

Chytrý IoT detektor proti neoprávněnému vniknutí do chráněného prostoru

Jakub Hrdlička

Bakalářská práce
2025



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav bezpečnostního inženýrství

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Jakub Hrdlička
Osobní číslo: A22219
Studijní program: B1032A020001 Bezpečnostní technologie, systémy a management
Forma studia: Prezenční
Téma práce: Chytrý IoT detektor proti neoprávněnému vniknutí do chráněného prostoru.
Téma práce anglicky: Smart IoT Detector Against Unauthorized Entry into a Protected Area.

Zásady pro vypracování

- Popište vhodné technologie pro detekci neoprávněného vniknutí do chráněného prostoru.
- Seznamte se s možnostmi odesílání zpráv z vlastního IoT zařízení na platformy pro instantní komunikace.
- Proveďte průzkum trhu s cílem identifikovat vhodný hardware pro tvorbu IoT zařízení.
- Navrhněte IoT zařízení, které detekuje neoprávněné vniknutí do chráněného prostoru se schopností notifikace vniknutí na vybranou platformu pro instantní komunikace.
- Vytvořte prototyp, který ověří realizovatelnost návrhu.
- Naznačte další vývoj v těchto technologiích.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. MATOUŠEK, David a Bohumil BRTNÍK. Aplikace algoritmů číslcového zpracování signálů. Praha: BEN – technická literatura, 2014. ISBN 978-80-7300-478-1.
2. LUKÁŠ, Luděk a kolektiv. Bezpečnostní technologie, systémy a management I. Zlín: VeRBuM, 2011. ISBN 978-80-87500-05
3. MALÝ, Martin. Hradla, volty, jednočipy. Vyd. 1. Praha: CZ.NIC, 2017. 512 s. ISBN 978-80-88168-26-3.
4. MALÝ, Martin. ESP32 prakticky Vyd. 1. Praha: CZ.NIC, 2024. 432 s. ISBN 978-80-88168-79-9.

Vedoucí bakalářské práce: **Ing. Rudolf Drga, Ph.D.**
Ústav bezpečnostního inženýrství

Datum zadání bakalářské práce: **11. prosince 2024**
Termín odevzdání bakalářské práce: **2. června 2025**



doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

Ing. Jan Valouch, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 11. prosince 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne
podpis autora

.....

ABSTRAKT

Bakalářská práce se zaměřuje na vytvoření bota pro platformu Discord, který online odesílá zprávy o stavu otevření / zavření dveří. Teoretická část zahrnuje základní informace o IoT, obecných požadavcích na HW a SW pro tvorbu IoT zařízení, popis různých komunikačních platforem a mikropočítačů. Na základě požadavků byl proveden výběr cílové komunikační platformy a mikropočítače se zdůvodněním volby. Praktická část pak definuje funkční požadavky na HW a SW zařízení, návrh HW zařízení, vytvoření SW pro bota odesílajícího zprávy na Discord, popis programu bota a samotné uvedení bota do provozu. Práce ukázala, že s dostupnou dokumentací a odpovídajícími schopnostmi je možné v domácích podmínkách postavit zajímavé a užitečné IoT zařízení.

Klíčová slova: IoT, FreeRTOS, Discord, espressif, ESP32, ESP32C3, WIFI

ABSTRACT

The Thesis focuses on creating a bot for Discord, that can send online messages about the state of closed/open door. The theoretical part includes basic information about IoT, general HW and SW requirements for creation of an IoT device, description of different communication platforms and microcomputers. Based on the requirements, the selection of the target communication platform and microcomputer has been made with justification for the choice. The practical part then defines the functional requirements for the HW and SW of the bot sending messages to Discord, the description of the bot program and the actual commissioning of the bot. The work has shown that with the available documentation and appropriate capabilities, it is possible to build an interesting and useful IoT bot even in home conditions.

Keywords: IoT, FreeRTOS, Discord, espressif, ESP32, ESP32C3, WIFI

Tímto bych chtěl poděkovat vedoucímu bakalářské práce, Ing. Rudolfovi Drgovi, Ph. D. za umožnění vzniku bakalářské práce, jeho konzultace, cenné rady a jeho ochotu. Také bych chtěl poděkovat mé rodinně a přátelům za podporu a trpělivost.

„I too am Whooo. But I'm also Wheee! So the Wheee balances the Whooo.” – Esquie

OBSAH

ÚVOD.....	11
I TEORETICKÁ ČÁST	12
1 TECHNOLOGIE PRO DETEKCI NEOPRÁVNĚNÉHO VNIKNUTÍ DO CHRÁNĚNÉHO PROSTORU.....	13
1.1 ROZDĚLENÍ DETEKTORŮ	13
2 INTERNET VĚCÍ (INTERNET OF THINGS)	15
2.1 JAK FUNGUJE IoT?	16
2.2 JAK CHCE PRÁCE VYUŽÍT IoT?	17
3 HW A SW PRO TVORBU IOT ZAŘÍZENÍ	18
3.1 POŽADAVKY NA HW A SW PRO TVORBU IoT ZAŘÍZENÍ	18
3.2 PŘEHLED DOSTUPNÝCH HW A SW PRO TVORBU IoT ZAŘÍZENÍ.....	19
3.2.1 Seeed Studio XIAO ESP32C3	20
3.2.2 Seeed Studio XIAO ESP32S3.....	21
3.2.3 Raspberry Pi Pico W	22
3.2.4 M5Stamp Pico Mate.....	23
3.2.5 M5Stamp Pico Mate DIY Kit	24
3.2.6 M5Stamp C3	25
3.2.7 M5Stamp C3U	26
3.2.8 M5Stamp S3.....	27
3.2.9 ESP32-C6-DEVKITM-1	28
3.2.10 Arduino Nano 33 IoT	29
3.2.11 Wemos S2 Mini Wifi ESP32-S2.....	30
3.3 VYBRANÝ HW A SW PRO TVORBU IoT ZAŘÍZENÍ	31
4 PLATFORMY PRO INSTANTNÍ KOMUNIKACI A MOŽNOST ODESÍLÁNÍ ZPRÁV Z IOT ZAŘÍZENÍ.....	32
4.1 FUNKČNÍ A NEFUNKČNÍ POŽADAVKY NA KOMUNIKAČNÍ PLATFORMU	32
4.2 PŘEHLED PLATFORM A JEJICH MOŽNOSTÍ INTEGRACE S IoT ZAŘÍZENÍM	32
4.2.1 DISCORD	32
4.2.2 TELEGRAM.....	33
4.2.3 SLACK.....	34
4.2.4 WhatsApp.....	34
4.3 VÝBĚR PLATFORMY PRO INTEGRACI IoT ZAŘÍZENÍ.....	35
4.3.1 Srovnání vlastností, důvod pro výběr konkrétní platformy (Discord)	35
5 POPIS INTEGRACE NA DISCORD	37
5.1 ZPŮSOBY KOMUNIKACE S DISCORD PLATFORMOU	37
5.2 AUTENTIZACE A AUTORIZACE VOLÁNÍ DISCORD BOTA.....	38
5.3 PROBLEMATIKA TLS A OVĚŘOVÁNÍ CERTIFIKÁTŮ DISCORDU	39
5.4 KNIHOVNA PRO KOMUNIKACI S DISCORD PLATFORMOU.....	40

II	PRAKTICKÁ ČÁST	41
6	FUNKČNÍ POŽADAVKY NA HW A SW BOTA	42
7	NÁVRH HW SCHÉMATU ZAŘÍZENÍ.....	43
8	VYTVÁŘENÍ BOTA PRO DISCORD.....	45
8.1	ZALOŽENÍ NOVÉ APLIKACE V DISCORD DEVELOPER PORTÁLU.....	45
8.2	KONFIGURACE DISCORD BOTA.....	45
9	NÁVRH SW BOTA	48
9.1	NAPLNĚNÍ FUNKČNÍCH POŽADAVKŮ	48
9.1.1	Požadavek 1 – Možnost předat do bota informace pro připojení k WiFi s tím, že tyto informace zůstanou uloženy v paměti i po resetu zařízení, vč. Výpadku napájení.	48
9.1.2	Požadavek 2 – Bot by měl indikovat stav připojení k WiFi – např. specifickým patternem blikání led	51
9.1.3	Požadavek 3 – Bot umí snímat stav otevření dveří	51
9.1.4	Požadavek 4 – Bot by se měl umět přihlásit do Discordu.....	53
9.1.5	Požadavek 5 – Bot by měl udržovat online stav na Discordu	54
9.1.6	Požadavek 6 – Po připojení k WiFi a Discordu bot pošle aktuální informaci o stavu senzoru.....	54
9.1.7	Požadavek 7 – Bot kontinuálně snímá stav senzoru otevřených dveří a změny stavu odesílá na Discord	55
9.1.8	Požadavek 8 – Bot umí reagovat na zmínky o něm na Discordu a reagovat na ně odesláním aktuálního stavu snímače otevřených dveří	56
9.2	STRUKTURA PROGRAMU	57
9.2.1	main.c	57
9.2.2	led task.h a led task.c	58
9.2.3	wifiprovisioning.h a wifiprovisioning.c	58
9.2.4	discordbot.h a discordbot.c.....	59
10	UVEDENÍ BOTA DO PROVOZU A OVĚŘENÍ FUNKČNOSTI	61
11	MOŽNOSTI DALŠÍHO ROZVOJE	65
	ZÁVĚR.....	66
	SEZNAM POUŽITÉ LITERATURY	67
	SEZNAM OBRÁZKŮ	72
	SEZNAM TABULEK.....	73
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	74
	SEZNAM PŘÍLOH.....	76

ÚVOD

V současné době, kdy se technologie stávají nedílnou součástí našeho každodenního života, roste také potřeba chránit osobní i komerční prostory proti neoprávněnému vniknutí.

Kombinace moderních senzorů, efektivních komunikačních platforem a pokročilých softwarových řešení umožňuje vytvářet inteligentní systémy, které nejen detekují narušení, ale také okamžitě informují odpovědné osoby o vzniklé situaci.

Internet věcí (IoT – Internet of Things) by se mohl stát klíčovým prvkem v oblasti bezpečnostních systémů díky své schopnosti propojovat fyzická zařízení s digitálním světem. IoT technologie umožňují rychlý sběr, zpracování a přenos dat, což zajišťuje efektivní monitorování a okamžitou reakci na bezpečnostní incidenty.

Tato bakalářská práce se zaměřuje na návrh chytrého IoT detektoru, který bude detekovat neoprávněné otevření dveří a zasílat upozornění prostřednictvím platformy Discord. Cílem je vytvořit jednoduchý, cenově dostupný a spolehlivý systém, který může být snadno integrován do domácností.

Tato bakalářská práce pokrývá jak teoretickou, tak praktickou část řešení bota. Teoretická část popisuje základní principy detekce vniknutí, technologie IoT a dostupné hardwarové a softwarové platformy. Hardwarové a softwarové platformy jsou popsány tak, aby bylo možné vybrat cílové technologie, na kterých je následně postavena praktická část.

Teoretická část krátce zmíní i způsob autentizace a autorizace volání IoT detektoru na vybranou platformu Discord. Zmíní se i o obraně proti útoku, jehož cílem by bylo zneužití identity bota k tomu, aby hlásil falešný stav zabezpečení prostoru.

Praktická část v úvodu definuje funkční požadavky na HW i SW řešení bota, které jsou pak dále rozpracovány. Je ukázáno, jak jsou jednotlivé požadavky naplněny. Těžištěm praktické části je vytvoření reálného modelu bota a vytvoření programu, který řídí jeho činnost. Je popsána struktura programu tak, aby bylo pro čtenáře pochopitelné, jak SW vybavení pracuje.

V praktické části je ukázáno i to, jak si stáhnout zdrojové kódy z github, sestavit SW, nakonfigurovat vše potřebné, spustit bota a pak jen sledovat, jak posílá zprávy, když se něco děje.

I TEORETICKÁ ČÁST

1 TECHNOLOGIE PRO DETEKCI NEOPRÁVNĚNÉHO VNIKNUTÍ DO CHRÁNĚNÉHO PROSTORU

Senzory přítomnosti objektů detekují pohyb případně i pouhou přítomnost osoby ve střeženém prostoru, případně reagují na určitou činnost v tomto objektu (v tomhle případě otevření dveří).

Tyto senzory se využívají v systému elektrické zabezpečovací signalizace pro střežení vnitřních prostor v budovách i na vnějších prostranstvích.

Systém elektrické zabezpečovací signalizace slouží k detekci vniknutí nežádoucích osob do objektu, monitoruje neoprávněný pohyb nežádoucích osob po objektu, sleduje sabotážní činnosti a signály o tomto narušení předává na určené místo, kterým pro nás bude platforma Discord.

Některé níže uvedené detektory se využívají i v jiných zapojeních, systémech odběru energie, v elektrických spotřebičích pro domácnost atd. Příkladným využitím je pohybové čidlo, které spíná osvětlení v dané místnosti.[1]

1.1 Rozdělení detektorů

Detektory se rozdělují do dvou skupin, pasivní a aktivní

Pasivní (v ČSN (Česká technická norma) 50131-1 označovány jako typ A) - k vlastní detekci nepotřebují elektrické napájení, pozitivní detekci vyvolá mechanická činnost nežádoucí osoby na předmětech ve střeženém prostoru, případně bezpečnostní činnost pověřené osoby. Mechanickou činností dochází k rozpojení či spojení kontaktu mechanického spínače či mikrospínače, a to buď vlastní činností, nebo přenosem, např. přes magnetické pole.[1]

Aktivní (v ČSN 50131-1 označovány jako typ B) - k vlastní detekci je třeba napájení detekčního čidla a souvisejícího vyhodnocovacího obvodu. Pozitivní detekci vyvolá už pouhá přítomnost, pohyb nebo činnost nežádoucí osoby ve střeženém prostoru. Fyzikální princip činnosti těchto senzorů je odvozený od změny měřitelných vlastností ve střeženém prostoru, vyvolaný přítomností nebo činností osob.[1]

Mezi pasivní detektory patří:

Magnetický kontakt – se nejčastěji využívá pro detekci nežádoucího otevření dveří (které jsou cílem téhle práce), oken nebo jiných otevíratelných částí. Sestává ze dvou částí, magnetické a kontaktní. Kontaktní se připevňuje na statickou a magnetická na pohyblivou část potenciálního vstupu do střeženého objektu. V klidovém režimu (zavřený vstup) jsou obě části od sebe vzdáleny o tzv. pracovní mezeru (pohybuje se v řádu milimetrů a je pevně definována výrobcem). V případě, že se části od sebe vzdálí nad hranici pracovní mezery (otevření vstupu), dojde k tak velké změně magnetického pole, že sepnou mechanický mikrospínač v kontaktní části, který citlivý na magnetické pole.[1]

Tísňový hlásič – slouží k vyvolání poplachu pověřenou osobou (obsluhou) při ohrožování osob či majetku ve střeženém objektu. Hlásiče bývají pevně instalované na veřejnosti skrytém místě v dosahu obsluhy, nejčastěji u pokladen. Standardní mechanické provedení tísňových je se sklopnou páčkou, která spíná mikrospínač.

V některých případech jsou používány bezdrátové tísňové hlásiče, které nosí pověřená osoba (nejčastěji bezpečnostní služba) stále u sebe.[1]

Detektor poslední bankovky – bývá využíván v přihrádkách pro uložení papírových bankovek u pokladen nebo trezorů. Využita je skutečnost, že papírové bankovky jsou dielektrikum. Vloženy jsou mezi kontakty speciálního spínače a tím je vlastně udržují v rozpojeném stavu. Odebráním poslední bankovky dochází k sepnutí speciálního spínače. [1]

Mezi aktivní detektory patří:

Akustické detektory – jsou založeny na analýze akustického signálu, který vzniká jako důsledek při vniku do střeženého prostoru. V drtivé většině se jedná o detekci zvuku tříštění skla při rozbití skleněné zábrany. Volbou vhodného algoritmu zpracování akustického signálu snímaného mikrofonom zcela minimalizuje možnost vzniku falešného poplachu.[1]

Světelné závory – detekují přerušení světelného svazku nebo změny kontrastu či osvětlení vyvolaných přítomností nebo pohybem v objektu. Skládají se z vysílací a přijímací části. Části se instalují naproti sobě tak, aby tvořily pomyslnou závoru v místech, kde je možný vstup do střeženého objektu. Vysílaný světelný paprsek bývá volen mimo oblast viditelného světla, v drtivé většině v infračervené části spektra.[1]

2 INTERNET VĚCÍ (INTERNET OF THINGS)

Internet věcí (dále jako IoT) zjednodušeně odkazuje na fyzické objekty propojené skrz drátové a bezdrátové technologie. Přesněji odkazuje na kolekci internetově propojených zařízení, které jsou schopné komunikovat nezávisle přes internet, bez potřeby osoby, která by započínala komunikaci

IoT je dnes všude. Používá se ve velké škále průmyslů a má využití jak pro korporátní, tak pro uživatelské použití. Například dnes automobily využívají tucty senzorů, které shromažďují a přeposílají data pro bezpečnost, údržbu, zábavu a mnoho dalších důvodů. Tyhle auta propojené internetem se považují za součást internetu věcí, jelikož komunikují s ostatními zařízeními skrz internet založenými na vstupech z prostředí, nejen skrze přímý lidský zásah[2]

2.1 Jak funguje IoT?

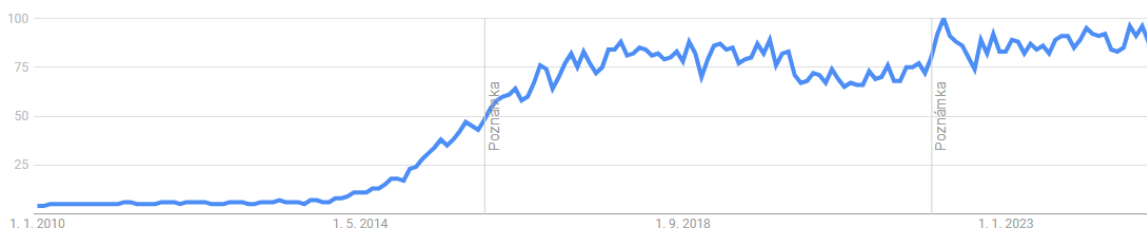
IoT funguje kombinací bezdrátových síťových technologií, fyzických zařízení, pokročilou analýzou dat a cloudovým computingem.

Základní proces toho, tak funguje IoT je následující:

- Skupina fyzických zařízení je drátově či bezdrátově propojena společně či individuálně k centrální oblasti
- Zařízení sbírají data z okolí pomocí senzorů
- Data se poté shromažďují buďto na cloudu, na serveru či na samotném zařízení
- Data se poté zpracovávají, často strojovým učením a umělou inteligencí
- Vyhodnocená data jsou použity fyzickým zařízením k provedení určité akce

Tento proces by se dal využít například u chytrého termostatu:

- Termostat má čidlo, které snímá teplotu v místnosti
- Termostat tato data ukládá a zpracovává
- Pokud teplota překročí určitou hodnotu, termostat teplotu automaticky reguluje na předurčenou hodnotu.
- Termostat přenáší pravidelné údaje o teplotě do externí databáze poskytovatele energie prostřednictvím bezdrátové sítě.
- Aplikace pro analýzu dat získává z těchto údajů v průběhu času poznatky, které umožňují zlepšit energetickou účinnost úpravou nastavení teploty termostatu.[2]

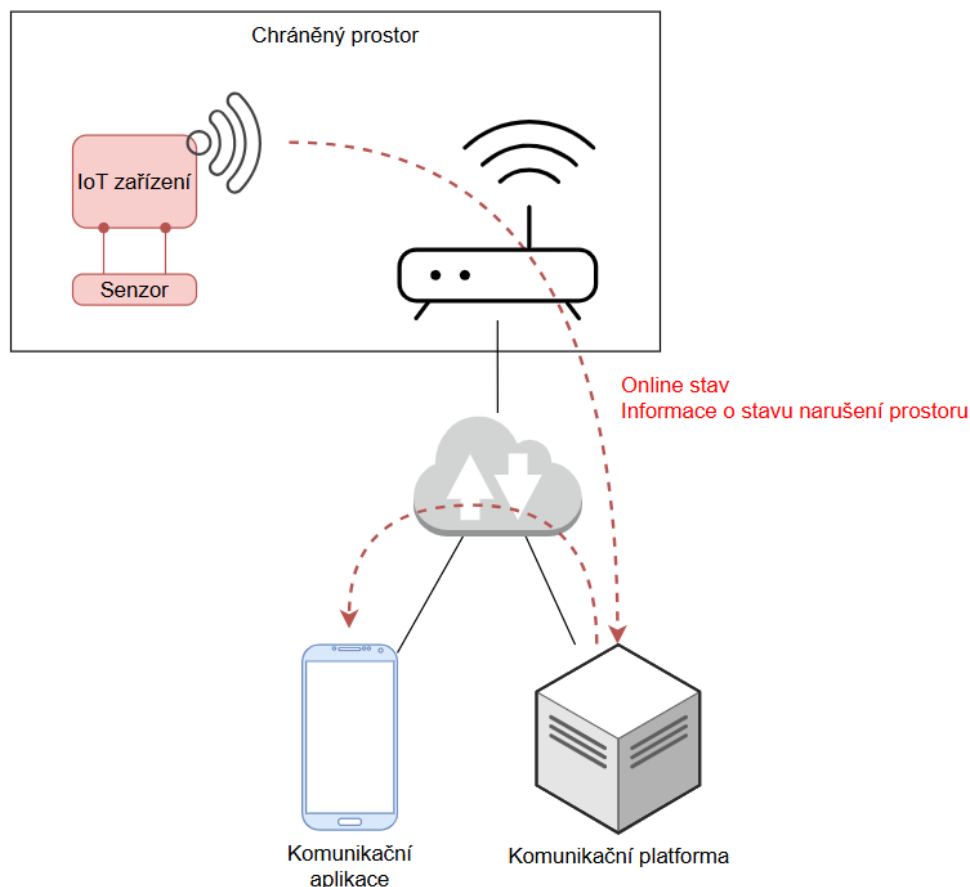


Obrázek 1 Vyhledávání pojmu IoT od roku 2010 po rok 2025

(zdroj: [3])

2.2 Jak chce práce využít IoT?

Blokové schéma zamýšleného využití IoT zařízení v rámci této práce je ukázáno na obrázku níže.



Obrázek 2 Blokové schéma IoT zařízení v kontextu této práce

(zdroj: vlastní)

IoT zařízení je připojeno na domácí WiFi síť a prostřednictvím routeru a poskytovatele internetu je připojeno ke vhodné platformě pro instantní komunikaci.

Stav chráněného prostoru je snímán jedním nebo více senzory připojenými k IoT zařízení. Pokud dojde ke změně, tak je tato změna zaslána jako zpráva určenému nebo určeným uživatelům. Uživatelé si tyto změny mohou prohlédnout např. prostřednictvím mobilní nebo webové aplikace.

Celý systém by měl umožnit uživatelům zaslat dotaz na IoT zařízení, a to by mělo umět odpovédět.

Více informací o požadavcích na systém bude uvedeno v samostatné kapitole dále v textu.

3 HW A SW PRO TVORBU IOT ZAŘÍZENÍ

Potřebný hardware pro tvorbu IoT zařízení bude obsahovat magnetický kontakt (jazyčkový spínač nebo Hallova sonda), který slouží k detekci stavu dveří (otevřeno/zavřeno). Tento senzor bude připojen ke vhodnému mikrokontroleru (např. ARM Cortex-M nebo ESP32), který bude sloužit jako hlavní řídicí jednotka celého systému.

Mikrokontroler je zodpovědný za zpracování signálu získaného z magnetického kontaktu. Když dojde ke změně stavu (např. při otevření dveří), mikrokontroler tento signál zaznamená, vyhodnotí a spustí předem definovanou akci.

Pro efektivní komunikaci se serverem je nezbytné, aby mikrokontroler měl k dispozici internetové připojení (Wi-Fi, Ethernet nebo mobilní síť). Nejčastěji se používají mikrokontrolery s integrovaným Wi-Fi modulem, jako je již zmíněný ESP32.

3.1 Požadavky na HW a SW pro tvorbu IoT zařízení

Na trhu existuje převeliké množství mikrokontrolerů, které se liší výkonem, spotřebou energie, paměťovou kapacitou, počtem vstupně-výstupních pinů a také cenou. Mikrokontrolery tvoří srdce většiny elektronických zařízení, výběr toho správného mikrokontroleru je tudíž velmi důležitý pro tento projekt, nevhodná volba může vést ke zbytečným nákladům, složité integraci nebo omezení funkcionality výsledného systému. Mikrokontroler by také měl mít integrovaný WiFi modul pro komunikaci s platformou a musí podporovat používaný software.

Velmi důležité kritérium pro výběr správného HW (hardware) je i možnost jednoduchého vývoje aplikací, ideálně s bohatou podporou existujících SW (software) knihoven, dostupného vývojového prostředí, dostupnost kvalitní dokumentace alespoň v anglickém jazyce. Špatná nebo neexistující dokumentace, či nedostatek existujících SW knihoven by mohl vést k tomu, že by nebylo možné splnit cíl této práce.

Požadavky na HW a SW lze shrnout do následujících bodů:

- Alespoň jeden digitální vstup
- Alespoň jeden digitální výstup, popř. zabudovaná LED (Light Emitting Diode)
- Podpora WiFi 2,4 nebo 5 GHz, režim klienta a režim AP (Access Point)
- USB (universal serial bus) port pro programování a ideálně i ladění

- Dostatečně výkonné CPU (Central Processing Unit) pro plynulý běh programu
- Dostatek Flash a RAM (Random Access Memory) paměti pro uložení a běh programu
- Podpora zabezpečené komunikace přes TLS 1.2 nebo 1.3 (Transport Layer Security, v1.2 dle RFC 5246, [4], v1.3 dle RFC 8446, [5])
- Knihovny pro komunikaci s https (Hypertext Transfer Protocol, RFC 9110 [6]) serverem
- Knihovny pro komunikaci přes WebSockets (RFC 6455 [7])
- Výhodou je podpora RTOS (real time operating system), např. FreeRTOS™ [8], popř. obdobného
- Programování v jazyce C nebo C++

3.2 Přehled dostupných HW a SW pro tvorbu IoT zařízení

Pro vyhledávání vhodného hardwaru byly využívány internetové obchody: laskakit.cz, pajenicko.cz, rpishop.cz.

Během zkoumání webů přicházely v úvahu vývojové desky krátce popsané v následujících podkapitolách.

Je však nutno upozornit na to, že se nabídka dostupných desek v čase mění – čtenář může snadno porovnat vývoj technologií i cen v době pozdějšího čtení. Pro čtenáře, kteří čtou tuto práci po několika letech po jejím napsání se mohou i usmát nad tím, jak slabé parametry zmiňované desky mají.

3.2.1 Seeed Studio XIAO ESP32C3

„Vývojová deska s MCU ESP32C3, integrovanou WiFi a Bluetooth 5, nabíjecím čipem pro akumulátor a nízkou spotřebou energie je vhodná pro rozmanité mobilní aplikace.“ [9].

Hlavní parametry:

Tabulka 1 Přehled parametrů Seed Studio XIAO ESP32C3

CPU	ESP32C3, 32 bitový 1 jádrový RISC-V (Reduced Instruction Set Computer), až 160MHz (Megahertz)
RAM	400KB (Kilobyte) SRAM (Static Random Access Memory)
FLASH	4MB (Megabyte)
Bezdrátové technologie	WiFi IEEE (Institute of Electrical and Electronics Engineers) 802.11b/g/n, stanice, SoftAP, SoftAP + stanice, Bluetooth 5 (vč. LE)
Nahrávání programu	USB
Ladění	USB
Podpora RTOS	FreeRTOS, ESP-IDF (Espressif IoT Development Framework)
Přibližná cena	145 Kč (korun českých)
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [10])

3.2.2 Sseed Studio XIAO ESP32S3

„Vývojová deska s MCU ESP32S3, integrovanou WiFi a Bluetooth 5, nabíjecím čipem pro akumulátor a nízkou spotřebou energie je vhodná pro rozmanité mobilní aplikace.“ [11].

Hlavní parametry:

Tabulka 2 přehled parametrů Sseed Studio XIAO ESP32S3

CPU	ESP32S3, 32 bitový 2 jádrový RISC-V, až 240MHz
RAM	8MB PSRAM
FLASH	8MB
Bezdrátové technologie	WiFi IEEE 802.11b/g/n, stanice, SoftAP, SoftAP + stanice, Bluetooth 5 (vč. LE)
Nahrávání programu	USB
Ladění	USB
Podpora RTOS	FreeRTOS, ESP-IDF
Přibližná cena	200 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [12])

3.2.3 Raspberry Pi Pico W

„Malý, rychlý a všestranný jednočipový počítač postavený na RP2040, prvním mikročipu od společnosti Raspberry Pi, s 2,4GHz bezdrátovým rozhraním.“ [13].

Hlavní parametry:

Tabulka 3 přehled parametrů Raspberry Pi Pico W

CPU	ARM Cortex M0+. 2 jádrový, 133 MHz
RAM	264KB SRAM
FLASH	2MB
Bezdrátové technologie	WiFi 802.11n, 2,4GHz, Bluetooth 5.2 (vč. LE)
Nahrávání programu	USB, Pico Debug Probe
Ladění	Pico Debug Probe
Podpora RTOS	FreeRTOS
Přibližná cena	170 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Menší než u ESP32

(zdroj: [14])

3.2.4 M5Stamp Pico Mate

„STAMP-PICO má ultrakompaktní design se dvěma 32bitovými mikroprocesory Xtensa® s nízkou spotřebou energie a frekvencí 240 MHz. Stamp Pico má velikost přibližně jako poštovní známka.“ [15].

Hlavní parametry:

Tabulka 4 přehled parametrů M5Stamp Pico Mate

CPU	ESP32-PICO-D4, 32 bitový, 2 jádrový, až 240MHz
RAM	520KB SRAM
FLASH	4MB
Bezdrátové technologie	WiFi 802.11 b/g/n, 2,4 GHz, Bluetooth 4.2
Nahrávání programu	Downloader
Ladění	JTAG nevyvedený
Podpora RTOS	FreeRTOS, ESP-IDF
Přibližná cena	160 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [16])

3.2.5 M5Stamp Pico Mate DIY Kit

„STAMP-PICO má ultrakompaktní design se dvěma 32bitovými mikroprocesory Xtensa® s nízkou spotřebou energie a frekvencí 240 MHz. Stamp Pico má velikost přibližně jako poštovní známka.“ [17].

Hlavní parametry:

Tabulka 5 přehled parametrů M5Stamp Pico Mate DIY Kit

CPU	ESP32-PICO-D4, 32 bitový, 2 jádrový, až 240MHz
RAM	520KB SRAM
FLASH	4MB
Bezdrátové technologie	WiFi 802.11 b/g/n, 2,4 GHz, Bluetooth 4.2
Nahrávání programu	Downloader (v ceně)
Ladění	JTAG nevyvedený
Podpora RTOS	FreeRTOS, ESP-IDF
Přibližná cena	530 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [19])

3.2.6 M5Stamp C3

„Stamp-C3 s MCU ESPRESSIF ESP32-C3 RISC-V s Wi-Fi a Bluetooth 5 (LE) konektivitou je vhodný pro zařízení edge IoT, jako jsou domácí spotřebiče a oblast průmyslové automatizace.“ [18].

Hlavní parametry:

Tabulka 6 přehled parametrů M5Stamp C3

CPU	ESP32C3, 32 bitový 1 jádrový RISC-V, až 160MHz
RAM	400KB SRAM
FLASH	4MB
Bezdrátové technologie	WiFi IEEE 802.11b/g/n, stanice, SoftAP, SoftAP + stanice, Bluetooth 5 (vč. LE)
Nahrávání programu	USB
Ladění	Nedostupné
Podpora RTOS	FreeRTOS, ESP-IDF
Přibližná cena	180 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [19])

3.2.7 M5Stamp C3U

„Stamp-C3U s MCU ESPRESSIF ESP32-C3 RISC-V s Wi-Fi 5 (LE) konektivitou je vhodný pro edge IoT zařízení, jako jsou domácí spotřebiče a oblast průmyslové automatizace.“ [20].

Hlavní parametry:

Tabulka 7 přehled parametrů M5Stamp C3U

CPU	ESP32C3, 32 bitový 1 jádrový RISC-V, až 160MHz
RAM	400KB SRAM
FLASH	4MB
Bezdrátové technologie	WiFi IEEE 802.11b/g/n, stanice, SoftAP, SoftAP + stanice, Bluetooth 5 (vč. LE)
Nahrávání programu	USB
Ladění	USB
Podpora RTOS	FreeRTOS, ESP-IDF
Přibližná cena	175 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [21])

3.2.8 M5Stamp S3

„Vývojový modul využívající čip ESP32-S3FN8, 8 MB Flash paměti a vysoce výkonným dvoujádrovým procesorem.“ [22]

Hlavní parametry:

Tabulka 8 přehled parametrů M5Stamp S3

CPU	ESP32-S3FN8, 32 bitový, 2 jádrový RISC-V, až 240MHz
RAM	8MB PSRAM
FLASH	8MB
Bezdrátové technologie	WiFi IEEE 802.11b/g/n, stanice, SoftAP, SoftAP + stanice, Bluetooth 5 (vč. LE)
Nahrávání programu	USB
Ladění	USB
Podpora RTOS	FreeRTOS, ESP-IDF
Přibližná cena	290 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [23])

3.2.9 ESP32-C6-DEVKITM-1

„The ESP32-C6-DevKitM-1 is a basic development board based on the ESP32-C6-MINI-1, a general purpose module with 4MB SPI flash. This board integrates complete Wi-Fi, Bluetooth LE, Zigbee and Thread features.“ [24].

Hlavní parametry:

Tabulka 9 přehled parametrů ESP32-C6-DEVKITM-1

CPU	ESP32-C6, HP 32 bitový 1 jádrový RISC-V, až 160MHz, LP 32 bitový 1 jádrový RISC-V, 20MHz
RAM	512KB
FLASH	4MB
Bezdrátové technologie	WiFi 6, 2,4 GHz IEEE 802.11a/x, Bluetooth 5.3, Zigbee, Threat (802.15.4)
Nahrávání programu	USB
Ladění	USB
Podpora RTOS	FreeRTOS, ESP-IDF
Přibližná cena	500 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [25])

3.2.10 Arduino Nano 33 IoT

„The Nano 33 BLE communication chipset can be both a Bluetooth client and host. Something quite unique in the world of microcontroller platforms. To see how easy it is to create a Bluetooth control panel or peripheral, explore the examples in the ArduinoBLE library.“ [26].

Hlavní parametry:

Tabulka 10 přehled parametrů Arduino Nano 33 IoT

CPU	SAMD21 Arm® Cortex®-M0, 32 bitový, až 48 MHz Bezdrátový modul: 2 jádrový 32 bitový, až 240 MHz
RAM	32KB Bezdrátový modul: 520 KB SRAM
FLASH	256KB Bezdrátový modul: 2MB
Bezdrátové technologie	WiFi IEEE 802.11b/g/n, 2,4GHz, Bluetooth 4.2 (vč. LE)
Nahrávání programu	USB
Ladění	Test pads
Podpora RTOS	Arduino, naportovaný FreeRTOS
Přibližná cena	730 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [27])

3.2.11 Wemos S2 Mini Wifi ESP32-S2

„Inovativní a kompaktní Wi-Fi modul, který je založen na čipu ESP32-S2FN4R2. Jako nová verze populárního Wemos D1 Mini nabízí S2 dvojnásobný počet I/O pinů ve stejném provedení. Tento modul je kompatibilní s většinou štítů (shields) pro Wemos D1 Mini, což usnadňuje jeho použití v existujících projektech. Díky podpoře programování v Arduino, MicroPythonu nebo ESP-IDF je WEMOS S2 Mini skvělou volbou pro všechny, kdo vyvíjejí inteligentní projekty a aplikace IoT.“[28].

Hlavní parametry:

Tabulka 11 přehled parametrů Wemos S2 Mini Wifi ESP32-S2

CPU	ESP32-S2, 32 bitů, 1 jádrový, až 240MHz
RAM	2MB PSRAM
FLASH	4MB
Bezdrátové technologie	WiFi IEEE 802.11b/g/n, 2,4GHz
Nahrávání programu	USB
Ladění	pravděpodobně USB
Podpora RTOS	FreeRTOS, ESP-IDF
Přibližná cena	120 Kč
Dokumentace	Velmi dobrá
Dostupnost knihoven	Velmi dobrá

(zdroj: [29])

3.3 Vybraný HW a SW pro tvorbu IoT zařízení

Z výše uvedených parametrů byla vybrána jako nejvhodnější deska Seeed Studio XIAO ESP32C3. Důvody výběru:

- kompaktní velikost
- schopnost nahrávat i ladit programy přes USB
- dobrá cena
- dostatek RAM i Flash paměti
- externí anténa (součást balení) zajišťující větší dosah
- dostupná dokumentace
- dostupné množství SW knihoven a příkladů, ať už od výrobce, či komunitních vývojářů

Možné nedostatky v porovnání s jinými alternativami:

- Nemá LED - lze snadno vyřešit připojením externí LED diody
- Potenciální bezpečnostní problémy plynoucí z toho, že se jedná o čínskou technologii.

Druhému bodu jsou věnovány různé diskuse a materiály. Např. vláda USA, resp. FCC úřad publikuje seznam zakázaných technologií a výrobců, které se mohou pro USA představovat bezpečnostní hrozbou [30]. Firma Espressif na tomto seznamu aktuálně není. Tomuto tématu se věnuje i diskuse na platformě Reddit, kde údajný zaměstnanec mluví o tom, že neví, že by espressif měla zadní dveře (back door) v žádném softwaru ani hardwaru. Také říká: „Co můžu říct s jistotou je, že espressif zaměstnává do firem mimo Čínu velký počet lidí, kteří z Číny nejsou. Máme pobočky například v Česku, Singapuru a Brazílii ve které neslouží jen jako obchody, ale také se tam provádí dost práce s hardwarem/softwarem. Takže se do něj mohou podívat i lidé, kteří nijak nespolupracují s Čínou.“ [31].

Větší studie je pak dostupná na webu Think Tanku ICIT, který se v USA věnuje bezpečnosti kritické infrastruktury. Studie mluví o ESP8266 a ESP32 jako o „perfektní zbrani“ kvůli masové distribuci, nízké ceně a vybavením mikropočítačů. K článku se vyjádřil Espressif. Vysvětluje např. nízkou cenu jako výsledek inovace, zpochybňuje závěry ICIT jako technicky nerealistické. [32]

4 PLATFORMY PRO INSTANTNÍ KOMUNIKACI A MOŽNOST ODESÍLÁNÍ ZPRÁV Z IOT ZAŘÍZENÍ

V této kapitole se práce věnuje řešerši vybraných platform pro instantní komunikaci. Cílem řešerše je vybrat platformu, která nejlépe definované funkční i nefunkční požadavky.

4.1 Funkční a nefunkční požadavky na komunikační platformu

- Použití zdarma
- Možnost založení identity IoT zařízení a propojení na vhodný kanál / topic
- Schopnost detekce, že je IoT zařazení online
- Možnost odesílání zpráv od uživatele k botovi (např. dotaz na stav)
- Možnost odesílání zpráv od bota k uživateli
- Dostupnost mobilní aplikace (popř. i PC aplikace / webového rozhraní)

4.2 Přehled platform a jejich možností integrace s IoT zařízením

4.2.1 DISCORD

„Discord je aplikace pro hlasovou, video a textovou komunikaci, kterou používají desítky milionů lidí, aby mohli komunikovat a trávit čas se svými kamarády a přáteli.

Lidé Discord používají denně k diskusím o různých tématech, od uměleckých projektů a rodinné výlety až po domácí úkoly a podporu duševního zdraví. Je to prostor pro komunity všech velikostí, nejčastěji jej ale využívají malé a aktivní skupiny lidí, kteří spolu pravidelně komunikují.“ [33]

Hlavní výhodou práce s Discordem je jeho výrazná podpora pro vývojáře. Platforma nabízí spolehlivé nástroje a přehledné rozhraní, které umožňují snadno integrovat vlastní funkce a automatizace přímo do serverů. Každý uživatel má možnost rychle a velmi jednoduše vytvořit svého vlastního bota prostřednictvím Discord Developer Portálu, což je specializované rozhraní určené pro vývoj a správu aplikací na Discordu.

Tento portál nabízí nejen možnost registrace a konfigurace botů, ale také rozsáhlou a přehlednou dokumentaci, která obsahuje podrobné návody, příklady kódu a popisy jednotlivých API (application programming interface) funkcí. Díky tomu mohou i amatérští vývojáři lehce porozumět základům tvorby botů a začít implementovat různé funkce – od

jednoduchých příkazů pro moderování serveru až po složité automatizační systémy nebo interaktivní aplikace.

Discord navíc podporuje širokou škálu programovacích jazyků (např. Python, JavaScript, C++) a oblíbených knihoven, jako je discord.py nebo discord.js, které vývojářům usnadňují práci a zkracují dobu potřebnou k vývoji nových funkcí. [34]

Další výhodou je aktivní komunita vývojářů na platformě Discord, kde si mohou začátečníci i pokročilí programátoři vyměňovat zkušenosti, získávat rady nebo spolupracovat na projektech.

Díky těmto nástrojům a podpoře je Discord jednou z nejprístupnějších platforem pro vytváření vlastních aplikací, botů a rozšíření, což umožňuje uživatelům přizpůsobit své servery přesně podle svých potřeb.

4.2.2 TELEGRAM

Telegram je cloudově založená komunikační aplikace vlastněna Ruskými podnikateli Pavlem a Nikolaiem Durovem. Uživatelé telegramu si můžou posílat textové zprávy, mít hovory, posílat si navzájem soubory, být ve skupinách až o 200 000 uživatelů a odebírat vysílací kanály. Službu lze používat přes mobilní aplikace, počítačové aplikace ale i skrz internetové prohlížeče, stejně jako Discord. Nejvýraznější výhodou Telegramu je jeho tajná funkce, která uživatelům umožňuje mít zabezpečené textové i hlasové konverzace skrz end-to-end enkrypci. [35]

Telegram má, stejně jako Discord, velmi dobrou podporu pro tvorbu botů pomocí telegramem vytvořeného bota s názvem BotFather, který vás provede celým procesem tvorby a správy botů. Pomocí jednoduchých textových příkazů umožňuje uživatelům vytvořit, nakonfigurovat a spravovat boty přímo v aplikaci Telegram, aniž by museli mít hluboké technické znalosti.

Telegram také poskytuje komplexní dokumentaci API, která je jasně strukturovaná a snadno pochopitelná. Vývojářům poskytuje přístup k podrobným informacím o metodách, parametrech a příkladech použití. Vývoj botů značně usnadňuje fakt, že Telegram podporuje více programovacích jazyků (např. Python, JavaScript, PHP(Hypertext preprocessor)) a má dostupné knihovny jako jsou python-telegram-bot nebo Telethon.[36]

Silná komunita vývojářů a fóra, kde mohou programátoři sdílet své zkušenosti, řešit problémy a nacházet inspiraci pro nové funkce a aplikace umožňuje začátečníkům jednodušší proces vyvíjení vlastních aplikací a botů.

4.2.3 SLACK

Slack je cloudová aplikace pro zasílání zpráv, která umožňuje pracovníkům organizace propojit se a spolupracovat. Uživatelé aplikace Slack mohou komukoli uvnitř i vně organizace posílat zprávy a využívat vyhrazená místa, tzv. kanály, k organizaci svých projektů, asynchronní práci, přístupu k informacím a sdílení aktualizací.

Slack se používá hlavně pro komunikaci na pracovišti a spolupráci na různých úkolech a projektech. Desktopová aplikace obsahuje postranní panel, který uživatelům umožňuje přístup ke kanálům, ke kterým se připojili, a ke všem konverzacím, kterých se účastní, otevírat přímé zprávy (DM), zobrazovat aktivity a všechny uložené položky a připomenutí. Uživatelé a správci si mohou postranní panel přizpůsobit úpravou svých předvoleb v aplikaci Slack.[37]

4.2.4 WhatsApp

WhatsApp je bezplatná multiplatformní služba pro zasílání zpráv. Umožňuje uživatelům chytrých telefonů iPhone a Android a počítačů Mac a Windows volat a vyměňovat si textové, fotografické, zvukové a video zprávy s ostatními uživateli po celém světě zdarma bez ohledu na zařízení příjemce. Na rozdíl od aplikací iMessage od společnosti Apple a Zprávy od společnosti Google, které vyžadují mobilní síť a službu krátkých zpráv (SMS), využívá WhatsApp ke komunikaci napříč platformami připojení Wi-Fi.

WhatsApp využívá Wi-Fi, což je cenově výhodné, a proto je oblíbený u uživatelů, kteří nemají datové tarify s neomezenými hovory a textovými zprávami. Její multiplatformní funkce je také oblíbená u lidí, kteří mají rodinu v zahraničí, cestují do zahraničí nebo žijí mimo Spojené státy.[38]

4.3 Výběr platformy pro integraci IoT zařízení

Při výběru vhodné platformy pro integraci IoT zařízení bylo důležité zohlednit několik klíčových faktorů, jako jsou dostupnost API, podpora pro vývojáře, flexibilita integrace, uživatelská základna a osobní preference při používání dané platformy.

Hlavními kandidáty se staly komunikační platformy Discord a Telegram. Obě tyto aplikace vynikají svou otevřeností, dobře zpracovanou dokumentací pro vývojáře a silnou komunitní podporou, která usnadňuje řešení případných problémů během vývoje bota.

Dalšími kandidáty byly také Slack a WhatsApp, avšak tyto aplikace byly rychle vyřazeny, Slack je sice robustní nástroj pro týmovou spolupráci, ale jeho API je složitější pro implementaci IoT zařízení a vyžaduje podrobnější znalosti platformy. WhatsApp omezuje možnosti vývoje botů a jejich nasazení přísnými pravidly, což ztěžuje flexibilní integraci. Navíc je primárně určen pro osobní komunikaci spíše než pro automatizaci a integraci IoT systémů

Osobní používání aplikací Slack a WhatsApp hrály také roli při výběru.

4.3.1 Srovnání vlastností, důvod pro výběr konkrétní platformy (Discord)

Tabulka 12 Srovnání vlastností komunikačních platform

Vlastnosti	Discord	Telegram	Slack	Whatsapp
Dostupnosti API	☑ Snadno dostupné	☑ Snadno dostupné	⚠ Komplexní API	⚠ Omezené API
Komunitní podpora	☑ Aktivní komunita	☑ Aktivní komunita	☑ Aktivní komunita	⚠ Omezená podpora
Náročnost implementace	☑ Nízká	☑ Nízká	✗ Vysoká	✗ Vysoká
Flexibilita bota	☑ Vysoká	☑ Vysoká	⚠ Střední	⚠ Střední
Reálná použitelnost pro IoT projekt	☑ Vysoká	☑ Vysoká	⚠ Střední	✗ Nízká

Explicitní online status	☒ Ano	☐ Omezený	☐ Omezený	☒ Ne
Dynamická změna statusu	☒ Ano	☒ Ne	☐ Omezený	☒ Ne
Vlastní textový status	☒ Ano	☒ Ne	☐ Omezený	☒ Ne
Viditelnost stavu pro uživatele	☒ Výborná	☐ Omezená	☐ Dostačující	☒ Nedostupná

(Zdroj: [34][36][39][40])

Po důkladném zvážení všech faktorů byla jako finální platforma vybrána aplikace Discord. Hlavní důvody pro tohle rozhodnutí byly:

- splňuje všechny funkční a nefunkční požadavky
- Jednoduchost používání – Discord poskytuje snadno pochopitelné API a mnoho komunitních knihoven, které dělají vývoj mnohem jednodušší.
- Flexibilita a rozšiřitelnost – Bot na Discordu může reagovat na širokou škálu událostí (např. textové příkazy, změny stavu uživatelů) a je snadno rozšiřitelný.
- Komunitní podpora – Aktivní vývojářská komunita a dostupné návody usnadňují řešení různých technických problémů.
- Osobní využití a preference – Discord je již dobře známá platforma s intuitivním uživatelským rozhraním a pravidelným využitím v rámci osobních i komunitních aktivit.

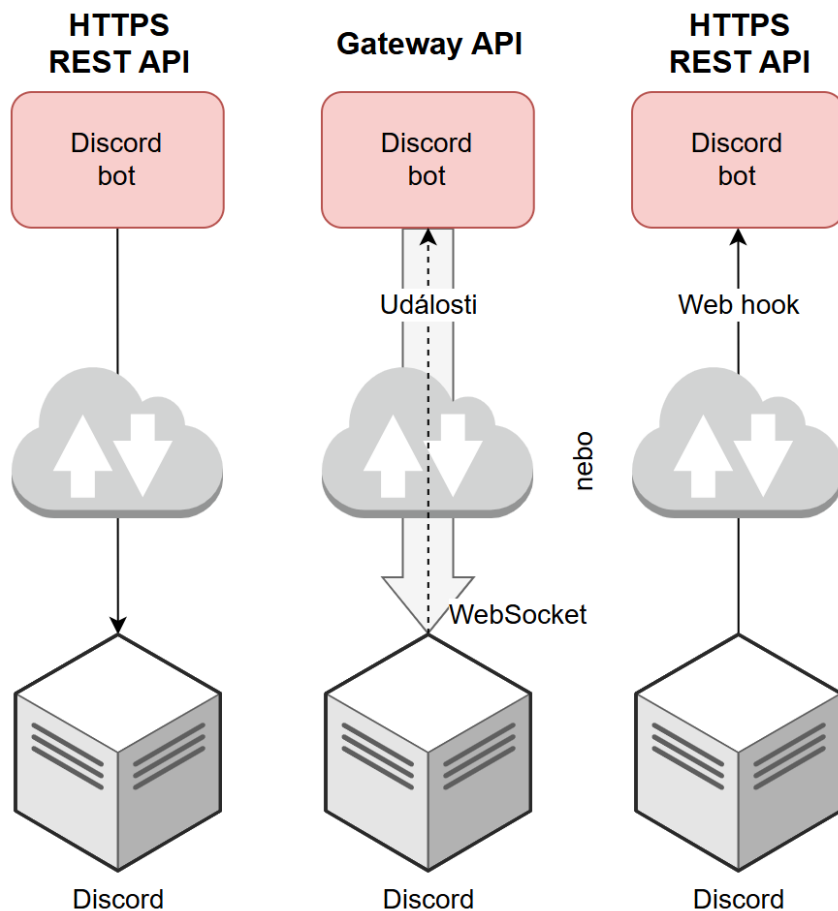
5 POPIS INTEGRACE NA DISCORD

Discord má velmi dobrou vývojářskou dokumentaci i vývojářský portál. Vývojářská dokumentace je dostupná na stránkách Discord Developer Portálu. Tato kapitola popíše důležité části dokumentace, které by měl mít vývojář Discord botů na paměti. V žádném případě však neuvádí všechny potřebné informace – vývojáři Discord botů by si měli vždy zkontrolovat aktuální dokumentaci.

5.1 Způsoby komunikace s Discord platformou

Discord aplikace a specificky Discord bot, jehož vývoj je předmětem této práce, má několik možností, jak komunikovat s Discord platformou[41]

- HTTP API – dovoluje interakce s Discord zdroji, jako jsou kanály, servery, uživatelé a zprávy
- Gateway API – umožňuje přijímat události v reálném čase prostřednictvím WebSocket spojení. Discord má možnost díky existenci spojení a události “Heartbeat”[42] detekovat online stav bota.
- Odchozí HTTP web hooky – jedná se o vzájemně vylučující se alternativu [43] Gateway API umožňující přijímání omezené sady událostí. Ač se jedná o technologicky (vývojářsky) snazší variantu přijímání událostí z Discord platformy, má dvě zásadní překážky bránící použití v tomto projektu:
- Discord provolává web hook, který musí být dostupný z veřejného internetu - to je pro mnoho domácností / objektů prakticky nedostupná možnost díky tomu, že prakticky všechna zařízení jsou “schované” za několika úrovněmi NAT.
- Web hooky neumožňují sledovat online stav bota - Discord tímto kanálem neprovolává “Heartbeat” události.



Obrázek 3 Možnosti komunikace Discord bota s Discord platformou

(zdroj: vlastní)

5.2 Autentizace a autorizace volání Discord bota

Aby platforma Discord rozpoznala identitu bota a byla schopna autorizovat volané služby (t.j. ověřit, že daný bot může provádět dané operace), je nutné, aby bot posílal v každém volání autorizační token.

Discord běžně pro autentizaci a autorizaci volání využívá OAuth2 protokol (RFC 6749 [44]), který za běžných okolností vyžaduje interaktivní aktivitu uživatele. Pro boty bez UI to však není možné provést, proto je možné pro boty vytvořit token na vývojářském portálu Discordu. Toto bude ukázáno v praktické části práce.[34]

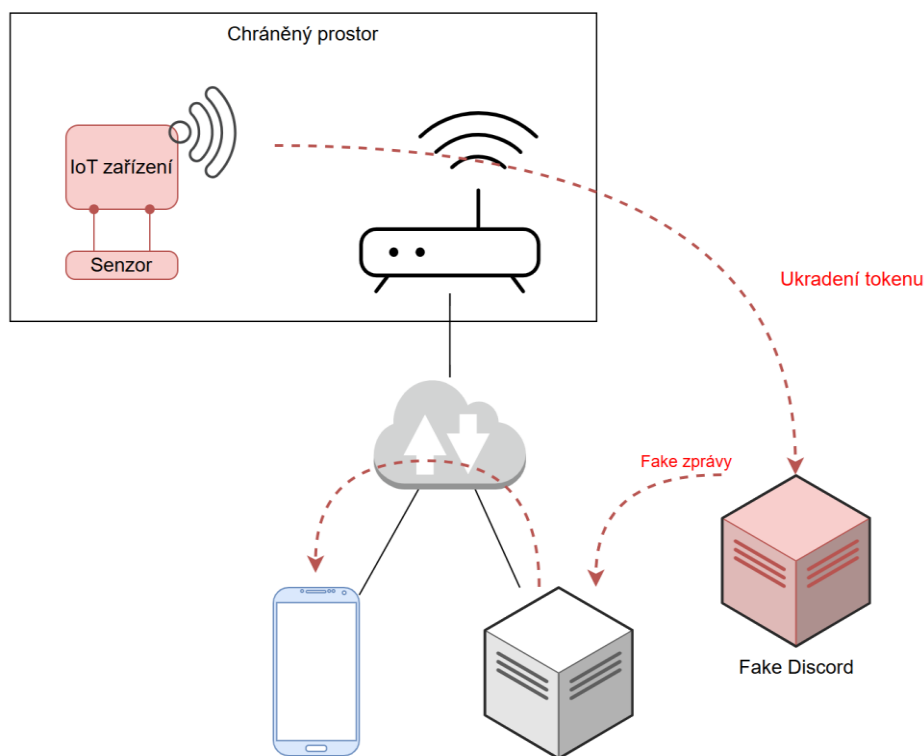
5.3 Problematika TLS a ověřování certifikátů Discordu

Discord má na svých serverech povolené protokoly TLS v1.2 a TLS v1.3.[45]

TLS (Transport Layer Security) je kryptografický protokol navržený k zajištění bezpečné komunikace přes počítačové sítě. Je nástupcem protokolu SSL (Secure Sockets Layer) a je široce používán k ochraně dat při přenosu mezi klientem (například webovým prohlížečem) a serverem. TLS je základem bezpečného internetu a jeho nejčastější implementace zahrnuje zabezpečení protokolů jako HTTPS, FTPS, SMTP, IMAP a dalších.[46]

Protože bot provolává Discord služby spolu s autorizačním tokenem, tak je důležité zajistit to, že autorizační token nezíská neoprávněná osoba (útočník). Pokud by útočník zmanipuloval DNS záznamy DNS serveru, přes který bot získává IP adresu Discordu, mohl by tento bot komunikovat na server útočníka a útočníkovi předat i svůj autorizační token.

Více o útocích tohoto typu lze dohledat pod pojmem “dns pharming” [47].



Obrázek 4 DNS spoofing útok

(zdroj: vlastní)

Bot tedy musí umět správně ověřit podpisy certifikátu Discord serveru a podpisy dalších vydávajících autorit až do bodu, kterému důvěřuje. Díky tomu je pak vůči takovým

útokům imunní. Současně ale musí mít možnost aktualizovat certifikát, který je bodem důvěry pro případ, kdy dojde k jeho výměně (popř. i výměně certifikační autority, kterou Discord využívá).

Tato práce bude předpokládat, že se výměna provede překompilováním kódu bota s novou konfigurací.

5.4 Knihovna pro komunikaci s Discord platformou

Pro účely tohoto projektu budou využity tyto mechanismy komunikace s Discord:

- HTTP API
- Gateway API

Sám Discord na svém dokumentačním portálu doporučuje to, aby vývojáři aplikací, pokud možno využili již existující knihovny pro komunikaci s Discordem.

Proto byl proveden i průzkum existujících knihoven / projektů, které by umožnily snadnou integraci na Discord, což značně sníží náročnost vlastní integrace a vytvoří prostor pro realizaci potřebných funkcí.

Jako nejvhodnější řešení se jeví použití projektu `esp-discord`, vytvořen uživatelem `abobija`. Projekt má MIT Licenci, která umožňuje volné použití i redistribuci pod podmínkou, že je licence a poznámka k autorským právům projektu redistribuovaná s výsledným dílem.

Součástí projektu je příklad `echo`, který umožňuje sledování zpráv na serveru.

Existuje i samostatný repozitář s více příklady využití knihovny. [48][49][50]

II PRAKTICKÁ ČÁST

6 FUNKČNÍ POŽADAVKY NA HW A SW BOTA

V minimalistickém pojetí bota lze definovat následující požadavky, které by měl jeho HW a SW zajistit. Požadavky jsou shrnuty v tabulce, přičemž je v tabulce uvedeno, jestli je nutné požadavek brát v úvahu při návrhu HW a SW.

Tabulka 13 Funkční požadavky na HW a SW bota

ID	Popis	HW	SW
P1	Možnost předat do bota informace pro připojení k WiFi	Ne	Ano
P2	Bot by měl indikovat stav připojení k WiFi - např. specifickým patternem blikání LED	Ano	Ano
P3	Bot umí snímat stav otevření dveří	Ano	Ano
P4	Bot by se měl umět přihlásit do Discordu	Ne	Ano
P5	Bot by měl udržovat online stav na Discordu	Ne	Ano
P6	Po připojení k WiFi a Discordu bot pošle aktuální informaci o stavu senzoru	Ne	Ano
P7	Bot kontinuálně snímá stav senzoru otevřených dveří a změny stavu odesílá na Discord	Ne	Ano
P8	Bot umí reagovat na zmínky o něm na Discordu a reagovat na ně odesláním aktuálního stavu snímače otevřených dveří	Ne	Ano

7 NÁVRH HW SCHÉMATU ZAŘÍZENÍ

Co se týče HW návrhu zařízení, je nutné primárně zohlednit:

- Požadavky na P2 a P3
- Případná HW specifika vybrané základní desky IoT[51]

Požadavek P2 je naplněn tím, že se k HW připojí LED dioda na vhodný výstupní pin. Pro zajištění ochrany před přetížením výstupního pinu je vhodné do série s LED diodou zapojit odpor – vzhledem k tomu, že HW pracuje na 3,3V, byla zvolena hodnota 150 Ω .

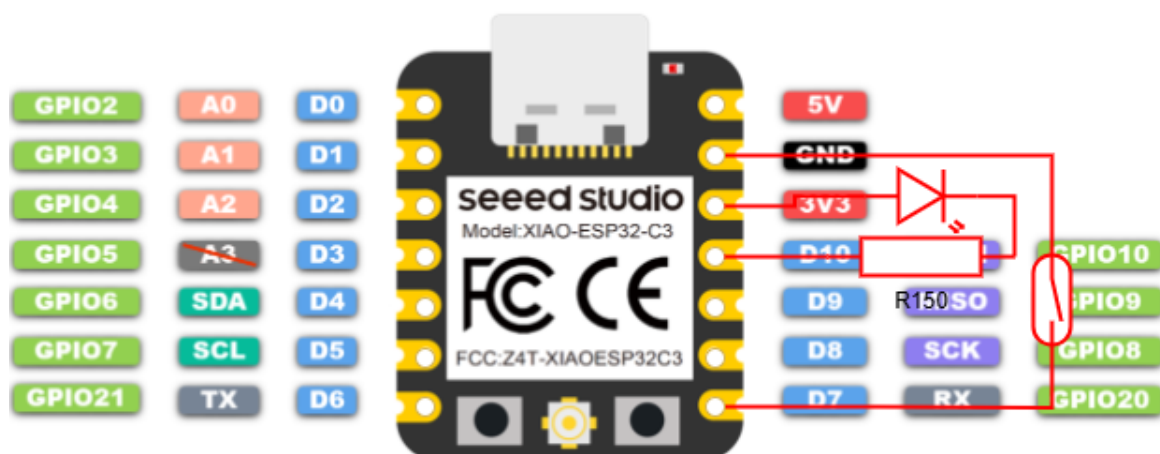
Požadavek P3 je naplněn připojením vhodného snímače otevření dveří. Nejjednodušší, a přitom velmi účinným se jeví použití magnetického jazýčkového relé v kombinaci s magnetem, který se umístí do dveří. Výhodou tohoto přístupu je to, že je možné stav dveří sledovat asynchronně s využitím přerušení reagujícího na změnu úrovně.

Při výběru vstupních / výstupních pinů je vhodné znát i omezení, která mohou dané piny mít. U vybraného procesoru je potřeba specificky dbát na to, že:

- některé piny mají specifický význam při bootu procesoru (GPIO2 (General Purpose Input Output), GPIO8 a GPIO9) - nebylo by příliš vhodné, aby se při náhodném resetu (např. výpadek napájení) dostalo zařízení do nevhodného stavu. [51]
- některé piny mají po resetu procesoru chování, které potenciálně může zařízení vystavit rizikům – byť tato explicitně nejsou v dokumentaci popsána a navíc by se jim dalo bránit. [51]

Pro zajištění minimalistického designu a s přihlédnutím na výše uvedené byly vybrány piny:

- GPIO10 – výstup ovládající LED diodu, přičemž stav “svítí” je při logické 0.
- GPIO20 – vstup snímající stav magnetického jazýčkového relé. Při inicializaci GPIO20 bude potřeba nastavit vnitřní pull-up, který v neaktivním stavu relé zajistí, že je vstup čten jako logická “1”. Zavřené dveře aktivují relé, to vstup připojí na zem (GND) a vstup bude tedy čten jako logická “0”.



Obrázek 5 Schéma zapojení HW

(upraveno z: [52])

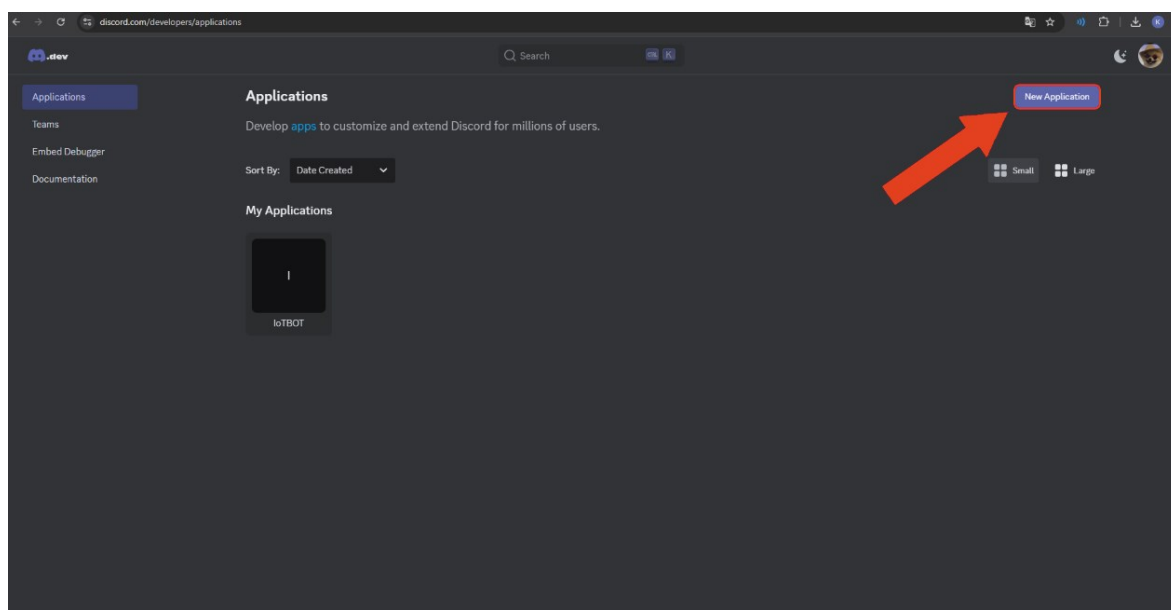
Na základě teoretické části a požadovaných funkcí na zařízení bylo navrženo toto HW schéma zařízení.

8 VYTVÁŘENÍ BOTA PRO DISCORD

Pro účely této práce je nutno vytvořit komunikační kanál mezi zařízením a uživatelem, který bude schopen v reálném čase doručovat upozornění o změně stavu sledovaných dveří.

8.1 Založení nové aplikace v Discord Developer Portálu

Prvním krokem je přihlášení do portálu pro vývojáře na stránkách Discord Developer Portálu. Zde je nutné vytvořit novou aplikaci kliknutím na tlačítko „New Application“. Po zadání názvu aplikace (např. IoTBOT) se vytvoří základní kostra bota.

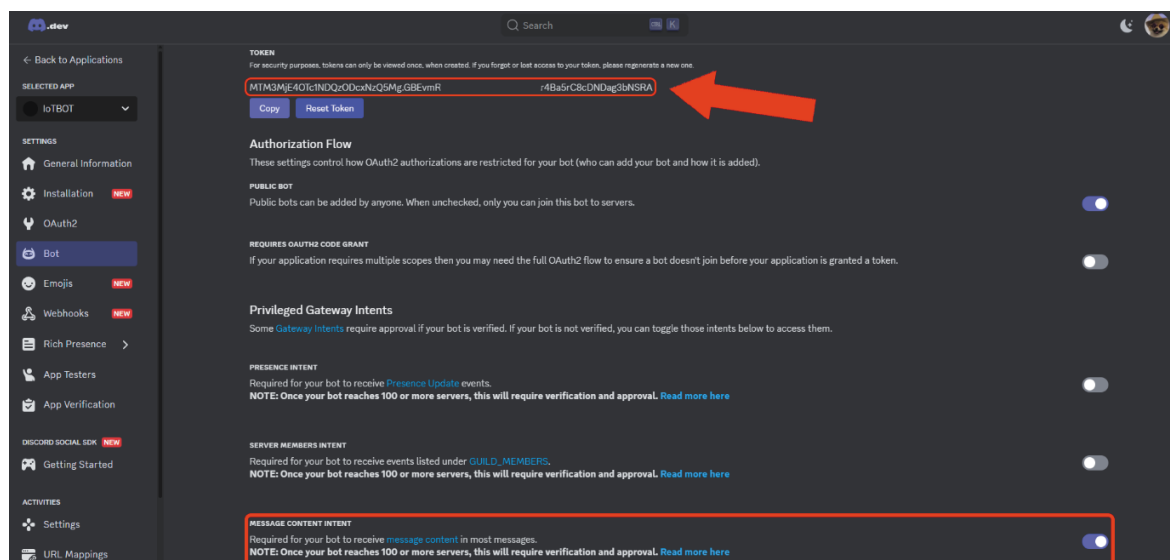


Obrázek 6 Vytvoření aplikace v Discord Dev Portálu

(upraveno z: [53])

8.2 Konfigurace Discord Bota

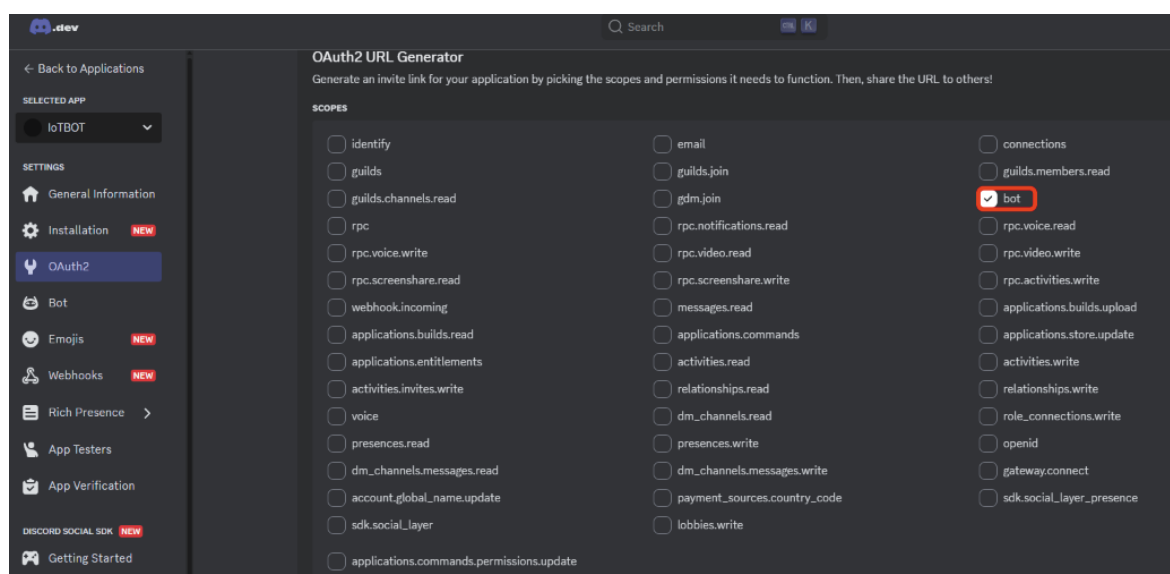
Po vytvoření aplikace je nutné přejít do sekce „Bot“, zde se nachází token bota, který slouží jako unikátní přístupový klíč, pomocí kterého se bot připojuje k Discordu. Tento token je nutno zkopírovat a bezpečně uložit, jelikož slouží k autentizaci programu. Kdyby došlo k jeho zveřejnění, mohlo by dojít k zneužití bota. V případě ztráty nebo zveřejnění tokenu, je možno jej na téhle stránce resetovat a vygenerovat nový. Další důležitou kolonkou je „MESSAGE CONTENT INTENT“, díky kterému bot bude moci sledovat obsah zpráv.



Obrázek 7 Token bota a povolení čtení obsahu zpráv

(upraveno z: [53])

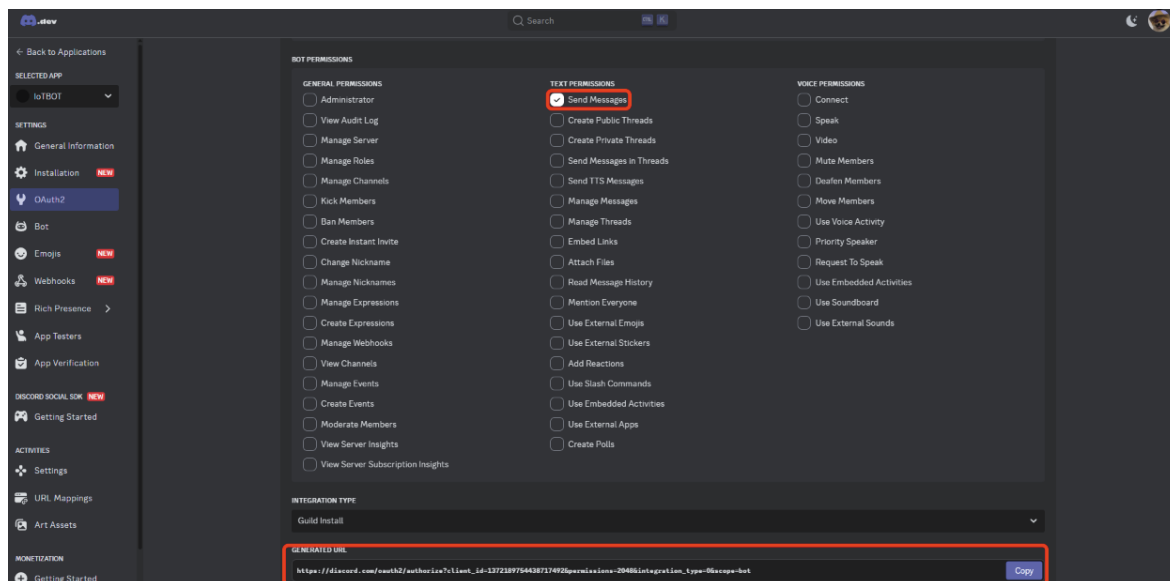
V záložce OAuth2 je potřeba zaškrtnout, co chceme, aby aplikace dělala, pro účely projektu je třeba zaškrtnout pouze políčko „bot“.



Obrázek 8 Přiřazení rozsahu bota

(upraveno z: [53])

Ve spodní části záložky musíme uvést, jaké úkony bude mít bot povolení provádět, pro účely projektu je třeba povolení pouze čtení zpráv. Po uložení změn stačí vložit vygenerovaný odkaz do prohlížeče, aby se bot mohl připojit na vybraný server.



Obrázek 9 Udělené povolení a generování linku pro pozvání bota na server

(upraveno z: [53])

V této kapitole byly nastaveny základní parametry bota a získán token, který bude později potřeba pro komunikaci SW s Discordem.

9 NÁVRH SW BOTA

SW vybavení bota by mělo realizovat všechny požadavky funkční specifikace, tj. P1-P8.

9.1 Naplnění funkčních požadavků

9.1.1 Požadavek 1 – Možnost předat do bota informace pro připojení k WiFi s tím, že tyto informace zůstanou uloženy v paměti i po resetu zařízení, vč. Výpadku napájení.

Platforma ESP-IDF podporuje WiFi provisioning, který s relativně malým úsilím realizuje potřebnou funkčnost. Použije se pro to sada funkcí `wifi_prov_mgr`, ke které existuje příklad použití a který byl z větší části přepoužit v této práci.[54][55] Nicméně, kód příkladu je komplexnější, demonstruje různé způsoby provisioningu a pro praktické použití není příliš vhodný, slouží pouze jako vodítko k vyřešení tohoto problému. Odladěná sekvence volání naplňujíc požadavek P1 je shrnuta ve zbytku této kapitoly.

Veškeré funkčnosti pro WiFi provisioning jsou uloženy v souboru `wifi_provisioning.c`.

Hlavní funkce pro spuštění provisioningu a připojení na WiFi je

```
esp_err_t wifi_provision(void)
```

Postupně provádí:

1. Inicializaci flash paměti pro uložení přihlašovacích údajů na vybranou WiFi, funkce

```
nvs_flash_init();
```

2. Inicializaci TCP/IP protokolu, funkce

```
esp_netif_init();
```

3. Inicializaci smyčky událostí, funkce

```
esp_event_loop_create_default();
```

4. Inicializaci obsluhy událostí pro WiFi provisioning, řízení sezení vrstvy protocomm (zajišťuje vlastní provisioning), sledování událostí získání IP adresy

```
esp_event_handler_register(WIFI_PROV_EVENT, ESP_EVENT_ANY_ID,  
    &event_handler, NULL);
```

```
esp_event_handler_register(PROTOCOMM_SECURITY_SESSION_EVENT,  
    ESP_EVENT_ANY_ID, &event_handler, NULL);
```

```
esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP,  
    &event_handler, NULL);
```

5. Inicializace rozhraní pro WiFi stanici a WiFi AP

```
esp_netif_create_default_wifi_sta();  
esp_netif_create_default_wifi_ap();
```

6. Inicializace WiFi s použitím výchozího nastavení

```
esp_wifi_init(&cfg);
```

7. Inicializace WiFi provisioning managera – použití nejjednoduššího schéma

```
wifi_prov_scheme_softap  
wifi_prov_mgr_init(config);
```

8. Ověření stavu provosioningu

```
wifi_prov_mgr_is_provisioned(&provisioned);
```

Pokud je zjištěno, že zařízení nemá nastavení WiFi, spustí se provisioning sekvencí kroků:

Vytvoření názvu AP, vytváří se patternem DBOT_ a hex reprezentace posledních tří bajtů MAC adresy, funkce

```
et_device_service_name(service_name, sizeof(service_name));
```

1. Spuštění provisioningu, funkce

```
wifi_prov_mgr_start_provisioning(security, NULL, service_name,  
service_key);
```

2. Deinicializován wifi provisioning manager, funkce

```
wifi_prov_mgr_deinit();
```

Registrována obsluha skupiny událostí sledujících připojení k WiFi, funkce

```
esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID,  
&event_handler, NULL);
```

3. Aktivován režim stanice, funkce

```
wifi_init_sta();
```

Pak program čeká na dokončení procesu provisioningu a / nebo připojení k WiFi, funkce

```
xEventGroupWaitBits(wifi_event_group, WIFI_CONNECTED_EVENT, true, true, portMAX_DELAY);
```

Vlastní provisioning údajů o WiFi (SSID a heslo) jsou do bota přeneseny pomocí aplikace ESP SoftAP Provisioning.

V rámci provisioningu a připojování na WiFi je sledován průběh pomocí událostí, které jsou zpracovávány funkcí **event_handler**:

- **skupina událostí WIFI_PROV_EVENT**

WIFI_PROV_START – indikuje start provisioningu

WIFI_PROV_CRED_RECV – indikuje přijetí SSID a hesla pro připojení k WiFi. Program pro účely ladění vypíše přijaté údaje na debug port.

WIFI_PROV_CRED_FAIL – indikuje chybu autentizace na WiFi během provisioningu. Reaguje na ní tak, že resetuje stav provisioningu.

WIFI_PROV_CRED_SUCCESS – indikuje úspěšné dokončení provisioningu

WIFI_PROV_END – indikuje ukončení provisioning služby, což je současně okamžik, kdy je vhodné deinitializovat provisioning managera.

- **skupina událostí WIFI_EVENT**

WIFI_EVENT_STA_START – indikuje zahájení režimu stanice, což je okamžik, kdy je možné inicializovat připojení k WiFi

WIFI_EVENT_STA_DISCONNECTED – indikuje odpojení od WiFi. V tomto případě je opět vhodné pokusit se k WiFi připojit znovu

WIFI_EVENT_AP_STA_CONNECTED – indikuje, že se na bota, který je v režimu AP připojila stanice (např. mobilní telefon použitý pro provisioning WiFi)

WIFI_EVENT_AP_STADISCONNECTED – indikuje odpojení stanice, např. i po několika minutách neaktivity.

- **skupina událostí IP_EVENT**

IP_EVENT_STA_GOT_IP – indikuje získání IP adresy z routeru. Reaguje na událost tak, že nastaví event bit indikující tuto skutečnost. Provisioning funkce může na základě toho být ukončena. Bot má IP adresu a může pokračovat v další činnosti.

- **Skupina událostí PROTOCOL_SECURITY_SESSION_EVENT** – informační události podsystému protocomm. Program pouze vypisuje informace o dalších přijatých událostech.

9.1.2 Požadavek 2 – Bot by měl indikovat stav připojení k WiFi – např. specifickým patternem blikání led

Funkce indikace stavu připojení k WiFi je relativně lehká. Využívá kód **led_task**, který umožňuje:

- nastavit port pro LED a úroveň, ve které je LED aktivní – bot je zapojen tak, že LED svítí při výstupu nastaveném na 0.
- držet frontu požadavků na blikání LED
- blikat pomalu
- blikat rychle
- bliknout definovaný počet krát za sebou – použitelné pro indikaci konkrétní chybové situace
- obsluha blikání je naprogramována jako samostatný FreeRTOS task

Potenciál kódu není aktuálně v programu využitý, využívá se jen funkčnosti rychlého blikání (bot není po startu připojený k WiFi) a pomalého blikání (bot se k WiFi úspěšně připojil).

Tímto způsobem je naplněn požadavek P2.

9.1.3 Požadavek 3 – Bot umí snímat stav otevření dveří

Změny stavu dveří – otevřeno / zavřeno – jsou detekovány změnou úrovně vstupu, na který je připojeno jazýčkové relé.

Kód pro snímání stavu dveří je v souboru **discordbot.c** (a hlavičkovém souboru **discordbot.h**).

Funkce pro inicializaci

```
esp_err_t configure_relay(gpio_num_t gpio_num)
```

1. Nastaví vstupně / výstupní port (GPIO)

- a. jako vstup, se zapnutým odporem na plus (pull-up) - pokud není jazýčkové relé sepnuto, je snímán stav 1.
- b. Změna vstupní rovně generuje přerušení (GPIO_INTR_ANYEDGE) - snímání událostí je možné plně asynchronně

2. Registruje handler událostí – funkci relay_isr_handler

Díky tomuto mechanismu je možné plně asynchronně hlídat změny stavu snímače otevřených dveří a reagovat na ně.

Ukázka kódu nastavení portu jako vstupního a nastavení handleru přerušení při změně stavu:

```
esp_err_t configure_relay(gpio_num_t gpio_num)
{
    esp_err_t r;
    TaskHandle_t xTaskToNotify = xTaskGetCurrentTaskHandle();

    gpio_config_t io_conf = {
        .intr_type = GPIO_INTR_ANYEDGE,
        .mode = GPIO_MODE_INPUT,
        .pin_bit_mask = BIT64(gpio_num),
        .pull_down_en = 0,
        .pull_up_en = 1};

    //configure pin
    r = gpio_config(&io_conf);
    if (r) goto FNRET;

    // hook isr handler for specific gpio pin
    r = gpio_isr_handler_add(gpio_num, relay_isr_handler, (void *)xTaskToNotify);
    if (r) goto FNRET;

FNRET:
    ESP_LOGI("configure_relay", "Initialization return code=%d", r);
    return r;
}
```

Obsluha přerušení je jednoduchá, minimalistický kód pouze pošle notifikaci do monitorovacího tasku.

Fragment kódu s obsluhou přerušení:

```
static void IRAM_ATTR relay_isr_handler(void *arg)
{
    TaskHandle_t xTaskToNotify = (TaskHandle_t) arg;
    BaseType_t xHigherPriorityTaskWoken = pdFALSE;
    if (xTaskToNotify)
    {
        vTaskNotifyGiveFromISR(xTaskToNotify, &xHigherPriorityTaskWoken);
        portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
    }
}
```

Pro správnou funkčnost je nutné předem inicializovat službu obsluhy přerušení GPIO pomocí

```
gpio_install_isr_service(0);
```

Přijetí notifikace bude popsáno u požadavku P7.

9.1.4 Požadavek 4 – Bot by se měl umět přihlásit do Discordu

Obsluha Discordu je založena na komponentě **abobija/esp-discord** dostupná na GitHubu, která je pod volnou licencí MIT. Knihovna obsahuje velmi dobrý startovací bod – příklad echo. Tento příklad demonstruje:

- Inicializaci připojení k Discord API
- Přijetí zprávy od uživatele
- Automatickou odpověď s obsahem původní zprávy[49][55]

Kód pro inicializaci připojení k discordu je v souboru **discordbot.c**

```
discord_config_t cfg = {.intents = DISCORD_INTENT_GUILD_MESSAGES};

bot = discord_create(&cfg);
if (bot==NULL) goto FNRET;

r=discord_register_events(bot, DISCORD_EVENT_ANY, bot_event_handler,
NULL);
if (r) goto FNRET;

r=discord_login(bot);
if (r) goto FNRET;
```

Upozornění: Pro to, aby přihlášení proběhlo, je nutné, aby byl při sestavení SW bota nastaven token získaný z developerského portálu Discordu, jak bylo popsáno výše.

9.1.5 Požadavek 5 – Bot by měl udržovat online stav na Discordu

Platforma Discord vyžaduje, aby bot uměl indikovat, že je stále připojený. Podle dokumentace je vyžadováno, aby ve specifických časových intervalech bot odesílal tzv. heartbeat zprávu. [42]

S využitím komponenty abobija/esp-discord není potřeba tuto funkcionalitu řešit, neboť je její integrální součástí. To je jedna z velkých výhod použité komponenty.

9.1.6 Požadavek 6 – Po připojení k WiFi a Discordu bot pošle aktuální informaci o stavu senzoru

Z mnoha důvodů je vhodné tento požadavek naplnit. Zejména však z důvodu, kdy by vypadlo internetové připojení, popř. napájení. Uživatel bota tak získá aktuální stav snímače, jakmile se obnoví komunikace bota s platformou Discord.

Realizace tohoto požadavku je s použitím komponenty abobija/esp-discord a příkladu echo jednoduchá. Knihovna podporuje sledování událostí, na které může kód využívající komponentu vhodně reagovat. V tomto případě postačuje na událost `DISCORD_EVENT_CONNECTED` reagovat tím, že bot odešle zprávu.

```
case DISCORD_EVENT_CONNECTED:
{
    discord_session_t *session = (discord_session_t *)data->ptr;
    connected=1;
    ESP_LOGI(TAG, "Bot %s#%s connected", session->user->username,
session->user->discriminator);
    send_relay_state(NULL);
}
```

Zpráva však může být odeslána jen v případě, že bot ví, do jakého kanálu ji má odeslat. Nastavení kanálu je popsáno v kapitole zabývající se uvedením bota do provozu.

9.1.7 Požadavek 7 – Bot kontinuálně snímá stav senzoru otevřených dveří a změny stavu odesílá na Discord

Hlavní funkcí bota je sledování stavu relé a odesílání stavu snímače na Discord kanál. Požadavek P3 zajistí sledování změn stavu snímače, požadavek P4 přihlášení do Discordu a P5 pak udržuje spojení aktivní.

Na to vše navazuje funkčnost ve funkci `relay_monitoring_task`, která postupně:

1. Konfiguruje port, ke kterému je připojen snímač otevřených dveří

```
r = configure_relay(gpio_num);
```

2. Inicializuje stav snímače s negovanou hodnotou aktuálního stavu – to zajistí, že při spuštění této úlohy dojde k pokusu o odeslání stavu snímače (ovšem odeslání je možné jen v případě, že je bot již připojen k Discordu

```
relay_state = !gpio_get_level(gpio_num);
```

3. Spustí nekonečnou smyčku, která detekuje změny stavu čidla

```
while (2 + 3 * 4 == 14)
{
    if (relay_state != gpio_get_level(gpio_num))
```

4. A při změně stavu si zapamatuje nový stav, pokusí se odeslat zprávu a uspí se na specifický krátký časový interval, který eliminuje odesílání zpráv při prokmitech kontaktů jazýčkového relé

```
{
    relay_state = gpio_get_level(gpio_num);
    ESP_LOGI("relay_monitoring_task", "Relay state changed to %d!",
relay_state);
    //send new state to Discord
    relay_state_changed(relay_state);
    //ignore short state changes
    vTaskDelay(pdMS_TO_TICKS(500));
}
```

5. Čeká na další změnu stavu (navazuje na notifikace odesílané v P3)

```
//wait for relay state change
    ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
```

9.1.8 Požadavek 8 – Bot umí reagovat na zmínky o něm na Discordu a reagovat na ně odesláním aktuálního stavu snímače otevřených dveří

Z praktického hlediska je výhodné, když má uživatel systému možnost ověřit, že bot správně pracuje, tj. je připojen na Discord a je schopen reagovat. Jako efektivní mechanismus se jeví odeslat botovi zprávu se zmínkou o něm. Tímto mechanismem si může uživatel i nastavit kanál, do kterého má bot odesílat zprávy.

S použitím komponenty abobija/esp-discord a příkladu echo je opět realizace jednoduchá. Stačí reagovat na událost `DISCORD_EVENT_MESSAGE_RECEIVED` a z ní:

1. Zjistit kanál, na který byla poslána

```
{
    discord_message_t *msg = (discord_message_t *)data->ptr;

    ESP_LOGI(TAG,
        "New message (dm=%s, author=%s#%s, bot=%s, channel=%s,
        guild=%s, content=%s)",
        !msg->guild_id ? "true" : "false",
        msg->author->username,
        msg->author->discriminator,
        msg->author->bot ? "true" : "false",
        msg->channel_id,
        msg->guild_id ? msg->guild_id : "NULL",
        msg->content);

    if (msg->content && msg->content[0])
    {
        send_relay_state(msg->channel_id);
    }
}
```

2. Odeslat stav relé

```
// we know channel_id
    ESP_LOGI(TAG, "Going to send message to
    channel_id=%s", cached_channel_id);
```

```
char *content = estr_cat("Door is ", gpio_get_level(RELAY_GPIO) ?
"OPEN " DISCORD_EMOJI_X : "closed " DISCORD_EMOJI_WHITE_CHECK_MARK);

discord_message_t msg = {.content = content, .channel_id =
cached_channel_id};

discord_message_t *sent_msg = NULL;
esp_err_t err = discord_message_send(bot, &msg, &sent_msg);
free(content);

if (err == ESP_OK)
{
    ESP_LOGI(TAG, "Relay status message successfully sent");

    if (sent_msg)
    { // null check because message can be sent but not returned
        ESP_LOGI(TAG, "Relay status message got ID #s", sent_msg->id?sent_msg->id:"UNKNOWN");
        discord_message_free(sent_msg);
    }
}
else
{
    ESP_LOGE(TAG, "Fail to send Relay status message");
}
```

9.2 Struktura programu

V předchozí kapitole bylo uvedeno, jakým způsobem jsou naplněny požadavky dané na SW bota a jsou tam uvedeny i ukázky kódu. Kompletní kód je přílohou této práce. Pro snazší orientaci v kódu, je v této kapitole uveden přehled souborů se zdrojovými kódy a popis funkcí v nich naprogramovaných.

9.2.1 main.c

Jedná se o hlavní program, který obsahuje funkce:

- **void app_main(void)** - hlavní uživatelská funkce, která je spuštěna systémem ESP-IDF pro inicializaci aplikace. Funkce provede inicializaci dalších tasků – zobrazování stavu přes LED, provisioning / připojení k WiFi a konečně vlastní funkci bota.[56]

9.2.2 led task.h a led task.c

Funkce pro obsluhu indikace stavu přes LED. Převzato původně z jiného projektu a upraveno pro potřeby použití v tomto projektu.

Publikované funkce:

- **void* led_init(const unsigned int gpio, const int on_state)** - inicializuje obsluhu LED na konkrétním GPIO portu s možností definice stavu, ve kterém LED svítí. Pro obsluhu LED spustí samostatný task.
- **void led_push_action(void *handle, t_led_action led_action, int repeats)** - vloží akci do fronty pro obsluhu LED. Aktuální podoba kódu umožňuje vložit do fronty až 8 akcí. Pokud se má akce donekonečna opakovat, tak se jí dává počet opakování -1.
- **void led_deinit(void *handle)** - odinicializuje obsluhu LED, což má za následek ukončení obslužné úlohy v jejím dalším cyklu.

Interní funkce:

- **static void led_task(void *handle)** - hlavní task obsluhující LED. Ve smyčce testuje, zda je požadovaná akce a tu pak vykonává. Pokud je fronta odinicializovaná, tak se ukončí.
- **static void led_init_action(t_led_state *led, uint64_t t)** - najde ve frontě novou akci a nastaví ji jako aktivní.
- **static void led_update_action(t_led_state *led, uint64_t t)** - hlavní výkonný kód, který obsluhuje stav LED akce a ve výsledku i stav pinu, na který je LED připojena.

9.2.3 wifiprovisioning.h a wifiprovisioning.c

Funkce pro provisioning WiFi a připojení k WiFi.

Publikované funkce:

esp_err_t wifi_provision(void) - funkce, která inicializuje provisioning WiFi a připojení k WiFi

Interní funkce:

- **static void event_handler(void *arg, esp_event_base_t event_base, event_id, void *event_data)** - funkce obsluhující události související s provisioningem a připojením k WiFi. Většina událostí je obsloužena jen za účelem diagnostiky stavu připojování.
- **static void wifi_init_sta(void)** - inicializuje režim WiFi stanice.
- **static void get_device_service_name(char *service_name, size_t max)** - vytvoří název SSID pro režim připojování přes AP.

9.2.4 discordbot.h a discordbot.c

Funkce pro provisioning sledování stavu čidla a napojení na platformu Discord.

Publikované funkce:

esp_err_t dib_start() - inicializuje task pro sledování stavu čidla a napojení na platformu Discord.

Interní funkce:

- **static void send_relay_state(char *channel_id)** - pokusí se odeslat zprávu o stavu čidla na Discord. Neodesílá v případech, kdy zjistí, že není dostupné připojené sezení s Discord nebo není známo ID Dascord kanálu, kam by se zpráva měla poslat.
- **static void bot_event_handler(void *handler_arg, esp_event_base_t base, int32_t event_id, void *event_data)** - zpracovává eventy komponenty pro obsluhu připojení na Discord.
- **static void relay_state_changed(int)** - funkce pro obsluhu změny stavu relé. Je volaná z monitorovací smyčky (není volaná z obsluhy přerušení).ukázala
- **static void IRAM_ATTR relay_isr_handler(void *arg)** - funkce, která obsluhuje logiku přerušení vyvolaného při změně stavu čidla. Kód by měl být co nejkratší, aby neblokoval vykonávání jiných operací.
- **esp_err_t configure_relay(gpio_num_t gpio_num)** - funkce, která konfiguruje GPIO pro sledování změn stavu čidla. Je napsána tak, že musí být

volána z tasku, který provádí monitoring (z důvodu předání notifikací do správného taksu).

- **static void relay_monitoring_task(void *arg)** - hlavní nekonečná smyčka, která reaguje na notifikace o změnách stavu čidla.

10 UVEDENÍ BOTA DO PROVOZU A OVĚŘENÍ FUNKČNOSTI

K funkci bota je zapotřebí rozšíření ESP IDF od firmy espressif, konkrétně verzi 5.3.2 do Visual Studio Code.

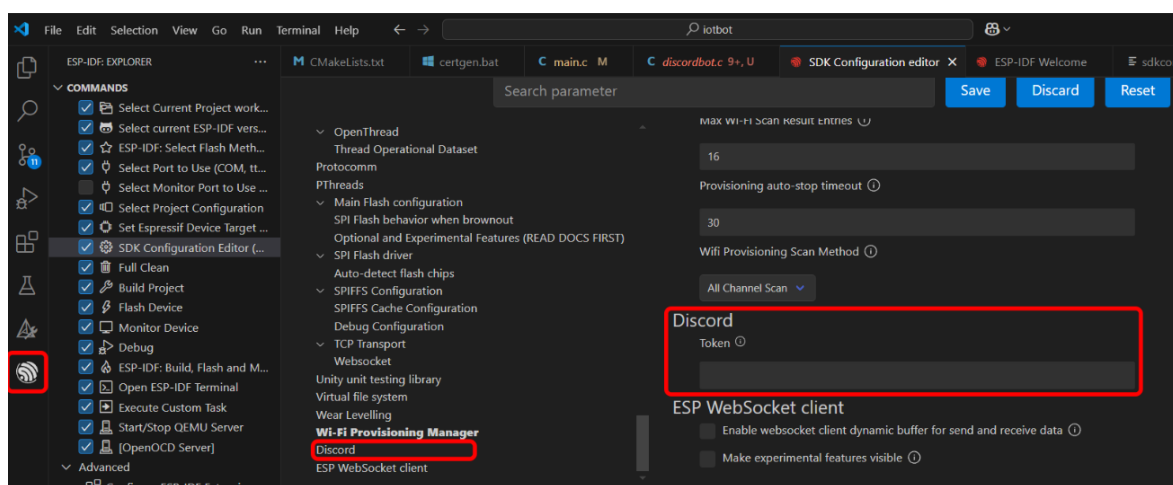
Pro volné stažení programu z internetu je třeba do terminálu postupně vložit tyhle řádky

```
cd [složka, do které chceme stáhnout program]
git clone https://github.com/Gnomecek/iotbot.git
cd iotbot
code .
```

Ke správnému chodu programu je třeba ve Visual Studio Code soubor „cert“ vložit do složky abobja__esp-discord ve složce managed_components.

- **Nastavení API klíče**

Dříve získaný token bota na Discord Developer Portalu vložíme do softwaru pomocí rozšíření ESP-IDF a knihoven pro discord a ESP32. [50]



Obrázek 10 Nastavení tokenu ve Visual Studio Code

(zdroj: vlastní)

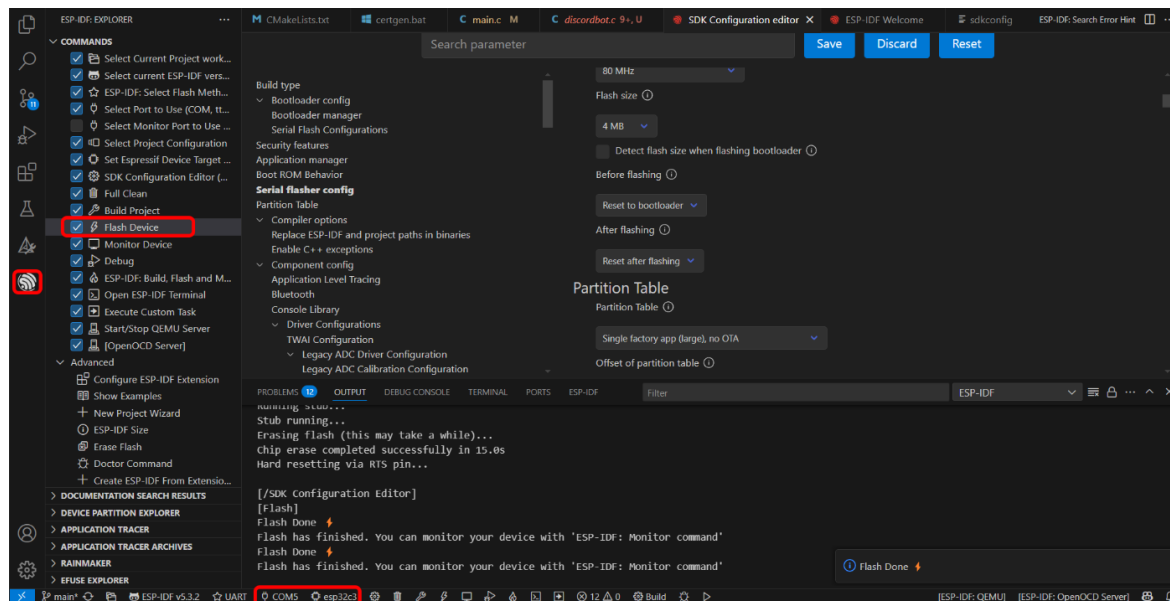
- **Nastavení kanálu**

Pro nastavení požadovaného kanálu, do kterého chceme zasílat zprávy přes bota, potřebujeme získat ID kanálu. Pro zobrazení ID kanálu je třeba přímo v aplikaci Discord povolit vývojářský režim v záložce “Pokročilé”. Následně stačí na požadovaný textový kanál kliknout pravým tlačítkem myši → Kopírovat ID kanálu.

Zkopírované ID kanálu je třeba vložit stejným, výše uvedeným, způsobem do kolonky „Discord Bot“

• Naprogramování Flash

V levé spodní části programu Visual Studio Code je třeba vybrat správný port, ve kterém je zařízení připojeno a také model zařízení (v tomto případě COM5 a esp32c3).



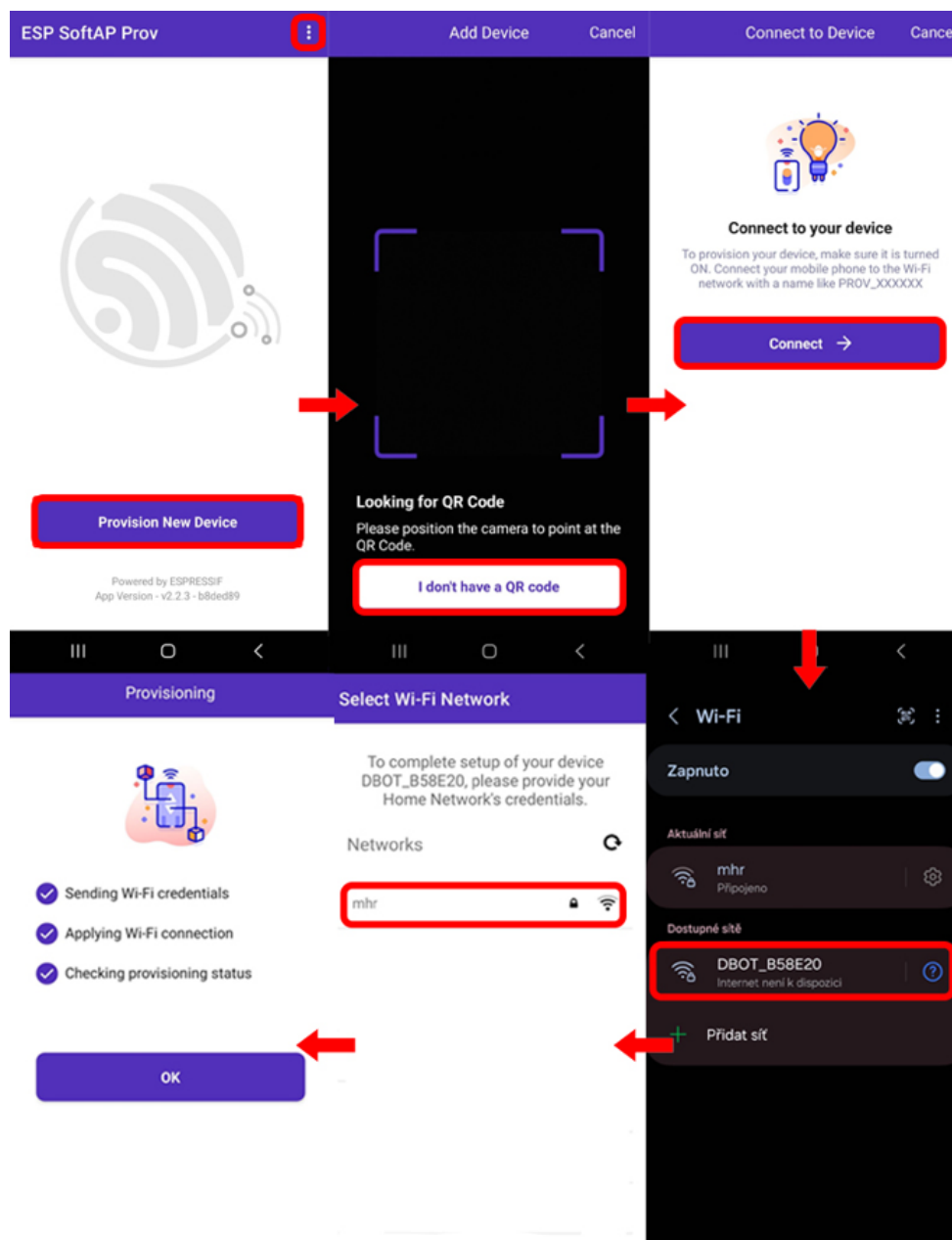
Obrázek 11 Flashování zařízení

(zdroj: vlastní)

• Nastavení wifi pomocí mobilu

Pro připojení zařízení WiFi je třeba jej buďto nastavit v konfiguraci SDK viz. kapitola 5.1.

Pro připojení zařízení k WiFi je zapotřebí volně přístupné aplikace na mobily – ESP SoftAP Provisioning od firmy espressif (Dostupné jak pro Android tak pro iOS). V aplikaci je třeba jako první v nastavení vypnout šifrovanou komunikaci. Následně → Provision New Device → I don't have a QR code → Connect → Připojit se k síti “DBOT_XXXXXX” → vybrat WiFi, na kterou chceme, aby se bot připojil.

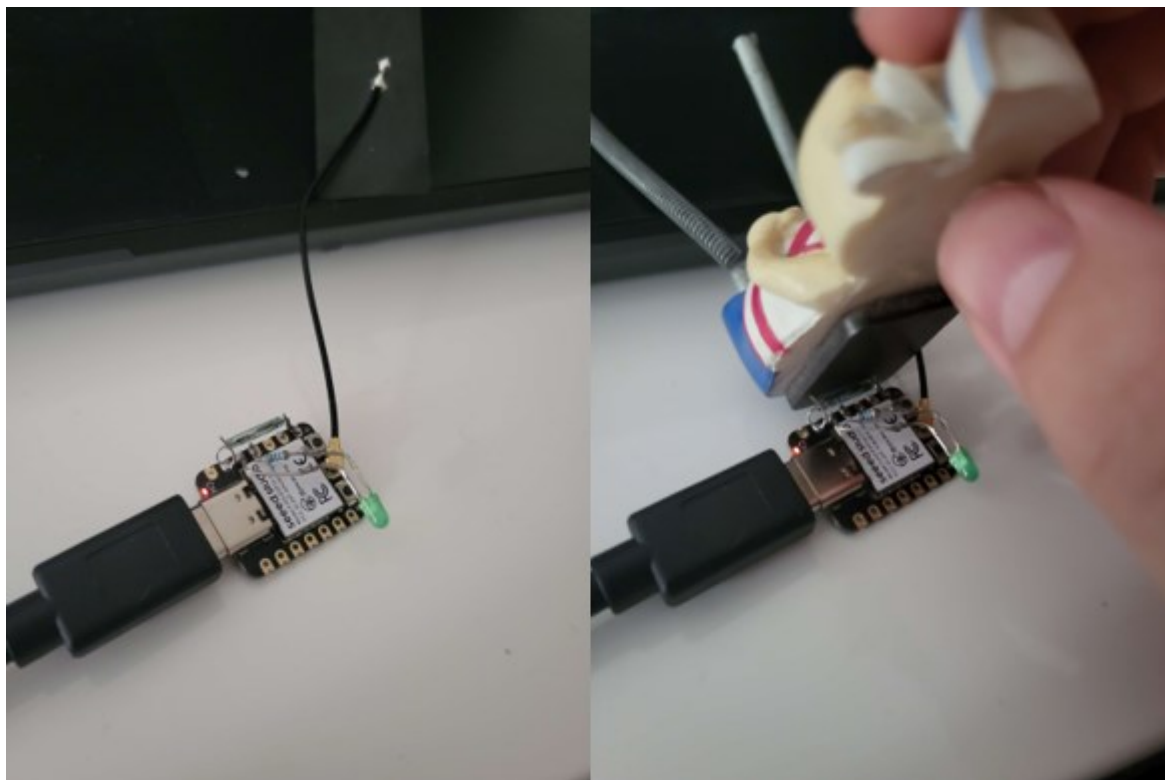


Obrázek 12 Postup připojení zařízení k WiFi přes aplikaci

(zdroj: vlastní)

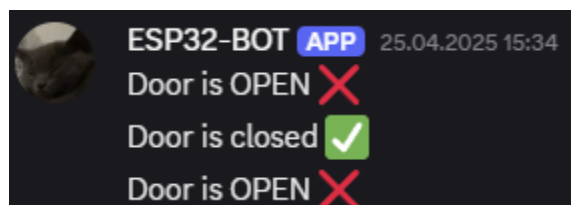
- **Otestování funkčnosti**

Po připojení bota k WiFi je třeba jen otestovat. K otestování stačí přiložit magnet k elektromagnetickému relé. Po přiložení/oddálení magnetu by měl bot poslat zprávu do vybraného kanálu na Discordu.



Obrázek 13 Zařízení bez přiloženého/přiloženým magnetem

(zdroj: vlastní)



Obrázek 14 Odesílané zprávy botem

(zdroj: vlastní)

Výše uvedené obrázky ověřují funkčnost správného zapojení HW, připojení zařízení k WiFi a fungující kód zařízení.

11 MOŽNOSTI DALŠÍHO ROZVOJE

Tento projekt má ještě spoustu možností, jak jej vylepšit. Jednou z prvních věcí je vyčištění kódu, který má redundantní obsah. Avšak jednou z nejdůležitějších věcí na vylepšení je implementace lepšího způsobu pro nastavení discord tokenu a výchozího discord kanálu, do kterého má bot odesílat zprávy – obě informace by bylo vhodné dostat do paměti bota buď prostřednictvím provisioningu WiFi nebo prostřednictvím web stránek, které by bot mohl publikovat po připojení k Wifi. Pokud uživatel chce změnit WiFi, na kterou je bot připojen, musí první smazat Flash paměť. Tento problém je také k zamyšlení při dalším rozvoji projektu. Velmi zajímavou částí rozvoje by pak mohlo být experimentování s takzvaným WiFi CSI sensing, kdy by teoreticky mohlo být možné sledovat i pohyb v místnosti na základě změn WiFi signálu. Toto by vyžadovalo další zařízení, nicméně by jeho použití mohlo být vyváženo i dalšími vlastnostmi sledování chráněného prostoru.[57]

ZÁVĚR

Tato bakalářská práce se zabývala průzkumem technologií IoT, dostupných hardwarových a softwarových platforem a návrhem chytrého IoT detektoru, který monitoruje neoprávněné vniknutí do chráněného prostoru prostřednictvím detekce otevření dveří a zasílá upozornění uživatelům prostřednictvím platformy Discord. Cílem bylo vytvořit systém, který je efektivní, spolehlivý a snadno integrovatelný do různých prostředí.

V bakalářské práci byla popsána základní architektura systémů detekce neoprávněného vniknutí, principy fungování senzorů a role internetu věcí (IoT) v oblasti zabezpečovacích technologií. Byly analyzovány dostupné hardwarové a softwarové platformy vhodné pro implementaci IoT zařízení, přičemž byla vybrána deska Sseed Studio XIAO ESP32C3 díky své kombinaci výkonu, cenové dostupnosti a dostupnosti dokumentace a vývojových nástrojů.

Za cenu cca 200Kč bylo postaveno zařízení, které je schopné detekovat otevírání / zavírání dveří a tyto události propagovat na komunikační platformu Discord. Pro provoz je dostatečné mít k dispozici napájení přes USB-C konektor, WiFi s přístupem k internetu a bezplatně vytvořený a provozovaný účet na platformě Discord. Uživatel systému tak může kdekoliv, kde má přístup k internetu sledovat, zda mu někdo nenarušuje jeho domov, chatu, či práci.

Výsledky této práce mohou dát základ novému směru dostupného hlídání či sledování chráněných prostorů. Autor věří, že práce může inspirovat i ostatní v budování podobných zařízení. Sám u této práce získal mnoho nových a cenných zkušeností, které bude možné využít v další soukromé tvorbě, při dalším studiu a taky v budoucím zaměstnání.

V projektu samotném je možné dále pokračovat a rozvíjet jej. Některé z možných směrů jsou popsány přímo v 11 kapitole práce.

SEZNAM POUŽITÉ LITERATURY

- [1] Mikeš, Jan. Profi elektrika.cz. [Online] 18. 2. 2016. [cit. 19. 03 2018] Dostupné z: <https://elektrika.cz/data/clanky/pdvnoo020611>. ISSN 1212-9933.
- [2] LUTKEVICH, Ben. *IoT basics and fundamentals: A guide for beginners*. Online. Computer Glossary, Computer Terms - Technology Definitions and Cheat Sheets from WhatIs.com - The Tech Dictionary and IT Encyclopedia. 2023. Dostupné z: <https://www.techtarget.com/whatis/feature/IoT-basics-A-guide-for-beginners>. [cit. 2025-01-05]..
- [3] Google Trends. *IoT: zájem v průběhu času*. [online]. Google, [cit. 2024-01-05]. Dostupné z: <https://trends.google.com/trends/explore?date=today%205-y&q=IoT&hl=cs>
- [4] *The Transport Layer Security (TLS) Protocol Version 1.2*. Online. IETF Datatracker. Srpen 2008. Dostupné z: <https://datatracker.ietf.org/doc/html/rfc5246>. [cit. 2025-05-27].
- [5] *The Transport Layer Security (TLS) Protocol Version 1.3*. Online. IETF Datatracker. Srpen 2018. Dostupné z: <https://datatracker.ietf.org/doc/html/rfc8446>. [cit. 2025-05-27].
- [6] *HTTP Semantics*. Online. IETF Datatracker. Červen 2022. Dostupné z: <https://datatracker.ietf.org/doc/html/rfc9110>. [cit. 2025-05-27].
- [7] *The WebSocket Protocol*. Online. The WebSocket Protocol. Prosinec 2011. Dostupné z: <https://datatracker.ietf.org/doc/html/rfc6455>. [cit. 2025-05-27].
- [8] FREERTOS. *FreeRTOS*. Online. 2003. Dostupné z: <https://www.freertos.org/>. [cit. 2025-05-27].
- [9] RPISHOP. *Seeed Studio XIAO ESP32C3*. Online. 2019. Dostupné z: <https://rpishop.cz/xiao/5607-seeed-studio-xiao-esp32c3.html>. [cit. 2025-05-27].
- [10] Seeed Studio. *XIAO ESP32C3 Getting Started*. [online]. Seeed Studio, [cit. 2024-01-05]. Dostupné z: https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/
- [11] RPISHOP. *Seeed Studio XIAO ESP32S3*. Online. 2019. Dostupné z: <https://rpishop.cz/498312/seeed-studio-xiao-esp32s3/>. [cit. 2025-05-27].

- [12] Seeed Studio. *XIAO ESP32S3 Getting Started*. [online]. Seeed Studio, [cit. 2024-01-05]. Dostupné z: https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/
- [13] RPISHOP. *Raspberry Pi Pico W*. Online. 2019. Dostupné z: <https://rpishop.cz/raspberry-pi-pico/5073-raspberry-pi-pico-w.html>. [cit. 2025-05-27].
- [14] Raspberry Pi Foundation. *Pico Series Documentation*. [online]. Raspberry Pi Foundation, [cit. 2024-01-05]. Dostupné z: <https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html#pico-1-family>
- [15] RPISHOP. *M5Stack M5Stamp Pico Mate s pinovými headery*. Online. 2019. Dostupné z: <https://rpishop.cz/m5stack/4229-m5stack-m5stamp-pico-mate-s-pinovymi-headery.html>. [cit. 2025-05-27].
- [16] M5Stack. *Stamp Pico*. [online]. M5Stack, [cit. 2024-01-05]. Dostupné z: https://docs.m5stack.com/en/core/stamp_pico
- [17] RPISHOP. *M5Stack M5Stamp Pico Mate DIY Kit*. Online. 2019. Dostupné z: <https://rpishop.cz/m5stack/5437-m5stack-m5stamp-pico-mate-diy-kit.html>. [cit. 2025-05-27].
- [18] RPISHOP. *M5Stack M5Stamp C3 Podložka s pinovými headery*. Online. 2019. Dostupné z: <https://rpishop.cz/m5stack/4391-m5stack-m5stamp-c3-podlozka-s-pinovymi-headery.html>. [cit. 2025-05-27].
- [19] M5Stack. *Stamp C3*. [online]. M5Stack, [cit. 2024-01-05]. Dostupné z: https://docs.m5stack.com/en/core/stamp_c3
- [20] RPISHOP. *M5Stack M5Stamp C3U Podložka s pinovými headery*. Online. 2019. Dostupné z: <https://rpishop.cz/m5stack/4692-m5stack-m5stamp-c3u-podlozka-s-pinovymi-headery.html>. [cit. 2025-05-27].
- [21] M5Stack. *Stamp C3U*. [online]. M5Stack, [cit. 2024-01-05]. Dostupné z: https://docs.m5stack.com/en/core/stamp_c3u
- [22] RPISHOP. *M5Stack M5Stamp Vývojový modul s WiFi komunikací*. Online. 2019. Dostupné z: <https://rpishop.cz/498325/m5stack-m5stamp-vyvojovy-modul-s-wifi-komunikaci/>. [cit. 2025-05-27].

- [23] M5Stack. *Stamp S3*. [online]. M5Stack, [cit. 2024-01-05]. Dostupné z: <https://docs.m5stack.com/en/core/StampS3>
- [24] LASKAKIT.CZ. *Espressif ESP32-C6-DEVKITM-1 WiFi 6, Bluetooth 5, Zigbee*. Online. 2016. Dostupné z: <https://www.laskakit.cz/en/espressif-esp32-c6-devkitm-1-wifi-6--bluetooth-5--zigbee/>. [cit. 2025-05-27].
- [25] Espressif Systems. *ESP32-C6-DevKitM-1 User Guide*. [online]. Espressif Systems, [cit. 2024-01-05]. Dostupné z: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32c6/esp32-c6-devkitm-1/user_guide.html
- [26] LASKAKIT.CZ. *Arduino Nano 33 IoT, original*. Online. 2016. Dostupné z: <https://www.laskakit.cz/arduino-nano-33-iot--original/>. [cit. 2025-05-27].
- [27] Arduino. *Nano 33 IoT*. [online]. Arduino, [cit. 2024-01-05]. Dostupné z: <https://docs.arduino.cc/hardware/nano-33-iot/>
- [28] PAJENICKO.CZ. *Wemos S2 Mini Wifi ESP32-S2*. Online. 2020. Dostupné z: <https://pajenicko.cz/komponenty/vyvojove-desky-a-mikroprocesory/wemos-s2-mini-wifi-esp32-s2>. [cit. 2025-05-27].
- [29] WEMOS. *S2 Mini*. [online]. WEMOS, [cit. 2024-01-05]. Dostupné z: https://www.wemos.cc/en/latest/s2/s2_mini.html
- [30] FEDERAL COMMUNICATIONS COMMISSION. *List of Equipment and Services Covered By Section 2 of The Secure Networks Act*. Online. 2019, 3. Zář 2024. Dostupné z: <https://www.fcc.gov/supplychain/coveredlist>. [cit. 2025-05-27].
- [31] *Espressif & Supply Chain Security*. Online. 2023. Dostupné z: https://www.reddit.com/r/esp32/comments/10ebhef/espressif_supply_chain_security/?rdt=47962. [cit. 2025-05-27].
- [32] *The Perfect Weapon, Hidden in Plain Sight*. Online. 2020. Dostupné z: <https://www.icitech.org/post/the-perfect-weapon-hidden-in-plain-sight-a-study-on-how-the-espressif-wi-fi-and-ble-chips-and-m>. [cit. 2025-05-27].
- [33] Discord. *What is Discord?* [online]. Discord, [cit. 2024-01-05]. Dostupné z: <https://discord.com/safety/360044149331-what-is-discord>
- [34] Discord. *Gateway Events Documentation*. [online]. Discord, [cit. 2024-01-05]. Dostupné z: <https://discord.com/developers/docs/events/gateway>

- [35] Encyclopedia Britannica. *Telegram*. [online]. Encyclopædia Britannica, [cit. 2024-01-05]. Dostupné z: <https://www.britannica.com/topic/Telegram-software>
- [36] Telegram. *Telegram APIs* [online] Telegram, [cit. 2024-01-05]. Dostupné z: <https://core.telegram.org/>
- [37] TechTarget. *Slack (software)*. [online]. TechTarget, [cit. 2024-01-05]. Dostupné z: <https://www.techtarget.com/searchcontentmanagement/definition/Slack-software>
- [38] TechTarget. *WhatsApp*. [online]. TechTarget, [cit. 2024-01-05]. Dostupné z: <https://www.techtarget.com/searchmobilecomputing/definition/WhatsApp>
- [39] Slack API. *Methods*. [online]. Slack, [cit. 2024-01-05]. Dostupné z: <https://api.slack.com/methods>
- [40] Meta for Developers. *WhatsApp Documentation*. [online]. Meta Platforms, [cit. 2024-01-05]. Dostupné z: <https://developers.facebook.com/docs/whatsapp>
- [41] Discord. *Overview of Apps*. [online]. Discord, [cit. 2024-01-05]. Dostupné z: <https://discord.com/developers/docs/quick-start/overview-of-apps#what-apis-can-apps-use>
- [42] Discord. *Heartbeat Event Documentation*. [online]. Discord, [cit. 2024-01-05]. Dostupné z: <https://discord.com/developers/docs/events/gateway-events#heartbeat>
- [43] Discord. *Preparing for Interactions*. [online]. Discord, [cit. 2024-01-05]. Dostupné z: <https://discord.com/developers/docs/interactions/overview#preparing-for-interactions>
- [44] Hardt, D. *The OAuth 2.0 Authorization Framework (RFC 6749)*. [online]. Internet Engineering Task Force (IETF), 2012 [cit. 2024-01-05]. Dostupné z: <https://datatracker.ietf.org/doc/html/rfc6749>
- [45] CDN77. *TLS Test Result for gateway.discord.gg*. [online]. CDN77, [cit. 2024-01-05]. Dostupné z: <https://www.cdn77.com/tls-test/result?domain=gateway.discord.gg>
- [46] Cloudflare. *What is TLS?* [online]. Cloudflare, [cit. 2024-01-05]. Dostupné z: <https://www.cloudflare.com/learning/ssl/what-is-tls/>
- [47] SSL.com. *Co je pharming?* [online]. SSL.com, [cit. 2024-01-05]. Dostupné z: <https://www.ssl.com/cs/blogy/co-je-pharming/>
- [48] abobija. *esp-discord Repository*. [online]. GitHub, [cit. 2024-01-05]. Dostupné z: <https://github.com/abobija/esp-discord>

- [49] abobija. *esp-discord License*. [online]. GitHub, [cit. 2024-01-05]. Dostupné z: <https://github.com/abobija/esp-discord/blob/main/LICENSE>
- [50] abobija. *esp-discord Examples*. [online]. GitHub, [cit. 2024-01-05]. Dostupné z: <https://github.com/abobija/esp-discord-examples>
- [51] Seeed Studio. *ESP32-C3 Datasheet*. [online]. 2023 [cit. 30. května 2025]. Dostupné z: https://files.seeedstudio.com/wiki/XIAO_WiFi/Resources/esp32-c3_datasheet.pdf
- [52] *Getting Started with Seeed Studio XIAO ESP32C3*. Online. Cca 2023, 14. srpen 2024. Dostupné z: https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/. [cit. 2025-05-27].
- [53] DISCORD. *Building your first Discord app*. Online. Cca 2015. Dostupné z: <https://discord.com/developers/docs/quick-start/getting-started>. [cit. 2025-05-27].
- [54] ESPRESSIF. *Wi-Fi Provisioning*. Online. ESP-IDF Programming Guide. 2015. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/v5.4.1/esp32c3/api-reference/provisioning/wifi_provisioning.html. [cit. 2025-05-30].
- [55] ESPRESSIF. *Wi-Fi Provisioning Manager Example*. Online. GitHub. 2015. Dostupné z: https://github.com/espressif/esp-idf/tree/v5.4.1/examples/provisioning/wifi_prov_mgr. [cit. 2025-05-30].
- [56] ESPRESSIF. *Application Startup Flow*. Online. ESP-IDF Programming Guide. 2015. Dostupné z: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32c3/api-guides/startup.html#app-main-task>. [cit. 2025-05-31].
- [57] *ESP-CSI*. Online. GitHub. 2021. Dostupné z: <https://github.com/espressif/esp-csi>. [cit. 2025-05-31].

SEZNAM OBRÁZKŮ

Obrázek 1 Vyhledávání pojmu IoT od roku 2010 po rok 2025	16
Obrázek 2 Blokové schéma IoT zařízení v kontextu této práce	17
Obrázek 3 Možnosti komunikace Discord bota s Discord platformou	38
Obrázek 4 DNS spoofing útok	39
Obrázek 5 Schéma zapojení HW	44
Obrázek 6 Vytvoření aplikace v Discord Dev Portálu	45
Obrázek 7 Token bota a povolení čtení obsahu zpráv	46
Obrázek 8 Přiřazení rozsahu bota	46
Obrázek 9 Udělené povolení a generování linku pro pozvání bota na server	47
Obrázek 10 Nastavení tokenu ve Visual Studio Code	61
Obrázek 11 Flashování zařízení	62
Obrázek 12 Postup připojení zařízení k WiFi přes aplikaci	63
Obrázek 13 Zařízení bez přiloženého/přiloženým magnetem	64
Obrázek 14 Odesílané zprávy botem	64

SEZNAM TABULEK

Tabulka 1 Přehled parametrů Seed Studio XIAO ESP32C3	20
Tabulka 2 přehled parametrů Sseed Studio XIAO ESP32S3	21
Tabulka 3 přehled parametrů Raspberry Pi Pico W	22
Tabulka 4 přehled parametrů M5Stamp Pico Mate	23
Tabulka 5 přehled parametrů M5Stamp Pico Mate DIY Kit.....	24
Tabulka 6 přehled parametrů M5Stamp C3.....	25
Tabulka 7 přehled parametrů M5Stamp C3U.....	26
Tabulka 8 přehled parametrů M5Stamp S3	27
Tabulka 9 přehled parametrů ESP32-C6-DEVKITM-1	28
Tabulka 10 přehled parametrů Arduino Nano 33 IoT	29
Tabulka 11 přehled parametrů Wemos S2 Mini Wifi ESP32-S2	30
Tabulka 12 Srovnání vlastností komunikačních platforem	35
Tabulka 13 Funkční požadavky na HW a SW bota.....	42

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AP	Access Point
API	Application Programming Interface
CPU	Central processing unit
ČSN	Česká technická norma
DIY	Do it yourself
DM	Direct Message
DNS	Domain Name System
ESP-IDF	Espressif IoT Developement Framework
FCC	Federal Communication Commussion
FTPS	File Transfer Protocol Secure
GND	Ground
GPIO	General Purpose Input Output
https	hypertext transfer protocol
HW	hardware
I/O	Input/Output
ICIT	Information and communication Technology
ID	identification
IEEE	Institute of Electrical and Electronics Engineers
IMAP	Internet Message Access Protocol
IoT	Internet of Things
IP	Internet Protocol
JTAG	Joint test action group
KB	Kilobyte
Kč	Korun českých
LED	Light emitting diode

MB	Megabyte
MIT	Massachusetts Institution of Technology
NAT	Network address translation
O2Auth	Open Authentication
PC	Personal Computer
PHP	PHP: Hypertext preprocessor, původně Personal homepage
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RTOS	Real time operation system
SMTP	Simple Mail Transfer Protocol
SRAM	Static Random Access Memory
SW	Software
TLS	Transport layer security
UI	User Interface
USA	United States of America
USB	Universal serial bus
WiFi	Wireless Fidelity
Ω	Ohm

SEZNAM PŘÍLOH

Příloha P I: iotbot

Příloha P II Ukázka bota

PŘÍLOHA P I: IOTBOT

Zdrojový kód je k práci přiložen v souboru s přílohami

PŘÍLOHA P II: UKÁZKA BOTA

Příloha obsahuje krátké video s ukázkou funkčního bota.