiál (PLA) je dostatečně pevný pro běžnou manipulaci a bezpečně izoluje veškeré elektrické spoje.

**Servisní přístup:** Konstrukce je navržena s ohledem na snadnou opravitelnost. Vrchní víko je odnímatelné (upevněné pomocí šroubů), což umožňuje rychlý přístup k bateriím i kabeláži v případě potřeby servisu nebo úpravy hardwaru.

**Budoucí vylepšení:** Aktuální verze krabičky je plně funkční, nicméně v budoucnu plánuji vylepšení v podobě povrchové úpravy (broušení a lakování) pro luxusnější vzhled či kompletně jiný desing pro jednodušší manipulaci nebo integraci magnetického uzavírání víka pro ještě snazší přístup do útrob zařízení.

## Výběr / návrh mechanických částí zařízení

Kompletní mechanický návrh šachovnice i všech figurek byl autorsky zpracován v softwaru **Autodesk Fusion 360**, což umožnilo vytvořit vnější designové šasi s integrovanými ovládacími prvky a složitou vnitřní strukturu pro přesné uložení senzorů, elektroniky a akumulátorů. Výroba byla realizována technologií 3D tisku z materiálu PLA, přičemž důraz byl kladen na konstrukční pevnost a servisovatelnost. Spodní víko je navrženo jako odnímatelné pro snadný přístup k hardwaru.

Obrázek - model šachovnice

### Obsah obrázku šachová figurka, desková hra, šachy Obsah generovaný pomocí AI může být nesprávný.Obsah obrázku šachová figurka, území, šachy, venku Obsah generovaný pomocí AI může být nesprávný.Obsah obrázku šachová figurka, venku, šachy, Šachovnice Obsah generovaný pomocí AI může být nesprávný.4.1.1 Rendery modelu:

Obrázek – render 1

Obsah obrázku venku, budova, obloha, území

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - render 3

Obrázek - render 2

# Odzkoušení hotové ročníkové práce v praxi

JE POTŘEBA AKTUALIZOVAT OBRAZKY S NEJNOVEJSIM PROTOTYPEM   
A DÁT JE DO PŘÍLOHY

Zde můžete vidět prototyp v akci:

Načítací sekvence :

Obsah obrázku zelené, světlo, interiér

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - načítací sekvence na 100 %

Možné tahy:

Obsah obrázku noc, tma, světlo, interiér

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - možné tahy bílého hráče

Zvednutá figurka A2 a možné jeho tahy:

Obsah obrázku osoba, světlo, klávesnice, interiér

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - možné tahy s ukázkou tahu bílého hráče

Možné tahy protivníka:

Obsah obrázku svíčka, osoba, světlo

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - tahy protivníka

Zvolená figurka B7 a jeho možné tahy:

Obsah obrázku interiér, svíčka, světlo

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - tahy protivníka s ukázkou jeho tahu

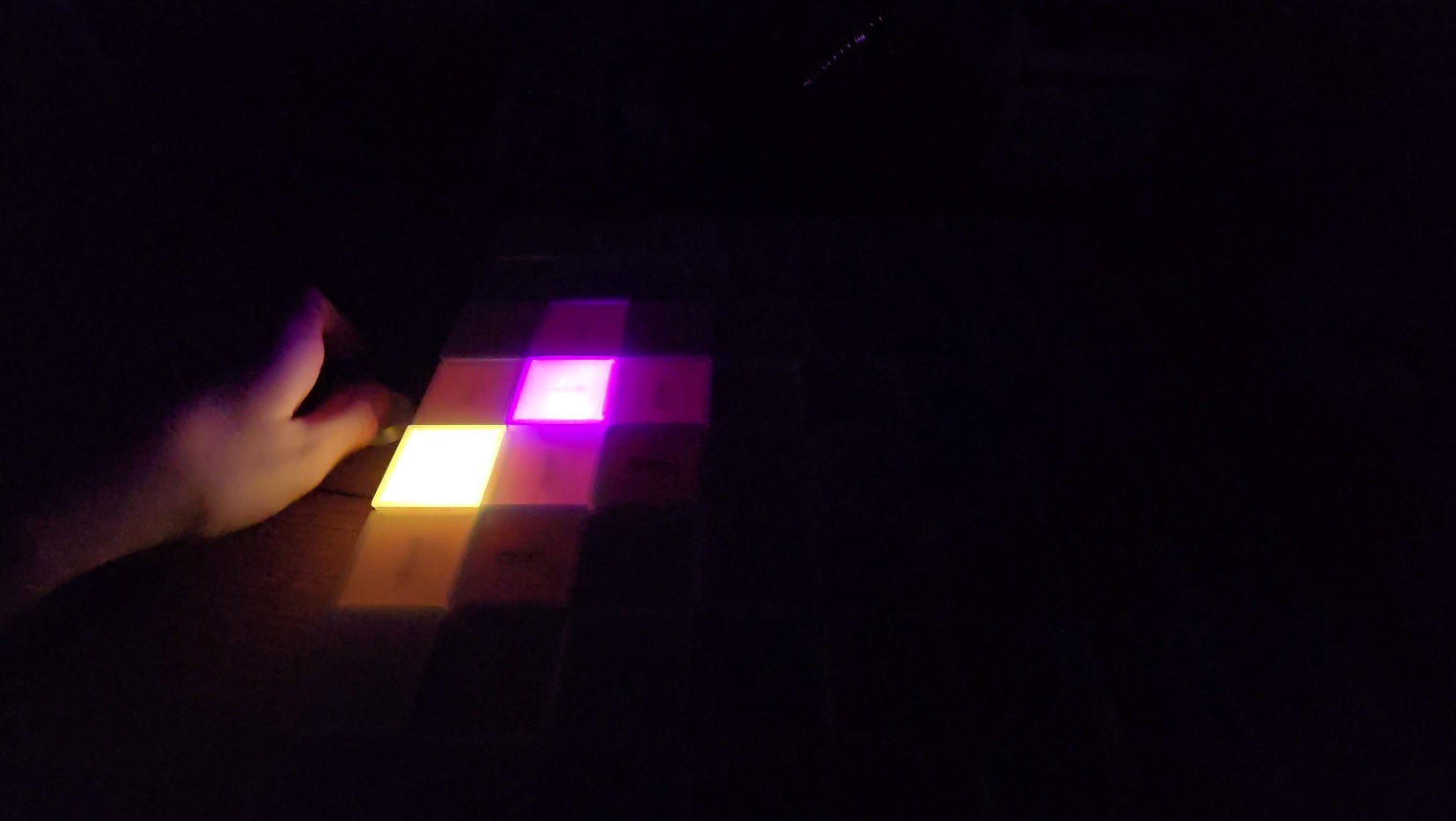
Možné tahy Pěštce na A4 a jeho možné tahy:

Obsah obrázku svíčka, osoba, světlo, interiér

Obsah generovaný pomocí AI může být nesprávný.

Obrázek - braní figurky

Figurka bere jinou figurku:



Obrázek - ukázka braní figurky v 2 fázi

# Závěr

Cílem této ročníkové práce byl návrh a kompletní realizace chytré šachovnice „CzechMate“ postavené na platformě ESP32-C6. Podařilo se nám vytvořit funkční zařízení, který v reálném čase detekuje polohu figurek pomocí matice jazýčkových spínačů a vizualizuje tahy prostřednictvím inteligentních RGB LED diod. Celý systém je integrován do vlastního šasi navrženého ve Fusion 360 a je schopen autonomního provozu díky bateriovému napájení s kapacitou 9600 mAh.

**Nejobtížnější části vývoje**: Během realizace jsme narazili na několik zásadních výzev. V hardwarové části bylo nejnáročnější precizní ruční pájení 64 polí a řešení problému s tzv. „ghostingem“ (falešnou detekcí sepnutí), který jsme museli eliminovat přidáním Schottkyho diod ke každému senzoru. V softwarové části byla největší překážkou stabilita multi-threadového prostředí v FreeRTOS. Museli jsme vyřešit kolize při přístupu k LED diodám pomocí mutexů a optimalizovat správu paměti RAM, abychom předešli pádům systému při výpočtech šachové logiky.

**Vize do budoucna**: Na projektu hodláme pracovat i nadále. Naší vizí je vylepšit vnější design šachovnice (broušení a lakování 3D tisku, či kompletně jiný design) a rozšířit firmware o hru proti vestavěnému šachovému enginu nebo o možnost hrát online proti lidem z celého světa. Zvažujeme také integraci magnetického zavírání víka pro snazší servisní přístup.

**Zhodnocení a přínos**: Práce na tomto projektu pro nás byla obrovským přínosem. Naučili jsme se nejen pokročilé modelování v CADu a precizní elektronickou montáž, ale především jsme pronikli do hloubky programování embedded systémů a operačních systémů RTOS. Svůj výkon hodnotíme kladně, jelikož se nám podařilo zrealizovat komplexní zařízení, které propojuje mechaniku, elektroniku a složitou softwarovou logiku do jednoho funkčního celku.

# Seznam obrázků

[Obrázek 1- Magnetický jazýčkový spínač (Reed switch) 5](#_Toc219573671)

[Obrázek 2 - Schottkyho diody 5](#_Toc219573672)

[Obrázek 3 - ESP32-C6 6](#_Toc219573673)

[Obrázek 4 - UPS Li-ion Boost modul s nabíječkou, 5V 6](#_Toc219573674)

[Obrázek 5 - GeB Li-Ion Baterie 3x18650 1S3P 3.7V 9600mAh 7](#_Toc219573675)

[Obrázek 6 - nteligentní RGB LED NeoPixel, WS2812B, 10x10, 5050, 5V 7](#_Toc219573676)

[Obrázek 7 - Tlačítko 6x6x5mm 8](#_Toc219573677)

[Obrázek 8 - Spínač kovový s aretací – Power, stříbrný 16mm 3-6V 8](#_Toc219573678)

[Obrázek 9 - Nabíječka Li-ion článku TP4056 s ochranou USB-C 9](#_Toc219573679)

[Obrázek 10 - blokové schéma 10](#_Toc219573680)

[Obrázek 11 - Schéma zapojení boost modulu 12](#_Toc219573681)

[Obrázek 12 - Schéma zapojení nabíjecího modulu 12](#_Toc219573682)

[Obrázek 13 - Schéma zapojení ledek WS2812B 12](#_Toc219573683)

[Obrázek 14 - Schéma zapojení matice skenující přítomnost figurek 13](#_Toc219573684)

[Obrázek 15 - Schéma zapojení komponentů k MCU 14](#_Toc219573685)

[Obrázek 16 - Diagram toku dat a komunikace mezi tasky 21](#_Toc219573686)

[Obrázek 17 - model šachovnice 24](#_Toc219573687)

[Obrázek 18 – render 1 25](#_Toc219573688)

[Obrázek 19 - render 3 25](#_Toc219573689)

[Obrázek 20 - render 2 25](#_Toc219573690)

[Obrázek 21 - načítací sekvence 26](#_Toc219573691)

[Obrázek 22 - možné tahy 26](#_Toc219573692)

[Obrázek 23 - možné tahy 27](#_Toc219573693)

[Obrázek 24 - tahy protivníka 27](#_Toc219573694)

[Obrázek - 25 tahy protivníka s ukázkou jeho tahu 28](#_Toc219573695)

[Obrázek 26 - braní figurky 28](#_Toc219573696)

# Reference

**[1] ESPRESSIF SYSTEMS.** *ESP32-C6 Series Datasheet: 2.4 GHz Wi-Fi 6 & Bluetooth 5 (LE)* [online]. Verze 1.2. Shanghai: Espressif Systems, 2024 [cit. 2026-06-09]. Dostupné z: <https://www.espressif.com/sites/default/files/documentation/esp32-c6_datasheet_en.pdf>

**[****2] WORLDSEMI.** *WS2812B Intelligent control LED integrated light source Datasheet* [online]. Dongguan: World-semi, 2011 [cit. 2026-06-09]. Dostupné z: <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>

**[3] ESPRESSIF SYSTEMS.** *ESP-IDF Programming Guide* [online]. 2024 [cit. 2026-06-09]. Dostupné z: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32c6/>

**[****4] AMAZON WEB SERVICES.** *FreeRTOS Kernel Developer Guide* [online]. 2024 [cit. 2026-06-09]. Dostupné z: [https://www.freertos.org/Documentation/02-Kernel/02-Kernel-developers-guide](https://www.google.com/search?q=https://www.freertos.org/Documentation/02-Kernel/02-Kernel-developers-guide)

**[5] HYMEL, Shawn.** *Introduction to RTOS (FreeRTOS)* [Video online]. Digi-Key Electronics, 2020 [cit. 2026-06-09]. Dostupné z: [https://www.youtube.com/playlist?list=PLB-czhE0JwoVdXPRTT71azQP1NbYNVksL](https://www.google.com/search?q=https://www.youtube.com/playlist%3Flist%3DPLB-czhE0JwoVdXPRTT71azQP1NbYNVksL)

**[6] LAGUE, Sebastian.** *Coding Adventure: Chess AI* [Video online]. 2019 [cit. 2026-06-09]. Dostupné z: <https://www.youtube.com/watch?v=U4ogK0MIzqk>

**[7] FIDE.** *Laws of Chess: For competitions starting on or after 1 January 2023* [online]. Lausanne: International Chess Federation, 2023 [cit. 2026-06-09]. Dostupné z: <https://handbook.fide.com/chapter/E012023>

**[8] KRUTINA, Alfréd.** *CZECHMATE Firmware Repository* [online]. 2026 [cit. 2026-06-09]. Dostupné z: <https://github.com/AlfredKrutina/chess_esp32_c6_devkit>